CONTENT-ADAPTIVE SUPERPIXEL SEGMENTATION VIA IMAGE TRANSFORMATION

Aleksandra Chuchvara, Atanas Gotchev

Tampere University, Tampere, Finland

ABSTRACT

We propose simple and efficient method that produces content-adaptive superpixels, i.e. smaller segments in content-dense areas and larger segments in content-sparse areas. Previous adaptive methods distribute superpixels over the image according to image content. In contrast, we transform the image itself to redistribute the content density uniformly across the image area. This transformation is guided by a significance map, which characterizes the 'importance' of each pixel. Arbitrary superpixel algorithm can be utilized to segment the transformed image into regular superpixels, providing a suitable representation for subsequent tasks. Regular superpixels in the transformed image induce content-adaptive superpixels in the original image facilitating the improved segmentation accuracy.

Index Terms— Superpixel, image segmentation

1. INTRODUCTION

Superpixel segmentation methods aim to group image pixels into small perceptually homogeneous regions with boundaries adhering to image contours. Superpixels often serve as an intermediate representation for high-level image processing tasks providing two major advantages over raw pixels: reduced computational complexity and better spatial support for region-based feature extraction. Consequently, superpixel over-segmentation has become a standard pre-processing step that have been used in wide range of computer vision and image processing applications, including semantic segmentation [1], object recognition and tracking [2, 3], depth and optical flow estimation [4, 5, 6], 3D reconstruction [7], localization [8], and many other.

It is difficult to define universal performance criteria for all superpixel algorithms, yet computational efficiency and segmentation accuracy with as few superpixels as possible are commonly desirable properties. In addition, many authors agree that superpixel regularity, in terms of size and shape, is beneficial for subsequent processing, e.g. feature extraction or graph construction. In practice, these are conflicting requirements, and a compromise has to be made between computational time, boundary adherence, regularity and number of superpixels. Existing superpixel algorithms put focus on different performance aspects that may be beneficial for a specific application, e.g. runtime [9, 10], boundary adherence [10, 11], regularity [12, 13], topology [9, 14].

The superpixel methods developed in the recent years can be roughly grouped into three categories: graph-based, clustering-based, and methods based on energy optimization. A recent benchmark study [15] presents a comprehensive evaluation and ranking of 28 state-of-the-art superpixel algorithms. Among the top-performing algorithms are graph-based ERS [11], energy-based SEEDS [10] and CFTP [9], and clustering-based SLIC [13]. While ERS and SEEDS can achieve high boundary precision, they lack a regularity constraint and produce superpixels with irregular shapes. Focusing on real-time performance, CFTP exhibits high boundary adherence, while a topology preserving term allows to control superpixel regularity. Simple and time-efficient SLIC provides direct control over the number of superpixels and their regularity. However, while regular superpixels have similar size and approximately uniform distribution, it is difficult to obtain optimal performance for images containing both homogeneous regions and fine structures. In this case, the choice of superpixel size is a trade-off between faithful representation of image details and excessive over-segmentation of homogeneous regions. To achieve a better trade-off between segmentation accuracy and the number of superpixels, few methods propose to split/merge initial uniform superpixels or distribute superpixel seeds non-uniformly depending on the image content [16, 17, 18]. This way, smaller and denser superpixels are produced in high complexity regions and larger superpixels in homogeneous regions. However, all these methods require geodesic distance evaluation and are much more time-consuming compared to fast superpixel methods such as [9, 10, 13].

We approach the task of content-sensitive segmentation from a different perspective. Instead of splitting/merging or adaptively distributing superpixels over the image, we propose to optimize the use of the image space by transforming the image itself, such that content-dense regions are stretched, whereas content-sparse regions are contracted. When the image content is distributed more uniformly over the whole image area, it is appropriate to segment image into regular size superpixels. By mapping the segmentation map back to the initial image space, we conversely obtain a content-adapted superpixel segmentation where the segments are smaller in content-dense areas and larger in content-sparse regions. Any superpixel segmentation method can be utilized to produce regular superpixels. As we do not alter the segmentation process, all important properties of the underlying segmentation method are preserved, e.g. connectivity, topology, and time-efficiency.

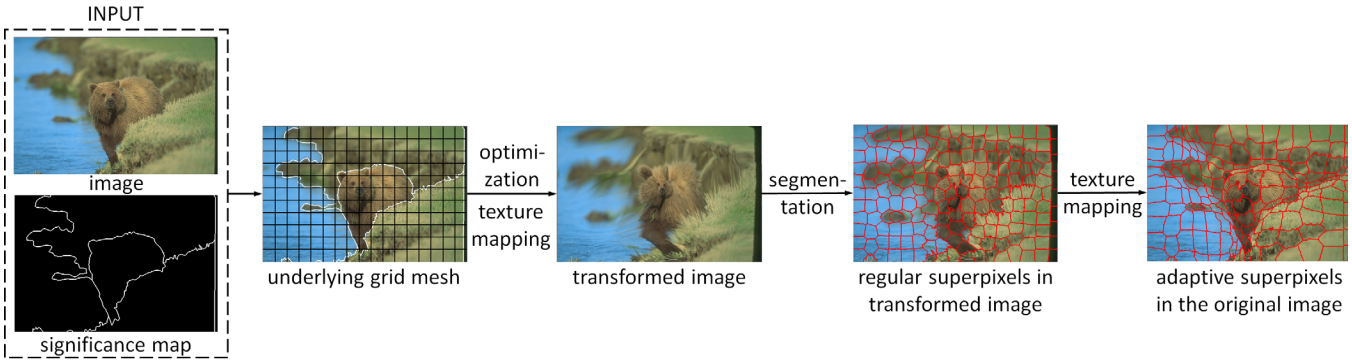


Fig. 1. The flow-chart of the proposed method.

2. PROPOSED APPROACH

The workflow of our method is illustrated in Fig. 1. As an input, we take a raster image and a significance map that provides 'importance' value for each pixel. The goal is to transform the input image so that the image regions occupy areas that are proportional to their significance. To obtain the transformation function, we represent the input image as a regular grid mesh and deform the initial mesh by minimizing a weighted quadratic energy, where all the weights are positive. The minimization is achieved by solving a sparse linear system of equations, where the positions of the grid vertices are variables in the global optimization process. This resembles stretch-minimization techniques that are used to produce low-stretch planar mesh parameterizations for 3D surfaces via redistribution (diffusion) of local mesh stretches [19, 20, 21]. A significance map is used to guide the weights choice ensuring that significant regions are naturally allowed to occupy a larger image area, while insignificant regions (e.g. regions with homogeneous content) are contracted. The system solution is a one-to-one mapping and edge flips never happen [21], i.e. the mesh will not fold on itself. Thus, we can apply texture mapping to render the transformed image. Likewise, segmentation for the original image is obtained from the segmentation map of the transformed image by texture-mapping.

2.1. Significance map

Significance map provides the relative importance of different regions in the image. Each entry in the map represents a significance measure of a single pixel in the input image, where values range between 0 and 1, with 0 assigned to non-significant pixels. Various kinds of prior information can be incorporated into significance map to guide the image transformation, e.g. edge or contour prior [22, 23, 24], depth map [25], saliency map, or any arbitrary combination of such. In our experiments, we use contour prior (Section 3.1). We use binary contours following the assumption that contour presence is significant independently of the edge strength.

2.2. Mesh representation

We represent the input image as a 4-connected grid mesh $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_N\}$ is the set of vertex positions and $E = \{(i,j)\}$ is the set of edges, N is the number of the mesh vertices. The vertices and edges form horizontal and vertical grid lines partitioning the image into quads. The number of rows and columns in the grid is proportional to the width and height of the input image. Significance of a quad can be defined based on pixel significance within the quad, e.g. as an average value. In our experiments, we use binary values and set quad significance to 1 if it has at least one significant pixel (i.e. if it contains a contour).

2.3. Optimization

In order to stretch or contract the grid quads according to their significance, we move the grid vertices to new locations so that the distance between two vertices v_i and v_j connected by an edge (i,j) is proportional to the significance of the quads sharing this edge. The new vertex locations are selected so that the following local quadratic energy is minimized:

$$E(v_i) = \sum_j w_{ij} \|v_i - v_j\|^2. \quad (1)$$

Here v_i is the vertex of the grid, and v_j is its one-link neighbor. The weight w_{ij} is a positive value inversely proportional to quad significance and defined as:

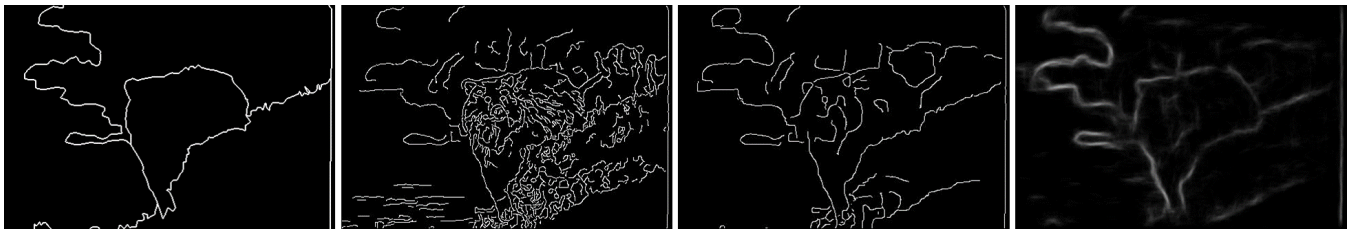
$$w_{ij} = \frac{1}{(\alpha + s_{ij})}, \quad (2)$$

where s_{ij} is the average significance of the quads sharing (i,j) and α is a positive constant added to avoid zero division.

The minimization of $E(v_i)$ is a classical least-squares optimization problem. The optimal position for v_i can be obtained by solving a sparse system of linear equations:

$$\sum_j w_{ij}(v_i - v_j) = 0, \quad (3)$$

subject to boundary constraints that are substituted into the linear system during the optimization.



(a) (b) (c) (d)

Fig. 2. Contour prior illustration: (a) example ground true contour map; (b) Canny edge detector [29]; (c) structure-texture decomposition [30]; (d) random forest contour detector [31].

The boundary constraints are the locations of the corner vertices that must remain fixed:

$$v_{tl} = (0,0)^T, v_{tr} = (w,0)^T, \\ v_{bl} = (0,h)^T, v_{br} = (w,h)^T, \tag{4}$$

where w is the image width and h is the image height. In addition, to keep the image shape rectangular all the rest boundary vertices are constrained to slide only along directions aligned with the boundary:

$$v_i(x) = \begin{cases} 0, & \text{if } v_i \text{ is on the left boundary} \\ w, & \text{if } v_i \text{ is on the right boundary} \end{cases}, \\ v_i(y) = \begin{cases} 0, & \text{if } v_i \text{ is on the top boundary} \\ h, & \text{if } v_i \text{ is on the bottom boundary} \end{cases}. \tag{5}$$

2.4. Image transformation

The edge contraction process redistributes edge lengths: edges of higher significance will get longer, while those of smaller significance will get shorter. It was proven [21], that if the weights are positive and symmetric, such optimization procedure does not generate edge flips, i.e. the mesh does not fold on itself. Hence, we can render the transformed image by texture-mapping the initial image onto the deformed grid mesh. We triangulate the deformed mesh and interpolate texture coordinates within each triangle in order to sample the input image appropriately. The texture mapping procedure is fast and fully supported by graphics hardware. Likewise, after the transformed image is segmented into regular superpixels, segmentation of the original image is obtained by mapping the transformed image segments back to the initial grid mesh.

3. EXPERIMENTAL RESULTS

We chose two representative algorithms to evaluate the proposed method: CFTP [9] and SLIC [13]. According to the benchmark study [15], CFTP provides the best quantitative results, while SLIC is the most widely used algorithm in practical applications; both algotithms produce regular superpixels. The algorithms are tested using benchmark source code [15] with the default parameters specified in the benchmark library. The same parameters are used to segment initial and transformed images. We then compare the results obtained with and without image transformation in order to demonstrate the relative performance improvement.

Commonly used quality evaluation metrics are boundary recall (BR) [26], achievable segmentation accuracy (ASA) [11] and undersegmentation error (UE) [27]. Better segmentation performance corresponds to higher BR and ASA, and lower UE. We use the benchmark implementation of these metrics, as defined in [15]. We perform our experiments on BSDS500 dataset [28] containing 500 images (200 training, 100 validation, 200 testing) with resolution 321×481. For each image, at least four ground truth human-labeled multi-class segmentations are provided.

For image transformation, we need to choose the density of the grid mesh. A finer mesh leads to better approximation of the significance map but increases the computational cost. In our experiments, we use a grid mesh containing approximately 400 quads, which produces sufficiently good results. Apart from that, our method requires only one additional parameter α, which is a positive constant that is used to vary the strength of the image deformation, α is fixed to 0.25 in all our experiments.

3.1. Contour prior

In our experiments, we use contour prior as a significance map. To investigate the influence of a contour map accuracy, we obtain contour maps using three different methods: standard Canny edge detection [29], filter-based structure-texture decomposition [30], and a contour detection based on a random forest classifier [31]. The example contour maps obtained by each method are shown in Fig. 2. Canny edge detector is fast and simple, however it is sensitive to textures, producing noisy edges that are not true contours (Fig. 2(b)). We can improve the Canny edge detection results by applying structure-texture decomposition method [30] that removes texture from an image by filtering the image gradient. As a result, we can extract faithful contours with relatively low computational complexity (Fig. 2(c)). Fast and accurate random forest contour detector [31] based on convolutional neural network provides a contour strength estimate at each image pixel (Fig. 2(d)). We apply Canny edge detector for binarization of these contours.

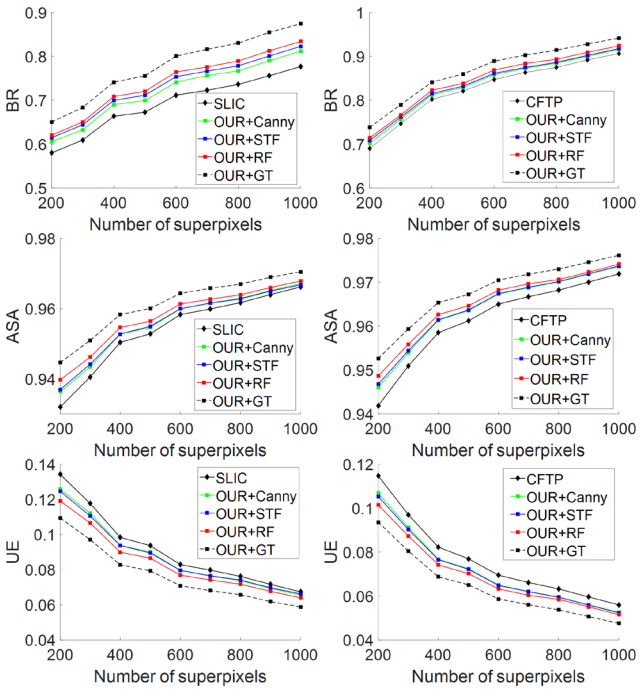


Fig. 3. Quantitative comparison on BSDS500 dataset with (a) SLIC, (b) CFTP: top row BR↑, middle row ASA↑, bottom row UE↓.

3.2. Performance

The results of quantitative comparison are shown in Fig. 3. The performance metrics are plotted against varying number of superpixels $K \in [200,300,\ldots,1000]$. As there are several ground truths segmentations available for each image, we report the results averaged over all ground truths and all images. The evaluation is performed using 200 images of the test subset. For each metric, the baseline performance curve of the underlying segmentation algorithm is provided along with several plots for our method obtained with different contour priors. It can be observed, that even when using basic Canny edge detection our method demonstrates improved performance over the baseline algorithms in terms of all three metrics. It is also clear that better contour detection accuracy yields better performance. We thus also include a plot obtained using a ground truth contour map (we use the rest of the ground truth maps for evaluation) that provides a good estimate of the achievable improvement of our method.

Fig. 4 provides visual results for comparison with CFTP and SLIC. As can be seen, the superpixels obtained after image transformation are content-adaptive. Smaller size of the superpixels near objects boundaries facilitates better object separation from the background regions, even when the colors between two regions are quite similar. Thus, for a specific number of superpixels, this allows to achieve higher boundary adherence compared to the baseline methods.

Since superpixel segmentation is usually used as a pre-processing step, runtime is an important performance factor.

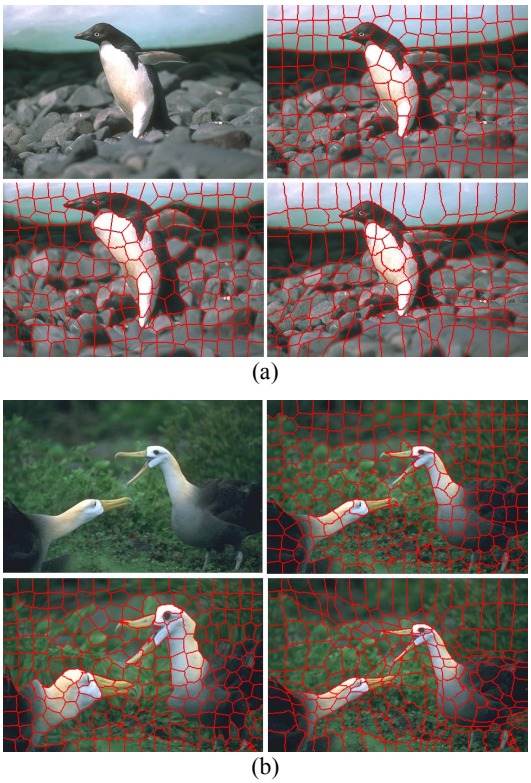


Fig. 4. Visual comparison with (a) SLIC and (b) CFTP. Clockwise from top left: input image; SLIC and CFTP superpixels; our final results; SLIC and CFTP segmentation of the transformed image

The most computationally intensive part of our method is the minimization of the quadratic energy function. This involves solving sparse linear system of size $2N \times 2N$, containing only five non-zero coefficients in each row, which can be solved efficiently using a sparse matrix solver. E.g., using Matlab implementation of preconditioned conjugate gradients [32] (iterations 100, error 10^{-6}) and a laptop with a 2.6 GHz CPU, we can obtain solution for N =500 in 10ms and for N =1150 in 18ms.

4. CONCLUSION

We have presented an efficient method for producing content-adaptive superpixels via image transformation that takes only a few milliseconds per image. Experimental results on the BSDS500 dataset demonstrate the advantages of our method over the baseline methods in application to image segmentation using a contour prior. In general, various kinds of prior information can be utilized to guide the image transformation. Furthermore, any superpixel segmentation algorithm can be applied to segment the transformed image into regular superpixels ensuring other desired segmentation properties, such as connectivity, topology, and time-efficiency. Therefore, we see the proposed method as a generic approach that can be applied in wide variety of applications. However, more experiments are needed to demonstrate its advantages in other applications.

5. REFERENCES

[1] J. Xiao and L. Quan. Multiple view semantic segmentation for street view images. In ICCV, 2009.

[2] J. Yan, Y. Yu, X. Zhu, Z. Lei, and S.Z. Li. Object detection by labeling superpixels. In CVPR, 2015.

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

[4] F. Liu, C. Shen, G. Lin, and I. Reid. Deep convolutional neural fields for depth estimation from a single image. In CVPR, 2015.

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

[6] J. Lu, H. Yang, D. Min, and M.N. Do. Patch match filter: efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In CVPR, 2013.

[7] A. Bódis-Szomorú, H. Riemenschneider, and L. van Gool. Superpixel meshes for fast edge-preserving surface reconstruction. In CVPR, 2015.

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

[9] J. Yao, M. Boben, S. Fidler, and R. Urtasun. Real-time coarse-to-fine topologically preserving segmentation. In CVPR, 2015.

[10] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool. SEEDS: Superpixels extracted via energy-driven sampling. In ECCV, 2012.

[11] M. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpiexl segmentation. In CVPR, 2011.

[12] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(12): 2290–2297, 2009.

[13] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11):2274—2282, 2012.

[14] H. Fu, X. Cao, D. Tang, Y. Han, and D. Xu. Regularity preserved superpixels and supervoxels. IEEE Trans. Multimedia, 16(4):1165–1175, 2014.

[15] D. Stutz, A. Hermans, and B. Leibe. Superpixels: An evaluation of the state-of-the-art. Computer Vision and Image Understanding, 166(C): 1–27, 2018.

[16] P. Wang, G. Zeng, R. Gan, J. Wang, and H. Zha. Structure-sensitive superpixels via geodesic distance. International Journal of Computer Vision, 103(1):1–21, 2013.

[17] Y.-J. Liu, M. Yu, B.-J. Li, and Y. He. Intrinsic manifold SLIC: A simple and efficient method for computing contentsensitivesuperpixels. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(3):653–666, 2018.

[18] A. Rubio, L. Yu, E. Simo-Serra, and F. Moreno-Noguer. BASS: Boundary-aware superpixel segmentation. In ICPR, 2016.

[19] S. Yoshizawa, A. Belyaev, and H. P. Seidel. A fast and simple stretch-minimizing mesh parameterization. In Proceedings Shape Modeling Applications, 2004.

[20] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. Foundations and Trends in Computer Graphics and Vision, 2(2): 105-171, 2006.

[21] M. S. Floater and K. Hormann. Recent advances in surface parameterization. In Multiresolution in Geometric Modelling, 2003.

[22] A. Moore, S. Prince, and J.Warrell. “Lattice cut” – constructing superpixels using layer constraints. In CVPR, 2010.

[23] S.-H. Lee, W.-D. Jang, and C.-S. Kim. Contour-constrained superpixels for image and video processing. In CVPR, 2017.

[24] R. Giraud, V.-T. Ta, and N. Papadakis. SCALP: Superpixels with contour adherence using linear path. In ICPR, 2016.

[25] D.Weikersdorfer, D. Gossow, and M. Beetz. Depth-adaptive superpixels. In ICPR, 2012.

[26] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. IEEE Transactions on PAMI, 26(5):530–549, 2004.

[27] P. Neubert and P. Protzel. Superpixel benchmark and comparison. In Proc. of Forum Bildverarbeitun, 2012.

[28] 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 ICCV, 2001.

[29] J. Canny. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986

[30] H. Lee, J. Jeon, J. Kim, and S. Lee. Structure-texture decomposition of images with interval gradient. In Computer graphics forum, 36:262–274, 2017.

[31] P. Dollar and C. L. Zitnick. Structured forests for fast edge detection. In ICCV, 2013.

[32] R. Barrett, M. Berry, and T. F. Chan. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. In SIAM, 1994.