To appear in Proc. of International Conference on Pattern Recognition (ICPR), 2014. DOI: not yet available

Compact Watershed and Preemptive SLIC: On improving trade-offs of superpixel segmentation algorithms

Peer Neubert and Peter Protzel
Chemnitz University of Technology
Department of Electrical Engineering and Information Technology
09126 Chemnitz, Germany
{peer.neubert, peter.protzel}@etit.tu-chemnitz.de

***Abstract*—A major insight from our previous work on extensive comparison of superpixel segmentation algorithms is the existence of several trade-offs for such algorithms. The most intuitive is the trade-off between segmentation quality and runtime. However, there exist many more between these two and a multitude of other performance measures. In this work, we present two new superpixel segmentation algorithms, based on existing algorithms, that provide better balanced trade-offs. Better balanced means, that we increase one performance measure by a large amount at the cost of slightly decreasing another. The proposed new algorithms are expected to be more appropriate for many real time computer vision tasks. The first proposed algorithm, Preemptive SLIC, is a faster version of SLIC, running at frame-rate (30 Hz for image size 481x321) on a standard desktop CPU. The speed-up comes at the cost of slightly worse segmentation quality. The second proposed algorithm is Compact Watershed. It is based on Seeded Watershed segmentation, but creates uniformly shaped superpixels similar to SLIC in about 10 ms per image. We extensively evaluate the influence of the proposed algorithmic changes on the trade-offs between various performance measures.**

I. Introduction

Superpixels are a special case of an image oversegmentation or - seen the other way around, a perceptual grouping of pixels. They have become key building blocks of many image processing and computer vision algorithms. They are used for object recognition [1], segmentation [2], scene change prediction [3], multi-class object segmentation [4], depth estimation [5], body model estimation [6] and many other tasks. Inspired by this multitude of applications, a considerable number of superpixel segmentation algorithms has been proposed. Important performance measures for superpixel algorithms are:

- Segmentation quality (e.g. boundary recall, undersegmentation error, segmentation stability)
- Runtime
- Characteristics of the superpixel size, shape and distribution over the image plane

A main insight from our previous work [8][9] with superpixels is that each algorithm is a trade-off between these performance measures. In theory, there may exist a perfect superpixel segmentation algorithm, that runs sufficiently fast and provides the perfect segments for the subsequent processing steps. However, in practice, we have to look for the existing algorithm that provides the best balanced trade-off for the task at hand.

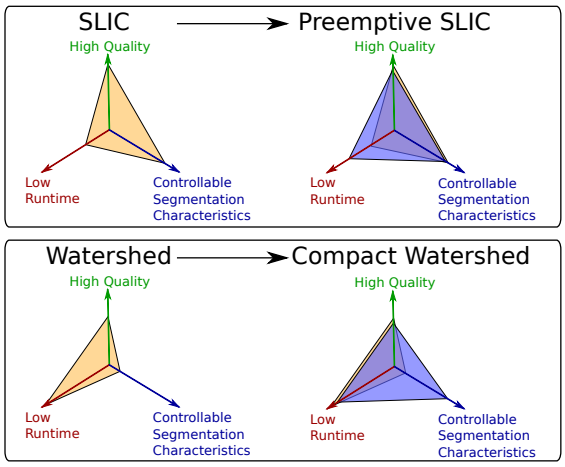

Fig. 1. In this work we propose two new adaptions of existing algorithms that change the balance in trade-offs of superpixel segmentation algorithms. The three colored axes visualize a set of typical trade-offs: segmentation quality measures, runtime and appropriate segmentation characteristics for subsequent processing steps (e.g. similar size). The application of the segmentation results may give a weighting to the different axes. But in general, a triangle with larger area indicates a better segmentation algorithm. The proposed Preemptive SLIC algorithm is a much faster version of SLIC [7], running at frame-rate (30 Hz) on a standard desktop CPU. This comes at the cost of slightly worse segmentation quality. The proposed Compact Watershed algorithm is a simple extension of a seeded watershed segmentation, that conserves the high speed and gives the user control over the compactness of the segments.

In this paper, we present two new adaptions of superpixel segmentation algorithms that better balance important trade-offs. Fig. 1 illustrates the main contributions of this paper. First, we present an adaption of SLIC [7], Preemptive SLIC, that preserves the high segmentation quality level of the original implementation and runs at 30 Hz on a standard desktop CPU. Therefore we propose changes on the algorithm, that adjust the trade-off between runtime and quality in favor of a frame-rate execution. We evaluate the positive and negative effects on segmentation quality and stability in detail based on our previously published benchmarks. Second, after speeding up a good algorithm, we try to improve the segmentation of a very fast algorithm: Seeded Watershed [10]. For application as typical superpixels the main issue

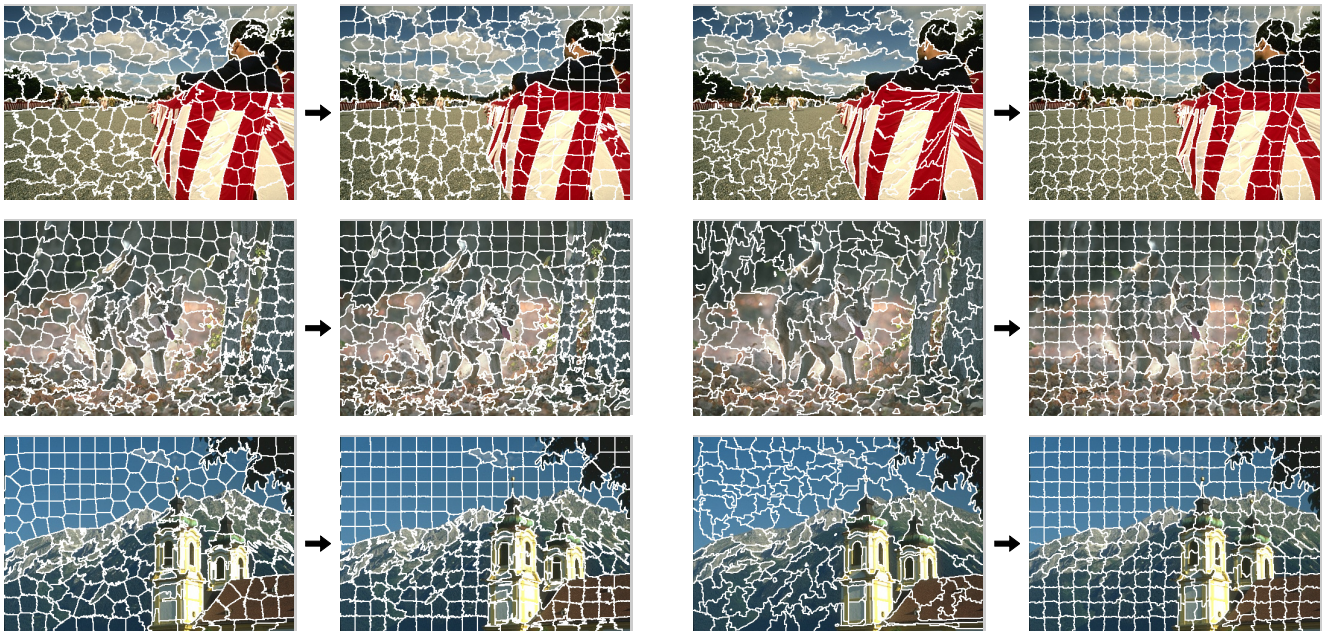

Fig. 2. Example segmentations. From left to right: SLIC, the proposed Preemptive SLIC, Watershed and the proposed Compact Watershed. Preemptive SLIC is three times faster than SLIC and results in similar segmentations. One can see how the segment borders are closer to the initial rectangular shape at homogeneous image regions. The third colum shows seeded Watershed segmentations. The irregular segment size and shape are clearly visible. The compactness constraint of Compact Watershed makes the segmentation much more regular (parameter $c = 1$). This improves undersegmentation error and motion discontinuity error but comes at the cost of lower boundary recall. Compact Watershed takes about 10 ms to segment an image of size (481×321) using a standard desktop CPU.

with Watershed segmentations is the highly irregular size and shape of the resulting segments. We propose and evaluate the extension with a compactness constraint that turns Watershed segmentation from an oversegmentation algorithm into a real superpixel algorithm. This shifts the balance at several trade-offs between metrics for segmentation quality, stability and runtime which we are going to evaluate in detail in the results section. Furthermore, we provide open source implementations of the two new adapted algorithms on our website[1].

The next section sets this paper in the context of related work, followed by descriptions of the algorithmic steps to change the trade-offs for SLIC and Watershed segmentation. The results section gives insights on the influences of the proposed algorithmic changes on the properties of the segmentations

II. Related Work

There is no clean distinction between an oversegmentation algorithm and superpixel segmentations. By consensus, an oversegmentation with more or less regularly shaped segments of similar size, that are uniformly distributed over the image plane is considered a superpixel segmentation. Thus each superpixel segmentation is an oversegmentation (but not vice versa). Nevertheless, in the following we use the term *over-segmentation* for oversegmentations that are not a superpixel segmentation. There exists a wide range of algorithms for both types. Extensive comparison of existing algorithms is beyond the scope of this paper but can be found in our prior work [8][9]. These comparisons reveal significant trade-offs in the existing algorithms. The most intuitive is the trade-off between segmentation quality and runtime. Moreover, at some metrics, oversegmentation algorithms perform better than superpixel algorithms, e.g. boundary recall. At boundary recall, neither regularly shaped segments nor a grid like distribution are helpful. For other metrics, e.g. undersegmentation error, constraints on the size and shape of the segments help to prevent segments from growing unbounded at homogeneous image regions. Such constraints also influence the stability of segmentations. From the existing algorithm comparisons on segmentation quality [9] and stability [8] we choose two interesting examples for trade-offs:

- SLIC shows good quality and stability but at lower speed
- Watershed shows worse quality and stability, but at very high speed

In this paper we present approaches to do both: to speed up the good-but-slow algorithm SLIC and to improve the worse-but-fast algorithm Watershed.

We are not the first laying hands on these well-know algorithms. E.g. there exists a frame-rate GPU version of SLIC [11] and for sure there exist various implementations and distance metrics for watershed segmentation, e.g. see [12] for an overview. However, to the best of our knowledge, we are the first to present a frame-rate CPU version of SLIC and a ready-to use compact superpixel segmentation based on watersheds. Moreover, we evaluate the properties of the new algorithms in detail.

In the following sections, we briefly introduce the proposed algorithms. We start with speeding up SLIC followed by

[1] http://www.tu-chemnitz.de/etit/proaut/forschung/superpixel.html

details on how we introduced compactness in a Watershed segmentation.

III. Preemptive SLIC: Making a good algorithm faster

Simple Linear Iterative Clustering (SLIC) is based on the concept of a local k-means clustering. The cluster centers are initialized on a uniform grid in the image plane. *Local k-means* denotes that each cluster considers only pixels in its spatial neighborhood. The metric during clustering is a weighted combination of color distance and spatial distance in the image plane. The weight of the spatial component (the "compactness" parameter) influences the regularity of the resulting segments shape and size.

SLIC provides good segmentation quality and stability and can be considered as state of the art in superpixel segmentation. Although SLIC belongs to the class of faster superpixel algorithms (e.g. compared to Normalized Cuts), the runtime of about 100 ms can be considered the main drawback (e.g. compared to Watershed). In the following we present our approach on speeding up SLIC by about factor 3. The results section evaluates the impact of this shift in the trade-off between quality and runtime.

Starting from the implementation of Achanta et. al[2], we speed up SLIC in two steps: The first is a simple but effective code optimization (dropping runtime from about 100 ms to 50 ms). This is followed by an algorithmic approximation of the original algorithm, yielding a final runtime of about 33 ms on images of size (481×321).

A. Code optimized SLIC

A runtime analysis of the original implementation shows, that more than half the time is spent for color conversions from RGB to LAB. There exist several fast approximations and implementations for this color space conversion. So the fist change is a simple replacement of the original color conversion algorithm with the one provided by OpenCV[3]. Section V gives insights on the resulting benefit for the runtime and a slight decrease in segmentation quality.

B. Preemptive SLIC

As second step, we propose an approximation of the original algorithm. The main idea behind SLIC is to use k-means in a local manner by reducing the potential member pixels for each cluster to a local neighborhood. However, this is just halfway gone: SLIC uses a single, global termination criterion. We propose to use a *local* termination criterion for each cluster to avoid revisiting clusters and image areas without any major changes since the last iteration. This preemptively stops the evolution of segment boundaries in homogeneous image regions. The termination criterion of the original SLIC algorithm is the following:

If the maximum number of iteration is reached or if there is no major change in any cluster, the algorithm finishes.

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

[3]http://opencv.org

In fact the original implementation uses a fixed number of 10 iterations. However, this way, potentially large image parts without any changes in the previous iteration(s) are updated every new iteration of the main k-means loop. Thus, we propose to introduce an individual termination criterion for each cluster:

If there has not been any major change in this cluster or any neighboring cluster in the last iteration, do not update this cluster.

The global termination criterion remains almost unchanged:

If the maximum number of iterations is reached or no cluster has been updated, we are done.

This significantly reduces the number of pixel accesses at homogeneous image regions. Even if the same number of iterations in the main loop are processed, not all clusters are updated in each iteration. However, a cluster that has not been updated in one iteration can become active in the next if a cluster in its neighborhood had a major change. So the additional computational overhead reduces to counting the number of added or removed pixel for each cluster and to evaluate this number for each cluster and its neighborhood before deciding whether to update this cluster or not in the current iteration. The effects on segmentation quality and runtime are evaluated in the results section V.

IV. Compact Watershed: Getting from oversegmentations to superpixels

Watershed segmentation is a very fast algorithm. However, it suffers from irregularly sized and shaped segments and strongly varying boundaries. The user has no influence on the segmentation characteristics. This section describes our straight forward approach on incorporating a controllable compactness constraint in a watershed segmentation to evaluate its influence on segmentation quality and characteristics. In this context, high *compactness* means that the superpixels are of approximately equal size and more or less regularly shaped in the absence of strong image gradients (e.g. like a recangle or a circle).

The idea of watershed segmentation origins from [13]. The intuitive idea comes from geography: when a landscape is flooded by falling waterdrops, dependent on the amount of water there are basins filled with water and dividing ridges between them. These ridges are the watersheds. Since watershed segmentation is a well known algorithm, there are various algorithmic implementations and adaptions. We start from the OpenCV implementation of the algorithm following [10]. OpenCV implements a seeded watershed segmentation (also called marker controlled watershed). The seeds are externally provided to the algorithm, e.g. as local gradient minima or for superpixel-like segmentation on a uniform grid. The seeds grow iteratively pixel by pixel until they reach a border to the segment around another seed. These borders form the watersheds. The next seed to expand by one pixel is chosen based on a distance function. In the OpenCV implementation, this distance function only incorporates the intensity or color value of the pixels. This results in strongly varying borders in homogeneous image regions and potentially highly irregularly shaped segments in the presence of image gradients.

To influence the compactness of the segmentation, we propose the simple extension of incorporating the distance to the seed point as well. This is very similar to the compactness component in SLIC. The resulting distance metric is the weighted combination of the conventional appearance based distance and the euclidean distance of the pixel to the segment seed. This results in a constraint on the size and elongation of the segments and thus favors the creation of compact segments.

Fig. 2 shows the strong impact on the visual appearance of the segments, their shape, size and distribution on the image plane. The following section evaluates the influence of the strength of the compactness parameter on various trade-offs between quality, stability and runtime.

V. Experiments and Results

In this section we compare the changes in the balance of the trade-offs form SLIC to Preemptive SLIC and from Watershed to Compact Watershed. The experiments build upon our previously published superpixel benchmarks [8][9]. While, in this work, we only compare the original algorithms with their proposed extension, we refer the reader to our previous work for a comparison with other algorithms (including e.g. Normalized Cuts [14], Felzenszwalb-Huttenlocher Segmentation [15], Edge Augmented Mean Shift [16], Quickshift [17] and Entropy Rate Superpixel Segmentation [18]).

A. Metrics for Comparison

The comparison is based on segmentation runtime, quality and stability. Used datasets are the Berkeley Segmentation Data Set BSDS500 [19] and the Sintel dataset [20]. We evaluate the following metrics:

Runtime is measured as the average runtime on the 200 BSDS500 test images. The images are of size 482×321. All computations are done on a Intel Core i7-3770 CPU @ 3.40GHz with 16 GB RAM.

Boundary recall measures the rate of object boundaries that are covered by superpixel boundaries. The ground-truth object boundaries are obtained as the manual labels in the BSDS500 dataset. We follow [9]: Given a ground truth boundary image G, the algorithms boundary image B and a maximum distance $d = 1$. TP (True Positives) is the number of boundary pixels in G for whose exist a boundary pixel in B in range d. FN (False Negatives) is the number of boundary pixels in G for whose does not exist a boundary pixel in B in range d. Then boundary recall is $BR = \frac{TP}{TP+FN}$.

Undersegmentation error is a somehow contrary measure to boundary recall. Algorithms with high boundary recall often have a high undersegmentation error. It measures how well ground-truth object segments can be recovered as combination of superpixels. Again, for computation of the metric, we follow [9]: Each ground truth segment S divides a superpixel P in an P_{in} and an P_{out} set of pixels. Being N the total number of pixels, undersegmentation error is computed as:

$$USE = \frac{1}{N}\left[\sum_{S \in GT}\left(\sum_{P:P\cap S\neq\emptyset} min(P_{in}, P_{out})\right)\right] \quad (1)$$

Motion undersegmentation error evaluates the stability of a segmentation algorithm by comparing segmentations of consecutive frames of a video. This measure is based on ground-truth optical flow information about the motion of each pixel from one frame to the other. It measures how well the segmentation algorithm finds the same regions or object boundaries independent of changes in the image. We use the equations and data provided in [8]: Being L_1 and L_2 segmentations of images connected through ground-truth optical flow field F, then L_1^F is the result of applying F on L_1. Similar to undersegmentation error, we compute the motion undersegmentation error as:

$$MUSE = \frac{1}{N}\left[\sum_{a \in L_1^F}\left(\sum_{b \in L_2 : a\cap b\neq\emptyset} min(b_{in}, b_{out})\right)\right] \quad (2)$$

Motion discontinuity error is also based on ground-truth optical flow data. It measures how well motion discontinuities in the image sequence are represented by the superpixel boundaries. Again, we use the equations and data from [8]: Given F, a ground truth optical flow field from an image I to another image, B the boundary image of a segmentation of image I, and $D(B)$ the distance transform of B containing for each pixel the distance to the nearest segment boundary, we define the Motion Discontinuity Error (MDE) as follows:

$$MDE = \frac{\sum_{(i,j)\in I} \|\nabla F(i,j)\|_2 \cdot D(B(i,j))}{\sum_{(i,j)\in I} \|\nabla F(i,j)\|_2} \quad (3)$$

For good performance at MDE metric, a superpixel algorithm should place segment boundaries between differently moving image parts. We start with the evaluation of Preemptive SLIC, followed by results on Compact Watershed.

B. Results on Preemptive SLIC

Fig. 3 *(left)* shows the runtime improvement for the code optimized version of SLIC and Preemptive SLIC. Code optimization decreased the runtime from about 100 ms for typical superpixel segment numbers, down to about 50 ms. In combination with the proposed local termination criterion, the resulting Preemptive SLIC algorithm takes about 33 ms and runs at frame rate (30 Hz) on a standard desktop computer. SLIC already showed to be well balanced at segmentation quality and stability before [8]. The boundary recall and undersegmentation error measurements in Fig. 3 show the slight decrease in segmentation quality for the code optimized version and Preemptive SLIC. The larger decrease comes from the code optimization, although it only changes the implementation of the color space transformation. The preemptive termination causes a comparatively small additional loss.

The general similarity between the original SLIC and Preemptive SLIC segmentation results can be seen in the example segmentations in Fig. 2. The local termination criterion and the resulting preemptive stop of the evolution of superpixel boundaries cause segments to remain more similar to the initial grid-like shape. This manifests as straight vertical or horizontal segment borders in homogeneous images areas. Since there is only few or even no cause for a deviation from the initial shape in the underlying image data at such homogeneous image

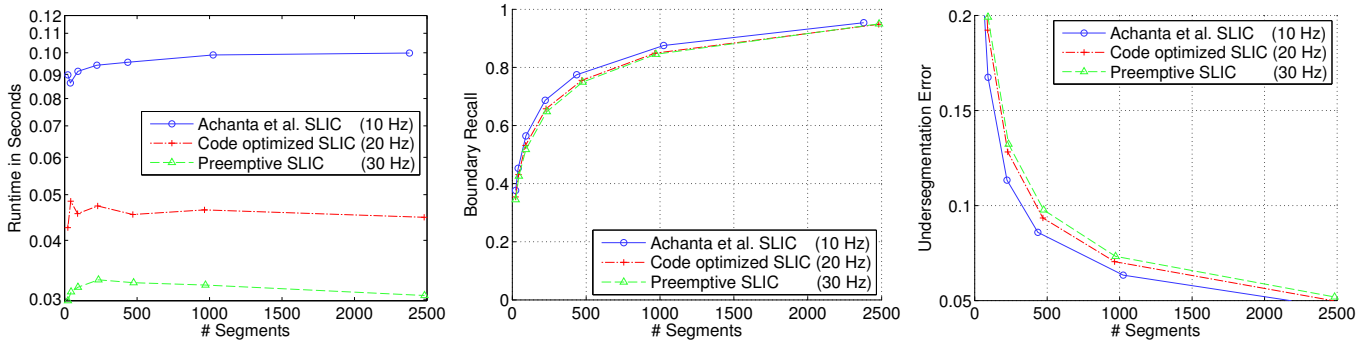

Fig. 3. We compared the original SLIC algorithm, the code optimized version and Preemptive SLIC on the BSDS500 dataset. The left plot shows the runtime improvement for the code optimized version and Preemptive SLIC. This comes at a slight decrease in boundary recall and higher undersegmentation error from the original SLIC to the code optimized version and further to Preemptive SLIC. *(Boundary recall: higher is better; Undersegmentation error: lower is better; Runtime: lower is better, logarithmic scale)*

regions, remaining such borders unchanged may improve the segmentation stability. Experiments on the ground-truth optical flow data support this assumption by showing a clear improvement in motion undersegmentation error, Fig. 4 *(left)*. As a reminder, motion undersegmentation error measures how well segmentations of consecutive frames of a video agree based on ground-truth knowledge about the motion of each pixel. Again, the improvement in segmentation stability shifts the balance in a trade-off: the detection of motion discontinuities becomes slightly worse, see Fig. 4 *(right)*.

C. Results on Compact Watershed

Watershed is a very fast algorithm. Evaluation on the BSDS500 dataset shows a runtime of about 8 ms for an image of size (481×321). A short look at the example segmentation in the third column of Fig. 2 reveals the main problem with Watershed segmentation: Although the seeds are initialized on a regular grid, the segments vary strongly in shape and size. The resulting segments of the proposed Compact Watershed (see fourth column in Fig. 2) are significantly more regularly distributed and shaped. However, they still follow object boundaries in the presence of sufficient image gradients. The results of the experiments on the BSDS500 dataset in Fig. 5 show the impact of the compactness constraint on the segmentation quality and runtime. The weight of the compactness constraint is controlled by the parameter c. $c = 0$ results in a standard seeded watershed segmentation. A strong compactness constraint prevents the segment borders to freely follow image gradients. The resulting negative influence on boundary recall can be seen in Fig. 5 *(left)*. So for pure boundary recall, a low compactness strength is favorable. At undersegmentation error the compactness constraint has two effects: On one hand, it decreases the error since it prevents precarious large segments that may result in high reconstruction penalties in the undersegmentation error computation. On the other hand, if the compactness constraint is too strong, it may cumber the adaption of superpixel segment borders to image gradients and thus to object borders. Fig. 5 *(mid)* reveals $c = 1$ as best choice for the strength of the compactness constraint. All other experiments use this setting. The additional computations for the extended distance function and the resulting changes in the evolution of the segments cause a slight increase in runtime. The stronger the compactness constraint, the larger

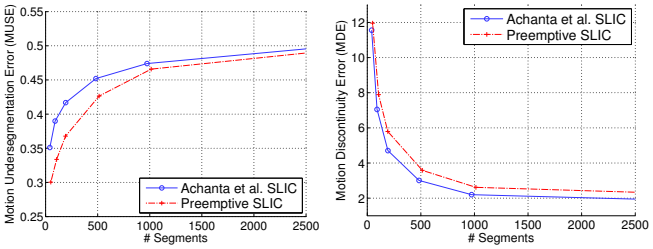

Fig. 4. Evaluation of Preemptive SLIC on the ground-truth optical flow dataset. The boundaries of Preemptive SLIC are more regular in homogeneous image regions since the local termination criterion stops optimization earlier. This shifts the balance between motion stability and coverage of motion discontinuities: Preemptive SLIC produces more stable segmentations indicated by a lower motion undersegmentation error. This comes at the cost of fewer detected motion discontinuities. *(At both metrics: lower is better)*

the additional runtime. For $c = 1$ this is about 1 ms for typical segment numbers.

The shift in the balance of trade-offs continues at the evaluation on the ground-truth optical flow dataset. For Compact Watershed the changes are vice versa to Preemptive SLIC. The compactness constraint is an influence on the segmentation that is not supported by any apparent image content. Therefore segment boundaries introduced by the compactness constraint do not move according to the image motion and are thus unstable. This effect is partially compensated by avoiding large segments that lay in differently moving image parts. The evaluation on the ground-truth optical flow dataset support this observation (Fig. 6): the motion undersegmentation error slightly increases while the motion discontinuity error slightly improves.

VI. Conclusion

We presented two adaptions of superpixel segmentation algorithms and evaluated the changes in trade-offs between various performance measures. The first new algorithm is Preemptive SLIC. It runs at frame-rate (30 Hz) on a standard desktop CPU and conserves the high segmentation quality level of SLIC. The runtime improvement comes from two step: an optimized implementation and an preemptive termination criterion for each local cluster. Preemptive SLIC also showed to create more stable segmentations.

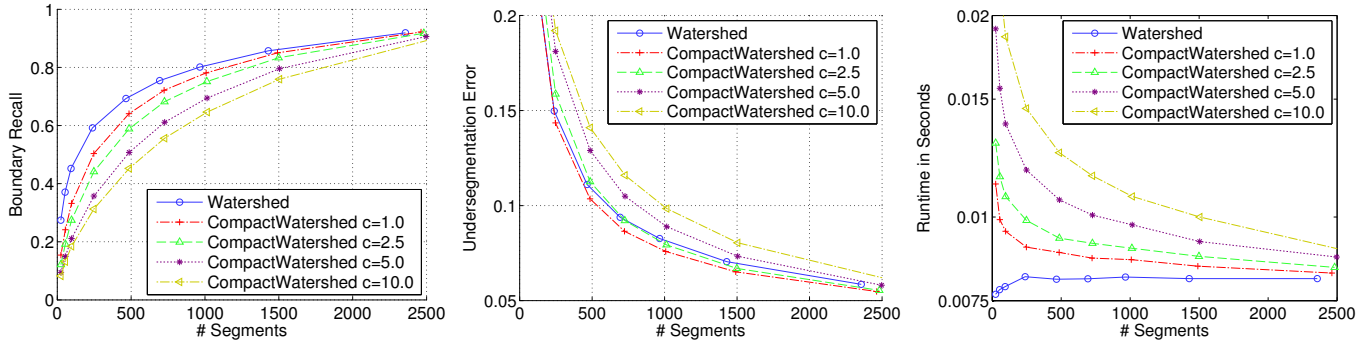

Fig. 5. We compared the original Watershed algorithm and Compact Watershed with various settings of the strength c of the compactness constraint on the BSDS500 dataset. *(left)* Boundary recall decreases with increasing compactness. For boundary recall and undersegmentation error measures, Watershed shows the same results as Compact Watershed with $c = 0$ (not visualized). *(mid)* Compactness has two influences on undersegmentation error: compact segments prevent from very large segments, but too strong compactness cumbers the adaption to object boundaries. This results in an optimum value of $c = 1$ in terms of undersegmentation error. *right)* The additional computations in the distance function slightly increase the runtime. Moreover, a strong compactness constraint and few segments compared to the image size hinder the evolution of segment boundaries and increase runtime as well. *(Boundary recall: higher is better; Undersegmentation error: lower is better; Runtime: lower is better, logarithmic scale)*

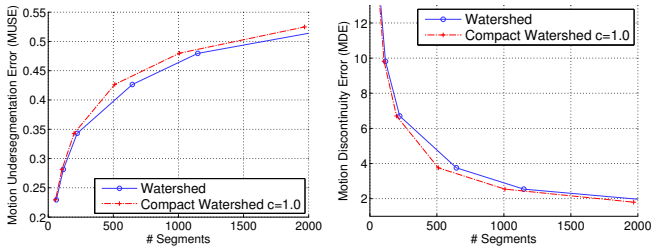

Fig. 6. Evaluation of Compact Watershed on the ground-truth optical flow dataset. Compactness is an appearance independent influence on the segment boundaries. This can decrease the segmentation stability as can be seen in the left plot. In contrast, the compactness also puts a constraint on the maximum distance of an image pixel to the nearest segment border and thus helps to cover motion discontinuities that are not indicated by image edges. The resulting improvement in motion discontinuity error can be seen in the right plot. *(At both metrics: lower is better)*

The second algorithm is Compact Watershed. It incorporates a compactness constraint in a seeded watershed segmentation and gives the user control over important characteristics of the segmentation. This turns Seeded Watershed from an oversegmentation algorithm to a real superpixel segmentation algorithm. The compactness is controlled with a single parameter. Compact Watershed takes about 10 ms for segmentation of an image of size (481×321).

We evaluated the influence of the algorithmic changes on the trade-offs between various performance measures. The individual importance of the different performance indicators depends on the application of the resulting segmentations. However, the proposed algorithms improve some performance measures for a large amount, while only slightly decreasing others. Open source implementations of both algorithms are provided on our website noted at the first page.

REFERENCES

[1] C. Pantofaru, C. Schmid, and M. Hebert, "Object recognition by integrating multiple image segmentations," in *European Conf. on Computer Vision (ECCV)*, 2008.

[2] P. Mehrani and O. Veksler, "Saliency segmentation based on learning and graph cut refinement," in *Brit. Mach. Vis. Conf.(BMVC)*, 2010.

[3] P. Neubert, N. Sünderhauf, and P. Protzel, "Appearance Change Prediction for Long-Term Navigation Across Seasons," in *Proc. of European Conference on Mobile Robotics (ECMR)*, 2013.

[4] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes, "Layered object detection for multi-class segmentation," *Proc. IEEE Conf. on Comp. Vision a. Pattern Recog. (CVPR)*, 2010.

[5] C. L. Zitnick and S. B. Kang, "Stereo for image-based rendering using image over-segmentation," *Int. J. Comput. Vision*, vol. 75, no. 1, 2007.

[6] G. Mori, "Guiding model search using segmentation," in *Int. Conf. Comp. Vision (ICCV)*, 2005.

[7] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. on Pat. Anal. and Mach. Intel.*, vol. 34, 2012.

[8] P. Neubert and P. Protzel, "Evaluating Superpixels in Video: Metrics Beyond Figure-Ground Segmentation." in *Proc. of British Machine Vision Conference (BMVC)*, 2013.

[9] P. Neubert and P.Protzel, "Superpixel benchmark and comparison," in *Proc. Forum Bildverarbeitung*, 2012.

[10] F. Meyer, "Color image segmentation," in *Int. Conf. Image Processing (ICIP)*, 1992.

[11] C. Y. Ren and I. Reid, "gslic: a real-time implementation of slic superpixel segmentation," University of Oxford, Department of Engineering Science, Tech. Rep., 2011.

[12] J. B. Roerdink and A. Meijster, "The watershed transform: Definitions, algorithms and parallelization strategies," *Fundamenta Informaticae*, 2000.

[13] H. Digabel and C. Lantuéjoul, *Quantitative Analysis of Microstructures in Materials Sciences, Biology and Medicine*, 1978, ch. Iterative Algorithms.

[14] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Int. Conf. Comp. Vision (ICCV)*, 2003.

[15] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision*, vol. 59, no. 2, 2004.

[16] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. on Pat. Anal. and Mach. Intel.*, vol. 24, 2002.

[17] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *European Conf. on Computer Vision (ECCV)*, 2008.

[18] M.-Y. Liu, O. Tuzell, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation." in *Proc. IEEE Conf. on Comp. Vision a. Pattern Recog. (CVPR)*, 2011.

[19] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, 2011.

[20] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, 2012.