Discrete-Continuous Gradient Orientation Estimation for Faster Image Segmentation

Michael Donoser and Dieter Schmalstieg
Institute for Computer Graphics and Vision
Graz University of Technology
{donoser,schmalstieg}@icg.tugraz.at

Abstract

The state-of-the-art in image segmentation builds hierarchical segmentation structures based on analyzing local feature cues in spectral settings. Due to their impressive performance, such segmentation approaches have become building blocks in many computer vision applications. Nevertheless, the main bottlenecks are still the computationally demanding processes of local feature processing and spectral analysis. In this paper, we demonstrate that based on a discrete-continuous optimization of oriented gradient signals, we are able to provide segmentation performance competitive to state-of-the-art on BSDS 500 (even without any spectral analysis) while reducing computation time by a factor of 40 and memory demands by a factor of 10.

1. Introduction

Image segmentation is still one of the main building blocks in many different computer vision applications. In general segmentation has the goal to partition an image into a set of meaningful atomic regions. Depending on the subsequent application, one mainly distinguishes two different scenarios for segments: (a) superpixel segmentation, where one aims at partitioning the image into a large number of regions (typically hundreds), which should not straddle object boundaries, and (b) segmentations into larger perceptually meaningful regions that are frequently used for subsequent recognition tasks and in this scope are referred to as object proposals.

To be able to handle the diverging requirements of the two scenarios, the state-of-the-art in image segmentation provides results in terms of a hierarchy [23, 28, 3], where the lower levels provide accurate superpixel segments, which are merged into larger hopefully semantically meaningful segments at higher levels. Especially the remarkable Ultrametric Contour Map (UCM) [3] has led to improved performance in several diverse computer vision applications that are based on a segment hierarchy.

For example, in [27] the UCM was used for distinguishing between internal and occlusion boundaries in video streams based on an optical flow analysis. The approach significantly outperformed the state-of-the-art in this field. In [21] the UCM was exploited to learn the parameters of a Conditional Random Field (CRF), which is frequently used for image labeling. Authors showed how to learn 10^5 parameters in a global optimal manner, allowing to derive previously unclear key findings for the area of parameter learning. In [2] the problem of segmenting and recognizing objects was addressed by combining bottom-up segmentation cues from the UCM with top-down sliding window part models. Evaluation showed that the proposed semantic segmentation method especially outperforms state-of-the-art approaches on articulated objects. In [16] a random field image labeling problem was formulated over a UCM hierarchy and it is demonstrated that optimal solutions for the defined graph, that is denoted as pylon model, can still be found by standard graph cut algorithms.

To sum up, the hierarchical segments delivered by the Ultrametric Contour Map (UCM) [3] algorithm have proven to be an indispensable tool for diverse computer vision applications. In general, finding the UCM of an image mainly consists of three subsequent steps: (a) estimating local gradient orientations and magnitudes based on evaluating powerful local feature cues using advanced learning techniques, (b) analysis of the oriented gradient signals to encode global dependencies using spectral partitioning procedures and (c) greedily merging oriented watershed segments to build the final hierarchical data structure. The UCM provides excellent results, but its main bottlenecks are the time and memory demanding steps of local feature processing and spectral partitioning. As a consequence, building the segment hierarchy takes on average a few minutes and requires Gigabytes of memory even for small images, which might prevent its usage in an even broader range of computer vision applications.

Some papers tried to address these performance issues by either exploiting additionally available hardware or by approximating core parts of the algorithm. For example in [6] the runtime for calculating the required gradient magnitudes was decreased by two orders of magnitudes due to effective parallelization on powerful Graphics Processing Units (GPUs). Nevertheless, this performance boost comes at the cost of requiring a powerful GPU, that e. g. is not available on mobile platforms. In contrast, in [28] some algorithmic improvements to the UCM pipeline were proposed. The main ideas were to base the segmentation on a simple edge detection approach and to approximate the spectral analysis by solving a reduced eigensystem, which enables more efficient calculation.

In this paper, we also address the runtime issues of the UCM approach. Our idea is to locally predict oriented gradient signals by analyzing mid-level patches in a discrete-continuous setup, hence we denote our approach as *DC-Seg*. We first train a random forest classifier for predicting prototypical edge templates using efficiently calculated features, where the prototypes are obtained by clustering ground truth segmentation patches. For each prototype cluster, we afterwards independently predict the continuous gradient orientation signals. During test time, we only have to estimate the mid-level patch features and pass them to the classifier and the corresponding subsequent regressor, which provides local gradient orientation measures in a fast manner. For the last step, we stick to the original UCM idea of calculating an oriented watershed transform and greedily merging the neighbors to provide the final segmentation hierarchy.

Such an approach of exploiting edge patch prototypes was also recently used in [18] for efficiently providing feature responses that are complementary to conventional gradient histogram representations. In a first step mid-level patches of manually generated contours are clustered to form so called sketch tokens, i. e. edge patch prototypes. Then a random forest classifier is applied to predict for all patches in the test image the sketch token assignment probability maps, and the obtained responses are exploited in a state-of-the-art detection framework. Promising results on standard datasets were achieved.

The main motivation for such a discrete-continuous approach is that for identifying the edge patch prototypes, a different representation, i. e. the ground truth segmentation instead of the image content, is used. The pre-assignment step to prototypes injects more prior information into the prediction problem, and as a consequence improves prediction quality. As we show in the experiments, our approach outperforms the approach of directly regressing the oriented gradient signals.

The experimental evaluation finally shows that our approach provides segmentation performance competitive to the state-of-the-art reducing computation time in comparison to UCM by a factor of 40 and memory demands by a factor of 10, even if we do not include spectral analysis. Additionally, it is easily possible to add a spectral analysis step, which improves performance even beyond the UCM results, but still at significantly lower runtime.

2. Related Work

Image segmentation is a well-researched field in computer vision and many different methods have been proposed over the last years. We can mainly group them into (a) the superpixel field or (b) the large segment field.

In the superpixel field three different methods have proven to be most popular: the global optimal graph based segmenter [12], the normalized cut based method [24], and the mean shift algorithm [7]. Recently, several new methods addressed specific limitations of the aforementioned algorithms. For example in [9] an algorithm denoted as SEEDS was introduced. It is based on a simple and efficient hill-climbing optimization, which continuously refines superpixel boundaries. Since the hill-climbing optimization can be stopped at any time, the method can be optimally adjusted to the available computational power. In [1] a fast method called SLIC was described that allows to specify the desired number of regular, compact superpixels in the returned segmentation result. Hence, for SLICs simply the approximate number of superpixels has to be specified as the only parameter. The method proposed in [20] returns segments that are forced to conform to a regular superpixel lattice. Such an approach has the advantage, that the segmentation result maintains a regular relationship between segments yielding fixed neighbourhood relations. Subsequent algorithms may benefit from such a relationship.

In the large segment field, the most popular approach is the aforementioned Ultrametric Contour Map (UCM) algorithm. The UCM was recently adapted in [23], where the greedy merging process in the final step of the UCM was explicitly addressed. The core idea is to base the merging step on a cascade of classifiers that is trained sequentially by maximizing the boundary recall. Such an approach enables to adapt the weights of the individual features to the different scales that are analyzed. Excellent results were shown on standard benchmarks, however at the cost of increased runtime. In [15] a higher-order correlation clustering method was proposed. The method starts from a fine-grained segmentation, and afterwards partitions the corresponding graph in a single clustering step, where the number of regions has to be specified. In [5] the Maximum Weight Independent Set (MWIS) algorithm was used to infer meaningful segments from a provided ensemble of distinct, low-level segmentations of the image. By specifying reasonable weights for each segment and

for each neighboring segment pair the MWIS outputs a single, unique partition of the image. If for example applied onto the UCM results, segmentation performance is further increased, because it implicitly selects optimal segments that may come from different levels of the UCM hierarchy.

3. Method Description

The core part of our image segmentation approach which we denote as *DC-Seg* is to predict local gradients for each pixel in a test image, by applying a discrete-continuous learning pipeline to local mid-level patch descriptors. We first have to train our model using labeled training data, i. e. we assume that we have given a set of color images and corresponding ground truth segmentations (Section 3.1). Once our model is trained, we can locally predict the gradients, which are then passed to an oriented watershed transform and the results are analyzed to obtain the hierarchical segmentation (Section 3.2).

3.1. Training

We assume that we have given a set of N color images

$$\mathcal{I} = \{I_1, I_2, \ldots, I_N\} \tag{1}$$

and a corresponding set of ground truth segmentations

$$\mathcal{S} = \left\{ S_1^1, S_1^2, \ldots, S_1^{k_1}, S_2^1, \ldots S_2^{k_2}, \ldots, S_N^{k_n} \right\} \tag{2}$$

where each image I_i might have a different number k_i of segmentation ground truths S_i^j. Each segmentation ground truth can be represented as a binary image B_i^j, highlighting the contours between neighboring segments.

The main goal of our training step is to learn how to effectively and efficiently predict an oriented gradient signal $\nabla = \{\nabla_1, \nabla_2, \ldots, \nabla_C\}$ for all patches in a test image, where C is the number of discretization levels for the orientation (fixed to 8 in all our experiments). The predicted local gradients are then used to infer a hierarchical segmentation structure. Obviously the final quality of the segmentation hierarchy is strongly depending on an accurate local prediction of the gradients.

The most straight forward way for predicting the oriented gradients would be to directly learn a regressor, that maps local patch descriptors to C different gradient orientations. As we show in the experimental evaluation, such a basic approach fails to provide satisfactory results. For this reason, we propose a discrete-continuous inference process. The idea is to first train a classifier that assigns the local patch to one of a set of prototypical edge patches $\mathcal{P} = \{P_1, P_2, \ldots, P_P\}$ or to a background class P_{P+1}. Afterwards all training patches assigned to a specific class P_i are used to define a prediction function $\mathcal{R}_i$ with $i = 1, \ldots, P$ (background predicts zero magnitudes) to obtain the required continuous gradient orientations.

To sum up, our overall training process consists of the following three steps: (a) identification of edge patch prototypes $\mathcal{P}$, (b) prototype classifier training and (c) oriented gradient regressor training.

Edge Patch Prototypes As a first step we aim at identifying a set of prototypical mid-level edge patches from our training data using the provided ground truth data as representation. Note that such mid-level patch prototypes were recently in the focus of several research papers [26, 10, 18]. We randomly select a set of positive edge patches p_i of the same size $S \times S$ from all binary images B_i^j provided in our training data set $\mathcal{S}$. Positive samples here means that these patches all have a segment boundary at their center pixel, i. e. the binary value of the center pixel of the patch in B_i^j is one.

To identify the prototype set $\mathcal{P}$ in general any unsupervised clustering approach can be applied. We stick to the standard approach of applying K-Means clustering, as it is frequently done in the field of image retrieval.

As suggested in [18], to be able to handle slight shifts of the edges within the local patches, we extract a shift-invariant descriptor based on orientated gradient histograms and use the descriptor output in the K-Means clustering. In such a way, we obtain a representative set of edge patch prototypes $\mathcal{P} = \{P_1, P_2, \ldots, P_P\}$ and the unique assignment of all positive edge patches to the prototypes. This information is now used for mid-level patch classification in our second step.

Patch Classifier Training Once we have obtained our P prototypes, we train a single mid-level patch classifier

$$f : \mathbb{R}^D \rightarrow \mathcal{L} \ , \tag{3}$$

which enables to classify a D-dimensional vector $\mathbf{x}$ (our mid-level patch descriptor) to one label within our label set $\mathcal{L} = \{l_1, l_2, \ldots l_{P+1}\}$. Hence, we define $P + 1$ labels, including P labels representing all prototypes obtained in the way as described in the previous paragraph and one additional background label.

For defining the training samples, we use all patches assigned to the corresponding cluster obtained by the K-Means approach for the positive edge patches (labels $l_1, l_2, \ldots l_P$), and for the background we randomly sample non-edge patches (center pixel is zero) from our training set (label l_{P+1}). As input to the classifier we use same features as proposed in [18], that have shown to be powerful and fast due to the usage of the integral image structure. Specifically, we exploit color channel, gradient magnitude, quantized gradient channel and self-similarity features for our segmentation approach.

As classifier we apply the widely used random forests approach [4], which naturally handles multi-class

problems in a probabilistic and efficient manner and has shown to outperform related approaches in several vision applications, especially if many training samples are available as in our scenario. Random forests were successfully applied in diverse computer vision fields like human pose recognition from depth images [25], object detection [13] or tracking [17].

In general the goal of random forests is to obtain the posterior distribution $p(l|\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^D$ is the observation (in our case the local patch descriptor) and $l \in \mathcal{L}$ is the discrete class label (in our case $l \in \{l_1, l_2, \dots l_{P+1}\}$). Based on imitating the natural behavior of seeking several opinions before making any crucial decision, random forests combine the prediction of several base classifiers (decision trees) inducing randomization at several levels to produce ensembles of strongly diversified models. Each tree uniquely splits the feature space into several bins and estimating the class-specific posterior probabilities simply boils down to counting the number of class occurrences within the individual bins. We follow the standard procedure of building the forest for classification as outlined in [8]. After we have trained our random forest, we are able to predict the class assignments of a previously unseen test sample (a mid-level patch descriptor) to our classes in a probabilistic manner.

Oriented Gradient Regression For each obtained edge patch prototype cluster, we additionally define a prototype-specific prediction function

$$r^l : \mathbb{R}^D \rightarrow \mathbb{R}^C \quad , \tag{4}$$

where l is an index for the prototype cluster analyzed and C is the number of gradient orientations considered. Based on the functions r^l, we can then predict the oriented gradient signals $\nabla = \{\nabla_1, \nabla_2, \dots, \nabla_C\}$ per mid-level patch that are required for the oriented watershed step outlined in the next paragraph. The core idea here is that we want to approximate the excellent gPb-performance [19], which is considered as state-of-the-art gradient detector. For this reason, we apply the gPb detector to all images within our training dataset $\mathcal{I}$. According to the randomly selected set of positive mid-level training patches, we consider the C gradient orientations of the patch center pixel, as provided by the gPb algorithm, as the ground truth data for training.

Since we have assigned each patch to one of our $P+1$ labels in the previous step, we have to define a regression function r^l independently for all positive classes assuming that for the background class the gradient magnitudes are all zero, i. e. in overall we have to find P independent functions r^l. We again describe the patches by the same features as outlined before. For predicting the orientation signals, we stick to a simple approach due to efficiency reasons. We represent each prototype cluster by the mean gradient orientation responses over all training samples assigned to the corresponding cluster denoted as $\overline{\nabla}^l = \left\{\overline{\nabla}^l_1, \overline{\nabla}^l_2, \dots, \overline{\nabla}^l_C\right\}$. For prediction we then use the probabilistic assignment values $p(l|\mathbf{x})$ to weight the corresponding mean gradient responses per cluster, which allows to predict the local gradient orientations for each mid-level patch within a test image. The predicted values are finally used in an oriented watershed based segmentation step, which is outlined in the next section.

3.2. Image Segmentation

After training our classifier f and the corresponding functions r^l per prototype as described in the previous section, we are now able to predict the local gradient orientation densely for a previously unseen test image in an efficient manner. We first densely calculate local features, which yields a D-dimensional vector $\mathbf{x} \in \mathbb{R}^D$ per pixel. These vectors are passed to the generic classifier f, which returns the class-specific posterior probabilities $p(l_i|\mathbf{x})$.

We then use a weighted combination of the average prototype gradient orientations for prediction by

$$\nabla = p(l_i|\mathbf{x})\ \overline{\nabla}^{l_i} \quad . \tag{5}$$

As a byproduct, the classification process allows to predict the probability e of having an edge at the center pixel of the mid-level patch by

$$e = \sum_{i=1\dots P} p(l_i|\mathbf{x}) = 1 - p(l_{P+1}|\mathbf{x}) \quad , \tag{6}$$

where the label $P+1$ again represents our background class. We use the estimated edge probabilities e to reweight the local gradient orientation signals for every pixel in the image. Note that the entire prediction pipeline is quite fast, since we use efficient features in combination with a very efficient classifier and prediction step.

The final step of our method is the same as in the Ultrametric Contour Map (UCM) [3] algorithm. We use the predicted local gradient orientations and calculate an over-segmentation by applying an oriented watershed segmentation, where the obtained segments define the highest level of detail considered, i. e. the leaf nodes of the final segmentation hierarchy. The regions of the finest level-of-detail are subsequently merged using a greedy strategy, which yields the segment hierarchy we are interested in. For more details on this step see [3].

Please note that in our segmentation pipeline, we completely omit any global optimization process, for example based on spectral clustering, as it is done in state-of-the-art approaches [3, 28]. Such a global optimization takes a long time, and this was the reason that e. g. in [28] a more efficient spectral analysis approach was proposed. As we show in the experiments, we achieve

quite competitive performance, even if we do not apply any global optimization. Nevertheless, any of the spectral approaches is applicable in our setup, since they are only an additionally applied analysis step. We show in the experiments that by including the spectral analysis we can slightly improve results, however at least at the cost of doubling the runtime.

4. Experiments

We evaluate our proposed segmentation method denoted as *DC-Seg* on the popular Berkeley segmentation dataset BSDS 500 [3]. This dataset consists of 500 color images, split into 200 training, 100 validation and 200 test images. We trained our local oriented gradient estimator as described in Section 3.1 on the 200 training images and used the validation set for parameter tuning. We fixed the patch size to $S = 31$ which leads to an 17 654 dimensional feature vector, built a codebook of $P = 150$ mid-level patch prototypes and considered $C = 8$ different gradient orientations, equally sampled in the interval $[0, \pi)$. For learning we used the random forest framework provided in the toolbox of Piotr Dollar [11]. For the classifier we trained 25 trees using 1000 positive samples per prototype class and the same number of negative samples per tree. The trees are all fully grown until the number of samples is below 5. We implemented our method in Matlab extending the publicly available Sketch Tokens code of [18]. Code is provided online at `http://vh.icg.tugraz.at`.

We compare our method to the state-of-the-art in image segmentation using the standard measures of segment coverage (*Cover* – per-pixel segment overlap, higher is better), Probabilistic Rand Index (*PRI* – pairwise compatibility of pixel assignments, higher is better) and Variation of Information (*VoI* – relative entropy, lower is better). More details on these measures can be found in [3]. The provided segmentation hierarchy allows to extract segmentation results of differing granularity by accessing specific levels of the hierarchy. Hence, we can derive two independent scores by optimizing the scale in two ways: (a) select one scale over the entire dataset (ODS) or selecting an individual, optimal scale per image (OIS).

We compare our method to the following state-of-the-art approaches: the Ultrametric Contour Map and its adapted version based on Canny edges (*ucm* and *canny-ucm*) [3], the reduced order spectral approach (*red-spec*) [28], the agglomerative merging segmentation algorithm (*agg-mer*) [14], the mean-shift algorithm (*mean-shift*) [7], the global optimal graph based segmenter (*graph-seg*) [12] and the normalized cut based method (*norm-cut*) [24]. For comparison, we first list the scores if we directly apply our proposed method as described in Section 3 (*DC-Seg*), which neglects a subsequent spectral analysis and as a consequence is the fastest.

BSDS 500	Cover		PRI		VoI	
	ODS	OIS	ODS	OIS	ODS	OIS
Human	0.72	0.72	0.88	0.88	1.17	1.17
ucm [3]	**0.59**	0.63	0.81	**0.85**	1.70	**1.52**
red-spec [28]	0.56	0.62	0.81	**0.85**	1.78	1.56
agg-mer [14]	0.56	0.60	0.81	0.84	1.78	1.66
mean-shift [7]	0.54	0.58	0.79	0.81	1.85	1.64
graph-seg [12]	0.52	0.57	0.80	0.82	2.21	1.87
norm-cut [24]	0.45	0.53	0.78	0.80	2.23	1.89
canny-ucm [3]	0.49	0.55	0.79	0.83	2.19	1.89
DC-Seg	0.58	0.63	**0.82**	**0.85**	1.75	1.59
DC-Seg-appr	**0.59**	**0.64**	**0.82**	**0.85**	1.69	**1.52**
DC-Seg-full	**0.59**	**0.64**	**0.82**	**0.85**	**1.68**	1.54

Table 1: Evaluation of image segmentation methods on the Berkeley segmentation dataset BSDS 500. Scores if selecting the optimal dataset scale (ODS) or the optimal image scale (OIS) are shown. To measure the quality of the obtained segmentation results, the coverage (Cover), the Probabilistic Rand Index (PRI) and the Variation of Information (VoI) is used. Best results are shown in bold. The corresponding runtimes are given in Table 2

Additionally, we also considered two global spectral analysis adaptations of our proposed approach using (a) the original spectral approach used in the UCM [3] and (b) the recently proposed reduced order spectral analysis [28] (*DC-Seg-full* and *DC-Seg-appr*). Such spectral extensions can be easily integrated in our framework, since they independently also predict oriented gradient signals. We simply combine our predicted gradients and the spectral predictions by a weighted linear combination defined as

$$\nabla = \alpha \, \nabla + (1 - \alpha) \, \nabla_{spec} \ , \tag{7}$$

where ∇_{spec} are the estimates obtained by the corresponding spectral analysis.

Table 1 follows the main evaluation protocol of BSDS 500 by summarizing all results by rounding the values to two digits. As can be seen, our proposed approach (*DC-Seg*) yields results that are competitive to the much more computationally demanding UCM approach. By incorporating a spectral analysis, we are even able to outperform UCM, still at a reduced runtime, since our approach applies a much more efficient feature extraction and learning pipeline.

Table 2 directly compares our approaches to the UCM in more detail, listing scores up to 4 digits and additionally the average runtime required for segmenting an image, evaluated on a desktop PC with 3.6 GHz quad-core Intel Core i7 processor. As can be seen, our proposed method without any spectral extension provides competitive performance, while reducing the runtime by a factor

	Time (s)	Cover	PRI	VoI
ucm	243.14	0.5865	0.8149	1.6953
DC-Seg	**5.90**	0.5834	0.8209	1.7494
DC-Seg-appr	13.34	**0.5939**	**0.8223**	1.6897
DC-Seg-full	143.83	0.5937	0.8218	**1.6834**

Table 2: Analysis of average runtime required for segmenting an image and performance on the BSDS 500 dataset using the ODS measure. Best results are shown in bold. As can be seen, even without a subsequent spectral analysis (*DC-Seg*) we achieve competitive performance at a reduced runtime of a factor of 40.

	ODS	OIS	Area-PR
ucm [3]	0.73	**0.76**	0.73
Sketch Tokens [18]	0.73	0.75	**0.78**
SCG [22]	**0.74**	**0.76**	0.77
DC-Seg	0.73	**0.76**	0.76

Table 3: Contour detection performance evaluation on BSDS 500. Best results are shown in bold. Comparison to three state-of-the-art approaches.

of 40 compared to the UCM algorithm. Furthermore, as expected adding the spectral analysis improves the performance, nevertheless comes at the cost of an increased runtime, for example doubling it if using the reduced order approximation. Our result also reassures the effectiveness of the reduced order spectral scheme proposed in [28], which provides similar results to the full spectral approach used in the UCM, but at significantly reduced runtime. Such a comparison on the entire Berkeley dataset was missing in [28]. Considering the memory profile, the UCM requires on average 2 Gigabytes of memory, whereas our method only uses 0.2 Gigabytes, reducing memory demands by a factor of 10. As a consequence, our approach enables the segmentation of megapixel images in a hierarchical manner on standard desktop PCs.

The resulting hierarchical segmentation result implicitly also defines a contour detection result for each test image. Hence, in Table 3, we give a comparison to the state-of-the-art in contour detection such as the sparse contour gradients [22].

An additional question that has to be answered is if the proposed discrete-continuous pipeline is a reasonable way to predict the oriented gradient signal for image segmentation. To answer this, we made two additional experiments. First, we tested the performance if training a regressor that directly predicts the local gradients, skipping our intermediate classification step. The direct regressor leads to a considerably worse performance (*Cover*: 0.54 vs. 0.58, *PRI*: 0.78 vs. 0.82, *VoI*: 2.01 vs. 1.75). This might be explained by the fact that for obtaining our prototypes, we use a representation base (the ground truth segments) that differs from the direct image content. Thus, we are able to inject more prior knowledge into the prediction process and as a consequence improve results. Second, we replaced our prediction by the gradient detector proposed in [22], which is the currently best-performing gradient detection method on the Berkeley dataset. This method is based on discriminatively trained sparse contour gradients, hence exploits the recently popular concept of sparse coding of patches. The method implicitly estimates local gradient orientations which allows to directly replace our discrete-continuous estimations with the one of [22]. We again get considerably worse performance (*Cover*: 0.49 vs. 0.58, *PRI*: 0.79 vs. 0.82, *VoI*: 2.33 vs. 1.75), which might be explained by the circumstance that our approach aims at directly predicting oriented gradients, that are well suited for the subsequent oriented watershed analysis. The oriented gradients predicted by [22] represent the patch by a sparse coding of prototypes, which seems to be not as suited for the subsequent segmentation analysis.

The scores analyzed do not allow a deeper insight into performance differences between segmentation results obtained with different methods, since the correspondences between the calculated scores and the input images are lost. In order to overcome this weakness, we compare our approach directly to the Ultrametric Contour Map (UCM) using scatter plots, as shown in Figure 1. Each point in this plots represents one of the 200 test images, where the x-value is the coverage score of our method, and the y-value is the coverage score of the UCM, both evaluated at the optimal image scale. Points on the diagonal line $y = x$ indicate identical deviation from the ground truth for both algorithms, while the distance to the line is a measure for the disagreement between the obtained segmentation approaches. Points above the diagonal line represent images, where our algorithm performs better, whereas points below indicate superior performance of the UCM. Such plots allow to directly analyze on which images performance differences are significant and additionally to identify easy-to-segment images according to their location along the main diagonal.

As can be seen, for 78 out of 200 images our non-spectral approach even provides better scores compared to the UCM. In overall on all images our returned segmentation result is in close range to the UCM, hence demonstrating that our much faster algorithm is indeed able to provide competitive results on all test images. This can also be seen by the four images where the scores diverge the most, which are highlighted in Figure 1. But even in these cases, the segmentations results are quite reasonable. Figure 2

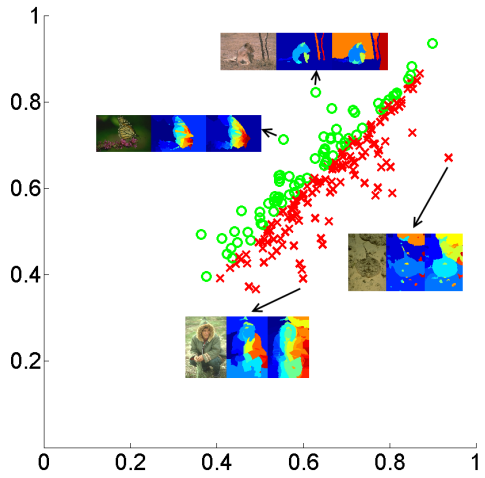

Figure 1: Comparison of segmentation performance using coverage scatter plots (UCM vs. proposed). Points above the diagonal represent images, where our method performs better. This discrimination is emphasized by different colors. Extreme images, where the scores diverge most, are included (left: image, middle: our result, right: UCM result).

furthermore shows some exemplary segmentation results comparing our proposed method to the UCM, where results are obtained at the optimal database level. As can be seen, results are also visually quite comparable, despite the fact that our approach is 40 times faster.

5. Conclusion

In this paper, we introduced a novel approach denoted as *DC-Seg* for providing a hierarchical segmentation result in short computation time. The core idea was to address computational complexity issues of the widely used Ultrametric Contour Map (UCM). We proposed a discrete-continuous approach for predicting oriented gradients using powerful but fast to calculate features, which are afterwards analyzed by an oriented watershed segmentation step that provides the final segmentation hierarchy. As we demonstrate in the experiments, our method achieves competitive segmentation performance, even if fully neglecting any subsequent global spectral analysis step while reducing computation time by a factor of 40 and memory demands by a factor of 10. Adding spectral analysis further boosts the performance even beyond UCM, but at least doubles the runtime. Implementing our approach on a powerful GPU might lead to a real-time capable segmentation tool, where for example videos might be segmented online, while the user can adapt the desired granularity on-the-fly.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 601139 CultAR (Culturally Enhanced Augmented Realities).

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 32:2274–2282, 2012.

[2] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. D. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *Proc. of Conf. on Comp. Vision and Pattern Rec. (CVPR)*, 2012.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 33(5), 2010.

[4] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[5] W. Brendel and S. Todorovic. Segmentation as maximum-weight independent set. In *Advances in Neural Information Proc. of Systems (NIPS)*, pages 307–315, 2010.

[6] B. Catanzaro, B. Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer. Efficient, high-quality image contour detection. In *Proc. of IEEE Intl. Conf. on Comp. Vision (ICCV)*, 2009.

[7] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 24(5):603–619, 2002.

[8] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7:81–27, 2012.

[9] M. V. den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool. Seeds: Superpixels extracted via energy-driven sampling. In *Proc. of European Conf. on Comp. Vision (ECCV)*, pages 13–26, 2012.

[10] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes paris look like paris? *ACM Transactions on Graphics (SIGGRAPH)*, 31(4), 2012.

[11] P. Dollár. Piotr's Image and Video Matlab Toolbox (PMT). http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html.

[12] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Intl. Journal of Comp. Vision (IJCV)*, 59(2):167–181, 2004.

[13] J. Gall and V. Lempitsky. Class-specific Hough forests for object detection. In *Proc. of Conf. on Comp. Vision and Pattern Rec. (CVPR)*, 2009.

[14] D. Hoiem, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. *Intl. Journal of Comp. Vision (IJCV)*, 91:328–346, 2011.

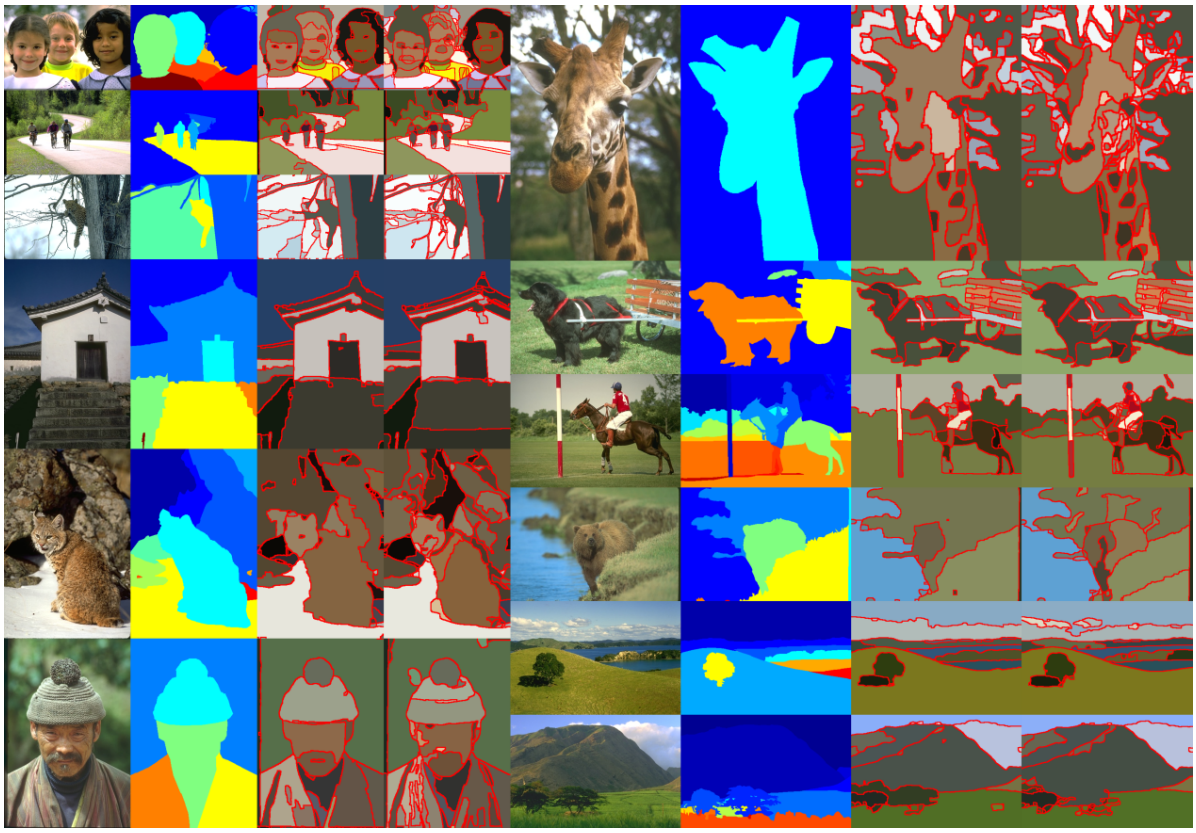

Figure 2: Exemplary segmentation results on Berkeley BSDS 500 obtained on the optimal database scale. First: input image, second: ground truth, third: our result, fourth: UCM result. The obtained segments are mapped to their mean RGB value and boundaries between segments are highlighted, to be able to see if the main image details are preserved.

[15] S. Kim, S. Nowozin, P. Kohli, and C. D. Yoo. Higher-order correlation clustering for image segmentation. In *Advances in Neural Information Proc. of Systems (NIPS)*, 2011.

[16] V. Lempitsky, A. Vedaldi, and A. Zisserman. A pylon model for semantic segmentation. In *Advances in Neural Information Proc. of Systems (NIPS)*, 2011.

[17] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proc. of Conf. on Comp. Vision and Pattern Rec. (CVPR)*, pages 775–781, 2005.

[18] J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *Proc. of Conf. on Comp. Vision and Pattern Rec. (CVPR)*, 2013.

[19] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *Proc. of Conf. on Comp. Vision and Pattern Rec. (CVPR)*, 2008.

[20] A. P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *Proc. of Conf. on Comp. Vision and Pattern Rec. (CVPR)*, 2008.

[21] S. Nowozin, P. V. Gehler, and C. H. Lampert. On parameter learning in CRF-based approaches to object class image segmentation. In *Proc. of European Conf. on Comp. Vision (ECCV)*, 2010.

[22] X. Ren and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *Advances in Neural Information Proc. of Systems (NIPS)*, pages 593–601, 2012.

[23] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *Proc. of Conf. on Comp. Vision and Pattern Rec. (CVPR)*, 2013.

[24] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 22(8):888–905, 2000.

[25] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proc. of Conf. on Comp. Vision and Pattern Rec. (CVPR)*, 2011.

[26] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *Proc. of European Conf. on Comp. Vision (ECCV)*, 2012.

[27] A. Stein and M. Hebert. Occlusion boundaries from motion: Low-level detection and mid-level reasoning. *Intl. Journal of Comp. Vision (IJCV)*, 82:325–357, 2009.

[28] C. J. Taylor. Toward fast and accurate segmentation. In *Proc. of Conf. on Comp. Vision and Pattern Rec. (CVPR)*, 2013.