Real-Time Semantic Segmentation with Label Propagation

Rasha Sheikh, Martin Garbade(✉), and Juergen Gall

Computer Science Institute III, University of Bonn, Bonn, Germany
rasha@uni-bonn.de, {garbade,gall}@iai.uni-bonn.de

Abstract. Despite of the success of convolutional neural networks for semantic image segmentation, CNNs cannot be used for many applications due to limited computational resources. Even efficient approaches based on random forests are not efficient enough for real-time performance in some cases. In this work, we propose an approach based on superpixels and label propagation that reduces the runtime of a random forest approach by factor 192 while increasing the segmentation accuracy.

1 Introduction

Although convolutional neural networks have shown a great success for semantic image segmentation in the last years [1–3], fast inference can only be achieved by massive parallelism as offered by modern GPUs. For many applications like mobile platforms or unmanned aerial vehicles, however, the power consumption matters and GPUs are often not available. A server-client solution is not always an option due to latency and limited bandwidth. There is therefore a need for very efficient approaches that segment images in real-time on single-threaded architectures.

In this work, we analyze in-depth how design choices affect the accuracy and runtime of random forests and propose an efficient superpixel-based approach with label propagation for videos. As illustrated in Fig. 1, we use a very efficient quadtree representation for superpixels. The superpixels are then classified by random forests. For classification, we investigate two methods. For the first method, we use the empirical class distribution and for the second method we model the spatial distributions of class labels by Gaussians. For video data, we propose label propagation to reduce the runtime without substantially decreasing the segmentation accuracy. An additional spatial smoothing even improves the accuracy.

We evaluate our approach on the CamVid dataset [4]. Compared to a standard random forest, we reduce the runtime by factor 192 while increasing the global pixel accuracy by 4 % points. A comparison with state-of-the-art approaches in terms of accuracy shows that the accuracy of our approach is competitive while achieving real-time performance on a single-threaded architecture.

G. Hua and H. Jégou (Eds.): ECCV 2016 Workshops, Part II, LNCS 9914, pp. 3–14, 2016.
DOI: 10.1007/978-3-319-48881-3_1

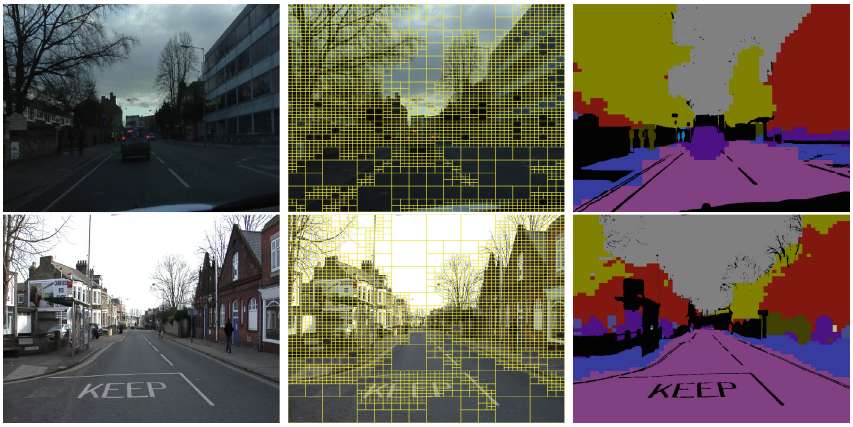

Fig. 1. For efficient segmentation, we use a quadtree to create superpixels and classify the superpixels by a random forests.

2 Related Work

A popular approach for semantic segmentation uses a variety of features like appearance, depth, or edges and classifies each pixel by a classifier like random forest or boosting [4,5]. Since pixel-wise classification can be very noisy, conditional random fields have been used to model the spatial relations of pixels and obtain a smooth segmentation [6,7]. Conditional random fields, however, are too expensive for many applications. In [8], a structured random forest has been proposed that predicts not a single label per pixel but the labels of the entire neighborhood. Merging the predicted neighborhoods into a single semantic segmentation of an image, however, is also costly. To speed up the segmentation, the learning and prediction of random forests has been also implemented for GPUs [9].

In the last years, convolutional neural networks have become very popular for semantic segmentation [1,2,10]. Recent approaches achieve accurate segmentation results even without CRFs [3]. They, however, require GPUs for fast inference and are too slow for single-threaded architectures. Approaches that combine random forests and neural networks have been proposed as well [8], however, at the cost of increasing the runtime compared to random forests.

Instead of segmenting each frame, segmentation labels can also be propagated to the next frame. Grundmann et al. [11] for example use a hierarchical graph-based algorithm to segment video sequences into spatiotemporal regions. A more advanced approach [12] proposes a label propagation algorithm using a variational EM based inference strategy. More recently, a fast label propagation method based on sparse feature tracking has been proposed [13]. Although our method can be used in combination with any real-time label propagation method like [13], we use a very simple approach that propagates the labels of quadtree superpixels, which have the same location and similar appearance as in the preceding frame.

3 Semantic Segmentation

We briefly describe a standard random forests for semantic image segmentation in Sect. 3.1. In Sect. 3.2, we propose a superpixel approach that can be combined with label propagation in the context of videos.

3.1 Random Forests

Random forests consists of an ensemble of trees [14]. In the context of semantic image segmentation, each tree infers for an image pixel $\mathbf{x}$ the class probability $p(c|\mathbf{x};\theta_t)$ where c is a semantic class and θ_t are the parameters of the tree t. The parameters θ_t are learned in a suboptimal fashion by sampling from the training data and the parameter space Θ. A robust estimator is then obtained by averaging the predictors

$$p(c|\mathbf{x}) = \frac{1}{T}\sum_t p(c|\mathbf{x};\theta_t), \tag{1}$$

where T is the number of trees in the forest. A segmentation of an image can then be obtained by taking the class with highest probability (1) for each pixel.

Learning the parameters θ_t for a tree t is straightforward. First, pixels from the training data are sampled which provide a set of training pairs $\mathcal{S} = \{(\mathbf{x}, c)\}$. The tree is then constructed recursively, where at each node n a weak classifier is learned by maximizing the information gain

$$\theta_n = \underset{\theta\in\tilde{\Theta}}{\operatorname{argmax}} \left\{ H(\mathcal{S}_n) - \sum_{i\in\{0,1\}} \frac{|\mathcal{S}_{n,i}|}{|\mathcal{S}_n|} H(\mathcal{S}_{n,i}) \right\}. \tag{2}$$

While $\mathcal{S}_n$ denotes the training data arriving at the node n, $\tilde{\Theta}$ denotes the set of sampled parameters and $H(\mathcal{S}) = -\sum_c p(c;\mathcal{S}) \log p(c;\mathcal{S})$ where $p(c;\mathcal{S})$ is the empirical class distribution in the set $\mathcal{S}$. Each weak classifier $f_\theta(\mathbf{x})$ with parameter θ splits $\mathcal{S}_n$ into the two sets $\mathcal{S}_{n,i} = \{(\mathbf{x}, c) \in \mathcal{S}_n : f_\theta(\mathbf{x}) = i\}$ with $i \in \{0, 1\}$. After the best weak classifier θ_n is determined, $\mathcal{S}_{n,0}$ and $\mathcal{S}_{n,1}$ is forwarded to the left or right child, respectively. The growing of the tree is terminated when a node becomes pure or $\mathcal{S}_n < 100$ (found using cross-validation). Finally, the empirical class distribution $p(c;\mathcal{S}_l)$ is stored at each leaf node l.

As weak classifiers $f_\theta(\mathbf{x})$, we use four types that were proposed in [5]:

$$R(\mathbf{x}+\mathbf{x}_1, w_1, h_1, k) - R(\mathbf{x}+\mathbf{x}_2, w_2, h_2, k) \le \tau \tag{3}$$

$$R(\mathbf{x}+\mathbf{x}_1, w_1, h_1, k) + R(\mathbf{x}+\mathbf{x}_2, w_2, h_2, k) \le \tau \tag{4}$$

$$|R(\mathbf{x}+\mathbf{x}_1, w_1, h_1, k) - R(\mathbf{x}+\mathbf{x}_2, w_2, h_2, k)| \le \tau \tag{5}$$

$$R(\mathbf{x}+\mathbf{x}_1, w_1, h_1, k) \le \tau. \tag{6}$$

The term $R(\mathbf{x}+\mathbf{x}_1, w_1, h_1, k)$ denotes the average value of feature channel k in the rectangle region centered at $\mathbf{x}+\mathbf{x}_1$ with $\mathbf{x}_1 \in [-100, \ldots, 100]$, width

$w_1 \in [1, \ldots, 24]$, and height $h_1 \in [1, \ldots, 24]$. As feature channels, we use the CIELab color space and the x- and y-gradients extracted by a Sobel filter. To generate $\tilde{\Theta}$, we randomly sample 500 weak classifiers without τ and for each sampled weak classifier we sample τ 20 times, i.e., $\tilde{\Theta}$ consists of 10,000 randomly sampled weak classifiers.

3.2 Superpixels with Label Propagation

A single tree as described in Sect. 3.1 requires on a modern single-threaded architecture 1500 ms for segmenting an image with 960×720 resolution. This is insufficient for real-time applications and we therefore propose to classify superpixels. In order to keep the overhead by computing superpixels as small as possible, we use an efficient quadtree structure. As shown in Fig. 1, the regions are not quadratic but have the same aspect ratio as the original image. Up to depth 3, we divide all cells. For deeper quadtrees, we divide a cell into four cells if the variance of the intensity, which is in the range of 0 and 255, within a cell is larger than 49. Instead of classifying each pixel in the image, we classify the center of each superpixel and assign the predicted class to all pixels in the superpixel. For training, we sample 1000 superpixels per training image and assign the class label that occurs most frequently in the superpixel.

While (1) uses the empirical class distribution $p(c; \mathcal{S}_l)$ stored in the leaves for classification, it discards the spatial distribution of the class labels within and between the superpixels ending in a single leaf. Instead of reducing the pixel-wise labels of the training data to a single label per superpixel, we model the spatial distribution by a Gaussian per class. To this end, we use the pixel-wise annotations of the superpixels ending in a leaf denoted by $\mathcal{S}_l = \{(\mathbf{x}_l, c_l)\}$. From all pixels $\mathbf{x}_l$ with class label $c_l = c$, we estimate a spatial Gaussian distribution $\mathcal{N}(\mathbf{y}; \mu_{c,l}, \Sigma_{c,l})$ where $\mathbf{y}$ is a location in the image and $\mu_{c,l}, \Sigma_{c,l}$ are the mean and the covariance of the class specific Gaussian. In our implementation, $\Sigma_{c,l}$ is simplified to a diagonal matrix to reduce runtime.

For inference, we convert a superpixel with width w, height h, and centered at $\mathbf{x}$ also into a Gaussian distribution $\mathcal{N}(\mathbf{y}; \mu_\mathbf{x}, \Sigma_\mathbf{x})$ where $\mu_\mathbf{x} = \mathbf{x}$ and $\Sigma_\mathbf{x}$ is a diagonal matrix with diagonal $((\frac{w}{2})^2, (\frac{h}{2})^2)$. The class probability for a single tree and a superpixel ending in leaf l is then given by the integral

$$p(c|\mathbf{x}; \theta_t) = \int \mathcal{N}(\mathbf{y}; \mu_{c,l}, \Sigma_{c,l}) \mathcal{N}(\mathbf{y}; \mu_\mathbf{x}, \Sigma_\mathbf{x}) d\mathbf{y} = \mathcal{N}(\mu_\mathbf{x}; \mu_{c,l}, \Sigma_{c,l} + \Sigma_\mathbf{x}) \quad (7)$$

$$\propto \exp\left(-\frac{1}{2} (\mu_\mathbf{x} - \mu_{c,l})^T (\Sigma_{c,l} + \Sigma_\mathbf{x})^{-1} (\mu_\mathbf{x} - \mu_{c,l})\right). \quad (8)$$

In our implementation, we omit the normalization constant and use (8). Several trees are combined as in (1). Instead of using only one Gaussian per class, a mixture of Gaussians can be used as well.

The accuracy can be further improved by smoothing. Let $N_\mathbf{x}$ be the neighboring superpixels of $\mathbf{x}$ including $\mathbf{x}$ itself. The class probability for the superpixel $\mathbf{x}$ is then estimated by

$$p(c|\mathbf{x}) = \frac{1}{|N_\mathbf{x}|} \sum_{\mathbf{y} \in N_\mathbf{x}} p(c|\mathbf{y}). \tag{9}$$

To reduce the runtime for videos, the inferred class for a superpixel can be propagated to the next frame. We propagate the label of a cell in the quadtree to the next frame, if the location and size does not change and if the mean intensity of the pixels in the cell does not change by more than 5. Otherwise, we classify the cell by the random forest.

4 Experiments

For the experimental evaluation, we use the CamVid dataset [4]. The images in this dataset have a resolution of 960 × 720 pixels. The CamVid dataset consists of 468 training images and 233 test images taken from video sequences. There is one sequence where frames are extracted at 15 Hz and 30 Hz and both are included in the training set. Most approaches discard the frames that were extracted at 15 Hz resulting in 367 training images. We report results for both settings. The dataset is annotated by 32 semantic classes, but most works use only 11 classes for evaluation, namely *road, building, sky, tree, sidewalk, car, column pole, fence, pedestrian, bicyclist, sign symbol*. We stick to the 11 class protocol and report the global pixel accuracy and the average class accuracy [3]. The runtime is measured on a CPU with 3.3 GHz single-threaded.

Our implementation is based on the publicly available CURFIL library [9], which provides a GPU and CPU version for random forests. As baseline, we use a random forest as described in Sect. 3.1. In Table 1, we report the accuracy and runtime for a single tree. The baseline denoted by *pixel stride 1* requires around 1500 ms for an image, which is insufficient for real-time applications. The runtime can be reduced by downsampling the image or classifying only a subset of pixels and interpolation. We achieved the best trade-off between accuracy and runtime for a stride of 15 pixels in x and y-direction. The final segmentation is then obtained by nearest-neighbor interpolation. Larger strides decreased the accuracy substantially. While this reduces the runtime by factor 5.6 without reducing the accuracy, the approach requires still 280 ms.

We now evaluate the superpixel based approach proposed in Sect. 3.2. We first evaluate superpixel classification based on the empirical class distribution $p(c; \mathcal{S}_l)$, which is denoted by *sp*. Compared to the baseline the runtime is reduced by factor 56 and compared to interpolation by factor 10 without reducing the accuracy. The proposed approach achieves real-time performance with a runtime of only 27.5 ms. Due to the efficient quadtree structure the computational overhead of computing the superpixels is only 2 ms.

In the following, we evaluate a few design choices. Converting an RGB image into the CIELab color space takes 1 ms. The comparison of *sp* (CIELab) with *sp - RGB* (RGB) in Table 1, however, reveals that the RGB color space degrades the accuracy substantially. We also investigated what happens if the number of parameters of the weak classifiers $f_\theta(\mathbf{x})$ (3)–(6) are reduced by setting $\mathbf{x}_1 = 0$,

Table 1. Results for one tree trained on all 468 training images. The last 4 rows report the results when only the sequences recorded with 30 Hz are used for training (367).

	Global pixel accuracy	Average class accuracy	Average time (ms)
pixel stride 1	63.54	40.61	1549
pixel stride 15	64.92	41.21	277
superpixel (sp)	65.11	41.13	27.50
sp - RGB	62.25	36.42	26.47
sp - fix region (sp-fr)	65.29	42.48	28.11
sp - 1 Gaussian	65.16	41.36	29.64
sp - 2 Gaussians	66.41	42.29	26.58
sp-fr - 2 Gaussians (sp-fr-Gauss2)	67.13	43.19	26.25
sp-fr-Gauss2 + smoothing	75.76	47.93	45.24
sp-fr-Gauss2 + propagate	66.49	42.88	18.24
sp-fr-Gauss2 + sm. + prop.	75.03	47.45	37.10
sp-fr-Gauss2 (367 images)	67.06	43.47	23.26
sp-fr-Gauss2 + smoothing (367 images)	74.80	48.00	41.71
sp-fr-Gauss2 + propagate (367 images)	67.00	43.21	15.89
sp-fr-Gauss2 + sm. + prop. (367 images)	74.67	47.19	34.50

which is denoted by *sp-fr*. It slightly increases the average class accuracy compared to *sp* since one region R is fixed to the pixel location which improves the accuracy for small semantic regions. Small regions, however, have a low impact on the global pixel accuracy. If we use (8) instead of the empirical class distribution to classify a superpixel, denoted by *sp - 1 Gaussian*, the accuracy does not improve but the runtime increases by 2 ms. If we use two Gaussians per class, one for the left side of the image and one for the right side, the accuracy increases slightly. Note that the runtime even decreases since (8) becomes more often zero for *2 Gaussians* than for *1 Gaussian*.

For the further experiments, we use the superpixel classification with fixed region and two Gaussians, denoted by *sp-fr-Gauss2*. As mentioned in Sect. 3.2 the superpixel classification can be improved by spatial smoothing, which is denoted by *smoothing*. This increases the accuracy substantially but also the runtime to 45 ms. The label propagation on the contrary reduces the runtime to 18 ms without a substantial decrease in accuracy. The smoothing can also be combined with label propagation. This gives nearly the same accuracy as *sp-fr-Gauss2 + smoothing*, but the runtime is with 37 ms lower.

If we use only the 367 images sampled at 30 Hz instead of all 468 images for training, the accuracy is the same but the runtime is reduced by around 3 ms. Since the larger set is based on sampling one sequence twice at 15 Hz and 30 Hz,

the larger set does not contain additional information and the accuracy therefore remains the same. The additional training data, however, increases the depth of the trees and thus the runtime. The classification without feature computation takes around 4 ms for a tree of depth 20 and 8–10 ms for a tree of depth 100. For 1000 superpixels sampled from each of the 468 training images, the trees can reach a depth of 100.

Table 2. Results for 10 trees trained on all 468 training images. The last 4 rows report the results when only the sequences recorded with 30 Hz are used for training (367).

	Global pixel accuracy	Average class accuracy	Average time (ms)
pixel stride 1	74.60	48.56	11288
pixel stride 15	74.58	48.69	301.7
CCF features	71.68	51.19	28476
sp-fr-Gauss2	77.49	51.29	105.3
sp-fr-Gauss2 + smoothing	79.62	51.77	131.5
sp-fr-Gauss2 + propagate	76.62	49.99	40.19
sp-fr-Gauss2 + sm. + prop.	78.56	50.79	58.75
sp-fr-Gauss2 (367 images)	77.43	51.22	102.5
sp-fr-Gauss2 + smoothing (367 images)	79.99	52.20	111.5
sp-fr-Gauss2 + propagate (367 images)	76.82	50.48	36.48
sp-fr-Gauss2 + sm. + prop. (367 images)	79.30	51.68	55.47

In Table 2, we report the accuracy and runtime for 10 trees. Increasing the number of trees from one to ten increases the global pixel accuracy of the baseline by 11 % points and the average class accuracy by 8 % points. We also evaluated the use of convolutional channel features (CCF) [15] which are obtained by the VGG-16 network [16] trained on the ImageNet (ILSVRC-2012) dataset. As in [17], the features are combined with axis-aligned split functions to build weak classifiers. Without finetuning the features do not perform better on this dataset. The extraction of CCF features is furthermore very expensive without a GPU. Similar to the baseline, the global pixel accuracy and average class accuracy is also increased for *sp-fr-Gauss2* by 10 and 8 percentage points, respectively. Only if spatial smoothing is added the increase is only 4 % points, but it still improves the accuracy. The runtime increases by factor 4, 2.9, 2.2, 1.6 for *sp-fr-Gauss2*, *sp-fr-Gauss2 + smoothing*, *sp-fr-Gauss2 + propagate*, *sp-fr-Gauss2 + sm. + prop.*, respectively. Compared to the baseline *pixel stride 1*, the runtime is reduced by factor 192 while increasing the accuracy if label propagation and smoothing are used. Figure 2 plots the accuracy and runtime of *sp-fr-Gauss2 + propagate* and *sp-fr-Gauss2 + sm. + prop.* while varying the number of trees.

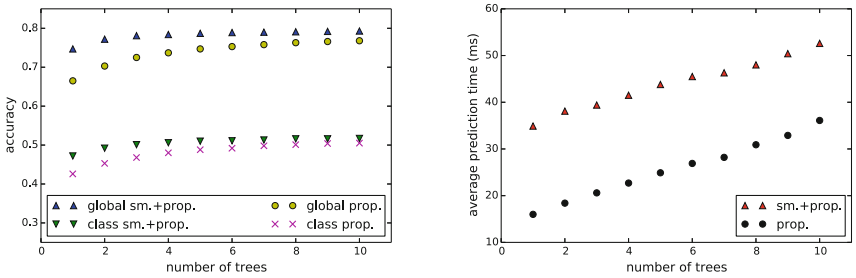

Fig. 2. Accuracy and average prediction time with respect to the number of trees.

The impact of the depth of the quadtree is shown in Fig. 3. The accuracy but also the runtime increases with the depth of the quadtree since the cells get smaller the deeper the trees are. Limiting the depth of the quadtrees to seven gives a good trade-off between accuracy and runtime. This setting is also used in our experiments.

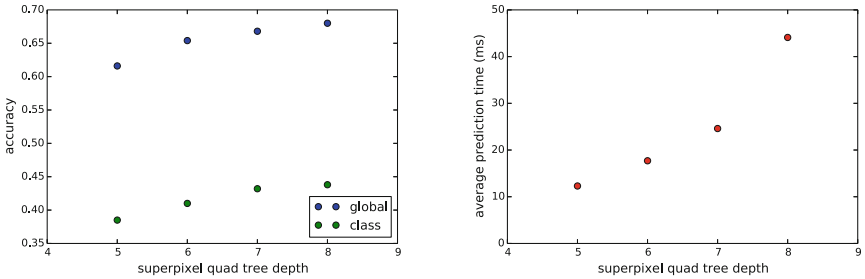

Fig. 3. Accuracy and average prediction time for one tree using different quadtree depths when creating superpixels. The results are reported for *sp-fr-Gauss2*.

We finally compare our approach with the state-of-the-art in terms of accuracy in Table 3. The first part of the table uses all training images for training. Our approach outperforms CURFIL [9] in terms of accuracy and runtime on a single-threaded CPU. Although the approach [18] achieves a higher global pixel accuracy, it is very expensive and requires 16,6 s for an image with resolution of 800 × 600 pixels. Our fastest setting requires only 40 milliseconds.

The second part of the table uses the evaluation protocol with 367 images. The numbers are taken from [3]. The convolutional neural network proposed in [3] achieves the best accuracy and requires around 2 s per image on a GPU. The methods based on CRFs [6] require 30 to 40 s for an image. The method [4] is based on random forests and structure-from-motion. It requires one second per image if the point cloud is already computed by structure-from-motion.

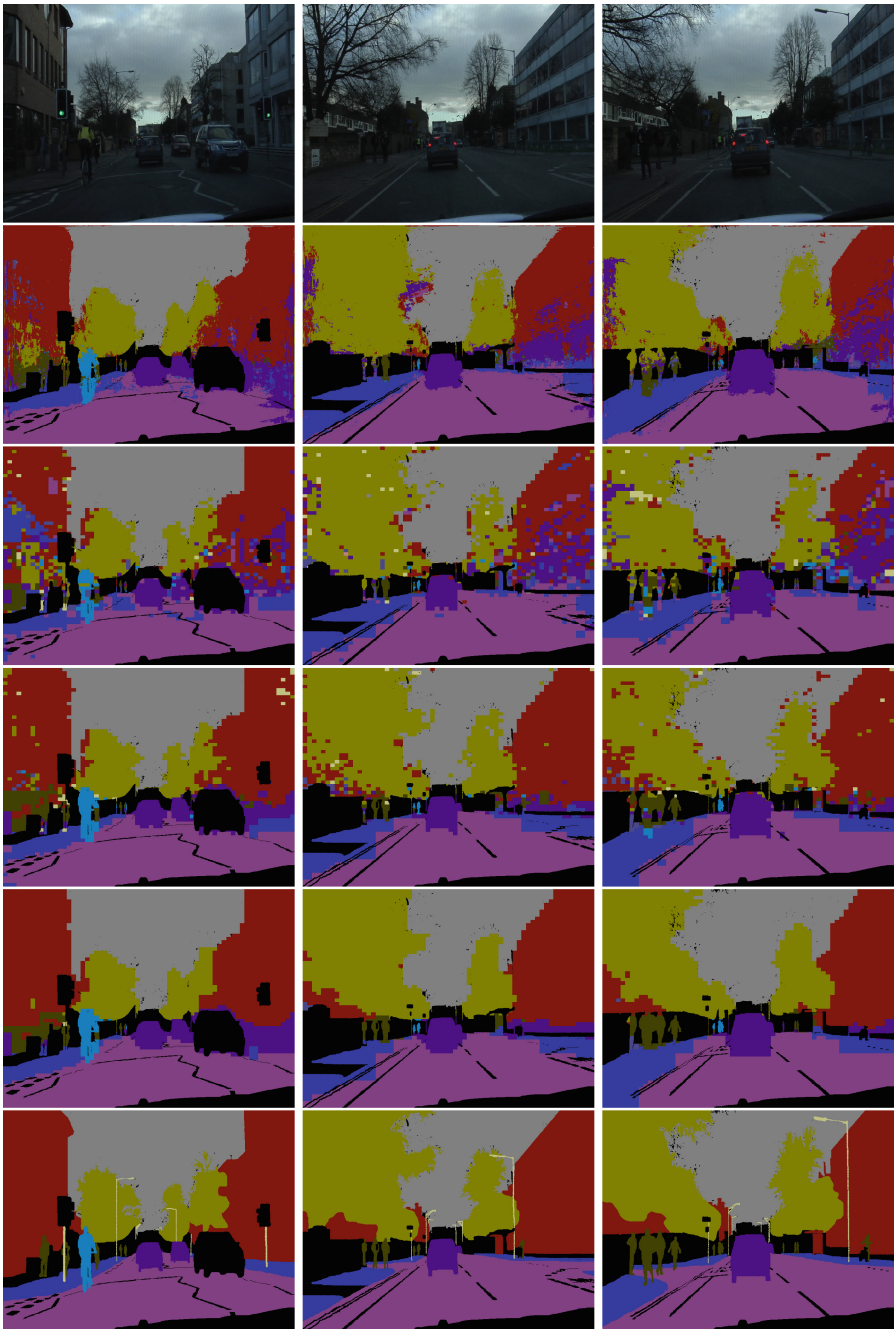

Fig. 4. Examples of segmentation results. First row: original image. Second row: *pixel stride 1*. Third row: *sp-fr*. Fourth row: *sp-fr-Gauss2 + propagate*. Fifth row: *sp-fr-Gauss2 + sm. + prop*. Sixth row: ground truth.

Table 3. Comparison with state-of-the-art approaches. The first six rows shows results for all 468 training images. The lower part report the results when only the sequences recorded with 30 Hz are used for training (367).

	Global pixel accuracy	Average class accuracy	Average time (ms)
Super Parsing [18]	83.3	51.2	
CURFIL [9]	65.9	49.8	34163
sp-fr-Gauss2	77.5	51.3	105.3
sp-fr-Gauss2 + smoothing	79.6	51.8	131.5
sp-fr-Gauss2 + propagate	76.6	50.0	40.2
sp-fr-Gauss2 + sm. + prop.	78.6	50.8	58.8
Appearance [4]	66.5	52.3	
SfM + Appearance [4]	69.1	53.0	
Boosting [6]	76.4	59.8	
Dense Depth Maps [20]	82.1	55.4	
Structured Random Forests [8]	72.5	51.4	
Neural Decision Forests [19]	82.1	56.1	
Local Label Descriptors [21]	73.6	36.3	
SegNet - 4 layer [3]	84.3	62.9	2000
Boosting + pairwise CRF [6]	79.8	59.9	
Boosting + Higher order [6]	83.8	59.2	
Boosting + Detectors + CRF [7]	83.8	62.5	
sp-fr-Gauss2 (367 images)	77.4	51.2	102.5
sp-fr-Gauss2 + smoothing (367 images)	80.0	52.2	111.5
sp-fr-Gauss2 + propagate (367 images)	76.8	50.5	36.5
sp-fr-Gauss2 + sm. + prop. (367 images)	79.3	51.7	55.5

The methods [8,19] are also too slow for real-time applications. In contrast, our approach segments an image not in the order of seconds but milliseconds while still achieving competitive accuracies. A few qualitative results are shown in Fig. 4.

5 Conclusion

In this work, we proposed a real-time approach for semantic segmentation on a single-threaded architecture. Compared to the baseline we reduced the runtime by factor 192 while increasing the accuracy. This has been achieved by combining an efficient superpixel representation based on quadtrees with random forests and combining label propagation with spatial smoothing. Compared to

the state-of-the-art in terms of accuracy, our approach achieves competitive results but runs in real-time without the need of a GPU. This make the approach ideal for applications with limited computational resources.

Acknowledgement. The work has been financially supported by the DFG project GA 1927/2-2 as part of the DFG Research Unit FOR 1505 Mapping on Demand (MoD).

References

1. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1915–1929 (2013)
2. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.: Semantic image segmentation with deep convolutional nets and fully connected CRFs. In: International Conference on Learning Representations (2015)
3. Badrinarayanan, V., Handa, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. CoRR abs/1505.07293 (2015)
4. Brostow, G.J., Shotton, J., Fauqueur, J., Cipolla, R.: Segmentation and recognition using structure from motion point clouds. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008. LNCS, vol. 5302, pp. 44–57. Springer, Heidelberg (2008). doi:10.1007/978-3-540-88682-2_5
5. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: IEEE Computer Vision and Pattern Recognition (2008)
6. Sturgess, P., Alahari, K., Ladicky, L., Torr, P.H.: Combining appearance and structure from motion features for road scene understanding. In: British Machine Vision Conference (2009)
7. Ladický, Ľ., Sturgess, P., Alahari, K., Russell, C., Torr, P.H.S.: What, where and how many? combining object detectors and CRFs. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 424–437. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15561-1_31
8. Kontschieder, P., Rota Bulò, S., Bischof, H., Pelillo, M.: Structured class-labels in random forests for semantic image labelling. In: IEEE International Conference on Computer Vision, pp. 2190–2197 (2011)
9. Schulz, H., Waldvogel, B., Sheikh, R., Behnke, S.: CURFIL: random forests for image labeling on GPU. In: Proceedings of the International Conference on Computer Vision Theory and Applications (2015)
10. Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. (2016)
11. Grundmann, M., Kwatra, V., Han, M., Essa, I.: Efficient hierarchical graph-based video segmentation. In: IEEE Computer Vision and Pattern Recognition, pp. 2141–2148 (2010)
12. Budvytis, I., Badrinarayanan, V., Cipolla, R.: Label propagation in complex video sequences using semi-supervised learning. In: British Machine Vision Conference, vol. 2257, pp. 2258–2259 (2010)
13. Reso, M., Jachalsky, J., Rosenhahn, B., Ostermann, J.: Fast label propagation for real-time superpixels for video content. In: IEEE International Conference on Image Processing (2015)

14. Criminisi, A., Shotton, J.: Decision Forests for Computer Vision and Medical Image Analysis. Springer, London (2013)
15. Yang, B., Yan, J., Lei, Z., Li, S.Z.: Convolutional channel features. In: IEEE International Conference on Computer Vision, pp. 82–90 (2015)
16. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
17. Iqbal, U., Garbade, M., Gall, J.: Pose for action - action for pose. CoRR abs/1603.04037 (2016)
18. Tighe, J., Lazebnik, S.: Superparsing. Int. J. Comput. Vision **101**(2), 329–349 (2013)
19. Bulo, S., Kontschieder, P.: Neural decision forests for semantic image labelling. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 81–88 (2014)
20. Zhang, C., Wang, L., Yang, R.: Semantic segmentation of urban scenes using dense depth maps. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 708–721. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15561-1_51
21. Yang, Y., Li, Z., Zhang, L., Murphy, C., Hoeve, J., Jiang, H.: Local label descriptor for example based semantic image labeling. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7578, pp. 361–375. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33786-4_27