Semantic Soft Segmentation Supplementary Material

YAĞIZ AKSOY, MIT CSAIL and ETH Zürich
TAE-HYUN OH, MIT CSAIL and HBKU QCRI
SYLVAIN PARIS, Adobe Research
MARC POLLEFEYS, ETH Zürich and Microsoft
WOJCIECH MATUSIK, MIT CSAIL

ACM Reference format:
Yağız Aksoy, Tae-Hyun Oh, Sylvain Paris, Marc Pollefeys, and Wojciech Matusik. 2018. Semantic Soft Segmentation Supplementary Material. *ACM Trans. Graph.* 37, 4, Article 72-Supp. (August 2018), 6 pages.
DOI: 10.1145/3197517.3201275

To supplement the main document, we provide the details of our feature vector estimation in Section 1, and additional results and comparisons in Figures 3-5.

1 GENERATING SEMANTIC FEATURE DESCRIPTORS

We begin by computing a set of per-pixel semantic features for each input image. In principle, the network generating the features can be easily replaced to improve the results in parallel to advances in semantic segmentation, or to change the definition of *semantic objects*, such as to serve fine-grained or instance-aware semantic segmentation scenarios.

We train a deep convolutional neural network cascaded with metric learning, to generate features that are similar if they belong to the same object class, and distant from each other otherwise. The network outputs per-pixel semantic features of $d = 128$ dimensions. For simplicity, we denote a semantic feature vector $\boldsymbol{f}_p \in \mathbb{R}^d$ for each pixel p.

The base network of our feature extractor is based on DeepLab-ResNet-101 [Chen et al. 2017]. The DeepLab model is built on a fully convolutional variant of ResNet-101 [He et al. 2015a] with atrous convolutions and atrous spatial pyramid pooling. In the DeepLab-ResNet-101, the `res4b22` layer is the most commonly used output as a generic feature, which is 2048 dimensional at one sixteenth of the original image resulution. Since our purpose is to extract per-pixel feature with plausible object contours and boundaries, directly leveraging multi-scale context information is favorable when compared to using a condensed feature at higher layer such as `res5b_relu`. We modify the architecture to take features from lower as well as higher-levels into account. We use the feature concatenation, motivated by [Bertasius et al. 2015; Hariharan et al. 2015], but we maintain a light representation to avoid large memory bottlenecks. We branch

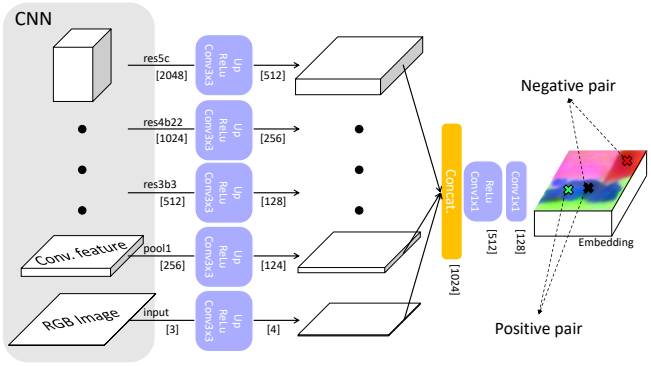

Fig. 1. Our network architecture. We extract the feature of intermediate representation of the base convolution neural network (we use the DeepLap variant of ResNet-101). The feature are compressed by 3×3 convolution followed by ReLu and bilinear up-sampling to have the same resolution with input. The concatenated features are fed to subsequent 1×1 convolution. On top of this feature, we apply sampling based metric learning in an end-to-end manner. We denote feature dimension as [#].

`input`, `pool1`, `res3b3`, `res4b22` and `res5c` layers to extract the features, followed by a 3×3 convolution with ReLu to compress the intermediate feature dimensions from $\{3, 256, 512, 1024, 2048\}$ to $\{4, 124, 128, 256, 512\}$, respectively, for a total of 1024 dimensions. We then upsample them via bilinear upsampling to the input image resolution, followed by two 1×1 convolution layers which gradually reduce 1024 feature dimension to 512 and then finally to $d = 128$. The final output of this process defines our per-pixel semantic features $\boldsymbol{f}_p$. Our architecture is visualized in Figure 1. It is worth pointing out that our architecture is fully convolutional, allowing it to be applied to inputs of arbitrary resolution. While Bertasius et al.; Hariharan et al. leverage the pre-trained network without re-training, we fine-tune the whole network for our purpose.

To train the whole network, we use L2 distance between pixel features as the metric to measure the semantic similarity. We will now describe our loss function on the pixel-level. Given a query vector $\boldsymbol{f}_p$ of a pixel p, we use positive vectors to pull the query towards positive one a and negative vectors to push to negative one for positive and negative examples from the query at a time [Hoffer and Ailon 2015]. Since we work on input image resolution, to easily utilize more data and be computationally more efficient, we use N-pair loss [Sohn 2016] with a slight modification. The N-pair loss

0730-0301/2018/8-ART72-Supp. $15.00
DOI: 10.1145/3197517.3201275

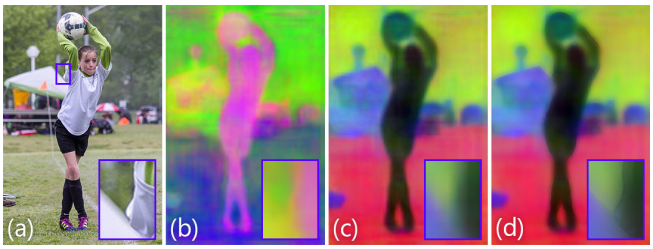

Fig. 2. We first generate a 128-dimensional feature vector per pixel for a given image (a). A random projection of 128 dimensions to 3 is shown in (b). We reduce the dimensionality of the features to 3 using principle component analysis per image (c). In order to align the feature vectors to the image edges, we first filter each of the 128 dimensions with guided filter [He et al. 2013] and then apply dimentionality reduction (d).

benefits data efficiency by hard negative data-mining style formulation and cross-entropy style loss to alleviate the slow convergence by loss-balancing in triplet loss [Hoffer and Ailon 2015]. While the N-pair loss is defined on an inner product-based metric, we replace it with L2 distance. Hence, our loss is defined by:

$$L_m = \frac{1}{|\mathcal{P}|} \sum_{p,q \in \mathcal{P}} \mathbb{I}[l_p = l_q] \log\left(\left(1 + \exp\left(\|f_p - f_q\|\right)\right) / 2\right) + \mathbb{I}[l_p \neq l_q] \log\left(1 + \exp\left(-\|f_p - f_q\|\right) / 2\right), \quad (1)$$

where $\mathcal{P}$ denotes the set of sampled pixels, $\|\cdot\|$ L2-norm (we divide it by d for normalization), $\mathbb{I}[\cdot]$ the indicator function that returns 1 if the statement is true and 0 otherwise, and l_p the semantic label of pixel p.

In (1), for positive pairs, *i.e.* $l_p = l_q$, the corresponding term $\log\left(\left(1 + \exp\left(\|f_p - f_q\|\right)\right) / 2\right)$ approaches zero. The conjugate relation applies to the negative pairs in the second term in (1). Since we only use this cue, whether two pixels belong to the same category or not, specific object category information is not used during training. Hence, our method is a class agnostic approach. This fact does not harm our overall goal of semantic soft segmentation as we aim to create soft segments that cover semantic objects, rather than classification of the objects in an image. This also enables us to take into account diversity of semantics and not be limited to user-selected classes.

We construct the set of sampled pixels $\mathcal{P}$ as follows. During training, we feed a single image as a mini-batch to the network, and we get the features for all pixels. Given an input image and its corresponding semantic ground-truth labels, we first randomly sample P_{inst} number of instances, then for each instance, we randomly sample P_{pix} number of pixels within each instance label mask, so that the number of pixels in each group are balanced. We minimize (1) for the selected samples, and we repeat the sampling 10 times per image, accumulate gradients from them, and update at once. We set $P_{\text{inst}} = 3$ and $P_{\text{pix}} = 1000$. We can easily compute (1) in a matrix form by using $\mathbf{D} = \mathbf{F}\mathbf{1}_d\mathbf{1}_d{}^T + (\mathbf{V}\mathbf{1}_d\mathbf{1}_d{}^T)^T - 2\mathbf{V}\mathbf{V}^T$, where $\mathbf{D}$ is the matrix containing L2 distances between the feature vectors, $\mathbf{1}_d$ is a row-vector of ones, and $\mathbf{F}$ contains the feature vectors of the samples pixels:

$$\mathbf{F} = \left[f_{1,1} \cdots f_{1,P_{\text{pix}}}, \mathbf{f}_{2,1} \cdots f_{2,P_{\text{pix}}}, \cdots f_{P_{\text{inst}},P_{\text{pix}}}\right]^T \quad (2)$$

We trained our network on the training split of COCO-Stuff [Caesar et al. 2016], which has 182 number of object and stuff categories with instance-level annotation. We initialized the base DeepLab part with the pretrained weights on the semantic segmentation task of MS-COCO [Lin et al. 2014] (80 categories), and the remaining parts with Xavier initialization [He et al. 2015b]. We set the learning rate to 5×10^{-4} for the base part and 5×10^{-3} for the rest to compensate for the random initialization. We use stochastic gradient descent with momentum 0.9, poly-learning rate decay of 0.9 as suggested by Chen *et al.* [2017], and weight decay of 5×10^{-4}. We also use drop-out with probability 0.5 for 1×1 convolutions at the two last stages. We train for $60k$ iterations and it roughly takes less than one day on an NVIDIA Titan X Pascal GPU.

1.1 Preprocessing

The 128-dimensional feature vectors f_p have enough capacity to represent a large diversity of real-world semantic categories. However, for a given image, as the number of object categories present in the scene is inherently limited, the effective dimensionality of the feature vector is much smaller. Following this fact, in order to make the graph construction (described in the main paper) more tractable and less prone to parameter-tuning, we reduce the dimensionality of the feature vectors to three using per-image principle component analysis.

One of the major shortcomings of semantic *hard* segmentation is its inaccuracy around object boundaries [Bertasius et al. 2016]. This fact is well-reflected in the generated feature vectors as well, as shown in Figure 2. In order to compute more effective affinities when we are inserting the semantic information into the graph, we regularize the feature vectors using guided filtering [He et al. 2013] with the guidance of the input image. This makes the features to be more consistent with hard boundaries in the image, as shown in Figure 2. We do this filtering for all 128 dimensions prior to the dimensionality reduction. Finally, we normalize the lower-dimensional features to be in the range $[0, 1]$ to get the three dimensional feature vectors $\tilde{f}_p$ to be used for affinity computations.

REFERENCES

Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. 2015. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *Proc. ICCV*.

Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. 2016. Semantic Segmentation with Boundary Neural Fields. In *Proc. CVPR*.

Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. 2016. COCO-Stuff: Thing and Stuff Classes in Context. `arXiv:1612.03716 [cs.CV]` (2016).

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2017. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* (2017).

Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. 2015. Hypercolumns for object segmentation and fine-grained localization. In *Proc. CVPR*.

Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. 2017. Mask R-CNN. In *Proc. ICCV*.

Kaiming He, Jian Sun, and Xiaoou Tang. 2013. Guided Image Filtering. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 6 (2013), 1397–1409.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015a. Deep Residual Learning for Image Recognition. `arXiv:1512.03385 [cs.CV]` (2015).

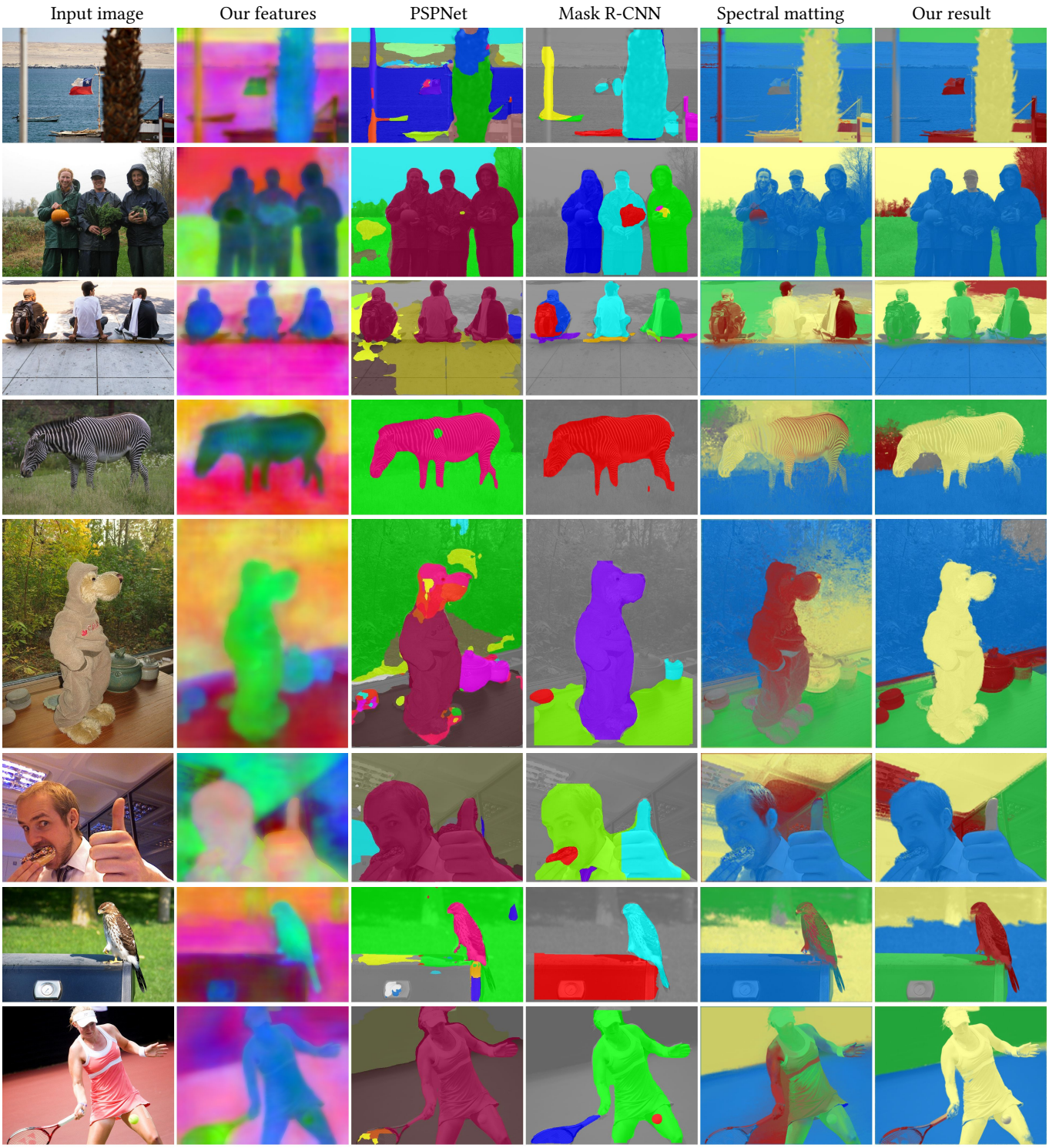

Fig. 3. We show our results together with that of Zhao *et al.* [2017] (PSPNet), He *et al.* [2017] (Mask R-CNN), and spectral matting [Levin et al. 2008]. The segmentations are overlayed onto the grayscale version of the input image for a better evaluation around segment boundaries. Notice the inaccuries of PSPNet and Mask R-CNN around object boundaries, and the soft segments of spectral matting extending beyond objects.

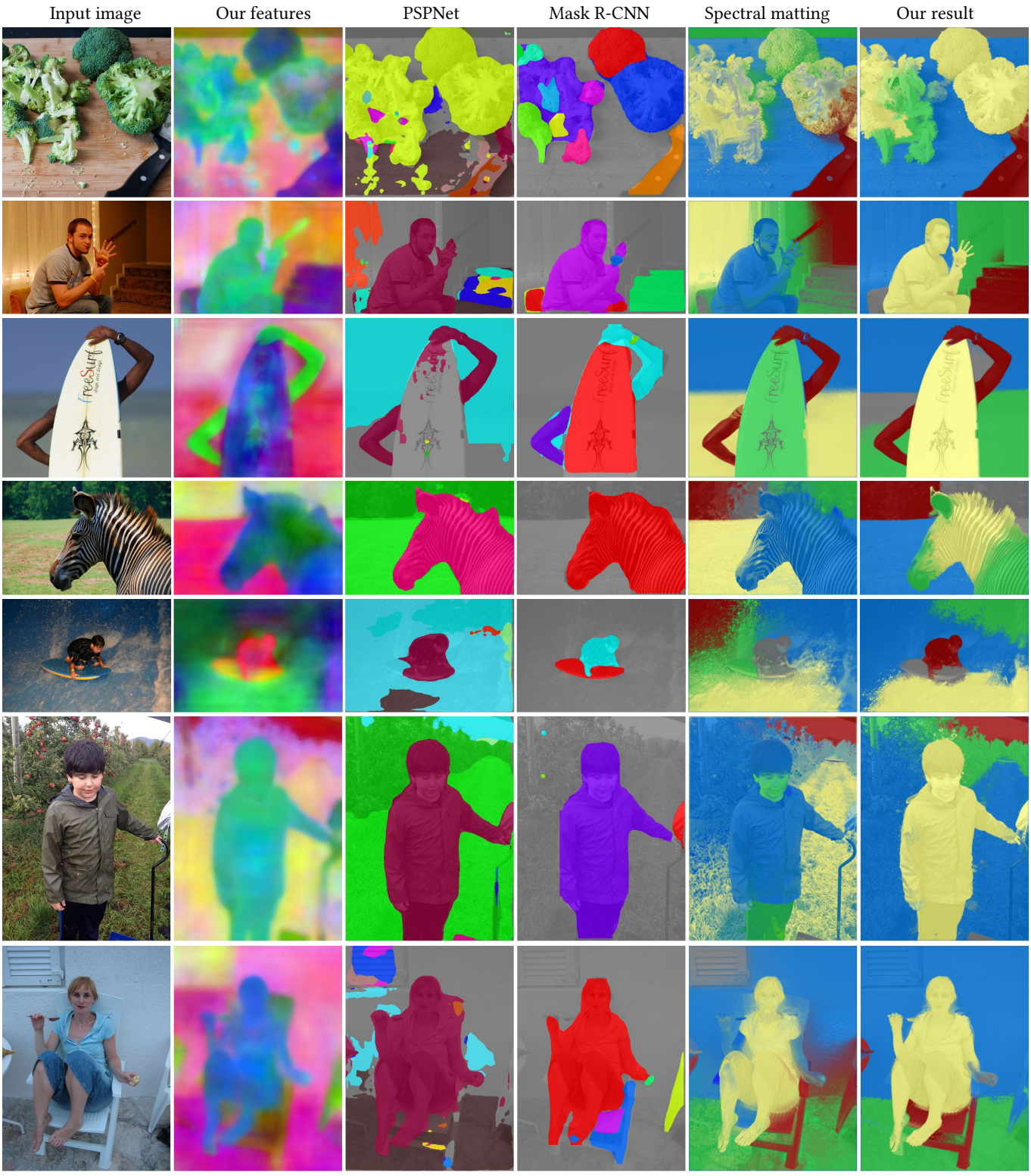

Fig. 4. We show our results together with that of Zhao *et al.* [2017] (PSPNet), He *et al.* [2017] (Mask R-CNN), and spectral matting [Levin et al. 2008]. The segmentations are overlayed onto the grayscale version of the input image for a better evaluation around segment boundaries. Notice the inaccuries of PSPNet and Mask R-CNN around object boundaries, and the soft segments of spectral matting extending beyond objects.

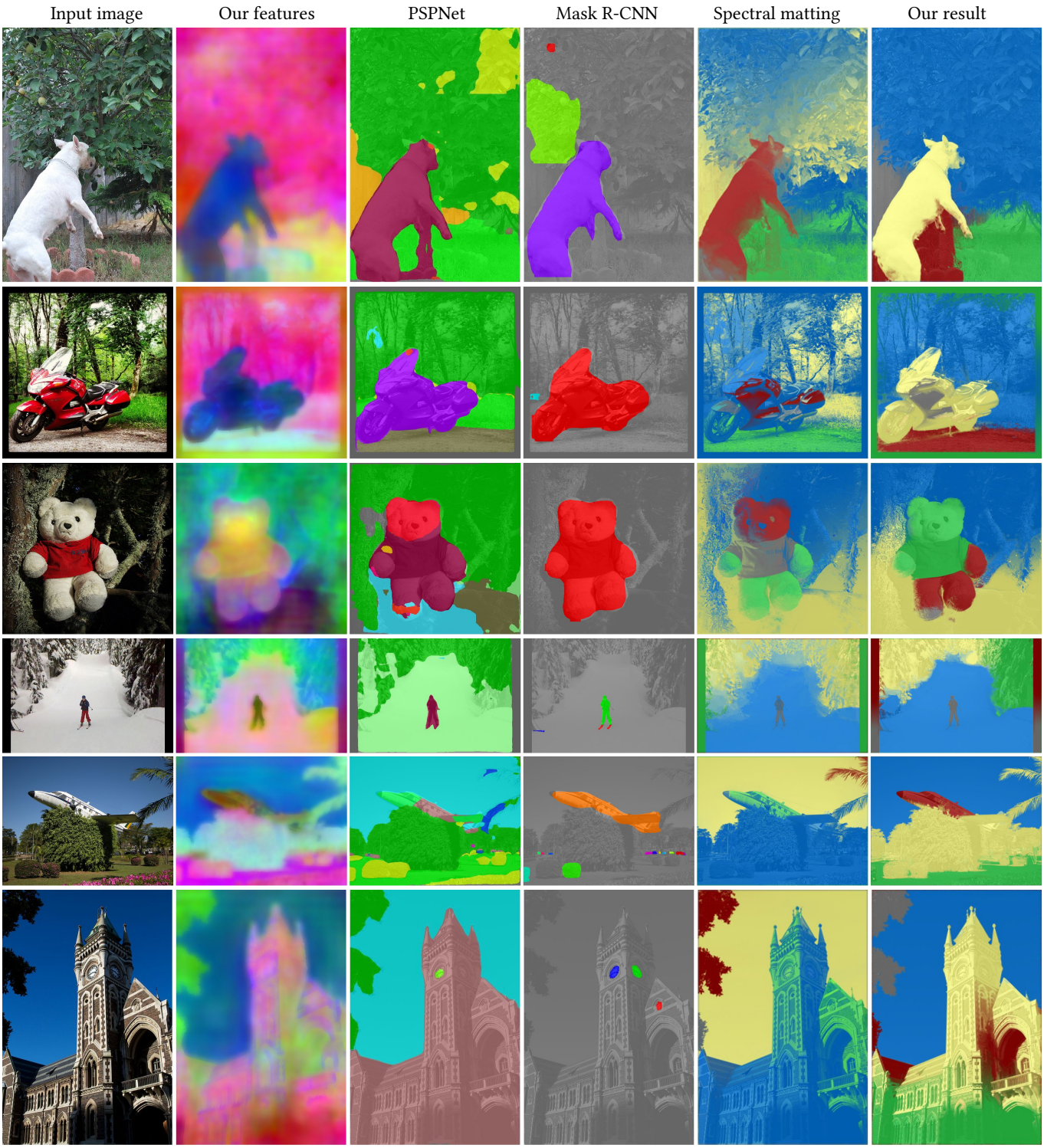

Fig. 5. We show our results together with that of Zhao *et al.* [2017] (PSPNet), He *et al.* [2017] (Mask R-CNN), and spectral matting [Levin et al. 2008]. The segmentations are overlayed onto the grayscale version of the input image for a better evaluation around segment boundaries. Notice the inaccuries of PSPNet and Mask R-CNN around object boundaries, and the soft segments of spectral matting extending beyond objects.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015b. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. ICCV*.

Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*.

Anat Levin, Alex Rav-Acha, and Dani Lischinski. 2008. Spectral Matting. *IEEE Trans. Pattern Anal. Mach. Intell.* 30, 10 (2008), 1699–1712.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Proc. ECCV*.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Proc. NIPS*.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid Scene Parsing Network. In *Proc. CVPR*.