Semantic Soft Segmentation

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

Fig. 1. We propose a method that can generate soft segments, i.e. layers that represent the semantically meaningful regions as well as the soft transitions between them, automatically by fusing high-level and low-level image features in a single graph structure. The semantic soft segments, visualized by assigning each segment a solid color, can be used as masks for targeted image editing tasks, or selected layers can be used for compositing after layer color estimation. Original images are from [Lin et al. 2014] (top-left, bottom-right), by Death to the Stock Photo (top-right) and by Y. Aksoy (bottom-left).

Accurate representation of soft transitions between image regions is essential for high-quality image editing and compositing. Current techniques for generating such representations depend heavily on interaction by a skilled visual artist, as creating such accurate object selections is a tedious task. In this work, we introduce *semantic soft segments*, a set of layers that correspond to semantically meaningful regions in an image with accurate soft transitions between different objects. We approach this problem from a spectral segmentation angle and propose a graph structure that embeds texture and color features from the image as well as higher-level semantic information generated by a neural network. The soft segments are generated via eigendecomposition of the carefully constructed Laplacian matrix fully automatically. We demonstrate that otherwise complex image editing tasks can be done with little effort using semantic soft segments.

CCS Concepts: • **Computing methodologies** → **Image segmentation**;

Additional Key Words and Phrases: soft segmentation, semantic segmentation, natural image matting, image editing, spectral segmentation

ACM Reference Format:
Yağız Aksoy, Tae-Hyun Oh, Sylvain Paris, Marc Pollefeys, and Wojciech Matusik. 2018. Semantic Soft Segmentation. *ACM Trans. Graph.* 37, 4, Article 72 (August 2018), 13 pages. https://doi.org/10.1145/3197517.3201275

Authors' addresses: Yağız Aksoy, MIT CSAIL, 32 Vassar St. Cambridge, MA, 02139, USA, ETH Zürich, Department of Computer Science, Rämistrasse 101, Zürich, 8092, Switzerland; Tae-Hyun Oh, MIT CSAIL, 32 Vassar St. Cambridge, MA, 02139, USA; Sylvain Paris, Adobe Research, One Broadway, Cambridge, MA, 02142, USA; Marc Pollefeys, ETH Zürich, Department of Computer Science, Rämistrasse 101, Zürich, 8092, Switzerland, Microsoft, Redmond, WA, USA; Wojciech Matusik, MIT CSAIL, 32 Vassar St. Cambridge, MA, 02139, USA.

1 INTRODUCTION

Selection and compositing are at the core of the image editing process. For instance, local adjustments often start with a selection, and combining elements from different images is a powerful way to produce new content. But creating an accurate selection is a tedious task especially when fuzzy boundaries and transparency are involved. Tools such as the *magnetic lasso* and the *magic wand* exist to assist users but they only exploit low-level cues and heavily rely on the users' skills and interpretation of the image content to

produce good results. Furthermore, they only produce binary selections that need further refinement to account for soft boundaries like the silhouette of a furry dog. Matting tools also exist to help users with this task but they only add to the tedium of the entire editing process.

An accurate pre-segmentation of the image can speed up the editing process by providing an intermediate image representation if it satisfies several criteria. First of all, such a segmentation should provide distinct segments of the image, while also representing the soft transitions between them accurately. In order to allow targeted edits, each segment should be limited to the extent of a semantically meaningful region in the image, e.g., it should not extend across the boundary between two objects. Finally, the segmentation should be done fully automatically not to add a point of interaction or require expertise from the artist. The previous approaches for semantic segmentation, image matting, or soft color segmentation fail to satisfy at least one of these qualities. In this paper, we introduce *semantic soft segmentation*, a fully automatic decomposition of an input image into a set of layers that cover scene objects, separated by soft transitions.

We approach the semantic soft segmentation problem from a spectral decomposition angle. We combine the texture and color information from the input image together with high-level semantic cues that we generate using a convolutional neural network trained for scene analysis. We design a graph structure that reveals the semantic objects as well as the soft transitions between them in the eigenvectors of the corresponding Laplacian matrix. We introduce a spatially varying model of layer sparsity that generates high-quality layers from the eigenvectors that can be utilized for image editing.

We demonstrate that our algorithm successfully decomposes images into a small number of layers that compactly and accurately represent the scene objects as shown in Figure 1. We later show that our algorithm can successfully process images that are challenging for other techniques and we provide examples of editing operations such as local color adjustment or background replacement that benefit from our layer representation.

2 RELATED WORK

Soft segmentation. Soft segmentation is decomposing an image into two or more segments where each pixel may belong partially to more than one segment. The layer contents change depending on the specific goal of the corresponding method. For instance, soft *color* segmentation methods extract soft layers of homogeneous colors using global optimization [Singaraju and Vidal 2011; Tai et al. 2007; Tan et al. 2016] or per-pixel color unmixing [Aksoy et al. 2016, 2017b]. While soft color segments are shown to be useful for several image editing applications such as image recoloring, their content typically does not respect object boundaries, not allowing targeted edits. To generate spatially connected soft segments, Singaraju and Vidal [2011] start from a set of user-defined regions and solve two-layer soft segmentation problems multiple times to generate multiple layers. Levin et al. [2008b], on the other hand, propose spectral matting, estimating a set of spatially connected soft segments automatically via spectral decomposition. Both Singaraju and Vidal [2011] and Levin et al. [2008b] construct their algorithms around the matting Laplacian [Levin et al. 2008a], which provides a powerful representation for local soft transitions in the image. We also make use of the matting Laplacian and spectral decomposition, following ideas from spectral matting. However, unlike previous work, we construct a graph that fuses higher-level information coming from a deep network with the local texture information in order to generate soft segments that correspond to semantically meaningful regions in the image.

Natural image matting. Natural image matting is the estimation of per-pixel opacities of a user-defined foreground region. The typical input to natural matting algorithms is a *trimap*, which defines the opaque foreground, the transparent background, and the unknown-opacity regions. While there are different approaches to this problem all of which make use of the color characteristics of the defined foreground and background regions, the most closely-related approaches to ours are categorized as *affinity-based* methods. The affinity-based methods, such as closed-form matting [Levin et al. 2008a], KNN matting [Chen et al. 2013], and information-flow matting [Aksoy et al. 2017a], define inter-pixel affinities to construct a graph that reflects the opacity transitions in the image. In contrary to natural image matting methods, we rely on automatically-generated semantic features in defining our soft segments instead of a trimap, and generate multiple soft segments rather than foreground segmentation. Although they appear similar, natural matting and soft segmentation have fundamental differences. Natural matting, with a trimap as input, becomes the problem of foreground and background color modeling, may it be through selection of color samples or propagation of color information. Meanwhile, soft segmentation focuses on detecting soft transitions that best serve the target application, in our case the ones corresponding to semantic boundaries.

Targeted edit propagation. Several image editing methods rely on user-defined sparse edits on the image and propagate them to the whole image. ScribbleBoost [Li et al. 2008] proposed a pipeline where they classify the objects specified by the user scribbles to allow edits targeting specific object classes in the image, and DeepProp [Endo et al. 2016] utilized a deep network to propagate class-dependent color edits. Eynard et al. [2014] constructs a graph and, parallel to our method, analyze the eigendecomposition of the corresponding Laplacian matrix to create coherent recoloring results. An and Pellacini [2008] and Chen et al. [2012] also define inter-pixel affinities and make use of the properties of the Laplacian matrices to solve for plausible propagations of the user-defined edits. While our results can also be used for targeted edits, rather than using edits defined a priori, we directly decompose the image into soft segments and let the artist use them as an intermediate image representation in various scenarios and using external image editing tools.

Semantic segmentation. Semantic segmentation has improved significantly with the introduction of deep neural networks. While a detailed report on semantic segmentation is beyond our scope, state-of-the-art in semantic segmentation include works on scene parsing by Zhao et al. [2017], instance segmentation methods by He et al. [2017] and Fathi et al. [2017], and work by Bertasius et al. [2016] which enhances semantic segmentation with color boundary cues. We also make use of a deep network for semantic features,

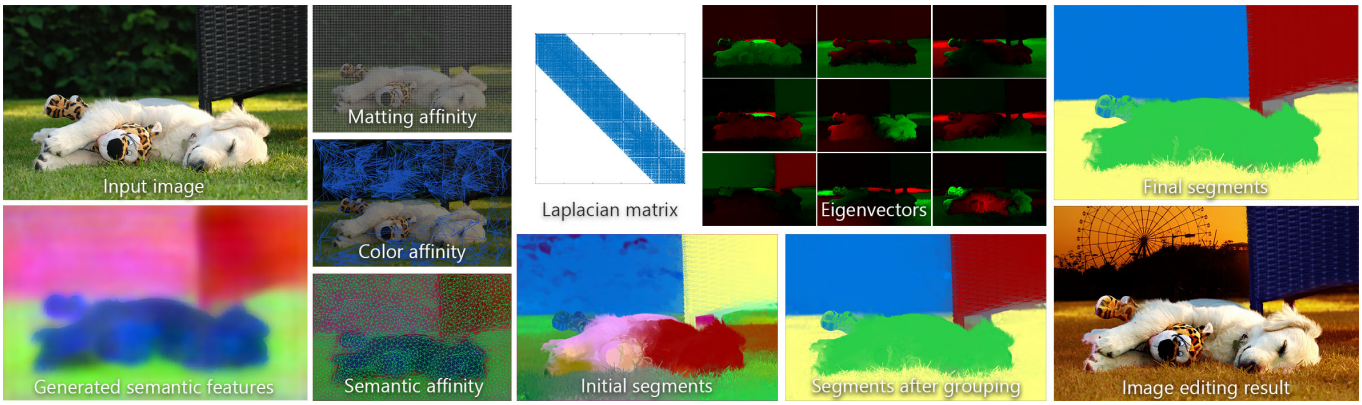

Fig. 2. For an input image, we generate per-pixel hyperdimensional semantic feature vectors and define a graph using the texture and semantic information. The graph is constructed such that the corresponding Laplacian matrix and its eigenvectors reveal the semantic objects and the soft transitions between them. We use the eigenvectors to create a set of preliminary soft segments and combine them to get semantically meaningful segments. Finally, we refine the soft segments so that they can be used for targeted image editing tasks. Image from [Lin et al. 2014], background in the editing result by Flickr user *rumpleteaser*.

but our soft segmentation method is class-agnostic, i.e. we are interested in an accurate segmentation of the image respecting semantic boundaries, but we do not aim to do classification or detection of a selected set of classes. Others also make use of class-agnostic semantic information to improve performance in video deblurring [Ren et al. 2017] or cinemagraph generation [Oh et al. 2017].

3 METHOD

We seek to automatically generate a *soft segmentation* of the input image, that is, a decomposition into layers that represent the objects in the scene including transparency and soft transitions when they exist. Each pixel of each layer is augmented with an opacity value $\alpha \in [0, 1]$ with $\alpha = 0$ meaning fully transparent, $\alpha = 1$ fully opaque, and in-between values indicating the degree of partial opacity. As other studies in this domain such as [Aksoy et al. 2017b; Singaraju and Vidal 2011], we use an additive image formation model:

$$(R, G, B)_{\text{input}} = \sum_i \alpha_i (R, G, B)_i \tag{1a}$$

$$\sum_i \alpha_i = 1, \tag{1b}$$

i.e., we express the input RGB pixels as the sum of the pixels in each layer i weighted by its corresponding α value. We also constrain the α values to sum up to 1 at each pixel, representing a fully opaque input image.

Our approach uses the same formalism as spectral matting in formulating the soft segmentation task as an eigenvector estimation problem [Levin et al. 2008b]. The core component of this approach is the creation of a *Laplacian* matrix L that represents how likely each pair of pixels in the image is to belong to the same segment. While spectral matting builds this matrix using only low-level local color distributions, we describe how to augment this approach with nonlocal cues and high-level semantic information. The original approach also describes how to create the layers from the eigenvectors of L using sparsification. We show how a relaxed version of this original technique actually yields better results. Figure 2 shows an overview of our approach.

3.1 Background

Spectral matting. Our approach builds upon the work of Levin et al. [2008a; 2008b]. They first introduced the matting Laplacian that uses local color distributions to define a matrix L that captures the affinity between each pair of pixels in a local patch, typically 5×5 pixels. Using this matrix, they minimize the quadratic functional $\boldsymbol{\alpha}^T L \boldsymbol{\alpha}$ subject to user-provided constraints, with $\boldsymbol{\alpha}$ denoting a vector made of all the α values for a layer. This formulation shows that the eigenvectors associated to small eigenvalues of L play an important role in the creation of high-quality mattes. Motivated by this observation, their subsequent work on spectral matting used the eigenvectors of L to build a soft segmentation [Levin et al. 2008b]. Each soft segment is a linear combination of the K eigenvectors corresponding to the smallest eigenvalues of L that maximizes *matting sparsity*, i.e., minimizes the occurrence of partial opacity. The segments are created by minimizing an energy function that favors $\alpha = 0$ and $\alpha = 1$:

$$\arg\min_{\{\boldsymbol{y}_i\}} \sum_{i,p} |\alpha_{ip}|^\gamma + |1 - \alpha_{ip}|^\gamma \quad \text{with: } \boldsymbol{\alpha}_i = \mathsf{E}\boldsymbol{y}_i \tag{2a}$$

$$\text{subject to: } \sum_i \alpha_{ip} = 1, \tag{2b}$$

where α_{ip} is the α value of p^{th} pixel of the i^{th} segment, E is a matrix containing the K eigenvectors of L with smallest eigenvalues, $\boldsymbol{y}_i$ is the linear weights on the eigenvectors that define the soft segments, and $\gamma < 1$ is a parameter that controls the strength of the sparsity prior.

While spectral matting generates satisfying results when the image contains a single well-identified object with distinct colors, it struggles with more complex objects and scenes. Being based solely on the matting Laplacian that considers only low-level statistics of small patches, it is limited in its ability to identify objects. In our work, we extend this approach to fuse semantic features in the same Laplacian formulation and capture higher-level concepts like scene objects and to have a broader view of the image data.

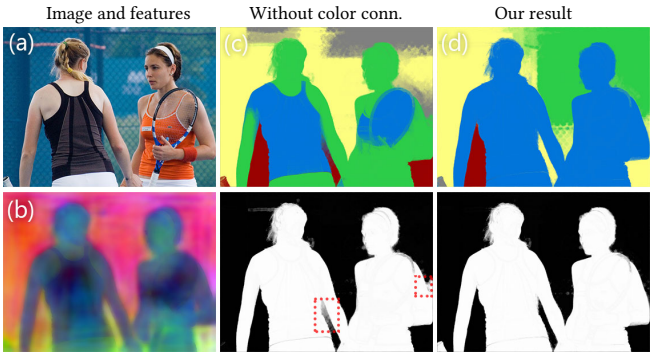

Fig. 3. The color-based nonlocal affinity we include helps the decomposition recover isolated regions such as disconnected background regions or long extensions, as the highlights point out. Image from [Lin et al. 2014].

Affinity and Laplacian matrices. Levin et al. [2008a] formulate their approach as a least-squares optimization problem that directly leads to a Laplacian matrix. An alternative approach is to express the *affinity* between pairs of pixels [Aksoy et al. 2017a]. Pairs with a positive affinity are more likely to have similar values, zero-affinity pairs are independent, and pairs with a negative affinity are likely to have different values. In this work, we will use the affinity approach and build the corresponding normalized Laplacian matrix using the well-known formula:

$$L = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}, \tag{3}$$

where W is a square matrix containing the affinity between all pairs of pixels and D is the corresponding degree matrix, i.e. a diagonal matrix with elements $W\mathbf{1}$, $\mathbf{1}$ being a row vector of ones. As noted by Levin et al., L may not always be a *true* graph Laplacian due to the presence negative affinities, but nonetheless shares similar properties such as being positive semidefinite.

3.2 Nonlocal Color Affinity

We define an additional low-level affinity term that represents color-based longer-range interactions. A naive approach would be to use larger patches in the definition of the matting Laplacian. However, this option quickly becomes impractical because it renders the Laplacian matrix denser. Another option is to sample pixels from a nonlocal neighborhood to insert connection while preserving some sparsity in the matrix. KNN matting [Chen et al. 2013] and information-flow matting [Aksoy et al. 2017a] have shown good results for medium-range interaction with such sampling. However, this strategy faces a trade-off between sparsity and robustness: fewer samples may miss important image features and more samples make the computation less tractable.

We propose a guided sampling based on an oversegmentation of the image. We generate 2500 superpixels using SLIC [Achanta et al. 2012] and estimate the affinity between each superpixel and all the superpixels within a radius that corresponds to 20% of the image size. The advantage of this approach is that each feature large enough to be a superpixel is represented, sparsity remains high because we use a single sample per superpixel, and it links possibly disconnected regions by using a large radius, e.g. when the background is seen through a hole in an object. Formally, we define the color affinity $w^C_{s,t}$ between the centroids of two superpixels s and t separated by a distance less than 20% of the image size as:

$$w^C_{s,t} = (\mathrm{erf}\,(a_c\,(b_c - \|\boldsymbol{c}_s - \boldsymbol{c}_t\|)) + 1)\,/\,2, \tag{4}$$

where $\boldsymbol{c}_s$ and $\boldsymbol{c}_t$ are the average colors of the superpixels of s and t that lies in $[0, 1]$, erf is the Gauss error function, and a_c and b_c are parameters controlling how quickly the affinity degrades and the threshold where it becomes zero. erf takes values in $[-1, 1]$ and its use here is mainly motivated by its sigmoidal shape. We use $a_c = 50$ and $b_c = 0.05$ in all our results. This affinity essentially makes sure the regions with very similar colors stay connected in challenging scene structures, and its effect is demonstrated in Figure 3.

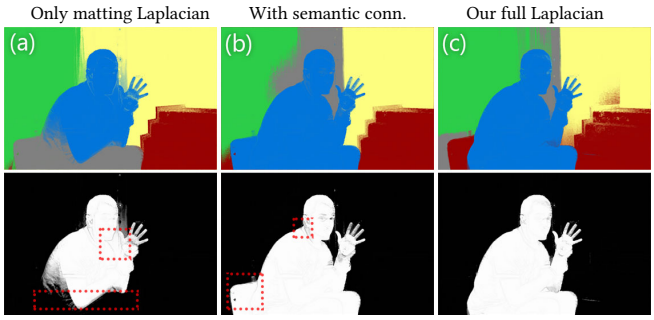

Fig. 4. The results of our entire pipeline using only the matting Laplacian (a), matting and semantic Laplacians (b) and the two together with the sparse color connections (c), for the image shown in Figure 5. The top row shows a distinct color for each produced soft segment, and the bottom row shows the extracted matte corresponding to the person. Due to the eigenvectors that are unable to represent the semantic cut between the person and the background, using only matting Laplacian results in the person soft segment including large portions of the background, as highlighted. Adding the sparse color connections provides a cleaner foreground matte.

3.3 High-Level Semantic Affinity

While the nonlocal color affinity adds long-range interactions to the segmentation process, it remains a low-level feature. Our experiments show that, without additional information, the segmentation still often merges image regions of similar color that belong to different objects. To create segments that are confined in semantically similar regions, we add a *semantic affinity* term, that is, a term that encourages the grouping of pixels that belong to the same scene object and discourages that of pixels from different objects. We build upon prior work in the domain of object recognition to compute a feature vector at each pixel that correlates with the underlying object. We compute the feature vectors via a neural network, as described in Section 3.5. The feature vectors are generated such that for two pixels p and q that belong to the same object $\boldsymbol{f}_p$ and $\boldsymbol{f}_q$ are similar, i.e. $\|\boldsymbol{f}_p - \boldsymbol{f}_q\| \equiv 0$, and for a third pixel r in a different semantic region, $\boldsymbol{f}_r$ is far away, i.e. $\|\boldsymbol{f}_p - \boldsymbol{f}_q\| \ll \|\boldsymbol{f}_p - \boldsymbol{f}_r\|$.

We define the semantic affinity also over superpixels. In addition to increasing the sparsity of the linear system, the use of superpixels also decrease the negative effect of the unreliable feature vectors in transition regions, as apparent from their blurred appearance in Figure 4. The superpixel edges are not directly used in the linear

Fig. 5. The image (a) and semantic features (b) are shown with several eigenvectors corresponding to the smallest eigenvalues[1] of the proposed Laplacian matrix (c, top row) and the matting Laplacian as used in spectral matting [Levin et al. 2008b] (d, bottom row). Green represents the positive values of an eigenvector while red shows negative. Our Laplacian matrix strongly reveals the semantic cuts in the eigenvectors while the matting Laplacian eigenvectors extend beyond the semantic edges, as the highlighted areas show. Image from [Lin et al. 2014].

system, the connections in the graph are between superpixel centroids. This information from the centroids then spreads to nearby pixels while respecting the image edges with the matting affinity term. With these vectors and the same oversegmentation in the previous section (§ 3.2), for each superpixel s, we associate its average feature vector $\tilde{\boldsymbol{f}}_s$ to its centroid p_s. We use these vectors to define an affinity term between each adjacent superpixels s and t:

$$w^S_{s,t} = \operatorname{erf}\left(a_s\left(b_s - \|\tilde{\boldsymbol{f}}_s - \tilde{\boldsymbol{f}}_t\|\right)\right), \tag{5}$$

with a_s and b_s parameters controlling the steepness of the affinity function and when it becomes negative. We discuss how to set them in Section 3.5. Defining negative affinities help the graph disconnect different objects while the positive values connect regions that belong to the same object.

Unlike the color affinity, the semantic affinity only relates nearby superpixels to favor the creation of connected objects. This choice of a *nonlocal* color affinity together with a local semantic affinity allows creating layers that can cover spatially disconnected regions of the same semantically coherent region. This often applies to elements like greenery and sky that often appear in the background, which makes them likely to be split into several disconnected components due to occlusions. As a result of including the local semantic affinity, the eigenvectors of L reveal object boundaries as demonstrated in Figure 4 and 5.

3.4 Creating the Layers

We create the layers by using the affinities described earlier in this section to form a Laplacian matrix L. We extract the eigenvectors from this matrix and use a two-step sparsification process to create the layers from these eigenvectors.

Forming the Laplacian matrix. We form a Laplacian matrix L by adding the affinity matrices together and using Equation 3:

$$L = D^{-\frac{1}{2}}\left(D - \left(W_L + \sigma_S W_S + \sigma_C W_C\right)\right)D^{-\frac{1}{2}} \tag{6}$$

where W_L is the matrix containing the matting affinities, W_C the matrix containing the nonlocal color affinities (§ 3.2), W_S the matrix with the semantic affinities (§ 3.3), and σ_S and σ_C parameters controlling the influence of each term, both set to be 0.01.

Constrained sparsification. We extract the eigenvectors corresponding to the 100 smallest eigenvalues of L. We form an intermediate set of layers using the optimization procedure by Levin et al. [2008b] on Eq. 2 with $\gamma = 0.8$. Unlike spectral matting that uses k-means clustering on the eigenvectors to initialize the optimization, we use k-means clustering on the pixels represented by their feature vectors $\boldsymbol{f}$. This initial guess is more consistent with the scene semantics and yields a better soft segmentation. We generate 40 layers with this approach and in practice, several of them are all zeros, leaving 15 to 25 nontrivial layers. We further reduce the number of layers by running the k-means algorithm with $k = 5$ on these nontrivial layers represented by their average feature vector. This approach works better than trying to directly sparsify the 100 eigenvectors into 5 layers, because such drastic reduction makes the problem overly constrained and does not produce good-enough results, especially in terms of matte sparsity. The initially estimated soft segments before and after grouping are shown in Figure 7. We have set the number of segments to 5 without loss of generalization; while this number could be set by the user depending on the scene structure, we have observed that it is a reasonable number for most images. Because these 5 layers are constrained to lie within the subspace of a limited number of eigenvectors, the achieved sparsity is suboptimal, leaving many semi-transparent regions in the layers, which is unlikely in common scenes. Next, we introduced a relaxed version of the sparsity procedure to address this issue.

Relaxed sparsification. To improve the sparsity of the layers, we relax the constraint that they are a linear combination of the eigenvectors. Instead of working with the coefficients $\boldsymbol{y}_i$ of the linear combination (Eq. 2), in this step, each individual α value is an unknown. We define an energy function that promotes matte sparsity

[1] In fact, the eigenvector corresponding to the smallest eigenvalue is not shown here as it is a constant vector for both matrices.

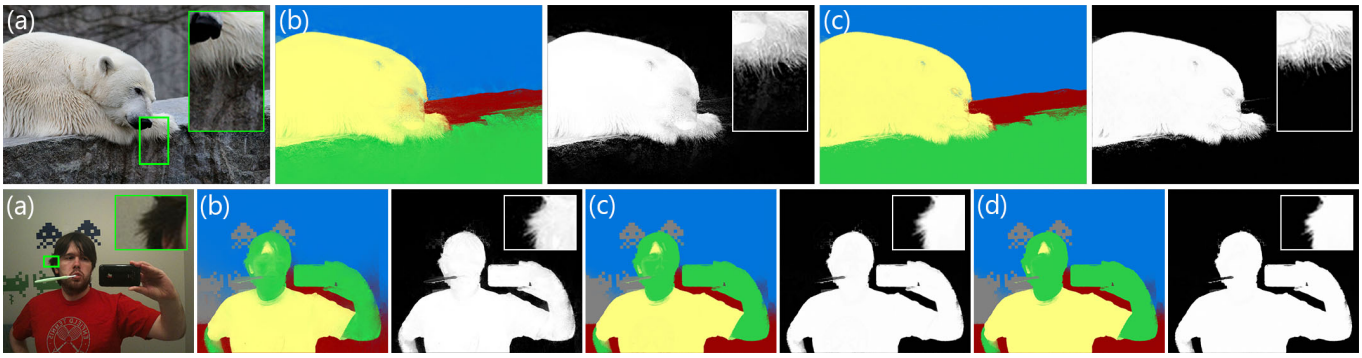

Fig. 6. The images (a) are shown with the results before pixel-level sparsification (b) and after (c). Color-coded segments are shown with a single alpha channel that corresponds to the foreground objects. This final step cleans spurious alpha values that occur due to the limited expressional power of the eigenvectors while preserving the soft transitions. The bottom example also features a sparsification result that uses a constant 0.9 as sparsity parameter γ (d), while we use spatially-varying γ_p which relaxes the sparsity constraint in transition regions. The effect of this can be seen in the inset, as our result (c) preserves the soft transitions around the hair while a constant parameter (d) results in an overly sparse result. Images from [Lin et al. 2014].

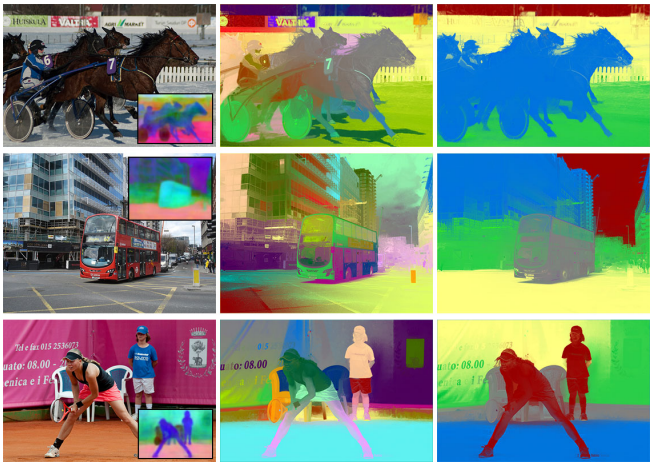

Fig. 7. The input image and computed semantic features are shown with the initially estimated soft segments with many layers (middle) and the intermediate soft segments after grouping (right). The soft segments are visualized by assigning each segment a solid color. Note that these results are refined further with relaxed sparsification. Images from [Lin et al. 2014].

on the pixel-level while respecting the initial soft segment estimates from the constrained sparsification and the image structure. We now define our energy term by term.

The first term relaxes the subspace constraint and only ensures that the generated layers remain close to the layers $\hat{\boldsymbol{\alpha}}$ created with the constrained sparsification procedure:

$$E_F = \sum_{ip} \left(\alpha_{ip} - \hat{\alpha}_{ip}\right)^2. \tag{7}$$

We also relax the sum-to-one requirement (Eq. 1b) to be integrated into the linear system as a soft constraint:

$$E_C = \sum_p \left(1 - \sum_i \alpha_{ip}\right)^2, \tag{8}$$

where α_{ip} is the α value of the p^{th} pixel in the i^{th} layer. The next term is the energy defined by the Laplacian L defining the spatial propagation of information defined in Eq.6:

$$E_L = \sum_i \boldsymbol{\alpha}_i^{\mathsf{T}} L \boldsymbol{\alpha}_i. \tag{9}$$

Finally, we formulate a sparsity term that adapts to the image content. Intuitively, partial opacities come from color transitions in the image because in many cases, it corresponds to a transition between two scene elements, e.g., the fuzzy transition between a teddy bear and the background. We use this observation to build a spatially varying sparsity energy:

$$E_S = \sum_{i,p} |\alpha_{ip}|^{\tilde{\gamma}_p} + |1 - \alpha_{ip}|^{\tilde{\gamma}_p} \tag{10a}$$

$$\text{with: } \tilde{\gamma}_p = \min(0.9 + \|\nabla \boldsymbol{c}_p\|, 1), \tag{10b}$$

where $\nabla \boldsymbol{c}_p$ is the color gradient in the image at pixel p computed using the separable kernels of Farid and Simoncelli [2004]. We design this term such that when $\tilde{\gamma}_p = 1$ on image regions where the gradient is large enough, the energy profile is flat for $\alpha_{ip} \in [0:1]$, i.e. the energy only acts as a penalty on values outside the valid range and lets α_{ip} take any value between 0 and 1. In comparison, in uniform regions where $\nabla \boldsymbol{c}_p \approx 0$, it encourages α_{ip} to be 0 or 1. These two effects combined favor a higher level of sparsity together with the softness of the opacity transitions. The effect of our spatially varying sparsity energy on preserving accurate soft transitions can be seen in Figure 6 (c,d).

Putting these terms together, we get the energy function

$$E = E_L + E_S + E_F + \lambda E_C. \tag{11}$$

A unit weight for each term works well except for the sum-to-one term E_C that represents the soft constraint with a higher weight $\lambda = 100$. Without the sparsity term E_S, E would be a standard least-squares energy function that can be minimized by solving a linear system. To handle E_S, we resort to an iterative reweighted least-squares solver that estimates a solution by solving a series of linear systems. We describe the detail of this approach in the rest of this section.

We name $N_i = 5$ the number of layers, N_p the number of pixels, $\mathbf{a}$ the vector made of all $\boldsymbol{\alpha}_i$'s and $\hat{\mathbf{a}}$ the vector made of all $\hat{\boldsymbol{\alpha}}_i$'s. The dimensionality of $\mathbf{a}$ and $\hat{\mathbf{a}}$ is $N_{ip} = N_i N_p$. For clarity, we also introduce the $N_{ip} \times N_{ip}$ identity matrix $\mathcal{I}$. With this notation, we rewrite E_F (Eq. 7) in matrix form:

$$E_F = (\mathbf{a} - \hat{\mathbf{a}})^\mathsf{T} \mathcal{I} (\mathbf{a} - \hat{\mathbf{a}}) \tag{12}$$

We included the redundant $\mathcal{I}$ in this equation for a clearer transition when deriving Eq. 17. For rewriting E_C (Eq. 8), we introduce the $N_i \times N_{ip}$ matrix C made by concatenating N_i identity matrices horizontally, the vector $\mathbf{1}_i$ made of N_i ones, and the vector $\mathbf{1}_{ip}$ made of N_{ip} ones:

$$E_C = (\mathbf{1}_i - C\mathbf{a})^2 = \mathbf{a}^\mathsf{T} C^\mathsf{T} C \mathbf{a} - \mathbf{a}^\mathsf{T} C^\mathsf{T} \mathbf{1}_i - \mathbf{1}_i^\mathsf{T} C \mathbf{a} + \mathbf{1}_i^\mathsf{T} \mathbf{1}_i \tag{13a}$$

$$= \mathbf{a}^\mathsf{T} C^\mathsf{T} C \mathbf{a} - 2\mathbf{a}^\mathsf{T} \mathbf{1}_{ip} + N_i, \tag{13b}$$

where we used $\mathbf{a}^\mathsf{T} C^\mathsf{T} \mathbf{1}_i = \mathbf{1}_i^\mathsf{T} C \mathbf{a}$, $C^\mathsf{T} \mathbf{1}_i = \mathbf{1}_{ip}$, and $\mathbf{1}_i^\mathsf{T} \mathbf{1}_i = N_i$. We then rewrite E_L:

$$E_L = \mathbf{a}^\mathsf{T} \mathcal{L} \mathbf{a} \tag{14a}$$

$$\text{with: } \mathcal{L} = \begin{bmatrix} L & 0 & \cdots & 0 \\ 0 & L & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & L \end{bmatrix}. \tag{14b}$$

For the sparsity term E_S, we introduce the approximate energy:

$$\tilde{E}_S = \sum_{i,p} u_{ip} (\alpha_{ip})^2 + v_{ip} (1 - \alpha_{ip})^2 \tag{15a}$$

$$\text{with: } u_{ip} = |\alpha'_{ip}|^{\tilde{\gamma}_p - 2} \quad \text{and} \quad v_{ip} = |1 - \alpha'_{ip}|^{\tilde{\gamma}_p - 2}, \tag{15b}$$

where α' is equal to the constrained sparsification result at the first iteration and to the solution of the previous iteration later. For the matrix reformulation, we use $\mathcal{D}_u$ the diagonal matrix built with the u_{ip} values, and $\mathbf{v}$ and $\mathcal{D}_v$ the vector and diagonal matrix built with the v_{ip} values:

$$\tilde{E}_S = \mathbf{a}^\mathsf{T} \mathcal{D}_u \mathbf{a} + (\mathbf{1}_{ip} - \mathbf{a})^\mathsf{T} \mathcal{D}_v (\mathbf{1}_{ip} - \mathbf{a}) \tag{16a}$$

$$= \mathbf{a}^\mathsf{T} (\mathcal{D}_u + \mathcal{D}_v) \mathbf{a} - 2\mathbf{a}^\mathsf{T} \mathbf{v} + \mathbf{1}_{ip}^\mathsf{T} \mathbf{v}, \tag{16b}$$

where we used $\mathcal{D}_v \mathbf{1}_{ip} = \mathbf{v}$ and $\mathbf{v}^\mathsf{T} \mathbf{a} = \mathbf{a}^\mathsf{T} \mathbf{v}$.

To derive a linear system, we sum all the energy terms in their matrix forms and write that the derivative with respect to $\mathbf{a}$ should be zero at a minimum. This leads to:

$$(\mathcal{L} + \mathcal{D}_u + \mathcal{D}_v + \mathcal{I} + \lambda C^\mathsf{T} C) \mathbf{a} = \mathbf{v} + \hat{\mathbf{a}} + \lambda \mathbf{1}_{ip} \tag{17}$$

We solve this equation using preconditioned conjugate gradient optimization [Barrett et al. 1994]. In our experiments, 20 iterations generate results with satisfactory sparsity. Figure 6 illustrates the benefits of our approach.

The size of the linear system is $N_i N_p$. While this is large, it remains tractable because the number of soft layers N_i is set to 5 and it is close to being block-diagonal, the only coefficients outside the diagonal coming from the sum-to-one term E_C that contributes $C^\mathsf{T} C$ to the system. Since C is made of 5 juxtaposed $N_p \times N_p$ identity matrices, $C^\mathsf{T} C$ is made of 25 $N_p \times N_p$ identity matrices in a 5×5 layout, i.e. it is very sparse and is easily handled by the solver.

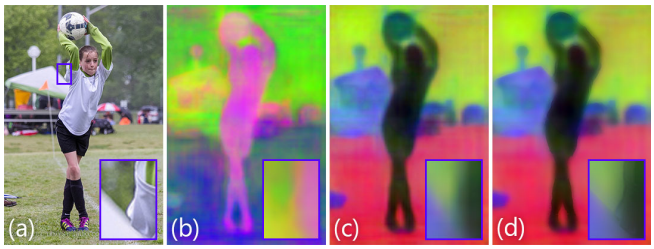

Fig. 8. We first generate a 128-dimensional feature vector per pixel for a given image (a). A random projection of 128 dimensions to 3 is shown in (b). We reduce the dimensionality of the features to 3 using principle component analysis per image (c). Before the dimensionality reduction, we edge-align the features with guided filter. Image from [Lin et al. 2014].

3.5 Semantic Feature Vectors

We defined our semantic affinity term (§ 3.3) with feature vectors $\boldsymbol{f}$ that are similar for pixels on the same object and dissimilar for pixels on different objects. Such vectors can be generated using different network architectures trained for semantic segmentation. In our implementation, we have combined a semantic segmentation approach with a network for metric learning. It should be noted that we do not claim the feature generation as a contribution and we only summarize the solution that we used in this section. A detailed description is provided in the supplementary material.

The base network of our feature extractor is based on DeepLab-ResNet-101 [Chen et al. 2017], but it is trained with a metric learning approach [Hoffer and Ailon 2015] to maximize the L2 distance between the features of different objects. We combine features at multiple stages of the network, motivated by Hariharan et al. [2015] and Bertasius et al. [2015], essentially combining the mid-level and high-level features together. Instead of using all the pixels of an image while training, we generate the features for all pixels but use only a set of randomly-sampled features to update the network. The network minimizes the distance between the features of samples having same ground-truth classes, and maximizes the distance otherwise. Since we only use this cue, i.e. 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 is suitable for 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. To utilize more data with computational efficiency, we use a slightly modified version of N-pair loss [Sohn 2016].

We train this network on the semantic segmentation task of the COCO-Stuff dataset [Caesar et al. 2016]. We refine the feature map generated by this network to be well-aligned to image edges using the guided filter [He et al. 2013]. We then use principal component analysis (PCA) to reduce the dimensionality to three. These preprocessing steps are visualized in Figure 8. While the original 128-dimensional vectors provide a good coverage of all the content we may encounter, each image only exhibits a small portion of it and reducing the dimensionality accordingly results in better accuracy per dimension. Finally, we normalize the vectors to take values in $[0, 1]$. This makes it easier to set parameters, especially in case of changing feature vector definitions. For all the results we present, we set a_s and b_s in Eq. 5 to be 20 and 0.2, respectively.

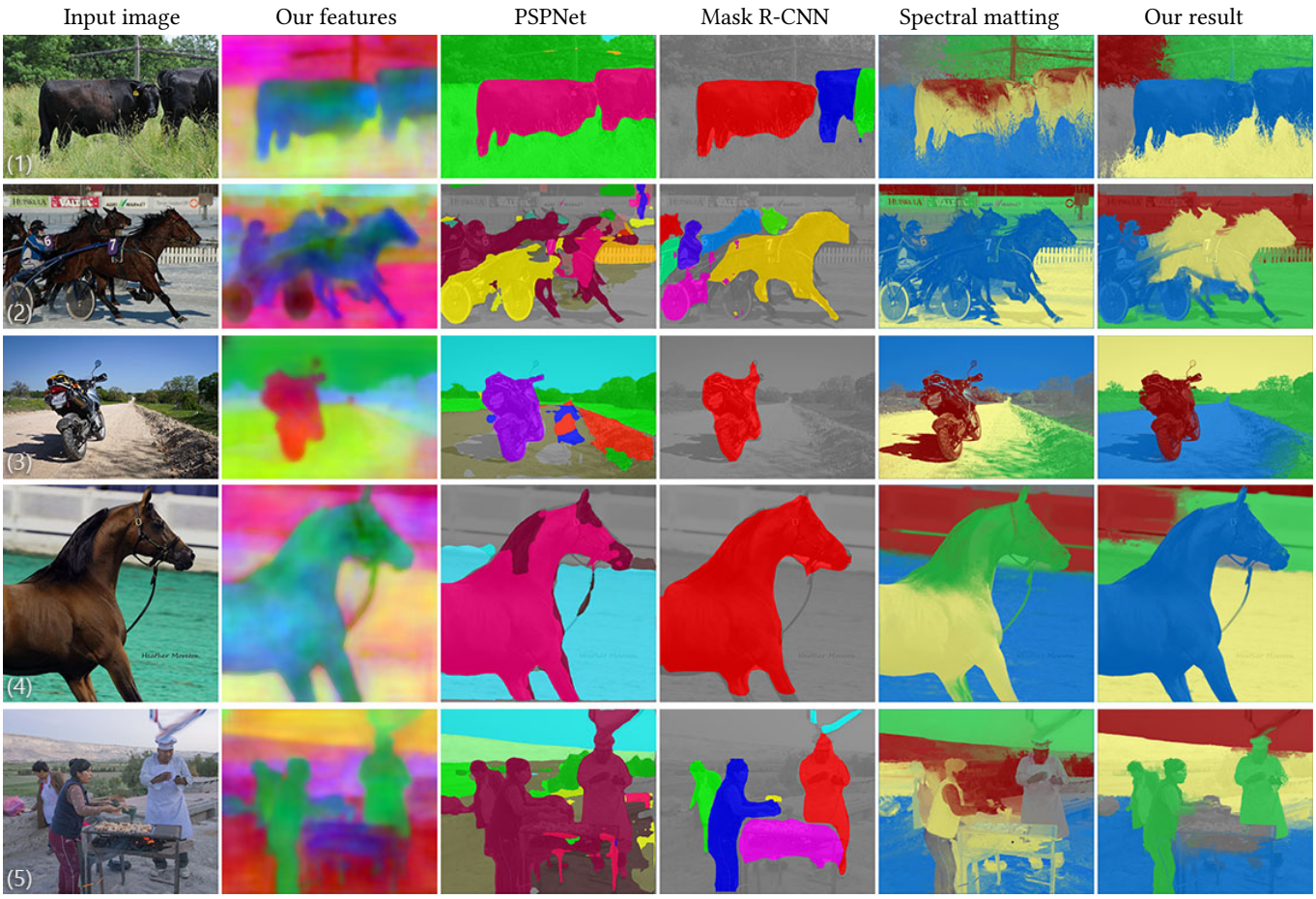

Fig. 9. We show our results together with that of Zhao et al. [2017] (PSPNet), He et al. [2017] (Mask R-CNN), and spectral matting [Levin et al. 2008b]. The segmentations are overlayed onto the grayscale version of the image for a better evaluation around segment boundaries. Notice the inaccuracies of PSPNet and Mask R-CNN around object boundaries, and the soft segments of spectral matting extending beyond object boundaries. Images from [Lin et al. 2014].

3.6 Implementation Details

We use the sparse eigendecomposition and direct solver available in MATLAB for our proof-of-concept implementation for the constrained sparsification stage of our algorithm. This step takes around 3 minutes for a 640×480 image. The relaxed sparsification step uses the preconditioned conjugate gradient optimization implementation of MATLAB. Each iteration typically converges in 50 to 80 iterations and the process takes around 30 seconds. The run-time of our algorithm grows linearly with the number of pixels.

4 EXPERIMENTAL ANALYSIS

Semantic soft segmentation, being at the intersection of semantic segmentation, natural image matting, and soft segmentation, is challenging to evaluate numerically. Semantic segmentation datasets provide binary labeling that is not always pixel-accurate, which makes them ill-suited for benchmarking semantic soft segmentation. Natural image matting methods are typically evaluated on dedicated benchmarks [Rhemann et al. 2009] and datasets [Xu et al. 2017]. These benchmarks are designed to evaluate methods that make use of a secondary input, called trimap, which defines the expected foreground and background, and an uncertain region. Further, the semantic aspect of our work is beyond the scope of these benchmarks. Soft color segmentation, on the other hand, is a problem that lacks a solid definition of ground truth. Although Aksoy et al. [2017b] proposed several blind metrics for evaluation, they are specifically designed for soft *color* segmentation and also ignores semantic aspects. As a result, we resort to qualitative comparisons with related methods and discuss the characteristic differences between the various approaches.

4.1 Spectral Matting and Semantic Segmentation

In Figures 9 and 10, we show our results together with that of spectral matting [Levin et al. 2008b] as the most related soft segmentation method to ours, and two state-of-the-art methods for semantic segmentation: the scene parsing method by Zhao et al. [2017] (PSPNet) and the instance segmentation method by He et al. [2017] (Mask

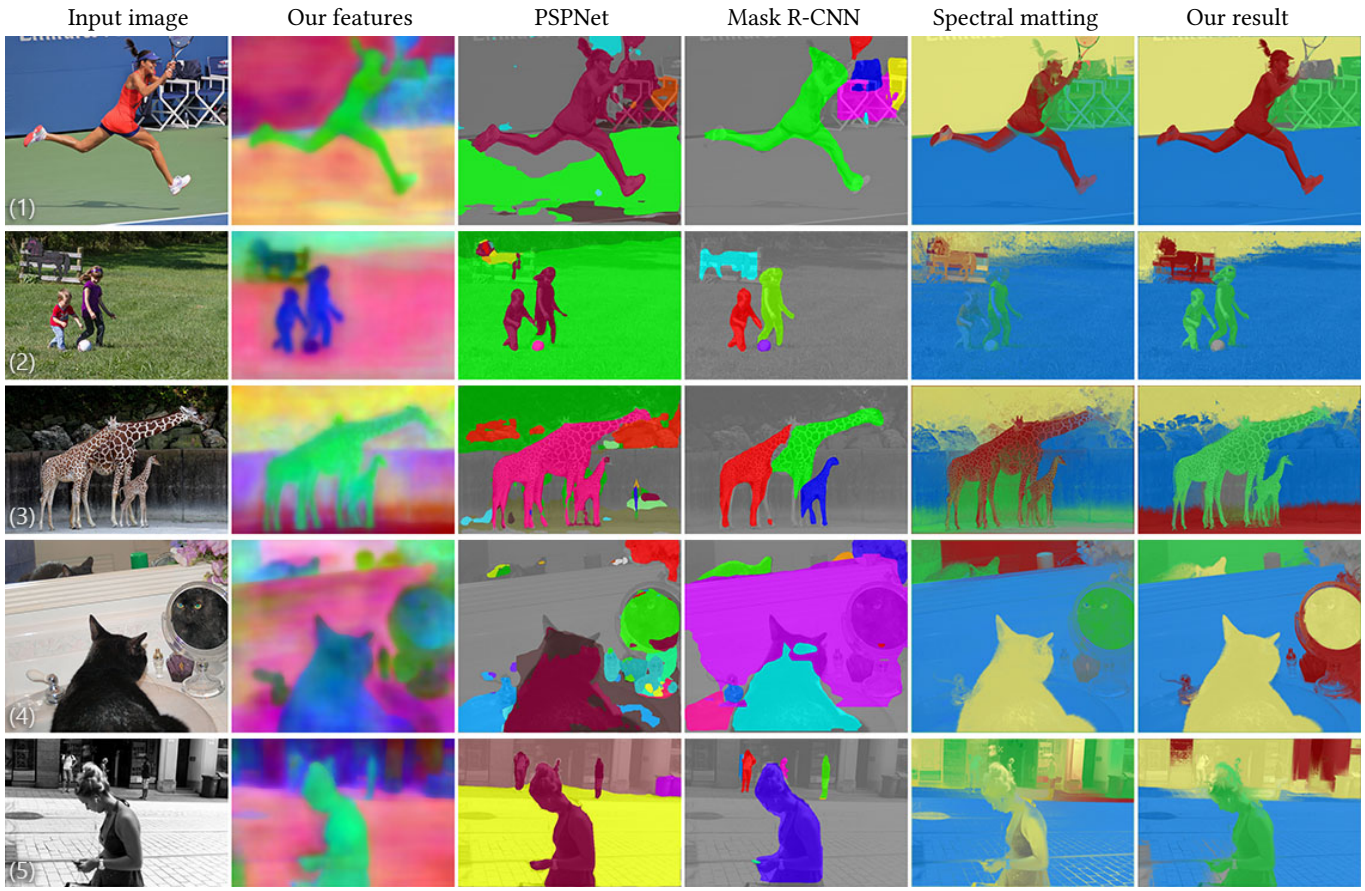

Fig. 10. We show our results together with that of Zhao et al. [2017] (PSPNet), He et al. [2017] (Mask R-CNN), and spectral matting [Levin et al. 2008b]. The segmentations are overlayed onto the grayscale version of the image for a better evaluation around segment boundaries. Notice the inaccuracies of PSPNet and Mask R-CNN around object boundaries, and the soft segments of spectral matting extending beyond object boundaries. Images from [Lin et al. 2014].

R-CNN). More of these comparisons are available in the supplementary material. Spectral matting generates around 20 soft segments per image, and provides several alternative foreground mattes by combining the soft segments to maximize an *objectness score*. These mattes are not definite results but are provided to the user as options, and showing all 20 segments would make the comparisons harder to evaluate. Instead, we apply our soft segment grouping method that uses the semantic features to the results of spectral matting.

The presented examples show that semantic segmentation methods, while being successful in recognizing and locating the objects in the image, suffer from low accuracy around the edges of the objects. While their accuracy is satisfactory for the task of the semantic segmentation, errors around object edges are problematic for image editing or compositing applications. On the other end of the spectrum, spectral matting is able to successfully capture most of the soft transitions around the objects. However, due to the lack of semantic information, their segments often cover multiple objects at once, and the alpha values are often not sparse for any given object. In comparison, our method captures objects in their entirety or subparts of them without grouping unrelated objects and achieves a high accuracy at edges, including soft transitions when appropriate.

It should be noted that it is not uncommon for our method to represent the same object in multiple segments such as the horse carriage in Figure 9 (2) or the background fence in Figure 9 (4). This is mainly due to the preset number of layers, five, sometimes exceeds the number of meaningful regions in the image. Some small objects may be missed in the final segments despite being detected by the semantic features, such the people in the background in Figure 10 (5). This is due to the fact that, especially when the color of the object is similar to the surroundings, the objects do not appear well-defined in the eigenvectors and they end up being merged into closeby segments. Our semantic features are not instance-aware, i.e. the features of two different objects of the same class are similar. This results in multiple objects being represented in the same layer such as the cows in Figure 9 (1), the people in Figure 9 (5) or the giraffes in Figure 10 (3). With instance-aware features, however, our method would be capable of generating separate soft segments for different instances of objects.

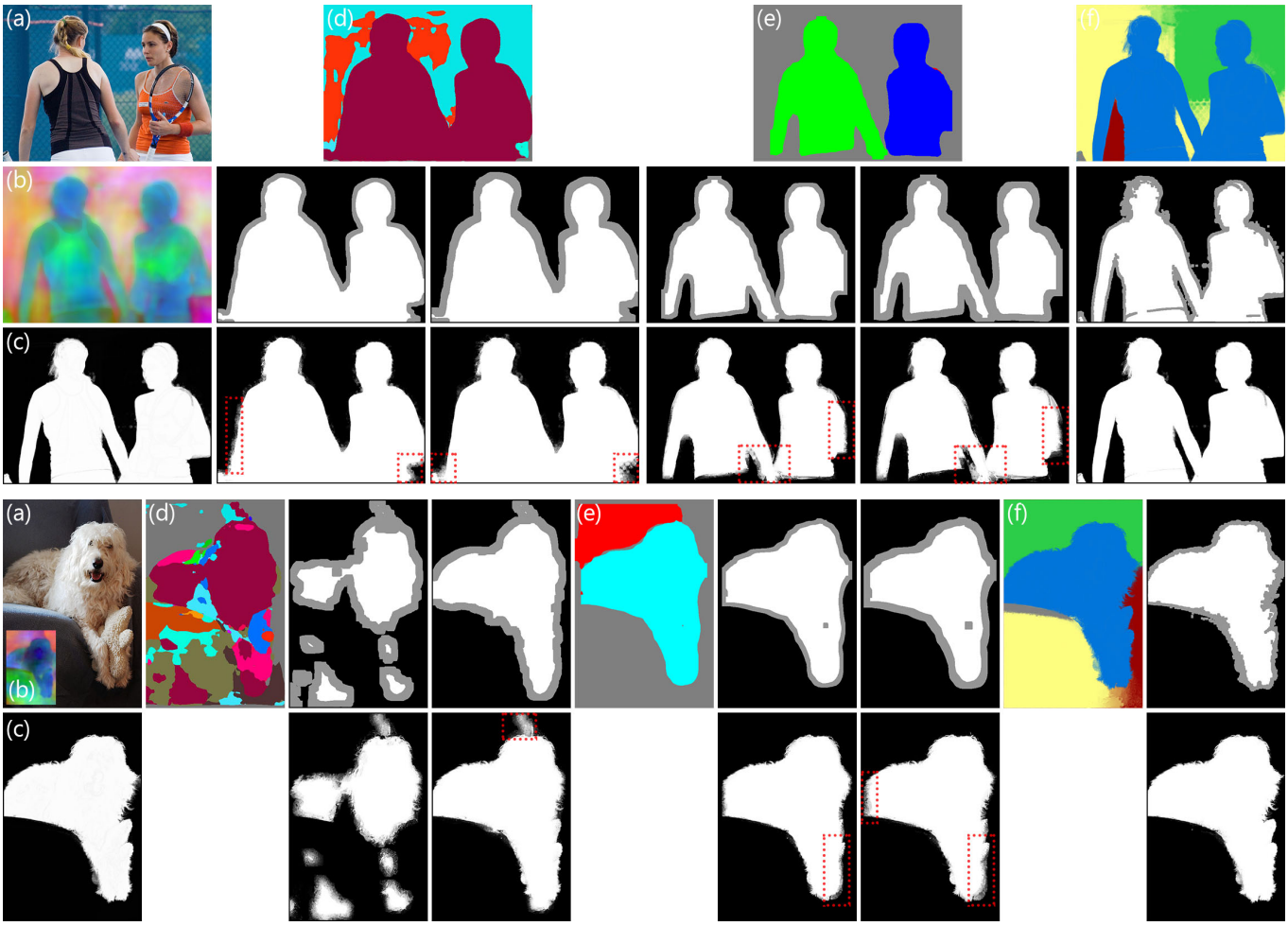

Fig. 11. From the input image (a) and our feature vectors (b), our method generates the matte shown in (c). We show that the trimaps with different unknown region widths, generated using the semantic segments by PSPNet [Zhao et al. 2017] (d) or Mask R-CNN [He et al. 2017] (e), fail to provide foreground and background regions reliably, which affects the matting result generated using information-flow matting [Aksoy et al. 2017a] negatively. In the bottom example, PSPNet trimaps are generated by selecting a single class (left) or all the classes that correspond to the object. We also provide the matting result using a trimap generated by our result (f) which demonstrates the performance of the matting algorithm given an accurate trimap. Images from [Lin et al. 2014].

Grayscale images are especially challenging for soft segmentation and image matting methods with the lack of color cues on which such methods typically rely. The performance of semantic segmentation methods, on the other hand, does not degrade substantially when processing a grayscale image. Figure 10 (5) demonstrates that our method can succesfully leverage the semantic information for soft segmentation of a grayscale image.

4.2 Natural Image Matting

In principle, semantic soft segments can be generated by cascading semantic segmentation and natural image matting. The *trimap*, defining the foreground, background, and soft transition regions, can be generated from the semantic hard segments to be fed to the natural matting method. Shen et al. [2016] and Qin et al. [2017] use similar approaches for class-specific problems. We show two examples of such scenario in Figure 11 to demonstrate the shortcomings of this approach by generating trimaps using Mask R-CNN and PSPNet results and estimating the mattes using a state-of-the-art matting method, information-flow matting [Aksoy et al. 2017a]. A strong assumption made by natural image matting methods is that the provided trimap is *correct*, i.e. the defined foreground and background regions are used as hard constraints to guide the methods in modeling the layer colors. Inaccuracies in the estimated semantic boundaries, however, often fails to provide reliable trimaps even with a large unknown-region width. This results in severe artifacts in the matting results, as highlighted in the figure. We show that the natural matting method succeeds given an accurate trimap, generated using our results for demonstration.

While general natural image matting is beyond the scope of our method, Figure 12 shows several examples where our method is able to generate satisfactory results on images from natural image matting datasets without requiring a trimap.

Fig. 12. Our soft segments and the corresponding mattes for the foreground objects. Note that trimaps usually provided for natural matting were not used to produce these results. Images from [Xu et al. 2017].

Fig. 13. Semantic soft segments by the proposed method and soft color segments by Aksoy et al. [2017b] shown together for conceptual comparison. Both methods are fully automated and only require the input image for soft segmentation. Image from [Bychkovsky et al. 2011].

4.3 Soft Color Segmentation

Soft color segmentation, a concept originally proposed by Tai et al. [2007], decomposes the input image into soft layers of homogeneous colors and have been shown to be useful for image editing and recoloring applications. As a conceptual comparison between semantic soft segments and soft color segments, Figure 13 shows our segments with that of unmixing-based soft color segmentation [Aksoy et al. 2017b]. For a more convenient qualitative comparison, we estimated the layer colors for our soft segments using the closed-form color estimation method [Levin et al. 2008a].

It is immediately visible that the content of soft color segments extend beyond the object boundaries, while our results show semantically meaningful objects in the same segment, regardless of their color content. As these representations are orthogonal to each other, they can be used in orchestration to generate targeted recoloring results.

Fig. 14. We show our soft segmentation results together with image editing results that were generated using per-layer operations or simple compositions to demonstrate the use of our segmentation in targeted image editing tasks. Images from [Bychkovsky et al. 2011] (1) and by Death to the Stock Photo (2,7).

4.4 Using Semantic Soft Segments for Image Editing

We demonstrate several use cases of our soft segments for targeted image editing and compositing in Figure 14. Figure 14(1,3,4,7) show compositing results where we estimated the layer colors for our segments using closed-form layer color estimation [Levin et al. 2008a]. Notice the natural soft transitions between the selected foreground layers and the novel background. The soft segments can also be used for targeted image edits where they are used to define masks for specific adjustment layers such as adding motion blur to the train in (2), color grading the people and the backgrounds separately in (5,6) and separate stylization of the hot-air balloon, sky, terrain and the person in (8). While these edits can be done via user-drawn masks or natural matting algorithms, our representation provides a convenient intermediate image representation to make the targeted edits effortless for the artist.

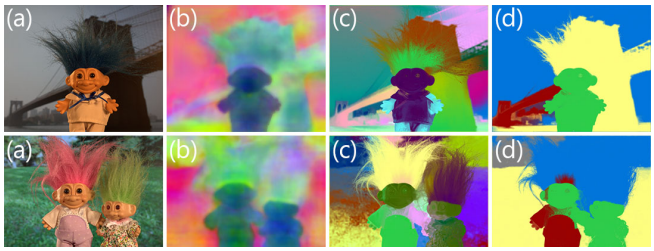

Fig. 15. Two failure cases are shown. Top example: In case of large regions covering different objects with very similar colors (a) our feature vectors (b) and segments before grouping (c) fail to identify the separate objects in the image and result in inaccurate segmentation (d). Bottom example: When our feature vectors fail to represent the objects, even if when the initial layers are able to generate accurate soft transitions (c) the grouping of the soft segments (d) may fail. Images from [Rhemann et al. 2009].

5 LIMITATIONS AND FUTURE WORK

While we are able to generate accurate soft segmentations of images, in our prototype implementation our solvers are not optimized for speed. As a result, our runtime for a 640×480 image lies between 3 and 4 minutes. The efficiency of our method can be optimized in several ways, such as multi-scale solvers, but an efficient implementation of linear solvers and eigendecomposition lies beyond the scope of our paper.

In the constrained sparsification step, we generate around 15-25 segments, which are then grouped using the feature vectors into 5. The number of layers was set via empirical observations, and in some cases, an object may be divided into several layers. While this does not affect the applicability of our method as combining those layers in editing is trivial, more sophisticated ways of grouping the layers such as through recognition and classification can be devised.

Our method does not generate separate layers for different instances of the same class of objects. This is due to our feature vectors, which does not provide instance-aware semantic information. Our soft segmentation formulation, however, is agnostic to the semantic features. Hence, a more advanced feature generator would make it possible to generate instance-level soft segmentation results when combined with a better-fitting segment-grouping strategy.

We have shown several results from natural matting datasets. However, it should be noted that we do not aim to solve the natural matting problem in general. Natural matting is a mature field with many specific challenges, such as generating accurate mattes around very similarly-colored foreground and background regions, and state-of-the-art methods depend on the color distributions of the two regions to increase performance around such areas. As Figure 15 demonstrates, our method may fail at the initial constrained sparsification step when the object colors are very similar, or the grouping of soft segments may fail due to unreliable semantic feature vectors around large transition regions.

6 CONCLUSION

We have proposed a method that generates soft segments that correspond to semantically meaningful regions in the image by fusing the high-level information from a neural network with low-level image features fully automatically. We have shown that by carefully defining affinities between different regions in the image, the soft segments with the semantic boundaries can be revealed by spectral analysis of the constructed Laplacian matrix. The proposed relaxed sparsification method for the soft segments can generate accurate soft transitions while also providing a sparse set of layers. We have demonstrated that while semantic segmentation and spectral soft segmentation methods fail to provide layers that are accurate enough for image editing tasks, our soft segments provide a convenient intermediate image representation that makes several targeted image editing tasks trivial, which otherwise require the manual labor of a skilled artist.

ACKNOWLEDGMENTS

We would like to thank Tunç Ozan Aydın and Adrien Bousseau for their feedback on the text, and James Minor for the voice-over of the accompanying video.

Y. Aksoy and T.-H. Oh were supported by QCRI-CSAIL Computer Science Research Program at MIT.

REFERENCES

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. 2012. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Trans. Pattern Anal. Mach. Intell.* 34, 11 (2012), 2274–2282.

Yağız Aksoy, Tunç Ozan Aydın, and Marc Pollefeys. 2017a. Designing Effective Inter-Pixel Information Flow for Natural Image Matting. In *Proc. CVPR.*

Yağız Aksoy, Tunç Ozan Aydın, Marc Pollefeys, and Aljoša Smolić. 2016. Interactive High-Quality Green-Screen Keying via Color Unmixing. *ACM Trans. Graph.* 35, 5 (2016), 152:1–152:12.

Yağız Aksoy, Tunç Ozan Aydın, Aljoša Smolić, and Marc Pollefeys. 2017b. Unmixing-Based Soft Color Segmentation for Image Manipulation. *ACM Trans. Graph.* 36, 2 (2017), 19:1–19:19.

Xiaobo An and Fabio Pellacini. 2008. AppProp: All-pairs Appearance-space Edit Propagation. *ACM Trans. Graph.* 27, 3 (2008), 40:1–40:9.

R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. 1994. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM.

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.*

V. Bychkovsky, S. Paris, E. Chan, and F. Durand. 2011. Learning Photographic Global Tonal Adjustment with a Database of Input/Output Image Pairs. 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).

Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang. 2013. KNN Matting. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 9 (2013), 2175–2188.

Xiaowu Chen, Dongqing Zou, Qinping Zhao, and Ping Tan. 2012. Manifold Preserving Edit Propagation. *ACM Trans. Graph.* 31, 6 (2012), 132:1–132:7.

Yuki Endo, Satoshi Iizuka, Yoshihiro Kanamori, and Jun Mitani. 2016. DeepProp: Extracting Deep Features from a Single Image for Edit Propagation. *Comput. Graph. Forum* 35, 2 (2016), 189–201.

D. Eynard, A. Kovnatsky, and M. M. Bronstein. 2014. Laplacian colormaps: a framework for structure-preserving color transformations. *Comput. Graph. Forum* 33, 2 (2014), 215–224.

H. Farid and E. P. Simoncelli. 2004. Differentiation of discrete multidimensional signals. *IEEE Trans. Image Process.* 13, 4 (2004), 496–508.

Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P. Murphy. 2017. Semantic Instance Segmentation via Deep Metric Learning. `arXiv:1703.10277 [cs.CV]` (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.

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

Anat Levin, Dani Lischinski, and Yair Weiss. 2008a. A Closed-Form Solution to Natural Image Matting. *IEEE Trans. Pattern Anal. Mach. Intell.* 30, 2 (2008), 228–242.

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

Y. Li, E. Adelson, and A. Agarwala. 2008. ScribbleBoost: Adding Classification to Edge-Aware Interpolation of Local Image and Video Adjustments. *Comput. Graph. Forum* 27, 4 (2008), 1255–1264.

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.*

Tae-Hyun Oh, Kyungdon Joo, Neel Joshi, Baoyuan Wang, In So Kweon, and Sing Bing Kang. 2017. Personalized Cinemagraphs Using Semantic Understanding and Collaborative Learning. In *Proc. ICCV.*

S. Qin, S. Kim, and R. Manduchi. 2017. Automatic skin and hair masking using fully convolutional networks. In *Proc. ICME.*

Wenqi Ren, Jinshan Pan, Xiaochun Cao, and Ming-Hsuan Yang. 2017. Video Deblurring via Semantic Segmentation and Pixel-Wise Non-Linear Kernel. In *Proc. ICCV.*

Christoph Rhemann, Carsten Rother, Jue Wang, Margrit Gelautz, Pushmeet Kohli, and Pamela Rott. 2009. A Perceptually Motivated Online Benchmark for Image Matting. In *Proc. CVPR.*

Xiaoyong Shen, Xin Tao, Hongyun Gao, Chao Zhou, and Jiaya Jia. 2016. Deep Automatic Portrait Matting. In *Proc. ECCV.*

D. Singaraju and R. Vidal. 2011. Estimation of Alpha Mattes for Multiple Image Layers. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 7 (2011), 1295–1309.

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

Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. 2007. Soft Color Segmentation and Its Applications. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 9 (2007), 1520–1537.

Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. 2016. Decomposing Images into Layers via RGB-space Geometry. *ACM Trans. Graph.* 36, 1 (2016), 7:1–7:14.

Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. 2017. Deep Image Matting. In *Proc. CVPR.*

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

Received January 2018; final version May 2018; accepted May 2018

Semantic Soft Segmentation Supplementary Material

YAĞIZ AKSOY, MIT CSAIL and ETH Zürich
TAE-HYUN OH, MIT CSAIL
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 `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.

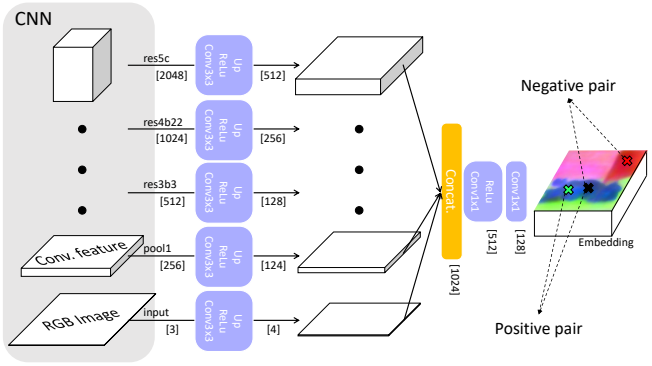

Fig. 1. Our network architecture. We extract the feature of intermediate representation of the base convolution neural network (we use the DeepLap variant of ResNet-101). The feature are compressed by 3×3 convolution followed by ReLu and bilinear up-sampling to have the same resolution with input. The 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 [#].

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 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

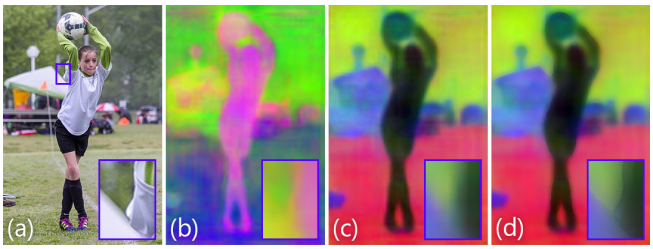

Fig. 2. We first generate a 128-dimensional feature vector per pixel for a given image (a). A random projection of 128 dimensions to 3 is shown in (b). We reduce the dimensionality of the features to 3 using principle component analysis per image (c). In order to align the feature vectors to the image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

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).

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*.

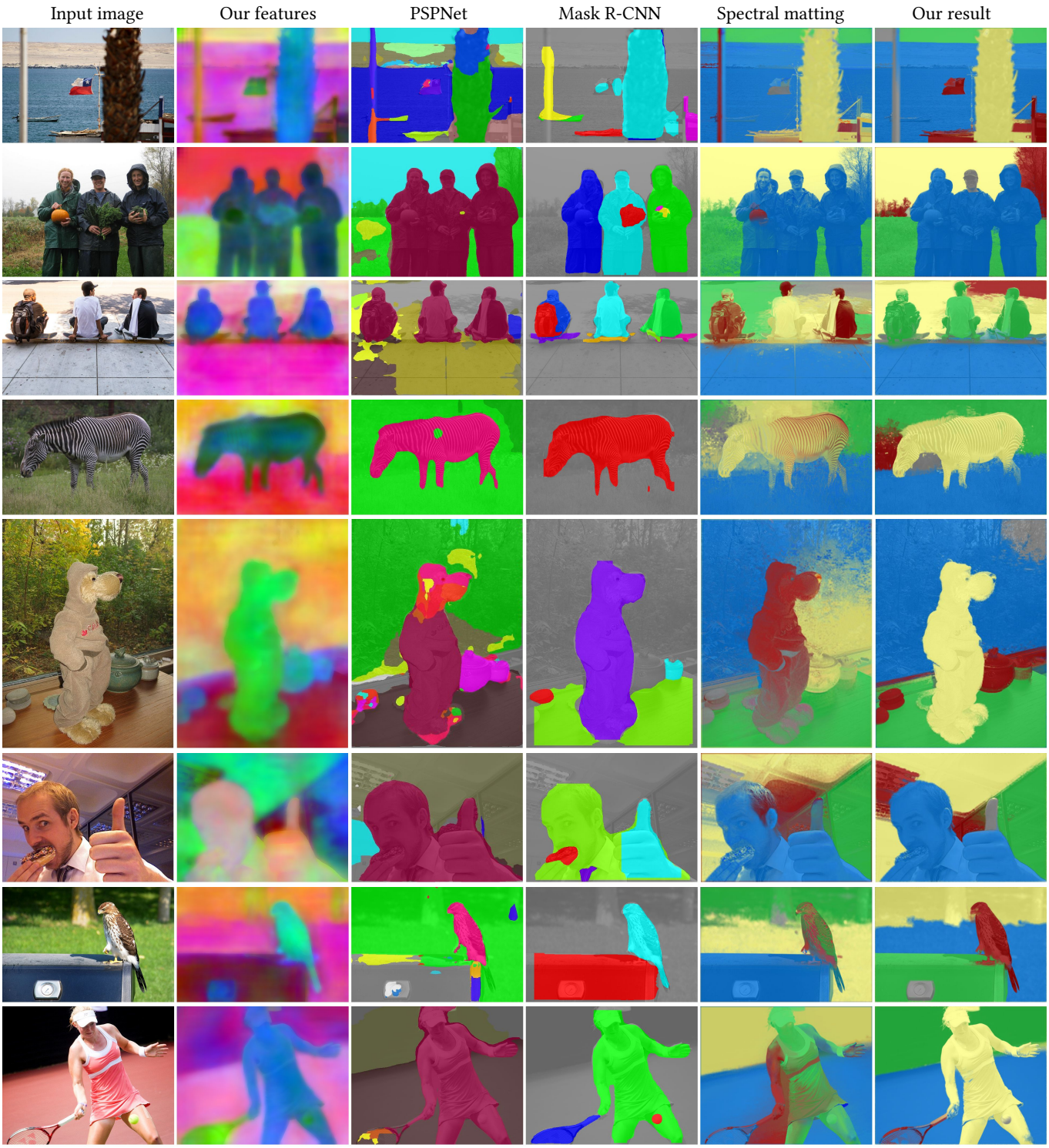

Fig. 3. We show our results together with that of Zhao *et al.* [2017] (PSPNet), He *et al.* [2017] (Mask R-CNN), and spectral matting [Levin et al. 2008]. The segmentations are overlayed onto the grayscale version of the input image for a better evaluation around segment boundaries. Notice the inaccuries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

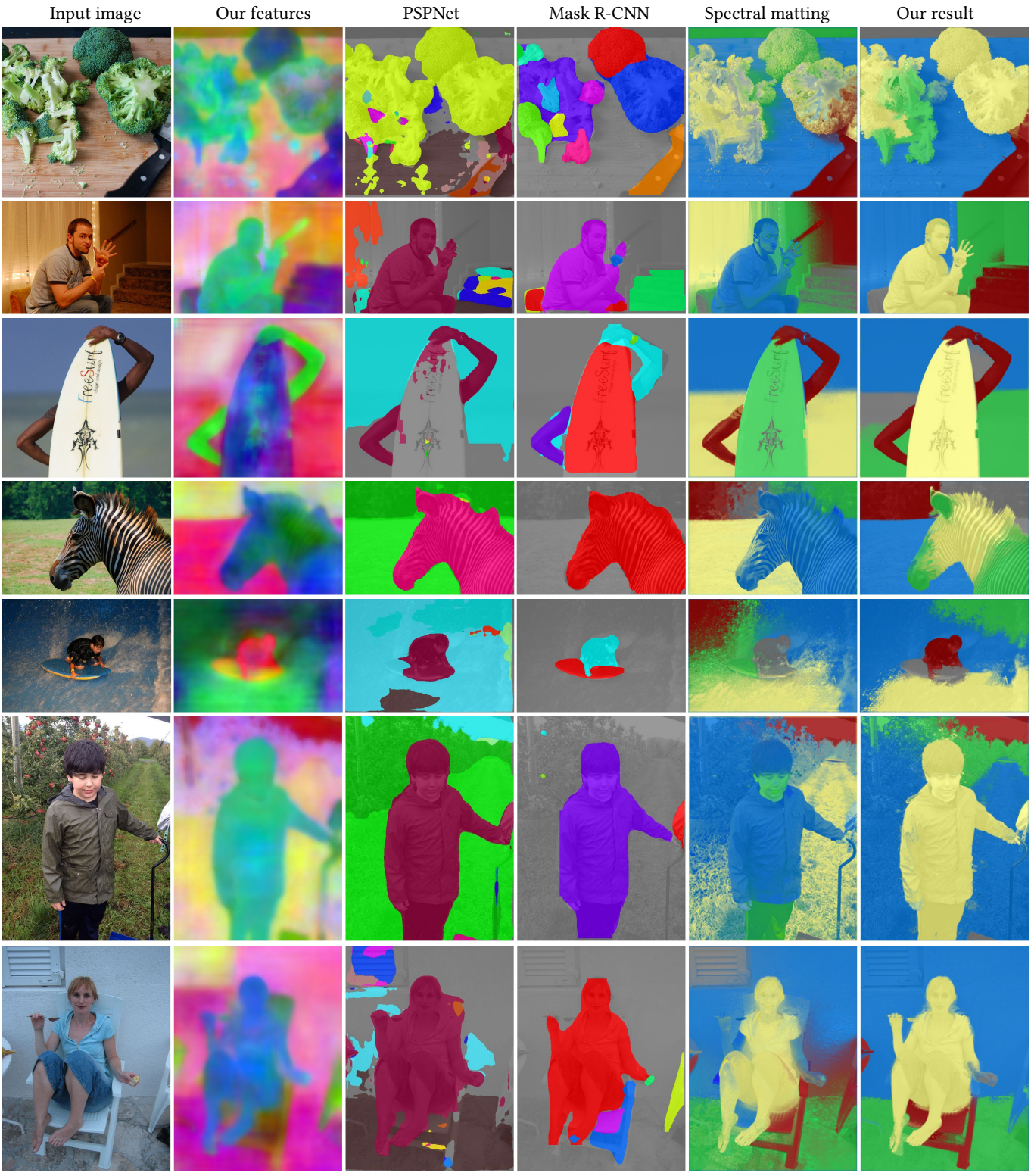

Fig. 4. We show our results together with that of Zhao *et al.* [2017] (PSPNet), He *et al.* [2017] (Mask R-CNN), and spectral matting [Levin et al. 2008]. The segmentations are overlayed onto the grayscale version of the input image for a better evaluation around segment boundaries. Notice the inaccuries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

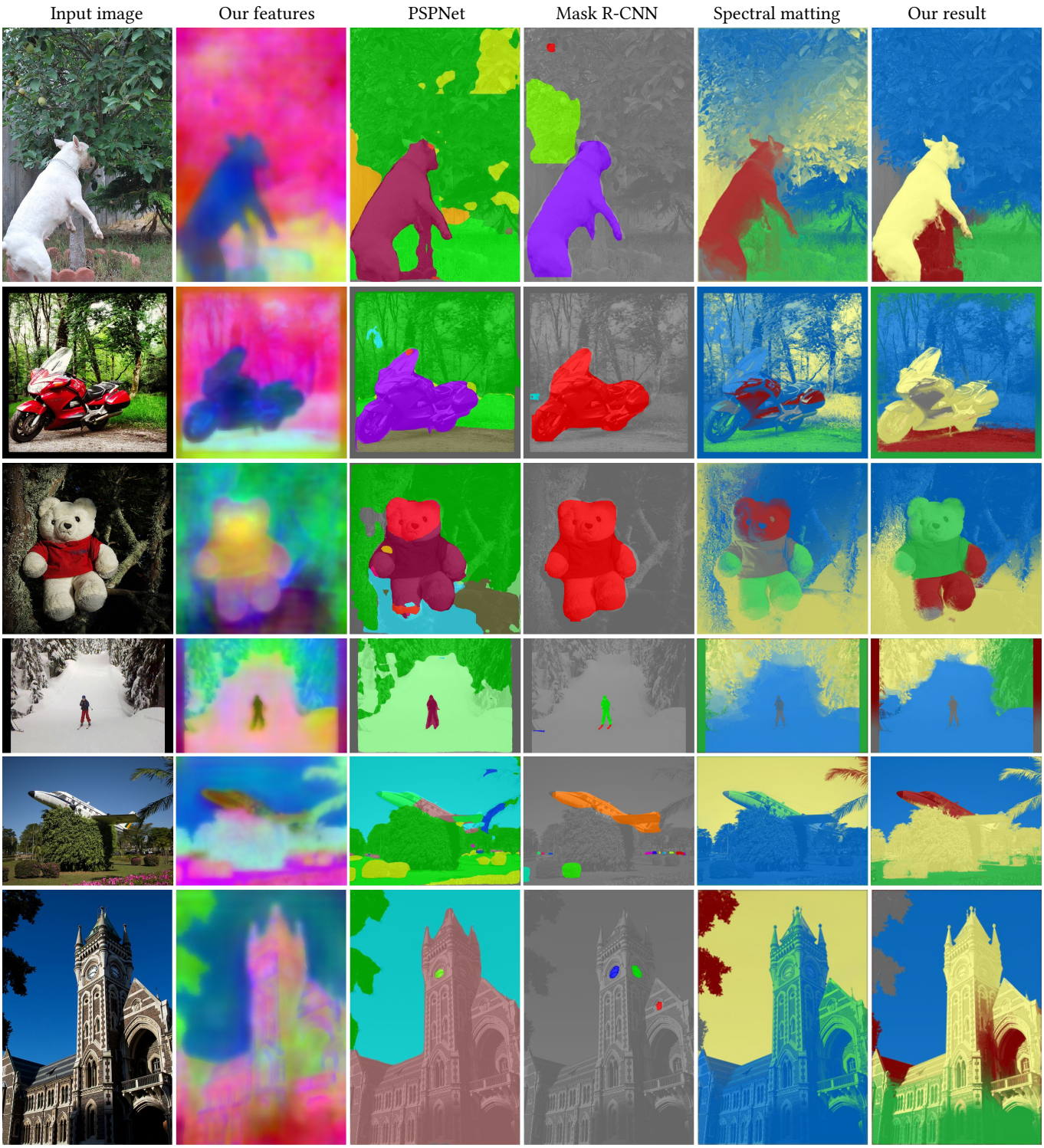

Fig. 5. We show our results together with that of Zhao *et al.* [2017] (PSPNet), He *et al.* [2017] (Mask R-CNN), and spectral matting [Levin et al. 2008]. The segmentations are overlayed onto the grayscale version of the input image for a better evaluation around segment boundaries. Notice the inaccuries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

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*.