Superpixel Segmentation with Fully Convolutional Networks

Fengting Yang Qian Sun
The Pennsylvania State University
fuy34@psu.edu, uestcqs@gmail.com

Hailin Jin
Adobe Research
hljin@adobe.com

Zihan Zhou
The Pennsylvania State University
zzhou@ist.psu.edu

Abstract

In computer vision, superpixels have been widely used as an effective way to reduce the number of image primitives for subsequent processing. But only a few attempts have been made to incorporate them into deep neural networks. One main reason is that the standard convolution operation is defined on regular grids and becomes inefficient when applied to superpixels. Inspired by an initialization strategy commonly adopted by traditional superpixel algorithms, we present a novel method that employs a simple fully convolutional network to predict superpixels on a regular image grid. Experimental results on benchmark datasets show that our method achieves state-of-the-art superpixel segmentation performance while running at about 50fps. Based on the predicted superpixels, we further develop a downsampling/upsampling scheme for deep networks with the goal of generating high-resolution outputs for dense prediction tasks. Specifically, we modify a popular network architecture for stereo matching to simultaneously predict superpixels and disparities. We show that improved disparity estimation accuracy can be obtained on public datasets.

1. Introduction

In recent years, deep neural networks (DNNs) have achieved great success in a wide range of computer vision applications. The advance of novel neural architecture design and training schemes, however, often comes a greater demand for computational resources in terms of both memory and time. Consider the stereo matching task as an example. It has been empirically shown that, compared to traditional 2D convolution, 3D convolution on a 4D volume (height$\times$width$\times$disparity$\times$feature channels) [17] can better capture context information and learn representations for each disparity level, resulting in superior disparity estimation results. But due to the extra feature dimension, 3D convolution is typically operating on spatial resolutions that are lower than the original input image size for the time and memory concern. For example, CSPN [8], the top-1 method on the KITTI 2015 benchmark, conducts 3D convolution at $1/4$ of the input size and uses bilinear interpolation to upsample the predicted disparity volume for final disparity regression. To handle high resolution images (*e.g.*, 2000×3000), HSM [42], the top-1 method on the Middlebury-v3 benchmark, uses a multi-scale approach to compute disparity volume at $1/8$, $1/16$, and $1/32$ of the input size. Bilinear upsampling is again applied to generate disparity maps at the full resolution. In both cases, object boundaries and fine details are often not well preserved in final disparity maps due to the upsampling operation.

In computer vision, superpixels provide a compact representation of image data by grouping perceptually similar pixels together. As a way to effectively reduce the number of image primitives for subsequent processing, superpixels have been widely adopted in vision problems such as saliency detection [41], object detection [32], tracking [37], and semantic segmentation [12]. However, superpixels are yet to be widely adopted in the DNNs for dimension reduction. One main reason is that, the standard convolution operation in the convolutional neural networks (CNNs) is defined on a regular image grid. While a few attempts have been made to modify deep architectures to incorporate superpixels [14, 11, 20, 34], performing convolution over an irregular superpixel grid remains challenging.

To overcome this difficulty, we propose a deep learning method to learn superpixels on the regular grid. Our key insight is that it is possible to associate each superpixel with a regular image grid cell, a strategy commonly used by traditional superpixel algorithms [22, 36, 10, 1, 23, 25, 2] as an initialization step (see Figure 2). Consequently, we cast superpixel segmentation as a task that aims to find association scores between image pixels and regular grid cells, and use a fully convolutional network (FCN) to directly predict such scores. Note that recent work [16] also proposes an end-to-end trainable network for this task, but this method uses a deep network to extract pixel features, which are then fed to a soft K-means clustering module to generate superpixels.

The key motivation for us to choose a standard FCN architecture is its simplicity as well as its ability to generate outputs on the regular grid. With the predicted superpixels, we further propose a general framework for down-

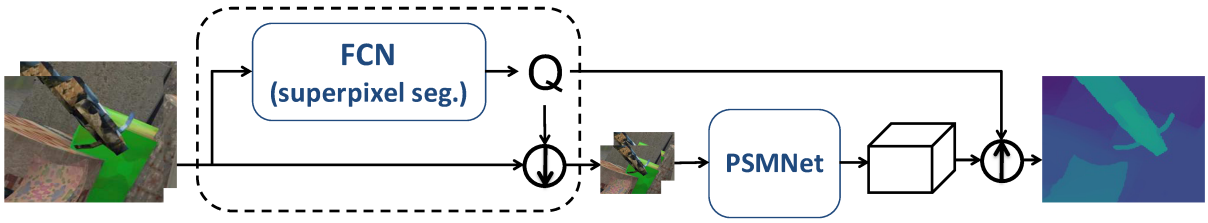


Figure 1. An illustration of our superpixel-based downsampling/upsampling scheme for deep networks. In this figure, we choose PSMNet [7] for stereo matching as our task network. The high-res input images are first downsampled using the superpixel association matrix Q predicted by our superpixel segmentation network. To generate a high-res disparity map, we use the same matrix Q to upsample the low-res disparity volume predicted by PSMNet for final disparity regression.

sampling/upsampling in DNNs. As illustrated in Figure 1, we replace the conventional operations for downsampling (*e.g.*, stride-2 convolutions) and upsampling (*e.g.*, bilinear upsampling) in the task network (PSMNet in the figure) with a superpixel-based downsampling/upsampling scheme to effectively preserve object boundaries and fine details. Further, the resulting network is end-to-end trainable. One advantage of our joint learning framework is that superpixel segmentation is now directly influenced by the downstream task, and that the two tasks can naturally benefit from each other. In this paper, we take stereo matching as an example and show how the popular network PSMNet [7], upon which many of the newest methods such as CSPN [8] and HSM [42] are built, can be adapted into our framework.

We have conducted extensive experiments to evaluate the proposed methods. For superpixel segmentation, experiment results on public benchmarks such as BSDS500 [3] and NYUv2 [28] demonstrate that our method is competitive with or better than the state-of-the-art w.r.t. a variety of metrics, and is also fast (running at about 50fps). For disparity estimation, our method outperforms the original PSMNet on SceneFlow [27] as well as high-res datasets HR-VS [42] and Middlebury-v3 [30], verifying the benefit of incorporating superpixels into downstream vision tasks.

In summary, the **main contributions** of the paper are: 1. We propose a simple fully convolutional network for superpixel segmentation, which achieves state-of-the-art performance on benchmark datasets. 2. We introduce a general superpixel-based downsampling/upsampling framework for DNNs. We demonstrate improved accuracy in disparity estimation by incorporating superpixels into a popular stereo matching network. To the best of our knowledge, we are the first to develop a learning-based method that simultaneously perform superpixel segmentation and dense prediction.

2. Related Work

Superpixel segmentation. There is a long line of research on superpixel segmentation, now a standard tool for many vision tasks. For a thorough survey on existing methods, we refer readers to the recent paper [33]. Here we focus on methods which use a regular grid in the initialization step.

Turbopixels [22] places initial seeds at regular intervals based on the desired number of superpixels, and grows them into regions until superpixels are formed. [36] grows the superpixels by clustering pixels using a geodesic distance that embeds structure and compactness constraints. SEEDS [10] initializes the superpixels on a grid, and continuously refines the boundaries by exchanging pixels between neighboring superpixels.

The SLIC algorithm [1] employs K-means clustering to group nearby pixels into superpixels based on a 5-dimensional positional and CIELAB color features. Variants of SLIC include LSC [23] which maps each pixel into a 10-dimensional feature space and performs weighted K-means, Manifold SLIC [25] which maps the image to a 2-dimensional manifold to produce content-sensitive superpixels, and SNIC [2] which replaces the iterative K-means clustering with a non-iterative region growing scheme.

While the above methods rely on hand-crafted features, recent work [35] proposes to learn pixel affinity from large data using DNNs. In [16], the authors propose to learn pixel features which are then fed to a differentiable K-means clustering module for superpixel segmentation. The resulting method, SSN, is the first end-to-end trainable network for superpixel segmentation. Different from these methods, we train a deep neural network to directly predict the pixel-superpixel association map.

The use of superpixels in deep neural networks. Several methods propose to integrate superpixels into deep learning pipelines. These works typically use pre-computed superpixels to manipulate learnt features so that important image properties (e.g., boundaries) can be better preserved. For example, [14] uses superpixels to convert 2D image patterns into 1D sequential representations, which allows a DNN to efficiently explore long-range context for saliency detection. [11] introduces a "bilateral inception" module which can be inserted into existing CNNs and perform bilateral filtering across superpixels, and [20, 34] employ superpixels to pool features for semantic segmentation. Instead, we use superpixels as an effective way to downsample/upsample. Further, none of these works has attempted to jointly learn superpixels with the downstream tasks.

Besides, our method is also similar to the deformable convolutional network (DCN) [9, 47] in that both can real-

ize adaptive respective field. However, DCN is mainly designed to better handle geometric transformation and capture contextual information for feature extraction. Thus, unlike superpixels, a deformable convolution layer does not constrain that every pixel has to contribute to (thus is represented by) the output features.

Stereo matching. Superpixel- or segmentation-based approach to stereo matching was first introduced in [4], and have since been widely used [15, 5, 19, 38, 6, 13]. These methods first segment the images into regions and fit a parametric model (typically a plane) to each region. In [39, 40], Yamaguchi *et al*. propose an optimization framework to jointly segments the reference image into superpixels and estimates the disparity map. [26] trains a CNN to predict initial pixel-wise disparities, which are refined using the slanted-plane MRF model. [21] develops an efficient algorithm which computes photoconsistency for only a random subset of pixels. Our work is fundamentally different from these optimization-based methods. Instead of fitting parametric models to the superpixels, we use superpixels to develop a new downsampling/upsampling scheme for DNNs.

In the past few years, deep networks [45, 31, 29, 44] taking advantage of large-scale annotated data have generated impressive stereo matching results. Recent methods [17, 7, 8] employing 3D convolution achieve the state-of-the-art performance on public benchmarks. However, due to the memory constraints, these methods typically compute disparity volumes at a lower resolution. [18] bilinearly upsamples the disparity to the output size and refine it using an edge-preserving refinement network. Recent work [42] has also explored efficient high-res processing, but its focus is on generating coarse-to-fine results to meet the need for anytime on-demand depth sensing in autonomous driving applications.

3. Superpixel Segmentation Method

In this section, we introduce our CNN-based superpixel segmentation method. We first present our idea of directly predicting pixel-superpixel association on a regular grid in Section 3.1, followed by a description of our network design and loss functions in Section 3.2. We further draw a connection between our superpixel learning regime with the recent convolutional spatial propagation (CSP) network [8] for learning pixel affinity in Section 3.3. Finally, in Section 3.4, we systematically evaluate our method on public benchmark datasets.

3.1. Learning Superpixels on a Regular Grid

In the literature, a common strategy adopted [22, 36, 10, 1, 23, 25, 2, 16] for superpixel segmentation is to first partition the $H \times W$ image using a regular grid of size $h \times w$ and consider each grid cell as an initial superpixel (i.e., a "seed"). Then, the final superpixel segmentation is obtained by finding a mapping which assigns each pixel $\mathbf{p} = (u, v)$ to one of the seeds $\mathbf{s} = (i, j)$. Mathematically, we can write the mapping as $g_{\mathbf{s}}(\mathbf{p}) = g_{i,j}(u, v) = 1$ if the (u, v)-th pixel belongs to the (i, j)-th superpixel, and 0 otherwise.

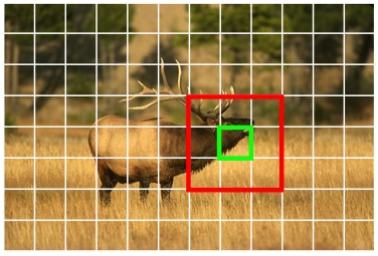


Figure 2. Illustration of $\mathcal{N}_{\mathbf{p}}$. For each pixel $\mathbf{p}$ in the green box, we consider the 9 grid cells in the red box for assignment.

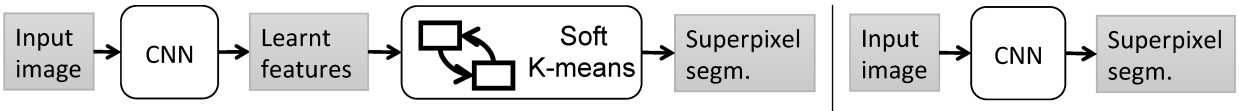
In practice, however, it is unnecessary and computationally expensive to compute $g_{i,j}(u, v)$ for all pixel-superpixel pairs. Instead, for a given pixel $\mathbf{p}$, we constraint the search to the set of surrounding grid cells $\mathcal{N}_{\mathbf{p}}$. This is illustrated in Figure 2. For each pixel $\mathbf{p}$ in the green box, we only consider the 9 grid cells in the red box for assignment. Consequently, we can write the mapping as a tensor $G \in \mathbb{Z}^{H \times W \times |\mathcal{N}_{\mathbf{p}}|}$ where $|\mathcal{N}_{\mathbf{p}}| = 9$.

While several approaches [22, 36, 10, 1, 23, 25, 2, 16] have been proposed to compute G, we take a different route in the paper. Specifically, we directly learn the mapping using a deep neural network. To make our objective function differentiable, we replace the hard assignment G with a soft association map $Q \in \mathbb{R}^{H \times W \times |\mathcal{N}_{\mathbf{p}}|}$. Here, the entry $q_{\mathbf{s}}(\mathbf{p})$ represents the probability that a pixel $\mathbf{p}$ is assigned to each $\mathbf{s} \in \mathcal{N}_{\mathbf{p}}$, such that $\sum_{\mathbf{s} \in \mathcal{N}_{\mathbf{p}}} q_{\mathbf{s}}(\mathbf{p}) = 1$. Finally, the superpixels are obtained by assigning each pixel to the grid cell with the highest probability: $\mathbf{s}^* = \arg\max_{\mathbf{s}} q_{\mathbf{s}}(\mathbf{p})$.

Although it might seem a strong constraint that a pixel can only be associated to one of the 9 nearby cells, which leads to the difficulty to generate long/large superpixels, we want to emphasize the importance of the compactness. Superpixel is inherently an over-segmentation method. As one of the main purposes of our superpixel method is to perform the detail-preserved downsampling/upsampling to assist the downstream network, it is more important to capture spatial coherence in the local region. For the information goes beyond the 9-cell area, there is no problem to segment it into pieces and leave them for downstream network to aggregate with convolution operations.

Our method vs. SSN [16]. Recently, [16] proposes SSN, an end-to-end trainable deep network for superpixel segmentation. Similar to our method, SSN also computes a soft association map Q. However, unlike our method, SSN uses the CNN as a means to extract pixel features, which are then fed to a soft K-means clustering module to compute Q.

We illustrate the algorithmic schemes of the two methods



(a) SSN (b) Ours

Figure 3. Comparison of algorithmic schemes. SSN trains a CNN to extract pixel features, which are fed to an iterative K-means clustering module for superpixel segmentation. We train a CNN to directly generate superpixels by predicting a pixel-superpixel association map.

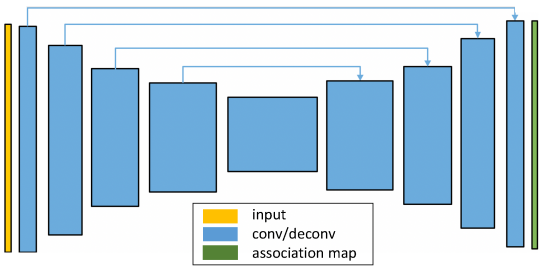


Figure 4. Our simple encoder-decoder architecture for superpixel segmentation. Please refer to the supplementary materials for detailed specifications.

in Figure 3. Both SSN and our method can take advantage of CNN to learn complex features using task-specific loss functions. But unlike SSN, we combine feature extraction and superpixel segmentation into a single step. As a result, our network runs faster and can be easily integrated into existing CNN frameworks for downstream tasks (Section 4).

3.2. Network Design and Loss Functions

As shown in Figure 4, we use a standard encoder-decoder design with skip connections to predict superpixel association map Q. The encoder takes a color image as input and produces high-level feature maps via a convolutional network. The decoder then gradually upsamples the feature maps via deconvolutional layers to make final prediction, taking into account also the features from corresponding encoder layers. We use leaky ReLU for all layers except for the prediction layer, where softmax is applied.

Similar to SSN [16], one of the main advantages of our end-to-end trainable superpixel network is its flexibility w.r.t. the loss functions. Recall that the idea of superpixels is to group similar pixels together. For different applications, one may wish to define similarity in different ways.

Generally, let $\mathbf{f}(\mathbf{p})$ be the pixel property we want the superpixels to preserve. Examples of $\mathbf{f}(\mathbf{p})$ include a 3-dimensional CIELAB color vector, and/or a N-dimensional one-hot encoding vector of semantic labels, where N is the number of classes, and many others. We further represent a pixel's position by its image coordinates $\mathbf{p} = [x, y]^T$.

Given the predicted association map Q, we can compute the center of any superpixel $\mathbf{s}$, $\mathbf{c_s} = (\mathbf{u_s}, \mathbf{l_s})$ where $\mathbf{u_s}$ is the property vector and $\mathbf{l_s}$ is the location vector, as follows:

$$\mathbf{u_s} = \frac{\sum_{\mathbf{p}:\mathbf{s}\in\mathcal{N}_\mathbf{p}} \mathbf{f}(\mathbf{p}) \cdot q_\mathbf{s}(\mathbf{p})}{\sum_{\mathbf{p}:\mathbf{s}\in\mathcal{N}_\mathbf{p}} q_\mathbf{s}(\mathbf{p})}, \quad \mathbf{l_s} = \frac{\sum_{\mathbf{p}:\mathbf{s}\in\mathcal{N}_\mathbf{p}} \mathbf{p} \cdot q_\mathbf{s}(\mathbf{p})}{\sum_{\mathbf{p}:\mathbf{s}\in\mathcal{N}_\mathbf{p}} q_\mathbf{s}(\mathbf{p})}. \tag{1}$$

Here, recall that $\mathcal{N}_\mathbf{p}$ is the set of surrounding superpixels of $\mathbf{p}$, and $q_\mathbf{s}(\mathbf{p})$ is the network predicted probability of $\mathbf{p}$ being associated with superpixel $\mathbf{s}$. In Eq (1), each sum is taken over all the pixels with a possibility to be assigned to $\mathbf{s}$.

Then, the reconstructed property and location of any pixel $\mathbf{p}$ are given by:

$$\mathbf{f}'(\mathbf{p}) = \sum_{\mathbf{s}\in\mathcal{N}_\mathbf{p}} \mathbf{u_s} \cdot q_\mathbf{s}(\mathbf{p}), \quad \mathbf{p}' = \sum_{\mathbf{s}\in\mathcal{N}_\mathbf{p}} \mathbf{l_s} \cdot q_\mathbf{s}(\mathbf{p}). \tag{2}$$

Finally, the general formulation of our loss function has two terms. The first term encourages the trained model to group pixels with similar property of interest, and the second term enforces the superpixels to be spatially compact:

$$L(Q) = \sum_\mathbf{p} \text{dist}(\mathbf{f}(\mathbf{p}), \mathbf{f}'(\mathbf{p})) + \frac{m}{S}\|\mathbf{p} - \mathbf{p}'\|_2, \tag{3}$$

where $\text{dist}(\cdot,\cdot)$ is the task specific distance metric depending on the pixel property $\mathbf{f}(\mathbf{p})$, S is the superpixel sampling interval, and m is a weight balancing the two terms.

In this paper, we consider two different choices of $\mathbf{f}(\mathbf{p})$. First, we choose the CIELAB color vector and use the ℓ_2 norm as the distance measure. This leads to an objective function similar to the original SLIC method [1]:

$$L_{SLIC}(Q) = \sum_\mathbf{p} \|\mathbf{f}_{col}(\mathbf{p}) - \mathbf{f}'_{col}(\mathbf{p})\|_2 + \frac{m}{S}\|\mathbf{p} - \mathbf{p}'\|_2. \tag{4}$$

Second, following [16], we choose the one-hot encoding vector of semantic labels and use cross-entropy $E(\cdot,\cdot)$ as the distance measure:

$$L_{sem}(Q) = \sum_\mathbf{p} E(\mathbf{f}_{sem}(\mathbf{p}), \mathbf{f}'_{sem}(\mathbf{p})) + \frac{m}{S}\|\mathbf{p} - \mathbf{p}'\|_2. \tag{5}$$

3.3. Connection to Spatial Propagation Network

Recently, [8] proposes the convolutional spatial propagation (CSP) network, which learns an affinity matrix to propagate information to nearly spatial locations. By integrating the CSP module into existing deep neural networks, [8] has demonstrated improved performance in affinity-based vision tasks such as depth completion and refinement. In this section, we show that the computation of superpixel centers using learnt association map Q can be written mathematically in the form of CSP, thus draw a connection between learning Q and learning the affinity matrix as in [8].

Given an input feature volume $X \in \mathbb{R}^{H\times W\times C}$, the convolutional spatial propagation (CSP) with a kernel size K and stride S can be written as:

$$\mathbf{y}_{i,j} = \sum_{a,b=-K/2+1}^{K/2} \kappa_{i,j}(a,b) \odot \mathbf{x}_{i\cdot S+a, j\cdot S+b}, \quad (6)$$

where $Y \in \mathbb{R}^{h\times w\times C}$ is an output volume such that $h = \frac{H}{S}$ and $w = \frac{W}{S}$, $\kappa_{i,j}$ is the output from an affinity network such that $\sum_{a,b=-K/2+1}^{K/2} \kappa_{i,j}(a,b) = 1$, and $\odot$ is element-wise product.

In the meantime, as illustrated in Figure 2, to compute the superpixel center associated with the (i,j)-th grid cell, we consider all pixels in the surrounding $3S \times 3S$ region:

$$\mathbf{c}_{i,j} = \sum_{a,b=-3S/2+1}^{3S/2} \hat{q}_{i,j}(a,b) \odot \mathbf{x}_{i\cdot S+a, j\cdot S+b}, \quad (7)$$

where

$$\hat{q}_{i,j}(a,b) = \frac{q_{i,j}(u,v)}{\sum_{a,b=-3S/2+1}^{3S/2} q_{i,j}(u,v)}, \quad (8)$$

and $u = i\cdot S + a, v = j\cdot S + b$.

Comparing Eq. (6) with Eq. (7), we can see that computing the center of superpixel of size $S \times S$ is equivalent to performing CSP with a $3S\times 3S$ kernel derived from Q. Furthermore, both $\kappa_{i,j}(a,b)$ and $q_{i,j}(u,v)$ represent the learnt weight between the spatial location (u,v) in the input volume and (i,j) in the output volume. In this regard, predicting Q in our work can be viewed as learning an affinity matrix as in [8].

Nevertheless, we point out that, while the techniques presented in this work and [8] share the same mathematical form, they are developed for very different purposes. In [8], Eq. (6) is employed repeatedly (with $S = 1$) to propagate information to nearby locations, whereas in this work, we use Eq. (7) to compute superpixel centers (with $S > 1$).

3.4. Experiments

We train our model with segmentation labels on the standard benchmark BSDS500 [3] and compare it with state-of-the-art superpixel methods. To further evaluate the method generalizability, we also report its performances without fine-tuning on another benchmark dataset NYUv2 [28].

All evaluations are conducted using the protocols and codes provided by [33][1]. We run LSC [23], ERS [24], SNIC [2], SEAL [35], and SSN [16] with the original implementations from the authors, and run SLIC [1] and ETPS [43] with the codes provided in [33]. For LSC, ERS, SLIC and ETPS, we use the best parameters reported in [33], and for the rest, we use the default parameters recommended by the original authors.

[1] https://github.com/davidstutz/superpixel-benchmark

Implementation details. Our model is implemented with PyTorch, and optimized using Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use L_{sem} in Eq. (5) for this experiment, with $m = 0.003$. During training, we randomly crop the images to size 208×208 as input, and perform horizontal/vertical flipping for data augmentation. The initial learning rate is set to 5×10^{-5}, and is reduced by half after 200k iterations. Convergence is reached at about 300k iterations.

For training, we use a grid with cell size 16×16, which is equivalent to setting the desired number of superpixels to 169. At the test time, to generate varying number of superpixels, we simply resize the input image to the appropriate size. For example, by resizing the image to 480×320, our network will generate about 600 superpixels. Furthermore, for fair comparison, most evaluation protocols expect superpixels to be spatially connected. To enforce that, we apply an off-the-shelf component connection algorithm to our output, which merges superpixels that are smaller than a certain threshold with the surrounding ones.[2]

Evaluation metrics. We evaluate the superpixel methods using the popular metrics including achievable segmentation accuracy (ASA), boundary recall and precision (BR-BP), and compactness (CO). ASA quantifies the achievable accuracy for segmentation using the superpixels as pre-processing step, BR and BP measure the boundary adherence of superpixels given the ground truth, whereas CO assesses the compactness of superpixels. The higher these scores are, the better the segmentation result is. As in [33], for BR and BP evaluation, we set the boundary tolerance as 0.0025 times the image diagonal rounded to the closest integer. We refer readers to [33] for the precise definitions.

Results on BSDS500. BSDS500 contains 200 training, 100 validation, and 200 test images. As multiple labels are available for each image, we follow [16, 35] and treat each annotation as an individual sample, which results in 1633 training/validation samples and 1063 testing samples. We train our model using both the training and validation samples.

Figure 5 reports the performance of all methods on BSDS500 test set. Our method outperforms all traditional methods on all evaluation metrics, except SLIC in term of CO. Comparing to the other deep learning-based methods, SEAL and SSN, our method achieves competitive or better results in terms of ASA and BR-BP, and significantly higher scores in term of CO. Figure 8 further shows example results of different methods. Note that, as discussed in [33], there is a well-known trade-off between boundary adherence and compactness. Although our method does not outperform existing methods on all the metrics, it appears to strike a better balance among them. It is also worth noting that by achieving higher CO score, our method is able to better capture spatially coherent information and avoids

[2] Code and models are available at https://github.com/fuy34/superpixel_fcn.

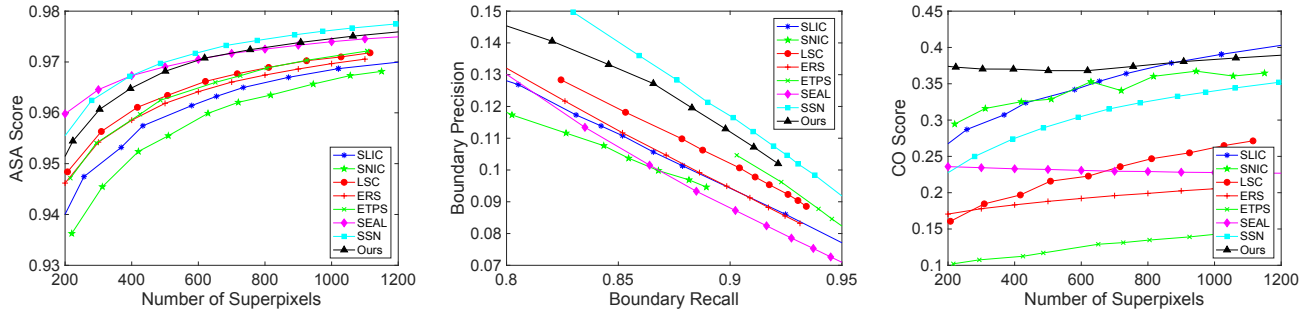


Figure 5. Superpixel segmentation results on BSDS500. **From left to right**: ASA, BR-BP, and CO.

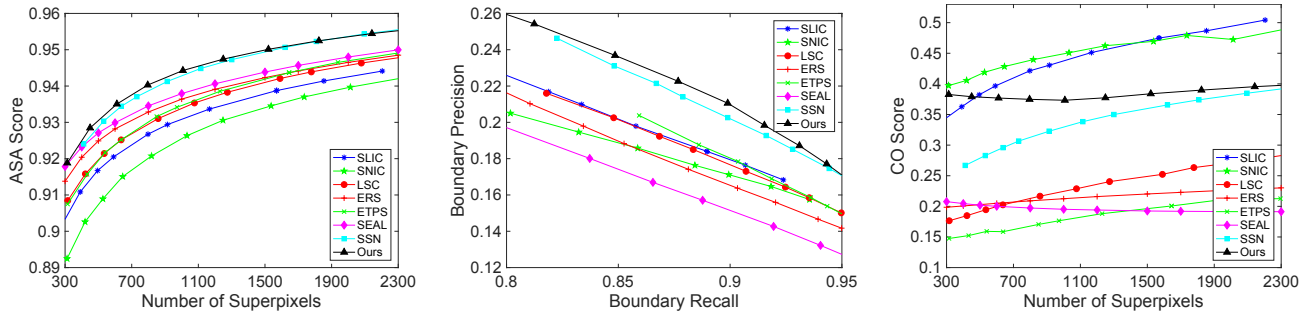


Figure 6. Superpixel segmentation results on NYUv2. **From left to right**: ASA, BR-BP, and CO.

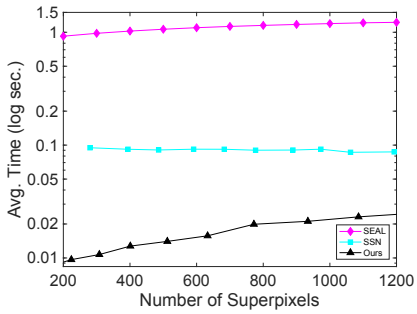


Figure 7. Average runtime of different DL methods w.r.t. number of superpixels. Note that y-axis is plotted in the logarithmic scale.

paying too much attention to image details and noises. This characteristic tends to lead to better generalizability, as shown in the NYUv2 experiment results.

We also compare the runtime difference among deep learning-based (DL) methods. Figure 7 reports the average runtime w.r.t. the number of generated superpixels on a NVIDIA GTX 1080Ti GPU device. Our method runs about 3 to 8 times faster than SSN and more than 50 times faster than SEAL. This is expected as our method uses a simple encoder-decoder network to directly generate superpixels, whereas SEAL and SSN first use deep networks to predict pixel affinity or features, and then apply traditional clustering methods (i.e., graph cuts or K-means) to get superpixels.

Results on NYUv2. NYUv2 is a RGB-D dataset originally proposed for indoor scene understanding tasks, which contains 1,449 images with object instance labels. By removing the unlabelled regions near the image boundary, [33] has developed a benchmark on a subset of 400 test images with size 608×448 for superpixel evaluation. To test the generalizability of the learning-based methods, we directly apply the models of SEAL, SSN, and our method trained on BSDS500 to this dataset without any fine-tuning.

Figure 6 shows the performance of all methods on NYUv2. Generally, all deep learning-based methods perform well as they continue to achieve competitive or better performance against the traditional methods. Further, our method is shown to generalize better than SEAL and SSN, which is evident by comparing the corresponding curves in Figure 5 and 6. Specifically, our method outperforms SEAL and SSN in terms of BR-BP and CO, and is one of the best in terms of ASA. The visual results are shown in Figure 8.

4. Application to Stereo Matching

Stereo matching is a classic computer vision task which aims to find pixel correspondences between a pair of rectified images. Recent literature has shown that deep networks can boost the matching accuracy by building 4D cost volume (height×width×disparity×feature channels) and aggregate the information using 3D convolution [7, 8, 46]. However, such a design consumes large amounts of memory because of the extra "disparity" dimension, limiting their ability to generate high-res outputs. A common remedy is to bilinearly upsample the predicted low-res disparity volumes for final disparity regression. As a result, object boundaries often become blur and fine details get lost.

In this section, we propose a downsampling/upsampling scheme based on the predicted superpixels and show how to integrate it into existing stereo matching pipelines to gener-

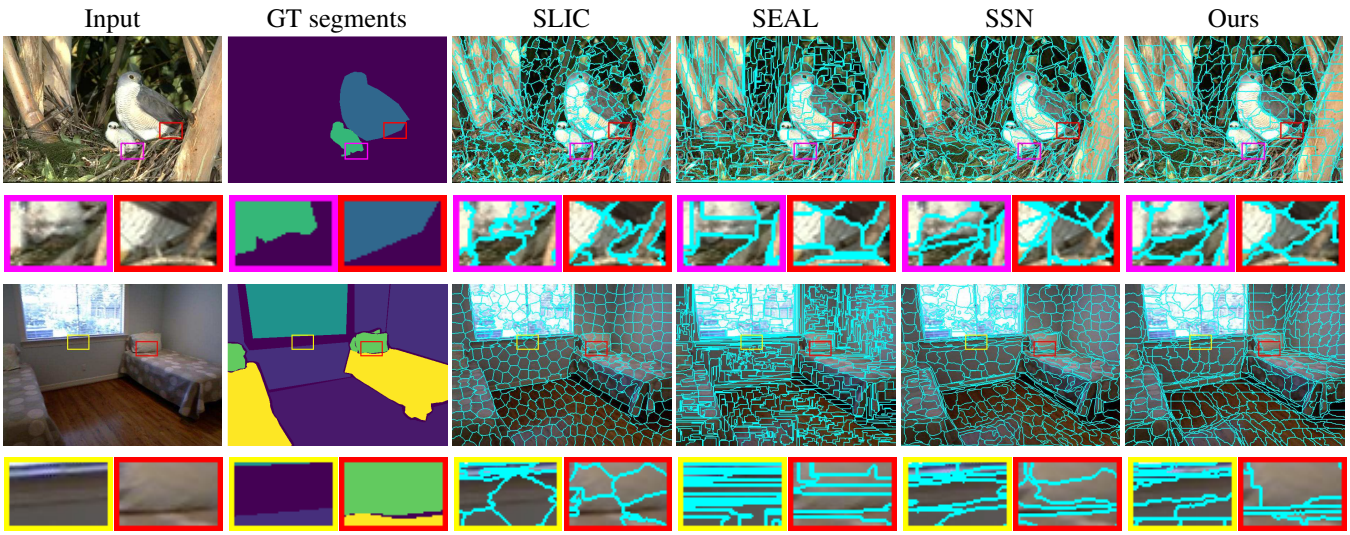


Figure 8. Example superpixel segmentation results. Compared to SEAL and SSN, our method is competitive or better in terms of object boundary adherence while generating more compact superpixels. **Top rows**: BSDS500. **Bottom rows**: NYUv2.

ate high-res outputs that better preserve the object boundaries and fine details.

4.1. Network Design and Loss Function

Figure 1 provides an overview of our method design. We choose PSMNet [7] as our task network. In order to incorporate our new downsampling/upsampling scheme, we change all the stride-2 convolutions in its feature extractor to stride-1, and remove the bilinear upsampling operations in the spatial dimensions. Given a pair of input images, we use our superpixel network to predict association maps Q_l, Q_r and compute the superpixel center maps using Eq. (1). The center maps (*i.e.*, downsampled images) are then fed into the modified PSMNet to get the low-res disparity volume. Next, the low-res volume is upsampled to original resolution with Q_l according to Eq. (2), and the final disparity is computed using disparity regression. We refer readers to the supplementary materials for detailed specification.

Same as PSMNet [7], we use the 3-stage smooth L_1 loss with the weights $\alpha_1 = 0.5$, $\alpha_2 = 0.7$, and $\alpha_3 = 1.0$ for disparity prediction. And we use the SLIC loss (Eq. (4)) for superpixel segmentation. The final loss function is:

$$L = \sum_{s=1}^{3} \alpha_s \Big(\frac{1}{N} \sum_{p=1}^{N} smooth_{L_1}(d_p - \hat{d}_p) \Big) + \frac{\lambda}{N} L_{SLIC}(Q) \tag{9}$$

where N is the total number of pixels, and λ is a weight to balance the two terms. We set $\lambda = 0.1$ for all experiments.

4.2. Experiments

We have conducted experiments on three public datasets, SceneFlow [27], HR-VS [42], and Middlebury-v3 [30] to compared our model with PSMNet. To further verify the benefit of joint learning for superpixels and disparity estimation, we trained two different models for our method. In the first model **Ours_fixed**, we fix the parameters in superpixel network and train the rest of the network (*i.e.*, the modified PSMNet) for disparity estimation. In the second model **Ours_joint**, we jointly train all networks in Figure 1. For both models, the superpixel network is pre-trained on SceneFlow using the SLIC loss. The experiments are conducted on 4 Nvidia TITAN Xp GPUs.

Results on SceneFlow. SceneFlow is a synthetic dataset contains 35,454 training and 4,370 test frames with dense ground truth disparity. Following [7], we exclude pixels with disparities greater than 192 in training and test time.

During training, we set $m = 30$ in the SLIC loss and randomly crop the input images into size 512×256. To conduct 3D convolution at $1/4$ of the input resolution as PSMNet does, we predict superpixels with grid cell size 4×4 to perform the $4\times$ downsampling/upsampling. We train the model for 13 epochs with batch size 8. The initial learning rate is 1×10^{-3}, and is reduced to 5×10^{-4} and 1×10^{-4} after 11 and 12 epochs, respectively. For PSMNet, we use the authors' implementation and train it with the same learning schedule as our methods.

We use the standard end-point-error (EPE) as the evaluation metric, which measures the mean pixel-wise Euclidean distance between the predicted disparity and the ground truth. As shown in Table 1, **Ours_joint** achieves the lowest EPE. Also note that **Ours_fixed** performs worse than the original PSMNet, which demonstrates the importance of joint training. Qualitative results are shown in Figure 9. One can see that both **Ours_fixed** and **Ours_joint** preserve fine details better than the original PSMNet.

Results on HR-VS. HR-VS is a synthetic dataset with ur-

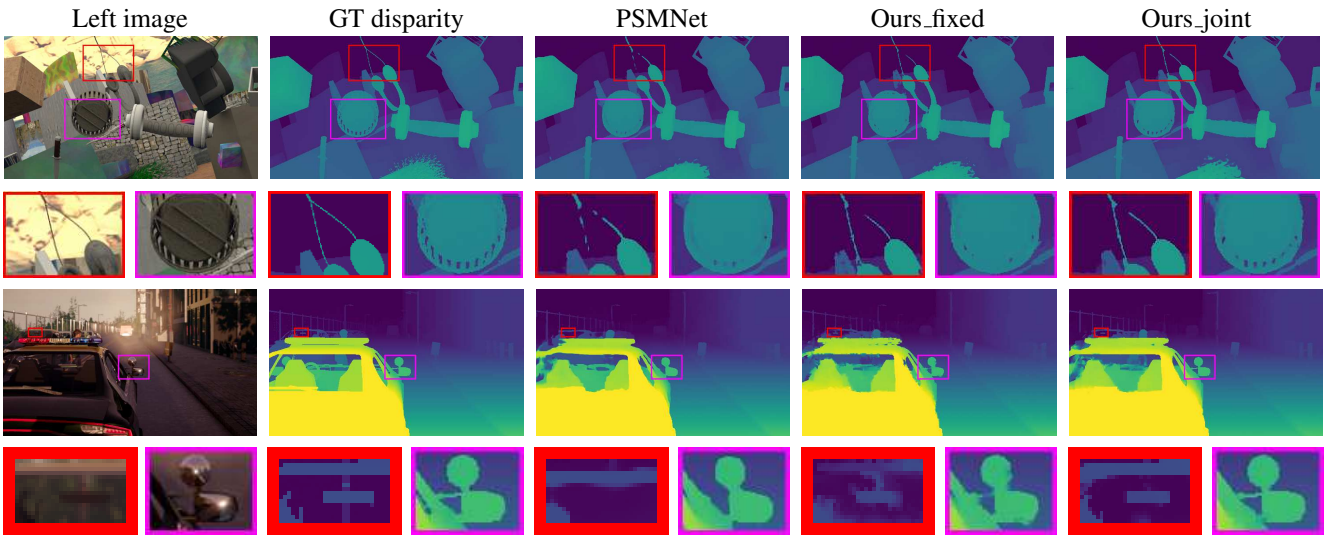


Figure 9. Qualitative results on SceneFlow and HR-VS. Our method is able to better preserve fine details, such as the wires and mirror frameworks in the highlighted regions. **Top rows**: SceneFlow. **Bottom rows**: HR-VS.

Table 1. End-point-error (EPE) on SceneFlow and HR-VS.

Dataset	PSMNet [7]	Ours_fixed	Ours_joint
SceneFlow	1.04	1.07	0.93
HR-VS	3.83	3.70	2.77

ban driving views. It contains 780 images at 2056×2464 resolution. The valid disparity range is [9.66, 768]. Because no test set is released, we randomly choose 680 frames for training, and use the rest for testing. Due to the relatively small data size, we fine-tune all three models trained on SceneFlow in the previous experiment on this dataset.

Because of the high resolution and large disparity, the original PSMNet cannot be directly applied to the full size images. We follow the common practice to downsample both the input images and disparity maps to 1/4 size for training, and upsample the result to full resolution for evaluation. For our method, we predict superpixels with grid cell size 16×16 to perform $16\times$ downsampling/upsampling. During training, we set $m = 30$, and randomly crop the images into size 2048×1024. We train all methods for 200 epochs with batch size 4. The initial learning rate is 1×10^{-3} and reduced to 1×10^{-4} after 150 epochs.

As shown in Table 1, our models outperform the original PSMNet. And a significantly lower EPE is achieved by joint training. Note that, comparing to SceneFlow, we observe a larger performance gain on this high-res dataset, as we perform $16\times$ upsampling on HR-VS but only $4\times$ upsampling on SceneFlow. Qualitative results are shown in Figure 9.

Results on Middlebury-v3. Middlebury-v3 is a high-res real-world dataset with 10 training frames, 13 validation frames[3], and 15 test frames. We use both training and validation frames to tune the **Our_joint** model pre-trained on SceneFlow with 16×16 superpixels. We set $m = 60$ and train the model for 30 epochs with batch size 4. The initial learning rate is 1×10^{-3} and divided by 10 after 20 epochs.

Note that, for the experiment, our goal is not to achieve the highest rank on the official Middlebury-v3 leaderboard. But instead, to verify the effectiveness of the proposed superpixel-based downsample/upsampling scheme. Based on the leaderboard, our model outperforms PSMNet across all metrics, some of which are presented in Table 2. The results again verify the benefit of the proposed superpixel-based downsample/upsampling scheme.

Table 2. Results on Middlebury-v3 benchmark.

Method	avgerr	rms	bad-4.0	A90
PSMNet_ROB [7]	8.78	23.3	29.2	22.8
Ours_joint	7.11	19.1	27.5	13.8

5. Conclusion

This paper has presented a simple fully convolutional network for superpixel segmentation. Experiments on benchmark datasets show that the proposed model is computationally efficient, and can consistently achieve the state-of-the-art performance with good generalizability. Further, we have demonstrated that higher disparity estimation accuracy can be obtained by using superpixels to preserve object boundaries and fine details in a popular stereo matching network. In the future, we plan to apply the proposed superpixel-based downsampling/upsampling scheme to other dense prediction tasks, such as object segmentation and optical flow estimation, and explore different ways to use superpixels in these applications.

Acknowledgement. This work is supported in part by NSF award #1815491 and a gift from Adobe.

[3]Named as additional dataset in the official website.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2274–2282, 2012. 1, 2, 3, 4, 5

[2] Radhakrishna Achanta and Sabine Süsstrunk. Superpixels and polygons using simple non-iterative clustering. In *CVPR*, pages 4895–4904, 2017. 1, 2, 3, 5

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2010. 2, 5

[4] Stan Birchfield and Carlo Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *ICCV*, pages 489–495, 1999. 3

[5] Michael Bleyer and Margrit Gelautz. A layered stereo algorithm using image segmentation and global visibility constraints. In *ICIP*, pages 2997–3000, 2004. 3

[6] Michael Bleyer, Carsten Rother, and Pushmeet Kohli. Surface stereo with soft segmentation. In *CVPR*, pages 1570–1577, 2010. 3

[7] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, pages 5410–5418, 2018. 2, 3, 6, 7, 8

[8] Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation network. *CoRR*, abs/1810.02695, 2018. 1, 2, 3, 4, 5, 6

[9] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, pages 764–773, 2017. 2

[10] Michael Van den Bergh, Xavier Boix, Gemma Roig, and Luc J. Van Gool. SEEDS: superpixels extracted via energy-driven sampling. *International Journal of Computer Vision*, 111(3):298–314, 2015. 1, 2, 3

[11] Raghudeep Gadde, Varun Jampani, Martin Kiefel, Daniel Kappler, and Peter V. Gehler. Superpixel convolutional networks using bilateral inceptions. In *ECCV*, pages 597–613, 2016. 1, 2

[12] Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan, and Daphne Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 80(3):300–316, 2008. 1

[13] Fatma Güney and Andreas Geiger. Displets: Resolving stereo ambiguities using object knowledge. In *CVPR*, pages 4165–4175, 2015. 3

[14] Shengfeng He, Rynson W. H. Lau, Wenxi Liu, Zhe Huang, and Qingxiong Yang. Supercnn: A superpixelwise convolutional neural network for salient object detection. *International Journal of Computer Vision*, 115(3):330–344, 2015. 1, 2

[15] Li Hong and George Chen. Segment-based stereo matching using graph cuts. In *CVPR*, pages 74–81, 2004. 3

[16] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel sampling networks. In *ECCV*, pages 363–380, 2018. 1, 2, 3, 4, 5

[17] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, and Peter Henry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, pages 66–75, 2017. 1, 3

[18] Sameh Khamis, Sean Ryan Fanello, Christoph Rhemann, Adarsh Kowdle, Julien P. C. Valentin, and Shahram Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *ECCV*, pages 596–613, 2018. 3

[19] Andreas Klaus, Mario Sormann, and Konrad F. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *ICPR*, pages 15–18, 2006. 3

[20] Suha Kwak, Seunghoon Hong, and Bohyung Han. Weakly supervised semantic segmentation using superpixel pooling network. In *AAAI*, pages 4111–4117, 2017. 1, 2

[21] Chloe LeGendre, Konstantinos Batsos, and Philippos Mordohai. High-resolution stereo matching based on sampled photoconsistency computation. In *BMVC*, 2017. 3

[22] Alex Levinshtein, Adrian Stere, Kiriakos N. Kutulakos, David J. Fleet, Sven J. Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12):2290–2297, 2009. 1, 2, 3

[23] Zhengqin Li and Jiansheng Chen. Superpixel segmentation using linear spectral clustering. In *CVPR*, pages 1356–1363, 2015. 1, 2, 3, 5

[24] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *CVPR*, pages 2097–2104. IEEE, 2011. 5

[25] Yong-Jin Liu, Cheng-Chi Yu, Minjing Yu, and Ying He. Manifold SLIC: A fast method to compute content-sensitive superpixels. In *CVPR*, pages 651–659, 2016. 1, 2, 3

[26] Wenjie Luo, Alexander G. Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *CVPR*, pages 5695–5703, 2016. 3

[27] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, pages 4040–4048, 2016. 2, 7

[28] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2, 5

[29] Jiahao Pang, Wenxiu Sun, Jimmy S. J. Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *ICCV Workshops*, pages 878–886, 2017. 3

[30] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition*, pages 31–42. Springer, 2014. 2, 7

[31] Amit Shaked and Lior Wolf. Improved stereo matching with constant highway networks and reflective confidence learning. In *CVPR*, pages 6901–6910, 2017. 3

[32] Guang Shu, Afshin Dehghan, and Mubarak Shah. Improving an object detector and extracting regions using superpixels. In *CVPR*, pages 3721–3727, 2013. 1

[33] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27, 2018. 2, 5, 6

[34] Teppei Suzuki, Shuichi Akizuki, Naoki Kato, and Yoshimitsu Aoki. Superpixel convolution for segmentation. In *ICIP*, pages 3249–3253, 2018. 1, 2

[35] Wei-Chih Tu, Ming-Yu Liu, Varun Jampani, Deqing Sun, Shao-Yi Chien, Ming-Hsuan Yang, and Jan Kautz. Learning superpixels with segmentation-aware affinity loss. In *CVPR*, pages 568–576, 2018. 2, 5

[36] Peng Wang, Gang Zeng, Rui Gan, Jingdong Wang, and Hongbin Zha. Structure-sensitive superpixels via geodesic distance. *International Journal of Computer Vision*, 103(1):1–21, 2013. 1, 2, 3

[37] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. Superpixel tracking. In *ICCV*, pages 1323–1330, 2011. 1

[38] Zeng-Fu Wang and Zhi-Gang Zheng. A region based stereo matching algorithm using cooperative optimization. In *CVPR*, 2008. 3

[39] Koichiro Yamaguchi, Tamir Hazan, David A. McAllester, and Raquel Urtasun. Continuous markov random fields for robust stereo estimation. In *ECCV*, pages 45–58, 2012. 3

[40] Koichiro Yamaguchi, David A. McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *ECCV*, pages 756–771, 2014. 3

[41] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *CVPR*, pages 3166–3173, 2013. 1

[42] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *CVPR*, pages 5515–5524, 2019. 1, 2, 3, 7

[43] Jian Yao, Marko Boben, Sanja Fidler, and Raquel Urtasun. Real-time coarse-to-fine topologically preserving segmentation. In *CVPR*, pages 2947–2955, 2015. 5

[44] Lidong Yu, Yucheng Wang, Yuwei Wu, and Yunde Jia. Deep stereo matching with explicit cost aggregation sub-architecture. In *AAAI*, pages 7517–7524, 2018. 3

[45] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17:65:1–65:32, 2016. 3

[46] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *CVPR*, pages 185–194, 2019. 6

[47] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets V2: more deformable, better results. In *CVPR*, pages 9308–9316. Computer Vision Foundation / IEEE, 2019. 2