Learning Propagation for Arbitrarily-structured Data

Sifei Liu[1], Xueting Li[1,2], Varun Jampani[1]*, Shalini De Mello[1], Jan Kautz[1]
[1]NVIDIA, [2]University of California, Merced

Abstract

Processing an input signal that contains arbitrary structures, e.g., superpixels and point clouds, remains a big challenge in computer vision. Linear diffusion, an effective model for image processing, has been recently integrated with deep learning algorithms. In this paper, we propose to learn pairwise relations among data points in a global fashion to improve semantic segmentation with arbitrarily-structured data, through spatial generalized propagation networks (SGPN). The network propagates information on a group of graphs, which represent the arbitrarily-structured data, through a learned, linear diffusion process. The module is flexible to be embedded and jointly trained with many types of networks, e.g., CNNs. We experiment with semantic segmentation networks, where we use our propagation module to jointly train on different data – images, superpixels and point clouds. We show that SGPN consistently improves the performance of both pixel and point cloud segmentation, compared to networks that do not contain this module. Our method suggests an effective way to model the global pairwise relations for arbitrarily-structured data.

1. Introduction

The individual visual elements of spatially distributed data, *e.g.*, pixels/superpixels in an image or points in a point cloud, exhibit strong pairwise relations. Capturing these relations is important for understanding and processing such data. For example, in semantic segmentation, where each pixel/point is assigned a semantic label, it is very likely that the points that are spatially and photometrically close, or structurally connected to each other have the same semantic label, compared to those that are farther away. We can make use of such similarity cues to infer the relationships among points and improve the propagation of information (*e.g.*, semantic labels, color *etc.*) between them. This pairwise relationship modeling is often called "affinity" modeling. Evidence from psychological [5, 42] and empirical studies in computer vision [37, 17, 27] suggests that general classification or regression problems can immensely benefit from the explicit modeling of pairwise affinities.

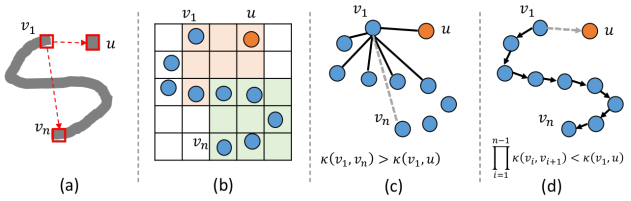


Figure 1. Groupings of different objects v and u in (a) with different strategies: (b) performing convolution on grids; explicit pairwise modeling via (c) fully-connected graphs, and (d) our path-aware propagation. Since v and u have the same color, we model the similarity (κ) using spatial closeness between two points.

With a dramatic rise in adoption for computer vision tasks, CNNs implicitly model pairwise relationships, as convolution filters learn to capture correlations across image pixels. Several extensions of CNNs to process arbitrarily-structured data (such as point clouds) have been proposed (*e.g.*, permutohedral lattice [1, 24, 43]) that go beyond processing regular grid-like structured images. They transform the data to some regular structures, such that convolutional filters can be learned for them. However, convolutions can only capture short-range pairwise relations and the filters are also content-agnostic as their weights are fixed once they are trained. As a result, we usually resort to using very deep network architectures to model all possible pairwise relations, and long-range pixel dependencies. As an alternative, several recent works [54, 10, 8, 24, 31, 7, 19, 34, 47, 36] propose neural network modules that can explicitly model pairwise relations, resulting in considerable improvements in CNN performance for a variety of computer vision tasks. However, most of them are designed on regularly-structured data, such as images and videos.

Despite the existence of these methods, several important challenges remain for processing arbitrarily-structured data such as point clouds: First, we hope such data can be represented with a more flexible structure, instead of regular-grids (such as voxel grids or permutohedral lattice), such that the original structure of the input data can be faithfully preserved. Second, as mentioned above, we hope to explicitly model the pairwise relations among their data el-

*The current affiliation is Google Research.

ements. Third, we hope to model the pairwise relations globally, but still adhere to the structures of the input data. Fig. 1 illustrates the above challenges, where the aim is to decide for the point v_1, which belongs to the curved object, whether v_n and u belong to the same object as v_1. As shown in Fig. 1(b), placing a curve on a grid and conducting convolution on top of it does not effectively correlate the elements. On the other hand, with explicit pairwise modeling as shown in Fig. 1(c), if we relate v_1 with the other points globally by independently computing their Euclidean distances, we will incorrectly model v_1 and v_n as "not similar", but v_1 and u as "similar", since they are spatially closer. Fig. 1(c) also belongs to the non-local propagation methods [27, 47, 54, 7, 24], which explicitly model pairwise relations via a fully-connected graph.

In this work, we aim to address all the above mentioned challenges by proposing a spatial generalized propagation network (SGPN), as illustrated in Fig. 1(d). Instead of transforming input points into a regular grid structure, we retain the original spatial structure of the data, but establish several directed acyclic graphs (DAGs) to connect adjacent points, where Fig. 1(d) shows a top-to-bottom DAG that faithfully adheres to the curved object v's structure. With our propagation operator, the distance between v_1 and v_n is determined by the accumulated connections of the adjacent elements between them. When the multiplication of the intermediate distances is small, we can correctly model v_1 and v_n as belonging to the same object, even though they are spatially far away.

We show that, theoretically, our propagation mechanism is equivalent to linear diffusion. More importantly, we propose a differentiable kernel operator such that even for DAGs, the strength of an edge between two connected nodes is learnable. Moreover, our entire framework is a flexible deep learning building block, where the SGPN can be embedded in, and jointly optimized with any type of network, *e.g.*, any baseline CNN for semantic segmentation. For the same reason our propagation module, which operates on arbitrarily-structured data, *e.g.*, point clouds, can also be easily combined with 2D CNNs that process images associated with the points, *e.g.*, the multi-view images corresponding to point clouds. We demonstrate the effectiveness of SGPN by applying it to different types of data, including image pixels, superpixels and point clouds, for the task of semantic segmentation. Experimental results show that our SGPN outperforms state-of-the-art methods on semantic segmentation with all types of data and consistently improves all the baseline models by reliable margins.

2. Related Work

Modeling irregularly-structured data. Irregular data domains refer to those that do not contain regularly ordered elements, *e.g.*, superpixels or point clouds. Deep learning methods that support processing irregular domains are far less than those that exist for regular domains, *e.g.*, images and videos. For modeling superpixels, the work of [22] uses superpixels inside CNNs by re-arranging them by their features. The work of [24] uses a superpixel convolution module inside a neural network, which results in some performance improvement [46, 25]. In comparison, quite a few networks have been designed for point clouds [29, 39, 40, 44, 43], where most target adapting CNN modules to unstructured data, instead of explicitly modeling the pairwise relationships between the points. On the other hand, while some propagation modules [27, 26, 51, 24] address affinity modeling for irregularly-structured data, they cannot address the challenge of preserving internal structures due to the non-local nature of their propagation.

Modeling pairwise affinity. Pairwise relations are modeled in a broad range of low- to high-level vision problems. Image filtering techniques including edge-preserving smoothing and image denoising [2, 45, 6, 21] are some of the most intuitive examples of applying pairwise modeling to real-world applications. The task of structured prediction [27, 28, 18], on the other hand, seeks to explicitly model relations in more general problems. Recently, many methods for modeling affinity have been proposed as deep learning building blocks [34, 48, 49, 47, 54, 31, 7, 24, 51], and several of them also propose to "learn" affinities [34, 47, 24, 51]. Besides these methods, diffusion theory [38] provides a fundamental framework that relates the task of explicit modeling of pairwise relations to physical processes in the real world, where many popular affinity building blocks [47, 51, 34] can be described by it.

Propagation networks. Our work is related to the recent spatial propagation networks (SPNs) [34, 12] for images, which learn affinities between pixels to refine pixel-level classification [34] or regression [52, 33, 12] tasks. SPNs model affinity via a differentiable propagation layer, where the propagation itself is guided by learnable, spatially-varying weights that are conditioned on the input image pixels. SPNs have the advantage of faithfully preserving complex image structures in image segmentation [34], depth estimation [52] and temporal propagation [33]. We show in the following section, that our work generalizes SPNs to arbitrary graphs, such that SPN can be viewed as a special case of our work on regular grids. Our work is also related to recurrent neural networks (RNN) on graphs [23, 15, 30]. However, unlike our work RNNs are not designed for *linear* diffusion on graphs, but instead target more general problems represented as graphs.

3. Spatial Generalized Propagation Network

Unlike images where pixels are placed on regular 2D grids, data such as superpixels or point clouds encountered

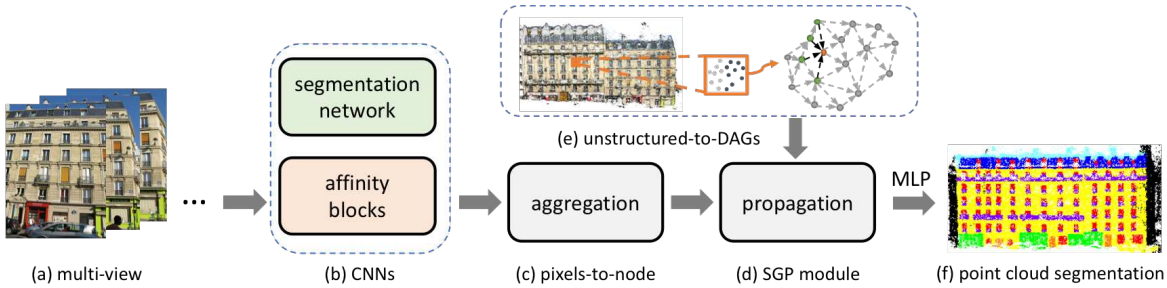


Figure 2. A general architecture of SGPN for point cloud segmentation. See Section 3 for details of individual modules.

in vision tasks have an undefined structure. In order to process such data with deep neural networks, they must be converted into some structures such as a high-dimensional lattice [43] or fully connected graphs [39, 47], on which the operations of convolution, pooling, etc, can be carried out. We take point cloud segmentation as an example in Fig. 2 to explain our method. We build a group of DAGs on the raw points, as shown in Fig. 2(e), by connecting the spatially adjacent points. In contrast with transforming the unstructured points to a rigid lattice, where the topologies of the raw points may be changed, and many unused vertices may exist (e.g., see Fig. 1(b) where many grid cells are unoccupied), DAGs neither change the input topologies, nor consume any extra memory in order to enforce a regular structure. Such a DAG structure is highly flexible since the input points of different objects can have different DAGs exactly adhering to their own shapes.

In terms of explicit pairwise modeling, in contrast to fully connected graphs [47] where points are densely connected (see Fig. 1(c)), the structure of DAGs also enables the propagation along different directions to be carried out, and "paths" along complex shapes of the input data (*e.g.* Fig. 1(d)) are modeled. We establish different directions with DAGs, *e.g.*, along the x, y and z axis for a point cloud in 3D (6 directions in total with positive and negative directions along each axis), where we show the left-to-right DAG in Fig. 2(e). The DAGs can be built on a global scope, *e.g.*, for a point cloud with millions of points, to support long-range propagation.

Once the DAG is constructed, we learn pairwise affinities between the DAG vertices and we use our SGPN propagation module for structured information propagation along the edges. SGPN can be attached on top of any CNN that provides initial (unary) features at DAG vertices. In the case of point clouds, the CNN can be an existing 3D network. To demonstrate the flexibility of SGPN and to leverage the potential of 2D CNNs, we obtain vertex features using a 2D CNN on the corresponding multi-view 2D images. We use a differentiable aggregation module 2(c) that transforms the pixel features into vertex features on the DAG. In the following part, we first describe the formulation of linear propagation on a DAG, assuming that the DAGs are given. Then, we show that it exactly performs linear diffusion on the DAGs. We emphasize the role of our SGPN – to implicitly learn to relate the vertices globally and to refine the embedded representations, by learning the representation for vertices (unary) and edges (pairwise), in (Section 3.2).

3.1. Formulation

Propagation on DAGs. Given a set of vertices $V = \{v_1, ..., v_N\}$ of a DAG, we denote the set of indices of the connected neighbors of v_i as K_i. For example, if a DAG is built along a direction from left to right, and V is a set of points in a point cloud, the vertices in K_i would be the points that are adjacent to v_i and are located spatially to the left of it (see Fig. 2(e)). We denote the feature of each vertex, before and after propagation, as $u \in \mathbb{R}^{N\times c}$ and $h \in \mathbb{R}^{N\times c}$, respectively, where u can be a c-channel feature map obtained from an intermediate layer of a segmentation CNN before propagation, and h would be its value after propagation. We call u and h as the unary and propagated features, respectively. The propagation operator updates the value of h for the various vertices of the graph recurrently (*e.g.*, from left-to-right) as:

$$h(i) = (1 - \sum_{k\in K_i} g_{ik})u(i) + \sum_{k\in K_i} g_{ik}h(k), \qquad (1)$$

where $\{g_{ik}\}$ is a set of learnable affinity values between v_i and v_k, which we denote as the edge representations.

A parallel formulation. In DAGs, since vertices are updated sequentially, propagating features from one vertex to another using linear diffusion in Eq. (1) results in poor parallel efficiency. Here we show that the propagation on DAGs can be re-formulated in a "time-step" manner, which can be implemented in a highly parallel manner. This is achieved via a slight modification of the topological sorting algorithm (see Alg. 1 in the supplementary material) used to construct the DAGs: we re-order the vertices into groups to ensure that (a) vertices in the same group are not linked to each other and can be updated simultaneously, and (b) each group has incoming edges only from its preceding groups. Taking an image as an example, we can construct a left-to-right DAG by connecting all pixels in the t^{th} column to those in the $(t+1)^{th}$ column (see Fig. 3(a)). That is, *column* in an image is equivalent to a *group* in a DAG, where in Eq. (1), pixels from the same column can be computed simultaneously. We denote the corresponding "unary" and

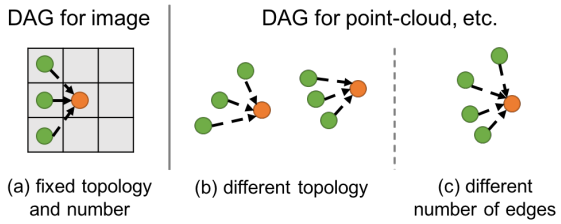


Figure 3. Comparisons of local connections of DAGs between (a) image pixels and (b) (c) irregularity-structured points.

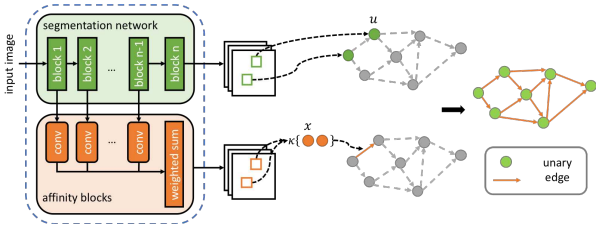


Figure 4. Learning the unary (green) features and the pairwise (orange) features for the edge representations of the DAG from a CNN.

"propagated" features for the vertices in the p^{th} group, before and after propagation as u_p and h_p, respectively. We perform propagation for each group as a linear combination of all its previous groups:

$$h_p = (I - d_p)u_p + \sum_{q=1}^{p-1} w_{pq} h_q, \tag{2}$$

where q is a group that precedes the group p along the direction of propagation. Suppose the p^{th} and q^{th} groups contain m_p and m_q vertices, respectively, w_{pq} is a $m_p \times m_q$ matrix that contains all the corresponding weights $\{g\}$ between vertices in h_p and h_q. Specifically, $d_p \in \mathbb{R}^{m_p \times m_p}$ is a diagonal degree matrix with a non-zero entry at i that aggregates the information from all the $\{w_{pq}\}$ as:

$$d_p(i,i) = \sum_{q=1}^{p-1} \sum_{j=1}^{m_q} w_{pq}(i,j). \tag{3}$$

Re-ordering vertices into groups results in the "time-step" form of Eq. (2), where the update for all vertices in the same group is computed simultaneously. For one direction with the number of groups as T, the computational complexity for propagating on the DAG is $O(T)$. Given Eq. (2), we need to explicitly maintain stability of propagation, which is described in the supplementary material.

Diffusion on Graphs Linear diffusion theory states that the filtering of signals can equivalently be viewed as the solution of a heat conduction or diffusion, where the change of the signal over time can be described as spatial differentiation of the signal at the current state [38]. The theory can be generalized to many other processing, such as refinement of segmentation, where the spatial differentiation needs to be replaced with a task-specific Laplacian matrix.

When fitting diffusion theory into deep neural network, we hope the Laplacian to be learnable and flexibly conditioned on the input signal, through a differentiable linear diffusion module – we achieve this goal on DAGs. We first introduce the notations, where $U = [u_1, ..., u_T] \in \mathbb{R}^{N \times c}$ and $H = [h_1, ..., h_T] \in \mathbb{R}^{N \times c}$ are the features of all the N ordered groups (U and H are re-ordered u and h in Eq. (1)) concatenated together. We re-write Eq. (2) as refining the features U through a global linear transformation $H - U = -LU$. We can derive from both Eq. (2) and Eq. (1) that L meets the requirement of being a Laplacian matrix, whose each row sums to zero. It leads to a standard diffusion process on graphs. Details of proof can be found in the supplementary material.

We note that being linear diffusion process on DAGs is an important property showing that the proposed algorithm is closely related to real physical processes widely used in many image processing techniques [38, 20, 4]. This connection also makes our model more interpretable, for example, the edge representations $\{g_{ik}\}$ in Eq. (1) then explicitly describe the strengths of diffusion in a local region.

3.2. Learning representations on DAGs.

Learnable edge representations. An edge representation $\{g_{ik}\}$ dictates whether the value of a vertex is passed along to its neighbor or not. For the task of semantic segmentation, a desired $\{g_{ik}\}$ should represent a semantic edge (*i.e.*, $g_{ik} = 0$ stops propagation across different categories and $g_{ik} > 0$ allows for propagation within a category) [8, 9, 35]. This implies that the edge representations should be learned and conditioned on the input pixel values instead of being fixed or manually defined. The work of [34] uses the values produced by a CNN as edge representations, *i.e.*, for left-to-right propagation, a 3-channel output is utilized to represent the edges connecting a pixel to its top-left, left, and bottom-left neighbors (Fig. 3(a)), respectively. However, such a method cannot generalize to arbitrarily-structured data since: (a) all vertices must have a fixed number of connected neighbors, and (b) all the connections of all pixels should have the same fixed topology or spatial layout. In contrast, here we are dealing with DAGs constructed from unstructured points (*e.g.*, point clouds) that do not follow either of these assumptions, see Fig. 3(b)(c).

To overcome this limitation, in our work each edge representation g_{ik} used in linear propagation in Eq. (1) is directly computed via a differentiable symmetric kernel function κ (*e.g.*, inner-product), such that $g_{ij} = \kappa(x_i, x_j), j \in K_i$, which is applied to the feature vectors x_i and x_j that are specifically computed to relate vertices v_i and v_j. We denote $x \in \mathbb{R}^{N \times c}$ as feature from a pairwise branch of the CNN. Encoding the graph's edge weights in this manner, allows for each vertex to have a different number and spatial distribution of connected neighbors. It also reduces the task of learning edge representations g_{ik} in Fig. 4 to that of learning common feature representations $\{x_i\}$ that relate the individual vertices. In detail, we use two types of local

similarity kernels:

Inner product (*-prod*). κ can be defined as an inner-product similarity:

$$\kappa(x_i, x_j) = \bar{x_i}^\top \bar{x_j} \tag{4}$$

Here $\bar{x}$ denotes a normalized feature vector, which can be computed in CNNs via Layer Normalization [3].

Embedded Gaussian (*-embed*). We compute the similarity in an embedding space via a Gaussian function.

$$\kappa(x_i, x_j) = e^{-\|x_i - x_j\|_F^2} \tag{5}$$

Since g_{ik} is allowed to have negative values, we add a learnable bias term to the embedded Gaussian and initialize it with a value of -0.5.

Learning Unary and Pairwise Features. Our network contains three blocks – a CNN block (Fig. 2(b)), that learns features from 2D images that correspond to the unstructured data (e.g., multi-view images for a point cloud, Fig. 2(a)), an aggregation block (Fig. 2(c)) to aggregate features from pixels to points, and a propagation (Fig. 2(d)) block that propagates information across the vertices of different types of unstructured data.

We use a CNN block to learn the unary u and pairwise x features jointly. The CNN block can be any image segmentation network (*e.g*. DRN [53]), where the unary term can be the feature maps before the output, or the previous upsampling layer (Fig. 4). Then, both features from image domain are aggregated by averaging the individual feature vectors from one local area corresponding to the same point, to the specific vertex or edge, as shown in Fig. 4.

Since we show that the edge representations $\{g_{ik}\}$ can be computed by applying a similarity kernel to pairs of features x_i and x_j, one could reuse the unary features (i.e., $u_i = x_i$) for computing pairwise affinities as well [47]. However, we find that for semantic segmentation, features from lower levels are of critical importance for computing pairwise affinities because they contain rich object edge or boundary information. Hence, we integrate features from all levels of the CNN, with simple convolutional blocks (*e.g*., one CONV layer for a block) to align the feature dimensions of $\{x\}$ and $\{u\}$. We further use a weighed-sum to integrate the feature maps from each block, where the weights are scalar, learnable parameters, and are initialized with 1 (see the dashed box in Fig. 4).

4. Semantic Segmentation via SGPNs

In this section, we introduce how to build DAGs and embed the learned representations, for refinement of semantic segmentation w.r.t different type of unstructured data.

4.1. Propagation on Pixels and Superpixels

Image. We use the 3-way connection proposed in [34] to build the DAGs for images, *i.e*. each pixel is connected to 3 of its adjacent neighbors in each direction, and propagation is performed in all 4 directions. Different from [34] where the graph edge representations are directly produced by a guidance network that is separate from the segmentation network, in this work we train a single segmentation network to jointly compute both the unary features and the edge representations as the similarity between pairwise features (x_i). Through this task, we demonstrate the effectiveness of our strategy for learning the edge representations, compared with [34] as presented in Section 5.

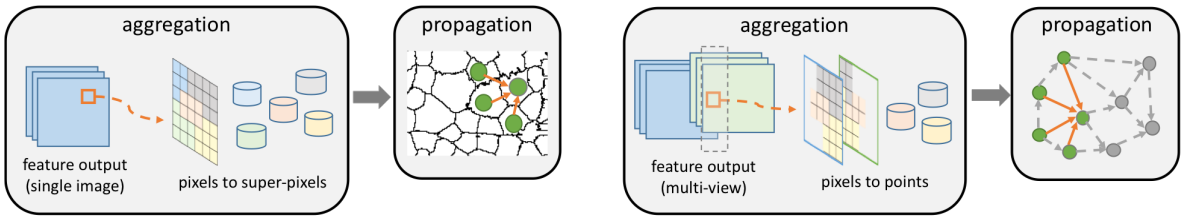
Superpixel. Superpixel is an effective representation to group large irregularly-shaped semantically similar regions of an image (see Fig 5), and thus reduce the number of input elements for subsequent processing tasks. However, it is not easy to utilize superpixels directly as image pixels as they are not arranged on regular grids. Our method can perform propagation on superpixels as an intermediate block by aggregating pixel-level features, performing propagation, and then projecting features from superpixels back to image pixels (we copy the single value of the superpixel to all the image-pixel locations that the superpixel covers).

To perform propagation, we preprocess each superpixel image by constructing a group of DAGs, where superpixels are the vertices, and the connections to their neighbors are the edges. Specifically, we search for the spatially adjacent neighbors of each superpixel, and group them into 4 groups along the 4 directions of the original image (i.e., $\rightarrow, \leftarrow, \uparrow, \downarrow$). To determine whether a superpixel is the neighbor of another superpixel along a specific direction, we compare the locations of their centroids (see an example in Fig. 5). For a 1024×2048 image from the Cityscapes dataset [14] with 15000 superpixels, T is around $100 \sim 200$ and $200 \sim 400$ for vertical and horizontal directions, respectively. This is far more efficient than performing propagation on the original pixels of high-resolution images.

4.2. Propagation on Point Clouds

Unlike many prior methods [39, 29] which learn features from raw points, our method flexibly maps image features to points, for which numerous off-the-shelf network architectures and pretrained weights can be utilized directly by point clouds. The joint 2D-3D training is conducted by establishing the correspondences between pixels and points via camera parameters (not the focus of this work), and aggregating features from CNNs to DAGs according to the correspondences. Note that the same point may correspond to pixels from multiple images (Fig. 5(b) dashed box), where we simply average the features across them. The construction of DAGs is similar to that of superpixels, except that the neighborhoods can be determined directly according to spatial distances between points.

Constructing DAGs Along Surfaces. We observe that constructing the graphs according to local object/scene sur-



(a) an instantiation for super-pixels (b) an instantiation for point clouds

Figure 5. Different diagrams are shown for aggregation and propagation along superpixels and point clouds. See details in Section 4.

faces, instead of XYZ Euclidean space, yields better performance (Section 5). This is consistent with the intuition that local regions belonging to the same smooth and continuous surface are more likely to come from the same object. Similar observations have been made in [13, 44, 40]. In detail, consider a set of neighboring points $k \in K_i$ in a spherical range of i, such that $\|\mathbf{P}(i) - \mathbf{P}(k)\| < R$. The distance between i and k is computed as $(\mathbf{P}(i) - \mathbf{P}(k)) \cdot \mathbf{n}(i)$, where $\mathbf{P}(i)$ denotes the world coordinates of i, and $\mathbf{n}(i)$ is the surface normal. A subset of neighbors with the smallest distances are selected, which are equivalent to a neighborhood in the Tangent space [44].

Geometric-aware Edge Representations. Aside from learning image pixel features, we found that incorporating geometry hints for each point is equally important [43]. The geometry information is the concatenation of point XYZ, surface normal and point color RGB in our work. We map this vector from point to pixels according to the correspondence indices to form a 9-channel input map with the same resolution as the input image, and apply one single Conv+ReLU unit before integrating them with the affinity block. To avoid absolute coordinates (e.g., X are around 50 in the training set, but 90 in the validation set), we replace (X, Y, Z) with the coordinate-gradient (dX, dY, dZ).

5. Experimental Results

We evaluate the performance of the proposed approach on the task of image semantic segmentation, in Sections 5.2 and 5.3, and point cloud segmentation in Section 5.4.

5.1. Datasets and Backbone Networks

Cityscapes [14]. This dataset contains 5000 high quality, high-resolution images finely annotated with street scenes, stuff and objects, in total with 19 categories. We use the standard training and validation sets. For all experiments, we apply random cropping, re-scaling (between the ratio of 0.8 to 1.3), rotation (± 10 degrees) and mirroring for training. We do not adopt any other data augmentation, coarse annotations and hard-mining strategies, in order to analyze the utility of the propagation module itself.

RueMonge2014 [41]. This dataset provides a benchmark for 2D and 3D facade segmentation, which contains 428 multi-view, high-resolution images, and a point cloud with approximately 1M 3D points that correspond to these images. The undetected regions are masked out and ignored in processing. Semantic segmentation labels for both image pixels and points are provided for a total of 7 categories. Our experiments use standard training and validation splits and the same data augmentation methods as the Cityscapes dataset.

Our experiments use two type of backbone networks. To compare against the baseline methods, mean Intersection over Union (IoU) is used as the evaluation metric.

Dilated Residual Networks (DRN). We use DRN-22-D, a simplified version of the DRN-C framework [53] as our primary backbone architecture. This network contains a series of residual blocks, except in the last two levels, each of which is equipped with dilated convolutions. The network is light-weight and divided into 8 consecutive levels and the last layer outputs a feature map that is $8\times$ smaller than the input image. One 1×1 convolution and a bilinear upsampling layer is used after it to produce the final segmentation probability map. We use the network pretrianed on ImageNet [16]. To make the settings consistent between the different experiments, we append our SGPN module to the output of level-8 of the DRN model.

Deeplab Network. We adopt the Deeplab [11] framework by replacing the original encoder with the architecture of a wider ResNets-38 [50] that is more suitable for the task of semantic segmentation. The encoder is divided into 8 levels and we append the SGPN to level-8.

5.2. Image Segmentation: SGPN on Pixels

Propagation with DRN. We perform pixel-wise propagation of the output of the DRN network, and compare it to its baseline performance. We re-train the baseline DRN model with the published default settings provided by the authors [53] and obtain the mean IoUs of 68.34 and 69.17, for single and multi-scale inference, respectively, on the validation set. For SGPN, in the pairwise block, we use features for each level except the last one, and the features from the first convolution layer. We call the features from levels 1 to 3 as lower level features, and 5 to 7 as higher level ones. For the lower level features, the pairwise block contains a combination of CONV+ReLU+CONV, while for the higher level features, we use a single CONV layer since they have the same resolution as the output of the encoder. The lower

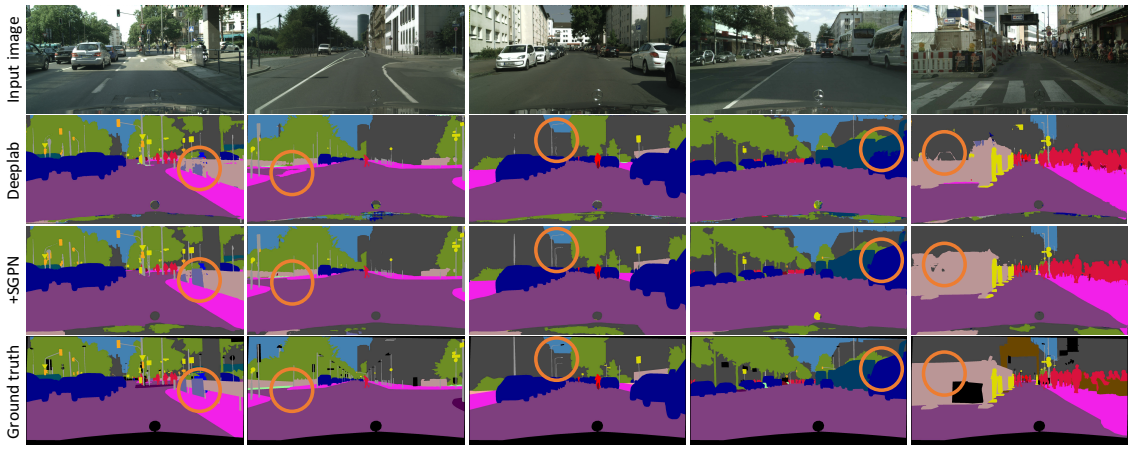


Figure 6. Semantic segmentation results on the Cityscapes dataset via the **Deeplab** network. Regions are circled for ease of comparison.

Table 1. Results for **DRN**-related networks for semantic segmentation on Cityscapes val set. We show the results of multi-scale testing, except for the +SPN, which shows better performance via single scale setting. *embed* and *prod* denote the embedded Gaussian and inner product kernels.

categories	road	sidewalk	building	wall	fence	pole	trafficlight	trafficsign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mIoU
Baseline (DRN) [53]	97.4	80.9	91.1	32.9	54.9	60.6	65.6	75.9	92.1	59.1	93.4	79.2	57.8	92.0	42.9	65.2	55.2	49.4	75.2	69.5
+SPN (single) [34]	97.7	82.7	91.3	34.4	54.6	61.8	65.9	76.4	92.2	62.2	94.4	78.5	56.8	92.7	47.8	70.2	63.6	52.0	75.3	71.1
+NLNN [47]	97.4	81.1	91.2	43.0	52.9	60.3	66.1	74.9	91.7	60.6	93.4	79.2	57.7	93.3	54.4	73.5	54.7	54.2	74.4	71.3
+SGPN-embed	98.0	83.8	92.2	48.5	**59.7**	64.1	70.1	79.4	92.6	**63.8**	**94.7**	82.0	60.7	94.9	62.5	77.7	51.1	**62.8**	**77.6**	74.5
+SGPN-prod	**98.1**	**84.4**	**92.2**	51.8	56.5	**65.8**	**71.2**	**79.4**	**92.7**	63.2	94.3	**82.7**	**65.1**	**94.9**	**73.8**	78.0	43.2	59.7	77.4	**75.0**
+SGPN-superpixels	97.6	82.4	91.0	**52.7**	52.9	58.4	66.1	75.9	91.8	62.2	93.6	79.4	58.2	93.3	62.4	**79.7**	57.1	60.2	75.1	73.2

and higher level features are added together to form the final pairwise features, with 128 channels. In addition, we use two deconvolution layers on the unary and pairwise feature maps to upsample them with a stride of 2, and convert them into 32 channels. Propagation is conducted on the $2\times$ upsampled unary features with compressed (32) channels. As mentioned before, we use the same connections between pixels as [34] in the propagation layer. The feature maps produced by the propagation layer are further bilinearly upsampled to the desired resolution. To better understand the capability of the SGPN module, we adopt the same loss function (i.e., Softmax cross-entropy), optimization solver and hyper-parameters in both the baseline and our model.

Comparison with SPN [34]. We produce a coarse segmentation map for each training image using the baseline network mentioned above. We then re-implement the SPN model in [34] and train it to refine these coarse segmentation maps. The SPN shows obvious improvements over the baseline model with mIoUs of 71.1 and 70.8 for single and multi-scale implementations, respectively. However, the SPN may not equally improve different models, because the edge representation is not jointly trained with the segmentation network, e.g., the multi-scale implementation does not consistently outperform the single-scale one.

Comparison with NLNN [47]. We compare with the NLNN – one of the most effective existing modules for learning affinity (Fig 1(c)). We implement this method to make it comparable to ours, by using the same pairwise-blocks as ours for computing affinities, but by replacing the propagation layer with the non-local layer (see details in [47]). This method achieves reasonably higher performance (mIoU: 71.3) versus the baseline method and is also comparable to the SPN method.

Among all the others, our SGPN method, with different kernels (Section 3.2) produces significantly better results with the final mIoU of 75, with most categories been obviously improved, as shown in Table 1.

Propagation with Deeplab Network. We embed the SGPN into a Deeplab-based network to show that it also improves the performance of a superior base network. We demonstrate the significant improvement achieved by our models, as measured by the mIoU as in Table 3. The complete table of results can be found in the supplementary material. Note that SPN does not show any gain on Deeplab, and NLNN consumes a large amount of GPU memory since a fully-connected graph needs to be constructed (i.e., an $N \times N$ matrix), and thus cannot be directly applied to large networks such as Deeplab.

5.3. Image Segmentation: SGPN on Superpixels

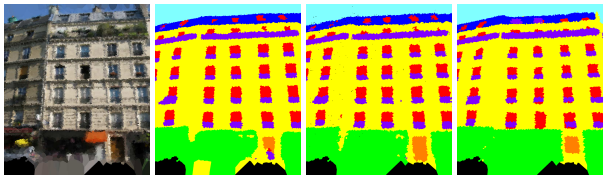
We implement SGPN with superpixels (15000 per image) created by the entropy rate superpixel method [32].

Table 2. Results for point cloud segmentation on the RueMonge2014 [41] val set. "image to points" is the direct mapping of 2D segmentation results to 3D points; "+method" is short for ours+method; "PF" denotes the pairwise features from 2D image; "PG" is the pairwise features with geometry-aware input; "TG" is short for tangent and "EU" is short for Euclidean.

	image segmentation		image to points		point-cloud segmentation					
method	SplatNet$_{2D}$	ours DRN	SplatNet$_{2D}$	ours DRN	SplatNet$_{2D3D}$	+CONV-1D	+PF/TG	+PG/TG	+PF+PG/EU	+PF+PG/TG
mean IoU (%)	69.30	68.17	68.26	69.16	69.80	70.35	72.19	72.43	72.16	**73.66**

Table 3. Results for **Deeplab** based networks for Cityscapes image semantic segmentation on the val set.

mean IoU (%)	Baseline [11]	+SGPN-embed	+SGPN-prod
single-scale	78.20	80.12	80.09
multi-scale	78.97	80.42	**80.90**



(a) input (b) DRN (c) SGPN (d) GT

Figure 7. Qualitative comparison visualized by points to image mapping. "DRN" is for the direct mapping of results from the DRN image segmentation to points, "SGPN" is for our method.

For this experiment, we use the same design as the SGPN network for pixels with the DRN backbone, but replace the pixel propagation layer with the superpixel propagation layer. Since the model still performs pixel-wise labeling, we can directly compare it with the other DRN-related networks, as shown in Table 1. Our SGPN with superpixel propagation shows considerable performance boost over the baseline model. The results demonstrate that SGPN introduces an effective way of utilizing superpixels, which are generally difficult for deep learning methods to process.

5.4. Semantic Segmentation on Point Clouds

Implementation. Processing of point clouds for facade segmentation is different from that of superpixels: while each image corresponds to one superpixel image, in this task, many images correspond to a single point could. Therefore, we do not need different DAGs for different images, instead, a single group of DAGs for the point is constructed. During training, each mini-batch contains a group of sampled patches from the multi-view images, where both the unary and pairwise feature maps across samples are aggregated and mapped on to the globally constructed DAGs. During testing, we perform propagation on the entire validation set, by obtaining the unary and pairwise feature maps from the CNN blocks of all images, and aggregate them on the entire point cloud. We use both 2D and 3D ground-truth semantic labels as the supervision signals.

Comparison with Baselines and SOTA. We use the DRN-22-D as the CNN block. To make fair comparisons against state-of-the-art work [43], we evaluate the performance of the DRN for the task of multi-view image segmentation. One direct way is to aggregate the results of image segmentation and map them on to 3D points (image to point in Table 2). In addition, we jointly train the CNN block and the propagation module by adding a single 1×1 1D CONV layer before the output. Table 2 shows the performance of the baseline models. Our DRN model shows comparable performance to [43] on both image labeling and point labeling with direct aggregation of features from multiple images (see column 1 and 2 in Table 2). The baseline model with the joint training strategy, denoted as "+CONV-1D", obtains the best results and outperforms the state-of-the-art method [43] (the SplatNet$_{2D3D}$ in Table 2), which is not jointly trained with 2D and 3D blocks.

Ablation Study. We show the performance of our proposed approach with (a) geometric information as an additional input stream for edge representation learning, and (b) using the Tangent space to construct the DAGs (PF+PG/TG in Table 2) shows the best results compared to the baseline and state-of-the-art methods [43]. To understand the contributions of individual factors, we carry out two ablation studies. First, we compare various input streams for learning the edge representation, by removing either the geometry information (Section 5.4), or the image pairwise features, from the CNN block (See +PF and +PG in Table 2). When removing the image stream, we use an independent CNN block using the geometry input to produce the pairwise feature maps. Second, we compare models with the same input settings for learning the edge representation, but using different ways to construct the DAGs, i.e., constructing neighborhoods via the Euclidean or Tangent spaces (Section 5.4) (See +PF+PG/EU and +PF+PG/TG in Table 2). The results demonstrate that, by solely applying the semantic feature representation learned from 2D images, or the geometry information, we can still obtain much higher mIoU compared to all of the baseline models. However, utilizing both of them yields the best performance. It indicates that both factors are essential for guiding the propagation on point clouds. On the other hand, constructing the DAGs for point clouds along the Tangent space shows a significant advantage over the Euclidean space.

6. Conclusion

In this work, we propose SGPN that models the global affinity for data with arbitrary structures, including but not limited to superpixels and point clouds. The SGPN conducts learnable linear diffusion on DAGs, with significant advantage of representing the underlying structure of the data. With the module, our method constantly outperforms state-of-the-art methods on semantic segmentation tasks with different types of input data.

References

[1] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762, 2010.

[2] Volker Aurich and Jörg Weule. Non-linear gaussian filters performing edge preserving diffusion. In *Mustererkennung 1995*, pages 538–545. Springer, 1995.

[3] Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[4] Christos George Bampis, Petros Maragos, and Alan C. Bovik. Graph-driven diffusion and random walk schemes for image segmentation. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 2016.

[5] Moshe Bar. Visual objects in context. *Nature Reviews Neuroscience*, 5:617–629, 2004.

[6] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *CVPR*, 2005.

[7] Siddhartha Chandra and Iasonas Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In *European Conference on Computer Vision*, pages 402–418. Springer, 2016.

[8] Liang-Chieh Chen, Jonathan T Barron, George Papandreou, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4545–4554, 2016.

[9] Liang-Chieh Chen, Jonathan T Barron, George Papandreou, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *CVPR*, 2016.

[10] Liang-Chieh Chen, Alexander Schwing, Alan Yuille, and Raquel Urtasun. Learning deep structured models. In *International Conference on Machine Learning*, pages 1785–1794, 2015.

[11] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[12] Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *ECCV*, 2018.

[13] Hang Chu, Wei-Chiu Ma, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Surfconv: Bridging 3d and 2d convolution for rgbd images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[14] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[15] Nathan de Lara and Edouard Pineau. A simple baseline algorithm for graph classification. *arXiv preprint arXiv:1810.09155*, 2018.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.

[17] Santosh K Divvala, Derek Hoiem, James H Hays, Alexei A Efros, and Martial Hebert. An empirical study of context in object detection. In *CVPR*, 2009.

[18] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[19] Raghudeep Gadde, Varun Jampani, Martin Kiefel, Daniel Kappler, and Peter V Gehler. Superpixel convolutional networks using bilateral inceptions. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[20] Leo Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (11):1768–1783, 2006.

[21] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *ECCV*, 2010.

[22] Shengfeng He, Rynson WH 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.

[23] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, 2016.

[24] Varun Jampani, Martin Kiefel, and Peter V Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4452–4461, 2016.

[25] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel sampling networks. In *European Conference on Computer Vision*, pages 363–380. Springer, 2018.

[26] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3D shape segmentation with projective convolutional networks. In *CVPR*, 2017.

[27] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011.

[28] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289, 2001.

[29] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NIPS*, 2018.

[30] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph lstm. In *ECCV*, 2016.

[31] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203, 2016.

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

[33] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Switchable temporal propagation network. *ECCV*, 2018.

[34] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. Learning affinity via spatial propagation networks. In *NIPS*, 2017.

[35] Sifei Liu, Jinshan Pan, and Ming-Hsuan Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *European Conference on Computer Vision*, 2016.

[36] Michael Maire, Takuya Narihira, and Stella X. Yu. Affinity cnn: Learning pixel-centric pairwise relations for figure/ground embedding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[37] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[38] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence*, 12(7):629–639, 1990.

[39] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[40] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[41] Hayko Riemenschneider, Andras Bodis-Szomoru, Julien Weissenberg, and Luc Van Gool. Learning where to classify in multi-view semantic segmentation. In *ECCV*, 2014.

[42] Gergo Sastyin, Ryosuke Niimi, and Kazuhiko Yokosawa. Does object view influence the scene consistency effect? *Attention, Perception, & Psychophysics*, 77(3):856–866, 2015.

[43] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.

[44] Maxim Tatarchenko*, Jaesik Park*, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3D. *CVPR*, 2018.

[45] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.

[46] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[47] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[48] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, 2018.

[49] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter. In *CVPR*, 2018.

[50] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. arXiv:1611.10080, 2016.

[51] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *CVPR*, pages 4606–4615, 2018.

[52] Xiangyu Xu, Deqing Sun, Sifei Liu, Wenqi Ren, Yu-Jin Zhang, Ming-Hsuan Yang, and Jian Sun. Rendering portraitures from monocular camera and beyond. In *ECCV*, 2018.

[53] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *CVPR*, 2017.

[54] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *CVPR*, pages 1529–1537, 2015.