Learning Depth with Convolutional Spatial Propagation Network

Xinjing Cheng, Peng Wang and Ruigang Yang, *Senior Member, IEEE*

Abstract—Depth prediction is one of the fundamental problems in computer vision. In this paper, we propose a simple yet effective convolutional spatial propagation network (CSPN) to learn the affinity matrix for various depth estimation tasks. Specifically, it is an efficient linear propagation model, in which the propagation is performed with a manner of recurrent convolutional operation, and the affinity among neighboring pixels is learned through a deep convolutional neural network (CNN). We can append this module to any output from a state-of-the-art (SOTA) depth estimation networks to improve their performances. In practice, we further extend CSPN in two aspects: 1) allow it to take sparse depth map as additional input, which is useful for the task of depth completion [2], [3]; 2) extend to 3D CSPN to handle features with one additional dimension, similar to the commonly used 3D convolution operation. It is effective in the task of stereo matching using 3D cost volume [4]. For the task of single image depth estimation and depth completion, we experiment the proposed CPSN conjunct algorithms over the popular NYU v2 [5] and KITTI [6] datasets, where we show that our proposed algorithms not only produce high quality (e.g., **30**% more reduction in depth error), but also run faster (e.g., **2** to **5** $\times$ faster) than previous SOTA spatial propagation network [7]. We also evaluated our stereo matching algorithm on the Scene Flow [8] and KITTI Stereo datasets [6], [9], and rank **1st** on both the KITTI Stereo 2012 and 2015 benchmarks , which demonstrates the effectiveness of the proposed module. The code of CSPN proposed in this work will be release at https://github.com/XinJCheng/CSPN.

Index Terms—Spatial Propagation Networks, Depth Estimation, Depth Completion, Stereo Matching.

1 Introduction

Depth estimation, *i.e.,* predicting per-pixel distance to the camera, from a single image or a pair of stereo images, has many applications in practice, *e.g.,* augmented realities (AR), autonomous driving [10], robotics [11], [12], [13]. It also serves as foundation supporting other computer vision problems, such as 3D reconstruction [14], [15] and recognition [16], [17].

For single image depth estimation, recent efforts have yielded high-quality outputs by taking advantage of deep fully convolutional neural networks [1], [18] and large amount of training data from indoor [5], [19], [20] and outdoor [6], [21], [22]. The improvement lies mostly in more accurate estimation of global scene layout and scales with advanced networks, such as VGG [23] and ResNet [24], and better local structure recovery through deconvolution operation [25], skip-connections [26] or up-projection [1]. Nevertheless, upon closer inspection of the output from a contemporary approach [2] (Fig. 1(b)), the predicted depths is still blurry and do not align well with the given image structure such as object silhouette.

Most recently, Liu *et al.* [7] propose to directly learn the image-dependent affinity through a deep CNN with spatial propagation networks (SPN), yielding better results comparing to the manually designed affinity on image segmentation. However, its propagation is performed in a scan-line or scan-column fashion, which is serial in nature. For instance, when propagating left-to-right, pixels at right-most column must wait the information from the left-most column to update its value. Intuitively, depth refinement commonly just needs a local context rather a global one.

Here we propose convolutional spatial propagation networks (CSPN), where the depths at all pixels are updated simultaneously within a local convolutional context. The long range context is obtained through a recurrent operation. Fig. 1 shows an example, the depth estimated from CSPN (e) is more accurate than that from SPN (d) and Bilateral filtering (c). In our experiments, our parallel update scheme leads to significant performance improvement in both speed and quality over the serial ones such as SPN.

Practically, depth from a single image is still an ill-posed problem that is under research, it attracts more interests of industry to joint consider depth from devices such as LiDAR [27] or stereo camera [28]. Therefore, in this work, we show how to easily extend the proposed CSPN to depth estimation tasks under these scenario, *i.e.,* depth completion [3] with sparse depth collected from LiDAR, and stereo matching [29] from a pair of images, by slightly adjusting the way of affinity learning, which also yields significantly improvements over other SOTA methods.

Specifically, depth completion, a.k.a. sparse to dense depth conversion, is a task of converting sparse depth samples to a dense depth map given corresponding image [2], [30]. This task can be widely applied in robotics and autonomous cars, where depth perception is often acquired through LiDAR, which usually generates sparse but accurate depth measurement. By combining the sparse measurements with images, we could generate a full-frame dense depth map. For this task, we consider three important requirements for an algorithm: (1) The dense depth map recovered should align with image structures; (2) The depth value from the sparse samples should be preserved, since they are usually from a reliable sensor; (3) The transition between sparse depth samples and their neighboring depths should be smooth and unnoticeable. In order to satisfy those requirements, we first add mirror connections based on the network from [2], which generates better depths as shown in Fig. 1(h). Then, we tried to embed the propagation into SPN in order to keep the depth value at sparse points. As shown in Fig. 1(i), it generates better details and lower error than

- *X. Cheng, P. Wang, R. Yang are with Baidu Research, Baidu Inc., Beijing, China.*

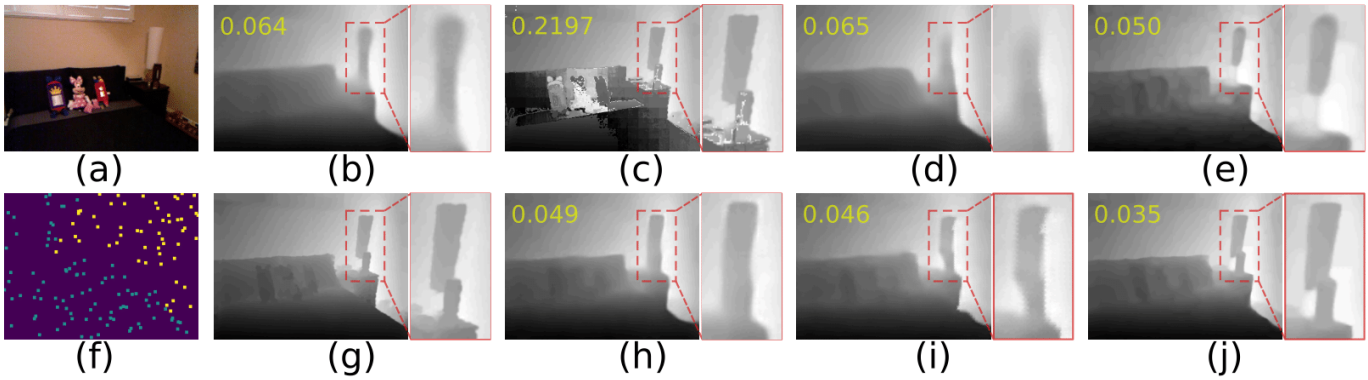


Fig. 1: (a) Input image; (b) Depth from [2]; (c) Depth after bilateral filtering; (d) Refined depth by SPN [7]; (e) Refined depth by CSPN; (f) Sparse depth samples (500); (g) Ground Truth; (h) Depth from our network; (i) Refined depth by SPN with depth sample; (j) Refined depth by CSPN with depth sample. The corresponding root mean square error (RMSE) is put at the left-top of each predicted depth map.

SPN without depth samples (Fig. 1(d)). Finally, changing SPN to our CSPN yields the best result (Fig. 1(j)). As can be seen, our recovered depth map with just 500 depth samples produces much more accurately estimated scene layouts and scales.

On the other hand, stereo matching estimates a disparity d for each pixel in the reference image. Specifically, for pixel (x,y) in the reference image, if its corresponding disparity is $d_{x,y}$, then the depth of this pixel could be calculated by $\frac{f*B}{d_{(x,y)}}$, where f is the camera's focal length and B is the distance between two camera centers. Current SOTA methods for solving this problem are also relying on the development of deep networks [4], [31], [32]. Most recently, GCNet [32] learns to incorporate geometrical context directly from the data, employing 3D convolutions (3DConv) over $height \times width \times disparity$ dimensions by separating the continuous disparity to *discretized disparity valued space*, yielding an end-to-end training system with results has better recovered scene structure. PSMNet [4] adopt similar idea, while induces extensions at *scale space* by using spatial feature pooling [33] at end of feature encoder and multi-scale outputs from stacked hourglass networks [34] with 3DConv. This motivates us to lift the 2D spatially propagating CSPN to 3D, where information can also propagate along the disparity value space and scale space, yielding more accurate estimated results with more details, which we will elaborate in Sec. 3.3.

We perform various experiments to validate our approaches on different tasks over several popular benchmarks for depth estimation. For single image depth estimation and depth completion, NYU v2 [5] and KITTI [6] are adopted. In both datasets, our approach is significantly better (relative 30% improvement in most key measurements) than previous deep learning based state-of-the-art (SOTA) algorithms [2], [30]. More importantly, it is very efficient yielding 2-5× acceleration comparing with SPN. For stereo depth estimation, the Scene Flow [8] and KITTI Stereo datasets [6], [9] are adopted, and we rank the 1_{st} on both the KITTI Stereo 2012 and 2015 benchmarks [1], which is also much better than the results from PSMNet [4] that we base on.

In summary, this paper has the following contributions:

1) We propose convolutional spatial propagation networks (CSPN) which is more efficient and accurate for depth estimation than the previous SOTA propagation strategy [7], without sacrificing the theoretical guarantee.
2) We extend CSPN to the task of converting sparse depth samples to dense depth map by using the provided sparse depths into the propagation process. It guarantees that the sparse input depth values are preserved in the final depth map. It runs in real-time, which is well suited for robotics and autonomous driving applications, where sparse depth measurement from LiDAR can be fused with image data.
3) We propose to lift 2D CPSN to 3D for stereo matching, which explores the correlation within both discrete disparity space and scale space. It helps the recovered stereo depth generate more details and avoid error matching from noisy appearance cause by sunlights or shadows *etc*..

The structure of this paper is organized as follows. We provide related work in Sec. 2, and elaborate the design and theoretical background of CSPN in Sec. 3.1. In Sec. 3.2.1 and Sec. 3.3, we present the details about our extension of CSPN to depth completion and stereo matching correspondingly. Finally, we evaluate the results of our algorithms on all the tasks quantitatively and qualitatively in Sec. 4.

2 Related Work

Depth estimating and enhancement/refinement have long been center problems for computer vision and robotics. Here we summarize those works in several aspects without enumerating them all due to space limitation.

Single view depth estimation via CNN and CRF. Deep neural networks (DCN) developed in recent years provide strong feature representation for per-pixel depth estimation from a single image. Numerous algorithms are developed through supervised methods [1], [18], [35], [36], semi-supervised methods [37] or unsupervised methods [38], [39], [40], [41]. and add in skip and mirror connections. Others tried to improve the estimated details further by appending a conditional random field (CRF) [42], [43], [44] and joint training [45], [46]. However, the affinity for measuring the coherence of neighboring pixels is manually designed based on color similarity or intervening contour [47] with RBF kernel [46].

Depth Enhancement. Traditionally, depth output can be also efficiently enhanced with explicitly designed affinity through image filtering [48], [49], or data-driven ones through total variation

1. http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo

(TV) [50], [51] and learning to diffuse [52] by incorporating more priors into diffusion partial differential equations (PDEs). However, due to the lack of an effective learning strategy, they are limited for large-scale complex visual enhancement.

Recently, deep learning based enhancement yields impressive results on super resolution of both images [53], [54] and depths [55], [56], [57], [58]. The network takes low resolution inputs and output the high-resolution results, and is trained end-to-end where the mapping between input and output is implicitly learned. However, these methods are only trained and experimented with perfect correspondent ground-truth low-resolution and high-resolution depth maps and often a black-box model. In our scenario, both the input and ground truth depth are non-perfect, *e.g.,* depths from a low cost LiDAR or a network, thus an explicit diffusion process to guide the enhancement such as SPN is necessary.

Learning affinity for spatial diffusion. Learning affinity matrix with deep CNN for diffusion or spatial propagation receives high interests in recent years due to its theoretical supports and guarantees [59]. Maire *et al.* [60] trained a deep CNN to directly predict the entities of an affinity matrix, which demonstrated good performance on image segmentation. However, the affinity is followed by an independent non-differentiable solver of spectral embedding, it can not be supervised end-to-end for the prediction task. Bertasius *et al.* [61] introduced a random walk network that optimizes the objectives of pixel-wise affinity for semantic segmentation. Nevertheless, their affinity matrix needs additional supervision from ground-truth sparse pixel pairs, which limits the potential connections between pixels. Chen *et al.* [62] try to explicit model an edge map for domain transform to improve the output of neural network.

The most related work with our approach is SPN [7], where the learning of a large affinity matrix for diffusion is converted to learning a local linear spatial propagation, yielding a simple while effective approach for output enhancement. However, as mentioned in Sec. 1, depth enhancement commonly needs local context, it might not be necessary to update a pixel by scanning the whole image. As shown in our experiments, our proposed CSPN is more efficient and provides much better results.

Depth estimation with given sparse samples. The task of sparse depth to dense depth estimation was introduced in robotics due to its wide application for enhancing 3D perception [30]. Different from depth enhancement, the provided depths are usually from low-cost LiDAR or one line laser sensors, yielding a map with valid depth in only few hundreds of pixels, as illustrated in Fig. 1(f). Most recently, Ma *et al.* [2] propose to treat sparse depth map as additional input to a ResNet [1] based depth predictor, producing superior results than the depth output from CNN with solely image input. However, the output results are still blurry, and does not satisfy our requirements of depth as discussed in Sec. 1. In our case, we directly embed the sampled depth in the diffusion process, where all the requirements are held and guaranteed.

Some other works directly convert sparse 3D points to dense ones without image input [3], [63], [64], whereas the density of sparse points must be high enough to reveal the scene structure, which is not available in our scenario.

Stereo with CNNs Stereo depth estimation has long been a center problem in computer vision. Traditionally, Scharstein and Szeliski [65] provides a taxonomy of stereo algorithms into four steps: matching cost calculations, matching cost aggregation, disparity calculation and disparity refinement [66], [67], [68].

CNNs were first introduced to stereo matching by Zbontar and LeCun [31] to replace the computation of the matching cost. Their method showed that by using CNNs, the matching could be more robust, and achieved SOTA results over KITTI Stereo benchmarks. However, the networks are still shallow, and it needs post-processing for refinement. Following [31], several methods were proposed to increase computational efficiency [69], [70], or matching cost accuracy [71] with stronger network and confidence predictions. Later, some works are focusing on post-process by incorporating top-down knowledge from objects such as Displets [72].

This inspires the study of stereo matching networks to develop a fully learnable architecture without manually designed processing. DispNet [8], FlowNet [73] are designed to find 2D optical flow by inserting two corresponding frames, which can be easily extend to stereo matching by limiting the searching within an disparity line. However, they did not fully take use of the limited range for stereo matching. In order to densely model per-pixel disparity matching, GCNet [32] proposes to generate a 3D cost volume of size $height \times width \times disparity$ by densely comparing the feature at pixel (i, j) from the reference image to all possible matching pixels within disparity line at target image. The network can figure out the best matching disparity through $soft-argmin$ operation. PWCNet [74] follows similar idea while having cost volume calculated within a local region within size of $d \times d$. PSMNet [4] embraces the experience of semantic segmentation studies, which exploits scale space through pyramid spatial pooling and hourglass networks for capturing global image context, yielding better results than GCNet. As can be seen, both GCNet and PSMNet are benefited from exploring a new dimension, *i.e.,* disparity value space and scale space respectively, which motivates us to extend CSPN to 3D. Built upon PSMNet, 3D CSPN considers modeling the relationship with diffusion along their proposed new dimension, and produces more robust results.

Spatial pyramid for hierarchical context. As we also explore scale space for a dense prediction model, we would like to review spatial pyramid pooling (SPP) [33] to provide more insight for our proposed model. Liu *et al.* [75] first propose SPP to increase the empirical receptive field of a fully convolutional network. Such an idea is demonstrated to be very effective in the study of both semantic segmentation [76], [77], [78], and depth estimation, *e.g.,* PSMNet [4] as we discussed. Here, in our perspective, the parameters for SPP form a scale space which is manually set and experimentally determined based on certain dataset by previous works [76]. However, our CSPN with 3D convolution can learn the affinity for fusing the proposed scale space, which softly discovers the proper scale of context for the network. We show in our experiments, such strategy effectively improve the depth estimation results over PSMNet. In the near future, we will extend such idea to semantic segmentation for validating the generalization of such a strategy.

3 Our Approach

In this section, we first introduce the CSPN module we proposed, which is an anisotropic diffusion process and the diffusion tensor is learned through a deep CNN directly from the given image. Then we describe how we applied this module to the each of tasks we mentioned above.

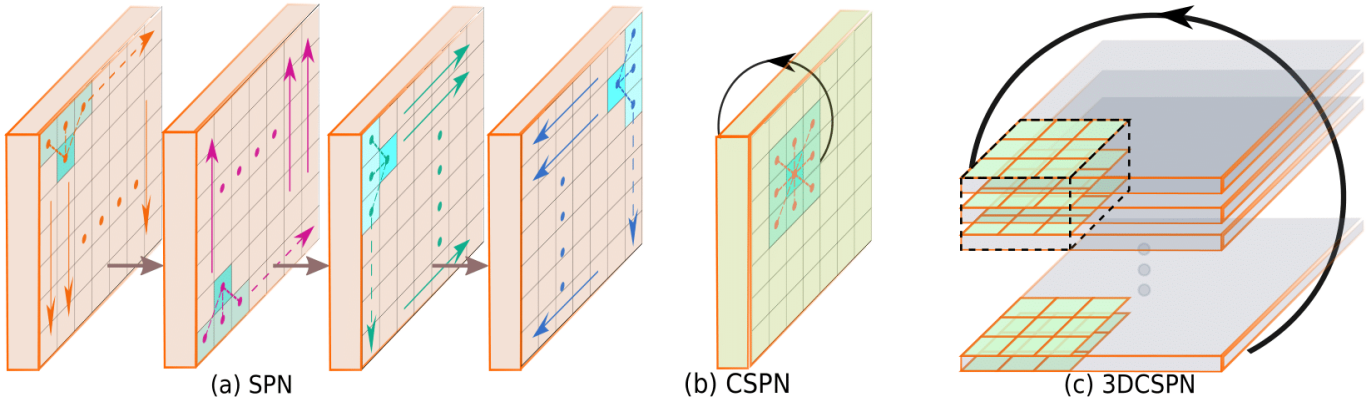


Fig. 2: Comparison between the propagation process in (a) SPN [7], (b) 2D CPSN and (c) 3D CSPN in this work. Notice for 3D CSPN, the dashed volume means one slice of the feature channel in a 4D volume with size of $disparity \times height \times width \times channel$ (Detailed in Sec. 3.3).

3.1 Convolutional spatial propagation network (CSPN)

Given a depth map $D_o \in \mathbf{R}^{m\times n}$ that is output from a network, and image $\mathbf{X} \in \mathbf{R}^{m\times n}$, our task is to update the depth map to a new depth map D_n within N iteration steps used the module we described in Sec. 3.1, which first reveals more details of the image, and second improves the per-pixel depth estimation results.

Fig. 2(b) illustrates our updating operation in 2D spatial propagation. Formally, without loss of generality, we can embed the depth map $D_o \in \mathbf{R}^{m\times n}$ to some hidden space $\mathbf{H} \in \mathbf{R}^{m\times n\times c}$. The convolutional transformation functional with a kernel size of k for each time step t could be written as,

$$\begin{aligned} \mathbf{H}_{i,j,t+1} &= \textstyle\sum_{a,b=-(k-1)/2}^{(k-1)/2} \kappa_{i,j}(a,b) \odot \mathbf{H}_{i-a,j-b,t} \\ \text{where,}\quad \kappa_{i,j}(a,b) &= \frac{\hat{\kappa}_{i,j}(a,b)}{\sum_{a,b,a,b\neq 0} |\hat{\kappa}_{i,j}(a,b)|}, \\ \kappa_{i,j}(0,0) &= 1 - \textstyle\sum_{a,b,a,b\neq 0} \kappa_{i,j}(a,b) \end{aligned} \tag{1}$$

where the transformation kernel $\hat{\kappa}_{i,j} \in \mathbf{R}^{k\times k\times c}$ is the output from an affinity network, which is spatially dependent on the input image. The kernel size k is usually set as an odd number so that the computational context surrounding pixel (i,j) is symmetric. $\odot$ is element-wise product. Following [7], we normalize kernel weights between range of $(-1,1)$ so that the model can be stabilized and converged by satisfying the condition $\sum_{a,b,a,b\neq 0} |\kappa_{i,j}(a,b)| \leq 1$. Finally, we perform this iteration N steps to reach a stationary distribution.

Correspondence to diffusion process with a partial differential equation (PDE). Similar with [7], here we show that our CSPN holds all the desired properties of SPN. Formally, we can rewrite the propagation in Eq. (1) as a process of diffusion evolution by first doing column-first vectorization of feature map $\mathbf{H}$ to $\mathbf{H}_v \in \mathbf{R}^{mn\times c}$.

$$\mathbf{H}_v^{t+1} = \begin{bmatrix} 1-\lambda_{0,0} & \kappa_{0,0}(1,0) & \cdots & 0 \\ \kappa_{1,0}(-1,0) & 1-\lambda_{1,0} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \cdots & \cdots & 1-\lambda_{m,n} \end{bmatrix} = \mathbf{G}\mathbf{H}_v^t \tag{2}$$

where $\lambda_{i,j} = \sum_{a,b} \kappa_{i,j}(a,b)$ and $\mathbf{G}$ is a $mn\times mn$ transformation matrix. The diffusion process expressed with a partial differential equation (PDE) is derived as follows,

$$\begin{aligned} \mathbf{H}_v^{t+1} &= \mathbf{G}\mathbf{H}_v^t = (\mathbf{I} - \mathbf{D} + \mathbf{A})\mathbf{H}_v^t \\ \mathbf{H}_v^{t+1} - \mathbf{H}_v^t &= -(\mathbf{D} - \mathbf{A})\mathbf{H}_v^t \\ \partial_t \mathbf{H}_v^{t+1} &= -\mathbf{L}\mathbf{H}_v^t \end{aligned} \tag{3}$$

where $\mathbf{L}$ is the Laplacian matrix, $\mathbf{D}$ is the diagonal matrix containing all the $\lambda_{i,j}$, and $\mathbf{A}$ is the affinity matrix which is the off diagonal part of $\mathbf{G}$.

In our formulation, different from [7] which scans the whole image in four directions (Fig. 2(a)) sequentially, CSPN propagates a local area towards all directions at each step (Fig. 2(b)) simultaneously, *i.e.,* with $k\times k$ local context, while larger context is observed when recurrent processing is performed, and the context acquiring rate is in an order of $O(kN)$.

In practical, we choose to use convolutional operation due to that it can be efficiently implemented through image vectorization, yielding real-time performance in depth refinement tasks.

Principally, CSPN could also be derived from loopy belief propagation with sum-product algorithm [79]. However, since our approach adopts linear propagation, which is efficient, while just a special case of pairwise potential with L2 reconstruction loss in graphical models. Therefore, to make it more accurate, we call our strategy *convolutional spatial propagation* in the field of diffusion process.

Complexity analysis. As formulated in Eq. (1), our CSPN takes the operation of convolution, where the complexity of using CUDA with GPU for one step CSPN is $O(\log_2(k^2))$, where k is the kernel size. This is because CUDA uses parallel sum reduction, which has logarithmic complexity. In theory, convolution operation can be performed parallel for all pixels and channels, which has a constant complexity of $O(1)$. Therefore, performing N-step propagation, the overall complexity for CSPN is $O(\log_2(k^2)N)$, which is irrelevant to image size (m,n).

SPN [7] adopts scanning row/column-wise propagation in four directions. Using k-way connection and running in parallel, the complexity for one step is $O(\log_2(k))$. The propagation needs to scan full image from one side to another, thus the complexity for SPN is $O(\log_2(k)(m+n))$. Though this is already more efficient than the densely connected CRF proposed by [80], whose implementation complexity with permutohedral lattice is $O(mnN)$, ours $O(\log_2(k^2)N)$ is more efficient since the number of iterations N is always much smaller than the size of image m,n. We show in our experiments (Sec. 4), with $k=3$ and $N=12$, CSPN already

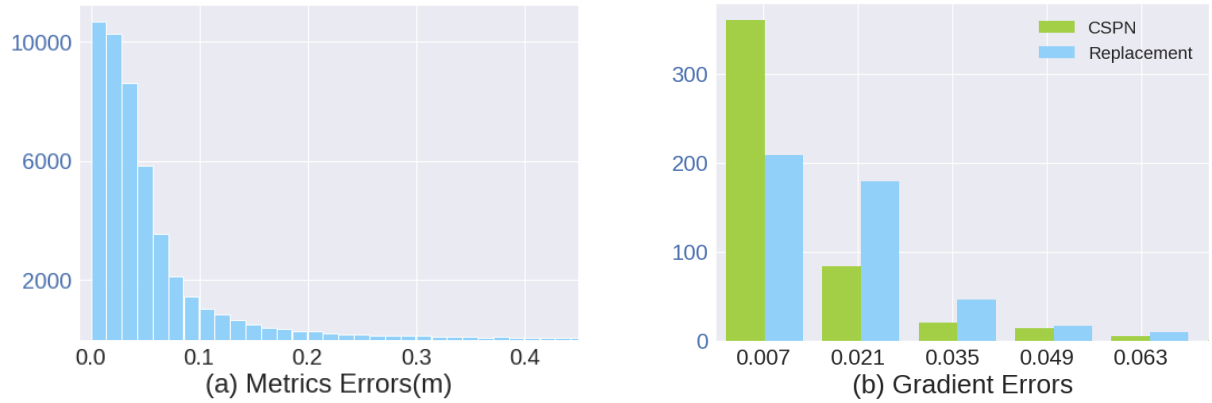


Fig. 3: (a) Histogram of RMSE with depth maps from [2] at given sparse depth points. (b) Comparison of gradient error between depth maps with sparse depth replacement (blue bars) and with ours CSPN (green bars), where ours is much smaller. Check Fig. 4 for an example. Vertical axis shows the count of pixels.

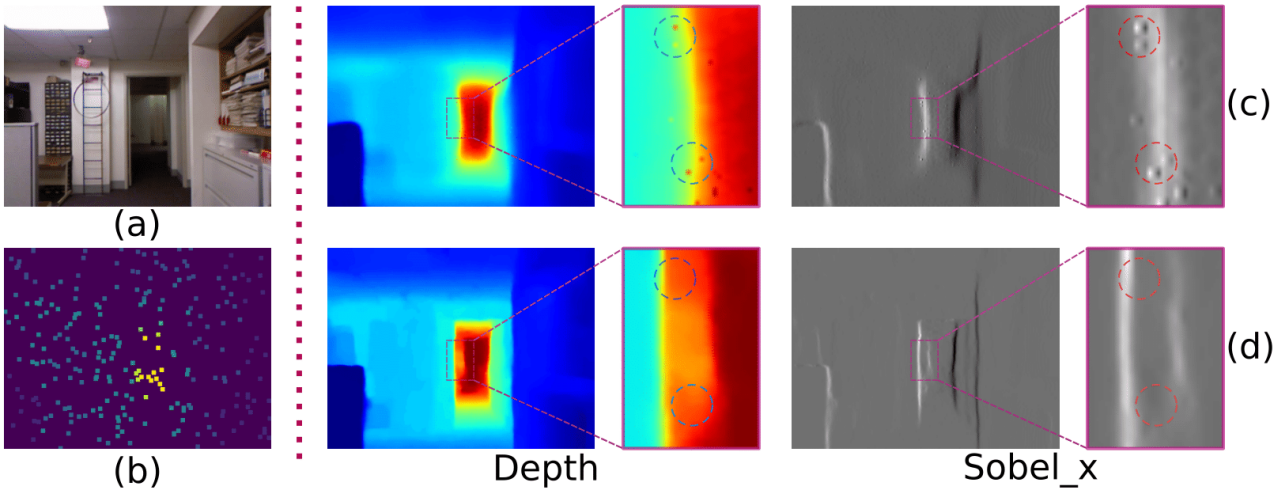


Fig. 4: Comparison of depth map [2] with sparse depth replacement and with our CSPN w.r.t. smoothness of depth gradient at sparse depth points. (a) Input image. (b) Sparse depth points. (c) Depth map with sparse depth replacement. (d) Depth map with our CSPN with sparse depth points. We highlight the differences in the red box.

outperforms SPN with a large margin (relative 30%) in accuracy, demonstrating both efficiency and effectiveness of the proposed approach.

3.2 CSPN for depth completion

In this application, we have an additional sparse depth map D_s (Fig. 4(b)) to help estimate a depth depth map from a RGB image. Specifically, a sparse set of pixels are set with real depth values from some depth sensors, which can be used to guide our propagation process.

3.2.1 Spatial Propagation with Sparse Depths

Similarly, we also embed the sparse depth map $D_s = \{d^s_{i,j}\}$ to a hidden representation $\mathbf{H}^s$, and we can write the updating equation of $\mathbf{H}$ by simply adding a replacement step after performing Eq. (1),

$$\mathbf{H}_{i,j,t+1} = (1 - m_{i,j})\mathbf{H}_{i,j,t+1} + m_{i,j}\mathbf{H}^s_{i,j} \tag{4}$$

where $m_{i,j} = \mathbf{I}(d^s_{i,j} > 0)$ is an indicator for the availability of sparse depth at (i, j).

In this way, we guarantee that our refined depths have the exact same value at those valid pixels in sparse depth map. Additionally, we propagate the information from those sparse depth to its surrounding pixels such that the smoothness between the sparse depths and their neighbors are maintained. Thirdly, thanks to the diffusion process, the final depth map is well aligned with image structures. This fully satisfies the desired three properties for this task which is discussed in our introduction (Sec. 1).

In addition, this process is still following the diffusion process with PDE, where the transformation matrix can be built by simply replacing the rows satisfying $m_{i,j} = 1$ in $\mathbf{G}$ (Eq. (2)), which are corresponding to sparse depth samples, by $\mathbf{e}^T_{i+j*m}$. Here $\mathbf{e}_{i+j*m}$ is an unit vector with the value at $i + j * m$ as 1. Therefore, the summation of each row is still 1, and obviously the stabilization still holds in this case.

Our strategy has several advantages over the previous state-of-the-art sparse-to-dense methods [2], [30]. In Fig. 3(a), we plot a histogram of depth displacement from ground truth at given sparse depth pixels from the output of Ma *et al.* [2]. It shows the accuracy of sparse depth points cannot preserved, and some pixels could have very large displacement (0.2m), indicating that directly training a CNN for depth prediction does not preserve the value of real sparse depths provided. To acquire such property, one may simply replace the depths from the outputs with provided sparse depths at those pixels, however, it yields non-smooth depth

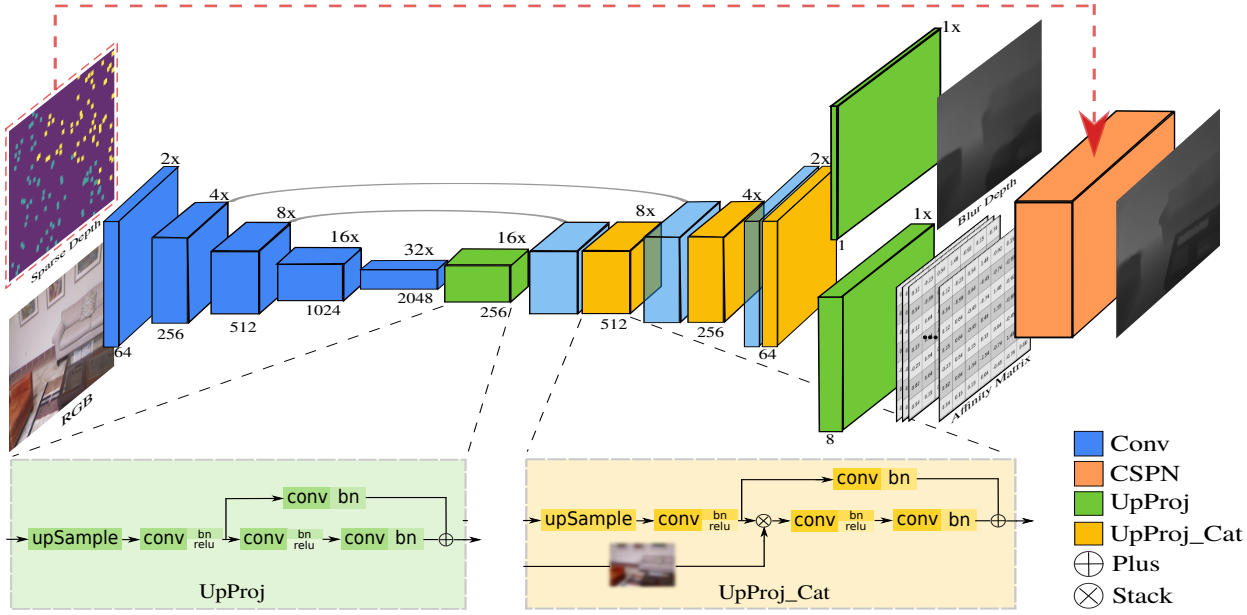


Fig. 5: Architecture of our networks with mirror connections for depth estimation via transformation kernel prediction with CSPN (best view in color). Sparse depth is an optional input, which can be embedded into the CSPN to guide the depth refinement.

gradient w.r.t. surrounding pixels. In Fig. 4(c), we plot such an example, at right of the figure, we compute Sobel gradient [81] of the depth map along x direction, where we can clearly see that the gradients surrounding pixels with replaced depth values are non-smooth. We statistically verify this in Fig. 3(b) using 500 sparse samples, the blue bars are the histogram of gradient error at sparse pixels by comparing the gradient of the depth map with sparse depth replacement and of ground truth depth map. We can see the difference is significant, 2/3 of the sparse pixels has large gradient error. Our method, on the other hand, as shown with the green bars in Fig. 3(b), the average gradient error is much smaller, and most pixels have zero error. InFig. 4(d), we show the depth gradients surrounding sparse pixels are smooth and close to ground truth, demonstrating the effectiveness of our propagation scheme.

3.2.2 Architecture estimating single image depths

We now explain our end-to-end network architecture to predict both the transformation kernel and the depth value, which are the inputs to CSPN for depth refinement. As shown in Fig. 5, our network has some similarity with that from Ma *et al.* [2], with the final CSPN layer that outputs a dense depth map.

For predicting the transformation kernel κ in Eq. (1), rather than building a new deep network for learning affinity same as Liu *et al.* [7], we branch an additional output from the given network, which shares the same feature extractor with the depth network. This helps us to save memory and time cost for joint learning of both depth estimation and transformation kernels prediction.

Learning of affinity is dependent on fine grained spatial details of the input image. However, spatial information is weaken or lost with the down sampling operation during the forward process of the ResNet in [1]. Thus, we add mirror connections similar with the U-shape network [26] by directed concatenating the feature from encoder to up-projection layers as illustrated by "UpProj_Cat" layer in Fig. 5. Notice that it is important to carefully select the end-point of mirror connections. Through experimenting three possible positions to append the connection, *i.e.,* after *conv*, after *bn* and after *relu* as shown by the "UpProj" layer in Fig. 5 , we found the last position provides the best results by validating with the NYU v2 dataset (Sec. 4.1.2). In doing so, we found not only the depth output from the network is better recovered, and the results after the CSPN is additionally refined, which we will show the experiment section (Sec. 4). Finally we adopt the same training loss as [2], yielding an end-to-end learning system.

3.3 3D CSPN for stereo matching

In this section, we present the second extension of CSPN for stereo depth estimation using 3D CSPN, as shown in Fig. 2(c). Consider a prediction from PSMNet with maximum disparity d, the output map from a pair of stereo images has shape of $D_o \in \mathbf{R}^{d\times h\times w}$, where h and w are feature height and width. Our task is to update the output map to a new map D_n within N iteration steps, where diffusion along all three dimensions are jointly performed, yielding a prediction revealing better details and structures inside the image. Formally, the depth our formation for 3D CSPN could be written as,

$$\mathbf{H}_{i,j,l,t+1} = \sum_{a,b,c=-(k-1)/2}^{(k-1)/2} \kappa_{i,j,l}(a,b,c) \odot \mathbf{H}_{i-a,j-b,l-c,t}$$
$$\text{where,} \quad \kappa_{i,j,l}(a,b,c) = \frac{\hat{\kappa}_{i,j,l}(a,b,c)}{\sum_{a,b,c|a,b,c\neq 0} |\hat{\kappa}_{i,j,l}(a,b,c)|},$$
$$\kappa_{i,j,l}(0,0,0) = 1 - \sum_{a,b,c|a,b,c\neq 0} \kappa_{i,j,l}(a,b,c) \qquad (5)$$

which simply adds a new dimension for propagation comparing to Eq. (1), and we can see the original theoretical properties are all well maintained by vertization over all three dimensions. We then perform such an operation w.r.t. both disparity value space at the end of PSMNet, and scale space for spatial pyramid pooling in the middle of the network, which we will elaborate later.

3.3.1 Architecture for stereo matching

Here, we first illustrates the full network architecture similar to PSMNet [78] in Fig. 6. The left and right images are fed to

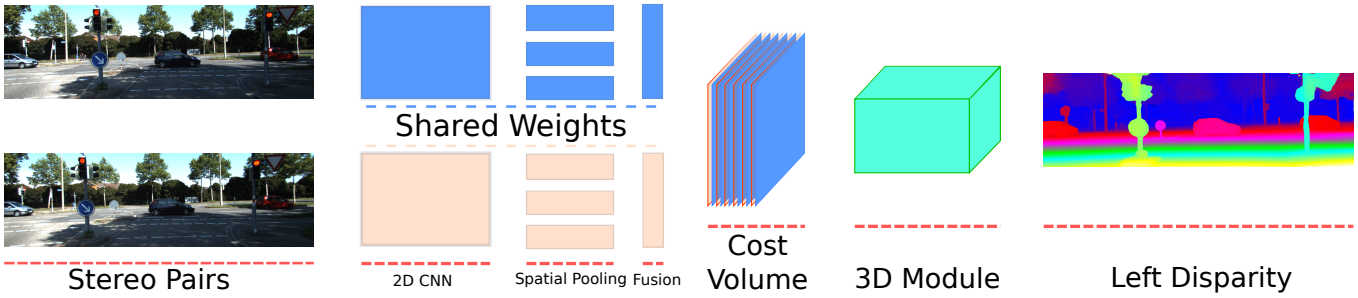


Fig. 6: Architecture of our networks for stereo depth estimation via transformation kernel prediction with 3D CSPN (best view in color).

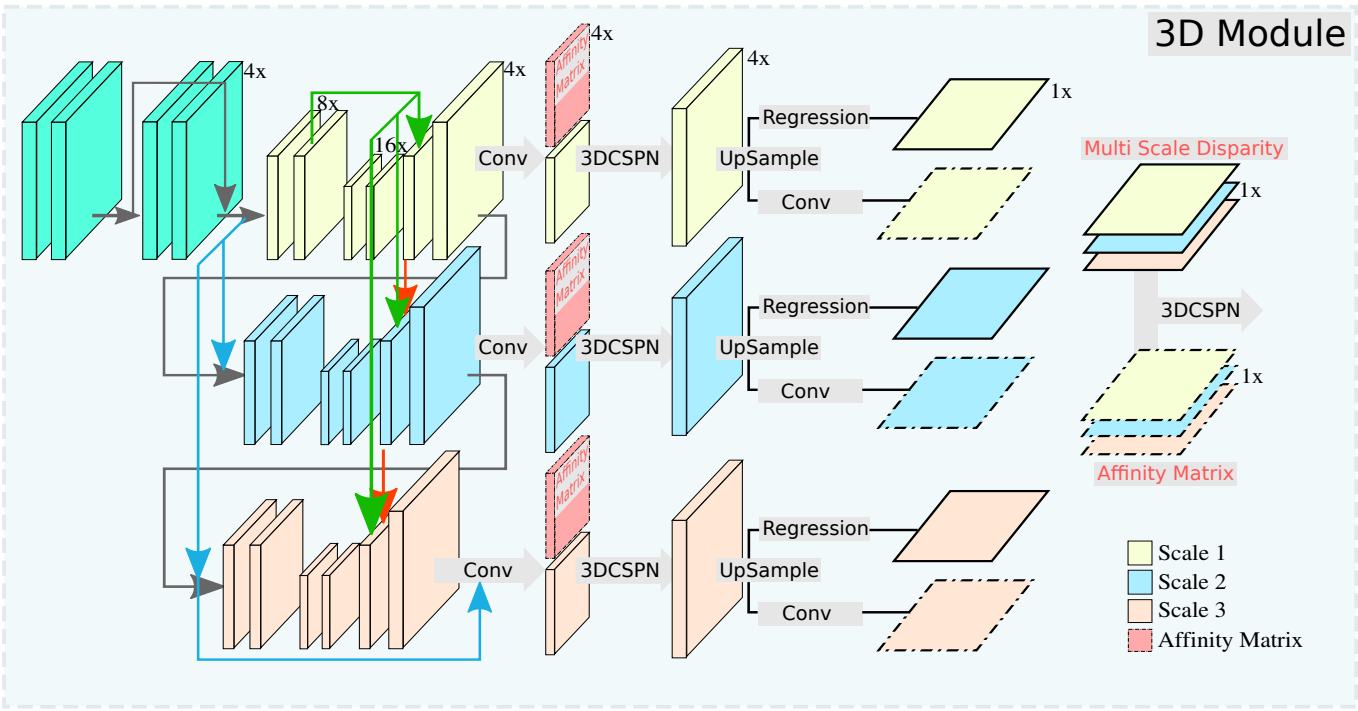


Fig. 7: Details of our 3D Module (best view in color). Downsample rate w.r.t. image size is shown at the right top corner of each block, *e.g.*, 4x means the size of the feature map is $\frac{h}{4} \times \frac{w}{4}$ where $h \times w$ is image size.

two weight-sharing CNN, yielding corresponding feature maps, a spatial pooling module for feature harvesting by concatenating representations from sub-regions with different sizes. The two produced feature maps are then used to form a 4D cost volume, which is fed into a 3D CNN for disparity regression. We refer readers to the original paper for more details due to space limitation. Here, we update their 3D module and spatial pooling with our proposed 3D CSPN, which will be clarified in the follow two paragraphs.

3D CSPN over disparity and scale space In Fig. 7, we zoom into the 3D module to clarify the architecture we applied for disparity regression. In PSMNet, three predicted disparity volumes with size of $d/4 \times h/4 \times w/4 \times 1$ are output at different stages from a stacked hourglass network. Here d, h, w is the maximum disparity, height and width of the input image correspondingly. Similar to the appending strategy of 2D CSPN for single image depth prediction in Sec. 3.3, after the disparity volume at each stage, we append a 3D CSPN with kernel size $k \times k \times k$ to combine the contexts from neighbor pixels, where the affinity matrix is learned from the same feature block as the outputs. Then, bilinear upsampling is applied to upsample a disparity volume to $d \times h \times w \times 1$ for disparity map regression, yielding an output with shape of $h \times w \times 1$.

To fuse the multiple disparity maps from different stages, PSMNet manually sets the weight to average the outputs. In our case, we concatenate them into a 4D volume with size $s \times h \times w \times 1$, where $s = 3$ is the number of disparity maps. Similarly, we can perform a 3D CSPN with kernel size as $s \times k \times k$ to connect the multi-stage predictions, which is conceptually similar as attention models for multi-scale feature fusion [82]. Last, we use feature padding with size of $[0, 1, 1]$, so that the first dimension is reduced to 1 with one iteration, and we obtain a single regressed disparity map with shape $h \times w \times 1$ for final depth estimation. We list the details of the kernel size k and iterations in ablation study at Sec. 4.2.

Spatial pyramid pooling as a special case of CSPN. The second module in the architecture we enhanced for stereo matching is the spatial pyramid pooling (SPP) as illustrated in Fig. 8(a). Here, we can see that SPP can be treated as a special case of CSPN given proper kernel size and convolution stride. Formally, given a feature map with size of $h \times w$, and target pooled feature map with size of $p \times q$, spatial pooling computes the mean value within each parted grid with size of $h/p \times w/q$. This is equivalent to one step CSPN (Eq. (1)) by setting both convolution kernel size

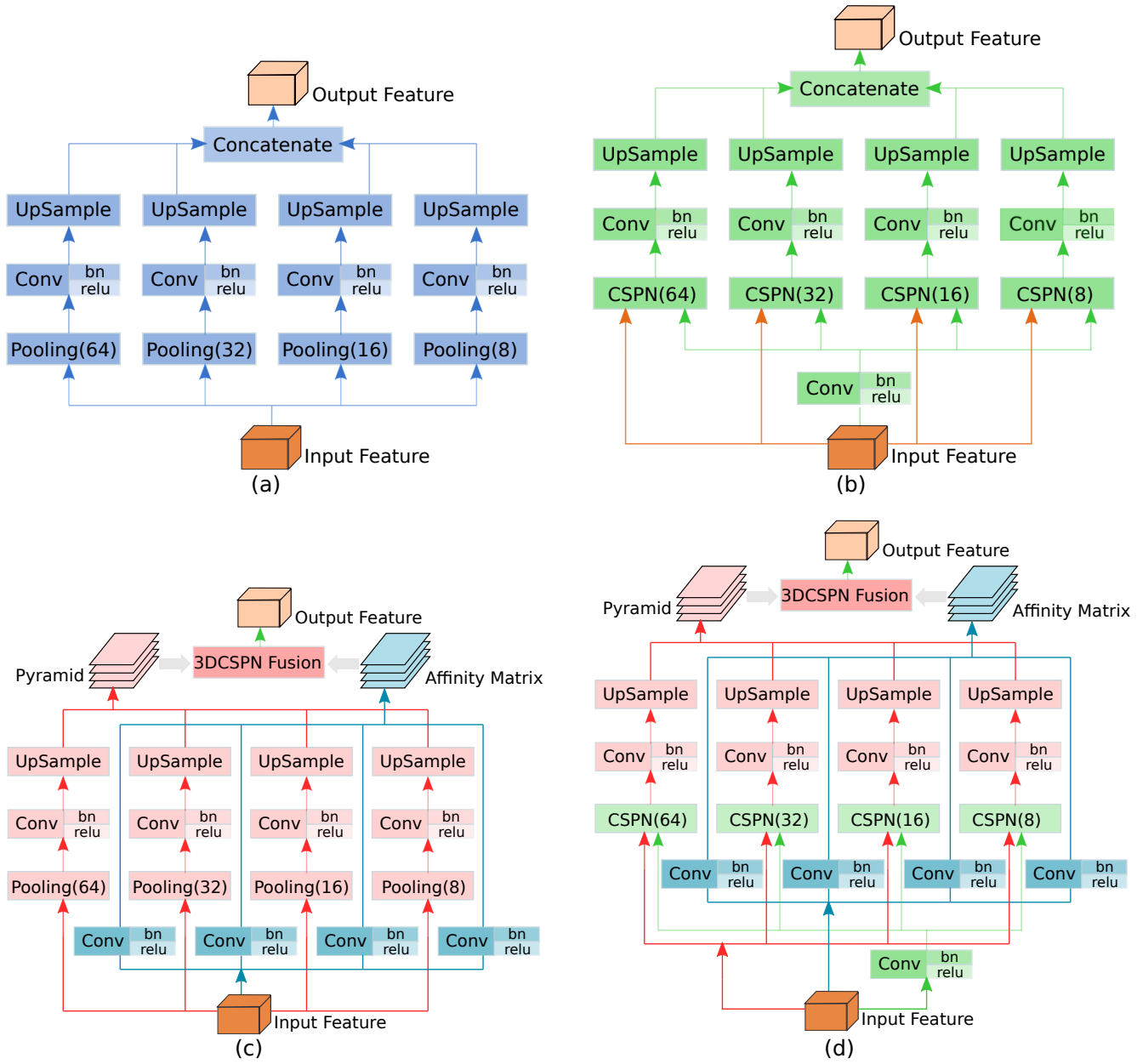


Fig. 8: Different structures of context pyramid module. (a) spatial pyramid pooling (SPP) module applied by PSMNet [78] (b) Our weighted SPP (WSPP) module using 2D CSPN with different kernel size and stride. (b) Our weighted spatial pyramid fusion (WSPF) using 3D CSPN. (d) Our final combined SPP module. (Details in Sec. 3.3.1)

and stride to be $h/p \times w/q$, and all the values in $\kappa(a,b)$ to be the same. However, we know that features can be very different at impacting the final performance as shown in many attention models [83]. Therefore, we propose to learn such a pooling kernel $\kappa(a,b)$ using CSPN for this SPP module. As shown in Fig. 8(b), in our case, we output an affinity matrix from the same feature block for SPP, based on which we do one step 2D CSPN, yielding the required pooled feature map with size of $p \times q$. Specifically, feature maps with target sizes of 64 x 64, 32 x 32, 16 x16 and 8 x 8 are adopted (Fig. 8(a)), and all the feature maps share the same network output for computing the pooling kernels. In other words, the network outputs an one channel weight map with size of $h \times w \times 1$, and for each target size of pooled feature, we first partitioning the weight map to pooling regions, and compute the pooling kernel $\kappa()$ within each region following Eq. (1). We call our strategy of multi-scale feature computing as weighted spatial pyramid pooling (WSPP) to simplify our description later.

Last, we need to fuse the feature maps from all the layers of spatial pyramid. Rather than direct concatenating all the pooled features into a feature map with size $h \times w \times lc$ as PSMNet, we adopt strategy similar to fusing multi-stage disparity maps, which is illustrated in Fig. 8(c). Specifically, we concatenate the output spatial pyramid features into a 4D volume with size $l \times h \times w \times c$, and learn a transformation kernel with size of $l \times 3 \times 3$, yielding a fused feature map with size $h \times w \times c$. It is then fed to cost volume computation at later stages (Fig. 6). Here, l is the layer number of the spatial pyramid, and we use one independent branch for computing the transformation kernel for each layer. We call this strategy as weighted spatial pyramid fusion (WSPF) to simplify our description. Our final spatial pooling strategy is a combination

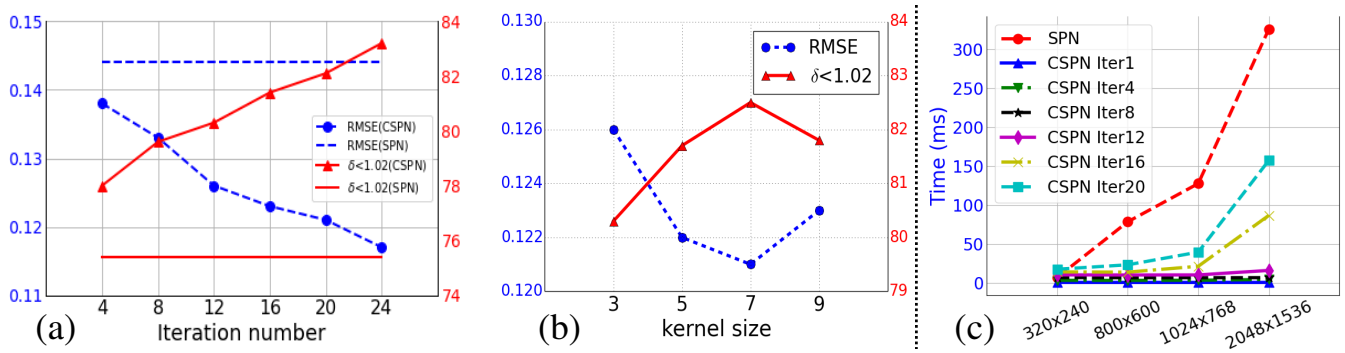


Fig. 9: Ablation study.(a) RMSE (left axis, lower the better) and $\delta < 1.02$ (right axis, higher the better) of CSPN w.r.t. number of iterations. Horizontal lines show the corresponding results from SPN [7]. (b) RMSE and $\delta < 1.02$ of CSPN w.r.t. kernel size. (c) Testing times w.r.t. input image size.

of WSPP and WSPF, as shown in Fig. 8(d), which produces significant performance boost over the original SPP module.

Finally, we also tried ASPP [84] to replace SPP for multi-scale feature pooling without feature size reduction. Specifically, ASPP use dilated convolution to obtain features within various context. Similarly, our CSPN can be performed in the same manner of dilated convolution by learning a spatial dependent transformation kernel. Therefore, we can also extend ASPP to weighted ASPP (WASPP) for computing hierarchical features. In our experiments, we adopt a set of dilation rates for ASPP including 6 x 6, 12 x 12, 18 x18 and 24 x 24, and found WASPP achieves better performance than that from WSPP, which we use at last for competing over the KITTI benchmarks.

Finally, for training the full network, We use the same *soft-argmin* disparity regression method proposed by GCNet [32] to convert finial discretized disparity to continuous value.

$$\widehat{d} = \sum_{d=0}^{D_{max}} d \cdot \sigma(-c_d) \tag{6}$$

Then, the continuous disparity is compared against the ground truth disparity value using the L_1 loss. Formally, the loss function is defined as:

$$L\left(d^*, \widehat{d}\right) = \frac{1}{N}\sum_{i=1}^{N} \|d^* - \widehat{d}\|_1, \tag{7}$$

where d^* is a ground truth disparity, and $\widehat{d}$ is the predicted disparity from Eq. (6).

4 Experiments

In this section, we first describe our implementation details, the datasets and evaluation metrics used in our experiments, and then present comprehensive evaluation of CSPN on various tasks we proposed.

4.1 Depth estimation and depth completion

In this section, we evaluate the conjunct algorithms with single image as input, which includes tasks of single image depth estimation and depth completion with sparse points.

Implementation details. Following the network proposed in [1], [2], the weights of ResNet in the encoding layers for depth estimation (Sec. 3.3) are initialized with models pretrained on the ImageNet dataset [86]. Our models are trained with SGD optimizer, and we use a small batch size of 24 and train for 40 epochs for all the experiments, and the model performed best on the validation set is used for testing. The learning rate starts at 0.01, and is reduced to 20% every 10 epochs. A small weight decay of 10^{-4} is applied for regularization. We implement our networks based on PyTorch [2] platform, and use its element-wise product and convolution operation for our one step CSPN implementation.

For depth, we show that propagation with hidden representation **H** only achieves marginal improvement over doing propagation within the domain of depth D. Therefore, we perform all our experiments direct with D rather than learning an additional embedding layer. For sparse depth samples, we adopt 500 sparse samples as that is used in [2].

4.1.1 Datasets and Metrics

All our experiments are evaluated on two datasets: NYU v2 [5] and KITTI Odometry [6], using commonly used metrics.

NYU v2. The NYU-Depth-v2 dataset consists of RGB and depth images collected from 464 different indoor scenes. We use the official split of data, where 249 scenes are used for training and we sample 50K images out of the training set with the same manner as [2]. For testing, following the standard setting [18], [46], the small labeled test set with 654 images is used the final performance. The original image of size 640×480 are first downsampled to half and then center-cropped, producing a network input size of 304×228.

KITTI odometry dataset. It includes both camera and LiDAR measurements, and consists of 22 sequences. Half of the sequence is used for training while the other half is for evaluation. Following [2], we use all 46k images from the training sequences for training, and a random subset of 3200 images from the test sequences for evaluation. Specifically, we take the bottom part 912×228 due to no depth at the top area, and only evaluate the pixels with ground truth.

Metrics. We adopt the same metrics and use their implementation in [2]. Given ground truth depth $D^* = \{d^*\}$ and predicted depth $D = \{d\}$, the metrics include: (1) RMSE: $\sqrt{\frac{1}{|D|}\sum_{d\in D}||d^* - d||^2}$. (2) Abs Rel: $\frac{1}{|D|}\sum_{d\in D}|d^* - d|/d^*$. (3) δ_t: % of $d \in D$, s.t. $max(\frac{d^*}{d}, \frac{d}{d^*}) < t$, where $t \in \{1.25, 1.25^2, 1.25^3\}$. Nevertheless, for the third metric, we found that the depth accuracy is very high when sparse depth is provided, $t = 1.25$ is already a very loosen criteria where almost 100% of pixels are judged as correct, which can hardly distinguish different methods as shown in (Tab. 1).

2. http://pytorch.org/

TABLE 1: Comparison results on NYU v2 dataset [5] between different variants of CSPN and other state-of-the-art strategies. Here, "Preserve SD" is short for preserving the depth value at sparse depth samples.

Method	Preserve "SD"	Lower the better		Higher the better					
		RMSE	REL	$\delta_{1.02}$	$\delta_{1.05}$	$\delta_{1.10}$	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$
Ma *et al.* [2]		0.230	0.044	52.3	82.3	92.6	97.1	99.4	99.8
+Bilateral [48]		0.479	0.084	29.9	58.0	77.3	92.4	97.6	98.9
+SPN [52]		0.172	0.031	61.1	84.9	93.5	98.3	99.7	99.9
+CSPN (Ours)		0.162	0.028	64.6	87.7	94.9	98.6	99.7	99.9
+UNet (Ours)		0.137	0.020	78.1	91.6	96.2	98.9	99.8	100.0
+ASAP [85]	✓	0.232	0.037	59.7	82.5	91.3	97.0	99.2	99.7
+Replacement	✓	0.168	0.032	56.5	85.7	94.4	98.4	99.7	99.8
+SPN [52]	✓	0.162	0.027	67.5	87.9	94.7	98.5	99.7	99.9
+UNet(Ours)+SPN	✓	0.144	0.022	75.4	90.8	95.8	98.8	99.8	100.0
+CSPN (Ours)	✓	0.136	0.021	76.2	91.2	96.2	99.0	99.8	100.0
+UNet+CSPN (Ours)	✓	**0.117**	**0.016**	**83.2**	**93.4**	**97.1**	**99.2**	**99.9**	100.0

Thus we adopt more strict criteria for correctness by choosing $t \in \{1.02, 1.05, 1.10\}$.

4.1.2 Ablation study for CSPN Module

Here, we evaluate various hyper-parameters including kernel size k, number of iterations N in Eq. (1) using the NYU v2 dataset for single image depth estimation. Then we provide an empirical evaluation of the running speed with a Titan X GPU on a computer with 16 GB memory.

Number of iterations. We adopt a kernel size of 3 to validate the effect of iteration number N in CSPN. As shown in Fig. 9(a), our CSPN has outperformed SPN [7] (horizontal line) when iterated only four times. Also, we can get even better performance when more iterations are applied in the model during training. From our experiments, the accuracy is saturated when the number of iterations is increased to 24.

Size of convolutional kernel. As shown in Fig. 9(b), larger convolutional kernel has similar effect with more iterations, due to larger context is considered for propagation at each time step. Here, we hold the iteration number to $N = 12$, and we can see the performance is better when k is larger while saturated at size of 7. We notice that the performance drop slightly when kernel size is set to 9. This is because we use a fixed number of epoch, *i.e.,* 40, for all the experiments, while larger kernel size induces more affinity to learn in propagation, which needs more epoch of data to converge. Later, when we train with more epochs, the model reaches similar performance with kernel size of 7. Thus, we can see using kernel size of 7 with 12 iterations reaches similar performance of using kernel size of 3 with 20 iterations, which shows CSPN has the trade-off between kernel size and iterations. In practice, the two settings run with similar speed, while the latter costs much less memory. Therefore, we adopt kernel size as 3 and number of iterations as 24 in our comparisons.

Concatenation end-point for mirror connection. As discussed in Sec. 3.3, based on the given metrics, we experimented three concatenation places, *i.e.,* after *conv*, after *bn* and after *relu* by fine-tuning with weights initialized from encoder network trained without mirror-connections. The corresponding RMSE are 0.531, 0.158 and 0.137 correspondingly. Therefore, we adopt the proposed concatenation end-point.

Running speed In Fig. 9(c), we show the running time comparison between the SPN and CSPN with kernel size as 3. We use the author's PyTorch implementation online. As can be seen, we can get better performance within much less time. For example, four iterations of CSPN on one 1024×768 image only takes 3.689 *ms*, while SPN takes 127.902 *ms*. In addition, the time cost of SPN is linearly growing w.r.t. image size, while the time cost of CSPN is irrelevant to image size and much faster as analyzed in Sec. 3.1. In practice, however, when the number of iterations is large, *e.g.,* "CSPN Iter 20", we found the practical time cost of CSPN also grows w.r.t. image size. This is because of PyTorch-based implementation, which keeps all the variables for each iteration in memory during the testing phase. Memory paging cost becomes dominant with large images. In principle, we can eliminate such a memory bottleneck by customizing a new operation, which will be our future work. Nevertheless, without coding optimation, even at high iterations with large images, CSPN's speed is still twice as fast as SPN.

4.1.3 Comparisons

We compare our methods against various SOTA baselines in terms of the two proposed tasks. (1) Refine the depth map with the corresponding color image. (2) Refine the depth using both the color image and sparse depth samples. For the baseline methods such as SPN [52] and Sparse-to-Dense [2], we use the released code released online from the authors.

NYU v2. Tab. 1 shows the comparison results. Our baseline methods are the depth output from the network of [2], together with the corresponding color image. At upper part of Tab. 1 we show the results for depth refinement with color only. At row "Bilateral", we refine the network output from [2] using bilateral filtering [48] as a post-processing module with their spatial-color affinity kernel tuned on our validation set. Although the output depths snap to image edges (Fig. 1(c)), the absolute depth accuracy is dropped since the filtering over-smoothed original depths. At row "SPN", we show the results filtered with SPN [7], using the author provided affinity network. Due to joint training, the depth is improved with the learned affinity, yielding both better depth details and absolute accuracy. Switching SPN to CSPN (row "CSPN") yields relative better results. Finally, at the row "UNet", we show the results of just modifying the network with mirror connections as stated in Sec. 3.3. The results turn out to be even better than that from SPN and CSPN, demonstrating that by simply adding feature from beginning layers, the depth can be better learned.

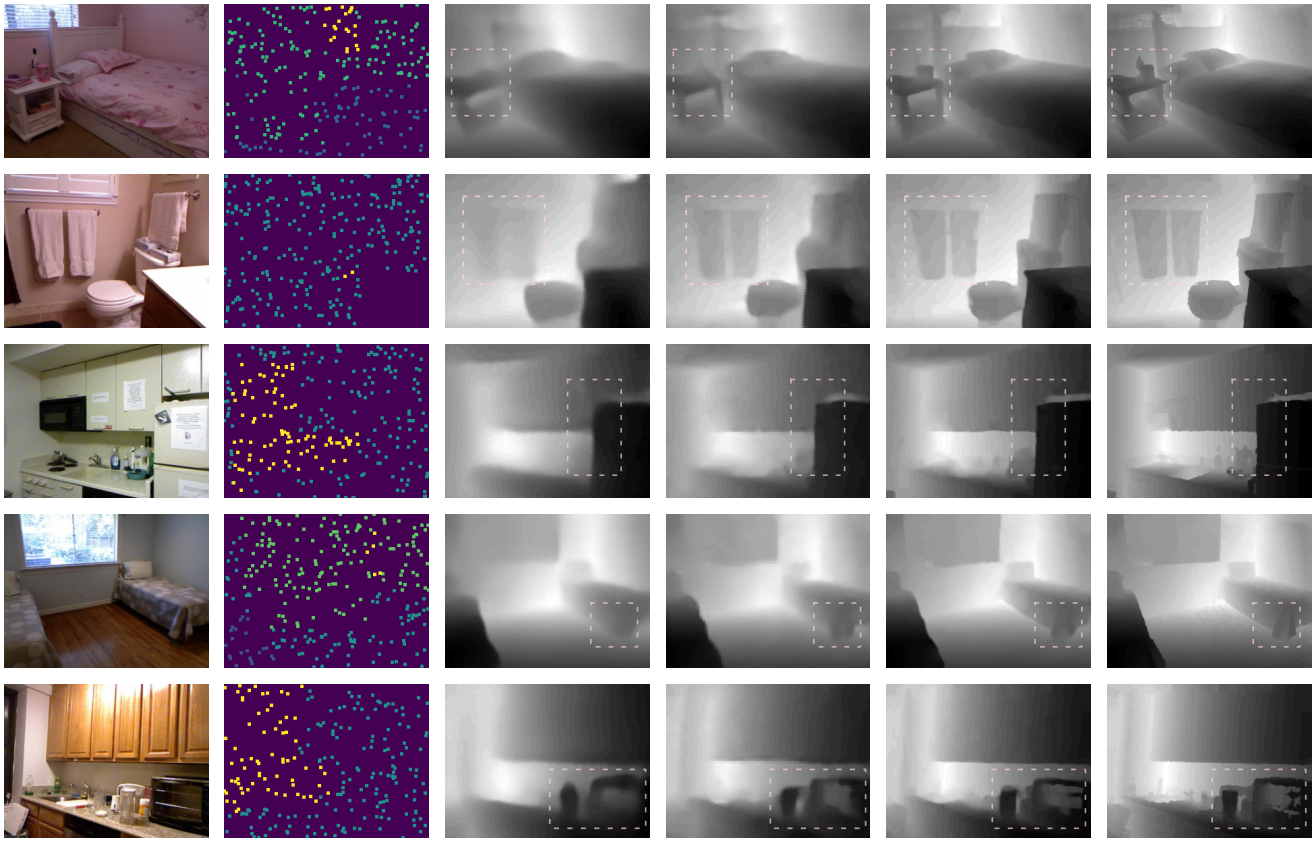


Fig. 10: Qualitative comparisons on NYU v2 dataset. (a) Input image; (b) Sparse depth samples(500); (c) Ma *et al.* [2]; (d) UNet(Ours)+SPN [52]; (e) UNet+CSPN(Ours); (f) Ground Truth. Most significantly improved regions are highlighted with yellow dash boxes (best view in color).

TABLE 2: Comparison results on KITTI dataset [6]

Method	Preserve "SD"	Lower the better		Higher the better					
		RMSE	REL	$\delta_{1.02}$	$\delta_{1.05}$	$\delta_{1.10}$	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$
Ma *et al.* [2]		3.378	0.073	30.0	65.8	85.2	93.5	97.6	98.9
+SPN [52]	✓	3.243	0.063	37.6	74.8	86.0	94.3	97.8	99.1
+CSPN(Ours)	✓	3.029	0.049	66.6	83.9	90.7	95.5	98.0	99.0
+UNet(Ours)		3.049	0.051	62.6	83.2	90.2	95.3	97.9	99.0
+UNet(Ours)+SPN	✓	3.248	0.059	52.1	79.0	87.9	94.4	97.7	98.9
+UNet+CSPN(Ours)	✓	**2.977**	**0.044**	**70.2**	**85.7**	**91.4**	**95.7**	**98.0**	**99.1**

At lower part of Tab. 1, we show the results using both color image and sparse depth samples, and all the results preserves the sparse depth value provided. We randomly select 500 depth samples per image from the ground truth depth map.

For comparison, we consider a baseline method using as-rigid-as-possible (ASAP) [85] warping. Basically the input depth map is warped with the sparse depth samples as control points. At row "ASAP", we show its results, which just marginally improves the estimation over the baseline network. For SPN, we also apply the similar replacement operation in Eq. (4) for propagation, and the results are shown at row "SPN", which outperforms both the results form ASAP and SPN without propagation of SD due to joint training helps fix the error of warping. At row "UNet + SPN", we use our UNet architecture for learning affinity with SPN, which outperforms "SPN", while we did not see any improvements compared with that only using UNet. Nevertheless, by replacing SPN with our CSPN, as shown in row "UNet + CSPN", the results can be further improved by a large margin and performs best in all cases. We think this is mostly because CSPN updates more efficiently than SPN during the training. Some visualizations are shown in Fig. 10. We found the results from CSPN do capture better structure from images (highlighted with dashed bounding boxes) than that from other state-of-the-art strategies.

KITTI. Tab. 2 shows the depth refinement with both color and sparse depth samples. Ours final model "UNet + CSPN" largely outperforms other SOTA strategies, which shows the generalization of the proposed approach. For instance, with a very strict metric $\delta < 1.02$, ours improves the baseline [2] from 30% to 70%, which is more than $2\times$ better. More importantly, CSPN is running very efficiently, thus can be applied to real applications. Some visualization results are shown at the bottom in Fig. 11. Compared to the network outputs from [2] and SPN refinement, CSPN sees

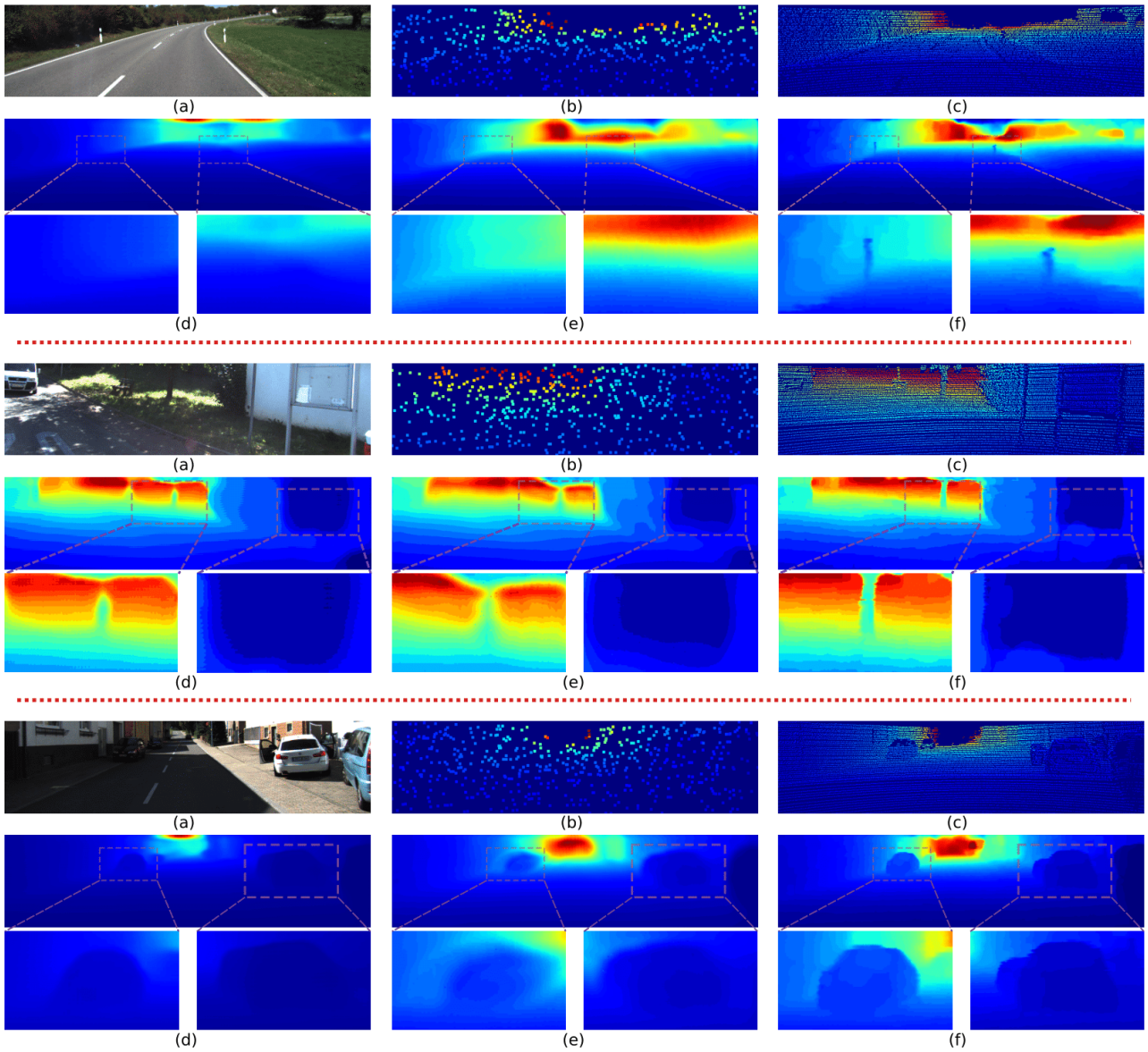


Fig. 11: Qualitative comparisons on KITTI dataset. (a) Input image; (b) Sparse depth samples(500); (c) Ground Truth; (d) Ma *et al.* [2]; (e) Ma [2]+SPN [52];(f) UNet+CSPN(Ours). Some details in the red bounding boxes are zoomed in for better visualization (best view in color).

much more details and thin structures such as poles near the road (first image (f)), and trunk on the grass (second image (f)). For the third image, we highlight a car under shadow at left, whose depth is difficult to learn. We can see SPN fails to refine such a case in (e) due to globally vast lighting variations, while CSPN learns local contrast and successfully recover the silhouette of the car. Finally, we also submit our results to the new KITTI depth completion challenge [3] and show that our results is better than previous SOTA method [3].

4.2 Stereo depth estimation

In this section, we evaluate the conjunct algorithms for stereo matching.

Implementation details. The base network we adopted is from the PSMNet [4], and we follow the same training strategies. Specifically, for learning CSPN, we adopt Adam [87] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and batch size is set to 16 for training on eight Nvidia P40 GPUs(each of 2). We performed color normalization on the entire dataset of Scene Flow for data preprocessing. During training, we crop the input image to 512×256. We first train our network from scratch on Scene Flow dataset for 10 epochs, and the learning rate during this periods is set to 0.001. When train on KITTI, we finetune the model obtained from Scene Flow for another 600 epochs. The learning rate starts from 0.001 and decrease 10% each 200 epochs. Acquired by accumulating Velodyne HDL-64 Laser scanner, KITTI stereo ground truth is relatively sparse, and we only compute the loss where LiDAR ground truth is available.

4.2.1 *Datasets and Metrics*

we evaluate our method on following datasets: Scene Flow [8], KITTI Stereo 2012 [6], KITTI Stereo 2015 [9]. **Scene Flow.** A large scale dataset contains 35454 training and 4370 test stereo pairs in 960x540 pixel resolution, rendered from various synthetic

3. http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_completion

TABLE 3: Ablation Studies for 3D Moudle on Scene Flow.

Method	Lower the Better		CSPN Parameters	
	EPE	RMSE	Propagation Times	Kernel Size
PSMNet [4]	1.119	5.763	0	0
+CSPN	0.992	5.142	24	3
+3DCSPN_ds	0.971	5.129	24	3
+3DCSPN_ss	1.007	4.731	1	3
+3DCSPN_ds_ss	**0.951**	**4.561**	24(ds)+1(ss)	3
WASPP + WSPF + 3DCSPN_ds_ss	**0.777**	**4.352**	1(WASPP)+1(SPPF)+24(ds)+1(ss)	3

TABLE 4: Ablation Studies for Context Pyramid Module on Scene Flow.

Method	Lower the Better		Addtional Setting			
	EPE	RMSE	3DCSPN_ds	2DCSPN	3DCSPN_Fusion	Dilation
SPP(PSMNet [4])	1.119	5.763				
SPP	0.971	5.129	✓			
WSPP	0.954	5.184	✓	✓		
ASPP	0.970	5.165	✓			✓
WASPP	0.902	4.954	✓	✓		✓
WSPF	0.905	5.036	✓		✓	
WASPP+WSPF	**0.827**	**4.555**	✓	✓	✓	✓

sequences. Pixels besides our max disparity are excluded in loss function.

KITTI Stereo 2012. A real-world dataset with street views from a driving car, consists of 194 training and 195 test stereo pairs in 1240x376 resolution. Ground truth has been aquired by accumulating 3D point clouds from a 360 degree Velodyne HDL-64 Laserscanner. We divided the whole training data into 160 training and 34 validate stereo pairs, we adopted color image as network input in this work.

KITTI Stereo 2015. Compared to KITTI 2012, KITTI 2015 consists of 200 training and 200 test stereo pairs in 1240x376 resolution. Also, it comprises dynamic scenes for which the ground truth has been established in a semi-automatic process. We further divided the whole training data into 160 training and 40 validate stereo pairs.

Metrics. Since different datasets have various metrics for comparison, we list the corresponding evaluation metric as follows,

Scene Flow: the end-point error (EPE) is used. Formally, the difference could be written as $EPE(d^*, \hat{d}) = \|d^* - \hat{d}\|_2$.

KITTI 2012 & 2015: the percentages of erroneous pixels. Specifically, a pixel is considered to be an erroneous pixel when its disparity error is larger than t pixels. Then, the percentages of erroneous pixels in non-occluded (Out-Noc) and all (Out-All) areas are calculated. Specifically, for benchmark 2012, $t \in \{2,3,4,5\}$. While for benchmark 2015, a pixel is considered to be wrong when the disparity error is larger than 3 pixels or relatively 5%, whichever is looser. In addition, results on both left image (D1-All) and right image (D2-All) are evaluated. We refer the reader to their original page [4] for more detailed information about other evaluated numbers. Here we only list the major metric for ranking different algorithms.

4.2.2 Ablation Study

We do various ablation studies based on the Scene Flow dataset to validate each component of our networks as shown in Fig. 7 and Fig. 8. We first train a baseline results with the code provided online by the author of PSMNet [5].

Study the 3D module. We first evaluate the components proposed in our 3D module (Fig. 7) in Tab. 3. In order to show that propagation in the new dimension benefits the results, we first adopt 2D CSPN for depth refinement as proposed for single image depth refinement over the three 2D disparity maps using the affinity predicted from the same feature, *i.e.,* "+CSPN". As expected, it reduces the EPE error from 1.119 to 0.992. Then, we switch the 2D CSPN to 3D CSPN as proposed in Sec. 3.3.1, *i.e.,* "+3DCSP_ds", the results are further improved to 0.971. Here, the footnote "ds" is short for disparity space, indicating the 3DCSPN is performed over the disparity outputs with shape $d \times h \times w \times 1$. "+3DCSPN_ss" shows the results by using 3D CSPN over the space for multi-stage outputs fusion, which also helps the performance from our baseline. Jointly using the two 3D CSPNs, *i.e.,* "+3DCSPN_ds_ss", yields the best result, outperforming our baseline method by a large margin. At last row, "WASPP+WSPF+3DCSPN_ss" shows the results of combining our 3D module with our enhanced SPP module together, which reduce the error around 30% w.r.t. to the baseline.

Study the SPP module. Here, we evaluate different components for enhancing the SPP module, as shown in Fig. 8. For all the variations, we adopt "3DCSPN_ds" as our 3D module for ablation study. As introduced in Sec. 3.3.1, "WSPP" means we use 2D CSPN over the spatial pooling grid, which reduces the EPE error from 0.971 to 0.954. We then study another spatial pooling strategy with dilated convolution, *i.e.,* "ASPP", which produces similar performance as SPP. Surprisingly, as shown in row "WASPP", jointly using our 2D CSPN with ASPP produces error much smaller (0.902) than that with SPP (0.954). At row "WSPF", we use similar fusion strategy to combine the pooled features from the spatial pyramid, which also significantly improves over the SPP baseline, reducing EPE error from 0.954 to 0.905. Finally, combining WASPP and WSPF, *i.e.,* "WASPP+WSPF", yields the best performance, which is selected as our final SPP module.

4. http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo

5. https://github.com/JiaRenChang/PSMNet

TABLE 5: Results on Scene Flow and KITTI Benchmark.

Method	Scene Flow	KITTI2012								KITTI 2015	
	EPE	2px		3px		4px		5px		All	Non-occluded
		Out-Noc	Out-All	Out-Noc	Out-All	Out-Noc	Out-All	Out-Noc	Out-All	D1-all	D1-all
MC-CNN [31]	3.79	3.90	5.45	2.43	3.63	1.90	2.85	1.64	2.39	3.88	3.33
SGM-Net [88]	4.50	3.60	5.15	2.29	3.50	1.83	2.80	1.60	2.36	3.66	3.09
DispNet v2 [72]	1.84	3.43	4.46	2.37	3.09	1.97	2.52	1.72	2.17	3.43	3.09
GC-Net [32]	2.51	2.71	3.46	1.77	2.30	1.36	1.77	1.12	1.46	2.67	2.45
iResNet-i2 [89]	1.40	2.69	3.34	1.71	2.16	1.30	1.63	1.06	1.32	2.44	2.19
PSM-Net [4]	1.09	2.44	3.01	1.49	1.89	1.12	1.42	0.90	1.15	2.32	2.14
EdgeStereo [90]	1.12	2.79	3.43	1.73	2.18	1.30	1.64	1.04	1.32	2.16	2.00
Ours(+3DCSPN_ds_ss)	**0.95**	**1.95**	**2.47**	**1.25**	**1.61**	**0.96**	**1.23**	**0.79**	**1.00**	**1.93**	**1.77**
Ours_Final	**0.78**	**1.79**	**2.27**	**1.19**	**1.53**	**0.93**	**1.19**	**0.77**	**0.98**	**1.74**	**1.61**

Fig. 12(c) shows a few examples of the output from the Scene Flow dataset, and we can see the predicted results are very close to ground truth, which are exceptionally good in handling detailed object structures.

4.2.3 Comparisons

To further validate the algorithm, in addition to comparing over the Scene Flow test set, we also submitted our results to KITTI 2012 and 2015 test evaluation server to compare against other SOTA methods proposed in recent years, including PSM-Net [4], iResNet-i2 [91], GC-Net [32], EdgeStereo [90], SGMNet [88], DispNet [72] and MC-CNN [31].

As summarized in Table 5, our method outperforms all others methods by a notable margin (above relatively 10%), and performs the best over all the major metrics both in KITTI 2012 [6] and 2015 [7]. By checking detailed numbers in KITTI 2015, we are better in improving the static background than foreground, which is reasonable because background has much larger amount of training pixels for learning the propagation affinity. Fig. 12 (a) and (b) show several example by comparing our algorithm to the baseline method PSMNet over KITTI 2012 and 2015 respectively, and we mark out the improved regions with dashed bounding boxes. As can be seen, not only better recovers the over all scene structure, CSPN is also superior in recovering detailed scene structures. More results are available in the KITTI leaderboard pages.

5 Conclusion

In this paper, we propose convolutional spatial propagation network (CSPN), which can be jointly learned with any type of CNN. It can be regarded as a linear diffusion process with guarantee to converge. Comparing with previous spatial propagation network [7] which learns the affinity, CSPN is not only more efficient (2-5× faster), but also more accurate (over 30% improvement) in terms of depth refinement. We also apply CSPN to depth completion by embedding sparse depth samples into the propagation process, and to stereo matching by adding another diffusion dimension over disparity space and feature scale space. For both tasks, CSPN provides superior improvement over other SOTA methods [2]. Since our framework is general, in the future, we plan to apply it to other tasks such as image segmentation and image enhancement.

6. http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo

7. http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo&eval_gt=all&eval_area=all

Acknowledgments

This work is supported by Baidu Inc.

References

[1] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 239–248. 1, 2, 3, 6, 9

[2] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," *ICRA*, 2018. 1, 2, 3, 5, 6, 9, 10, 11, 12, 14

[3] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," *3DV*, 2017. 1, 3, 12

[4] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5410–5418. 1, 2, 3, 12, 13, 14

[5] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," *ECCV*, 2012. 1, 2, 9, 10

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012. 1, 2, 9, 11, 12

[7] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz, "Learning affinity via spatial propagation networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 1519–1529. 1, 2, 3, 4, 6, 9, 10, 14

[8] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *CVPR*, 2016. 1, 2, 3, 12

[9] M. Menze, C. Heipke, and A. Geiger, "Object scene flow," *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018. 1, 2, 12

[10] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730. 1

[11] D. Murray and J. J. Little, "Using real-time stereo vision for mobile robot navigation," *autonomous robots*, vol. 8, no. 2, pp. 161–171, 2000. 1

[12] J. Biswas and M. Veloso, "Depth camera based localization and navigation for indoor mobile robots," in *RGB-D Workshop at RSS*, vol. 2011, 2011, p. 21. 1

[13] A. U. Haque and A. Nejadpak, "Obstacle avoidance using stereo camera," *arXiv preprint arXiv:1705.04114*, 2017. 1

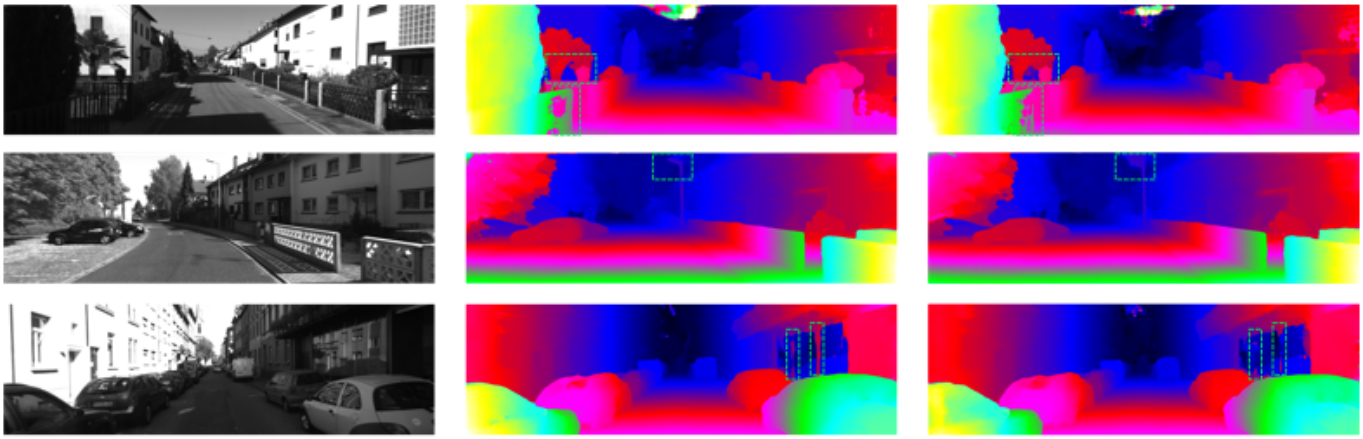
[14] B. Bascle and R. Deriche, "Stereo matching, reconstruction and refinement of 3d curves using deformable contours," in *Computer Vision, 1993. Proceedings., Fourth International Conference on*. IEEE, 1993, pp. 421–430. 1

[15] C. Zhang, Z. Li, Y. Cheng, R. Cai, H. Chao, and Y. Rui, "Meshstereo: A global stereo model with mesh alignment regularization for view interpolation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2057–2065. 1

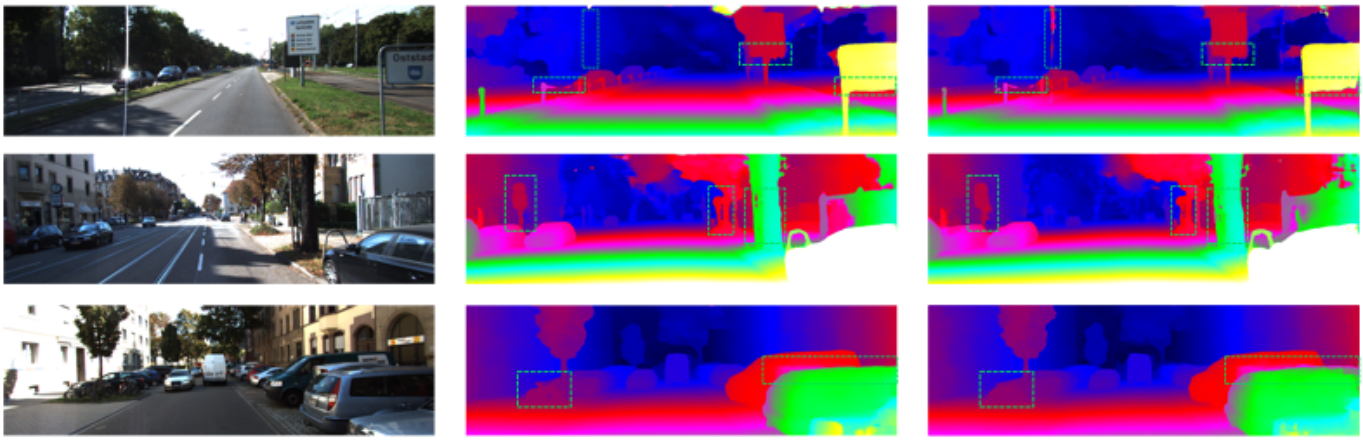
[16] G. Xu and Z. Zhang, *Epipolar geometry in stereo, motion and object recognition: a unified approach*. Springer Science & Business Media, 2013, vol. 6. 1

[17] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun, "3d graph neural networks for rgbd semantic segmentation," in *ICCV*, 2017. 1

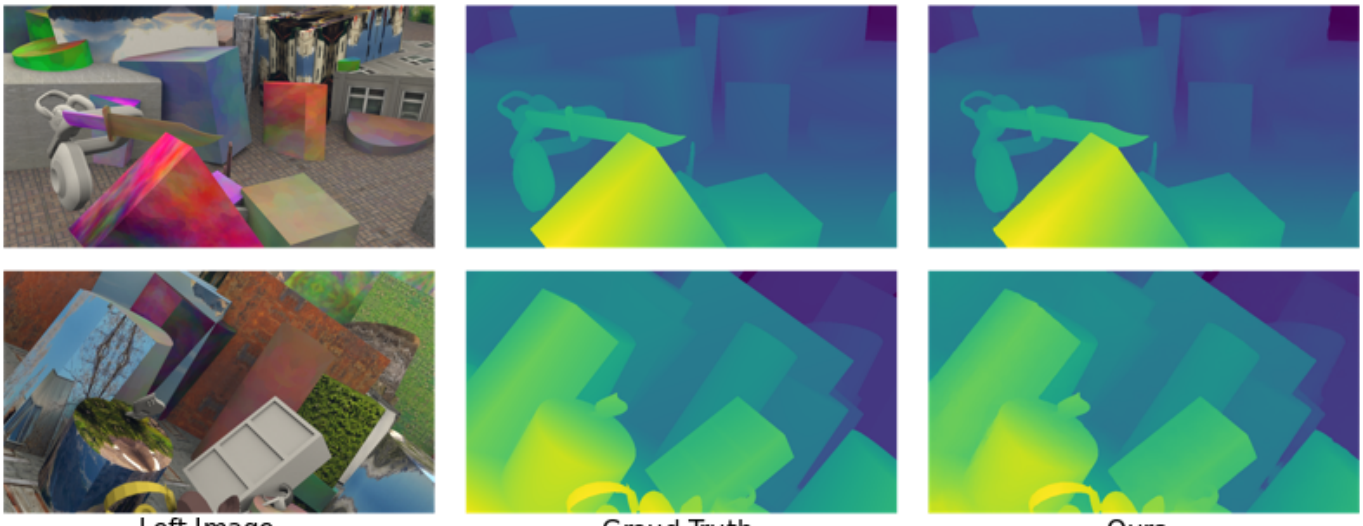
[18] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, 2015. 1, 2, 9



(a) Qualitative comparison on Kitti Stereo 2012, Significantly improved regions are highlight with green dash boxs (best view in color)



(b) Qualitative comparison on Kitti Stereo 2015, Significantly improved regions are highlight with green dash boxs (best view in color)



(c) Qualitative comparison on Scene Flow dataset

Fig. 12: Qualitative results. By learning affinity matrix in our model and propagate it to leverage context better, we can handle more challenging case

[19] J. Xiao, A. Owens, and A. Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels," in *ICCV*, 2013. 1

[20] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017. 1

[21] S. Wang, M. Bai, G. Mattyus, H. Chu, W. Luo, B. Yang, J. Liang, J. Cheverie, S. Fidler, and R. Urtasun, "Torontocity: Seeing the world with a million eyes," *ICCV*, 2017. 1

[22] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang, "The apolloscape dataset for autonomous driving," *arXiv preprint arXiv:1803.06184*, 2018. 1

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014. 1

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CVPR*, 2016. 1

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015. 1

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015, pp. 234–241. 1, 6

[27] Velodyne Lidar, "HDL-64E," http://velodynelidar.com/, 2018, [Online; accessed 01-March-2018]. 1

[28] RealSense, "D400," https://realsense.intel.com/stereo/. 1

[29] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1088–1095. 1

[30] Y. Liao, L. Huang, Y. Wang, S. Kodagoda, Y. Yu, and Y. Liu, "Parse geometry from a line: Monocular depth estimation with partial laser observation," *ICRA*, 2017. 1, 2, 3, 5

[31] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, no. 1-32, p. 2, 2016. 2, 3, 14

[32] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," *CoRR, vol. abs/1703.04309*, 2017. 2, 3, 9, 14

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*, 2014. 2, 3

[34] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 483–499. 2

[35] X. Wang, D. Fouhey, and A. Gupta, "Designing deep networks for surface normal estimation," in *CVPR*, 2015. 2

[36] J. Li, R. Klein, and A. Yao, "A two-streamed network for estimating fine-scaled depth maps from single rgb images," in *ICCV*, 2017. 2

[37] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," 2017. 2

[38] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," 2017. 2

[39] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, 2017. 2

[40] Z. Yang, P. Wang, W. Xu, L. Zhao, and N. Ram, "Unsupervised learning of geometry from videos with edge-aware depth-normal consistency," in *AAAI*, 2018. 2

[41] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia, "Lego: Learning edge with geometry all at once by watching videos," in *CVPR*, 2018, pp. 225–234. 2

[42] P. Wang, X. Shen, Z. Lin, S. Cohen, B. L. Price, and A. L. Yuille, "Towards unified depth and semantic prediction from a single image," in *CVPR*, 2015. 2

[43] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *CVPR*, June 2015. 2

[44] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs," in *CVPR*, 2015. 2

[45] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, "Conditional random fields as recurrent neural networks," in *ICCV*, 2015. 2

[46] P. Wang, X. Shen, B. Russell, S. Cohen, B. L. Price, and A. L. Yuille, "SURGE: surface regularized geometry estimation from a single image," in *NIPS*, 2016. 2, 9

[47] J. Shi and J. Malik, "Normalized cuts and image segmentation," *TPAMI*, vol. 22, no. 8, pp. 888–905, 2000. 2

[48] J. T. Barron and B. Poole, "The fast bilateral solver," in *ECCV*, 2016. 2, 10

[49] K. Matsuo and Y. Aoki, "Depth image enhancement using local tangent plane approximations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3574–3583. 2

[50] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof, "Image guided depth upsampling using anisotropic total generalized variation," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 993–1000. 2

[51] D. Ferstl, M. Ruther, and H. Bischof, "Variational depth superresolution using example-based edge representations," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 513–521. 2

[52] R. Liu, G. Zhong, J. Cao, Z. Lin, S. Shan, and Z. Luo, "Learning to diffuse: A new perspective to design pdes for visual analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 12, pp. 2457–2471, 2016. 2, 10, 11, 12

[53] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision*. Springer, 2014, pp. 184–199. 3

[54] J. Yang, X. Ye, K. Li, C. Hou, and Y. Wang, "Color-guided depth recovery from rgb-d data using an adaptive autoregressive model," *IEEE TIP*, vol. 23, no. 8, pp. 3443–3458, 2014. 3

[55] X. Song, Y. Dai, and X. Qin, "Deep depth super-resolution: Learning depth super-resolution using deep convolutional neural network," in *ACCV*. Springer, 2016, pp. 360–376. 3

[56] T.-W. Hui, C. C. Loy, and X. Tang, "Depth map super-resolution by deep multi-scale guidance," in *European Conference on Computer Vision*. Springer, 2016, pp. 353–369. 3

[57] H. Kwon, Y.-W. Tai, and S. Lin, "Data-driven depth map refinement via multi-scale sparse representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 159–167. 3

[58] G. Riegler, M. Rüther, and H. Bischof, "Atgv-net: Accurate depth super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 268–284. 3

[59] J. Weickert, *Anisotropic diffusion in image processing*. Teubner Stuttgart, 1998, vol. 1. 3

[60] M. Maire, T. Narihira, and S. X. Yu, "Affinity cnn: Learning pixel-centric pairwise relations for figure/ground embedding," in *CVPR*, 2016, pp. 174–182. 3

[61] G. Bertasius, L. Torresani, S. X. Yu, and J. Shi, "Convolutional random walk networks for semantic image segmentation," *arXiv preprint arXiv:1605.07681*, 2016. 3

[62] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform," in *CVPR*, 2016, pp. 4545–4554. 3

[63] K. Zimmermann, T. Petricek, V. Salansky, and T. Svoboda, "Learning for active 3d mapping," *ICCV*, 2017. 3

[64] L. Ladicky, O. Saurer, S. Jeong, F. Maninchedda, and M. Pollefeys, "From point clouds to mesh using regression," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 3

[65] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002. 3

[66] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2008. 3

[67] P. Heise, S. Klose, B. Jensen, and A. Knoll, "Pm-huber: Patchmatch with huber regularization for stereo matching," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2360–2367. 3

[68] L. Wang, H. Jin, R. Yang, and M. Gong, "Stereoscopic inpainting: Joint color and depth completion from stereo images," in *CVPR*. IEEE, 2008, pp. 1–8. 3

[69] Y. Feng, Z. Liang, and H. Liu, "Efficient deep learning for stereo matching with larger image patches," in *Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2017 10th International Congress on*. IEEE, 2017, pp. 1–5. 3

[70] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5695–5703. 3

[71] A. Shaked and L. Wolf, "Improved stereo matching with constant highway networks and reflective confidence learning," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4641–4650. 3

[72] F. Guney and A. Geiger, "Displets: Resolving stereo ambiguities using object knowledge," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4165–4175. 3, 14

[73] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *IEEE conference on computer vision and pattern recognition (CVPR)*, vol. 2, 2017, p. 6. 3

[74] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8934–8943. 3

[75] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *arXiv preprint arXiv:1506.04579*, 2015. 3

[76] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017. 3

[77] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *ICLR*, 2016. 3

[78] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890. 3, 6, 8

[79] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001. 4

[80] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," *NIPS*, 2012. 4

[81] I. Sobel, "History and definition of the sobel operator," *Retrieved from the World Wide Web*, 2014. 5

[82] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *CVPR*, June 2016. 7

[83] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008. 8

[84] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. 9

[85] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1134–1141, 2005. 10, 11

[86] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255. 9

[87] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 12

[88] A. Seki and M. Pollefeys, "Sgm-nets: Semi-global matching with neural networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA*, 2017, pp. 21–26. 14

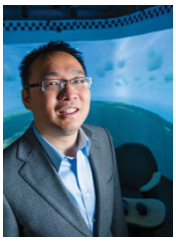
[89] Z. Liang, Y. Feng, Y. Guo, H. Liu, L. Qiao, W. Chen, L. Zhou, and J. Zhang, "Learning deep correspondence through prior and posterior feature constancy," *arXiv preprint arXiv:1712.01039*, 2017. 14

[90] X. Song, X. Zhao, H. Hu, and L. Fang, "Edgestereo: A context integrated residual pyramid network for stereo matching," *arXiv preprint arXiv:1803.05196*, 2018. 14

[91] Z. Liang, Y. Feng, Y. G. H. L. W. Chen, and L. Q. L. Z. J. Zhang, "Learning for disparity estimation through feature constancy," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2811–2820. 14

Xinjing Cheng is a research scientist with Robotics and Autonomous Driving Lab, Baidu Research, Baidu Inc., Beijing, China. Before that, he was a research assistant with the Intelligent Bionic Center, Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences(CAS), Shenzhen, China. His current research interests include computer vision, deep learning, robotics and autonomous driving.

Peng Wang is a senior research scientist in Baidu USA LLC. He obtained his Ph.D. degree in University of California, Los Angeles, advised by Prof. Alan Yuille. Before that, he received his B.S. and M.S. from Peking University, China. His research interest is image parsing and 3D understanding, and vision based autonomous driving system. He has around 20 published papers in ECCV/CVPR/ICCV/NIPS.



Ruigang, Yang is the chief scientist for 3D vision at Baidu. He is also a full professor at the University of Kentucky (on leave). His research interests include 3D computer vision and 3D computer graphics, in particular 3D modeling and 3D data analysis. He has published over 100 papers with an H-index of 48. He is an Associate Editor for IEEE T-PAMI. He has been a program co-chair for 3DIMPVT (now 3DV) 2011 and WACV 2014, and he has been area chairs for both ICCV and CVPR multiple times.