Depth Estimation via Affinity Learned with Convolutional Spatial Propagation Network

Xinjing Cheng(✉), Peng Wang, and Ruigang Yang

Baidu Research, Baidu Inc., Beijing, China
{chengxinjing,wangpeng54,yangruigang}@baidu.com

Abstract. Depth estimation from a single image is a fundamental problem in computer vision. In this paper, we propose a simple yet effective convolutional spatial propagation network (CSPN) to learn the affinity matrix for depth prediction. Specifically, we adopt an efficient linear propagation model, where 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 apply the designed CSPN to two depth estimation tasks given a single image: (1) Refine the depth output from existing state-of-the-art (SOTA) methods; (2) Convert sparse depth samples to a dense depth map by embedding the depth samples within the propagation procedure. The second task is inspired by the availability of LiDAR that provides sparse but accurate depth measurements. We experimented the proposed CSPN over the popular NYU v2 [1] and KITTI [2] datasets, where we show that our proposed approach improves not only quality (e.g., 30% more reduction in depth error), but also speed (e.g., 2 to 5$\times$ faster) of depth maps than previous SOTA methods. The codes of CSPN are available at: https://github.com/XinJCheng/CSPN.

Keywords: Depth estimation · Convolutional spatial propagation

1 Introduction

Depth estimation from a single image, *i.e.*, predicting per-pixel distance to the camera, has many applications from augmented realities (AR), autonomous driving, to robotics. Given a single image, recent efforts to estimate per-pixel depths have yielded high-quality outputs by taking advantage of deep fully convolutional neural networks [3,4] and large amount of training data from indoor [1,5,6] and outdoor [2,7,8]. The improvement lies mostly in more accurate estimation of global scene layout and scales with advanced networks, such as VGG [9] and ResNet [10], and better local structure recovery through deconvolution operation [11], skip-connections [12] or up-projection [4]. Nevertheless, upon closer

X. Cheng and P. Wang—Equal contribution.

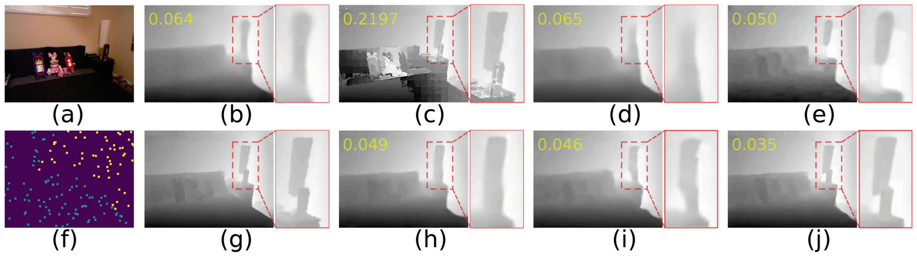

Fig. 1. (a) Input image; (b) Depth from [13]; (c) Depth after bilateral filtering; (d) Refined depth by SPN [14]; (e) Refined depth by CSPN; (f) Sparse depth samples (500); (g) Ground Truth; (h) Depth from our network; (i) Refined depth by SPN with depth sample; (j) Refined depth by CSPN with depth sample. The corresponding root mean square error (RMSE) is put at the left-top of each predicted depth map.

inspection of the output from a contemporary approach [13] (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.* [14] 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, we show that the proposed strategy can also be easily extended to convert sparse depth samples to a dense depth map given corresponding image [13,15]. 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 [13], 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 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. We experiment our approach over two popular benchmarks for depth estimation, *i.e.*NYU v2 [1] and KITTI [2], with standard evaluation criteria. 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 [13,15]. More importantly, it is very efficient yielding 2-5$\times$ acceleration comparing with SPN. 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 [14], 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.

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 [3,4,16,17], semi-supervised methods [18] or unsupervised methods [19–22]. and add in skip and mirror connections. Others tried to improve the estimated details further by appending a conditional random field (CRF) [23–25] and joint training [26,27]. However, the affinity for measuring the coherence of neighboring pixels is manually designed.

Depth Enhancement. Traditionally, depth output can be also efficiently enhanced with explicitly designed affinity through image filtering [28,29], or data-driven ones through total variation (TV) [30,31] and learning to diffuse [32] 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 [33,34] and depths [35–38]. 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 [39]. Maire *et al.* [40] 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.* [41] 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.* [42] 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 [14], 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 Sect. 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 [15]. 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.* [13] propose to treat sparse depth map as additional input to a ResNet [4] 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 Sect. 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 [43–45], whereas the density of sparse points must be high enough to reveal the scene structure, which is not available in our scenario.

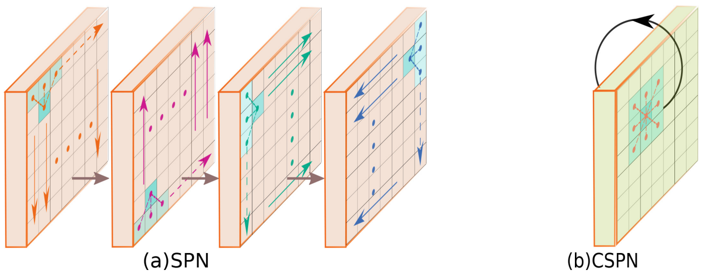

Fig. 2. Comparison between the propagation process in SPN [14] and CPSN in this work.

3 Our Approach

We formulate the problem as an anisotropic diffusion process and the diffusion tensor is learned through a deep CNN directly from the given image, which guides the refinement of the output.

3.1 Convolutional Spatial Propagation Network

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

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

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

where the transformation kernel $\hat{\kappa}_{i,j} \in \mathbb{R}^{k\times k\times c}$ 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 [14], 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 [14], 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 $\mathbf{H}_v \in \mathbb{R}^{mn \times c}$.

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

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 [14] 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 [46]. 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.

3.2 Spatial Propagation with Sparse Depth Samples

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.

Similarly, we also embed the sparse depth map $D_s = \{d_{i,j}^s\}$ 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),

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

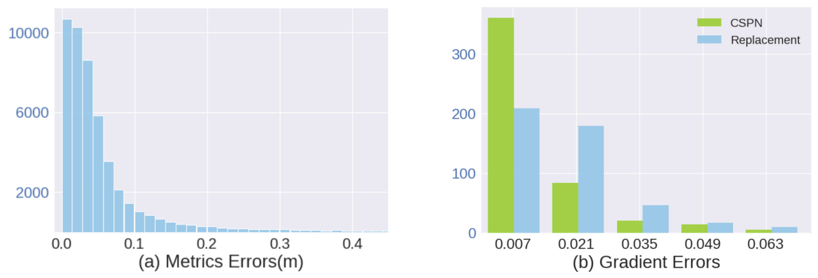

Fig. 3. (a) Histogram of RMSE with depth maps from [13] at given sparse depth points. (b) Comparison of gradient error between depth maps with sparse depth replacement (blue bars) and with ours CSPN (green bars), where ours is much smaller. Check Fig. 4 for an example. Vertical axis shows the count of pixels. (color figure online)

where $m_{i,j} = \mathbb{I}(d^s_{i,j} > 0)$ 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 (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 [13, 15]. 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.* [13]. It shows the accuracy of sparse depth points cannot preserved, and some pixels could have very large displacement (0.2 m), 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 gradient w.r.t.surrounding pixels. In Fig. 4(c), we plot such an example, at right of the figure, we compute Sobel gradient [47] 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

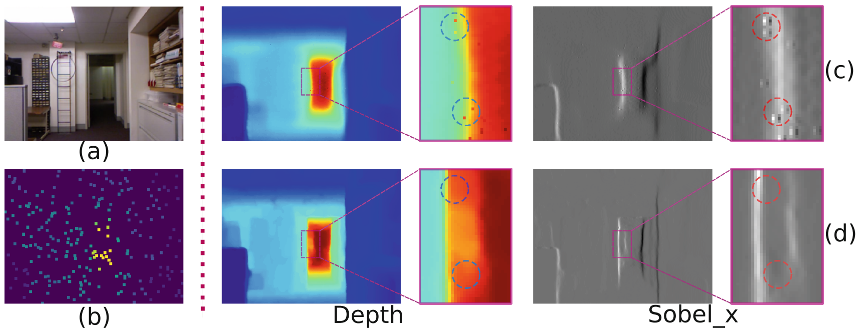

Fig. 4. Comparison of depth map [13] with sparse depth replacement and with our CSPN w.r.t.smoothness of depth gradient at sparse depth points. (a) Input image. (b) Sparse depth points. (c) Depth map with sparse depth replacement. (d) Depth map with our CSPN with sparse depth points. We highlight the differences in the red box (color figure online).

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. In Fig. 4(d), we show the depth gradients surrounding sparse pixels are smooth and close to ground truth, demonstrating the effectiveness of our propagation scheme.

3.3 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 addition, 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 [14] 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 [48], 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 (Sect. 4), with $k = 3$ and $N = 12$, CSPN already outperforms SPN with a large margin (relative 30%), demonstrating both efficiency and effectiveness of the proposed approach.

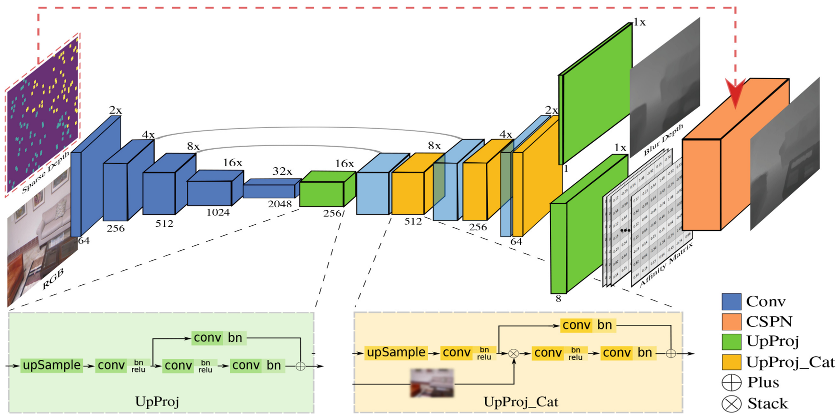

Fig. 5. Architecture of our networks with mirror connections for depth estimation via transformation kernel prediction with CSPN (best view in color). Sparse depth is an optional input, which can be embedded into the CSPN to guide the depth refinement.

3.4 End-to-End Architecture

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.* [13], 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.* [14], 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 [4]. Thus, we add mirror connections similar with the U-shape network [12] 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 (Sect. 4.2). 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 (Sect. 4). Finally we adopt the same training loss as [13], yielding an end-to-end learning system.

4 Experiments

In this section, we describe our implementation details, the datasets and evaluation metrics used in our experiments. Then present comprehensive evaluation of CSPN on both depth refinement and sparse to dense tasks.

Implementation details. The weights of ResNet in the encoding layers for depth estimation (Sect. 3.4) are initialized with models pretrained on the ImageNet dataset [49]. 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 [1] 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 $\mathbf{H}$ 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 [13].

4.1 Datasets and Metrics

All our experiments are evaluated on two datasets: NYU v2 [1] and KITTI [2], 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 [13]. For testing, following the standard setting [3,27], 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 [13], 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 [13]. 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,

[1] http://pytorch.org/.

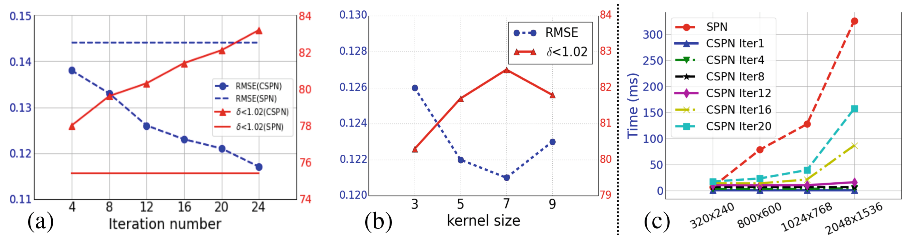

Fig. 6. Ablation study.(a) RMSE (left axis, lower the better) and $\delta < 1.02$ (right axis, higher the better) of CSPN w.r.t.number of iterations. Horizontal lines show the corresponding results from SPN [14]. (b) RMSE and $\delta < 1.02$ of CSPN w.r.t.kernel size. (c) Testing times w.r.t.input image size.

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 (Table 1). Thus we adopt more strict criteria for correctness by choosing $t \in \{1.02, 1.05, 1.10\}$.

4.2 Parameter Tuning and Speed Study

We first evaluate various hyper-parameters including kernel size k, number of iterations N in Eq. (1) using the NYU v2 dataset. 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. 6(a), our CSPN has outperformed SPN [14] (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. 6(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.

Table 1. Comparison results on NYU v2 dataset [1] 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.*[13]		0.230	0.044	52.3	82.3	92.6	97.1	99.4	99.8
+Bilateral [28]		0.479	0.084	29.9	58.0	77.3	92.4	97.6	98.9
+SPN [32]		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 [50]	✓	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 [32]	✓	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

Concatenation end-point for mirror connection. As discussed in Sect. 3.4, 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. 6(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 Sect. 3.3. 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.

4.3 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 [32] and Sparse-to-Dense [13], we use the released code released online from the authors.

NYU v2. Table 1 shows the comparison results. Our baseline methods are the depth output from the network of [13], together with the corresponding color image. At upper part of Table 1 we show the results for depth refinement with color only. At row "Bilateral", we refine the network output from [13] using bilateral filtering [28] 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 [14], 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 Sect. 3.4. 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.

At lower part of Table 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) [50] 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. 7. 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. Table 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 [13] 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. 8. Compared to the network outputs from [13] and SPN refinement, CSPN sees much more details and thin

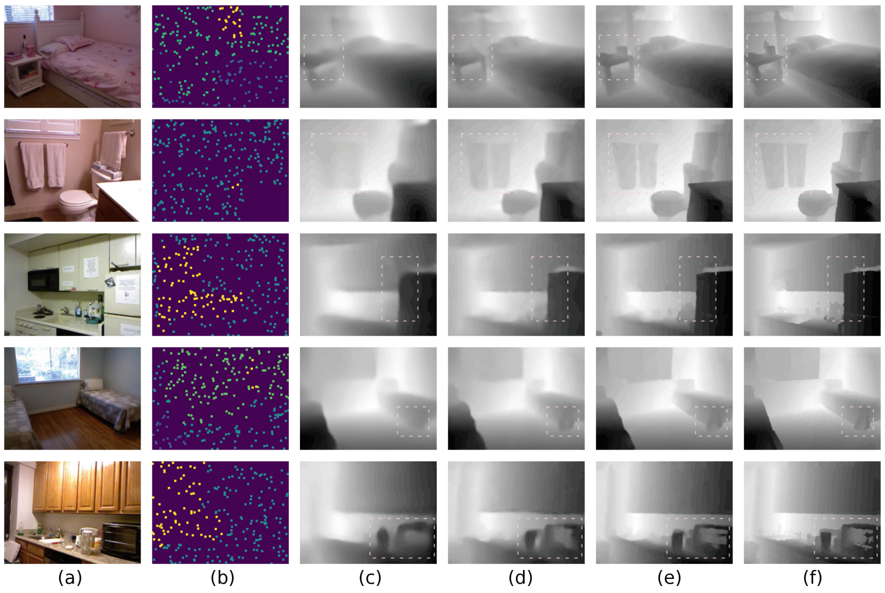

Fig. 7. Qualitative comparisons on NYU v2 dataset. (a) Input image; (b) Sparse depth samples(500); (c) Ma *et al.*[13]; (d) UNet(Ours)+SPN[32]; (e) UNet+CSPN(Ours); (f) Ground Truth. Most significantly improved regions are highlighted with yellow dash boxes (best view in color). (color figure online)

Table 2. Comparison results on KITTI dataset [2]

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.* [13]		3.378	0.073	30.0	65.8	85.2	93.5	97.6	98.9
+SPN [32]	✓	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**

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 [2] and show that our results is better than previous SOTA method [45].

[2] http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_completion.

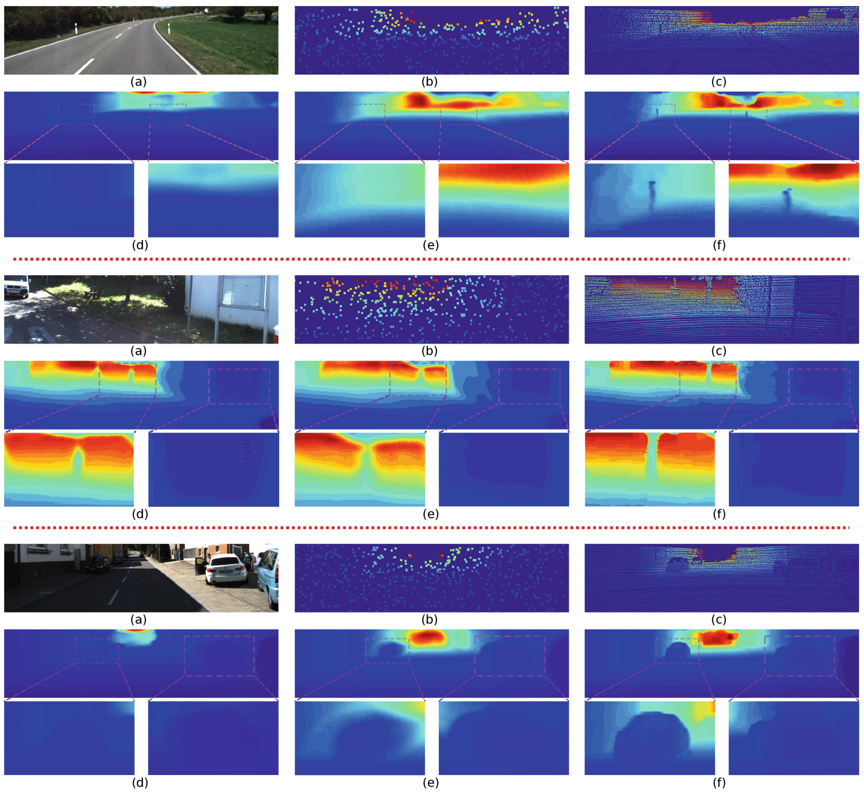

Fig. 8. Qualitative comparisons on KITTI dataset. (a) Input image; (b) Sparse depth samples(500); (c) Ground Truth; (d) Ma *et al.*[13]; (e) Ma[13]+SPN [32];(f) UNet+CSPN(Ours). Some details in the red bounding boxes are zoomed in for better visualization (best view in color). (color figure online)

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 [14] 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 extend CSPN by embedding sparse depth samples into the propagation process, which provides superior improvement over other SOTA methods [13]. Since our framework is general, in the future, we plan to apply it to other tasks such as image segmentation and enhancement.

References

1. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 746–760. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33715-4_54
2. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012)
3. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV (2015)
4. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 2016 Fourth International Conference on 3D Vision (3DV), pp. 239–248. IEEE (2016)
5. Xiao, J., Owens, A., Torralba, A.: Sun3d: a database of big spaces reconstructed using sfm and object labels. In: ICCV (2013)
6. Chang, A., et al.: Matterport3D: Learning from RGB-D data in indoor environments. In: International Conference on 3D Vision (3DV) (2017)
7. Wang, S., et al.: Torontocity: Seeing the world with a million eyes. In: ICCV (2017)
8. Huang, X., et al.: The apolloscape dataset for autonomous driving. arXiv preprint arXiv:1803.06184 (2018)
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
11. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
12. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234–241 (2015)
13. Ma, F., Karaman, S.: Sparse-to-dense: depth prediction from sparse depth samples and a single image. In: ICRA (2018)
14. Liu, S., De Mello, S., Gu, J., Zhong, G., Yang, M.H., Kautz, J.: Learning affinity via spatial propagation networks. In: Advances in Neural Information Processing Systems, pp. 1519–1529 (2017)
15. Liao, Y., Huang, L., Wang, Y., Kodagoda, S., Yu, Y., Liu, Y.: Parse geometry from a line: monocular depth estimation with partial laser observation. In: ICRA (2017)
16. Wang, X., Fouhey, D., Gupta, A.: Designing deep networks for surface normal estimation. In: CVPR (2015)
17. Li, J., Klein, R., Yao, A.: A two-streamed network for estimating fine-scaled depth maps from single rgb images. In: ICCV (2017)
18. Kuznietsov, Y., Stuckler, J., Leibe, B.: Semi-supervised deep learning for monocular depth map prediction (2017)
19. Godard, C., Mac Aodha, O., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency (2017)
20. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: CVPR (2017)
21. Yang, Z., Wang, P., Xu, W., Zhao, L., Ram, N.: Unsupervised learning of geometry from videos with edge-aware depth-normal consistency. In: AAAI (2018)

22. Yang, Z., Wang, P., Wang, Y., Xu, W., Nevatia, R.: Lego: Learning edge with geometry all at once by watching videos. In: CVPR, pp. 225–234 (2018)
23. Wang, P., Shen, X., Lin, Z., Cohen, S., Price, B.L., Yuille, A.L.: Towards unified depth and semantic prediction from a single image. In: CVPR (2015)
24. Liu, F., Shen, C., Lin, G.: Deep convolutional neural fields for depth estimation from a single image. In: CVPR (June 2015)
25. Li, B., Shen, C., Dai, Y., van den Hengel, A., He, M.: Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In: CVPR (2015)
26. Zheng, S., et al.: Conditional random fields as recurrent neural networks. In: ICCV (2015)
27. Wang, P., Shen, X., Russell, B., Cohen, S., Price, B.L., Yuille, A.L.: SURGE: surface regularized geometry estimation from a single image. In: NIPS (2016)
28. Barron, J.T., Poole, B.: The fast bilateral solver. In: ECCV (2016)
29. Matsuo, K., Aoki, Y.: Depth image enhancement using local tangent plane approximations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3574–3583 (2015)
30. Ferstl, D., Reinbacher, C., Ranftl, R., Rüther, M., Bischof, H.: Image guided depth upsampling using anisotropic total generalized variation. In: 2013 IEEE International Conference on Computer Vision (ICCV), pp. 993–1000. IEEE (2013)
31. Ferstl, D., Ruther, M., Bischof, H.: Variational depth superresolution using example-based edge representations. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 513–521 (2015)
32. Liu, R., Zhong, G., Cao, J., Lin, Z., Shan, S., Luo, Z.: Learning to diffuse: a new perspective to design pdes for visual analysis. IEEE Trans. Pattern Anal. Mach. Intell. **38**(12), 2457–2471 (2016)
33. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8692, pp. 184–199. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10593-2_13
34. Yang, J., Ye, X., Li, K., Hou, C., Wang, Y.: Color-guided depth recovery from rgb-d data using an adaptive autoregressive model. IEEE TIP **23**(8), 3443–3458 (2014)
35. Song, X., Dai, Y., Qin, X.: Deep depth super-resolution: learning depth super-resolution using deep convolutional neural network. In: Lai, S.-H., Lepetit, V., Nishino, K., Sato, Y. (eds.) ACCV 2016. LNCS, vol. 10114, pp. 360–376. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54190-7_22
36. Hui, T.-W., Loy, C.C., Tang, X.: Depth map super-resolution by deep multi-scale guidance. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 353–369. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_22
37. Kwon, H., Tai, Y.W., Lin, S.: Data-driven depth map refinement via multi-scale sparse representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 159–167. (2015)
38. Riegler, G., Rüther, M., Bischof, H.: ATGV-Net: accurate depth super-resolution. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 268–284. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_17
39. Weickert, J.: Anisotropic diffusion in image processing. Volume 1. Teubner Stuttgart (1998)

40. Maire, M., Narihira, T., Yu, S.X.: Affinity cnn: Learning pixel-centric pairwise relations for figure/ground embedding. In: CVPR, pp. 174–182 (2016)
41. Bertasius, G., Torresani, L., Yu, S.X., Shi, J.: Convolutional random walk networks for semantic image segmentation. arXiv preprint arXiv:1605.07681 (2016)
42. Chen, L.C., Barron, J.T., Papandreou, G., Murphy, K., Yuille, A.L.: Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In: CVPR, pp. 4545–4554 (2016)
43. Zimmermann, K., Petricek, T., Salansky, V., Svoboda, T.: Learning for active 3d mapping. In: ICCV (2017)
44. Ladicky, L., Saurer, O., Jeong, S., Maninchedda, F., Pollefeys, M.: From point clouds to mesh using regression. In: The IEEE International Conference on Computer Vision (ICCV) (2017)
45. Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., Geiger, A.: Sparsity invariant cnns. 3DV (2017)
46. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Trans. Inf. theory **47**(2), 498–519 (2001)
47. Sobel, I.: History and definition of the sobel operator. Retrieved from the World Wide Web (2014)
48. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. In: NIPS (2012)
49. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, 2009, CVPR 2009, pp. 248–255. IEEE (2009)
50. Igarashi, T., Moscovich, T., Hughes, J.F.: As-rigid-as-possible shape manipulation. ACM Trans. Gr. (TOG) **24**(3), 1134–1141 (2005)