

Confidence Propagation through CNNs for Guided Sparse Depth Regression

Abdelrahman Eldesokey, *Student Member, IEEE*, Michael Felsberg, *Senior Member, IEEE*,
and Fahad Shahbaz Khan, *Member, IEEE*

Abstract—Generally, convolutional neural networks (CNNs) process data on a regular grid, *e.g.* data generated by ordinary cameras. Designing CNNs for sparse and irregularly spaced input data is still an open research problem with numerous applications in autonomous driving, robotics, and surveillance. In this paper, we propose an algebraically-constrained normalized convolution layer for CNNs with highly sparse input that has a smaller number of network parameters compared to related work. We propose novel strategies for determining the confidence from the convolution operation and propagating it to consecutive layers. We also propose an objective function that simultaneously minimizes the data error while maximizing the output confidence. To integrate structural information, we also investigate fusion strategies to combine depth and RGB information in our normalized convolution network framework. In addition, we introduce the use of output confidence as an auxiliary information to improve the results. The capabilities of our normalized convolution network framework are demonstrated for the problem of scene depth completion. Comprehensive experiments are performed on the KITTI-Depth benchmark and the results clearly demonstrate that the proposed approach achieves superior performance while requiring only about 5% of the number of parameters compared to the state-of-the-art methods.

Index Terms—Sparse data, CNNs, Depth completion, Normalized convolution, Confidence propagation

1 INTRODUCTION

SENSORS with dense output such as monochrome, color, and thermal cameras have been extensively exploited by machine learning methods in many computer vision applications. Images generated by these sensors are typically fully dense due to their passive nature and different image regions are initially equally relevant to the machine learning algorithms. However, other, mostly active, sensors such as ToF cameras, LiDAR, RGB-D, and event cameras produce sparse output. This sparsity is usually caused by their active sensing, which leaves many data regions empty. The sparse output from these sensors imposes fundamental challenges on the machine learning methods as data relevance is not uniform and further processing is required to either reconstruct or ignore these missing regions.

The degree of sparsity and data pattern differ from one sensor to another and machine learning methods should be able to handle different scenarios. Handling sparsity would open up for numerous applications in robotics, autonomous driving, and surveillance due to the depth information made available by active sensors. Therefore, a major task is *scene depth completion*, which aims to reconstruct a dense depth map from the sparse output produced by active depth sensors. Scene depth completion is crucial for tasks that require situation awareness for decision support. Besides, the availability of a reliability measure, *i.e.* confidence, is also desirable since it gives an indication about the trustworthiness of the output. A confidence measure would be highly beneficial for safety and decision making applications such

as obstacle detection and avoidance in autonomous driving.

A key problem in scene depth completion is the identification of missing values and distinguishing them from regions with zero values. One way to identify missing data is using binary validity masks with zeros at regions with missing values and ones otherwise. Validity masks have been extensively used in the literature [1], [2], [3], [4], [5], [6] to inform the learning method about the missing regions in the data. However, validity masks suffer from saturation in multi-stage learning such as Convolutional Neural Networks (CNNs) as shown by [7]. Instead, the saturation problem can be avoided by treating the binary validity masks as continuous confidence fields describing the reliability of the data. Additionally, this enables confidence propagation, which helps to keep track of the reliability of the data throughout the processing pipeline.

Most recent works for solving the scene depth completion task are based on CNNs, and show a great success [1], [3], [5], [6], [7], [8]. Typically, a deep CNN is trained to construct a dense depth map given either a sparse depth input only or sparse depth aside with an RGB image. The former case is denoted as *unguided* depth completion, while the latter is called *guided* depth completion. The role of the network in both cases is to learn the manifold where the data live in. Since the publicly available datasets for scene depth completion such as the KITTI-Depth dataset [1] have a very high spatial resolution, the state-of-the-art methods [6], [7], [8] demand huge CNNs with millions of parameters to solve the problem. Unfortunately, this hinders the deployment of such methods in autonomous driving and robotic systems with limited computational and memory resources. It is desirable to design compact CNN architectures, while propagating confidences, for real-world applications with limited resources.

- All authors are with the Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.
E-mail: abdelrahman.eldeskey@liu.se
- Fahad Khan is also with the Inception Institute of Artificial Intelligence Abu Dhabi, UAE

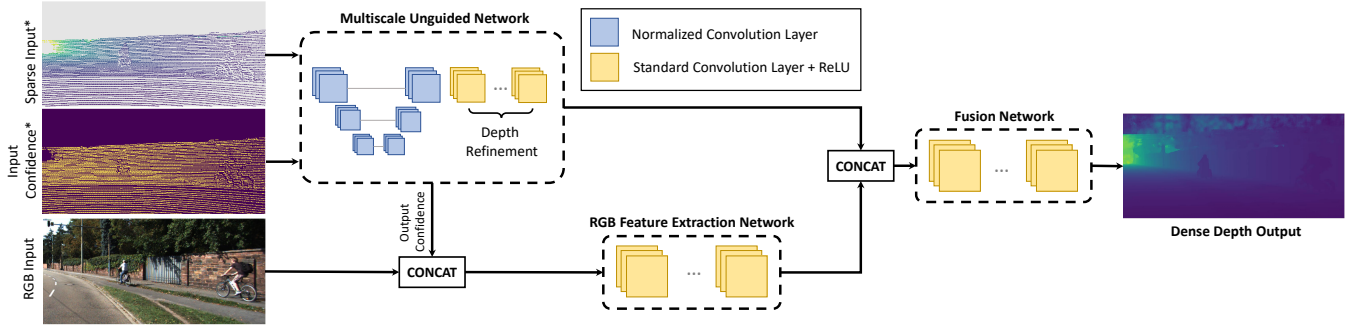


Fig. 1. Our scene depth completion pipeline on an example image from the KITTI-Depth dataset [1]. The input to the pipeline is a very sparse projected LiDAR point cloud, an input confidence map which has zeros at missing pixels and ones otherwise, and an RGB image. The sparse point cloud input and the input confidence are fed to a multi-scale unguided network that acts as a generic estimator for the data. Afterwards, the continuous output confidence map is concatenated with the RGB image and fed to a feature extraction network. The output from the unguided network and the RGB feature extraction networks are concatenated and fed to a fusion network which produces the final dense depth map. [*Images were dilated for visual clarity]

In this paper, we introduce the normalized convolution layer, which allows performing unguided scene depth completion on highly sparse data with a smaller number of parameters than related methods. Our proposed method treats the validity masks as a continuous confidence field and we propose a new criteria to propagate confidences between CNN layers, thereby allowing us to produce a point-wise continuous confidence map for the output from the network. Furthermore, we algebraically constrain the network learned filters to be non-negative, acting as a weighting function for the neighborhood. This allows the network to converge faster and achieves remarkably better results. We also propose a loss function that aims to simultaneously minimize the data error and maximize the output confidence.

The proposed normalized convolution network generally performs well on smooth surfaces when using only the depth information. However, it performs less well across edges due to lack of structural information. This can be mitigated by using guidance from RGB images in order to integrate useful structural information into our network. Both RGB and depth information can be fused in our proposed framework in multiple ways. In this work, we investigate both early and late fusion schemes in two state-of-the-art architectures. The first is a multi-stream architecture inspired by [5] which is highly relevant to our proposed method and the second is an encoder-decoder architecture with skip-connections inspired by [4], [9]. In addition, we introduce the use of output confidences as guidance information aside with the RGB images. On the KITTI-Depth benchmark, our proposed method achieves state-of-the-art results while requiring a significantly lower number of parameters ($\sim 355k$ parameter) compared to all the existing state-of-the-art methods (with millions of parameters), which proves the efficiency of our method. An illustration of the proposed pipeline is shown in Figure 1.

The rest of the paper is organized as follows. Section 2 gives an overview of the relevant work in the literature. In section 3, we describe the normalized convolution framework in details. Section 4 describes our early work on unguided depth completion in [10]. Section 5 introduces our proposed approach to fuse sparse depth, RGB images,

and the output confidences. Finally, we perform extensive experiments on both our prior work [10] and the proposed fusion schemes in section 6. The conclusion is given in section 7.

2 RELATED WORK

Scene depth completion has become a fundamental task in computer vision ever since the emergence of active sensors with depth capabilities. Generally, it was treated as a hole-filling or inpainting problem using classical image processing methods [11], [12], [13]. Recently, with the advent of deep learning, specifically Convolutional Neural Networks (CNNs), scene depth completion has matured into a separate task than inpainting. Typically, scene depth completion is performed on depth maps with optional guidance from RGB images. Differently, inpainting is mostly performed on RGB or grayscale images. In addition, the objective of scene depth completion is to minimize some error measure such as the $L1$ or the $L2$ norm, while inpainting aims also to provide realistic output [4].

For the task of unguided scene depth completion, where the input is only the depth map, Chodosh *et al.* [3] utilized compressed sensing to handle the sparsity, while using a binary mask to filter out the missing values. Ma *et al.* [8] utilized an encoder-decoder architecture with self-supervised framework to predict the dense output. Uhrig *et al.* [1] proposed a sparsity-invariant convolution layer that utilizes binary validity masks to normalize the sparse input. The proposed layer was used to train a network with a sparse depth map aside with a binary validity mask as input and a dense depth map as output. Similarly, [6] utilized the sparsity-invariant layer in more complex CNN architectures. Hua and Gong [5] proposed a similar layer, which uses the trained convolution filter to normalize the sparse input. Contrarily, Jaritz *et al.* [7] compared different architectures and argued that the use of validity masks degrades the performance due to the saturation of the masks at early layers within the CNN. This effect is avoided by the use of continuous confidences as proposed in our prior work on unguided depth completion [10].

Due to the sub-optimal performance of unguided methods across edges, several recent approaches urged to use

guidance from auxiliary data such as RGB images or surface normals. Ma *et al.* [8] used an early fusion scheme to combine sparse depth input with the corresponding RGB image, which was demonstrated to perform very well. On the other hand, Jaritz *et al.* [7] argued that late-fusion performs better with their proposed architecture, which was also demonstrated in [5]. Wirges *et al.* [14] used a combination of RGB images and surface normals to guide the process of depth upsampling. Konno *et al.* [15] utilized a residual interpolation method to combine a low-resolution depth map with a high resolution RGB image to produce a high resolution depth map.

Different to the aforementioned approaches, we proposed a normalized convolution layer in [10], which takes in a sparse input aside with a continuous confidence map to perform unguided scene depth completion. Different to [5], we impose algebraic constraints on the trained filters to be non-negative, which allow the network to converge faster while requiring significantly lower number of parameters. Further, we derive a confidence propagation criteria between layers, which enables producing a point-wise continuous confidence map aside with the dense output from the CNN. As an extension to our prior work [10], we use guidance from RGB images, and Different to [6], [7], [8], we also use guidance from the output confidence produced by the unguided network. Our results clearly show that using guidance from the output confidence leads to a significant improvement in performance. Our proposed multi-stream architecture with late fusion achieves remarkable results compared to published state-of-the-art methods while requiring significantly lower number of parameters than comparable methods. This demonstrates the efficiency of our proposed method, which eliminated the need for a huge number of parameters to achieve state-of-the-art results.

3 NORMALIZED CONVOLUTION

The concept of Normalized Convolution was first introduced by Knutsson and Westin [16] based on the theory of confidence-equipped signals. Assume a discrete finite signal f that is periodically sampled. At each sample point, the neighborhood is finite and can be represented as a vector $\mathbf{f} \in \mathbb{C}^n$. This signal is accompanied by a confidence field c , which describes the reliability of each sample value. Confidences are typically non-negative and a zero confidence indicates the absence of the corresponding sample value. The confidence field c is sampled in the same manner as f and the confidence of each neighborhood is represented as a finite vector $\mathbf{c} \in \mathbb{C}^n$.

In [16], the signal is modeled locally by projecting each sample \mathbf{f} onto a subspace spanned by some basis functions, e.g. polynomials, complex exponentials, or the naïve basis. The set of basis functions $\{\mathbf{b}_i \in \mathbb{C}^n\}_1^m$ have the same dimensionality as the sample \mathbf{f} and its corresponding confidence \mathbf{c} . The sample \mathbf{f} is expressed with respect to the basis functions arranged into the columns of a $n \times m$ matrix \mathbf{B} as:

$$\mathbf{f} = \mathbf{B}\mathbf{r} , \quad (1)$$

where \mathbf{r} holds the coordinates of the sample \mathbf{f} with respect to the basis functions.

Finding these coordinates is usually formulated as a weighted least-square problem:

$$\arg \min_{\mathbf{r} \in \mathbb{C}^n} \|\mathbf{B}\mathbf{r} - \mathbf{f}\|_{\mathbf{W}} , \quad (2)$$

where the sample \mathbf{f} is weighted by the confidence \mathbf{c} and the basis functions are weighted by applicability functions $\mathbf{a} \in \mathbb{C}^n$, that act as confidence measures for the basis. The solution $\hat{\mathbf{r}}$ to this weighted least-squares problem reads:

$$\begin{aligned} \hat{\mathbf{r}} &= (\mathbf{B}^* \mathbf{W} \mathbf{B})^{-1} \mathbf{B}^* \mathbf{W} \mathbf{f} , \\ &= (\mathbf{B}^* \mathbf{D}_a \mathbf{D}_c \mathbf{B})^{-1} \mathbf{B}^* \mathbf{D}_a \mathbf{D}_c \mathbf{f} , \end{aligned} \quad (3)$$

where \mathbf{D}_a and \mathbf{D}_c are diagonal matrices with \mathbf{a} and \mathbf{c} on the main diagonal, respectively, and they act as weights for the least-squares problem. This formulation can be applied to the whole signal f and its corresponding confidence c in a convolution-like structure.

3.1 The Applicability Function

The applicability function \mathbf{a} is required to be non-negative and it acts as a windowing function for the basis, e.g. giving more importance to the central part of the signal over the vicinity. The choice of the applicability depends on the application and the characteristics of the signal. For example, for orientation estimation, it is desired that the applicability is isotropic [17]. On the other side, image inpainting requires the applicability to be anisotropic depending on the local structure of the image [18]. The handcrafting of the applicability function is not within the scope of this paper as we aim to learn it.

3.2 Propagating Confidences

A core advantage of normalized convolution is the separation between the signal and the confidence allowing confidence adaptive processing and determination of output confidence. The output confidence reflects the density of the input confidence as well as the coherence of the data under the chosen basis. Westelius [19] proposed a measure for the output confidence defined as:

$$\mathbf{c}_{\text{out}} = \left(\frac{\det \mathbf{G}}{\det \mathbf{G}_0} \right)^{\frac{1}{m}} , \quad (4)$$

where $\mathbf{G} = \mathbf{B}^* \mathbf{D}_a \mathbf{D}_c \mathbf{B}$ and $\mathbf{G}_0 = \mathbf{B}^* \mathbf{D}_a \mathbf{B}$. This corresponds to a geometric ratio between the Gramians of the basis \mathbf{B} in case of partial and full confidence. Similarly, Karlholm [20] proposed another measure defined as:

$$\mathbf{c}_{\text{out}} = \frac{1}{\|\mathbf{G}^{-1}\|_2 \|\mathbf{G}_0\|_2} . \quad (5)$$

These two measures were shown to perform well in case of the polynomial and exponential basis [19], [20].

3.3 The Naïve Basis

The most basic choice for the basis is a constant function, and it is denoted as the naïve basis. In this case, the applicability acts as a convolution filter. This choice of applicability simplifies (3) to:

$$\begin{aligned} \hat{\mathbf{r}} &= (\mathbf{1}^* \mathbf{D}_a \mathbf{D}_c \mathbf{1})^{-1} \mathbf{1}^* \mathbf{D}_a \mathbf{D}_c \mathbf{f} \\ &= \frac{\mathbf{a} \cdot (\mathbf{c} \odot \mathbf{f})}{\mathbf{a} \cdot \mathbf{c}} , \end{aligned} \quad (6)$$

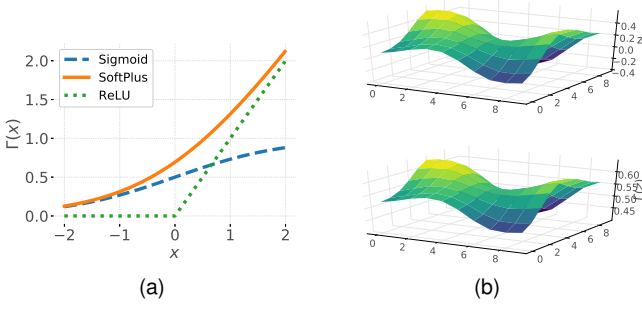


Fig. 2. (a) Examples for differentiable functions with non-negative co-domain, (b) Applying the SoftPlus function to a 2D surface preserves the surface trend.

where the matrix multiplication has been replaced with point-wise multiplication since \mathbf{D}_a and \mathbf{D}_c are diagonal matrices. This can be formulated for the whole signal f as a convolution operation as follows:

$$\hat{r}[k] = \frac{\sum_i^n a[i] f[k-i] c[k-i]}{\sum_i^n a[i] c[k-i]}. \quad (7)$$

4 UNGUIDED NORMALIZED CNNs

Based on our prior work [10], CNNs can be used to learn the optimal applicability function in case of the naïve basis.

4.1 Training the Applicability

As explained in section 3.1, the applicability is a windowing function and it needs to be non-negative. This is enforced in CNN frameworks by applying a suitable differentiable function with non-negative co-domain acting on the convolution kernels prior to the forward pass. During back-propagation, the weights will be differentiated with respect to this function using the chain rule. Examples for differentiable functions with non-negative co-domain are shown in Figure 2a. Figure 2b shows how the SoftPlus function, $\Gamma(z) = \log(1 + \exp(z))$, translates the co-domain of a 2D surface to be non-negative while preserving the surface trend.

Given a function $\Gamma(\cdot)$ with a non-negative co-domain, the gradients of the weight for the l^{th} convolution layer are obtained as:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{W}_{m,n}^l} = \sum_{i,j} \frac{\partial \mathbf{E}}{\partial \mathbf{Z}_{i,j}^l} \cdot \frac{\partial \mathbf{Z}_{i,j}^l}{\partial \Gamma(\mathbf{W}_{m,n}^l)} \cdot \frac{\partial \Gamma(\mathbf{W}_{m,n}^l)}{\partial \mathbf{W}_{m,n}^l}, \quad (8)$$

where \mathbf{E} is the loss between the network output and the ground truth, and $\mathbf{Z}_{i,j}^l$ is the output of the l^{th} layer at locations i, j depending on the weight elements $\mathbf{W}_{m,n}^l$. Accordingly, the forward pass for normalized convolution is defined as:

$$\mathbf{Z}_{i,j}^l = \frac{\sum_{m,n} \mathbf{Z}_{i+m,j+n}^{l-1} \mathbf{C}_{i+m,j+n}^{l-1} \Gamma(\mathbf{W}_{m,n}^l)}{\sum_{m,n} \mathbf{C}_{i+m,j+n}^{l-1} \Gamma(\mathbf{W}_{m,n}^l) + \epsilon} + \mathbf{b}^l, \quad (9)$$

where \mathbf{C}^{l-1} is the output confidence from the previous layer, $\mathbf{W}_{m,n}^l$ is the applicability in this context, \mathbf{b}^l is the bias and ϵ is a constant to prevent division by zero. Note that this is formally a correlation, as it is a common notation in CNNs.

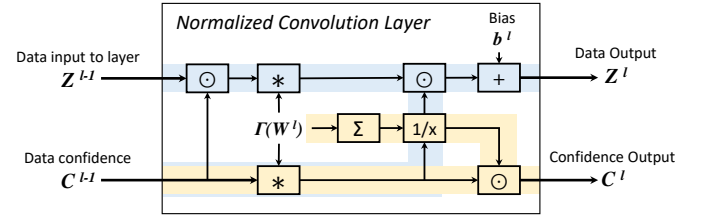


Fig. 3. An illustration of the Normalized Convolution layer that takes in two inputs: data and confidence. The Normalized Convolution layer outputs a data term and a confidence term. Convolution is denoted as $*$, the Hadamard product (point-wise) as \odot , summation as Σ , and point-wise inverse as $1/x$.

4.2 Propagating the Confidence

The confidence output measures described in section 3.2 have been shown to give reasonable results in case of non-naïve basis [19], [20]. In our earlier work [10], we proposed a confidence output measure for the naïve basis case. The proposed measure is derived from (4) and can utilize the already computed terms in the forward path. The measure is defined as:

$$\mathbf{C}_{i,j}^l = \frac{\sum_{m,n} \mathbf{C}_{i+m,j+n}^{l-1} \Gamma(\mathbf{W}_{m,n}^l) + \epsilon}{\sum_{m,n} \Gamma(\mathbf{W}_{m,n}^l)}. \quad (10)$$

This measure allows propagating confidence between CNN layers without facing the problem of "validity masks saturation" as described in [7], which affects several methods in the literature [1], [4], [5].

4.3 The Normalized CNN Layer

The standard convolution layer in CNN frameworks can be replaced by a normalized convolution layer with minor modifications. First, the layer takes in two inputs simultaneously, the data and its confidence. The forward pass is then modified according to (9) and the back-propagation is modified to include a derivative term for the non-negativity enforcement function $\Gamma(\cdot)$ as described in (8). To propagate the confidence to consecutive layers, the already-calculated denominator term in (9) is normalized by the sum of the filter elements as shown in (10). An illustration of the Normalized CNN layer is shown in Figure 3.

4.4 The Loss Function

In networks that perform pixel-wise tasks such as inpainting, upsampling, or segmentation, it is very common to use the $L1$ or the $L2$ norm. However, the former ignores outliers and focuses on the global level, while the latter focuses on local regions that have outliers. A good compromise is the Huber norm [21], which is defined as:

$$\|z - t\|_H = \begin{cases} \frac{1}{2}(z - t)^2, & |z - t| < \delta \\ \delta|z - t| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases} \quad (11)$$

The Huber norm corresponds to the $L2$ norm if the error is less than δ and to the $L1$ norm otherwise. Usually, the value of δ is set to 1 within CNN frameworks and referred to as the *Smooth L1* loss.

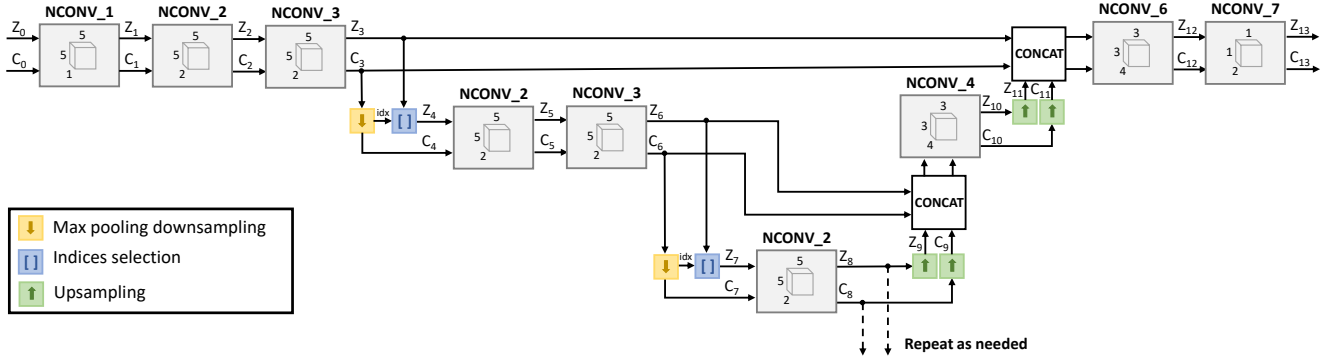


Fig. 4. Our proposed multi-scale architecture for the task of unguided scene depth completion that utilizes normalized convolution layers. Downsampling is performed using max pooling on confidence maps and the indices of the pooled pixels are used to select the pixels with highest confidences from the feature maps. Different scales are fused by upsampling the coarser scale and concatenating it with the finer scale. A normalized convolution layer is then used to fuse the feature maps based on the confidence information. Finally, a 1×1 normalized convolution layer is used to merge different channels into one channel and produce a dense output and an output confidence map.

In networks with normalized convolution layers, it is desirable to minimize the data error and maximize the output confidence at the same time. Thus, a loss function that simultaneously achieves both objectives is desired. Assume a data error term using the Huber norm:

$$\mathbf{E}_{i,j} = \|\mathbf{Z}_{i,j}^L - \mathbf{T}_{i,j}\|_H, \quad (12)$$

where $\mathbf{Z}_{i,j}^L$ is the data output from the last layer L and $\mathbf{T}_{i,j}$ is the data ground truth. This is complemented with a term to maximize the confidence and the total loss $\tilde{\mathbf{E}}$ becomes:

$$\tilde{\mathbf{E}}_{i,j} = \mathbf{E}_{i,j} - \frac{1}{p} [\mathbf{C}_{i,j}^L - \mathbf{E}_{i,j} \mathbf{C}_{i,j}^L], \quad (13)$$

where $\mathbf{C}_{i,j}^L$ is the output confidence and p is the epoch number. The confidence term is decaying by dividing it by the epoch number p to prevent it from dominating the loss when the data error term starts to converge.

4.5 Unguided Normalized CNN Architecture

In [10], we proposed a hierarchical multi-scale architecture inspired by the U-Net [9], which shares weights between different scales. The architecture acts as a generic estimator for different scales and gives a good approximation for the dense output at a very low computation cost. An illustration of the architecture is shown in Figure 4. At the first scale, a normalized convolution layer takes in the sparse input as well as a confidence map. Afterwards, two normalized convolution layers are applied followed by downsampling. The downsampling process is performed by applying a max pooling operator on the output confidence from the last normalized layer while keeping the indices of the pooled values as in unpooling operations [22]. These indices are then used to select the values from the features maps that have the highest confidences. This enables propagating the most confident data to the subsequent scale. To maintain the absolute levels of confidences after downsampling, we divide the downsampled confidences by the Jacobian of the scaling.

The aforementioned pipeline is repeated as required depending on the sparsity level of the data. In order to fuse different scales, the output and the corresponding confidence

from the last normalized convolution layers are upsampled using nearest-neighbor interpolation and concatenated with the corresponding scale through a skip connection. After each concatenation, a new normalized convolution layer is utilized to fuse data from the two scales based on their confidences. Finally, a 1×1 normalized convolution layer is used to fuse different channels into one channel that corresponds to the dense output. In addition to the dense output, a confidence output is also available that holds information about the confidence distribution of the output. The output confidence can be useful for safety application or for subsequent stages in the pipeline.

5 GUIDED NORMALIZED CNNs

In this section, we extend the unguided normalized convolution architecture described in section 4 with RGB and output confidence guidance. The unguided architecture acts as a generic estimator for different scales that is learned from the data. However, this generic estimator shows weaknesses at local regions with discontinuities such as edges and rough surfaces. Figure 5 shows an example of the spatial error distribution for the output from the unguided normalized convolution network on the task of depth completion. It is clear that regions with edges have larger errors than flat regions. Therefore, auxiliary data such as RGB images and surface normals can be used to alleviate this problem by providing contextual information to the network.

5.1 RGB Image Fusion

RGB images can be very useful in guiding the network since convolution layers typically act as feature extractors. These features are usually edges, corners, color, or texture. Providing this information to the network was demonstrated to improve the results [6], [7], [8], especially across edges and in rough surfaces. Therefore, we incorporate RGB information into our network to handle discontinuities at edges.

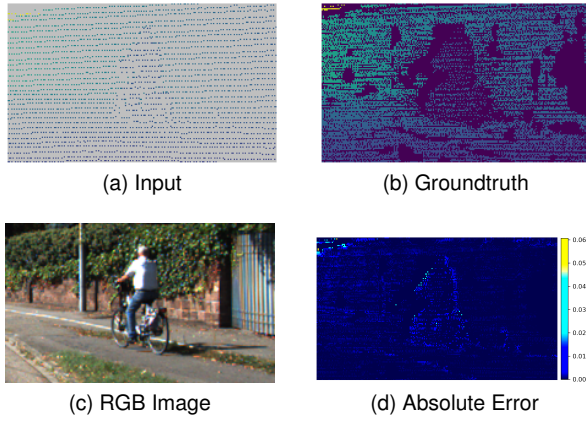


Fig. 5. An example for the spatial error distribution of the output from our unguided normalized convolution network on the task of depth completion. (d) shows that the error is distributed around edges.

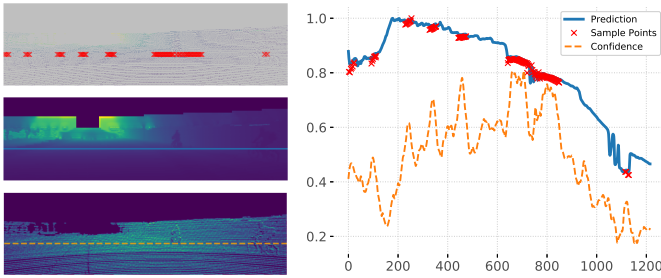


Fig. 6. An example of the output confidence from our unguided normalized convolution network on the task of depth completion. Images on the left are from top-to-bottom: sparse input, the dense output from the unguided normalized convolution network and the output confidence. The plot on the right shows the corresponding values for row 217. The red crosses are the sample points from the sparse input, the blue curve is the dense prediction and the orange curve is the output confidence (smoothed). It is shown that regions with high density of sample points tend to have a higher confidence. Note that all values are normalized to $[0;1]$.

5.2 The Output Confidence Fusion

The RGB data is fused with a new form of auxiliary data, which is the output confidence from the unguided normalized convolution network. This output confidence holds useful information about the reliability of each pixel in the image. For example, regions in the sparse input that have a high density of sample points should have a higher confidence in the output. Figure 6 illustrates how the output confidence from our unguided network (the dashed orange curve) is correlated with the density of the sample points in the input (the red crosses). The figure also shows how our unguided network can find a good approximation (the blue curve) for the sparse input. Therefore, We use the output confidence as an input to our guided network to provide information about reliability of different pixels in the output from the unguided network. We will demonstrate in the experiments that the use of the output confidence improves the depth results with approximately 10%.

5.3 Network Architecture

We aim to fuse the sparse depth, the RGB image, and the output confidence to produce a dense depth map. Therefore,

we look into two of the commonly used architectures from the literature on the task of guided scene depth completion. The first is a simple multi-stream network inspired by [5] which shares similarities with our proposed work and the second is an encoder-decoder architecture with skip-connections inspired by [4], [8], [9], which was demonstrated to achieve state-of-the-art results. The former employs a late-fusion scheme for combining different streams, while the latter adopts an early-fusion scheme. Table 1 summarizes some methods in the literature under this categorization.

We investigate all cases from Table 1 with both architectures and both fusion schemes. First, we utilize a multi-stream network with late fusion as is shown in Figure 7a. One stream contains our unguided network described in section 4 followed by refinement layers and the other contains the image concatenated with the output confidence from the unguided network. Eventually, both streams are fused by concatenation and then fed to a fusion network that produces the final output. In addition, we train the same network in an early fusion manner as illustrated in Figure 7c. Note that the number of channels for the depth stream was added to the RGB stream, while the number of channels for the fusion network were kept unchanged.

Secondly, we adopt a multi-scale encoder-decoder network with late fusion as shown in Figure 7b. One stream contains our unguided network followed by an encoder, where all convolution layers apply a stride of 2 to perform downsampling and a ReLU activation. In the other stream, both RGB image and output confidence from the unguided network are concatenated and then fed to an encoder similar to the previous one, but with a larger number of channels per layer. At the decoder, feature maps from both streams are upsampled and then concatenated with the feature maps having the same scale from the encoder. Afterwards, convolution is performed followed by Leaky ReLU activation with $\alpha = 0.2$. The final layer produces the final dense output. Similarly, we also apply an early fusion scheme to this architecture by concatenating both the output and the output confidence from the unguided network with the RGB image. Then, they are fed into a similar encoder-decoder as illustrated in Figure 7d. In this way, we evaluate all options listed in Table 1.

For all networks described here, we aim to reduce the number of parameters for computational efficiency. Therefore, we use a fixed number of feature channels of 16 per each input channel. For example, sparse depth input has only one channel, so we use 16 features at all layers in the depth stream, while we use $16 \times 4 = 64$ for the RGB image and the output confidence. Our experiments will demonstrate how the use of the unguided normalized convolution

| | Early Fusion | Late Fusion |
|-----------------|--------------|-------------|
| Multi-stream | • | [5] |
| Encoder-decoder | [4], [8] | [6], [7] |

TABLE 1

A categorization of the state-of-the-art methods depending on the architecture and the fusion scheme.

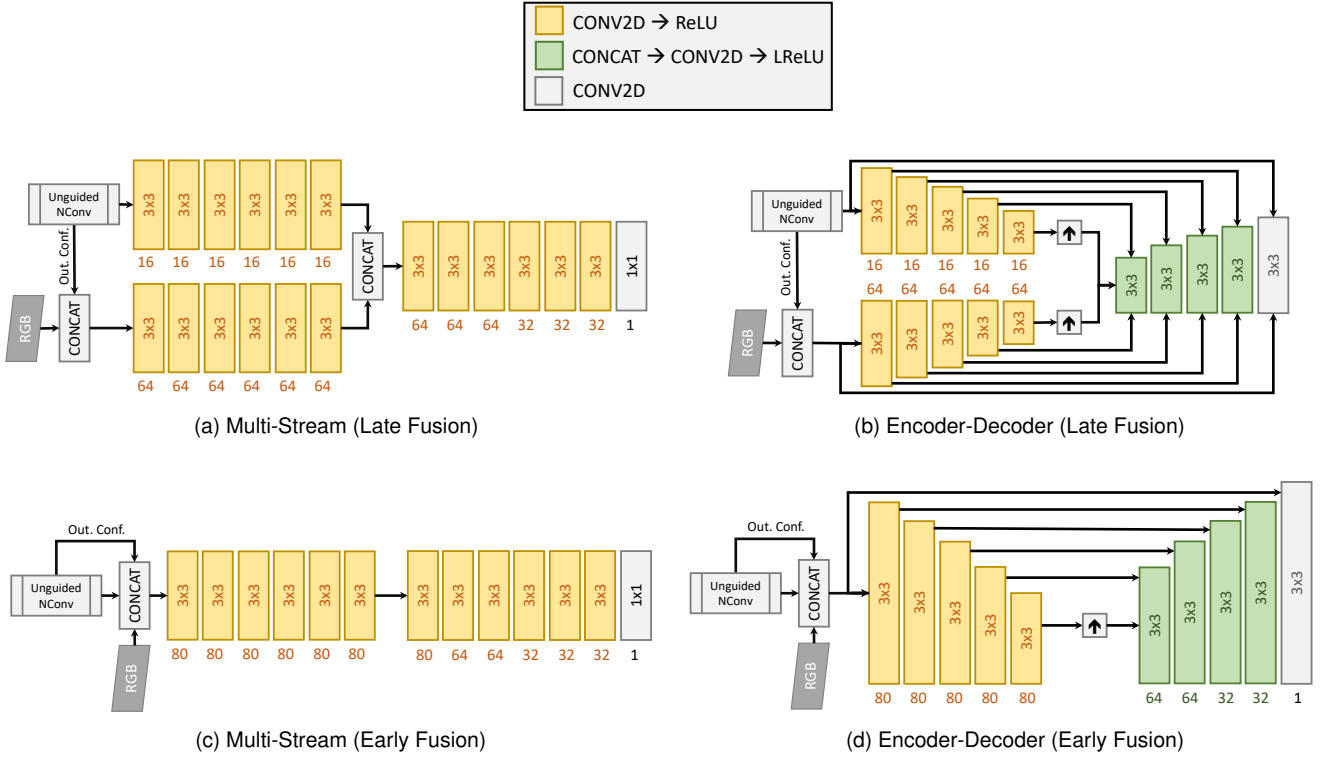


Fig. 7. **(a)** A multi-stream architecture that contains a stream for depth and another stream for RGB+Output Confidence feature extraction. Afterwards, a fusion network combines both streams to produce the final dense output. **(d)** A multi-scale encoder-decoder architecture where depth is fed to the unguided network followed by an encoder and output confidence and RGB image are concatenated then fed to a similar encoder. Both streams have skip-connection to the decoder between the corresponding scales. **(c)** is similar to **(a)**, but with early fusion and **(d)** is similar to **(b)** but with early fusion.

sub-network allows achieving state-of-the-art results on the task of guided depth completion without requiring huge networks with millions of parameters as in [6], [7], [8].

6 EXPERIMENTS

To demonstrate the capabilities of our proposed approach, we evaluate our proposed networks on the KITTI-Depth Benchmark [1] for the task of depth completion.

6.1 Dataset

The KITTI-Depth dataset [1] includes depth maps from projected LiDAR point clouds that were matched against the depth estimation from the stereo cameras. The depth images are highly sparse with only 5% of the pixels available and the rest is missing. The dataset has 86k training images, 7k validation images, and 1k test set images on the benchmark server with no access to the ground truth. The test set will be used for evaluation against other methods, while a subset of the validation set (1k images) will be used for the analysis of our own method. We also use the RGB images from the raw data of the original KITTI dataset [23]. It is worth mentioning that the groundtruth of the KITTI-Depth dataset is incomplete since pixels that were not consistent with the groundtruth from the stereo disparity have been removed.

6.2 Experimental Setup

All experiments were performed on a workstation with 6 CPU cores, 112 GB of RAM, and an NVIDIA Tesla V100 GPU

with 16 GB of memory. All guided networks were trained until convergence on the full training set with a batch size of 4 using the ADAM optimizer with an initial learning rate of $1e-4$ and a decaying factor of 0.1 after 20 epochs. When only the unguided normalized convolution network is trained, we train on 10k images of the training set for 5 epochs using the ADAM optimizer with a learning rate of 0.01.

6.3 Evaluation Metrics

We adopt the evaluation metrics used in the benchmark: the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) computed on the depth values. The MAE is an unbiased error metric takes an average of the error across the whole image and it is defined as:

$$MAE(Z, T) = \frac{1}{MN} \left[\sum_{i=0}^N \sum_{j=0}^M |Z(i, j) - T(i, j)| \right], \quad (14)$$

while the RMSE penalizes outliers and it is defined as:

$$RMSE(Z, T) = \frac{1}{MN} \left[\sum_{i=0}^N \sum_{j=0}^M |Z(i, j) - T(i, j)|^2 \right]^{1/2}. \quad (15)$$

Additionally, we also use iMAE and iRMSE, which are calculated on the disparity instead of the depth, i.e. inverse depth.

| | MAE [mm] | RMSE [mm] | iMAE [1/km] | iRMSE [1/km] |
|----------------|---------------|---------------|----------------|-----------------|
| MS-Net[LF] | 209.56 | 908.76 | 0.90 | 2.50 |
| MS-Net[EF] | 209.75 | 932.01 | 0.92 | 2.64 |
| EncDec-Net[LF] | 295.92 | 1053.91 | 1.31 | 3.42 |
| EncDec-Net[EF] | 236.83 | 1007.71 | 0.99 | 2.75 |

TABLE 2

A quantitative comparison between different fusion schemes (described in section 5) on the selected **validation** set of the KITTI-Depth dataset [1]. The different fusion schemes are *MS-Net[LF]*, which is the multi-stream architecture with late fusion (Figure 7a), *MS-Net[EF]*, which applies early fusion (Figure 7c), *EncDec-Net[LF]*, which is the encoder-decoder architecture with late fusion (Figure 7b), and *EncDec-Net[EF]*, which applies early fusion (Figure 7d). *MS-Net[LF]* achieves the best results with respect to all evaluation metrics.

6.4 Evaluating Guided Normalized CNNs

First, we evaluate the different architectures described in section 5.3. The first architecture is a multi-stream network with early fusion denoted as *MS-Net[EF]* and its variant with late fusion *MS-Net[LF]*. The second architecture is an encoder-decoder architecture with early-fusion denoted as *EncDec-Net[EF]* and its variant with late fusion *EncDec-Net[LF]*. For a fair comparison and to neutralize any influence from inefficient gradients, we train the unguided network separately using our proposed loss in (13) and then attach it to the guided networks in comparison while freezing its weights. All networks are trained using the Huber norm loss described in (11). Table 2 shows that the multi-stream network with late fusion, *MS-Net[LF]*, outperforms all the other networks with respect to all evaluation metrics. The multi-stream network with early fusion, *MS-Net[EF]*, achieves similar results with respect to MAE and iMAE, but the RMSE and iRMSE are slightly higher. For the encoder-decoder networks, *EncDec-Net[EF]* with early fusion achieves better results than the network with late fusion contrarily to the multi-stream network.

Next, we compare our best performing architecture, *MS-Net[LF]*, with the state-of-the-art methods.

6.5 State-of-the-art Comparison

We compare *MS-Net[LF]* against all published methods that have been submitted to the KITTI-Depth benchmark [1]. *SparseConv* [1] proposed a sparsity invariant layer that normalizes the sparse input using a binary validity mask. They also created three baselines: *CNN*, which trains a simple network directly on the sparse input, *CNN+mask*, which concatenates the validity mask with the sparse input and trains the same network, and *NN+CNN*, which performs nearest neighbor interpolation on the sparse input and then trains a refinement network. *ADNN* [3] employed compressed sensing within CNNs to handle the sparsity in data. *IP-Basic* [24] applied an extensive search on variations of morphology and simple image processing techniques to interpolate the sparse input. *Spade* [7] proposed an encoder-decoder architecture with late-fusion to reconstruct a dense output from the sparse input. *Sparse-to-Dense* [8] proposed a self-supervised approach to alleviate the incomplete groundtruth in the KITTI-Depth dataset. Their self-supervised approach requires a sequence of sparse depth

| | MAE [mm] | RMSE [mm] | iMAE [1/km] | iRMSE [1/km] |
|-----------------------------------|---------------|---------------|----------------|-----------------|
| CNN [1] | 620.00 | 2690.00 | - | - |
| CNN+mask [1] | 790.00 | 1940.00 | - | - |
| SparseConv [1] | 481.27 | 1601.33 | 1.78 | 4.94 |
| NN+CNN [1] | 416.14 | 1419.75 | 1.29 | 3.25 |
| ADNN [3] | 439.48 | 1325.37 | 3.19 | 59.39 |
| IP-Basic [24] | 302.60 | 1288.46 | 1.29 | 3.78 |
| NConv-CNN (d) (ours) | 360.28 | 1268.22 | 1.52 | 4.67 |
| Spade (d) [7] | 248.32 | 1035.29 | 0.98 | 2.60 |
| Sparse-to-Dense (d) [8] | 288.64 | 954.36 | 1.35 | 3.21 |
| HMS-Net (d) [6] | 258.48 | 937.48 | 1.14 | 2.93 |
| Spade (gd) [7] | 234.81 | 917.64 | <i>0.95</i> | 2.17 |
| NConv-CNN-L1 (gd) (ours) | 207.77 | 859.22 | 0.92 | 2.52 |
| HMS-Net (gd) [6] | 253.47 | 841.78 | 1.13 | 2.73 |
| NConv-CNN-L2 (gd) (ours) | 233.26 | 829.98 | 1.03 | 2.60 |
| Sparse-to-Dense (gd) [8] | 249.95 | 814.73 | 1.21 | 2.80 |

TABLE 3

Quantitative results for methods in comparison on the KITTI-Depth benchmark [1]. The best method is shown in **bold** and the second best is shown in *italic*. The performance is shown on the **test** set for which the results were submitted to the benchmark evaluation server with no access to the ground truth. For all methods, (d) denotes that only the sparse depth input was used, while (gd) denotes that both the sparse depth and the RGB images were used. Our method *NConv-CNN-L1 (gd)* outperforms all methods with respect to the MAE error with a large margin. Our method *NConv-CNN-L2 (gd)* trained using the L2-norm achieves second best results with respect to RMSE with a small margin to the top-performing method *Sparse-to-Dense (gd)*.

and RGB images to reconstruct a dense depth map. Finally, *HMS-Net* derived variations of the sparsity invariant layer that were used to deploy larger and more complex networks.

As mentioned earlier, our *MS-Net[LF]* with late fusion provides the best results. We compare our *MS-Net[LF]* (denoted as *NConv-CNN-L1 (gd)* for the benchmark) architecture against the state-of-the-art methods. Since the KITTI-Depth benchmark ranks methods based on the RMSE error, we train the same network but with an L2-norm loss and we denote it as *NConv-CNN-L2 (gd)*. Note that *NConv-CNN (d)* is our unguided normalized convolution network described in section 4.

Quantitative results on the test set of the KITTI-Depth dataset [1] using evaluation metrics described above are shown in Table 3. Our method *NConv-CNN-L1 (gd)* outperforms all the other methods with respect to the MAE with a large margin. When our method was trained using the L2-norm, it was able to perform the second best with respect to RMSE with a small margin compared to *Sparse-to-Dense (gd)*. However, our method has a significantly lower number of parameters ($\sim 355k$) compared to *Sparse-to-Dense (gd)* which has $\sim 5.5M$ parameters. This demonstrates that our method achieves state-of-the-art results while requiring a very small number of parameters. *Spade (gd)* on the other hand achieves the best results with respect to iRMSE since it was trained on disparity using the L2-norm. However, our method *NConv-CNN-L1 (gd)* still outperformed *Spade (gd)* with respect to iMAE despite being trained on depth.

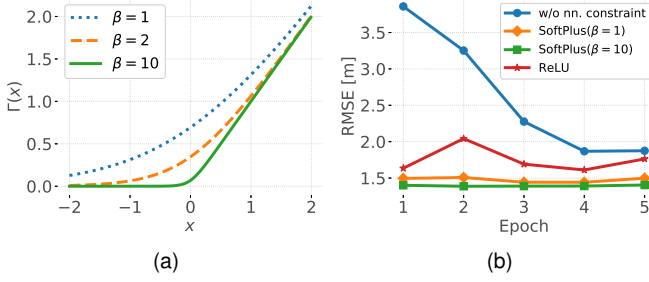


Fig. 8. (a) The SoftPlus function with different scaling factors. At $\beta = 10$, the SoftPlus function gives a good differentiable approximation of the ReLU function. (b) Convergence curves for our unguided network with the presence and absence of the non-negativity constraint.

6.6 The Choice of the Non-negative Function

The choice of the non-negative function mainly depends on the desired characteristics. The most obvious choice is to clamp non-negative values as in the ReLU function. However, the ReLU has problems with the discontinuous derivative. Therefore, we consider a good continuous approximation for the ReLU, the SoftPlus function $\Gamma(x) = \frac{1}{\beta} \log(1 + \exp(\beta x))$. With the right choice of the β , we can have a very good approximation of the ReLU function, e.g. when $\beta = 10$, as shown in Figure 8a. The derivative of the SoftPlus function is continuous which gives more flexibility to the network during training. Figure 8b shows how the choice of the non-negative function affects the convergence of the network. The ReLU function shows poor convergence and keeps fluctuating as the derivatives are not continuous and the network struggles to converge. SoftPlus on the other hand converges very fast due to the continuous derivative and with the right choice of β , the results are improved.

6.7 The Non-negativity Constraint Impact

To study the effect of enforcing non-negativity constraints on our trained filters, we compare the convergence of our proposed unguided normalized convolution module described in section 4 in the presence and the absence of the non-negativity enforcement. Since our proposed confidence measure cannot be used in the absence of the non-negativity constraints, we propagate confidences by applying a max pooling operations on the confidence map as in [1], [5], [6]. Besides, only the data error term in our proposed loss in (13) is used because of the absence of output confidence.

| Method [Non-Negativity Function] | MAE [mm] | RMSE [mm] | iMAE [1/km] | iRMSE [1/km] |
|---|---------------|---------------|----------------|-----------------|
| NConv-CNN-L1 (gd) [SoftPlus($\beta = 10$)] | 209.56 | 908.76 | 0.90 | 2.50 |
| NConv-CNN-L1 (gd) [N/A] | 277.67 | 1042.65 | 1.28 | 3.50 |

TABLE 4

The impact of enforcing non-negativity on normalized convolution layers. *NConv-CNN-L1 (gd)* with SoftPlus achieves significantly better results than the case without enforcing non-negativity. The results shown are for the selected **validation** set of the KITTI-Depth dataset [1].

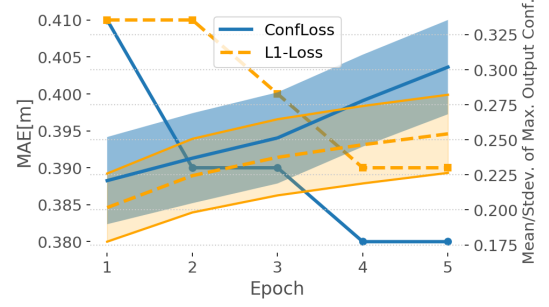


Fig. 9. The impact of the proposed loss on confidence levels. The right axis represents the mean and standard deviation of maximum output confidence value over all images, while the left axis has the MAE in meters. When using a loss with only a data term (Huber norm Loss), output confidence levels are lower, while our proposed loss achieves monotonically increasing confidence levels as well as a lower MAE. Note that the shaded area represents the standard deviation.

Both networks were trained on 10k training images for 5 epochs with a constant learning rate of 0.01. The networks were trained on the disparity instead of depth since both networks perform better when trained on disparity. Figure 8b shows the convergence curves for both networks. When enforcing the non-negativity constraints, the network converges after 1 epoch, while the other network not enforcing the non-negativity constraint starts to converge after 4 epochs and to a higher error value. This demonstrates that our proposed non-negativity constraint helps the network to converge faster and to achieve significantly better results.

The overall effect of the non-negativity constraints on the guided network is shown in Table 4. Discarding the non-negativity constraint significantly degrades the results with respect to all evaluation metrics. This is potentially caused by the lack of guidance provided by the output confidence or by the the poor estimation of depth produced by the unguided network.

6.8 The Impact of the Proposed Loss

To study the impact of the confidence term in our proposed loss in (13), we train our unguided normalized convolution network described in section 4 twice: once using the proposed loss function with confidence term and once using only the Huber norm loss (12). Figure 9 shows the mean and the standard deviation of the maximum output confidence over images in the selected validation set on the right axis and the MAE error on the left axis. The network trained with our proposed loss produces a monotonically increasing confidence map while improving the data error until convergence. On the other hand, the network trained with only the Huber norm loss has lower levels of output confidence in general and it also converges to a higher MAE.

6.9 The Learned Filters

The unguided normalized convolution network acts as a multi-scale generic estimator for the data. During training, this generic estimator is learned from the data using back-propagation. Some examples of the learned filters are shown in Figure 10. The first row of the figure shows some of the

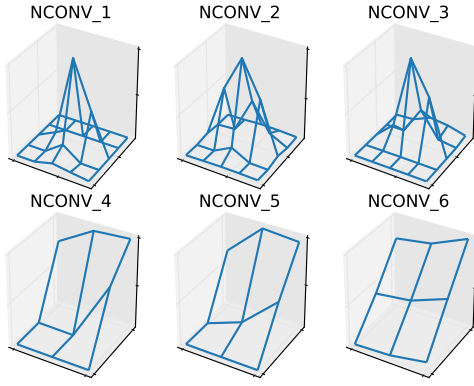


Fig. 10. A visualization of the learned filters from our proposed unguided normalized convolution network. Layer names match their correspondences in Figure 4.

learned filters for layers NCONV[1-3], which are asymmetric low-pass filters. Those filters attempt to construct the missing pixels from their neighborhood. On the other hand, the second row of the figure shows the learned filters for layers NCONV[4-6], which resemble linear ramps. Those filters try to scale the output from each scale for an efficient fusion with other scales.

6.10 Guided Normalized CNN Ablation Study

In this section, we study the effect of different components of our best performing guided network. Since the benchmarks is ranked based on the RMSE, then we use *NConv-CNN-L2 (gd)* as a baseline. The baseline is *MS-Net[LF]* in section 5.3 trained using an L2-norm loss. When the whole network was trained end-to-end, the results are slightly degraded as shown in Table 5. This could be a result of vanishing gradients since the network becomes deeper. Removing either the depth refinement layers or the output confidence has almost the same influence on the performance of the network as shown in Table 5. The reason might be that the depth refinement layers contribute to handling some outliers that violate the estimation from the unguided network. On the other hand, the use of the output confidence provides the RGB stream with information about regions with low confidence that are highly likely

| | MAE [mm] | RMSE [mm] | iMAE [1/km] | iRMSE [1/km] |
|-----------------------|---------------|---------------|----------------|-----------------|
| Baseline | 233.25 | 870.82 | 1.03 | 2.75 |
| Baseline (end-to-end) | 244.98 | 886.09 | 1.11 | 3.02 |
| Baseline w/o DR | 245.11 | 912.82 | 1.10 | 3.03 |
| Baseline w/o OC | 244.87 | 919.55 | 1.09 | 2.97 |

TABLE 5

Baseline refers to *NConv-CNN-L2 (gd)*, *DR* refers to the Depth Refinement layers as indicated in Figure 1 and *OC* is the use of the output confidence in the RGB feature extraction network. The results shown in this table are on the selected **validation** set of the KITTI-Depth dataset [1]. When the baseline is trained end-to-end, the performance is slightly degraded. The depth refinement layers also contribute to the results. Discarding the output confidence, degrades the results.

| | #Params | Runtime [sec] |
|-----------------------------------|--------------------------------------|---------------|
| Sparse-to-Dense (d) [8] | 5.53×10^6 | 0.04 |
| SparseConv [1] | 2.5×10^4 | 0.01 |
| ADNN [3] | 1.7×10^3 | 0.04 |
| NConv-CNN (d) (ours) | 4.8×10^3 | 0.01 |
| Sparse-to-Dense (gd) [8] | 5.54×10^6 | 0.08 |
| Spade (gd) [7] | $\sim 5.3 \times 10^6$ | 0.07 |
| NConv-CNN-L2 (gd) (ours) | 3.55×10^5 | 0.02 |

TABLE 6

Number of parameters and runtime for some methods in comparison (lower is better). Our unguided network *NConv-CNN (d)* has a significantly lower number of parameters than all other unguided networks. Our guided network *NConv-CNN-L2 (gd)* has at least one order of magnitude fewer parameters than all other guided methods. Both our unguided and guided networks have the lowest runtime due to the small number of parameters. Note that the exact number of *Spade (gd)* [7] is not mentioned in the paper, so we give the number of parameters for the NASNet [25] that they utilize.

to contribute to the error. Therefore, discarding the output confidence increases the error by approximately 10%, which demonstrates its contribution.

6.11 Number of Parameters and Runtime Comparison

In this section, we compare the number of parameters and the runtime for some of the methods in comparison. The number of parameters is calculated from the network descriptions in the related papers, while the runtime is taken from the benchmark server [1]. Table 6 shows that our unguided network *NConv-CNN (d)* has the lowest number of parameters and runtime compared to all other unguided methods, which makes it most suitable for embedded applications with limited computational resources. We maintain the low number of parameters in our guided network *NConv-CNN-L2 (gd)* that has 355k parameters, which is at least one order of magnitude fewer than all other guided methods in the comparison. This leads to a the lowest runtime of 0.02 seconds among the guided methods which satisfies real-time constraints and has a high potential to maintain the real-time performance if evaluated on embedded devices. The huge decrease in the number of parameters did not degrade the results as was shown in the quantitative results earlier, which demonstrates the efficiency of our proposed method. Note that the runtime between our unguided and guided network do not scale linearly as our unguided network includes many time consuming operations such as downsampling, slicing and upsampling, while the guided network adds only convolution operations.

6.12 Qualitative Results

Figure 11 shows some qualitative results for the top performing methods from the benchmark server. For our method, we show examples from *NConv-CNN-L2 (gd)* that was trained using the L2-norm loss which achieved the lowest RMSE error. Generally, our method and the other two methods in comparison perform equally well with some minor differences. Our method performs better with tiny details (highlighted using the yellow boxes) such as the car edges in the first row and the poles far away in the

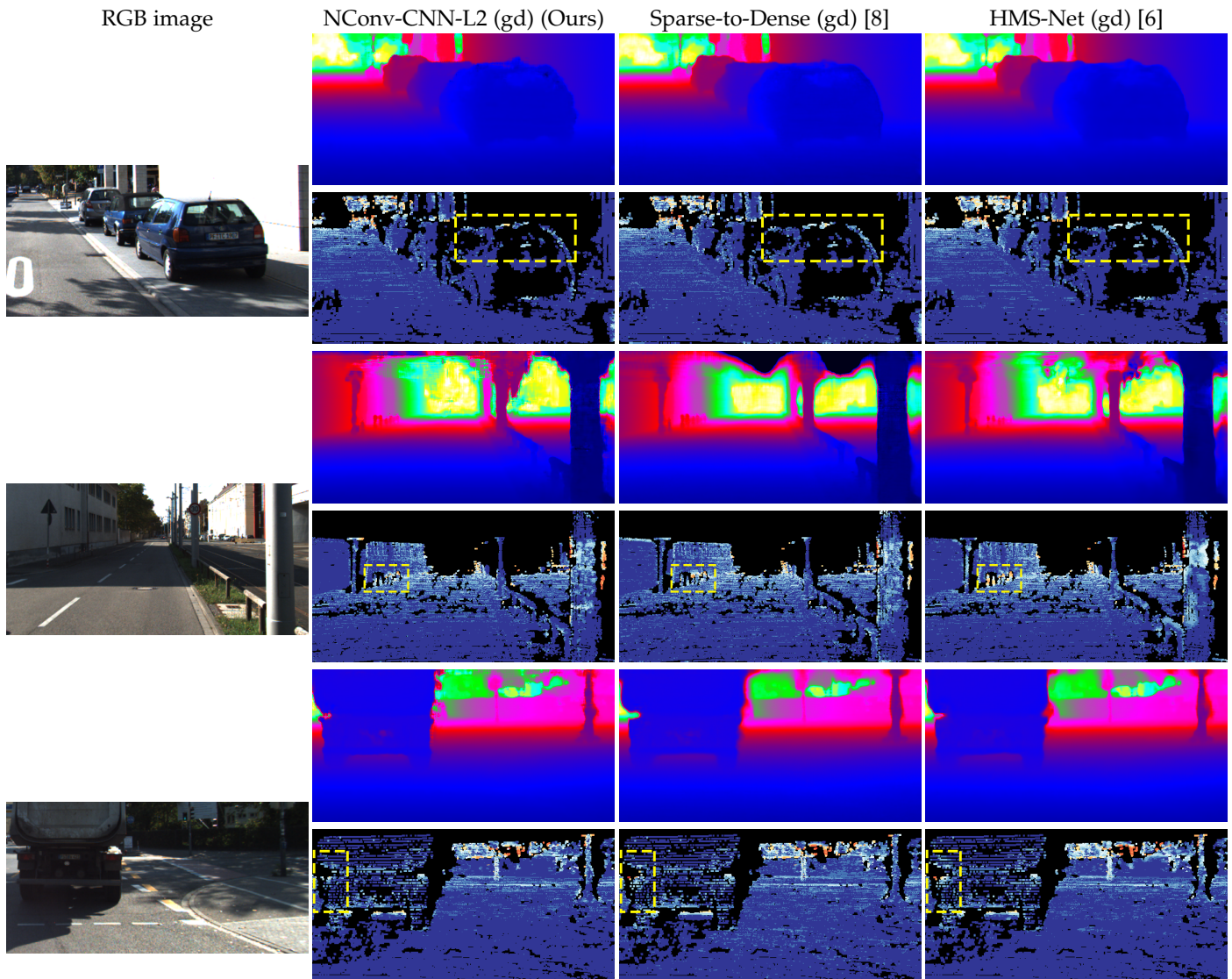


Fig. 11. Some qualitative examples for the top three performing methods from the KITTI-Depth dataset [1] on the task of scene depth completion. Our method *NConv-CNN-L2 (gd)* performs slightly better in handling outliers as highlighted with the yellow boxes, while *Sparse-to-Dense* produces smoother edges due to the use of a smoothness loss. Note that this figure is best viewed on screens.

second row. Our method was also able to remove outliers and highly sparse regions as shown on the third row. On the other hand, *Sparse-to-Dense (gd)* produces smoother edges than our method and *HMS-Net (gd)* due to the use of a smoothness loss that penalizes the second derivative of the prediction during training.

7 CONCLUSION

In this paper, we have proposed a normalized convolution layer for unguided scene depth completion on highly sparse data by treating the validity masks as a continuous confidence field. We proposed a method to propagate confidences between CNN layers. This enabled us to produce a point-wise continuous confidence map for the output from the deep network. For fast convergence, we algebraically constrained the learned filters to be non-negative, acting as a weighting function for the neighborhood. Furthermore, a loss function is proposed that simultaneously minimizes

the data error and maximizes the output confidence. Finally, we proposed a fusion strategy to combine depth and RGB information in our normalized convolution network framework for incorporating structural information. We performed comprehensive experiments on the KITTI-Depth benchmark and achieved superior performance with significantly fewer network parameters compared to the state-of-the-art.

ACKNOWLEDGMENTS

This research is funded by Vinnova through grant CYCLA, the Swedish Research Council through a framework grant for the project Energy Minimization for Computational Cameras (2014-6227), CENIIT grant (18.14), and VR starting grant (2016-05543).

REFERENCES

- [1] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity Invariant CNNs," aug 2017. [Online]. Available: <http://arxiv.org/abs/1708.06500>
- [2] J. S. Ren, L. Xu, Q. Yan, and W. Sun, "Shepard convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 901–909.
- [3] N. Chodosh, C. Wang, and S. Lucey, "Deep Convolutional Compressed Sensing for LiDAR Depth Completion," mar 2018. [Online]. Available: <http://arxiv.org/abs/1803.08949>
- [4] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image Inpainting for Irregular Holes Using Partial Convolutions," apr 2018. [Online]. Available: <http://arxiv.org/abs/1804.07723>
- [5] J. Hua and X. Gong, "A normalized convolutional neural network for guided sparse depth upsampling," in *IJCAI*, 2018, pp. 2283–2290.
- [6] Z. Huang, J. Fan, S. Yi, X. Wang, and H. Li, "HMS-Net: Hierarchical Multi-scale Sparsity-invariant Network for Sparse Depth Completion," *ArXiv e-prints*, Aug. 2018.
- [7] M. Jaritz, R. de Charette, E. Wirbel, X. Perrotton, and F. Nashashibi, "Sparse and dense data with cnns: Depth completion and semantic segmentation," *arXiv preprint arXiv:1808.00769*, 2018.
- [8] F. Ma, G. Venturelli Cavalheiro, and S. Karaman, "Self-supervised Sparse-to-Dense: Self-supervised Depth Completion from LiDAR and Monocular Camera," *ArXiv e-prints*, Jul. 2018.
- [9] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation." Springer, Cham, oct 2015, pp. 234–241. [Online]. Available: http://link.springer.com/10.1007/978-3-319-24574-4_{_}28
- [10] A. Eldesokey, M. Felsberg, and F. S. Khan, "Propagating confidences through cnns for sparse data regression," in *The British Machine Vision Conference (BMVC)*, Northumbria University, Newcastle upon Tyne, England, UK, 3-6 September, 2018, 2018.
- [11] N. Yang, Y. Kim, and R. Park, "Depth hole filling using the depth distribution of neighboring regions of depth holes in the kinect sensor," in *2012 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC 2012)*, Aug 2012, pp. 658–661.
- [12] Y. Shen, J. Li, and C. L., "Depth map enhancement method based on joint bilateral filter," in *2014 7th International Congress on Image and Signal Processing*, Oct 2014, pp. 153–158.
- [13] Y. Chiu, J. Leou, and H. Hsiao, "Super-resolution reconstruction for kinect 3d data," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, June 2014, pp. 2712–2715.
- [14] S. Wirges, B. Roxin, E. Rehder, T. Khner, and M. Lauer, "Guided depth upsampling for precise mapping of urban environments," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 1140–1145.
- [15] Y. Konno, Y. Monno, D. Kiku, M. Tanaka, and M. Okutomi, "Intensity guided depth upsampling by residual interpolation," in *The Abstracts of the international conference on advanced mechatronics: toward evolutionary fusion of IT and mechatronics: ICAM 2015.6*. The Japan Society of Mechanical Engineers, 2015, pp. 1–2.
- [16] H. Knutsson and C.-F. Westin, "Normalized and differential convolution," in *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*. IEEE, 1993, pp. 515–523.
- [17] G. Farnéback, "Polynomial expansion for orientation and motion estimation," Ph.D. dissertation, Linköping University Electronic Press, 2002.
- [18] T. Q. Pham and L. J. Van Vliet, "Normalized averaging using adaptive applicability functions with applications in image reconstruction from sparsely and randomly sampled data," in *Scandinavian Conference on Image Analysis*. Springer, 2003, pp. 485–492.
- [19] C.-J. Westelius, "Focus of attention and gaze control for robot vision," Ph.D. dissertation, Linköping University, Computer Vision, The Institute of Technology, 1995.
- [20] J. Karlholm, "Local signal models for image sequence analysis," Ph.D. dissertation, Linköping University, Computer Vision, The Institute of Technology, 1998.
- [21] P. J. Huber *et al.*, "Robust estimation of a location parameter," *The annals of mathematical statistics*, vol. 35, no. 1, pp. 73–101, 1964.
- [22] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.
- [23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.
- [24] J. Ku, A. Harakeh, and S. L. Waslander, "In defense of classical image processing: Fast depth completion on the cpu," *arXiv preprint arXiv:1802.00036*, 2018.
- [25] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition."

PLACE
PHOTO
HERE



Abdelrahman Eldesokey is a Ph.D. student at the Computer Vision Laboratory, Linköping University, Sweden. . He received his M.Sc. degree in Informatics from Nile University, Egypt in 2016. He also became a Wallenberg AI, Autonomous Systems and Software Program (WASP) affiliated student in 2017. His research interests include deep learning for computer vision with a focus on sparse data.



Fahad Shahbaz Khan is an Associate Professor (Universitetslektor and Docent) at Computer Vision Laboratory, Linköping University, Sweden and Lead Scientist at Inception Institute of Artificial Intelligence, UAE. He received the M.Sc. degree in Intelligent Systems Design from Chalmers University of Technology, Sweden and a Ph.D. degree in Computer Vision from Autonomous University of Barcelona, Spain. From 2012 to 2014, he was post doctoral fellow at Computer Vision Laboratory, Linköping University, Sweden. From 2014 to 2018, he was research fellow at Computer Vision Laboratory, Linköping University, Sweden. His research interests are in object recognition, action recognition and visual tracking. He has published articles in high-impact computer vision journals and conferences in these areas.