Depth Coefficients for Depth Completion

Saif Imran Yunfei Long Xiaoming Liu Daniel Morris
Michigan State University
428 S. Shaw Ln, EB 2120, East Lansing, MI 48824
{imransai,longyunf,liuxm,dmorris}@msu.edu

Abstract

Depth completion involves estimating a dense depth image from sparse depth measurements, often guided by a color image. While linear upsampling is straight forward, it results in artifacts including depth pixels being interpolated in empty space across discontinuities between objects. Current methods use deep networks to upsample and "complete" the missing depth pixels. Nevertheless, depth smearing between objects remains a challenge. We propose a new representation for depth called Depth Coefficients (DC) to address this problem. It enables convolutions to more easily avoid inter-object depth mixing. We also show that the standard Mean Squared Error (MSE) loss function can promote depth mixing, and thus propose instead to use cross-entropy loss for DC. With quantitative and qualitative evaluation on benchmarks, we show that switching out sparse depth input and MSE loss with our DC representation and cross-entropy loss is a simple way to improve depth completion performance, and reduce pixel depth mixing, which leads to improved depth-based object detection.

1. Introduction

Active depth sensing has achieved significant gains in performance and demonstrated its utility in numerous applications over the last two decades. High-resolution depth estimation contributes towards 3D scene understanding [28], object detection [25], classification and tracking [9], and object shape estimation [6]. Important sensors include Intel RealSense and Microsoft Kinect 2 for indoor applications, and Lidars such as the Velodyne VLP-64 for longer-range outdoor applications such as automotive safety and autonomy. While performance in range and resolution are improving, the cost for higher-resolution Lidars remains prohibitive for numerous applications. As a result there is significant ongoing effort into improving the resolution, while lowering the cost of 3D sensors [33, 24, 14].

Our application goal is high-resolution shape analysis and object detection using an inexpensive, low-resolution Lidar, complemented with a high-resolution camera. The key step of using a color image to guide depth super-resolution is called *depth completion*. Current state-of-the-art methods [20, 4, 19], rely on deep convolutional networks and perform well at up-sampling within-object depths.

While these methods score well on Root-Mean-Square

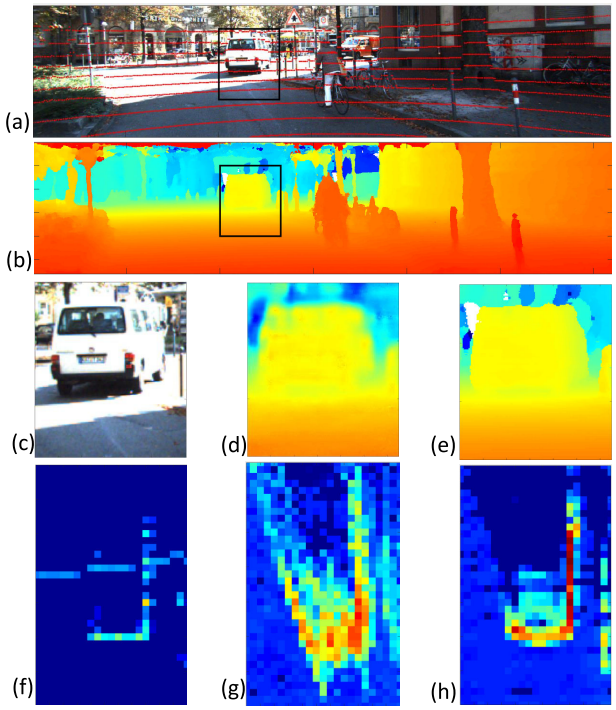


Figure 1: Our depth completion uses (a) a color image and the subsampled (16-row) Lidar points projected into image plane to estimate (b), a dense depth image. (c-e) are zoomed-in view of input color image, super-resolved depth of Ma et al. [19] and ours respectively. (f-h) are bird's eye view of input sparse Lidar data, (d), and (e), respectively. Colors in the bird's eye view show the number of height pixels in each cell/pixel. So a smeared object shape has height pixels spread out around the object boundary. Notice the smearing of depth at the object boundaries in (g) compared to (h). These depth-mixing pixels impact qualitative appearance as well as subsequent tasks, such as object detection and pose estimation.

Error (RMSE), nevertheless they still generate *mixed-depth pixels*. We define mixed-depth pixels as those pixels whose estimated depth places them at neither the foreground nor background object, but in-between the objects. Since mixed-depth pixels occur primarily at depth discontinuities, they typically constitute a small fraction of the total pixel count and various loss measures. Nevertheless, their impact on the quality of depth maps and projected point-clouds is significant including spurious points in mid-air and connecting surfaces between separate objects.

This paper aims to investigate the cause of the mixed-depth pixels and propose a solution. We investigate how current depth-image representation leads to depth mixing, and propose an alternative representation called Depth Coefficients (DC) to avoid this. We also examine how loss functions, such as MSE, favor mixed-depth pixels in certain cases. Using our newly proposed DC representation, we leverage cross-entropy loss to avoid promoting depth mixing. Finally we propose a pair of evaluation metrics that can be used in place of, or to complement, RMSE and MAE. Unlike RMSE and MAE, our new metrics penalize mixed-depth pixels and so may be better quality measures for evaluating depth completion. Sample output is shown in Fig. 1.

The contributions of this paper are: (1) an analysis of the cause of mixed-depth pixels, (2) a novel depth representation that reduces depth mixing, (3) a new use of cross entropy as a depth loss function, (4) two evaluation metrics that penalize mixed-depth pixels, and (5) demonstration of improved object detection from super-resolved depth.

2. Related Work

Depth completion The substantially lower resolution of depth sensors compared to color cameras has been a motivator for depth completion. Early work by Diebel and Thrun [7] used markov random fields to guide upsampling, and this was followed by a variety improvements including bilateral filters [34], robust regularization [24], hand-crafted filters [8] and image segmentation [18]. More recently deep convolutional neural networks (CNNs) have taken the lead and moved on from the Middlebury dataset [27] to larger datasets, NYU2 [23] and KITTI [10]. Leading contenders, including [20, 4, 19], perform well on these datasets with both regular and irregularly sampled data. Our work uses a similar network as [19], but with focus on the depth representation and loss function, instead of the architecture.

Depth representation Measurements of 3D shape can be represented multiple ways, each with its own advantages and drawbacks. 3*D point clouds* are widely used in object detection [25], segmentation [3] and surface normal estimation [22]. Their advantages include being precise, straight from sensors, and that euclidean distances can be calculated between point cloud clusters. However, direct convolutions are not possible with point clouds; object surfaces are not fully represented by point clouds since they are sparse and unorganized. *Voxels* can provide a regular grid for object detection [35], object classification and orientation estimation [26], but can be memory intensive at high resolutions. *Depth images*, sometimes considered 2.5D representations, have been used for RGBD fusion and instance segmentation [30, 13]. They naturally encode sensor viewing rays and adjacency between points. They are compact representations and with their regular grids can be processed with CNN in an analogous way to color image super-resolution [31, 32]. This is the representation of choice for colorization techniques and fusion [23] as well as depth completion.

While depth images are popular for depth completion, we will examine an important drawback: the tendency to generate mixed-depth pixels between surfaces. One goal of our DC representation is to remedy this drawback.

Loss function for depth completion A key component of depth completion is the choice of loss function. Recent work has explored loss functions including L2 [4, 19], L1 [20], inverse-L1 [14], and softmax losses on depth [16]. While these loss functions can achieve low error on measures including RMSE, MAE, iMAE, often it comes at the cost of smoothing out depth estimate at object boundaries. In this way, the sharp boundaries are lost/smeared and object shapes are distorted. We propose to impose cross-entropy on our probabilistic representation, and show this gives both high performance and sharp boundaries.

3. Mixed-depth Pixels

Depth completion aims to estimate unknown depths at image pixels using surrounding depth pixels plus the color image. This is particularly challenging at depth discontinuities; here a foreground object occludes the background and the unknown pixel depths typically belong to either the foreground or background (see Fig. 2). In this section we consider three contributing factors to depth mixing: depth ambiguity, the loss function, and depth representation.

3.1. Depth Ambiguity

Depth completion often faces an ambiguity problem for pixels at object boundaries: do these pixels belong to the foreground or background object. To address this ambiguity, we first define the learning task: find model parameters θ that best predict depth d_i, given sparse depth and color image data x_i, with distribution p_{data}:

$$\hat{\theta} = \arg\max_{\theta} \mathbb{E}_{p_{data}} \left[\log p_{model}(d_i | x_i; \theta) \right]. \qquad (1)$$

Here the expectation is performed over training data with distribution p_{data}. The term p_{model} is a probabilistic model for how well depth d_i is predicted by data x_i.

Initially let's consider that the image data x_i consist only of sparse depth values and no color images. Given

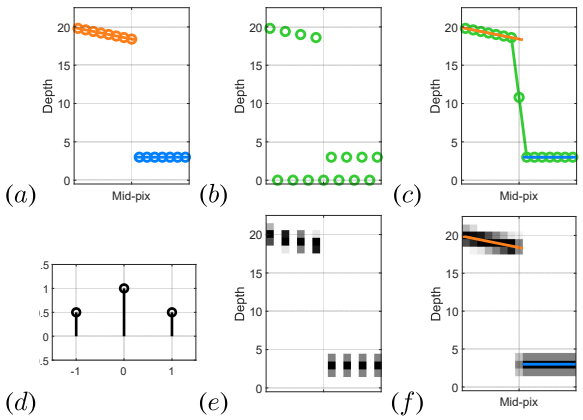


Figure 2: An example of depth mixing, and how DC avoids it. (a) A slice through a depth image showing a depth discontinuity between two objects. (b) An example sparse depth representation: each pixel either has a depth value or a zero. (c) The result of a 1D convolution, shown in (d), applied to the sparse depth. This estimates the missing pixel, but generates a mixed-depth pixel between the two objects. (e) A DC representation of the sparse depth. Each pixel with a depth measurement has three non-negative coefficients that sum to 1 (shown column-wise). (f) The result of applying the same filter (d) to DC in (e). Missing depths are interpolated and notably there is no depth mixing between the objects.

a sparsely sampled object, we can ask whether a pixel near a depth discontinuity boundary belongs to the foreground or background. If the boundary is unknown, it will be ambiguous whether it is foreground or background. What this means in terms of Eq. 1 is that given the same data x_i, there are at least two compatible depths: $d^{(1)}$ for foreground and $d^{(2)}$ for background. If a color image is available, it may be possible to exactly infer the boundary between objects, and resolve this ambiguity. However, often this is not the case; the boundary is not clear and hence the ambiguity persists. In this paper we show that how this ambiguity is addressed has important implications for depth estimation.

3.2. Depth Loss Functions with Ambiguity

One of the more popular loss functions is Mean Squared Error (MSE). In part this is because the MSE gives the maximum likelihood solution to Eq. 1 when $p_{model}(d_i|x_i;\theta)$ is Gaussian. Here we consider the implications of using MSE when there are depth ambiguities. First we address explicit ambiguities: there are two examples of data x_i, one having depth $d^{(1)}$ and the other $d^{(2)}$. The MSE loss is:

$$\text{MSE}(d) = \frac{1}{2}\left(||d - d^{(1)}||^2 + ||d - d^{(2)}||^2\right), \quad (2)$$

which is minimum when

$$\hat{d} = \frac{1}{2}(d^{(1)} + d^{(2)}). \quad (3)$$

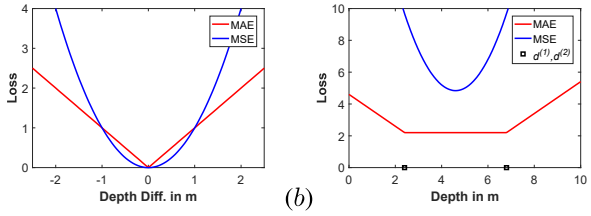


Figure 3: (a) shows MSE and MAE loss functions. These perform an expectation over the probability of the data. Now consider an ambiguous case where a pixel's depth has equal probability being $d^{(1)}$ or $d^{(2)}$, shown as black squares in (b). Minimum MSE estimate, $\hat{d}$, is the mid-point, while MAE has equal loss for all points between these two depths. This illustrates why MSE prefers mixed-depth pixels, and MAE fails to penalize them.

And so the estimated point is a mixed-depth pixel falling half-way between the foreground and background objects. An illustration of this is in Fig. 3(b).

The same issue can occur even without explicit ambiguities. Assume we have a perfectly trained model that gives minimum MSE, and there are ambiguous situations as defined in Sec. 3.1. Then as shown above, the minimum MSE solution will be for the model to predict mixed-depth pixels.

Mean Absolute Error (MAE), has a similar issue, yet not as severe. As in Fig. 3, in the pairwise ambiguity case, the MAE loss of mixed-depth pixels is equal to the loss at the actual values. Thus while MAE loss does not prefer mixed-depth pixels like MSE, nevertheless mixed-depth solutions may not be sufficiently penalized to avoid them.

3.3. Depth Evaluation

RMSE [1] and MAE are used to evaluate depth completion. This is concerning if there are depth ambiguities, since RMSE favors solutions with mixed-depth pixels. MAE, while not favoring these, may only penalize them weakly. Thus we need alternative evaluation metrics that properly penalize mixed-depth pixels.

3.4. Depth Representation

An important factor impacting depth mixing, is how depth is represented when input into a depth completion CNN. The usual representation, which we call a sparse depth image, is to project known depth points into the image plane, setting the pixel value at that location to the depth, and setting the remaining pixels to zero. The CNN applies a sequence of convolutions and non-linear operations to the sparse depth image eventually predicting a dense depth image. To understand the tendency of CNNs to generate mixed depths, consider a slice through a simple two-object scene illustrated in Fig. 2(a-d). Applying a smoothing convolution

[1] RMSE has the same parametric minimum as MSE.

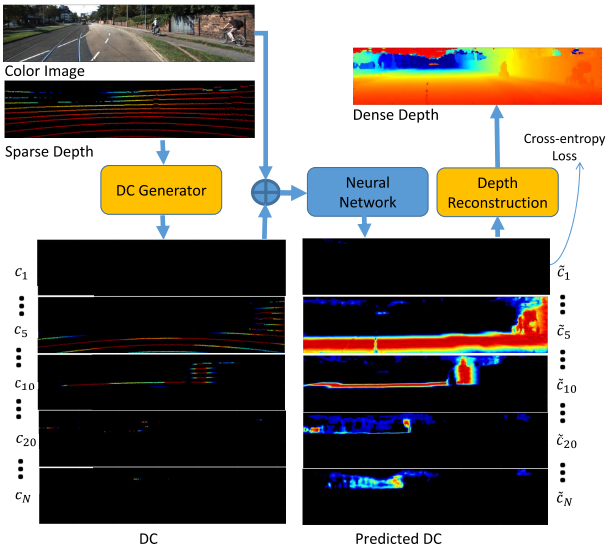


Figure 4: An overview of our method. Sparse depth is converted into Depth Coefficients with multiple channels, each channel holding information of a certain depth range. This, along with color, is input to the neural network. The output is a multi-channel dense depth density that is optimized using cross entropy with a ground-truth DC. The final depth is reconstructed based on the predicted density.

from (d) generates the dense depth estimate in (c). Notice the predicted center pixel depth is an intermediate of the two object depths – an example of depth mixing creating a spurious point in empty space.

To avoid depth mixing, the mid-pixel in Fig. 2(c) should be predicted purely from the (foreground or background) object to which it belongs. In 1D it is simple to find convolutions that avoid depth mixing, e.g., averaging to the right or left. But for 2D depth images with unevenly distributed measurements, it is much more complicated to avoid mixing, as depth boundaries can have many shapes with respect to known depth pixels. At the very least, doing so requires learning a complex network. The simple alternative, presented next, is to use a representation where convolutions can directly generate hypotheses without depth mixing.

4. Methodology

This section proposes a new representation for depth that can be used by broad class of CNN architectures for depth completion. The required modifications to a standard depth-completion CNN are small: just the input, the output and the loss function. Using this depth representation has potential to overcome the depth-mixing issues described in Sec. 3. Fig. 4 shows the overview of our approach, and how the input and output of a CNN are reinterpreted.

4.1. Depth Coefficients

We seek a depth representation that not only can easily avoid pixel depth mixing, but also enables interpolation between depths within objects. One solution is to use a discrete one-hot depth representation. Despite enabling convolutions without depth mixing, its drawback is the trade-off between a loss in depth accuracy and a very large number of channels to represent depth. Here we propose an alternative that has the benefits of one-hot encoding, but requires far fewer channels and eliminates the accuracy loss issue.

To represent a dense or sparse depth image we create a multi-channel image of the same size, with each channel representing a fixed depth, $D = \{D_1, \ldots, D_N\}$. The depth values increase in even steps of size b. In choosing the number of channels (or bins) we trade-off memory vs. precision. For our applications, we chose 80 bins to span the full depth being modeled, and this determines the bin width, b. Thus each pixel i has a vector of values, $\boldsymbol{c}_i = \{c_{i1}, \ldots, c_{iN}\}$, which we call *Depth Coefficients* (DC), that represents its depth, d_i. Note that multi-channel representations have been used before [29] for intensity images. But one of our novelty lies in applying the discrete-channel representations for depth images so that the continuity of depth values are preserved. The following describes the DC representation in more detail.

We constrain these coefficients to be non-negative, sum to 1, and give the depth as the inner product with the channel depths:

$$d_i = \sum_j c_{ij} D_j. \tag{4}$$

Note this representation is not unique as many combinations of coefficients may produce the same depth.

So we use the following simple, sparse representation with three non-zero coefficients to represent depth. Let k be the index of the depth channel closest to pixel depth d_i and $\delta = \frac{d_i - D_k}{b}$. Since b is the spacing between adjacent bin depths, the D_{k-1} and D_{k+1} bins can be expressed in terms of the center bin as $D_{k-1} = D_k - b$ and $D_{k+1} = D_k + b$ respectively. With this substitution all the terms cancel on the right-hand side of Eq. 5 leaving d_i. The DC vector for pixel i is:

$$\boldsymbol{c}_i = (0, \ldots, 0, \frac{0.5 - \delta}{2}, 0.5, \frac{0.5 + \delta}{2}, 0, \ldots, 0), \tag{5}$$

where three non-zero terms are $(c_{i(k-1)}, c_{ik}, c_{i(k+1)})$. This is unique for each d_i, satisfies Eq. 4, and sums to 1.

More than representing a continuous value as weighted sum of discrete bins [15], we claim that using DC to represent depth provides a much simpler way for CNNs to avoid depth mixing. The first step of a CNN is typically an image convolution with N_{in} input channels. For sparse depth input, $N_{in} = 1$, and so all convolutions apply equally to all depths, resulting mixing right from the start. For DC input,

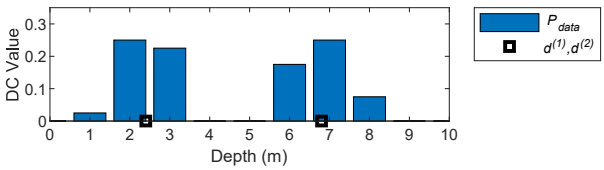


Figure 5: An illustration of P_{data} modeled as the sum of the DC of the two points from Fig. 3. The estimated $\hat{c}_{ij}$ with minimum cross-entropy loss, Eq. 6, will exactly match P_{data}, providing a multi-modal density. A pixel depth estimate using Eq. 7 will find the depth of one of the peaks, and not a mixed-depth value.

depths are divided over $N_{in} = N$ input channels, resulting in two important capabilities. First, CNNs can learn to avoid mixing depths in different channels as needed. This is similar to voxel-based convolutions [12, 21] which avoid mixing spatially-distant voxels. This effect is illustrated in Fig. 2(e-f), where a multi-channel input representation, (e), allows convolutions to avoid mixing widely spaced depths. Second, since convolutions apply to all channels simultaneously, depth dependencies, like occlusion effects, can be modeled and learned by neural networks.

4.2. Loss Function

As shown in Sec. 3, MSE leads to depth mixing when there is depth ambiguity. One way to avoid this is, rather than estimate depth directly, to estimate a more general probabilistic representation of depth. Now DC can provide a probabilistic depth model, both for p_{data} and p_{model} in Eq. 1. Minimizing the cross entropy of the predicted output $\tilde{\boldsymbol{c}}$, representing $p_{data}(\tilde{d}_i|x_i;\theta)$, is equivalent to minimizing the KL divergence with $\boldsymbol{c}$. In this way, we can learn to estimate $p_{model}(d_i|x_i;\theta)$ parameterized with DC. Our cross-entropy loss for pixel i is defined as:

$$L_i^{ce}(c_{ij}) = -\sum_{j=1}^{N} c_{ij} \log \tilde{c}_{ij}, \tag{6}$$

where c_{ij} terms are the DC elements of the ground truth obtained using Eq. 5. Training a network to predict $\tilde{c}_{ij}$ that minimizes L_i^{ce} is equivalent to maximizing Eq. 1.

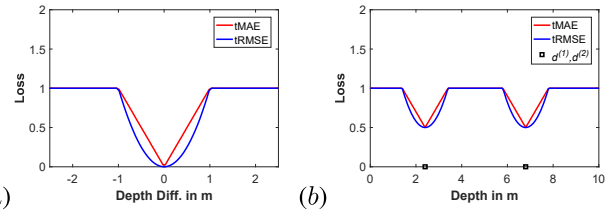
Use of cross-entropy loss has two main advantages. The first is that depth ambiguities no longer result in a preference for mixed-depth pixels. As illustrated in Fig. 5, DC models multi-modal densities, and as we show in the next section our depth estimate will find the location of the maximum peak at one of the depths. Second, optimizing cross entropy leads to much faster convergence than MSE, which suffers from gradients going to zero near the solution.

4.3. Depth Reconstruction

There are a number of options for depth reconstruction. We can use Eq. 4, and substitute $\hat{c}_{ij}$ for c_{ij} for pixel i. However, the predicted coefficients may be multi-modal as in Fig. 5, and it may be preferable to estimate the maximum likelihood solution. We can estimate the depth for the peak via the maximum coefficient $c_{ik} \in \boldsymbol{c}_i$ and its two neighbors:

$$\hat{d}_i = \frac{\hat{c}_{i(k-1)} D_{(k-1)} + \hat{c}_{ik} D_k + \hat{c}_{i(k+1)} D_{(k+1)}}{\hat{c}_{i(k-1)} + \hat{c}_{ik} + \hat{c}_{i(k+1)}}. \tag{7}$$

A third way is for the network to directly predict depth in addition to DC.



(a) (b)

Figure 6: (a) tRMSE and tMAE with threshold $t = 1$, compare to MSE and MAE in Fig. 3(a). (b) When there is a depth ambiguity of at least t, in each case the minima will be at the true depth and the ambiguous depth, while the mixed-depth region between them will be penalized equally to other large-error regions.

4.4. New Evaluation Metrics

While RMSE and MAE are useful metrics for overall depth completion performance, we showed in Sec. 3 that these encourage, or at least do not sufficiently penalize, depth mixing. Thus we propose two complementary metrics that focus on depth surface accuracy and penalize depth mixing equally to other large errors. These metrics are Root Mean Squared Thresholded Error (tRMSE) and Mean Absolute Thresholded Error (tMAE), defined as follows:

$$\text{tRMSE} = \sqrt{\sum_{i=1}^{P} \frac{\min((\tilde{y}_i - \hat{y}_i)^2, t^2)}{P}}, \tag{8}$$

$$\text{tMAE} = \sum_{i=1}^{P} \frac{\min(|\tilde{y}_i - \hat{y}_i|, t)}{P}. \tag{9}$$

Here P is the number of pixels, t the threshold distance distinguishing within-surface variation from inter-object separation, $\tilde{y}_i$ the ground-truth value and $\hat{y}_i$ the estimated value.

4.5. Uses of Depth Completion

Ultimately the choice of algorithm and evaluation metric should depend on the use of depth completion. We identify two uses where depth mixing can have a significant impact. The first is to create dense, pixel-colored, 3D environment models from lower-resolution depth sensors. Now mixed-depth pixels occurring in empty space between objects, illustrated in Fig. 3, can create connecting surfaces and negatively impact the visual quality of the 3D environment.

Method	RMSE	MAE	REL	tMAE	tRMSE	δ_1	δ_2	δ_3	δ_4	δ_5
Ma [20]	0.236	0.13	0.046	0.068	0.075	52.3	82.3	92.6	97.1	99.4
Bilateral [2]	0.479	-	0.084	-	-	29.9	58.0	77.3	92.4	97.6
SPN [17]	0.172	-	0.031	-	-	61.1	84.9	93.5	98.3	99.7
Unet [5]	0.137	0.051	0.020	-	-	78.1	91.6	96.2	98.9	99.8
CSPN [5]	0.162	-	0.028	-	-	64.6	87.7	94.9	98.6	99.7
CSPN+UNet [5]	**0.117**	-	0.016	-	-	83.2	93.4	97.1	99.2	**99.9**
Ours-all	0.118	**0.038**	**0.013**	0.042	**0.053**	86.3	95.0	97.8	**99.4**	**99.9**
Ours-3coeff	0.131	**0.038**	**0.013**	**0.040**	0.054	**86.8**	**95.4**	**97.9**	99.3	99.8

Table 1: Quantitative results of NYU2 (Done on Uniform-500 Samples + RGB) (units in m).

Another use of depth completion is for data fusion and subsequent tasks such as object detection. If effective, this could enable low-cost, low-resolution sensors to be upgraded when combined with a color camera. We compare object detection performance with super-resolved depths both with and without our contribution.

4.6. Architectures

We selected a standard network for depth completion [19], and modified the input and output. On the input, $80/48$ channels of depth/color respectively were fed into the initial convolutions and then concatenated for further propagation into the network. On the output, 80 channels are predicted (rather than a single channel) using a 1×1 convolution. This output is trained using cross entropy loss on a DC representation of semi-dense depth. Using s similar strategy, other depth completion networks can also leverage the advantages of DC.

5. Experiments

5.1. Experimental Protocols

We evaluate DC representation by means of two publicly available datasets: KITTI (outdoor scenes) and NYU2 (indoor scenes) respectively to demonstrate the performance of our algorithm. We use KITTI depth completion dataset [33] for both training and testing. The dataset is created by aggregating Lidar scans from 11 consecutive frames into one, producing a semi-dense ground truth with roughly 30% annotated pixels. The dataset consists of $85,898$ training data, $1,000$ selected validation data, and $1,000$ test data without ground truth. We truncate the top 90 rows of the image during training since it contains no Lidar measurements.

The NYU-Depth v2 dataset consists of RGB and depth images collected from 464 different scenes. We use the official split of data, where 249 scenes are used for training and we sample 50K images out of the training similar to [20]. For testing, the standard labelled set of 654 images is used. The original image size is first downsampled to half, and then center-cropped, producing a network input

Method	MAE	RMSE	iMAE	iRMSE	tMAE	tRMSE
Ma [19]	65.2	174.3	-	-	59.4	69.5
DC-all	38.6	**142.3**	1.55	**2.23**	36.1	50.9
DC-3coeff	**37.8**	160.6	**1.53**	2.41	**33.4**	**47.2**

Table 2: Depth completion results on KITTI validation benchmark with 16-row Lidar input (units cm).

spatial dimension of 304×208. For comparison purposes, we choose the state of the arts in both outdoor [19] and indoor scenes [20, 5] using RGBD depth sensors.

Sub-Sampling Depth completion on uniformly subsampling tends to be easier than irregular subsampling; Ma *et al.* [19] reported improved performance with uniform subsampling. But in real scenarios, sparse sensors such as Lidar often generate non-uniform, structured patterns when projected into the image plane. Since our application is to estimate dense depth from inexpensive Lidars in outdoor scenes, we simulate lower resolution Lidars by subsampling 32 and 16 rows from 64R Lidar (depth acquisition sensor used by KITTI). We subsample the points based on selecting a subset of evenly spaced rows of 64R raw data provided by KITTI (splitted based on the azimuth angle in Lidar space) and then projecting the points into the image.

Error Metrics We use the standard error metrics: RMSE, MAE, Mean Absolute Relative Error (MRE), and δ_i. δ_i is the percentage of predicted pixels whose relative error is within a relative threshold (higher being better), defined as:

$$\delta_i : \frac{\mathbf{card} \max(\frac{\hat{y}_i}{y_i}, \frac{y_i}{\hat{y}_i}) < \delta_i}{\mathbf{card}(\{y_i\})} \tag{10}$$

We also include our proposed metrics: tMAE, and tRMSE.

Implementation Details The experiment is implemented in Tensorflow 1.10 [1]. We use Adam optimizer for training with an initial learning rate of 10^{-4} and decreased to half every 5 epochs. We use a single GPU 1080 Ti with 32G RAM for training and evaluation. Since GPU memory does not support full-sized KITTI images, we train it with patches of size 224×224 and batch-size of 3. For NYU2

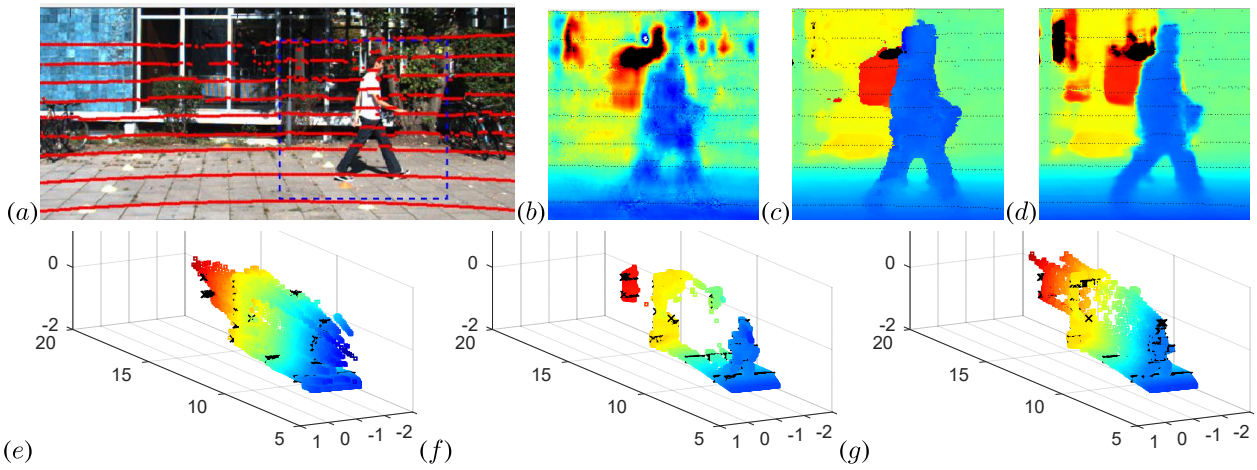


Figure 7: Depth completion with 16-row Lidar. (a) scene, (b, e) show Ma *et al.* [19] with significant mixed pixels. (c, f) show our 3-coefficient estimation, demonstrating very little depth mixing. (d, g) show our estimation with all coefficients.

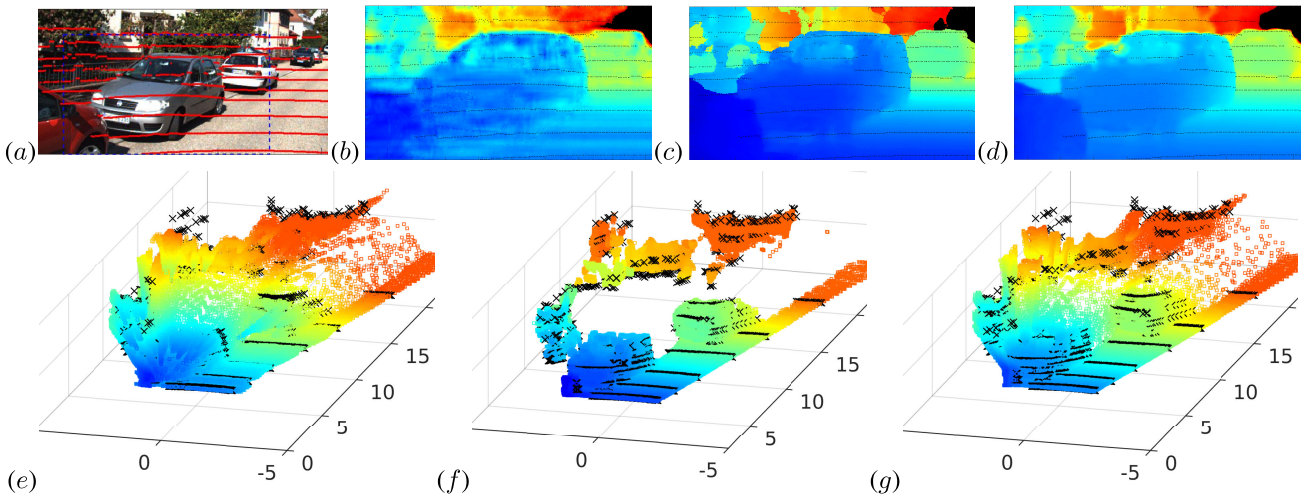


Figure 8: Another depth completion example with 16-row Lidar, where all subfigures are defined the same as Fig. 7. Interestingly, higher RMSE is reported on 3-coefficient estimation as opposed to all-coefficient estimation.

dataset, we select batch-size of 10 and continue the training for $7 - 10$ epochs.

5.2. Results

Table 1 shows a comparison on the NYU2 dataset. Our method shows improvements in all metrics except RMSE. And interestingly, unlike CSPN + Unet [5], our method requires no fine-tuning networks (which increases the inference time) to sharpen boundaries.

Table 2 reports a quantitative comparison of our method with our implementation of Ma *et al.* [19] performing depth completion on 16-row Lidar. We select 16-row Lidar since they are inexpensive and commercially feasible for automakers. Results are on KITTI's validation set of $1,000$ images. Due to GPU memory constraints, our patch sizes and batch sizes were smaller and so our implementation performance is lower than in [19]. However, from a comparison perspective, our network architecture is the same (more discussion in sec 4.6), and so the improved results of our method are due to using DC input and cross-entropy loss on the output.

5.3. Ablation Studies

Table 3 shows depth completion performance as Lidar sparsity increases. At each sparsity level results are shown for depth estimated from 3 coefficients, see Eq. 7, and all coefficients, Eq. 4. While they both have roughly the same MAE, 3-coefficient prediction has smaller tMAE and

Sparsity	MAE	RMSE	tMAE	tRMSE
64R-3coeff	24.1	121.2	20.3	34.4
64R-all	25.2	106.1	23.9	37.4
32R-3coeff	31.0	132.2	24.4	39.5
32R-all	31.1	115.8	27.6	42.2
16R-3coeff	37.8	160.6	33.4	47.2
16R-all	38.6	142.3	36.1	50.5

Table 3: Performance evaluation at different levels of Lidar sparsity (KITTI dataset). 64R, 32R and 16R refers to 64-row, 32-row, 16-row respectively. Units in cm.

Input	Loss	MAE	RMSE	tMAE	tRMSE
SP	MSE	6.63	15.28	5.96	6.97
DC	MSE	6.10	15.32	5.72	6.73
SP	CE	9.53	17.81	6.75	7.56
DC	CE	**3.82**	**11.85**	**4.24**	**5.37**

Table 4: A comparison whether DC on the input or DC with cross entropy (CE) on output has the dominant effect. It turns out that individually their effect is small, but together have a large impact (NYU2 dataset). Units in cm.

tRMSE but larger RMSE. Likely this is due to fewer mixed-depth pixels, as can be seen in Figs. 7 and 8.

One interesting question is whether the gains we are seeing are coming from use of DC on the input or use of DC plus cross-entropy on the output. Table 4 compares all four combinations of inputs and outputs and finds that by far the biggest gains are when DC is used in both input and output. We ablate on how the number of DC channels affects efficiency, in Tab. 5. In each of the variation, we create the DC from 2.5D depth and recover the 2.5D depth from DC on-the-fly. There is some computational penalty to DC, but it is relatively small, and can be remedied by reducing the number of channels.

Another application of depth completion is to improve on object detection. While it might seem intuitive that at higher resolution, estimated dense depth could give better vehicle detection, often this is not the case, and we are not aware of other past literature reporting this. Likely mixed-depth pixels have a large negative impact on object detection. Indeed, Tab. 6 shows worse car detection on Ma's output than on the raw 16-row sparse data. However, our method is able to outperform sparse depth, an important step towards improving Lidar-based object detection.

6. Conclusion

Upsampling depth in a manner that respects object boundaries is challenging. Deep networks have shown progress in achieving this, but nevertheless still generate mixed-depth pixels. Our work tackles this problem on both the input and the output sides of a prediction network. On the input, our Depth Coefficients represent depth without loss in accuracy (unlike binning) while separating pixels by depth so that it is simple for convolutions to avoid depth mixing. On the output side, instead of directly predicting depth, we predict a depth density using cross entropy on the Depth Coefficients. This is a richer representation that avoids depth mixing and can enable deeper levels of fusion and object detection. Indeed we show that, unlike other upsampling methods, our dense depth estimates can improve object detection compared to sparse depth. Including Depth Coefficients on the input and output of networks is an easy and simple way to achieve better performance.

We showed that in the case of ambiguities, MSE is a flawed metric to evaluate depth completion. Now that depth completion methods are producing high-quality dense depths, our proposed metrics, tRMSE and tMAE, are preferable as they reward high-probable depth estimates and give equal penalty to large errors, which are mostly mixed-depth pixels.

Method	MAE (cm)	RMSE (cm)	Infer. time (ms)
2.5D (SP)-1C.	65.2	174	130
DC-10C-3coeff	50.2	169	140
DC-20C-3coeff	45.4	165	145
DC-40C-3coeff	37.8	161	161
DC-80C-3coeff	38.4	167	202

Table 5: Performance and efficiency on KITTI validation benchmark using 16R Lidar points.

Upsample:	3D Bounding Box			Bird's Eye View Box		
	Easy	Med.	Hard	Easy	Med.	Hard
Raw 16R	54.4	36.2	31.3	73.6	**58.1**	**50.4**
Ma [19]	36.7	23.0	18.5	56.2	33.8	29.7
DC-3coeff	**64.9**	**41.9**	**34.7**	**78.1**	54.0	45.6

Table 6: Average precision (%) for 3D detection and pose estimation of cars on KITTI [11] using Frustum PointNet [25]. The baseline, Raw-16R, uses 16 rows from the Lidar, while Ma's method [19] and our method start by densely upsampling these 16-row data. In each case, the method is trained on $3,712$ frames and evaluated on $3,769$ frames, of the KITTI 3D object detection benchmark [11] using an intersection of union (IOU) measure of 0.7. Only our method improves on the baseline, and this is the most significant for 3D bounding boxes.

Acknowledgements

This work was partially supported with funds from Changan US, and the authors greatly appreciate inputs from Radovan Miucic, Ashish Sheikh and Gerti Tuzi.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org. 6

[2] J. T. Barron and B. Poole. The Fast Bilateral Solver. In *Proc. European Conf. Computer Vision (ECCV)*, pages 617–632, 2016. 6

[3] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and Recognition using Structure from Motion Point Clouds. In *Proc. European Conf. Computer Vision (ECCV)*, pages 44–57. Springer, 2008. 2

[4] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich. Estimating Depth from RGB and Sparse Sensing. In *Proc. European Conf. Computer Vision (ECCV)*, pages 167–182, 2018. 1, 2

[5] X. Cheng, P. Wang, and R. Yang. Depth Estimation via Affinity Learned with Convolutional Spatial Propagation Network. In *Proc. European Conf. Computer Vision (ECCV)*, pages 108–125, 2018. 6, 7

[6] Y. Cui, S. Schuon, S. Thrun, D. Stricker, and C. Theobalt. Algorithms for 3D Shape Scanning with a Depth Camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(5):1039–1050, 2013. 1

[7] J. Diebel and S. Thrun. An Application of Markov Random Fields to Range Sensing. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 291–298. MIT Press, 2006. 2

[8] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof. Image Guided Depth Upsampling using Anisotropic Total Generalized Variation. In *Proc. Int. Conf. Computer Vision (ICCV)*, pages 993–1000, 2013. 2

[9] B. Fortin, R. Lherbier, and J. Noyer. A Model-Based Joint Detection and Tracking Approach for Multi-Vehicle Tracking with LiDAR Sensor. *IEEE Trans. on Intelligent Transportation Systems*, 16(4):1883–1895, Aug 2015. 1

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets Robotics: The Kitti Dataset. *Int. J. Robot. Research*, 32(11):1231–1237, 2013. 2

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 8

[12] S. Ghadai, X. Lee, A. Balu, S. Sarkar, and A. Krishnamurthy. Multi-Resolution 3D Convolutional Neural Networks for Object Recognition. *arXiv preprint arXiv:1805.12254*, 2018. 5

[13] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *Proc. European Conf. Computer Vision (ECCV)*, pages 345–360. Springer, 2014. 2

[14] M. Jaritz, R. De Charette, E. Wirbel, X. Perrotton, and F. Nashashibi. Sparse and Dense Data with CNNs: Depth Completion and Semantic Segmentation. In *Int. Conf. 3D Vision (3DV)*, pages 52–60, 2018. 1, 2

[15] L. Ladickỳ, B. Zeisl, and M. Pollefeys. Discriminatively Trained Dense Surface Normal Estimation. In *Proc. European Conf. Computer Vision (ECCV)*, volume 2, page 4, 2014. 4

[16] Y. Liao, L. Huang, Y. Wang, S. Kodagoda, Y. Yu, and Y. Liu. Parse Geometry from a Line: Monocular Depth Estimation with Partial Laser Observation. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 5059–5066. IEEE, 2017. 2

[17] R. Liu, G. Zhong, J. Cao, Z. Lin, S. Shan, and Z. Luo. Learning to Diffuse: A New Perspective to Design PDEs for Visual Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(12):2457–2471, 2016. 6

[18] J. Lu and D. Forsyth. Sparse Depth Super Resolution. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2245–2253, 2015. 2

[19] F. Ma, G. V. Cavalheiro, and S. Karaman. Self-supervised Sparse-to-Dense: Self-supervised Depth Completion from LiDAR and Monocular Camera. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019. 1, 2, 6, 7, 8

[20] F. Ma and S. Karaman. Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. 1, 2, 6

[21] D. Maturana and S. Scherer. Voxnet: A 3D Convolutional Neural Network for Real-time Object Recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 5

[22] N. J. Mitra and A. Nguyen. Estimating Surface Normals in Noisy Point Cloud Data. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 322–328. ACM, 2003. 2

[23] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor Segmentation and Support Inference from RGB-D Images. In *Proc. European Conf. Computer Vision (ECCV)*, pages 746–760, 2012. 2

[24] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon. High Quality Depth Map Upsampling for 3D-TOF Cameras. In *Proc. Int. Conf. Computer Vision (ICCV)*, pages 1623–1630, Nov 2011. 1, 2

[25] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 918–927, 2018. 1, 2, 8

[26] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning Deep 3D Representations at High Resolutions. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 3577–3586, 2017. 2

[27] D. Scharstein and R. Szeliski. High-Accuracy Stereo Depth Maps using Structured Light. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 1, June 2003. 2

[28] B. Schwarz. LiDAR: Mapping the World in 3D. *Nature Photonics*, 4(7):429, 2010. 1

[29] L. Sevilla-Lara and E. Learned-Miller. Distribution Fields for Tracking. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1910–1917. IEEE, 2012. 4

[30] L. Shao, Y. Tian, and J. Bohg. Clusternet: Instance Segmentation in RGB-D Images. *arXiv preprint arXiv:1807.08894*, 2018. 2

[31] Y. Tai, J. Yang, and X. Liu. Image Super-Resolution via Deep Recursive Residual Network. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 3147–3155, 2017. 2

[32] Y. Tai, J. Yang, X. Liu, and C. Xu. Memnet: A Persistent Memory Network for Image Restoration. In *Proc. Int. Conf. Computer Vision (ICCV)*, pages 4539–4547, 2017. 2

[33] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity Invariant CNNs. In *Int. Conf. 3D Vision (3DV)*, pages 11–20, 2017. 1, 6

[34] Q. Yang, R. Yang, J. Davis, and D. Nister. Spatial-Depth Super Resolution for Range Images. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007. 2

[35] Y. Zhou and O. Tuzel. Voxelnet: End-to-End Learning for Point Cloud based 3D Object Detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018. 2