Abstract

Depth completion starts from a sparse set of known depth values and estimates the unknown depths for the remaining image pixels. Most methods model this as depth interpolation and erroneously interpolate depth pixels into the empty space between spatially distinct objects, resulting in depth-smearing across occlusion boundaries. Here we propose a multi-hypothesis depth representation that explicitly models both foreground and background depths in the difficult occlusion-boundary regions. Our method can be thought of as performing twin-surface extrapolation, rather than interpolation, in these regions. Next our method fuses these extrapolated surfaces into a single depth image leveraging the image data. Key to our method is the use of an asymmetric loss function that operates on a novel twin-surface representation. This enables us to train a network to simultaneously do surface extrapolation and surface fusion. We characterize our loss function and compare with other common losses. Finally, we validate our method on three different datasets; KITTI, an outdoor real-world dataset, NYU2, indoor real-world depth dataset and Virtual KITTI, a photo-realistic synthetic dataset with dense groundtruth, and demonstrate improvement over the state of the art.

1. Introduction

Depth completion problems involve estimating a dense depth image from sparse depth measurements of active depth sensors, often guided by a high-resolution modality; e.g., RGB sensors. Solving depth completion has extensive applications, e.g., scene understanding [24], object shape estimation [5], and 3D object detection in autonomous driving [21].

Step-like object discontinuities are an inherent property of 3D scenes, and are challenging to model well with depth completion and depth super-resolution methods. It is important to maintain depth discontinuities to facilitate object shape and pose estimation. Most prior works rely on conventional regression losses for depth completion which, albeit promising results in depth accuracy, suffer from depth smear-

Figure 1: Our depth completion algorithm can input Lidar data and image (a), and extrapolate the estimates of foreground depth \( d_1 \) (b) and background depth \( d_2 \) (c), along with a weight \( \sigma \) (e). Fusing all three leads to the completed depth (d). The foreground-background depth difference (f) \( d_2 - d_1 \) is small except at depth discontinuities.

ing and hence shape distortion of objects. While [12] tackles this depth mixing problem, it has high computational and memory demand for accommodating many channels at high resolution. Another interesting work [19] learns non-local spatial affinity with spatial propagation network to eliminate smearing, but suffers from significant inference times and poor generalization due to sparse patterns (See Tab. 6).

One fundamental challenge of recovering depth discontinuity is that pixels at boundary regions suffer from ambiguity as to whether they belong to a foreground depth or background depth. Some methods seek to reduce the impact of ambiguities through intelligent regularization of the loss function [10] or ranking loss [33] at potential occlusion boundaries; and others explicitly train detectors for depth discontinuities [11] via full dense ground truth data. Unfortunately large scale datasets like KITTI have only partial dense ground truth depth (see Fig. 2 (a)) which is sparse on boundaries, making it difficult to explicitly label occlusion boundaries. Some recent works have leveraged prior information,
e.g., estimated semantic maps [40, 7, 32] and estimated depth maps [23] for object boundary recovery/refinement. Our approach instead seeks to explicitly model ambiguity and leverage it in depth completion. We noted that Depth coefficients [12] can also model ambiguities on account of its non-parametric probability distribution, but maintaining many high-resolution channels is computationally and memory expensive, and also suffer from the binning resolution. Instead of using multiple channels with binning, our method, named TWIN-Surface Estimation (TWISE), uses a two-surface representation which is much more efficient and can explicitly model ambiguity by finding difference between the twin surface depths. We believe that naturally encoding the foreground and background pixels at the boundary would enable the effective learning of the step-wise discontinuity with lower memory and computational requirement.

In order to train a twin-surface estimator, we propose a pair of asymmetric loss functions that naturally bias estimates toward foreground and background depth surfaces. The asymmetry in the losses is key to separation of foreground and background depths at ambiguous pixels. We also incorporate a fusion channel that automatically combines the foreground and background depths into a final depth estimate for each pixel, by selecting a foreground/background depth at the ambiguous regions and mixing the two depths at non-ambiguous regions.

Of particular concern is the lack of dense and reliable ground-truth depth data in outdoor scenes needed for accurate evaluation of depth estimates. KITTI, a realistic outdoor scene dataset, offers semi-dense ground-truth, created by accumulating lidar points but suffers from noisy depth samples (outliers) at boundaries and dynamic objects [30]. Indoor dataset like NYU2 provides dense GT only by using some colorization techniques that can cause smoothing at object boundaries. Currently the preferred evaluation metric of choice for ranking depth completion methods is RMSE. In this paper, we study the effects of outlier noise present in ground-truth data on RMSE and note that MAE is a more consistent metric for both cases of noisy and clean ground-truth, as validated on the synthetic KITTI dataset.

The contributions of this paper are as follows:

- We propose a twin-surface representation that can estimate foreground, background and fused depth.
- We adopt a pair of asymmetric loss functions to explicitly predict foreground-background object surfaces.
- We validate our theory in KITTI, a challenging outdoor scene dataset for depth completion, and show the superiority of our method on several metrics, and also show it generalizes well to variable sparsity and offers competitive inference times over the SoTA.
- We suggest that in presence of outliers, MAE is a more consistent metric to rank methods compared to RMSE, and we validate this claim with extensive experiments in KITTI, a synthetic dataset for urban driving scenario.

2. Related Works

Depth Completion Deep neural networks (DNNs) have been applied to the depth completion problem, in works such as Sparse-to-Dense [16], DDP [38], and Spade RGBsD [13]. These works show that by using standard encoder-decoder architecture (ResNet and MobileNet), it is possible to improve depth estimation accuracy via regression losses like $L_2$, $L_1$ and inverse $L_2$ losses. Deep-Lidar [22] estimates surface normal and dense depth using multiple DNNs to assist in further fine-tuning dense depth. Both [22] and [38] rely on synthetic data and various labels for learning depth representations. Recently, works have opted to optimize depth using 3D geometric constraints like depth-normal consistency [39, 36] to improve depth completion. Xu et al. create geometric consistency between the surface normal and depth in 3D, but use another refinement network for improved depth estimation [36]. Another recent trend is to learn spatial propagation of pixels in 2D depth space for depth completion problems in fixed [4] or variable receptive field [19, 37]. Although results are highly encouraging, these methods suffer from poor inference times and generalizability on variable sparsity. Researchers have also looked into learning 3D features for depth completion using continuous convolution in 3D space [2], point cloud completion [34], 3D graph neural networks [35] for dynamic construction of local neighborhood regions.

Depth Representations Depth maps, as 2.5D representations, have been used for RGBD fusion and instance segmentation [26, 8]. They naturally encode sensor viewing rays and adjacency between points. They are compact representations and their regular grids can be processed with CNNs in an analogous way to image super-resolution [27, 28]. This is the representation of choice for colorization techniques and fusion [18] as well as depth completion.

We propose a 2-layered representation of depth to model occlusion boundaries. The concept of layered representation of depth has been well known in graphics community. LDIs (Layered Depth Images) are first proposed by Shade et al. [25] as intermediate representation for efficient image-based rendering. These are gathered by accumulating depth values via z-buffering from multiple depth images of nearby view points. Tulisani et al. [29] infer 2-layered depth representation (recovering depth of visible and non-visible scene) from a single input image by learning view-synthesis from multiview camera guided supervision. Hedman et al. [9] propose a 3D photo reconstruction algorithm that builds multi-layered geometric representation of the scene by warping several depth maps and stitching color and depth panoramas for front and back-scene surfaces. In all these cases, multilayered representation is constructed/learned from multi-
camera viewpoints/depthmaps of the scene. In our case, we estimate these 2-layered representation on a single camera viewpoint with our proposed loss functions.

**Loss Functions in Depth Completion** A key component of depth completion is the choice of loss functions. Recent work has explored loss functions including $L_2$ [3, 16], $L_1$ [17], inverse-$L_1$ [13], Huber loss [2] and Softmax loss on depth [15]. Another elegant way is to use combination of $L_1 + L_2$ [19], which can leverage the benefits of both $L_1$ and $L_2$ losses. While these loss functions can achieve low error on metrics including RMSE, MAE, iMAE, often it comes at the cost of smoothing depth estimates across object boundaries. In addition to the aforementioned losses, people increasingly use Chamfer distance on point cloud [34], depth-normal constraint [36], Cosine loss [22], in a multi-learning framework to improve depth completion accuracy. Nevertheless, smoothing across sharp boundaries remains a concern in many of these methods. Imran et al. [12] show that cross-entropy (CE) loss can generate sharp boundaries, although performs worse in the RMSE metric.

We learn foreground and background depth by proposing two asymmetric loss functions, and the final depth using a fusion loss. The asymmetric loss metric has been used in Vogel et al. [31], for the different purpose of denoising input images. We propose to use asymmetric loss functions to learn biased estimators of FG/BG surface, and learn to select/blend (fusion loss) between FG/BG surface, and that, we claim, helps to recover depth discontinuity.

**3. Methodology**

Depth completion involves two quite different challenges which can be at odds. The first is to interpolate missing pixel depths within objects leveraging nearby sparse depths. The second is to accurately find the occlusion boundaries of objects and ensure that interpolated pixels belong to either the foreground or background object. We propose a method that aims to perform both tasks well.

Our approach divides depth completion into two simpler problems, each of which can be more easily learned by a network. The first problem is depth interpolation without boundary determination. Rather than estimating a single surface which must model step functions at depth discontinuities, Our key novelty is to estimate twin surfaces. A foreground surface extrapolates the foreground object depth up to and beyond boundaries, while a background surface extrapolates the background depth up to and behind the occluding object. Then the second problem is to find the boundary and determine a single depth by fusing these two surfaces. We find the color image is particularly useful in aiding surface fusion. Both of these components are illustrated in Fig. 2.

**3.1. Ambiguities and Expected Loss**

Ambiguities have a significant impact on depth completion, and it is useful to have a quantitative way to assess their impact. Here we propose using the expected loss to predict and explain the impact of ambiguities on trained networks.

By an ambiguity we mean, not that there isn’t a unique true solution, but rather that from a measurement it is difficult for the algorithm and/or human to decide between two or more distinct solutions. Ambiguity can be more formally defined as follows. Given measurement data that sparsely samples the scene, the number of ambiguities is equal to the number of different true scenes, i.e. true depth maps in our case, that could have generated the sparse measurement. This number depends on what variations occur in actual data. For simplicity we treat each pixel ambiguity independently of other pixels, and so the ambiguities for a pixel are the possible depth values it could take that are consistent with the measurement.

We anticipate the level of ambiguity to vary across a scene. For example, pixels on flat surfaces will be well-constrained by nearby pixels and have low ambiguity. In contrast, pixels near depth discontinuities may have large depth ambiguity. There is often insufficient data from the depth image to decide whether the pixel is on the foreground or background.

A corresponding color image can help resolve ambiguities as to which object a pixel belongs. However, exactly how to leverage color images to resolve ambiguities in CNNs is one of the open challenges in depth completion. Our work aims to offer a solution to this problem by explicitly estimating ambiguities and resolving them within the network.

To assess the impact of ambiguities on our network, we build a quantitative model. Consider a single pixel whose depth, $d$, we seek to estimate. Next assume that the pixel has a set of ambiguities, $d_1$ each with probability $p_i$. This probability measures of how likely it is that the ground truth will take the corresponding depth, given our modeled scene
assumptions. Now consider a loss function on the error for each pixel, \( L(d - d_i) \), where \( d_i \) is the ground truth depth. The expected loss as a function of depth is:
\[
E\{L(d)\} = \sum p_i L(d - d_i). \tag{1}
\]

This expected loss is important because if a network is trained on representative data then it will be trained to minimize the expected loss. Thus by examining the expected loss we can predict the behavior of our network at ambiguities, and so justify the design of our method.

3.2. Asymmetric Linear Error

Our method uses a pair of error functions which we call the Asymmetric Linear Error (ALE), and its twin, the Relected Asymmetric Linear Error (RALE), defined as:
\[
ALE_\gamma(\epsilon) = \max \left( -\frac{1}{\gamma}, \gamma \epsilon \right) \tag{2}
\]
\[
RALE_\gamma(\epsilon) = \max \left( \frac{1}{\gamma}, -\gamma \epsilon \right) \tag{3}
\]
Here \( \epsilon \) is the difference between the measurement and the ground truth, \( \gamma \) is a parameter, and \( \max(a, b) \) returns the larger of \( a \) and \( b \). The ALE and RALE are generalizations of the absolute error, and are identical to the absolute error when \( \gamma = 1 \). The difference is that the negative side of ALE is weighted by \( 1/\gamma \) and the positive weighed by \( \gamma \). The RALE is simply the reflection of the ALE over the \( \varepsilon = 0 \) line. Both are illustrated in Fig. 3 (a,b).

Note that if \( \gamma \) is replaced by \( 1/\gamma \), both the ALE and RALE are reflected. Thus, without loss of generality, in this work we restrict \( \gamma \geq 1 \).

3.3. Foreground and Background Estimators

We make a further simplifying assumption in our analysis that there are at most binary ambiguities per pixel. A binary ambiguity is described by a pixel having probabilities \( p_1 \) and \( p_2 \) of depths \( d_1 \) and \( d_2 \) respectively. When \( d_1 < d_2 \) we call \( d_1 \) the foreground depth and \( d_2 \) the background depth. Such a binary ambiguity is likely to occur near object-boundary depth discontinuities.

To estimate the foreground depth we propose minimizing the mean ALE over all pixels to obtain \( \hat{d}_1 \), the estimated foreground surface. To predict the characteristics of \( d_1 \) from a trained network at ambiguous pixels, we examine the expected ALE, as shown in Fig. 3 (c). This is piecewise linear and has two corners, one at \( d_1 \) and the other at \( d_2 \). The lower of these will determine the minimum expected loss, and hence what an ideal network will predict. Using Eqs. (2) and (1), we obtain expected losses: \( L(d_1) = p_2(d_2 - d_1)/\gamma \), and \( L(d_2) = p_1(d_2 - d_1)/\gamma \). From this it is straightforward to see \( L(d_1) < L(d_2) \) when:
\[
\gamma > \sqrt{\frac{p_2}{p_1}} \tag{4}
\]
This equation shows the sensitivity of the foreground estimator to \( \gamma \); the higher \( \gamma \), the lower the probability on foreground \( p_1 \) needed for the minimum to be at the foreground depth \( d_1 \).

To estimate the background depth, \( \hat{d}_2 \), at boundaries we propose minimizing the expected RALE. The same analysis will apply to this as to the ALE; and we obtain the same constraint on \( \gamma \) as in Eq. (4), except that the probability ratio is inverted.

Fig. 1 (b) shows an example foreground depth estimate, (c) the background depth and (f) the depth difference. We observe that at pixels far from depth discontinuities, as well as the sparse input-depth pixels, the foreground depth is very close to the background depth indicating no ambiguity.

3.4. Fused Depth Estimator

We desire to have a fused depth predictor that can do both interpolation and extrapolation at surfaces depending on ambiguous and non-ambiguous regions. The foreground and background depth estimates provide lower and upper bounds on the depth for each pixel. We express the final fused depth estimator \( \hat{d}_f \) for the true depth \( d_t \) as a weighted combination of the two depths:
\[
\hat{d}_f = \sigma \hat{d}_1 + (1 - \sigma) \hat{d}_2. \tag{5}
\]
where \( \sigma \) is an estimated value between 0 and 1. We use a mean absolute error as part of the fusion loss:
\[
F(\sigma) = |\hat{d}_f - d_t| = |\sigma \hat{d}_1 + (1 - \sigma) \hat{d}_2 - d_t|. \tag{6}
\]
The expected loss for this is
\[
L_\epsilon(\sigma) = E\{F(\sigma)\} = p|\sigma \hat{d}_1 + (1 - \sigma) \hat{d}_2 - d_1| + (1 - p)|\sigma \hat{d}_1 + (1 - \sigma) \hat{d}_2 - d_2|. \tag{7}
\]
Here, $p = p_1$, and $p_2 = 1 - p$. This has a minimum at $\sigma = 1$ when $p > 0.5$ and a minimum at $\sigma = 0$ when $p < 0.5$. Of course this assumes that depth is either $d_1$ or $d_2$.

Depth fusion occurs by optimizing the loss of Eq. (7) to predict a separate $\sigma$ for each pixel. In this way our fusion step is an explicit determination of whether a pixel is foreground or background or a combination. An example estimated $\sigma$ is shown in Fig. 1 (e).

3.5. Depth Surface Representation

We have developed three separate loss functions whose individual optimizations give us three separate components of a final depth estimate for each pixel. Based on the characterization of our losses, we require a network to produce a 3-channel output. Then for simplicity we combine all loss functions into a single loss:

$$L(c_1, c_2, c_3) = \frac{1}{N} \sum_j ^N (\mathbb{L}_{E_L}(c_{1j}) + \mathbb{R}_{E_L}(c_{2j})) + L_{\mathbb{S}}(s(c_{3j})))$$

(8)

Here $c_{ij}$ refers to pixel $j$ of channel $i$, $s()$ is a Sigmoid function, and the mean is taken over all $N$ pixels. We interpret the output of these three channels for a trained network as $c_1 \rightarrow d_1$, $c_2 \rightarrow d_2$, and $s(c_3) \rightarrow \sigma$, and combine them as in Eq. (5) to obtain a depth estimator $d_i$ for each pixel.

3.6. Implementation Details

Architecture This work presents novel loss functions linked to a multi-channel depth representation. These can be easily incorporated into a variety of network architectures with minimal change to the network. Specifically we selected the multistack network [14], with the author-provided code. We choose this network due to its fast inference time, lower number of parameters than [16], and its near-SoTA performance. The changes we made were three output channels and instead of one at each stacked hourglass network, and we use our loss function for the optimization. We used 64 channels in the encoder-decoder network as that provided their highest performing results. More details are shared in the supplementary material.

Training and Inference We followed the training protocol in [14] with multi-scale supervision on our 3 channels. The total loss is a weighted sum of the multiple resolution losses $L_1$, where $L_1$ is the full resolution 3-channel loss in Eq. (8), $L_2$ is half-resolution and $L_3$ quarter-resolution: $L = \omega_1 L_1 + \omega_2 L_2 + \omega_3 L_3$. The multiscale stage training protocol sets $\omega_1 = \omega_2 = \omega_3 = 1$ during the first 10 epochs, reduces $\omega_2 = \omega_3 = 0.1$, and continues to train for another 10 epochs. For the last 10 epochs we set $\omega_1 = 0$, and complete training after 30 epochs. Using Adam optimizer with an initial learning rate of $10^{-3}$ and decrease to half every 5 epochs, we train a full sized image with gradient accumulation every 4 samples in a batch. We use PyTorch [20] for our implementation.

4. Experimental Results

Dataset We evaluate the proposed algorithm on the standard KITTI Depth Completion dataset [6], a real-world outdoor scene, NYU2, with indoor scenes [18], and Virtual KITTI [1], a synthetic dataset with photo-realistic images and dense ground-truth depth. KITTI depth is created by aggregating LiDAR scans from 11 consecutive frames into one, producing a semi-dense ground truth (GT) with 30% annotated depth pixels. The sparsity of GT makes depth estimation more challenging. Note that we do not require any synthetic depth data for pre-training as used by [38, 22] to improve performance. The dataset consists of 85K, 1K, and 1K samples for training, validation, and testing respectively. Although the training set has different image sizes, the test and validation sets are cropped to a uniform size of $352 \times 1,216$.

Although created in a real world scenario, the semi-dense GT produced by Uhrig et al. [30] has far fewer depth points on object boundaries (see Fig. 2 (a)), and is susceptible to outliers. As we claim our method works well on boundaries, we also evaluate on VKITTI 2.0, a synthetic dataset with clean and dense GT depth at depth discontinuities. The VKITTI 2.0, created by the Unity game engine, contains 5 different camera locations (15° left, 15° right, 30° left, 30° right, clone) in addition to 5 different driving sequences. Additionally, there are stereo image pairs for each camera location. For training and testing, we only use the clone (forward facing camera) with stereo image pairs. For VKITTI training, 2k training images were created from driving sequences 01, 02, 06, and 018 respectively. For testing, we use sequence 020 at the left stereo camera, and choose every other frames, with total 420 images. We subsample the dense GT depth in azimuth-elevation space to simulate lidar-like pattern as sparse inputs. Further, we create the pseudo GT following [30] to study the effects of outlier noise on training and evaluation. More details are shared in the supplementary material.

To show the generalizability of our method, we also evaluate on NYU-Depth v2 dataset [18], which consists of RGB and depth images obtained from Kinect in 464 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 [22, 19]. 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 dimension of $304 \times 208$. Unlike [19], we use the same loss function for all the datasets.

Metrics The standard metrics used by KITTI include RMSE, MAE, iMAE and iRMSE. Since RMSE is used as
<table>
<thead>
<tr>
<th>Method</th>
<th>MAE</th>
<th>RMSE</th>
<th>tMAE</th>
<th>tRMSE</th>
<th>TMSE</th>
<th>TRMSE</th>
<th>Infer. time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ma et al. [16]</td>
<td>249.95/269.2</td>
<td>814.73/878.5</td>
<td>1.21/1.34</td>
<td>2.80/3.25</td>
<td>~/190.15</td>
<td>~/297.48</td>
<td>0.081</td>
</tr>
<tr>
<td>Depth-Normal [36]</td>
<td>235.17/236.67</td>
<td>777.05/811.07</td>
<td>1.79/1.11</td>
<td>2.42/2.45</td>
<td>~/--</td>
<td>~/--</td>
<td>--</td>
</tr>
<tr>
<td>Deeplidar [22]</td>
<td>226.50/215.38</td>
<td>758.40/887.0</td>
<td>1.15/1.10</td>
<td>2.56/2.51</td>
<td>~/162.75</td>
<td>~/266.79</td>
<td>0.097</td>
</tr>
<tr>
<td>3DDepthNet [34]</td>
<td>226.20/208.96</td>
<td>798.70/893.23</td>
<td>1.02/0.98</td>
<td>2.30/2.37</td>
<td>~/--</td>
<td>~/--</td>
<td>--</td>
</tr>
<tr>
<td>Uber-ParseNet [2]</td>
<td>221.19/217.0</td>
<td>752.88/785.0</td>
<td>1.14/1.08</td>
<td>2.34/2.36</td>
<td>~/--</td>
<td>~/--</td>
<td>--</td>
</tr>
<tr>
<td>MultiStack [14]</td>
<td>220.41/220.40</td>
<td>762.80/798.80</td>
<td>0.98/1.0</td>
<td>2.30/2.57</td>
<td>~/157.90</td>
<td>~/270.15</td>
<td>0.018</td>
</tr>
<tr>
<td>DC-3e [12]</td>
<td>215.75/215.04</td>
<td>965.87/1011.3</td>
<td>0.98/0.94</td>
<td>2.43/2.50</td>
<td>~/141.67</td>
<td>~/238.5</td>
<td>0.112</td>
</tr>
<tr>
<td>CSPN++ [4]</td>
<td>209.28/210.5</td>
<td>743.69/744.5</td>
<td>0.90/0.9</td>
<td>2.07/2.07</td>
<td>~/--</td>
<td>~/--</td>
<td>0.200</td>
</tr>
<tr>
<td>DDP [38]</td>
<td>205.40/205.0</td>
<td>836.00/836.0</td>
<td>0.86/0.86</td>
<td>2.12/2.12</td>
<td>~/--</td>
<td>~/--</td>
<td>--</td>
</tr>
<tr>
<td>NLSPN [19]</td>
<td>199.59/198.64</td>
<td>741.68/771.8</td>
<td>0.84/0.83</td>
<td>1.99/2.03</td>
<td>~/138.81</td>
<td>~/248.88</td>
<td>0.225</td>
</tr>
<tr>
<td>TWISE</td>
<td>195.58/193.40</td>
<td>840.20/879.40</td>
<td>0.82/0.81</td>
<td>2.08/2.19</td>
<td>~/131.60</td>
<td>~/239.80</td>
<td>0.022</td>
</tr>
</tbody>
</table>

Table 1: Depth completion on the Test/Validation sets of KITTI, with 64R LiDAR and RGB input (units in mm).

Figure 4: Comparison of our method with SoTA methods with whole and zoom in views (a) showing Color Images (b) DC [12], (c) MultiStack [14] (d) NLSPN [19] and our method (e). Four different regions of the image from two different instants are selected to show depth quality from diverse areas.

the preferred metric for depth completion, most SoTA methods on the KITTI leaderboard use MSE as their primary loss. We also include tMAE and tRMSE metrics proposed in [12] since it can discount outlier depth pixels (i.e., floating depth pixels around boundary regions) and give a better evaluation of depth pixels at and within object boundaries.

4.1. Results

Quantitative Results Tab. 1 compares the performance on KITTI's test/validation sets, with a 64-row LiDAR and color image as input. We list the SoTA methods with performance quoted from their papers. The inference times are calculated on a single GPU of GTX 1080 Ti. The method [19] with lowest RMSE achieves this at the expense of inference time. We outperform the SoTA methods in other metrics including MAE, and iMAE. The exception is RMSE, by which the methods are ranked in the KITTI leaderboard. That leads us to investigate in which areas our method perform better and worse, which we examine next.

Qualitative Results Fig. 4 shows our depth estimation quality compared to baselines. We choose three best SoTA methods: MultiStack [14], NLSPN [19], and DC [12]. Different local regions including poles, trees, cars, and traffic signs, illustrate the depth quality of close- and long-range depth pixels. The zoomed-in view shows the substantial improvement of our depth map over SoTA, especially along sharp object boundaries. [14] has a more blurred estimation around boundaries leading to mixed depth pixels and holes within objects, such as on the traffic poles and van. Although [19] has reduced mixed depths and more tighter boundary, depth mixing still exists (blurriness at object boundaries), additionally it suffers from jagged boundary edges and streaking artifacts.

Qualitative Parsing Fig. 5 offers a more detailed analysis of our method by showing different estimation at foreground, background depths and fused depth respectively. We choose five zoom-in views from diverse objects, e.g., tree, poles, car, and even pixels at far-away depth pixels. It shows that our fused depth estimator can learn to choose foreground and background regions well, resulting in a clear shape estimation of objects. We note that it is biased to choose the foreground surface as ambiguity increases, e.g., relatively large depth gap between foreground and background surfaces (see depth difference in Fig. 5 (f)). This can be explained by the fact that there are more supervision at the close-up region than the far-away region on account of uneven distribution.
Top performances on MAE and TMAE show the positive side effect of our loss addressing the smearing problem at the boundary. We particularly note that TWISE performs better than a standard $L_1$ loss on both the backbone networks, leading to believe that TWISE offers more benefit than a mere trade-off between MAE and RMSE.

Effect of $\sigma$ on Estimated Surfaces Another interesting evaluation is the importance of learned $\sigma$ on different estimated surfaces. In Tab. 4, we evaluate estimated depths for different combinations of $\sigma$ and compare individually its depth completion metrics. The performance is evaluated on our best model in Tab. 1, except for the row with “no color”, where we train without color input on the same network of our best model. From Tab. 4, foreground and background depth surface estimates, as usual, have higher error metric, since they are individually a biased estimate of depth. If we fix $\sigma$ at 0.5, we see it is possible to achieve decent performance on MAE and RMSE on account of averaging (interpolation) between the two surfaces. We make a binary choice between foreground and background surface if $\sigma > 0.5$ and the results are worse than averaging. In addition, we see $\sigma$ does not learn effectively without color input. So high-resolution imagery helps to learn effective $\sigma$ and resolve ambiguities at the boundaries.

Effect of $\gamma$ on Performance Since $\gamma$ impacts the separation of foreground and background surfaces, we perform an ablation to assess its impact on TWISE. Tab. 5 shows depth completion performance with several $\gamma$ values. With
Figure 9: (a) Results on Virtual KITTI experiments trained on clean GT and synthesized semi-dense respectively (units in cm). (b) MAE and (c) RMSE curves of scatter plots (Semi-Dense vs Clean GT) for different loss functions (colored symbols) and two backbone networks (MultiStack [14] and ResNet-18 [16]). Methods trained with the same backbone network are connected.

<table>
<thead>
<tr>
<th>Sparsity</th>
<th>Method</th>
<th>MAE</th>
<th>RMSSE</th>
<th>TMAE</th>
<th>TRMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>64R</td>
<td>DC [12]</td>
<td>279.1</td>
<td>1125.1</td>
<td>183.1</td>
<td>292.3</td>
</tr>
<tr>
<td></td>
<td>MultiStack [14]</td>
<td>229.4</td>
<td>889.7</td>
<td>156.8</td>
<td>265.0</td>
</tr>
<tr>
<td></td>
<td>NLSPN [19]</td>
<td>219.1</td>
<td>868.0</td>
<td>147.7</td>
<td>263.4</td>
</tr>
<tr>
<td></td>
<td>TWISE</td>
<td>203.8</td>
<td>927.6</td>
<td>134.1</td>
<td>240.5</td>
</tr>
<tr>
<td>32R</td>
<td>DC</td>
<td>392.7</td>
<td>1456.2</td>
<td>232.1</td>
<td>350.7</td>
</tr>
<tr>
<td></td>
<td>MultiStack</td>
<td>439.2</td>
<td>1288.8</td>
<td>275.4</td>
<td>402.3</td>
</tr>
<tr>
<td></td>
<td>NLSPN</td>
<td>392.4</td>
<td>1229.2</td>
<td>248.2</td>
<td>373.8</td>
</tr>
<tr>
<td></td>
<td>TWISE</td>
<td>327.9</td>
<td>1242.6</td>
<td>204.9</td>
<td>324.3</td>
</tr>
<tr>
<td>16R</td>
<td>DC</td>
<td>477.7</td>
<td>1777.3</td>
<td>259.5</td>
<td>382.9</td>
</tr>
<tr>
<td></td>
<td>MultiStack</td>
<td>528.4</td>
<td>1904.3</td>
<td>308.6</td>
<td>439.5</td>
</tr>
<tr>
<td></td>
<td>NLSPN</td>
<td>497.1</td>
<td>1843.8</td>
<td>286.8</td>
<td>419.2</td>
</tr>
<tr>
<td></td>
<td>TWISE</td>
<td>414.0</td>
<td>1481.1</td>
<td>237.3</td>
<td>365.1</td>
</tr>
<tr>
<td>8R</td>
<td>DC</td>
<td>653.7</td>
<td>2311.9</td>
<td>288.5</td>
<td>420.6</td>
</tr>
<tr>
<td></td>
<td>MultiStack</td>
<td>672.58</td>
<td>1841.6</td>
<td>353.2</td>
<td>486.8</td>
</tr>
<tr>
<td></td>
<td>NLSPN</td>
<td>669.05</td>
<td>1889.5</td>
<td>340.3</td>
<td>475.2</td>
</tr>
<tr>
<td></td>
<td>TWISE</td>
<td>532.1</td>
<td>1782.5</td>
<td>275.6</td>
<td>409.4</td>
</tr>
<tr>
<td>RGB</td>
<td>DC</td>
<td>2423.8</td>
<td>4433.6</td>
<td>715.4</td>
<td>797.2</td>
</tr>
<tr>
<td></td>
<td>MultiStack</td>
<td>2070.4</td>
<td>4185.1</td>
<td>635.7</td>
<td>735.4</td>
</tr>
<tr>
<td></td>
<td>NLSPN</td>
<td>2192.9</td>
<td>4362.35</td>
<td>646.0</td>
<td>743.6</td>
</tr>
<tr>
<td></td>
<td>TWISE</td>
<td>1964.1</td>
<td>4078.8</td>
<td>612.0</td>
<td>716.5</td>
</tr>
</tbody>
</table>

Table 6: Row sparsity impact on SoTA depth completion methods.

$\gamma = 1$, the loss is equivalent to MAE. As $\gamma$ increases, the gap between foreground and background surface increases. At small $\gamma$ values, the interpolation benefits, thus leading to lower MAE, TMAE, TRMSE, since it is easier to interpolate between two nearby surfaces; however, in the meantime extrapolation suffers, thus leading to higher RMSE. At larger $\gamma$, the slope between two surfaces increase, and interpolation becomes harder. We choose $\gamma = 2.0$ in our experiment as a compromise between interpolation and extrapolation.

**Effect of Sparsity on Depth Performance** We also ran an extensive ablation study on generalization of SoTA methods due to sparsity. Sparsity is created by subsampling lidar-points in azimuth-elevation space to simulate lidar-like structured patterns. All the SoTA methods compared have been retrained using the author provided code with variable sparse input patterns. Tab. 6 shows that TWISE has better generalization and exhibits significantly less errors in all the metrics compared to SoTA methods. With more sparsity, TWISE is able to beat the RMSE metrics of methods supervised by standard losses. Particularly interesting is the fact that TWISE can be used for monocular depth estimation with no sparse depth input.

**Synthetic Experiments with VKITTI** Using both semi-dense GT and clean GT of VKITTI, we ran experiments on different loss functions using two different backbone networks. The conclusion is drawn by training and evaluation on noisy semi-dense and clean GT respectively. The results are shown in Fig. 9 (a). Several inferences can be drawn from the scatter plot of Fig. 9 (b) and (c). Firstly, the MAE score is smooth and monotonic as opposed RMSE which zigzags. This implies that given a MAE score on semi-dense, we are able to predict its score on the clean dataset as well. The ranking of the methods in both the datasets is the same for MAE but not RMSE. As a result, we can conclude that MAE is a superior metric to RMSE for comparing and ranking depth completion methods.

Secondly, TWISE is more than a trade-off between MAE and RMSE. One of the objective of TWISE is to improve depth points at discontinuity regions. But KITTI semi-dense GT lacks dense ground-truth depth points, and contains outliers in the boundary regions owing to methodology adopted in creating the GT. In presence of outliers, RMSE in TWISE suffers the most, but when clean GT can be provided, RMSE in TWISE performs as well as those methods with the $L_2$ loss.

**5. Conclusion**

In this paper we propose TWISE, a new twin-surface representation and estimation method for depth images. Our proposed asymmetric loss functions, ALE and RALE, bias these twin surface estimates towards the foreground and background at pixels with depth ambiguity. A third channel of our output fuses these estimates to achieve a single surface estimate. This solution simplifies the task of learning depth discontinuities, and as a result better maintains step-wise depth discontinuities across boundaries, and generates SOTA depth estimates. We also compared the robustness of MAE and RMSE as metrics for ranking depth completion methods and our analysis suggests that MAE is a superior metric in presence of noisy GT datasets. In future, we would like to improve our estimates at far-away depth pixels where learning suffers due to sparsity of ground-truth pixels.
References


[35] Xin Xiong, Haipeng Xiong, Ke Xian, Chen Zhao, Zhiguo Cao, and Xin Li. Sparse-to-dense depth completion revisited: Sampling strategy and graph construction+. 2020.


