NOVEL DEPTH REPRESENTATIONS FOR DEPTH COMPLETION WITH APPLICATION IN 3D OBJECT DETECTION

By

Saif Muhammad Imran

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering - Doctor of Philosophy

2022

ABSTRACT

NOVEL DEPTH REPRESENTATIONS FOR DEPTH COMPLETION WITH APPLICATION IN 3D OBJECT DETECTION

By

Saif Muhammad Imran

Depth completion refers to interpolating a dense, regular depth grid from sparse and irregularly sampled depth values, often guided by high-resolution color imagery. The primary goal of depth completion is to estimate depth. In practice methods are trained by minimizing an error between predicted dense depth and groundtruth depth, and are evaluated by how well they minimize this error. Here we identify a second goal which is to avoid smearing depth across depth discontinuities. This second goal is important because it can improve downstream applications of depth completion such as object detection and pose estimation. However, we also show that the goal of minimizing error can conflict with the goal of eliminating depth smearing.

In this thesis, we propose two novel representations of depths that can encode depth discontinuity across object surfaces by allowing multiple depth estimation in the spatial domain. In order to learn these new representations, we propose carefully designed loss functions and show their effectiveness in deep neural network learning. We show how our representations can avoid inter-object depth mixing and also beat state of the art metrics for depth completion.

The quality of ground-truth depth in real-world depth completion problems is another key challenge for learning and accurate evaluation of methods. Ground truth depth created from semi-automatic methods suffers from sparse sampling and errors at object boundaries. We show that the combination of these errors and the commonly used evaluation measure has promoted solutions that mix depths across boundaries in current methods. The thesis proposes alternate depth completion performance measures that reduce preference for mixed depths and promote sharp boundaries.

The thesis also investigates whether additional points from depth completion methods can help in a challenging and high-level perception problem; $3D$ object detection. It shows the effect of different depth noises originated from depth estimates on detection performances and proposes some effective ways to reduce noise in the estimate and overcome architecture limitations. The method is demonstrated on both real-world and synthetic datasets.

I would like to dedicate this thesis to my parents ABM Saiful Islam and Zareen Salma, my wife Nazifa and my lovely 1 year old daughter Qaanita who has been a continuous inspiration for me during the journey.

ACKNOWLEDGMENTS

I would like to express deepest gratitude to my advisors Dr. Daniel Morris and Dr. Xiaoming Liu for the support and funding throughout the turbulent years of my PhD journey. I thoroughly enjoyed working with them at late night hours before submission deadlines. My lab colleagues Mehmet Alper and Yunfei Long have been always through thick and thin on this arduous journey. I would like to thank Dr. Liu's lab group members; especially Garrick, Shengjie, Amin and Luan for brainstorming and sharing vision ideas and facilitating access to server gpus and setups. Working with them was a true inspiration on its own. I am indebted to my wife for her unwavering support, understanding and dealing with crisis mode when needed, my parents who came all the way from home to support me with my then expecting wife and all the household chores. Everytime I lost hope, their prayers and consolation kept me persistent to my target. And finally all praise goes to Almighty Allah for all the blessing and love that comes along the way.

TABLE OF CONTENTS

LIST OF TABLES . . . . . **x**

LIST OF FIGURES . . . . . **xiii**

Chapter 1 Background and Motivation **1**
1.1 Introduction 1
1.2 Depth Estimation using Multi-Modal Sensors: A Motivation 1
1.3 Depth Ambiguity 6
1.4 Thesis Outline 8

Chapter 2 Depth Representation **10**
2.1 Introduction 10
2.2 Representations of Depth 11
2.2.1 3D Point Clouds 12
2.2.2 Voxels 12
2.2.3 Polygon Meshes 13
2.2.4 SingleView and Multiview Depth Maps 14
2.3 Multichannel Representation of Depth: Depth Coefficients 14
2.3.1 Depth Representation 15
2.3.1.1 Discrete Representation 16
2.3.1.2 Probability Representation 16
2.3.2 Mathematical Notations 17
2.3.3 Depth Reconstruction from True DC 18
2.4 Dual Channel Representation of Depth: Twin Surface 19
2.4.1 Depth Reconstruction from Foreground and Background Surfaces 20
2.5 Proposed Evaluation Metrics 21
2.6 Conclusion 24

Chapter 3 Depth Coefficients for Depth Completion **25**
3.1 Introduction 25
3.2 Related Works in Depth Completion 25
3.3 Avoiding Depth Mixing by Convolution 28
3.3.1 Motivation 30
3.3.2 Proposed Loss Function 31
3.3.2.1 Depth Loss Functions with Ambiguity 32
3.3.2.2 Cross Entropy as Loss Measure 33
3.4 Learning by Deep Neural Network 34
3.4.1 Neural Network Architecture 36
3.4.2 Depth Reconstruction 36
3.5 Experiments and Results 37

3.5.1 Experimental Protocols . . . 37
3.5.1.0.1 Sub-Sampling . . . 38
3.6 Conclusion . . . 40

Chapter 4 Depth Completion Using TWIN Surface Extrapolation at Occlusion Boundaries . . . 41
4.1 Introduction . . . 41
4.2 Related Works . . . 42
4.2.1 Depth Completion . . . 42
4.2.2 Depth Representations . . . 44
4.2.3 Loss Functions in Depth Completion . . . 45
4.3 Ambiguities and Expected Loss . . . 45
4.4 Methodology . . . 46
4.4.1 Ambiguities and Expected Loss . . . 47
4.4.2 Asymmetric Linear Error . . . 49
4.4.3 Foreground and Background Estimators . . . 50
4.4.4 Fused Depth Estimator . . . 51
4.4.5 Depth Surface Representation . . . 52
4.4.6 Implementation Details . . . 52
4.4.6.1 Architecture . . . 52
4.4.6.2 Training and Inference . . . 53
4.5 Experimental Results . . . 54
4.5.1 Dataset . . . 54
4.5.2 Metrics . . . 56
4.5.3 Results . . . 56
4.5.3.1 Quantitative Results . . . 56
4.5.3.2 Qualitative Results . . . 56
4.5.3.3 Qualitative Parsing . . . 58
4.5.3.4 Relative Error Maps . . . 58
4.5.3.5 Outlier Errors and Analysis on KITTI Semi-Dense GT . . . 61
4.5.3.6 Quantitative Results on NYU2 . . . 63
4.5.4 Ablation Studies . . . 63
4.5.4.1 Effect of Loss Functions . . . 64
4.5.4.2 Effect of σ on Estimated Surfaces . . . 65
4.5.4.3 Effect of γ on Performance . . . 65
4.5.4.4 Effect of Sparsity on Depth Performance . . . 66
4.5.4.5 Synthetic Experiments with VKITTI . . . 66
4.6 Conclusion . . . 68

Chapter 5 3D Object Detection from Noisy Depth . . . 70
5.1 Introduction . . . 70
5.2 Related Works . . . 74
5.2.1 Depth Completion and Depth Prediction . . . 74
5.2.2 $3D$ Object Detection with Multi-Modal Sensors . . . 74
5.2.3 3D Object Detection from Estimated Depth . . . 76

5.3 Impact of Noisy Depth on Object Detection . . . 77
5.3.1 Baseline Results . . . 78
5.3.2 Noise Modelling . . . 79
5.3.3 Architectures . . . 83
5.3.3.1 Promotion of Noisy Pixels using FPS . . . 83
5.3.3.2 Downsampling in Point-Based Architecture . . . 84
5.3.4 Remedies to Tackle Noisy Depth . . . 85
5.3.4.1 Filtering Smeared Points . . . 85
5.3.4.2 Filtering Background Clutter . . . 86
5.3.4.3 Filtering Pixels within r distance from raw LiDAR . . . 86
5.4 Experiments and Results . . . 88
5.4.1 Dataset . . . 88
5.4.1.1 KITTI . . . 88
5.4.1.2 Virtual KITTI . . . 88
5.4.2 Metrics . . . 89
5.4.3 Architecture . . . 89
5.4.4 Implementation Details . . . 90
5.4.5 Results . . . 90
5.5 Conclusion . . . 96

Chapter 6 Conclusion and Future Works . . . 97
6.1 Conclusions . . . 97
6.1.1 Contributions . . . 97
6.1.1.1 Multi-Channel and Dual Channel Representation . . . 97
6.1.1.2 Loss Functions . . . 98
6.1.1.3 Noisy GT and Metrics . . . 98
6.1.1.4 3D Object Detection from Noisy Depth . . . 99
6.1.2 Future Improvements . . . 99

BIBLIOGRAPHY . . . 101

LIST OF TABLES

Table 1.1: Comparisons of active and passive depth sensors 4

Table 3.1: Quantitative results of NYU2 (Done on Uniform-500 Samples + RGB) (units in m). 39

Table 3.2: 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. 39

Table 3.3: 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. 39

Table 3.4: Average precision (%) for 3D detection and pose estimation of cars on KITTI [1] using Frustum PointNet [2]. The baseline, Raw-16R, uses 16 rows from the Lidar, while Ma's method [3] 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 [1] 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. 40

Table 4.1: Depth completion on the Test/Validation sets of KITTI, with 64R LiDAR and RGB input (units in mm). 54

Table 4.2: Error metrics for different image regions on TWISE. 61

Table 4.3: Relation between Disparity Error and Depth Error in metric units (cm). Note that KITTI Outliers are defined by: > 3 pix disparity error and 5% error. . . . 62

Table 4.4: Depth completion results on NYU2 [4]. 63

Table 4.5: Effect of different loss functions. Compared to single channel losses, CE requires 80 channel, while TWISE requires 3 channel. 64

Table 4.6: Effect of learned σ in TWISE, evaluated by our best model. 64

Table 4.7: Effect of γ on depth completion performance. 64

Table 4.8: Row sparsity impact on SoTA depth completion methods. 68

Table 5.1: 3D Object Detection results ($3D$ and Bird-eye view (BEV) average precision respectively) with different depth resolutions. Object refers to object detection [5] and Depth refers to depth completion [6] network. Raw refers to 64R LiDAR, Semi-Dense GT refers to the GT created by accumulating LiDAR points used for supervision of depth completion network in KITTI. We use the results of two depth completion networks (MultiHourGlass [6] and TWISE [7]) for comparison purposes. 78

Table 5.2: AP Comparison of raw (64R LiDAR) and dense depth (TWISE) at different depth ranges in *meters*. Numerical results also show dense depth performs worse compared to raw LiDAR at all categories; with the gap increasing at higher depth range and more difficult categories respectively. 79

Table 5.3: $3D$ Object Detection with augmented LiDAR. L refers to 64R LiDAR, D refers to Dense Depth with TWISE, F refers to sigma filter 5.3.4.1, SF refers to filter used with semantic mask, and Aug.F refers to LiDAR augmented with dense depth estimate using the semantic mask and filter. 91

Table 5.4: Performance evaluation of different architectures on a validation set of KITTI with dense depth from TWISE. D refers to Dense Depth with TWISE, F refers to sigma filter 5.3.4.1. It shows the filter designed with TWISE output can improve detection performance significantly by pruning floating and ambiguous depth pixels. 92

Table 5.5: 3D Object detection results with defined sampling regions to reduce background clutter and bypass architecture limitations. Filtered depth refers to the TWISE filter 5.3.4.1, 64R and 128R are depth sampled in azimuth and elevation space to simulate a LiDAR, while semantic filtered depth refers to the estimated semantic mask created by [8], and used to filter dense depth in image plane based on vehicle pixels. The dense depth pixels are reduced further by grid sampling in 3D at grid spacing 0.1m. 94

Table 5.6: 3D Object Detection results with the apriori sampling region and different radius configurations used to gather background pixels around the relevant object pixels. Sem filtered refers to sigma filter and semantic map used to filter out background and floating depth points. Sem-Mask in last row refers to binary semantic information which is concatenated with the input pointcloud as additional information to the detection network. 95

Table 5.7: Average precision (%) for 3D detection and pose estimation of cars on VirtualKITTI using PointRCNN2[[5]]/VoxelRCNN[9]. Alldep refers to depth pixels grid sampled at 0.1m, while semantic pixel refers to selected pixels using GT semantic masks. Note that pixels within $r = 0.3$m radius of semantics are also taken into consideration. 95

Table 5.8: 3D detection and pose estimation of cars on VirtualKITTI [1] using different types of depth noises. Gaussian Blur smooths depth along boundaries and simulate smeared depth; Depth Misalign. simulates depth error along the optical ray. 96

Table 5.9: Average precision (%) for 3D detection and pose estimation of cars on VirtualKITTI [1] using PointRCNN2. 2D Bounding box refers to the GT bounding boxes within which depth pixels are sampled, while Semantic refers to GT semantics. Note that pixels within $r = 0.3$m radius of semantic pixels are also taken into consideration. 96

LIST OF FIGURES

Figure 1.1: Performance comparison of depth completion method using different input sensor modalities (raw depth, monocular and monocular + raw depth). A standard deep learning method (Ma et al. [10]) is used in outdoor KITTI depth completion dataset for evaluation. The maximum depth range available for evaluation is 85m. x-axis refers to the no. of LiDAR scanlines projected to the image as raw depth, y-axis refers to the RMSE metric in cm. 0 scanline resembles depth estimation using monocular camera only. It shows that monocular and sparse depth as input modality can reduce the performance gap by around 5 times compared to monocular camera only just by increasing the raw depth measurements. The performance gap between raw depth and multimodal input (monocular + depth) decreases as the number of raw depth measurements increase. 5

Figure 2.1: An illustration of different representations of depth; (a) represents point-cloud data (Courtesy of Caltech) (b) represents voxel grid data (courtesy of IIT Kharagpur), (c) represents triangle mesh (Courtesy of UW), and (d) represents multiview representations of depth (Courtesy of Stanford) by means of depth maps. 11

Figure 2.2: Illustration of DC. The depth pixel (red circle) in the image plane is projected back to the 3D space by a pixel ray. The depth bins are quantized along that pixel ray. Each bin is then assigned a weight (depth coefficients) on few depth bins based on the proximity of the depth with the depth bins. 15

Figure 2.3: Representing depth by means of multiple depth bins or *depth coefficients* to preserve depth precision. In this toy example, we divide the depth range of 10m into 10 bins, each bin spaced 1m apart from each other. (a) depth representation by a single bin, which loses precision (b) depth representation by finite number of bins which preserve precision. 17

Figure 2.4: Dual Channel Representation 20

Figure 2.5: Illustration why RMSE and MAE metric are not good at boundaries. x-axis shows the $(\hat{y_i} - \tilde{y_i})$ where $\hat{y_i}$ and $\tilde{y_i}$ are estimated depth and groundtruth depth respectively, y-axis shows the loss functions. (a) and (b) shows the metric for MSE and MAE, and TRMSE and TMAE of depth and their characteristic loss curve respectively. (c) and (d) shows the expected MSE and MAE, and expected TRMSE and TMAE when depth pixels are missing between $(10m)$ and $(13m)$ (see text). From (a) and (b), both MSE and MAE and TMSE and TMAE shows perfect prediction/no errors when it coincides with GT, but for MSE and MAE the errors continue to increase if the depth estimates are far away from the GT. However, for TRMSE and TMAE, a fixed error occurs for points beyond a threshold. From (c) and (d), the expected MSE and MAE favors estimates in-between $10m$ and $13m$, and thus, indirectly promote depth mixing. However, the TMSE and TMAE favor either points $(10m)$ or $13m$. In a way, the TMSE and TMAE do not penalize depth estimates when one surface/depth point $(10m)$ is chosen instead of the other surface/depth point $(13m)$, and only account for intra-surface variations of depth estimates. 22

Figure 3.1: 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. [3] 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 height 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. 26

Figure 3.2: Illustration of Depth Mixing. (a) shows sparse measurements (red) in color image, (b) shows estimated depth map [10], (c) shows pointcloud generated from the depthmap (b). The black crosses in (c) are the sparse measurements. The red $3D$ box indicates the position of the car. Between the foreground mode (car) and the background mode (walls of the building), the $3D$ points floating in between are the mixed depth pixels. We say the car is smeared along its boundary. 29

Figure 3.3: 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. 31

Figure 3.4: (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. 33

Figure 3.5: An illustration of P_{data} modeled as the sum of the DC of the two points from Fig. 3.4. The estimated $\hat{c}_{ij}$ with minimum cross-entropy loss, Eq. 3.3, will exactly match P_{data}, providing a multi-modal density. A pixel depth estimate using Eq. 3.4 will find the depth of one of the peaks, and not a mixed-depth value. 34

Figure 3.6: 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. 35

Figure 3.7: Our CNN architecture modified from [3] with 80-channel DC at the input, and 80-channel cross-entropy loss. The residual blocks are defined in the dashed red region, and the top branch consists of ResNet 34 [11]. 36

Figure 3.8: Depth completion with 16-row Lidar. (a) scene, (b, e) show Ma *et al.* [3] with significant mixed pixels. (c, f) show our 3-coefficient estimation, demonstrating very little depth mixing. (d, g) show our estimation with all coefficients. 38

Figure 3.9: Another depth completion example with 16-row Lidar, where all subfigures are defined the same as Fig. 3.8. Interestingly, higher RMSE is reported on 3-coefficient estimation as opposed to all-coefficient estimation. 38

Figure 4.1: 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 σ (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. . . 43

Figure 4.2: Depth smearing across boundaries. We show the ground truth depth (colored red) overlaid on an image (a), depths estimated by the SoTA method [6] (b), our fused depths (c), our estimated weights σ (d), a depth slice of [6] (e), fused depth and σ slices (f), and foreground and background slice (g). Our extrapolation ability in (g) results in the sharp depth boundary in (f), rather than the smeared depth in (e). . . . 46

Figure 4.3: (a) The *ALE* from Eq. (4.2) is asymmetric around its minimum at the origin. (b) The *RALE* from Eq. (4.3) is a reflection of the *ALE*. We use the *ALE* for foreground surface estimation and the *RALE* for background estimation. (c) A pixel depth is shown with two ambiguities at depths d_1 and d_2 and probabilities $p1$ and $p1$ respectively. The black line shows the expected *ALE* which is the probability-weighted sum of two *ALE* functions, see Eq. (4.1). The expected *ALE* will have a minimum at one of the marked corners occurring at d_1 and d_2. The minimum will be at d_1 if Eq. (4.4) is satisfied, as it is in this case with $p_1 = p_2$, and so acts as a foreground depth estimator. 49

Figure 4.4: Incorporating 3-channel at the output of the Hour-glass network used in [6]. SD^n and FD^n are the sparse inputs and fused depth obtained from FG^n, BG^n, and σ^n at multi resolution scale n respectively. 52

Figure 4.5: Comparison of our method with SoTA methods with whole and zoom in views (a) showing Color Images (b) DC [12], (c) MultiStack [6] (d) NLSPN [13] and our method (e). Four different regions of the image from two different instants are selected to show depth quality from diverse areas. 54

Figure 4.6: Input image (a), its zoom-in views (d), our estimation on foreground depth (b), background depth (c), fused depth (e), and the depth difference between foreground and background depth (f). 57

Figure 4.7: Difference of TWISE vs MultiStack [6] in (a) Absolute Error (AE) and (b) Squared Error (SE) respectively. The red indicates the most gain of ours over [6], marked by 'o'; while the blue is vice-versa, marked by 'x'. Zoom in for details. 59

Figure 4.8: (a) Magenta is a histogram of absolute error differences $A(i)$ for $A(i) > 0$ (where MultiStack errors $>$ TWISE errors) and green is a histogram of $|A(i)|$ for $A(i) < 0$ (where TWISE errors $>$ MultiStack errors). (b) Corresponding histograms for squared pixel error differences $S(i)$. 60

Figure 4.9: Color images (top) and depth error maps in $0 - 5$m (bottom). 61

Figure 4.10: Semi-dense GT depths overlaid on color images. Zoom-in views show foreground/background depths are incorrectly spread (dilated/constricted) across boundaries of poles, traffic signs etc. visible in color images. 62

Figure 4.11: (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 [6] and ResNet-18 [10]). Methods trained with the same backbone network are connected. 65

Figure 4.12: KITTI sparse patterns of (a) $64R$, (b) $32R$, (c) $16R$, and (d) $8R$ subsampled LiDAR respectively overlaid on a color image. 67

Figure 5.1: Object detection from a SOTA architecture [5] trained with dense inaccurate pointclouds backprojected from a depth image by SOTA depth completion method i.e. TWISE [7]. The figure showing (a) several false positive detections (red 3D cuboids) on the estimated high-density pointcloud. Estimated (colored) and LiDAR (white) pointclouds also shown in $3D$ space (b) Estimated depth map in 2D space from where the estimated pointcloud in (a) is originated from. The LiDAR points are also shown in white dots, and (c) shows the estimated and groundtruth $3D$ bounding boxes projected to a 2D image space of the scene. Red and blue boxes are predicted and groundtruth $3D$ bounding boxes respectively. It shows that bush and phone booth are being wrongly classified as cars. 72

Figure 5.2: 3D detections from high-density pointcloud. The color image and sparse depth is fed into a depth completion network, and dense depth estimate is obtained. The high-density depthmap is then converted to pointcloud in 3D, and trained with a SOTA object detection network, the end-result is 3D detections (red 3D bounding boxes) in pointcloud. 77

Figure 5.3: Different types of depth noise on estimated depth; the first column shows color images, the second column shows depth images, and the third column shows 3D pointclouds; (a), (b) and (c) show *noise-fg*, indicating smeared points within the car resulting in misalignment of the predicted and GT bounding boxes as shown in red and blue respectively; (d), (e) and (f) show *noise-inbet*, indicating smeared pointcloud between two objects resulting in the wrong orientation of the 2nd car; and (g), (h) and (i) show *noise-bg*, the background points (phone-booth) wrongly classified as a car since its outer surface looks similar to a car shape. 82

Figure 5.4: Selecting 4096 points from (a) 64R raw LiDAR and (b) dense depth (noisy environments) of TWISE. FPS samples more background and outlier points at boundaries of tree trunks and buildings from noisy dense depth. 84

Figure 5.5: Subsampling in point-based architecture. The dense depth points get subsampled to 64 encoded points at the final encoder level. 84

Figure 5.6: Removing smeared points by TWISE. (a) shows the rectangular region $0.2 < \sigma < 0.8$ and depth difference (difference between BG and FG in TWISE) $>= 3m$ as smeared depth points. The σ parameter refers to σ parameter learned in TWISE. (b) and (c) show the unfiltered and filtered depth points in $3D$ space respectively. As shown, most of the floating depth pixels at the scene in (b) is filtered out at (c). 85

Figure 5.7: Sampling to select relevant pixels from dense depth. $(a) \approx 315k$ with dense depth, $(b) \approx 200k$ after pruning ambiguous pixels based on sigma filter 5.3.4.1, $(c) \approx 35k$ after grid sampling at 0.2m, $(d) \approx 15k$ points after using object level semantic mask and gathering neighboring points 0.3m from the semantic pixels in 3D. 87

Figure 5.8: Object detection performance comparison in bird eye view with different depth inputs. (a) show color image, and BEV detection on (b) sparse depth, (c) dense depth, (d) depth filterd with sigma filter (5.3.4.1), (e) augmented LiDAR with variable radius respectively. (f), (g), (h), and (i) show another example of a scene with color image and bird eye view of all the methods subsequently. In both the examples, augmented LiDAR has best performance compared to all other methods. 93

Chapter 1

Background and Motivation

1.1 Introduction

In 3D vision, *depth* perception is the visual ability to estimate the world in 3D dimensions and gives us the physical distance of an object relative to a calibrated sensor. It is critical for 3D scene understanding and is widely used in autonomous driving and navigation, high-definition mapping and augmented reality. Accurate, dense and high-resolution depth is desired for 3D scene reconstruction and accurate scene perception.

1.2 Depth Estimation using Multi-Modal Sensors: A Motivation

In the context of automotive sensing and perception, depth plays a large role in 3D object detection [2, 14], classification, localization [15, 3], tracking [16], shape estimation [17], 3D reconstruction [18] and modelling [19]. Depth is necessary to measure any 3D object dimensions and locate its position with respect to the ego-vehicle, and reconstruct the 3D structure of the environment for mapping, localization and even path-planning and collision avoidance in an autonomous driving environment. There are several ways to estimate depth; estimation using active depth sensors and passive sensors, or combination of these sensors. The following explains the methods of

estimating depth on each of these types and the motivation of using depth estimation using multi-modal sensors.

Active depth sensing involves transmitting a signal into the scene and measuring its impact in real-time. These sensors can also be known as three-dimensional (3D) range finders, which means they can acquire multi-point distance information across a Field-of-View (FoV). These sensors may emit laser pulses (time-of-flight or ToF sensors), infra-red beams (active stereo) or structured light patterns (fixed pattern or programmed patterns) on surrounding environments to measure depths. Active stereo sensors (e.g. Intel RealSense D Series, Structure Core) work with an infrared pattern projector in addition to the principle of triangulation by means of two cameras, separated by a baseline distance to measure depth. The infrared (IR) projector helps to increase the fidelity and reliability even in low-light conditions. However, the IR projectors on these sensors are limited in range, and these relegates the sensors to near and mid-range applications, regardless of the baseline distance. Structured light sensors (e.g. Zivid, Intel RealSense SR series, Kinect v1) combine low cost with high-fidelity 3D data capture, as well as performance in a wide set of lighting conditions, except for direct or indirect bright light. This is because infra-red light used by these sensors gets overpowered by infra-red light in the same bandwidth that is naturally present in the environment. Time of flight or TOF sensors (e.g. Velodyne, Ouster, Terabee) send out packets of modulated or unmodulated infra-red or laser pulses, and record the time it takes for the signal to return. They are less susceptible to interference from bright and indirect sunlight, but more susceptible to absorption of transmitted signals on dark, rough, or specular surfaces. The accuracy, and range of these sensors are based on the power consumption of the emitters, modulation and specific technologies used (see Tab. 1.1). Normally, they have mm - cm level accuracy, but either they are sparse, or have low resolution, or limited range capacity, or all of them. Amongst all the sensors, ToF sensors like LiDARs are widely used for longer-range applications for automotive safety and

navigation.

Passive depth sensing refers to depth estimation using passive camera sensors by leveraging depth cues of the visible environment. This depth cues come in the form of relative object sizes at different ranges from the camera, occlusions, texture gradients, shading, aerial perspective etc. Depth estimation are typically done in regular 2D grids and thus offer high-resolution depth. Different types are possible; single-view monocular, multi-view monocular, motion-based monocular, and stereo depths. The most accurate amongst them are the stereo depth sensors (e.g. StereoLabs ZED, Ensenso), which operate on the principle of triangulation between two cameras with a baseline distance to estimate depth from disparity. They come in a wide range of baseline distances, and thus can operate at different depth ranges. However, their accuracy depends on matching features between two images, image resolution, stereo calibration and their performance suffers in low-light conditions, textureless, smooth or specular surfaces. Some more advanced learning algorithms [20, 21, 22] leverage neural networks and novel loss functions to tackle those weaknesses, but computational and algorithm complexity makes them slow and less readily adaptable for realtime applications. Depth estimation using single view [23, 24, 25] has been popular research problem over the last decade and on due to availability of cheap and high-resolution sensors like cameras. The learning algorithms often leverage availability of ground-truth depth [24, 26], novel CNN architectures [27, 28], or loss functions [29, 30] to improve depth accuracy, but the performance gap is still wide compared to existing active depth sensors (see Tab. 1.1). Multi-view [31, 32] and motion-based [15, 33] monocular depth estimation are also widely researched to leverage multiple viewpoints of the camera or motion-based cues, but depth errors are still in the range of meters due to occlusions, moving objects, shadows or textureless surfaces and errors in camera pose.

While performance in range and resolution are improving, the cost for higher-resolution Li-

Property	Active Sensing			Passive Sensing		
	Active Stereo	**Structured Light**	**ToF**	**Single-view mono.**	**Multi-view mono**	**Passive Stereo**
Light Property	Infra-red beam projector	Known pattern of light	Known speed of light	N/A	N/A	N/A
Effective Range in meters	< 10 med range depends on IR projector	< 3 very short to med range, depending on illum. power	< 100 Short to long range depends on laser power and modulation	Theoretically Infinite	Theoretically Infinite	< 20 depends on baseline between cameras
Depth Accuracy	mm to cm Gradually diminishing with distance, difficulty with smooth, textureless surface	mm to cm Rapid fall-off beyond projection range	mm to cm Rapid fall-off beyond projection range	m Specific to Algo	m Specific to Algo	mm to cm Specific to Algo, Gradually diminishing with distance
Low Light performance	Good	Excellent	Excellent	Poor	Poor	Poor
Scanning Speed	Medium Limited by software complexity	Fast Limited by camera speed	Fast Limited by sensor speed	N/A	N/A	N/A
Latency	Medium	Medium	Low	High	High	Medium
Software Complexity	Medium	Low/Middle	Low	High	High	High
Sensor Cost	$	$$	$$ - $$$	$	$	$

Table 1.1: Comparisons of active and passive depth sensors

DARs remains prohibitive for numerous applications. As a result there is significant ongoing effort into improving the resolution, while lowering the cost of 3D sensors [34, 35, 36]. Pixel level fusion of multimodal sensors i.e. LiDAR and camera [3] or radars and camera [37], are recently being explored to improve depth resolution. There are several motivations to attain pixel level association of depth in image grid. One motivation is that depth sensors are pretty accurate and do not suffer the problem of scale, but the depth measurements are sparse compared to color images in a scene, which offer high-resolution imagery. Another motivation to obtain depth for each color pixel is for the purpose of RGB-D applications such as scene modeling, colorizing pointclouds etc. The idea of fusion is to backproject depth points from LiDARs into image plane given we know relative transformation between the two sensors. Once the sparse depth measurements are projected into image, *depth completion* problems can then be defined as completing 'missing' information of depth in the full resolution image grid, aided by color image.

Early works on depth completion [38, 39, 35] started with Middlebury dataset [40] [41] where generated depth map is sparsely sampled to simulate raw input depth. This dataset has mostly synthetic and indoor scenes, and have controlled lighting conditions. Since then, research has

moved on from the Middlebury dataset to larger datasets, NYU2 [4], SUN RGBD Dataset [42], Scan-Net [43]. These are mostly possible after the advent of RGBD depth sensors like Kinect, Intel RealSense etc. Some of the depth completion tasks using these datasets are [44, 45, 46, 47, 48]. But the datasets are mostly in indoor planar scenes, with more focus on detection, pose estimation and semantics recovery of object rather than depth super-resolution tasks. Also, the depth-range of the sensors were limited to few meters. It has been only in recent times that depth completion in outdoor scenes have gained considerable attention, with the advent of LiDARs that can promise both range and accuracy. Some of the pioneering works in the fields are [3, 49, 50].

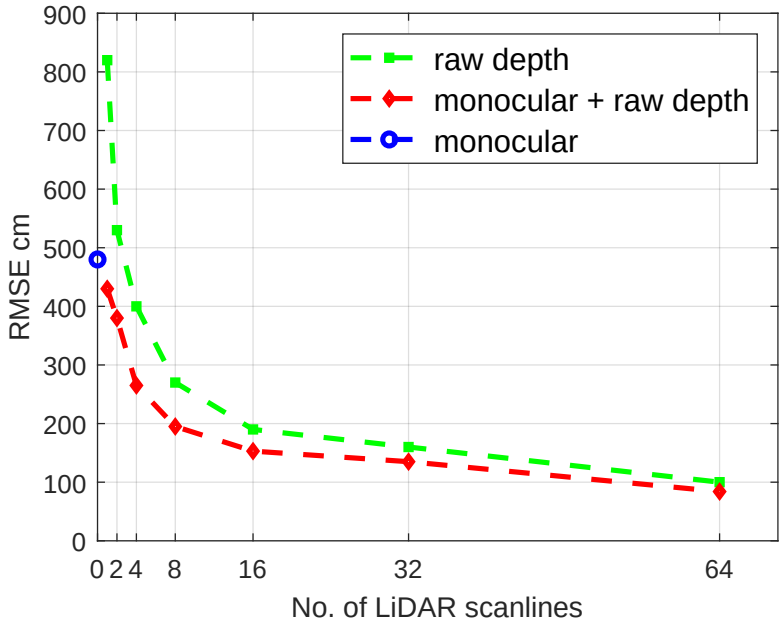


Figure 1.1: Performance comparison of depth completion method using different input sensor modalities (raw depth, monocular and monocular + raw depth). A standard deep learning method (Ma et al. [10]) is used in outdoor KITTI depth completion dataset for evaluation. The maximum depth range available for evaluation is 85m. x-axis refers to the no. of LiDAR scanlines projected to the image as raw depth, y-axis refers to the RMSE metric in cm. 0 scan-line resembles depth estimation using monocular camera only. It shows that monocular and sparse depth as input modality can reduce the performance gap by around 5 times compared to monocular camera only just by increasing the raw depth measurements. The performance gap between raw depth and multimodal input (monocular + depth) decreases as the number of raw depth measurements increase.

We choose to tackle depth completion problem by seeking to maximize the resolution of 3D

sensing and improve accuracy. We ask: given a 3D sensor, can we upgrade its depth resolution by adding a higher resolution color camera, and fusing the sensor data? Often, accuracy of estimation turns out to be a big issue depending on which input modalities we choose for estimating high-resolution depth. To put accuracy into some perspective, consider the Fig. 1.1, where we check the performance between raw depth, monocular and monocular + depth as multimodal input to a standard learning network [10]. An outdoor depth completion dataset (KITTI [1]) is used for evaluation. The maximum depth range available for evaluation is 85m. 0 scan-line refers to depth estimation using monocular camera only. It shows the performance gap can be reduced roughly by 5 times (480cm for monocular, 87cm for monocular + depth with 64 LiDAR scanlines) if multimodal sensors are used as input modality.

In short, we tackle the problem of depth completion in this dissertation using multi-modal fusion (depth sensors + monocular camera). We project sparse depth measurements from depth sensors into color image provided we have accurate extrinsic calibration parameters i.e. rotation and translation with respect to the other sensor; and estimate dense depth for each pixel in image plane, aided by color image.

1.3 Depth Ambiguity

Although depth completion has been increasingly used in wide applications in both indoor [3, 50, 48] and outdoor scenes [10, 51], the usefulness is still limited since the solutions often fail to tackle a very important problem in depth completion tasks: depth smearing across object boundaries. We now introduce the problem of depth ambiguity across boundaries and how it creates depth smearing at these discontinuous regions.

A key property of a $3D$ scene is it has step edges between two surfaces. These sharp disconti-

nuities emerge at the occluding boundaries of objects in natural scenes [52, 53]. When depth points from LiDARs are projected into uniform and dense 2D camera grid, missing pixels in the grid face an ambiguity problem for pixels at object boundaries: do these pixels belong to the foreground or background object? Failure to resolve depths at ambiguity would create depth mixing which would smear object boundaries and cause distortion in object shapes. Using high-resolution information from another modality (particularly from RGB) turns out to be useful in order to detect and recover sharp depth discontinuities. But problems of depth mixing remain and it goes worse with more sparsity. Some researches [52, 54] address the depth-mixing problem by putting regularization constraints on estimated edgemaps/discontinuity maps in the cost function. But they do not address the problem completely, since predicting the edgemaps/discontinuity maps is challenging in the first place and often not readily available in indoor/outdoor scenes.

To understand ambiguity better, lets define some key terms first in depth completion. We would like to learn model parameters θ that can fill in all the missing pixels in the 2D dense grid with depth d_i given we have sparsely sampled depth and color image x_i as input data. The learning task is supervised by groundtruth data d_{gt}. The data is defined by a probability distribution p_x. The aim of the learning task is to learn θ that best predict depth d_i, given sparse depth and color image data x_i, with distribution p_x:

$$\hat{\theta} = \arg\max_{\theta} \mathbb{E}_{p_{data}} \left[\log p_{model}(d_i | x; \theta) \right]. \tag{1.1}$$

Here the expectation is performed over training data with distribution p_{data}. The term p_{model} is the probability of estimating d_i given data x and parameters $theta$. Ideally, our model can learn perfect depth given we have infinite training data.

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

To simplify the problem, let us consider two flat surfaces perpendicular to the z axis and a depth discontinuity at its boundary. Within a surface the method can exactly estimate depth, but close to the boundary there is an ambiguity. Given sparse depth samples in both the surfaces, we can ask whether a pixel near a depth discontinuity belongs to the foreground or background surface. If the boundary is unknown, it will be ambiguous to decide whether it is foreground or background. What this means in terms of Eq. 1.1 is that given the same data x, 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. Most depth completion methods often fail to resolve this ambiguity at those regions and as a result introduce depth mixing at both the surface depths. In this dissertation we design our representations carefully to handle this case. We show that the way we address this ambiguity has important implications for depth completion problems.

1.4 Thesis Outline

The objective of the dissertation is to propose a novel depth completion method that can preserve shape and boundary of objects for use in real-life perception applications like $3D$ object detection.

Chapter 2 provides a theory of existing depth representations and introduces our novel representations of depth using multichannel (Depth Coefficients) and dual channel representations (Twin Surface). We discuss several important properties of these two representations that helps to encode ambiguity well across object surfaces or boundaries. We also propose some effective depth completion performance metrics e.g TMAE and TRMSE that can reduce preference for mixed depths and promotes sharp boundaries.

Chapter 3 digs deeper into our proposed multi-channel (Depth Coefficients or DC) representation, and how it can be learned in neural network. It shows how DC can avoid depth mixing during learning and depth reconstruction. We also propose an effective loss function that works amazingly well in learning this representation.

Chapter 4 delves deeper into learning dual-channel (Twin Surface or TWISE) representation, which also avoids issues with memory constraints in DC and improves computational efficiency significantly. We also discuss the quality of ground-truth depth in real-world depth completion problems and how accurate evaluation of methods is affected by the presence of its outliers. We find some potential pitfalls of commonly used evaluation measure i.e. RMSE, how it promotes mixed depth solutions and study the robustness of some common metrics (MAE and RMSE) in presence of outliers in groundtruth data.

Chapter 5 shows an application of our depth completion solution in $3D$ object detection. This chapter investigates whether *estimated* high-resolution depth from depth completion methods can help in object detection. It shows the effect of different depth noises originated from depth estimates on detection performances and proposes some effective ways to reduce noise in the estimate and overcome architecture limitations. The proposed findings are run on both real-world and synthetic datasets. Finally we conclude the thesis with Chapter 6 with our findings, limitations and some possible future directions of depth completion problems.

Chapter 2

Depth Representation

Depth representations are an integral component in depth completion problems and learning high-resolution depth. In this chapter, we discuss different types of depth representations that exist in literature and introduce our depth representation and the motivations behind these representations. This chapter also explains two proposed metrics that promote fairer depth completion metric comparisons by discounting mixed depth evaluation across object boundaries. We use these metrics as performance measures for evaluation of depth completion performance throughout the rest of the thesis.

2.1 Introduction

Depth data provides rich $3D$ information about the geometry of an object, hence its adequate representation is of significant importance in computer vision tasks. There are different types of $3D$ representations available, and some representations are efficient for each individual tasks like acquisition, rendering, manipulation, data analysis and even animation. But a single representation might not be effective for all tasks. We realize that representation also affects learning depth completion. Hence we seek an effective depth representation that can facilitate learning depth accurately both within objects, and across its boundaries.

Sparse depth points projected in the image plane is often called depth map, which is a single channel representation of depth in image grid. Although it is the default choice in depth com-

pletion tasks, we show that this representation fails to capture empty space between objects [55]; an interesting property of a $3D$ scene, and as a result, learning algorithms often smear depths of foreground and background objects and fill in empty space between them. We propose two novel depth representations that can model $3D$ scenes effectively, both at ambiguous and non-ambiguous regions. We also study in this chapter that conventional metrics used to evaluate depth completion algorithms are not often effective measures to evaluate depth at depth boundaries.

In summary, this chapter introduces our novel representations for depth completion; i.e. multi-channel *depth coefficients* and dual channel *twin surface* depth representations. We elaborate some interesting properties of these representations. We also propose a novel evaluation metric that tends to address the performance, *more strictly*, on depth discontinuity or occlusion boundaries. The uses of these representations in depth completion are also a contribution of the thesis and will be discussed more in the subsequent chapters.

2.2 Representations of Depth

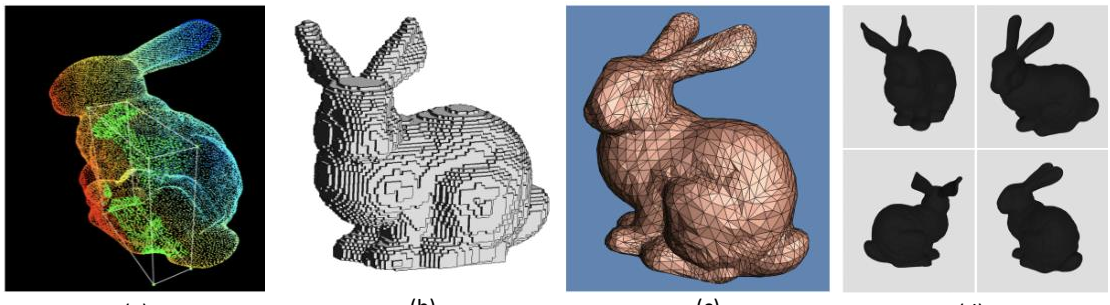


Figure 2.1: An illustration of different representations of depth; (a) represents point-cloud data (Courtesy of Caltech) (b) represents voxel grid data (courtesy of IIT Kharagpur), (c) represents triangle mesh (Courtesy of UW), and (d) represents multiview representations of depth (Courtesy of Stanford) by means of depth maps.

Shapes and sizes of $3D$ objects can be measured using active and passive depth sensors. These

$3D$ measurements are typically stored in raw pointclouds, which can later be efficiently represented multiple ways depending on application scenarios, see 2.1. Some of these formats pose new challenges to deep learning architecture models while also provide opportunity for novel and efficient solutions. Some of the most common representations include:

2.2.1 3D Point Clouds

3*D point clouds* are a collection of unordered set of sampled points on surfaces of 3D objects. Each point is encoded by its own set of *X*, *Y*, *Z* coordinates in 3D space. They are widely used in object detection [2], segmentation [56] and surface normal estimation [57]. Their advantages include registering precise locations of objects and are typically decoded directly from raw pulse signals in ToF sensors. Also, it can be realized as a small set of Euclidean subsets that have global parameterization and common system of coordinates. But learning pointcloud is challenging due to irregularly structured points and lack of connectivity information in-between points, which might cause ambiguity on distinguishing multiple surfaces close to each other. Also, pointclouds are memory intensive representation, so upsampling pointcloud of a $3D$ scene might come with memory and computational limitations.

2.2.2 Voxels

Voxels specify occupancy on a regularly spaced lattice in cartesian or polar coordinates. Each data point can be single-dimensional specifying opacity, occupancy, or consist of multi-dimensional information, such as color, probability of occupancy etc. in addition to opacity. Discrete and quantized system of coordinates can also be used to define grid-structured representation like voxels. Viewpoint information about the $3D$ shape can be encoded by classifying voxels into visible,

occluded, or empty regions. This regular grid can be used for object detection [58], object classification and orientation estimation [59]. Dense voxel grids can be memory intensive at high resolutions, since this stores both occupied and non-occupied parts of the scene. However, sparse voxel representations are also available to tackle memory issues. A more recent $3D$ volumetric representation for effeciently finding neighborhood voxels is oct-tree based [60, 59] voxels, which are typically varying-size voxels, and have the capacity to store high-resolution data by using hierarchical data structure. However, none of these voxel-based representations preserve the shape of $3D$ objects precisely. Additionally there are issues with artifacts from discretizing the surface, as well as high data and computational costs.

2.2.3 Polygon Meshes

Polygon meshes consist of a set of polygon facets with shared vertices that can approximate a geometric surface. The vertices are associated with a connectivity list which describes how these vertices are connected to each other. Polygon meshes facilitate rendering because they represent underlying geometric surfaces of objects. However, depending on the resolution, it is also memory intensive and requires high computational power for processing. Meshes can also be defined directly on image grids; but there is possibility that this representation suffer from poor definition of vertex connectivity due to ambiguous or missing information in the scene. Using regular convolutional neural networks (CNN) on raw mesh data might not be feasible because of irregular representation, connectivity issues due to edge ambiguity, different resolution at different parts of the scene and non-uniformity of data [61]. However, graph CNNs are applicable to meshes, where meshes can be represented as graph data structures [62]. Such an approach gives promising new direction over processing $3D$ data represented in meshes.

2.2.4 SingleView and Multiview Depth Maps

Singleview depth maps are 2.5D representations of 3D data, and closely reflects raw depth data in 2D grids captured from data acquisition devices e.g. ToF sensors. *Multiview Depth Maps*, are combination of multiple singleview depth images, captured from different point of views of the camera. Representing $3D$ data in this way allows learning multiple features to reduce the effect of noise, incomplete data, occlusion and illumination problems. They have been used for RGBD fusion and instance segmentation [63, 64]. This representation in regular grids can be processed with CNN in an analogous way to color image super-resolution [65, 66]. This is the representation of choice for colorization techniques and fusion [4] as well as depth completion. However the question of how many views are sufficient to model the $3D$ shape is still open. Also, sparsity of single-valued depth data in $2.5D$ representations might have adverse effects on dense convolutions [34], and might lead to depth mixing and smearing problems at boundaries, see 3.3.

2.3 Multichannel Representation of Depth: Depth Coefficients

Depth completion algorithms typically take single channel depth representation as the default choice for estimating missing depth pixels in regular 2D image grid. However, ambiguities can exist around depth boundaries, and majority of depth completion algorithms suffer at these regions by introducing smearing, i.e. mixing depth of multiple surfaces that are possible in that pixel location. To tackle ambiguity, we deem it necessary to represent depth as multi-channel representation to suggest possibilities of multiple depths.

In this section, we discuss why we opt for a multichannel depth representation to estimate accurate depths at ambiguous regions, introduce *Depth Coefficients (DC)* and discuss the properties of it. We also explained how to reconstruct depth from DC.

2.3.1 Depth Representation

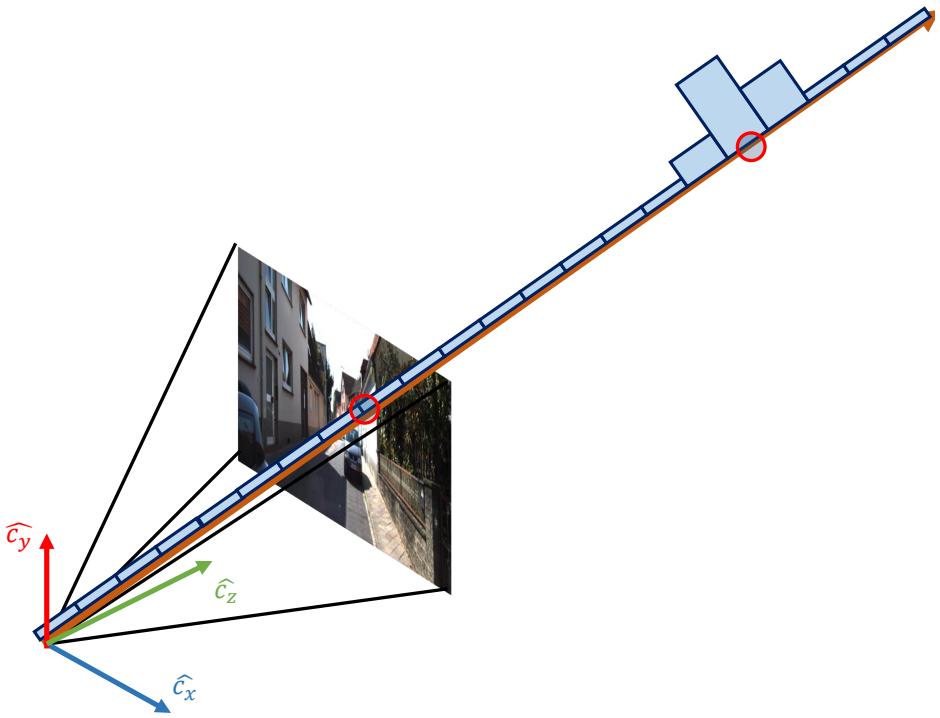


Figure 2.2: Illustration of DC. The depth pixel (red circle) in the image plane is projected back to the 3D space by a pixel ray. The depth bins are quantized along that pixel ray. Each bin is then assigned a weight (depth coefficients) on few depth bins based on the proximity of the depth with the depth bins.

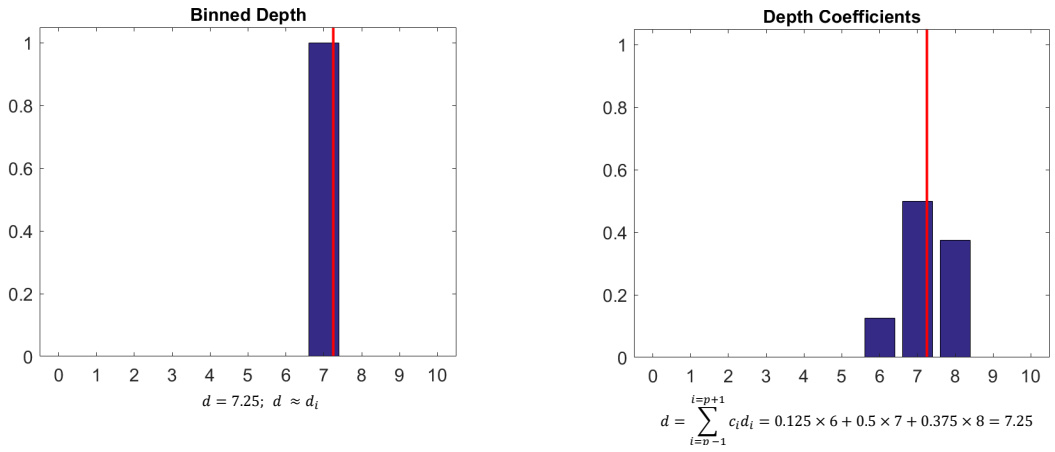
We seek a depth representation that can model depth ambiguity in a $3D$ scene, and preferably be used to resolve that ambiguity. So instead of representing depth as single value in that pixel, we use a discrete set of weights or coefficients, called *depth coefficients* (DC) which represents a single depth. Geometrically, DC represents depth along pixel rays. In simple terms, these coefficients are weights of quantized depth bins 2.3 (b) existing along the trajectory of that ray, and sum of product of these weights and the depth bins can give us depth in that pixel location. We now list several interesting properties of depth coefficients that make it a useful representation for modelling ambiguity.

2.3.1.1 Discrete Representation

Depth coefficients are weights assigned to multiple depth bins along a pixel ray to preserve the precision of a single depth value (see Fig. 2.2). With depth coefficients, depth is realized as sum of weighted bins. However, spreading the weights into all available depth bins can be memory intensive and make the learning task harder. So we seek only a finite *number* of coefficients that makes it sparse and thus can preserve both precision and memory. In doing so, we make the signal energy concentrated at possible depth bins; typically the main bin and its neighboring bins, see Fig. 2.3. These kind of bin representations already exist in literature in several other applications like tracking [67], bilateral filtering [68], channel smoothing [69], joint image alignment [70] etc. But the way we use channels/bins to represent depth is novel and unique.

2.3.1.2 Probability Representation

Ambiguities and uncertainties can be modelled by probabilities. So we seek a probabilistic representation of depth rather than single depth value in each pixel location. Depth coefficients offer discrete probability representation in quantized depth bins on any pixel location in a 2D grid. This is an important property since depth is ambiguous at depth discontinuities or object boundaries. Single modal probability distribution can be used to model ambiguity by increasing the uncertainty or standard deviation of depth. Although single modal distribution can model non-ambiguous region quite well, it is preferable to suggest multiple depth scenario by using multi-modal distribution in ambiguous regions. This is preferable since it gives the possibility to choose one depth over the other without mixing the two possible depths. Mixture of gaussians can be used for multi-modal representation, but which particular gaussian should be weighed more is not obvious. In that respect, DC can be used to model both single-modal and multi-modal distributions respectively. It



(a) (b)

Figure 2.3: Representing depth by means of multiple depth bins or *depth coefficients* to preserve depth precision. In this toy example, we divide the depth range of 10m into 10 bins, each bin spaced 1m apart from each other. (a) depth representation by a single bin, which loses precision (b) depth representation by finite number of bins which preserve precision.

is noteworthy that although we use a single modal DC to represent depth, it is possible to estimate both single modal and multi-modal DC. The higher weighted mode in the estimated DC can be used to choose the final depth.

2.3.2 Mathematical Notations

Let us now represent a dense or sparse depth image by means of *depth coefficients*. We create a multi-channel image, all of the same spatial resolution, with each channel centered at a pre-determined uniformly spaced depth; $\boldsymbol{D} = \{D_1, \ldots, D_N\}$, where D_j refers to depth of whole image at j-th channel. The depth values increase in uniform 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 cover the full range of depth up to 80m, and this determines the bin width, b; i.e. 1m apart. Thus each pixel i has a vector of values, $\boldsymbol{c}_i = [c_{i1}, \ldots, c_{iN}]$, which we call *Depth Coefficients* (DC), that encodes its depth, d_i. We constrain these coefficient vector to be non-negative, sum of all elements to 1, and give the depth as its inner product with the quantized channel depths:

$$d_i = \sum_j c_{ij} D_j. \tag{2.1}$$

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

So we use the following simple representation with minimum number of non-zero coefficients (in our case three) to represent depth. We select 3 to make the distribution of weights compact and symmetric. It also facilitates learning since there are only 3 coefficients to learn compared to 5 or higher odd number. Let k be the index of the depth channel closest to pixel depth d_i, b is the spacing between adjacent bin depths and $\delta = \frac{d_i - D_k}{b}$ be the fraction of residual depth with respect to b. D_{k-1} and D_{k+1} bins can be expressed in terms of the center bin depth 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. 2.2 leaving d_i. Considering the center bin depth has the maximum weight of 0.5, the other neighboring weights depend on the residual δ. The DC vector for pixel i is:

$$\boldsymbol{c}_i = \left[0, \ldots, 0, \frac{0.5 - \delta}{2}, 0.5, \frac{0.5 + \delta}{2}, 0, \ldots, 0\right], \tag{2.2}$$

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. (2.1), and sums to 1.

2.3.3 Depth Reconstruction from True DC

To construct depth from true DC, for each pixel, we can just use the calculated coefficients from DC and use the Eq. (2.1) as stated above. In an ideal DC with only three non-zero coefficients, it is possible to have precise reconstruction of depth from DC. However, it is possible that estimated DC has incorrect coefficients or more than three non-zero coefficients and still represent depth.

This case is illustrated in 3.4.2.

2.4 Dual Channel Representation of Depth: Twin Surface

Multichannel representation (DC) requires multiple channels and thus more memory to preserve depth precision than single channel depth. But DC has the advantage that it can represent depth ambiguities for every pixel, and can present multiple ambiguities by using multiple peaks in the form of multi-modal gaussian distribution. However, in real scenes most depth pixels have no ambiguity since they are typically from the same surface, and those that typically are ambiguous can only have two possibilities; depth from either foreground or background surface.

The phenomenon can be illustrated by Fig. 2.4 where dual peak DC suggest two possible depths. It is also possible to represent ambiguity by a dual channel depth, i.e. foreground and background depth for each pixel, which can represent minimum and maximum depth for each pixel at ambiguity. We call this a dual channel representation of depth, or twin depth. Ambiguity typically exists around boundaries of objects since it is possible to be foreground or background depth at the same pixel location. Sparse depth are likely to miss at or around boundaries, and interpolation of foreground or background depth in a single channel can cause smearing. But it is possible to model this ambiguity by using dual channel depth; each channel representing a foreground and background depth respectively. This simplistic approach can save memory dramatically and can model real-world scene in any depth without losing precision. Hence we propose a more memory efficient representation that uses just dual channels per pixel, rather than multiple channels.

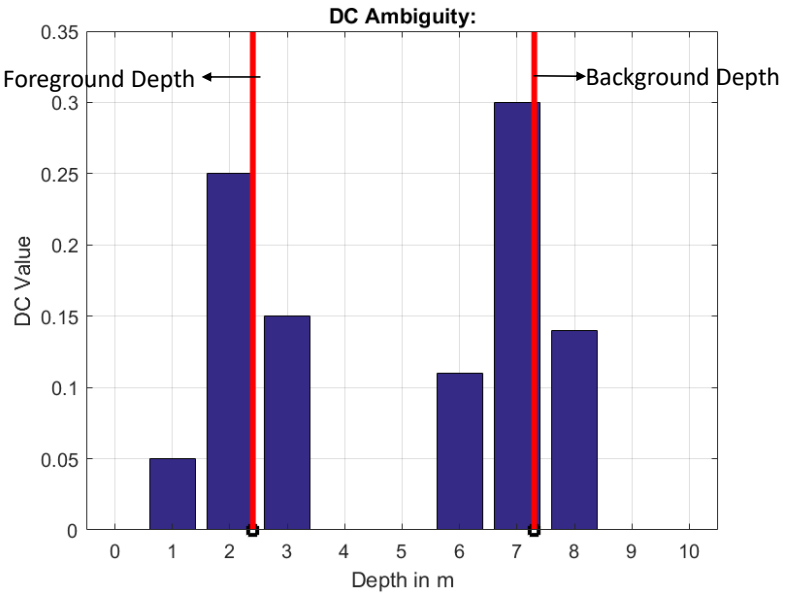


Figure 2.4: Dual Channel Representation

2.4.1 Depth Reconstruction from Foreground and Background Surfaces

Although a dual channel depth is used to model ambiguity, it is desired to reconstruct a final depth from this representation. Let us define foreground depth and background depth by d_1 and d_2 respectively. At non-ambiguous regions, d_1 and d_2 represent the same surface, and at ambiguous region, they can represent foreground and background surfaces respectively. The true depth can be represented by using the following equation:

$$d_t = \sigma d_1 + (1 - \sigma) d_2 \tag{2.3}$$

where value of σ ranges between 0 - 1. By this representation, we allow σ to choose between foreground and background surface when there is ambiguity, and can choose any value at non-ambiguous region. This reconstruction also allows some level of interpolation, within object surface by mixing the two depths.

2.5 Proposed Evaluation Metrics

While RMSE and MAE are useful metrics for overall depth completion performance, they are not effective measures of evaluating performance at sharp discontinuities across occlusion boundaries. In this section, we show why RMSE and MAE are not good metrics to evaluate the sharpness at depth discontinuity and propose a new metric to quantify it.

The two most common error metrics to evaluate depth completion tasks are MAE and RMSE respectively. Both MAE and RMSE reflect the deviation of estimated depth from groundtruth depth. While MAE calculates the mean of the absolute deviation, or error residuals of all the data, RMSE calculates the mean of the squared error residual. Since the errors are squared before they are averaged, RMSE gives a relatively high weight to large errors. This means RMSE should be more useful when large errors are particularly undesirable. In the context of depth evaluation, both these metrics are useful when depth is evaluated on the same surface, but this scenario changes when multiple surfaces exist and there are equal probability for the estimated depth to reside on any one surface.

In order to understand the implication of RMSE when multiple surfaces exist, let us consider probability of depth pixel having either foreground ($d^{(1)}$) or background surface $d^{(2)}$. The expectation of the probability, or mean of this metric occurs at the midpoint (smeared pixel) of these two depths since penalty of the estimated smeared pixel minimizes at the midpoint of the two surfaces. MAE, although less severe, offers minimum penalty for any pixel between these two surfaces. Nevertheless, mixed depth pixels are not sufficiently penalized in this metric either. The scenarios can be best illustrated in Fig. 2.5 (c).

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

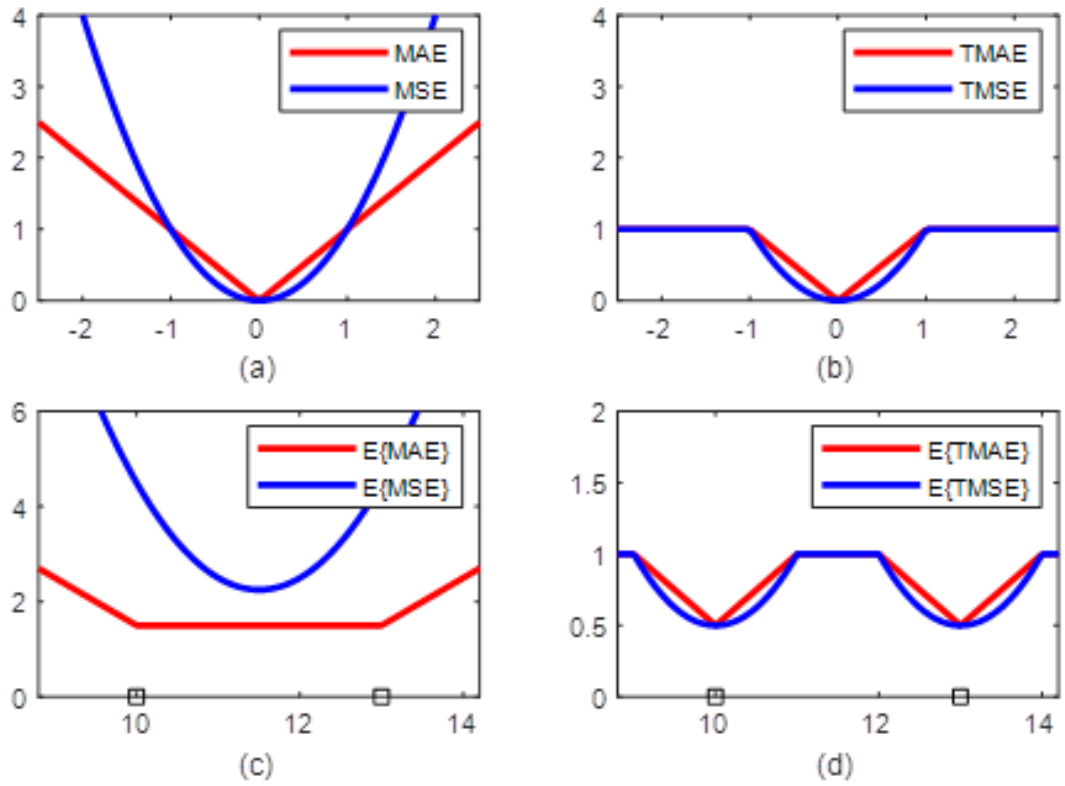


Figure 2.5: Illustration why RMSE and MAE metric are not good at boundaries. x-axis shows the $(\hat{y}_i - \tilde{y}_i)$ where $\hat{y}_i$ and $\tilde{y}_i$ are estimated depth and groundtruth depth respectively, y-axis shows the loss functions. (a) and (b) shows the metric for MSE and MAE, and TRMSE and TMAE of depth and their characteristic loss curve respectively. (c) and (d) shows the expected MSE and MAE, and expected TRMSE and TMAE when depth pixels are missing between $(10m)$ and $(13m)$ (see text). From (a) and (b), both MSE and MAE and TMSE and TMAE shows perfect prediction/no errors when it coincides with GT, but for MSE and MAE the errors continue to increase if the depth estimates are far away from the GT. However, for TRMSE and TMAE, a fixed error occurs for points beyond a threshold. From (c) and (d), the expected MSE and MAE favors estimates in-between $10m$ and $13m$, and thus, indirectly promote depth mixing. However, the TMSE and TMAE favor either points $(10m)$ or $13m$. In a way, the TMSE and TMAE do not penalize depth estimates when one surface/depth point $(10m)$ is chosen instead of the other surface/depth point $(13m)$, and only account for intra-surface variations of depth estimates.

Error (tRMSE) and Mean Absolute Thresholded Error (tMAE), defined as follows:

$$\mathrm{tRMSE} = \sqrt{\sum_{i=1}^{P} \frac{\min((\tilde{y}_i - \hat{y}_i)^2, t^2)}{P}}, \tag{2.4}$$

$$\mathrm{tMAE} = \sum_{i=1}^{P} \frac{\min(|\tilde{y}_i - \hat{y}_i|, t)}{P}. \tag{2.5}$$

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.

For TRMSE and TMAE, fixed error occurs for points beyond a threshold t, as illustrated in Fig. 2.5 (b). Considering any inter-surface variation are not more than t, any error that goes beyond t are equally penalized regardless of whether the estimated depth falls midway within the two possible surfaces or falls at the incorrect surface. As a result, mixed depth pixel is not favored more over depth pixel that estimates the incorrect surface. We argue that mixed depth pixel is equally detrimental to estimated depth pixel on the wrong surface, if not worse. Thus this metric is ideally suited for evaluation around boundaries since most of the mixed depth pixels reside around boundaries.

To illustrate the idea mathematically, consider a single pixel with a density function defined over its depth. Assume that this density is greater than zero for a set of points (corresponding to probable surfaces) and zero elsewhere. The expectation of this density function is shown in Fig.2.5 (d). All depths, separated by at least t from any probable depth, will have a fixed expectation cost of t which is greater than the cost of depths closer than t to a probable depth. Unlike RMSE, there are no local minima at mixed-depth points ($> t$ from a probable point). Unlike MAE, all mixed-depth points have greater cost than points close to probable surfaces. Hence it does not favor any mixed depth pixel over pixel that chooses the wrong surface.

2.6 Conclusion

In this chapter we introduce two novel representation of depth; multichannel depth coefficients (DC) and dual channel (twin surface), two depth representations that can model ambiguity of realistic 3D scene. Multi-channel depth coefficients is a discrete probability distribution function (discrete pdf) of depth. We show how to reconstruct depth from DC. In order to save memory requirement, we also show how a dual channel representation can be used to model ambiguity and propose a reconstruction method from these twin representation. Both these representation can potentially avoid depth mixing around boundaries of objects. We also explained the problems of conventional evaluation metrics on object boundaries and proposed a novel metric to deal with depth evaluation on boundaries. 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.

The next two chapters deal with how these two representations are used in a deep neural network model for solving depth completion problems.

Chapter 3

Depth Coefficients for Depth Completion

3.1 Introduction

This chapter focuses on using our proposed multichannel depth representation called *Depth Coefficients (DC)* in a deep learning framework for estimating dense depth pixels. We show how *DC* is able to avoid mixed depth pixels during learning. We also examine how loss functions, such as MSE, favor mixed-depth pixels in certain cases. Using our proposed DC representation, we leverage cross-entropy loss to avoid promoting depth mixing. Finally we show one utility of depth estimation using DC in $3D$ object detection. We show resolving ambiguity across boundaries can improve $3D$ object detection performance by recovering object shape and pose. Sample result is shown in Fig. 4.1.

The contributions of this section are: (1) First use of DC in a neural network (2) a new use of cross entropy as a depth loss function, (3) demonstration of improved object detection from super-resolved depth.

3.2 Related Works in Depth Completion

Evolution of Depth Completion problems with Datasets: The substantially lower resolution of depth sensors compared to color cameras has been a motivator for depth completion. Early work by Diebel and Thrun [38] used markov random fields to guide upsampling, and this was followed

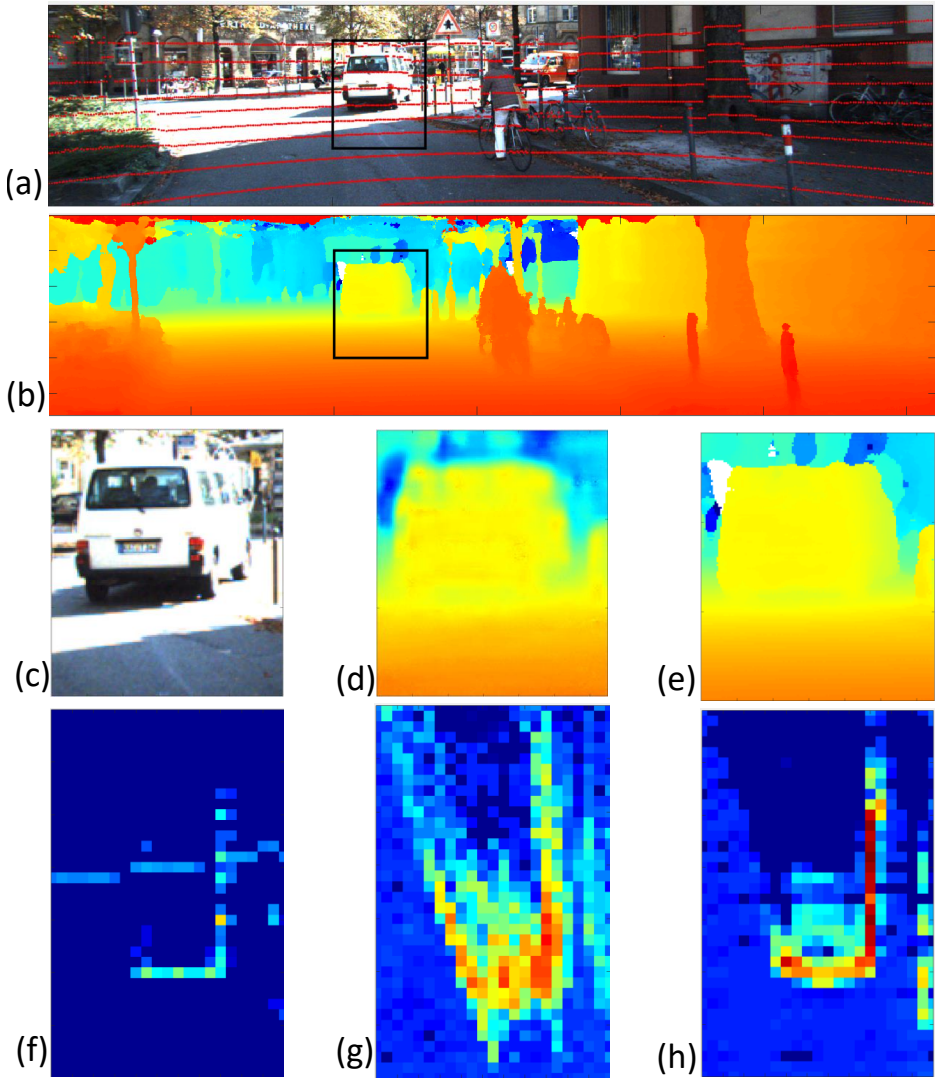


Figure 3.1: Our depth completion uses (a) a color image and the subsampled (16-row) Lidar points projected into image plane to estimate (b), a dense depth image. (c-e) are zoomed-in view of input color image, super-resolved depth of Ma et al. [3] and ours respectively. (f-h) are bird's eye view of input 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 height 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.

by a variety of improvements including bilateral filters [39], robust regularization [35, 71], hand-crafted filters [72] and image segmentation [73]. But most of the evaluation results were done on Middlebury dataset [41]. They are mostly synthetic and indoor scenes, have controlled lighting

conditions, and donot have ample data for training in deep neural networks. More recently deep convolutional neural networks (CNNs) have taken the lead and research has moved on from the Middlebury dataset [40] to larger datasets, NYU2 [4], SUN RGBD Dataset [42], Scan-Net [43]. These are mostly possible after the advent of accurate of RGBD depth sensors like Kinect, Intel RealSense etc. Some of the depth completion tasks using these datasets are [45, 46, 47, 48], and all of them used deep learning networks since there are now more training data available. But the datasets are mostly in indoor planar scenes, with more focus on detection, pose estimation and semantics recovery of object rather than depth super-resolution tasks. Also, the depth-range of the sensors were limited to few meters.

It has been only in recent times when depth completion in outdoor scenes were dealt with tremendous interest for improving $3D$ perception algorithms, mostly for autonomous driving. But mostly synthetic datasets [74, 75, 76] have dense depth maps that is exactly colocated with color imagery, and questions remain on the photo-realisticity of the scenes and different subsampling tools researchers use to downsample the dense depth maps, and thus cannot replace real world datasets [77, 78, 79]. But unfortunately all these datasets focus on 3D object detection, segmentation and tracking applications. The only realistic dataset to-date that focus on dense depth estimation tasks is KITTI [34]. Even though GT data provided by this dataset is semi-sparse, one of the main advantages of this dataset is it being realistic, outdoor scene, and enough data to train on neural networks. Most of the recent outdoor depth completion tasks [36, 3, 51] use this dataset for benchmarking. Our work uses a similar network as [3], but with focus on the depth representation and loss function, instead of the architecture.

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 [48, 3], L1 [47], inverse-L1 [36], and softmax losses on depth [50]. 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.

3.3 Avoiding Depth Mixing by Convolution

In order to define depth mixing, we first define some necessary terminologies. If the closest depth pixels of an interest point are separated by a given distance, we say there is depth discontinuity across the interest point. The *foreground* and *background* modes are defined by whether the interest point is closer (foreground mode) or further away (background mode) than each other from the sensor. It is possible that the modes have intra-surface variations due to noise or roughness of the surface (foliage, potholes in the road etc), but we claim that in order to separate between two modes/surfaces, the intra-mode depth variation should have maximum threshold t. In this section, for outdoor scenes, the t is 1m, and for indoor scenes, it is 0.1m.

We define mixed depth pixels as estimated depth pixels which are in between the two depths of foreground and background mode. Mixed depth pixels occur at empty spaces across mode boundaries. The phenomenon is best illustrated by Fig. 3.2. The consequence of depth mixing is smeared shape of the objects.

Now that we have explained *depth coefficients* and its representation in 2.3 and depth mixing, the question comes to how to incorporate DC in a deep neural network with an effective loss function that would avoid depth mixing and smearing across object boundaries. Ideally, we would like to get no depth mixing across object boundaries throughout the $3D$ scene, but this phenomena is often limited by the resolution of channel gaps in DC. Let us consider we uniformly set tm

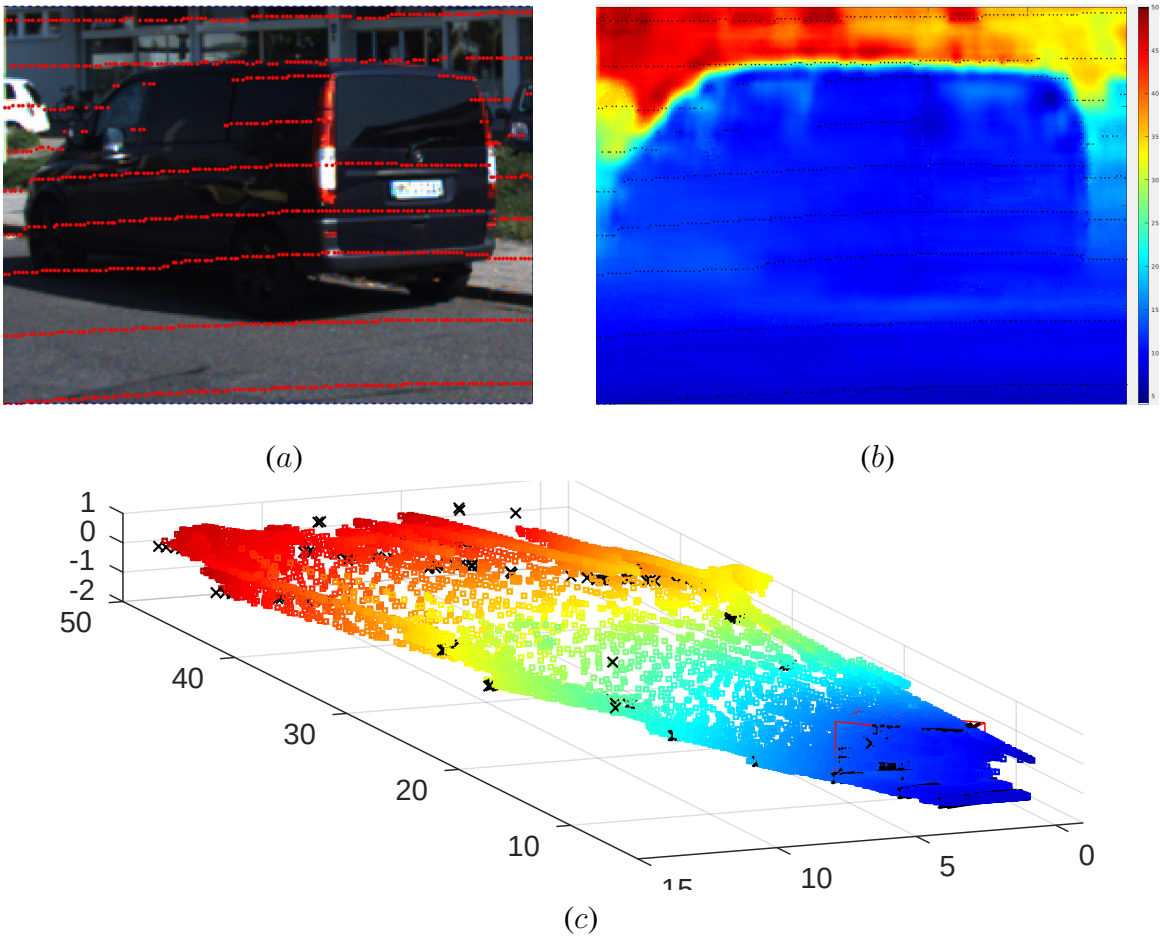


Figure 3.2: Illustration of Depth Mixing. (a) shows sparse measurements (red) in color image, (b) shows estimated depth map [10], (c) shows pointcloud generated from the depthmap (b). The black crosses in (c) are the sparse measurements. The red $3D$ box indicates the position of the car. Between the foreground mode (car) and the background mode (walls of the building), the $3D$ points floating in between are the mixed depth pixels. We say the car is smeared along its boundary.

spacing between DC channels. So our aim is to minimize depth mixing across two object depths separated by atleast tm away from each other. This section explains the motivation how DC can avoid depth mixing during convolutions; discusses further that by using DC, we can apply cross-entropy as an effective loss function to resolve depth both with-in and across object surfaces. The section then goes on to give an overview of the deep learning architecture.

3.3.1 Motivation

The fact that dense convolutions with DC can avoid depth mixing can be explained by a motivating example (see Fig. 3.3). Consider two planar depths in a depth image, separated by $5t$m away from each other. We take a slice of the depth image, and sparsely subsample the $1D$ depth signal. Now if we run a $1D$ convolving FIR filter on this signal, it will mix the depth of two planar depths across depth discontinuity. Now consider the case when the sparse depth signal is converted to DC. This time the DC channels are spatially separated by tm in depth. This time we run the $1D$ FIR filter on DC along *DC bins*; now the missing weights/coefficients are interpolated only between neighboring bins and as a result, there is no mixing between weights of spatially separated channels more than tm away from each other.

Similarly, 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, 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 [80, 81] which avoid mixing spatially-distant voxels. This effect is illustrated in Fig. 3.3(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

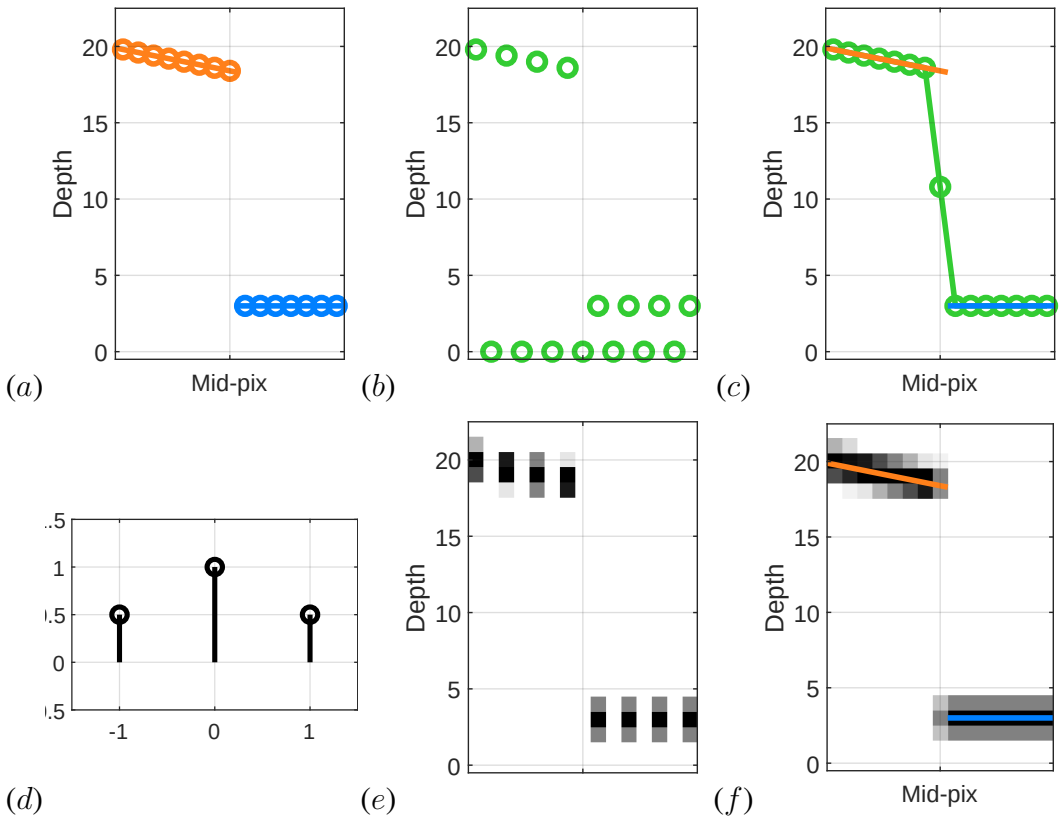


Figure 3.3: An example of depth mixing, and how DC avoids it. (a) A slice through a depth image showing a depth discontinuity between two objects. (b) An example sparse depth representation: each pixel either has a depth value or a zero. (c) The result of a 1D convolution, shown in (d), applied to the 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.

be modeled and learned by neural networks.

3.3.2 Proposed Loss Function

Once we are motivated that dense convolutions can avoid depth smearing on DC channels, we explore some of the loss functions to optimize our neural network parameters. We found that designing an optimization loss over GT DC rather than GT depth is also vital for avoiding depth smearing across boundaries. We discuss in this section why some of the conventional loss functions used for depth estimation encourage depth smearing across boundaries. We then propose cross-

entropy loss to address this phenomena.

3.3.2.1 Depth Loss Functions with Ambiguity

One of the more popular loss functions in depth completion tasks is Mean Squared Error (MSE). In part this is because the MSE gives the maximum likelihood solution to Eq. 1.1 when $p_{model}(d_i|x;\theta)$ is gaussian. We consider the simple case when data is gaussian. The maximum likelihood optimizer gives an optimum estimator that is close to the mean of data x.

Now we consider the implications of using MSE when there are depth ambiguities. We define ambiguity as regions across depth discontinuity when there are probability of a depth pixel having either foreground and background mode as defined in 1.3. Given there are two modes from data x, the foreground mode having depth $d^{(1)}$ and the background mode having $d^{(2)}$. Given multi-modal density function from data x, the MSE loss for depth d_i at pixel i is:

$$\text{MSE}(d_i) = \frac{1}{2}\left(||d - d^{(1)}||^2 + ||d - d^{(2)}||^2\right), \tag{3.1}$$

which is minimum when

$$\hat{d_i} = \frac{1}{2}(d^{(1)} + d^{(2)}). \tag{3.2}$$

And so the estimated depth pixel is a mean of the foreground and background depths. An illustration of this is in Fig. 3.4(b). This solution is only good when the foreground and background depth are coming from within the same surface (e.g. road surface, walls etc) since regression/interpolation is desirable within same surface, but the solution is undesirable when the depths are coming from different surfaces.

Mean Absolute Error (MAE), has a similar issue, yet not as severe. As in Fig. 3.4, in the pairwise ambiguity case, the MAE loss of mixed-depth pixels is equal to the loss at the actual

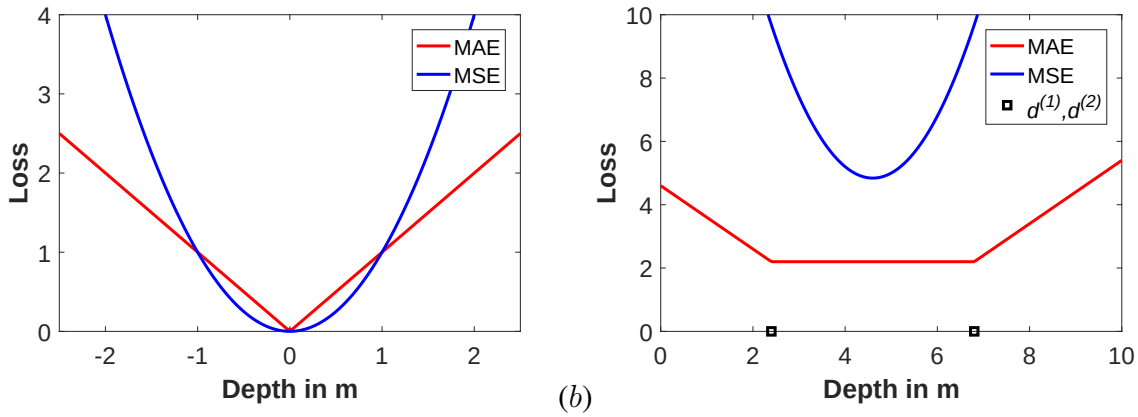


Figure 3.4: (a) shows MSE and MAE loss functions. These perform an expectation over the probability of the data. Now consider an ambiguous case where a pixel's depth has equal probability being $d^{(1)}$ or $d^{(2)}$, shown as black squares in (b). Minimum MSE estimate, $\hat{d}$, is the mid-point, while 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.

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.

3.3.2.2 Cross Entropy as Loss Measure

As shown in Sec. 3.3.2.1, minimizing MSE leads to depth mixing when there is depth ambiguity (note that ambiguity only exists at inference when we estimate depths on each pixel given sparse depth measurements, and possibly, color). One way to avoid this is, rather than estimating depth directly, we can 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. 1.1. 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 rather than learn to estimate d_i, Our cross-entropy loss for pixel i is defined as:

$$L_i^{ce}(c_{ij}) = -\sum_{j=1}^{N} c_{ij} \log \tilde{c}_{ij}, \tag{3.3}$$

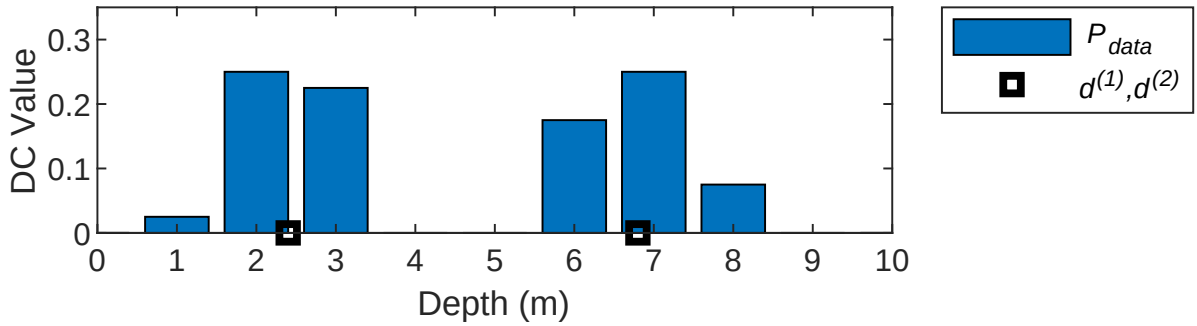


Figure 3.5: An illustration of P_{data} modeled as the sum of the DC of the two points from Fig. 3.4. The estimated $\hat{c}_{ij}$ with minimum cross-entropy loss, Eq. 3.3, will exactly match P_{data}, providing a multi-modal density. A pixel depth estimate using Eq. 3.4 will find the depth of one of the peaks, and not a mixed-depth value.

where c_{ij} terms are the DC elements of the ground truth obtained using Eq. 2.2. Training a network to predict $\tilde{c}_{ij}$ that minimizes L_i^{ce} is equivalent to maximizing Eq. 1.1.

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. 3.5, 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.

3.4 Learning by Deep Neural Network

Now that we are certain that we need to inject sparse DC and estimate dense DC in and out of a deep learning model, we set our sight as to what is a good network architecture that can incorporate DC naturally. Also we need a model to recover dense depth from estimated DC. This section describes in detail our deep learning framework to estimate depth from DC. The first part gives an overview of the model, the second part explains the neural network architecture of our model, and the third part explains how we recover depth from estimated DC.

The three major blocks in our deep learning framework are (a) Sparse Depth 2 DC block (b)

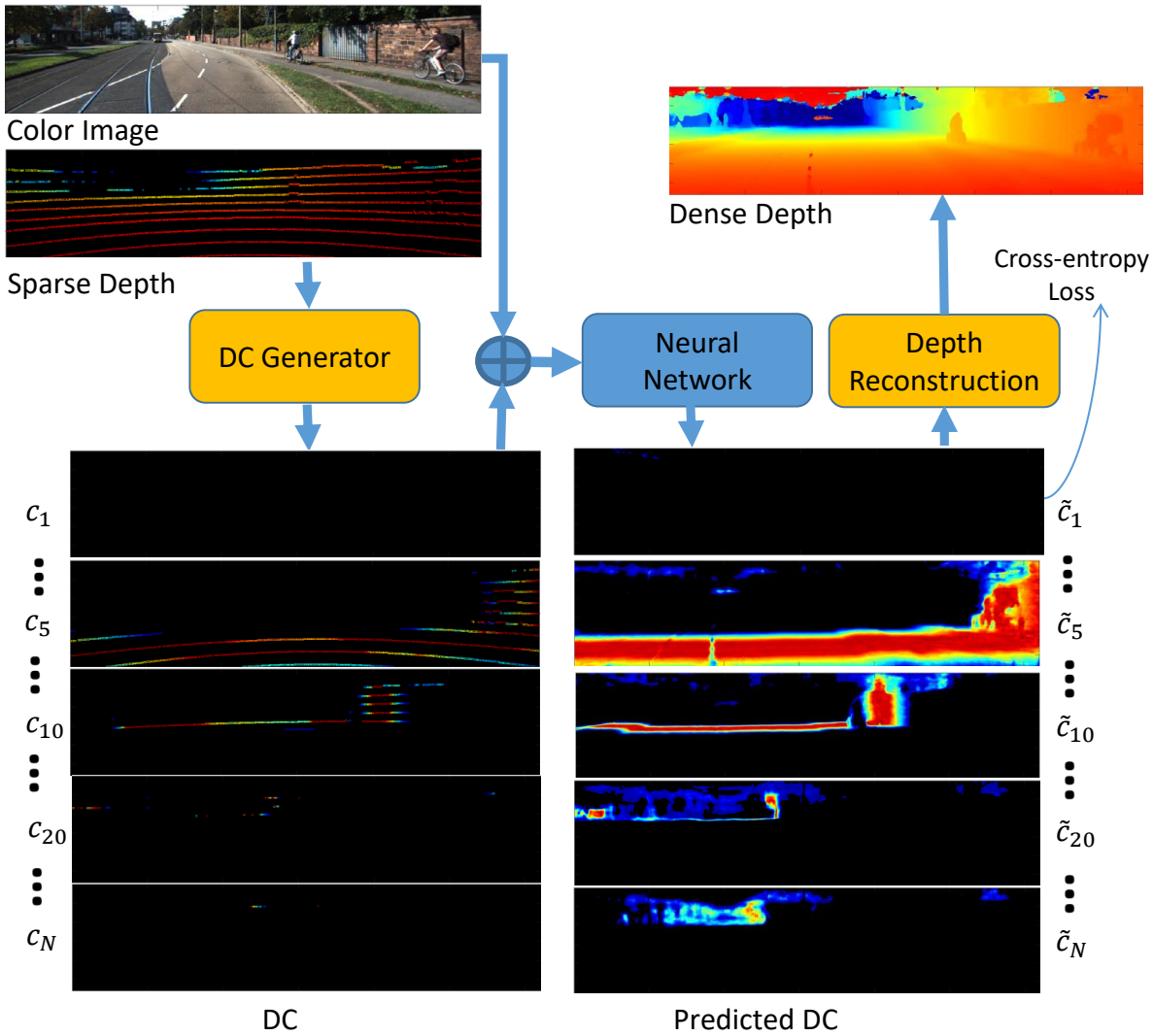


Figure 3.6: An overview of our method. Sparse depth is converted into Depth Coefficients with multiple channels, each channel holding information of a certain depth range. This, along with color, is input to the neural network. The output is a multi-channel dense depth density that is optimized using cross entropy with a ground-truth DC. The final depth is reconstructed based on the predicted density.

Neural Network block (c) Dense DC to Depth block as shown in Fig. 3.6. The sparse depth measurements are first converted to sparse DC. This operation is non-differentiable though. The deep network estimates DC, and we optimize the solution on GT DC converted from ground-truth depth based on cross-entropy loss. Finally, we convert dense DC estimate to dense depth estimate.

3.4.1 Neural Network Architecture

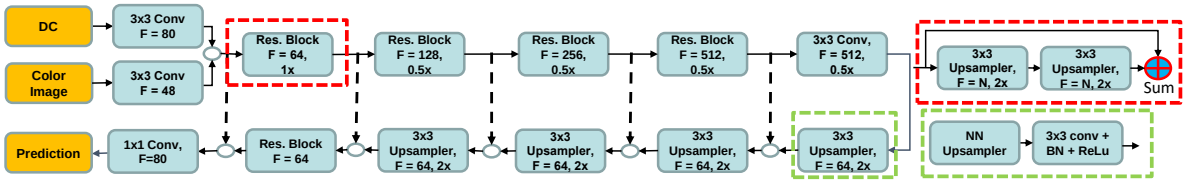


Figure 3.7: Our CNN architecture modified from [3] with 80-channel DC at the input, and 80-channel cross-entropy loss. The residual blocks are defined in the dashed red region, and the top branch consists of ResNet 34 [11].

We selected a standard network for depth completion [3], and modified the input and output. Single channel depth is first converted to 80C DC. On the input, $80/48$ channels of DC and color were then fed into the initial convolutions respectively 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. It is straightforward to convert a depth network into a DC network using this strategy. The downside, though, is feeding in more channels (80 in our case), creates more memory requirements. The estimated DC output of the network is trained using cross entropy loss on a DC representation of semi-dense depth.

3.4.2 Depth Reconstruction

Since we optimize the neural network parameters based on GT DC, it is possible that the estimated DC infers mulimodal density functions at ambiguous pixels. The case is best illustrated in Fig. 3.5. The question is then how to recover depth from estimated DC.

There are a number of options for depth reconstruction from DC. We can use Eq. 2.1, and substitute $\hat{c}_{ij}$ for c_{ij} for pixel i. However, the predicted coefficients may be multi-modal as in Fig. 3.5, and it may be preferable to estimate the maximum likelihood solution. We now show that representing depth in depth coefficients is guaranteed to avoid depth mixing if we only select the

main bin (peak of the probability vector) and its two closest bins (far and near bin) to construct the depth. This step is key to cut off interactions with far away bins and avoid depth mixing. We found out that near object boundaries, the DC vector of that pixel has estimated weights typically spanned over large number of bins (large uncertainty along boundaries). So to avoid depth mixing, we detect the peak bin and its associated nearby bins to reconstruct the depth.

We can estimate the depth for the peak via the maximum coefficient $c_{ik} \in \boldsymbol{c}_i$ and its two neighbors. This gives us:

$$\hat{d}_i = \frac{\hat{c}_{i(k-1)} D_{(k-1)} + \hat{c}_{ik} D_k + \hat{c}_{i(k+1)} D_{(k+1)}}{\hat{c}_{i(k-1)} + \hat{c}_{ik} + \hat{c}_{i(k+1)}}. \tag{3.4}$$

3.5 Experiments and Results

3.5.1 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 [34] 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 [47]. 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 spatial dimension of 304×208. For comparison purposes, we choose the state of the arts in both outdoor [3] and indoor scenes [47, 51] using RGBD depth sensors.

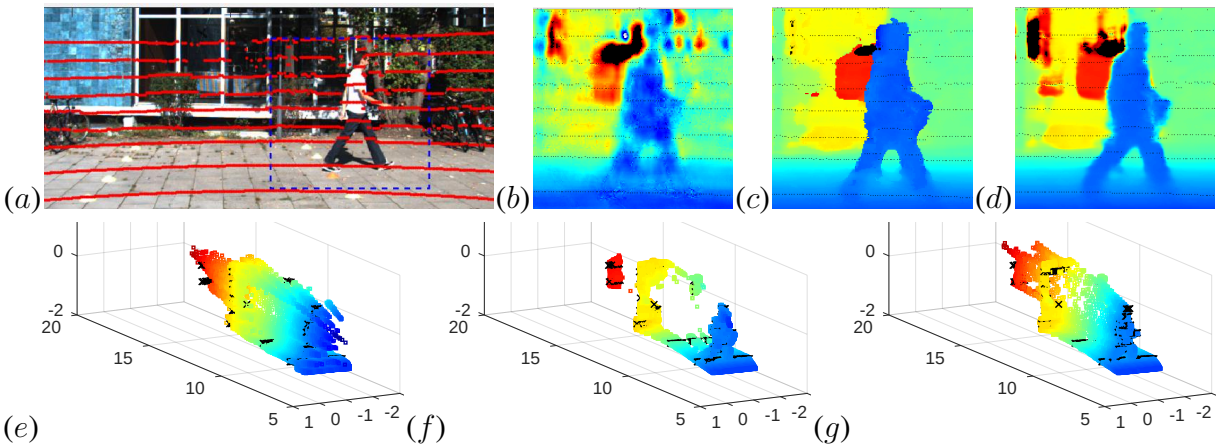


Figure 3.8: Depth completion with 16-row Lidar. (a) scene, (b, e) show Ma *et al.* [3] with significant mixed pixels. (c, f) show our 3-coefficient estimation, demonstrating very little depth mixing. (d, g) show our estimation with all coefficients.

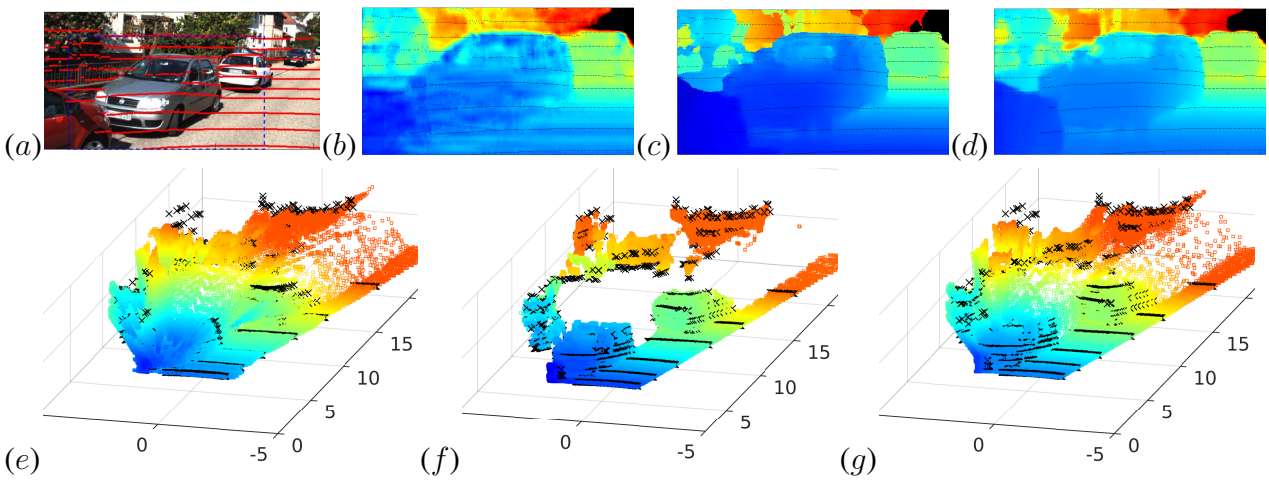


Figure 3.9: Another depth completion example with 16-row Lidar, where all subfigures are defined the same as Fig. 3.8. Interestingly, higher RMSE is reported on 3-coefficient estimation as opposed to all-coefficient estimation.

3.5.1.0.1 Sub-Sampling 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

Method	RMSE	MAE	REL	tMAE	tRMSE	δ_1	δ_2	δ_3	δ_4	δ_5
Ma [47]	0.236	0.13	0.046	0.068	0.075	52.3	82.3	92.6	97.1	99.4
Bilateral [82]	0.479	-	0.084	-	-	29.9	58.0	77.3	92.4	97.6
SPN [83]	0.172	-	0.031	-	-	61.1	84.9	93.5	98.3	99.7
Unet [51]	0.137	0.051	0.020	-	-	78.1	91.6	96.2	98.9	99.8
CSPN [51]	0.162	-	0.028	-	-	64.6	87.7	94.9	98.6	99.7
CSPN+UNet [51]	**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 3.1: Quantitative results of NYU2 (Done on Uniform-500 Samples + RGB) (units in m).

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 3.2: 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 3.3: 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.

reporting this. Likely mixed-depth pixels have a large negative impact on object detection. Indeed, Tab. 3.4 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.

	3D Bounding Box			Bird's Eye View Box		
Upsample:	Easy	Med.	Hard	Easy	Med.	Hard
Raw 16R	54.4	36.2	31.3	73.6	**58.1**	**50.4**
Ma [3]	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 3.4: Average precision (%) for 3D detection and pose estimation of cars on KITTI [1] using Frustum PointNet [2]. The baseline, Raw-16R, uses 16 rows from the Lidar, while Ma's method [3] 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 [1] 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.

3.6 Conclusion

In this chapter, we introduce *depth coefficients* (DC) in a neural network model. On the input, DC represents 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. Also using a similar strategy (using a depth-to-DC and DC-to-depth converter), it is possible to incorporate DC in any neural network architecture. Indeed we show that, unlike other upsampling methods, our dense depth estimates can improve object detection compared to sparse depth.

Chapter 4

Depth Completion Using TWIN Surface Extrapolation at Occlusion Boundaries

4.1 Introduction

In the previous chapter, we show how multi-channel depth representation can be used to model depth ambiguity at object boundaries or step-like discontinuities. It is important to maintain depth discontinuities to facilitate object shape and pose estimation. However, it has high computational and memory demand for accommodating many channels at high 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 are 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 [49]. 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 VKITTI dataset.

In summary, this chapter focuses on a twin-surface representation that can estimate foreground, background and fused depth. We also design a pair of assymmetric loss functions that can explicitly predict foreground-background object surfaces that can be used in any neural network learning paradigm. We discuss further that in presence of outliers in ground-truth, MAE is a more consistent metric to rank methods compared to RMSE, and we validate this claim with extensive experiments in VKITTI, a synthetic dataset for urban driving scenario. Finally we show the effectiveness and superiority of our method by comparing with SoTA method on both challenging outdoor and indoor scenarios.

4.2 Related Works

4.2.1 Depth Completion

Deep neural networks (DNNs) have been applied to the depth completion problem, in works such as Sparse-to-Dense [10], DDP [84], and Spade RGBsD [36]. These works show that by using standard encoder-decoder architecture (ResNet and MobileNet), it is possible to improve depth

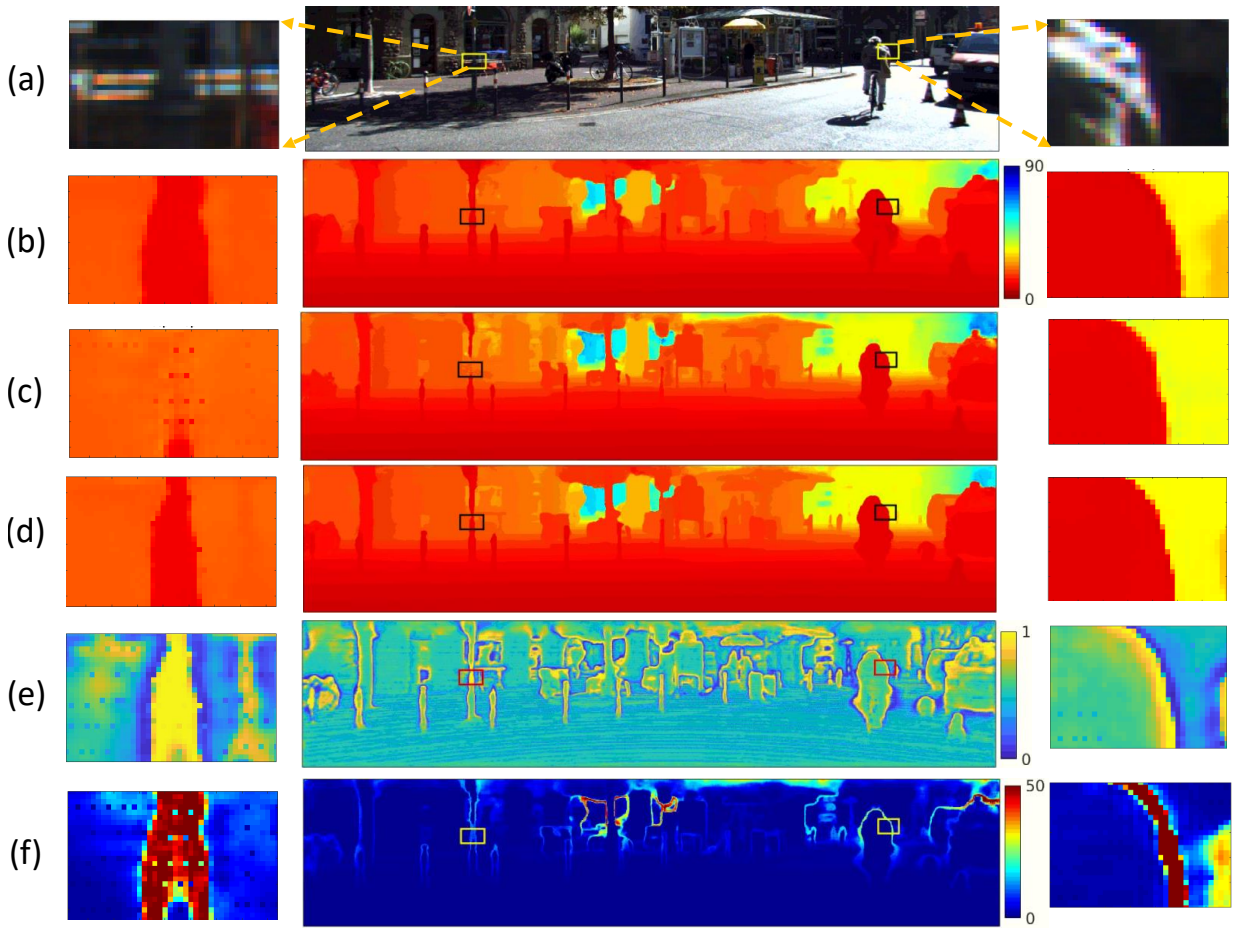


Figure 4.1: Our depth completion algorithm can input LiDAR data and image (a), and extrapolate the estimates of foreground depth d_1 (b) and background depth d_2 (c), along with a weight σ (e). Fusing all three leads to the completed depth (d). The foreground-background depth difference (f) $d_2 - d_1$ is small except at depth discontinuities.

estimation accuracy via regression losses like L_2, L_1 and inverse L_1 losses. Deep-Lidar [85] estimates surface normal and dense depth using multiple DNNs to assist in further fine-tuning dense depth. Both [85] and [84] 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 [86, 87] 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 [87]. Another recent trend is to learn spatial propagation of pixels in 2D depth space for depth completion problems in fixed [88] or variable receptive field [13, 89]. Although re-

sults 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 [90], point cloud completion [91], 3D graph neural networks [92] for dynamic construction of local neighborhood regions.

4.2.2 Depth Representations

Depth maps, as 2.5D representations, have been used for RGBD fusion and instance segmentation [63, 64]. 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 [93, 94]. This is the representation of choice for colorization techniques and fusion [4] 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.* [95] 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. Tulsiani *et al.* [96] 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.* [97] 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, multi-layered 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.

4.2.3 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 [48, 10], L_1 [47], inverse-L_1 [36], Huber loss [90] and Softmax loss on depth [50]. Another elegant way is to use combination of L_1 + L_2 [13], 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 [91], depth-normal constraint [87], Cosine loss [85], 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 assymetric loss functions, and the final depth using a fusion loss. The assymetric loss function has been used in Vogel *et al.* [98], for the different purpose of denoising input images. We propose to use assymetric 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.

4.3 Ambiguities and Expected Loss

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.

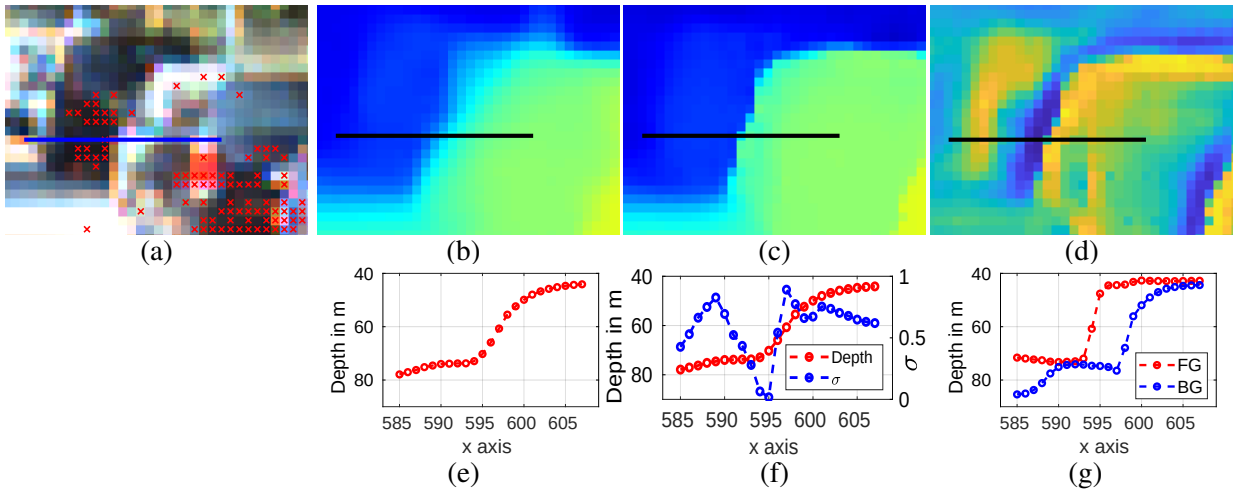


Figure 4.2: Depth smearing across boundaries. We show the ground truth depth (colored red) overlaid on an image (a), depths estimated by the SoTA method [6] (b), our fused depths (c), our estimated weights σ (d), a depth slice of [6] (e), fused depth and σ slices (f), and foreground and background slice (g). Our extrapolation ability in (g) results in the sharp depth boundary in (f), rather than the smeared depth in (e).

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. 4.2.

4.4 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. 4.2.

4.4.1 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.

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_i, 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, $L(d - d_t)$, where d_t is the ground truth depth. The *expected loss* as a function of depth is:

$$E\{L(d)\} = \sum_i p_i L(d - d_i). \tag{4.1}$$

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.

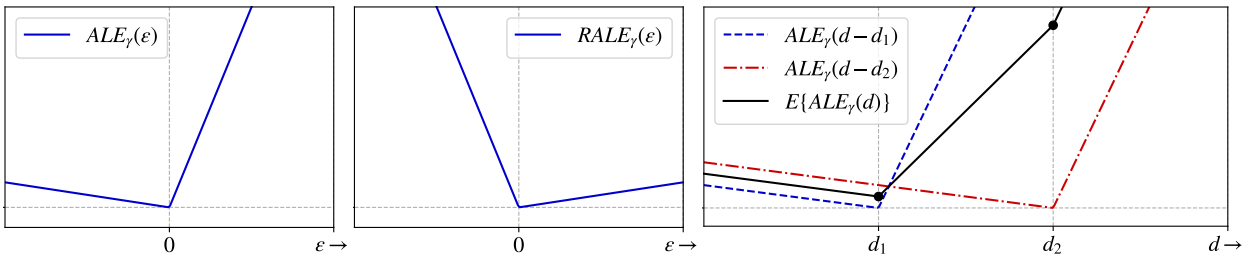


Figure 4.3: (a) The *ALE* from Eq. (4.2) is asymmetric around its minimum at the origin. (b) The *RALE* from Eq. (4.3) is a reflection of the *ALE*. We use the *ALE* for foreground surface estimation and the *RALE* for background estimation. (c) A pixel depth is shown with two ambiguities at depths d_1 and d_2 and probabilities $p1$ and $p1$ respectively. The black line shows the expected *ALE* which is the probability-weighted sum of two *ALE* functions, see Eq. (4.1). The expected *ALE* will have a minimum at one of the marked corners occurring at d_1 and d_2. The minimum will be at d_1 if Eq. (4.4) is satisfied, as it is in this case with $p_1 = p_2$, and so acts as a foreground depth estimator.

4.4.2 Asymmetric Linear Error

Our method uses a pair of error functions which we call the Asymmetric Linear Error (*ALE*), and its twin, the Reflected Asymmetric Linear Error (*RALE*), defined as:

$$ALE_\gamma(\varepsilon) = \max\left(-\frac{1}{\gamma}\varepsilon, \gamma\varepsilon\right), \tag{4.2}$$

$$RALE_\gamma(\varepsilon) = \max\left(\frac{1}{\gamma}\varepsilon, -\gamma\varepsilon\right). \tag{4.3}$$

Here ε is the difference between the measurement and the ground truth, γ 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 $\frac{1}{\gamma}$ and the positive weighted by γ. The *RALE* is simply the reflection of the *ALE* over the $\varepsilon = 0$ line. Both are illustrated in Fig. 4.3 (a,b).

Note that if γ is replaced by $\frac{1}{\gamma}$, both the *ALE* and *RALE* are reflected. Thus, without loss of generality, in this work we restrict $\gamma \geq 1$.

4.4.3 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 $\hat{d_1}$ from a trained network at ambiguous pixels, we examine the expected *ALE*, as shown in Fig. 4.3 (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. (4.2) and (4.1), 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:

$$\gamma > \sqrt{\frac{p_2}{p_1}}. \tag{4.4}$$

This equation shows the sensitivity of the foreground estimator to γ; the higher γ, 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 γ as in Eq. (4.4), except that the probability ratio is inverted.

Fig. 4.1 (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.

4.4.4 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 $\hat{d}_t$ for the true depth d_t as a weighted combination of the two depths:

$$\hat{d}_t = \sigma \hat{d}_1 + (1 - \sigma)\hat{d}_2. \tag{4.5}$$

where σ is an estimated value between 0 and 1. We use a mean absolute error as part of the fusion loss:

$$F(\sigma) = |\hat{d}_t - d_t| = |\sigma \hat{d}_1 + (1 - \sigma)\hat{d}_2 - d_t|. \tag{4.6}$$

The expected loss for this is

$$\begin{aligned} L_e(\sigma) =& E\{F(\sigma)\} = p|\sigma \hat{d}_1 + (1 - \sigma)\hat{d}_2 - d_1| + \\ & (1 - p)|\sigma \hat{d}_1 + (1 - \sigma)\hat{d}_2 - d_2|. \end{aligned} \tag{4.7}$$

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. (4.7) to predict a separate σ 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 σ is shown in Fig. 4.1 (e).

4.4.5 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:

$$L(c_1, c_2, c_3) = \frac{1}{N}\sum_{j}^{N}(ALE_{\gamma}(c_{1j}) + RALE_{\gamma}(c_{2j}) + L_e(s(c_{3j}))). \tag{4.8}$$

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 $c_1 \rightarrow \hat{d}_1$, $c_2 \rightarrow \hat{d}_2$ and $s(c_3) \rightarrow \sigma$, and combine them as in Eq. (4.5) to obtain a depth estimator $\hat{d}_t$ for each pixel.

4.4.6 Implementation Details

4.4.6.1 Architecture

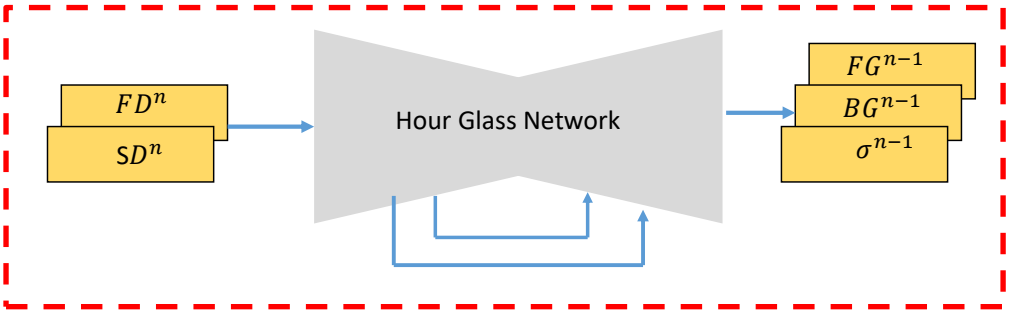


Figure 4.4: Incorporating 3-channel at the output of the Hour-glass network used in [6]. SD^n and FD^n are the sparse inputs and fused depth obtained from FG^n, BG^n, and σ^n at multi resolution scale n respectively.

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 [6], with the author-provided code. The only modification we made are at the last layer of the network, where we used three channels representing d_1 (foreground estimate), d_2 (background estimate), and σ (see 4.4). We repeat this strategy in the hourglass networks in all the three multi-resolution levels. Please see [6] for more details of the network. We choose this network due to its fast inference time, lower number of parameters than [10], 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.

4.4.6.2 Training and Inference

We followed the training protocol in [6] with multi-scale supervision on our 3 channels. The total loss is a weighted sum of the multiple resolution losses L_i, where L_1 is the full resolution 3-channel loss in Eq. (4.8), 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 $\omega_2 = \omega_3 = 0$ and complete training after 30 epochs. Using Adam optimizer with an initial learning rate of $10e-3$ and decrease to half every 5 epochs, we train a full sized image with gradient accumulated every 4 samples in a batch. We use PyTorch [99] for our implementation.

Method	MAE	RMSE	iMAE	iRMSE	TMAE [12]	TRMSE [12]	Infer. time (sec.)
Ma *et al.* [10]	249.95/269.2	814.73/878.5	1.21/1.34	2.80/3.25	–/190.15	–/297.48	0.081
Depth-Normal [87]	235.17/236.67	777.05/811.07	1.79/1.11	2.42/2.45	–/–	–/–	–
DeepLidar [85]	226.50/215.38	758.40/**687.0**	1.15/1.10	2.56/2.51	–/162.75	–/266.79	0.097
3DepthNet [91]	226.2/208.96	798.40/693.23	1.02/0.98	2.36/2.37	–/–	–/–	–
Uber-FuseNet [90]	221.19/217.0	752.88/785.0	1.14/1.08	2.34/2.36	–/–	–/–	–
MultiStack [6]	220.41/223.40	762.20/798.80	0.98/1.0	2.30/2.57	–/157.90	–/270.15	**0.018**
DC-3co [12]	215.75/215.04	965.87/1011.3	0.98/0.94	2.43/2.50	–/141.67	–/238.5	0.112
CSPN++ [88]	209.28/–	743.69/–	0.90/–	2.07/–	–/–	–/–	0.200
DDP [84]	205.40/–	836.00/–	0.86/–	2.12/–	–/–	–/–	–
NLSPN [13]	199.59/198.64	**741.68**/7f71.8	0.84/0.83	**1.99/2.03**	–/138.81	–/248.88	0.225
TWISE	**195.58/193.40**	840.20/879.40	**0.82/0.81**	2.08/2.19	**–/131.60**	**–/239.80**	0.022

Table 4.1: Depth completion on the Test/Validation sets of KITTI, with 64R LiDAR and RGB input (units in mm).

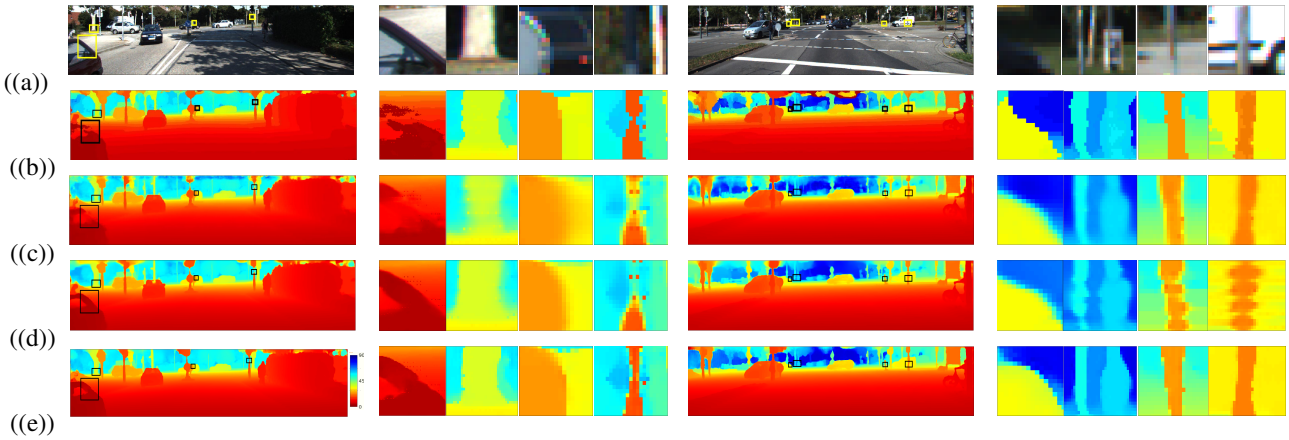


Figure 4.5: Comparison of our method with SoTA methods with whole and zoom in views (a) showing Color Images (b) DC [12], (c) MultiStack [6] (d) NLSPN [13] and our method (e). Four different regions of the image from two different instants are selected to show depth quality from diverse areas.

4.5 Experimental Results

4.5.1 Dataset

We evaluate the proposed algorithm on the standard KITTI Depth Completion dataset [1], a real-world outdoor scene, NYU2, with indoor scenes [4], and Virtual KITTI [100], 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 [84, 85] 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.* [49] has far fewer depth points on object boundaries (see Fig. 4.2 (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^o left, 15^o right, 30^o left, 30^o 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 [49] to study the effects of outlier noise on training and evaluation. More details are shared in the supplementary.

To show the generalizibility of our method, we also evaluate on NYU-Depth v2 dataset [4], 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 [85, 13]. 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 304×208. Unlike [13], we use the *same* loss function for all the datasets.

4.5.2 Metrics

The standard metrics used by KITTI include RMSE, MAE, iMAE and iRMSE. Since RMSE is used as 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.

4.5.3 Results

4.5.3.1 Quantitative Results

Tab. 4.1 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 [13] 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 are our method perform better and worse, which we examine next.

4.5.3.2 Qualitative Results

Fig. 4.5 shows our depth estimation quality compared to baselines. We choose three best SoTA methods: MultiStack [6], NLSPN [13], 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. [6] has a more blurred estimation around boundaries leading to mixed

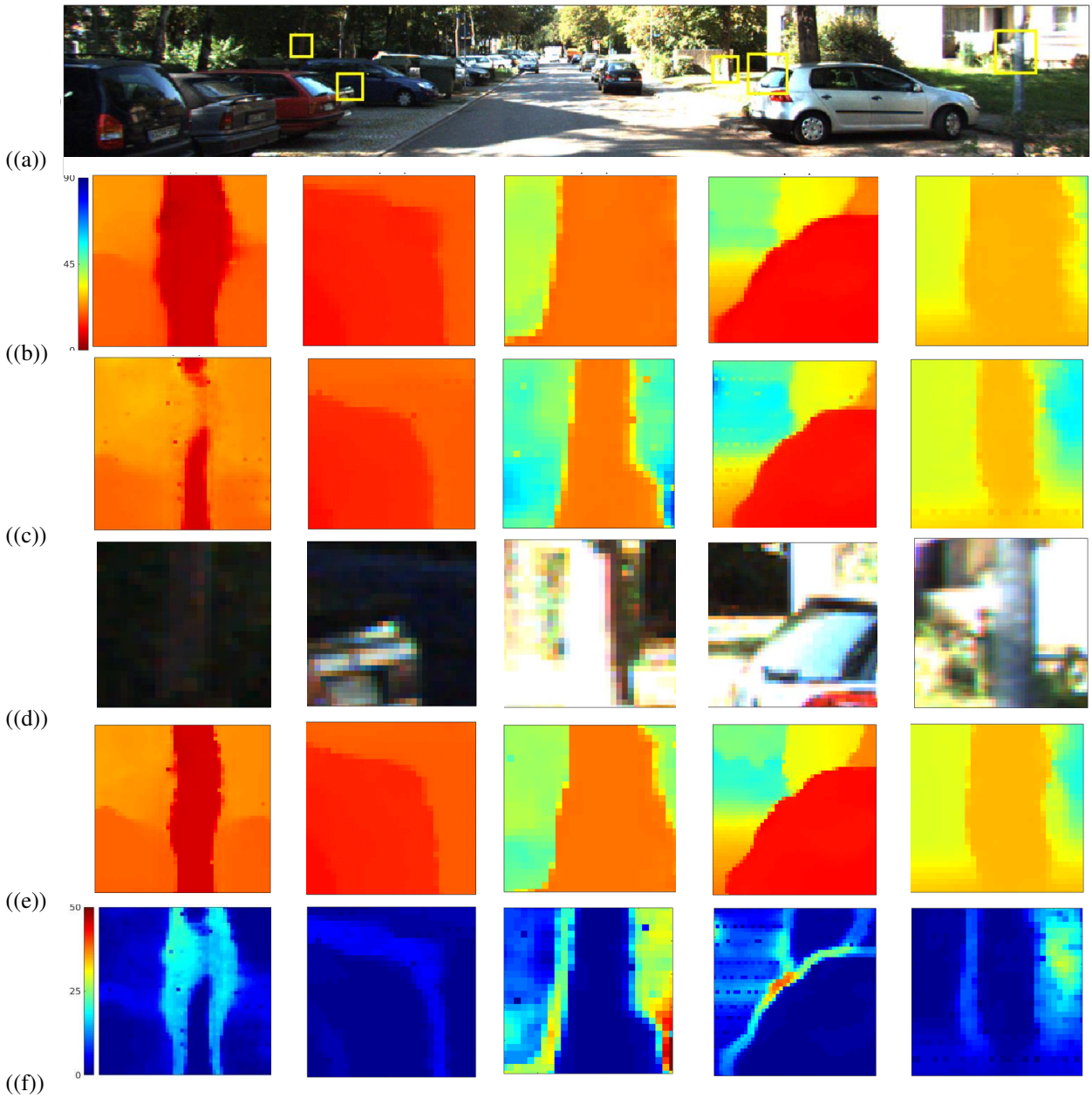


Figure 4.6: Input image (a), its zoom-in views (d), our estimation on foreground depth (b), background depth (c), fused depth (e), and the depth difference between foreground and background depth (f).

depth pixels and holes within objects, such as on the traffic poles and van. Although [13] 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.

4.5.3.3 Qualitative Parsing

Fig. 4.6 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. 4.6 (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 of GT depth pixels.

4.5.3.4 Relative Error Maps

It is worthwhile to examine where our method has lower errors in comparison with our baseline method [6] which uses MSE. Two types of errors are examined: the absolute difference between estimated depth and GT, and its squared version, which are referred as AE and SE respectively. Errors are evaluated on semi-dense GT data. We calculate relative error maps by the difference of error maps of Absolute Error, $A(i)$, and Squared Error, $S(i)$, of two methods respectively to show the gains of our method over MultiStack [6]. The error differences are calculated by the following equation:

$$A(i) = |\hat{d}_M(i) - d_t(i)| - |\hat{d}_T(i) - d_t(i)|, \tag{4.9}$$

$$S(i) = |\hat{d}_M(i) - d_t(i)|^2 - |\hat{d}_T(i) - d_t(i)|^2, \tag{4.10}$$

where $\hat{d}_M$ and $\hat{d}_T$ are depth estimates of MultiStack [6] and TWISE respectively. $A(i)$ and $S(i)$ are Absolute Error Difference and Squared Error Difference of pixel i on two competing methods

Figure 4.7: Difference of TWISE vs MultiStack [6] in (a) Absolute Error (AE) and (b) Squared Error (SE) respectively. The red indicates the most gain of ours over [6], marked by 'o'; while the blue is vice-versa, marked by 'x'. Zoom in for details.

respectively. For a particular pixel, when $A(i)$ and $S(i)$ is $(+)$ve, TWISE is performing better then MultiStack and vice-versa for $(-)$ve values. We note that the errors are evaluated only where there are valid ground-truth pixels.

As shown in Fig. 4.7, our method wins in substantially more pixels than losing. Errors in our method often comes from few pixels at boundary regions, when a FG depth is erroneously chosen over a BG depth/vice versa; we term them as outliers *e.g.*, see depth error at the traffic sign pixels, edge of tree-trunk etc close to/at the boundary. These outliers with large depth errors are strongly weighted by the RMSE metric, leading to our worse performance on that metric.

To further our analysis, we do a statistical evaluation 4.8 on 200 samples of the validation set (chosen every 5 samples from KITTI's $1,000$ validation set).

For the statistical analysis, we do a histogram binning of $A(i)$ for pixels where $A(i) > 0$ (Multistack $>$ TWISE is equivalent to performance gain of TWISE over MultiStack) and of $|A(i)|$ for pixels where $A(i) < 0$ (TWISE $>$ MultiStack is equivalent to performance gain of MultiStack

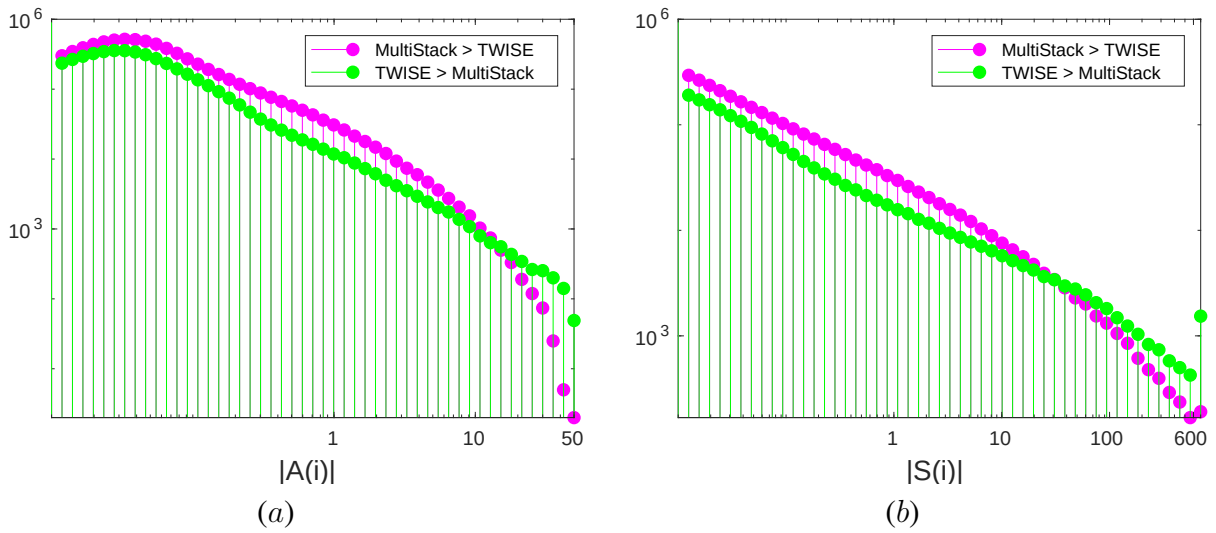


Figure 4.8: (a) Magenta is a histogram of absolute error differences $A(i)$ for $A(i) > 0$ (where MultiStack errors $>$ TWISE errors) and green is a histogram of $|A(i)|$ for $A(i) < 0$ (where TWISE errors $>$ MultiStack errors). (b) Corresponding histograms for squared pixel error differences $S(i)$.

over TWISE). There histograms are plotted together in Fig. 4.8(a). Analogous histograms are plotted for the squared error difference, $S(i)$, in Fig. 4.8(b). These histograms show that TWISE has less error than Multi-Stack [6] for most pixels ($\sim 2.70 * 10^6$) compared to just ($\sim 6,100$) pixels where Multi-stack bests TWISE. The average image in this set has $13,500$ pixels where TWISE is better versus 31 pixels where MultiStack is better.

The reason for large RMSE errors in TWISE is believed to be caused by the outliers (erroneous FG/BG depth selection by TWISE) closer to object boundaries. The outliers are penalized heavily by RMSE metric as opposed to floating depth pixels estimated by MultiStack; as a result, our depth estimate suffers in that metric. As representative examples in Fig. 4.9, the error maps show depth errors around the boundary, and missing thin objects like poles. The reasoning can be further enhanced by the Tab. 4.2. In this analysis, we leverage GT semantics provided by KITTI semantic segmentation dataset. In 140 images, FG objects are poles, boundaries, traffic signs, vehicle, person and the rest as background. For each image, we label all pixels where distances to object boundaries less than 3 pixels are referred as edge pixels and the remaining as inside object pixels.

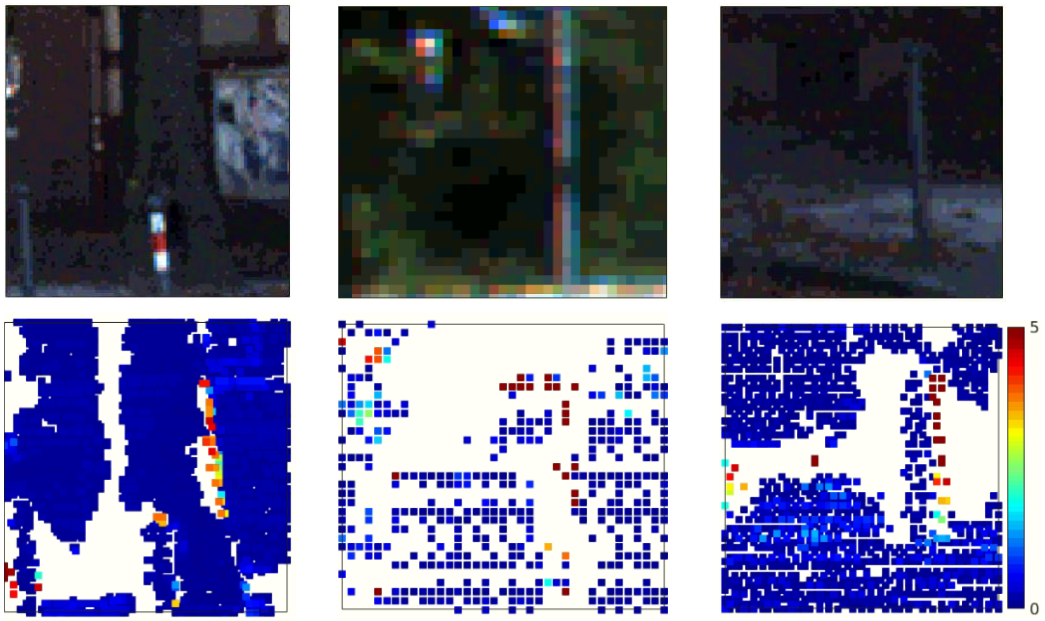


Figure 4.9: Color images (top) and depth error maps in $0 - 5$m (bottom).

Area	MAE	RMSE	TMAE	TRMSE
Inside Object	196.1	752.3	138.6	327.3
Edge Pixels	731.6	2396.9	304.4	454.6
Whole Image	215.1	880.9	144.6	254.3

Table 4.2: Error metrics for different image regions on TWISE.

Tab. 4.2 validates substantial larger errors are around boundary.

4.5.3.5 Outlier Errors and Analysis on KITTI Semi-Dense GT

While outliers can be caused by wrong estimation of foreground/background depth, another important source of outliers is incorrect labelling of ground-truth depths in KITTI. As a result, loss functions that are more sensitive to outliers (i.e. MSE loss) can be negatively influenced by the presence of noise. We highlight the noisy ground-truth labels in KITTI in the next section. In this section we show some evidence of outliers (noisy ground-truth depth) on boundaries of objects in KITTI's semi-dense GT.

Uhrig [49] proposed an approach to generate large-scale semi-dense GT data ($85k$ training

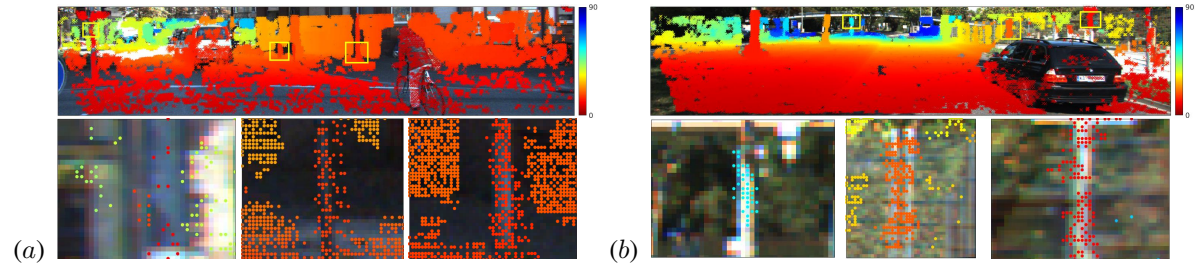


Figure 4.10: Semi-dense GT depths overlaid on color images. Zoom-in views show foreground/background depths are incorrectly spread (dilated/constricted) across boundaries of poles, traffic signs etc. visible in color images.

images) on realistic outdoor scenes suitable for neural network training. Although the approach is scalable on any dataset, it creates noisy ground-truth depth. Uhrig's analysis shows that the semi-dense GT has larger errors on dynamic objects and large-range pixels. Additionally, we show that it also contains incorrect depth labels on some boundaries of objects. In both (a) and (b) of Fig. 4.10, we show zoomed in views of how foreground and background depths that are incorrectly spread across the boundaries of the poles, traffic signs, trees etc. of color images.

Our analysis shows that the outliers in the semi-dense GT are caused by a variety of reasons;

- Noisy rotation R, and translation t obtained from the IMU sensor
- Timing synchronization between camera trigger and time taken to spin one lidar revolution
- Consistency Check on Stereo-Global Matching algorithm which introduce boundary artifacts
- Accumulation of lidar points from dynamic objects.

MAE (in pixel)	RMSE (in pixel)	KITTI Outliers*	MAE (in cm)	RMSE (in cm)
0.35	0.84	0.31	38.6	94.1

Table 4.3: Relation between Disparity Error and Depth Error in metric units (cm). Note that KITTI Outliers are defined by: > 3 pix disparity error and 5% error.

Method	RMSE (m)	REL	$\delta_{1.25}$	$\delta^2_{1.25}$	$\delta^3_{1.25}$
DC-3co [12]	0.118	0.013	99.4	**99.9**	**100.0**
DeepLidar [85]	0.115	0.022	99.3	**99.9**	**100.0**
DepthNormal [87]	0.112	0.018	99.5	**99.9**	**100.0**
GNN [92]	0.106	0.016	**99.6**	**99.9**	**100.0**
TWISE	0.097	0.013	**99.6**	**99.9**	**100.0**
NLSPN [13]	**0.092**	**0.012**	**99.6**	**99.9**	**100.0**

Table 4.4: Depth completion results on NYU2 [4].

In order to evaluate the depth quality of semi-dense GT, Uhrig [49] used the manually cleaned training set of 2015 KITTI stereo benchmark as reference data. The depth evaluation is done in pixel units. We realize that it is equally important to evaluate the semi-dense ground-truth depths in metric units to notice the *effect of boundary outliers* on semi-dense ground-truth depth metric performance. We translate the error in pixel units to error in metric units in Tab. 4.3, by converting the ground-truth disparity to depth using KITTI's provided intrinsics. It shows the noisy semi-dense ground-truth depths suffering from boundary noise and dynamic objects can also have significant errors in metric units. It is also a possible indication that lowering the RMSE error in semi-dense GT might result in learning the noise inherent in semi-dense ground-truth.

4.5.3.6 Quantitative Results on NYU2

: Results on NYU2 are shown in Tab. 4.4, based on its standard metrics. We are currently ranked the *second* in all standard metrics. Note that compared to NLSPN [13], ours is $10\times$ faster in inference on KITTI. The results also show that TWISE is equally generalizable to indoor scenes.

4.5.4 Ablation Studies

In this section, we conduct extensive ablation studies to investigate the effect of different parameters of our proposed loss. We train with $1/6$ data ($\sim$12K training samples) due to resource constraints, and maintain this protocol for all ablations unless otherwise noted.

	Res-18 [10]				MultiStack [6]			
Loss	MAE	RMSE	TMAE	TRMSE	MAE	RMSE	TMAE	TRMSE
L_1 [47]	282.6	110.6	181.8	295.6	211.0	950.0	138.6	246.0
L_2 [10]	341.2	987.8	244.6	349.5	247.4	880.0	170.3	285.0
L_2+L_1 [13]	298.8	**972.2**	206.5	316.7	231.8	**887.5**	156.9	271.2
Huber [90]	288.6	1039.6	198.0	302.1	222.6	927.1	153.9	256.0
CE [12]	279.1	1125.1	184.3	**239.1**	–	–	–	–
TWISE	**275.5**	1045.1	**181.1**	294.0	**201.3**	927.6	**134.1**	**240.1**

Table 4.5: Effect of different loss functions. Compared to single channel losses, CE requires 80 channel, while TWISE requires 3 channel.

Options	MAE	RMSE	TMAE	TRMSE
$\hat{d}_t = \hat{d}_1$ ($\sigma = 1$)	306.9	1109.9	204.4	314.8
$\hat{d}_t = \hat{d}_2$ ($\sigma = 0$)	295.4	1092.9	193.9	306.1
$\hat{d}_t = 0.5 * (\hat{d}_1 + \hat{d}_2)$ ($\sigma = 0.5$)	220.7	**854.8**	148.2	262.4
$\hat{d}_t = \hat{d}_1 / \hat{d}_2 \vert \sigma > 0.5$	261.0	1008.0	180.4	287.9
No color	222.4	1067.5	139.2	247.8
$\hat{d}_t = \sigma\hat{d}_1 + (1-\sigma)\hat{d}_2$	**193.4**	879.4	**131.1**	**236.0**

Table 4.6: Effect of learned σ in TWISE, evaluated by our best model.

γ	MAE	RMSE	TMAE	TRMSE
1.0	223.1	950.1	145.8	257.0
1.5	207.8	947.9	138.1	245.1
2.0	**201.3**	927.6	**134.1**	**240.1**
2.5	204.4	932.5	136.1	242.5
5.0	207.1	923.4	138.7	246.1
10	216.1	**922.8**	146.7	255.4

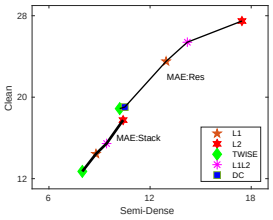
Table 4.7: Effect of γ on depth completion performance.

4.5.4.1 Effect of Loss Functions

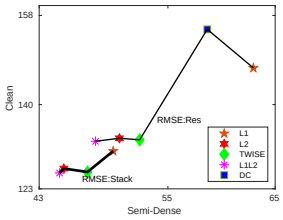
We show that performance of our loss function is network agnostic. Tab. 4.5 refers to different loss functions typically used in SoTA depth estimation works. Although L_2 is a widely used loss for estimating depth [10, 6, 48], L_1 loss [47], Huber loss [90], $L_1 + L_2$ [13] are some of the widely used losses for depth completion. We compare our TWISE loss with all others, including the CE loss [12]. 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.

Supervision		Noisy Semi-Dense GT				Clean GT			
Backbone	Method	MAE	RMSE	TMAE	TRMSE	MAE	RMSE	TMAE	TRMSE
MultiStack [6]	L_1	8.79	49.9	7.09	16.02	14.43	130.62	6.16	18.28
	L_2	10.40	45.35	8.61	17.89	17.75	127.14	8.45	20.18
	$L_1 + L_2$	9.42	**44.90**	8.23	16.82	15.45	**126.20**	7.14	19.30
	TWISE	**7.98**	47.5	**6.25**	**15.35**	**12.71**	126.4	**5.22**	**16.67**
ResNet-18 [10]	CE	10.50	58.67	8.64	**16.57**	19.03	155.24	**8.26**	18.29
	L_1	12.95	62.97	14.68	19.25	23.52	147.42	12.51	29.33
	L_2	17.45	50.48	17.48	21.26	27.48	133.21	15.57	36.73
	$L_1 + L_2$	14.21	**48.25**	15.80	20.10	25.40	**132.6**	14.35	32.47
	TWISE	**10.24**	52.37	**8.42**	16.77	**18.88**	132.94	9.45	**18.17**

(a)



(b)



(c)

Figure 4.11: (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 [6] and ResNet-18 [10]). Methods trained with the same backbone network are connected.

4.5.4.2 Effect of σ on Estimated Surfaces

Another interesting evaluation is the importance of learned σ on different estimated surfaces. In Tab. 4.6, we evaluate estimated depths for different combinations of σ and compare individually its depth completion metrics. The performance is evaluated on our best model in Tab. 4.1, except for the row with "no color", where we train without color input on the same network of our best model. From Tab. 4.6, foreground and background depth surface estimates, as usual, have higher error metric, since they are individually a biased estimate of depth. If we fix σ 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 σ does not learn effectively without color input. So high-resolution imagery helps to learn effective σ and resolve ambiguities at the boundaries.

4.5.4.3 Effect of γ on Performance

Since γ impacts the separation of foreground and background surfaces, we perform an ablation to assess its impact on TWISE. Tab. 4.7 shows depth completion performance with several γ values. With $\gamma = 1$, the loss is equivalent to MAE. As γ increases, the gap between foreground and

background surface increases. At small γ 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 γ, 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.

4.5.4.4 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. We simulate lower resolution LiDARs by subsampling 32R, 16R, 8R rows from 64R lidar (depth acquisition sensor used by KITTI). The different sparse patterns can be seen in Fig. 4.12. We subsample the points based on selecting a subset of evenly spaced rows of 64R raw data provided by KITTI (split based on the azimuth angle in the lidar space) and then projecting the points into the image. All the SoTA methods compared have been retrained using the author provided code with variable sparse input patterns. Tab. 4.8 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.

4.5.4.5 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. 4.11 (a). Several

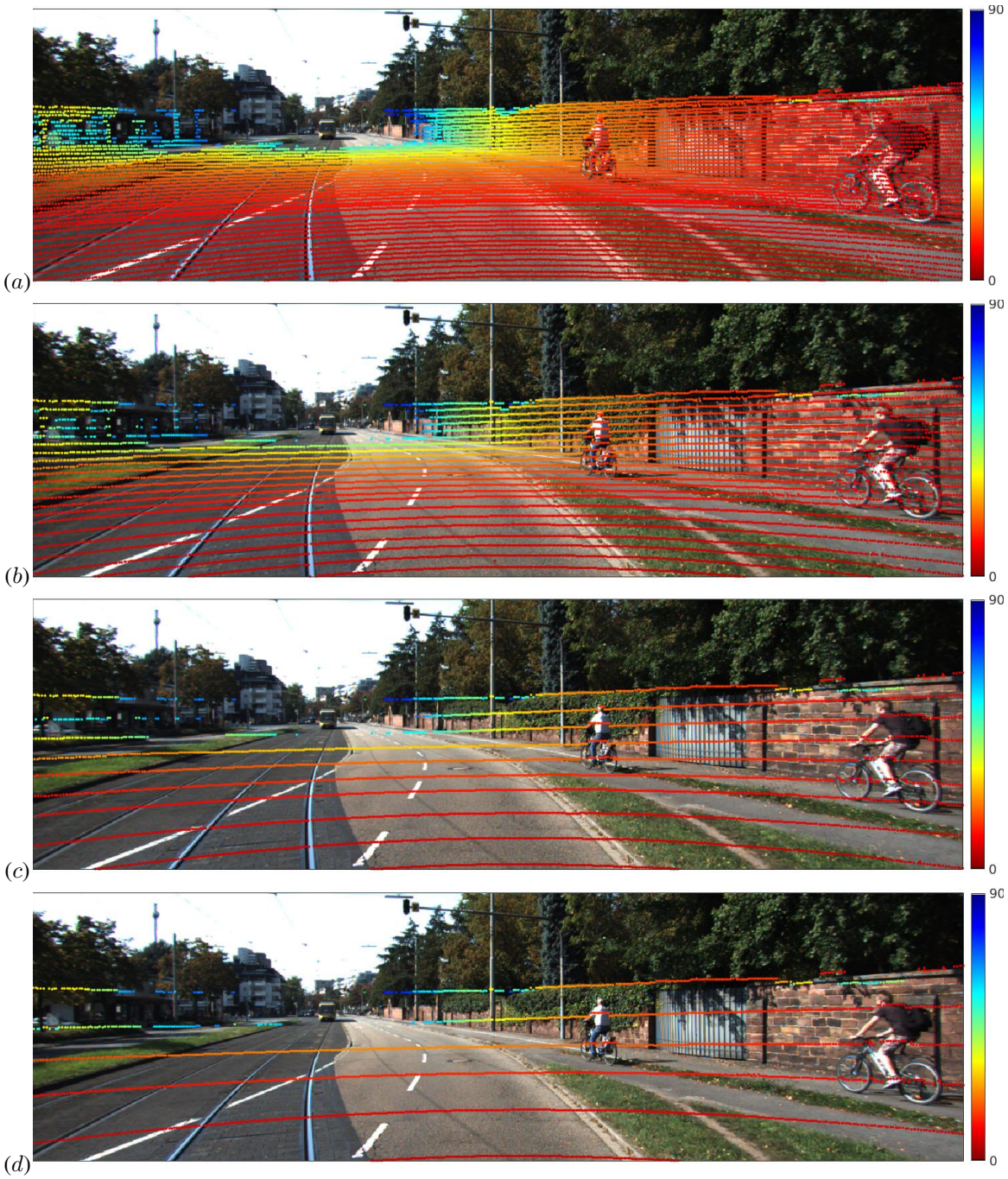


Figure 4.12: KITTI sparse patterns of (a) $64R$, (b) $32R$, (c) $16R$, and (d) $8R$ subsampled LiDAR respectively overlaid on a color image.

inferences can be drawn from the scatter plot of Fig. 4.11 (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. Additionally, the rank-

Sparsity	Method	MAE	RMSE	TMAE	TRMSE
64R	DC [12]	279.1	1125.1	183.1	292.3
	MultiStack [6]	229.4	889.7	156.8	265.0
	NLSPN [13]	219.1	**868.0**	147.7	263.4
	TWISE	**201.3**	927.6	**134.1**	**240.1**
32R	DC	392.7	1456.2	232.1	350.7
	MultiStack	439.2	1288.8	275.4	402.3
	NLSPN	392.4	**1229.2**	248.2	373.8
	TWISE	**327.9**	1242.6	**204.9**	**324.3**
16R	DC	477.7	1777.3	259.5	382.9
	MultiStack	528.4	1504.3	308.6	439.5
	NLSPN	497.1	1483.1	286.8	419.2
	TWISE	**414.0**	**1481.1**	**237.3**	**365.1**
8R	DC	634.7	2311.9	288.5	420.6
	MultiStack	672.58	1841.6	353.2	486.8
	NLSPN	669.05	1869.5	340.3	475.2
	TWISE	**532.1**	**1782.5**	**275.6**	**409.4**
RGB	DC	2423.8	4433.6	715.4	797.2
	MultiStack	2070.4	4185.1	635.7	735.4
	NLSPN	2192.9	4362.35	646.0	743.6
	TWISE	**1964.1**	**4078.8**	**612.0**	**716.5**

Table 4.8: Row sparsity impact on SoTA depth completion methods.

ing 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 more 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.

4.6 Conclusion

In this chapter 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.

Chapter 5

3D Object Detection from Noisy Depth

5.1 Introduction

In autonomous driving, active depth acquisition sensors like LiDARs and radars are paramount in scene understanding and perception problems like 3D object detection and localization [101, 9, 102, 103], 3D semantic segmentation [104, 105, 106] and navigation problems [107, 108], but they become expensive with high-resolution, long-range depth and accuracy of the sensors. Depth sensors, typically have cm level accuracy and record raw depth measurements in $3D$ point-clouds, which are irregular sampled points of object surfaces. Recovering 3D structure and shapes from pointclouds is important for perception and 3D surface reconstruction, but challenging due to sparseness of data, missing depth points on objects due to occlusions and surface properties. As a result, high density pointclouds are often desired. Since high-density and long-range depth sensors are expensive [1], there has been active research problems in *estimating* high-density depth using cheap sensors like stereo [109, 21, 23] (stereo depth estimation), monocular camera [110, 29] (monocular depth estimation), or jointly using low-resolution LiDARs or radars with high-resolution color imagery (depth completion) [37, 7, 13]. Unfortunately, former methods for estimated depth often results in inaccurate 3D pointcloud. The question we are trying to answer in this chapter is whether additional but depth points with lower accuracy from depth completion can help in perception problems like 3D object detection and pose estimation. Extensive experiments

[1] https://arstechnica.com/cars/2018/05/why-bulky-spinning-lidar-sensors-might-be-around-for-another-decade/

with synthetic (Virtual KITTI) and real datasets (KITTI dataset) have been performed using SoTA architectures to validate the findings.

There are different ways of densifying sparse depth points or pointclouds; upsampling onto a 2D regular grid or 3D irregular (pointcloud) structures. Each of these methods is fundamentally different from the other. The idea of upsampling an irregular pointcloud in 3D [111, 112, 113] is interesting since the sparse depth points are already in real metric space, scale and empty space between objects are well captured in this space, and no artifacts are created from perspective distortion and occlusions. Also the densification of points can be made uniform althroughout the 3D space. However, these upsampling methods are still limited to a single surface or multiple surfaces from the same object and require 3D models of objects for supervision. In real-world scenes, multiple discontinuous surfaces and disconnected regions can exist. In this chapter, we focus on depth completion which performs depth upsampling in a 2D regular grid from LiDAR pointclouds projected into the image plane. In this way, high-resolution color imagery can be leveraged more readily for filling incomplete depth pixels in a 2D grid.

A big difference with doing upsampling in 2D space as opposed to 3D space is that depth-completed map is non-uniformly dense. Close-range points are dense while long-range points are sparse in 3D. Also, only visible surfaces can be readily upsampled in the image grid. As a result, artifacts can be created by interpolation, erosion or dilation of available depth pixels in the grid. Common artifacts are floating depth pixels between objects, holes in occluded objects, suppression of thin and small structures by foreground objects. However, one key reason of doing upsampling in 2D space is that a depth image samples the environment in a similar fashion to a data source (LiDAR and video) with density of points falling off with range; while 3D grids do not naturally sample the environment in this way. Also, traditional 2D convolutional neural networks (CNNs) can be readily applied on image grids, and it requires far-less computational and memory

footprints compared to point-based, voxel-based or mesh-based architectures. Thus 2D grids are still a preferred choice for upsampling depth-maps.

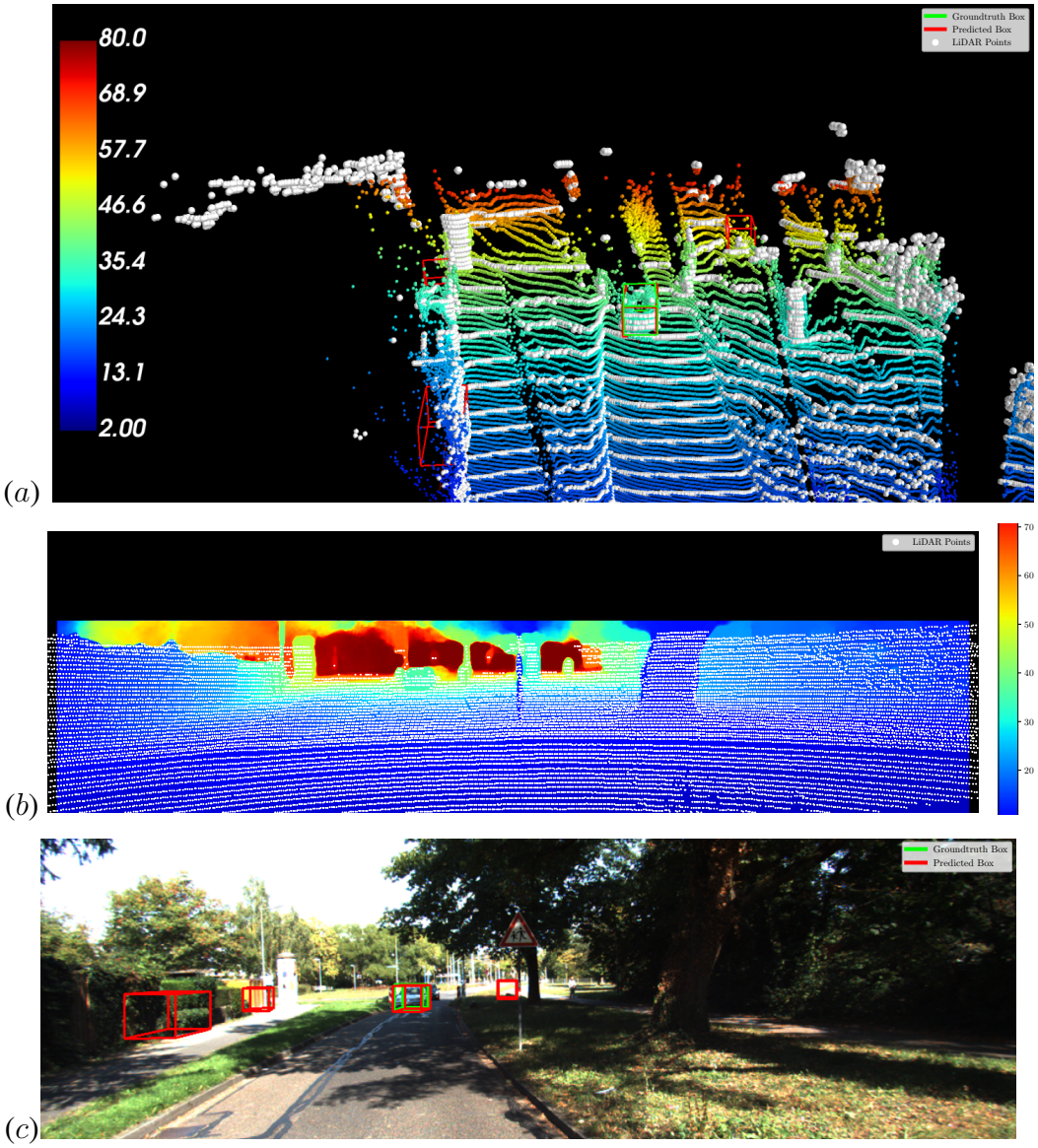


Figure 5.1: Object detection from a SOTA architecture [5] trained with dense inaccurate pointclouds back-projected from a depth image by SOTA depth completion method i.e. TWISE [7]. The figure showing (a) several false positive detections (red 3D cuboids) on the estimated high-density pointcloud. Estimated (colored) and LiDAR (white) pointclouds also shown in $3D$ space (b) Estimated depth map in 2D space from where the estimated pointcloud in (a) is originated from. The LiDAR points are also shown in white dots, and (c) shows the estimated and groundtruth $3D$ bounding boxes projected to a 2D image space of the scene. Red and blue boxes are predicted and groundtruth $3D$ bounding boxes respectively. It shows that bush and phone booth are being wrongly classified as cars.

Several estimation errors can arise by upsampling depth maps. The errors can change with long-range depths, boundary ambiguity, highly sparse or non-existing LiDAR points on thin and small structures and far-away objects. We note that raw LiDAR points also have errors, but they are typically in cm-level accuracy. Compared to LiDAR points, the scale of estimation errors can still range from few cms to few meters depending on the distance of the objects from the camera, ambiguous surfaces or boundaries etc. We model some of these errors as noise with certain characteristics (see sec. 5.3.2) and would like to use the two terms interchangeably.

We show in this chapter that SoTA object detection performances often suffer from depth error from estimated dense depths (see Fig. 5.1). We discover different types of these estimation errors that typically exist in depth completion results. Similar detection performance drops are also evident when we simulate the error as noise with certain characteristics in Virtual KITTI (VKITTI) [100], a synthetic dataset with a similar setup to the KITTI [1] dataset. Interestingly, our experiment reveals that high-resolution depths do contribute to better object detection performance if depth-maps are noiseless and free of artifacts, as is the case for VKITTI. That leads us to design simple, yet elegant noise filtering techniques to tackle depth error from estimated depth maps. We conclude that reducing depth error and sampling depth points from relevant areas are keys to improvement in detection performances with high-density but noisy depth points.

The main contributions of the chapter are summarized as follows:

- We investigate the effect of high-resolution but noisy depth on $3D$ object detection with SoTA point-based and voxel-based neural network architectures.
- We study the effect of noise-free and noisy depth on a synthetic dataset on $3D$ object detection.
- We propose an effective way to leverage estimated depth from depth completion for better

object detection performance.

5.2 Related Works

5.2.1 Depth Completion and Depth Prediction

While depth completion involves the use of LiDAR and high-resolution color imagery to complete the remaining missing depth pixels, depth prediction involves estimating depth from color image only. Quite naturally, depth completion has a lower depth error metric compared to monocular/stereo depth estimation since LiDAR depth measurements are additionally used as input signals. Deep Neural Networks (DNNs) are applied to both sets of problems. Both these problems are relying more on geometric and semantic constraints like depth-normal consistency [86, 87, 110] to improve depth completion and monocular depth prediction performances. Some methods devise novel depth representations [12, 7, 27, 114] for improving depth metric performances.

5.2.2 $3D$ Object Detection with Multi-Modal Sensors

$3D$ object detection is a widely researched and challenging perception problem in vision and a wide variety of sensors are used for this purpose. Some of the most common sensors in a typical sensor suite of autonomous vehicles are LiDARs, cameras, radars, GPS and even ultrasonics. Based on the different types of sensors, detection performance varies widely. We will focus our discussion to LiDARs and camera, as these are broadly used to develop perception algorithms.

The widest and most commonly used sensors are LiDARs, which generate unordered and irregular pointclouds in $3D$ with cm level accuracy. The depth-based $3D$ object detector networks encode these point clouds with point-based [101, 115, 116, 117] and voxel-based neural networks

[9, 118, 119] to represent the geometry of a scene. Both 2-stage RPN networks and single-stage networks are ubiquitously used. 2-stage networks have a region proposal network followed by a refinement network to finetune the regression parameters of the bounding boxes. Typically, 2 stage networks have better detection performance compared to single-stage detectors, although SSDs are more suitable for real-time scenarios. Some of the recent researches [120, 121] prefer bird eye view (BEV) representations of these pointclouds as input to traditional image-based CNNs to improve efficiency of these 3D detections.

A comparatively much cheaper sensor in the sensor suite is a camera, where researches explore ways to infer 3D bounding box parameters like physical size and orientation from predicted 2D bounding boxes in image space [122, 123, 124], under the assumption that perspective projection of 3D bounding boxes fit tightly with its respective 2D detection window. Strong prior shapes of 3D objects [125], shape and geometry of the objects [126], spatial context of the scene [127], and even temporal information [128] are all taken into account for improving monocular 3D detections. However, due to a lack of direct depth measurements, the performance gap between LiDAR based detection and camera only detection is still wide. Recent trends estimate depth [129, 130] or leverage network pre-trained on depth [131] to estimate $3D$ bounding box parameters from a monocular image which boosts up accuracy to some extent, but performance still suffers due to erroneous depth estimation.

To improve the robustness and accuracy of 3D detection algorithms, several existing studies propose to fuse multiple sensors (LiDARs, camera, radars etc) to take advantage of their complementary characteristics; raw LiDARs having depth measurements to cm level accuracy, color images having dense appearance features of a specific field of view of the scene, radars having velocity information of objects and more robust to adverse weather conditions etc. Fusion is necessary to transform the sensor modalities to a common coordinate space since different modalities

have different viewpoint locations. Amongst all the sensors, LiDARs and cameras are the most common sensors widely researched for multimodal fusion [132] to improve perception algorithms. Early-level or pixel-level fusion [2, 133], mid-level or intermediate-feature level [14, 5, 134, 135], and late-level or detection-level fusion [136] all exist in literatures with advantages and disadvantages. But in all these methods, LiDARs with high-quality depth measurements play a big role in detection performances [132] due to measuring precise localization of objects in the 3D scene, robustness to different lighting conditions etc.

5.2.3 3D Object Detection from Estimated Depth

Since high-resolution LiDARs are expensive, some recent research endeavors propose to estimate 3D object detections using estimated depth from monocular [137], stereo [138] or even low-resolution LiDARs [139] instead of using high-resolution raw depth measurements. Some of the pioneering works in this field are [138, 139], which create LiDAR-like representation from estimated depths by sampling depth in azimuth-elevation space. Qian et al. [140] use end-to-end learning on both depth and object detection networks. Weng et al. [137] use instance segmentation mask on estimated depth map to reduce the effect of smeared/floating depth pixels for monocular $3D$ object detection. Although all the above methods create a 64R pseudo-LiDAR representation, little study is done on the effect of high-resolution depth points estimated in 2D dense grid on object detection performance. In this chapter, we study the effect of depth points on detection performance using estimated depth from a depth completion perspective. In our approach, the input to our depth network is a color image and raw LiDAR measurements, we use estimated depth as input to $3D$ object detection network. The aim is to see whether high-density but estimated pointcloud, when applied over SoTA object detection neural networks, can help in improving 3D object detection compared to raw LiDAR pointcloud.

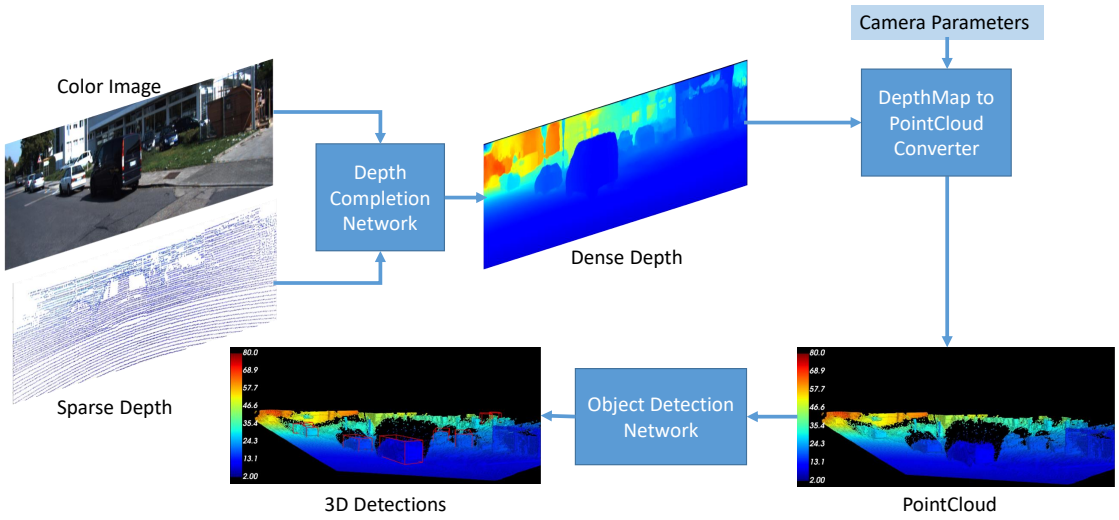


Figure 5.2: 3D detections from high-density pointcloud. The color image and sparse depth is fed into a depth completion network, and dense depth estimate is obtained. The high-density depthmap is then converted to pointcloud in 3D, and trained with a SOTA object detection network, the end-result is 3D detections (red 3D bounding boxes) in pointcloud.

Our preliminary investigation suggests that additional depth points from *raw* depth completion result do not help in improving $3D$ object detection as shown in Fig.5.1 and Tab. 5.1. Our next section investigates the root cause of the problem and proposes some effective strategies to improve detection with high-density depth points.

5.3 Impact of Noisy Depth on Object Detection

In this section, we show several key insights of using raw dense depth completion results for 3D object detection. We show the effect of depth errors on detection problems, architecture bottlenecks present in existing SoTA architectures when handling high-density depth points, and ways to tackle these depth errors to improve depth completion performance.

5.3.1 Baseline Results

We start with a 2-stage SoTA architecture for pointcloud and apply standard loss functions typically present in $3D$ object detection problems. The only change is at the data input of the network, where instead of using 64R LiDAR points, we backproject the dense depth estimate in 2D to pointcloud input as 3D using camera intrinsics (see Fig. 5.2). To facilitate batch training, we keep the initial number of depth points fixed at 50k points. The results are as shown in Tab 5.1.

Network	Input	Car 3D AP			Car BEV AP		
		Easy	Med.	Hard	Easy	Med.	Hard
Object	Raw	**89.0**	**78.69**	**76.68**	**93.08**	**87.12**	**85.0**
Object	Semi-Dense GT	75.82	60.09	59.02	82.31	67.85	67.22
Depth + Object	MultiHourGlass	40.21	25.2	20.65	51.8	31.64	26.7
Depth + Object	TWISE	82.04	59.79	52.80	89.42	68.30	61.0

Table 5.1: 3D Object Detection results ($3D$ and Bird-eye view (BEV) average precision respectively) with different depth resolutions. Object refers to object detection [5] and Depth refers to depth completion [6] network. Raw refers to 64R LiDAR, Semi-Dense GT refers to the GT created by accumulating LiDAR points used for supervision of depth completion network in KITTI. We use the results of two depth completion networks (MultiHourGlass [6] and TWISE [7]) for comparison purposes.

From Tab. 5.1, we compare two depth completion methods, and TWISE still gives the best performance owing to less smearing point [7] (refer to Sec. 4.5.3.4) at the boundaries. However, it is still not enough to help improve object detection performance using raw 64R LiDAR, which has, on average $\approx$ 18k points at the input, compared to $\approx$ 200k high-density depth points. Most interestingly, object detection with semi-dense KITTI groundtruth gives worse performance than TWISE. The possible factors are non-uniform distribution of close and far points, pruning of LiDAR points when there are inconsistencies with stereo depth, and depth noise in the form of outliers (refer to Sec. 4.5.3.5 on creating semi-dense groundtruth) indicating that semi-dense depth can also contribute to deteriorating performance in 3D object detection.

We further evaluate the object detection at two different ranges of depth; $0-30$m and $30-70$m 5.2 and compare the performance on different categories of objects; easy, medium, and hard based on object size and level of occlusions. We see that the performance gap between sparse and dense pointcloud increases at moderate or hard categories of objects and far-away objects. It indicates that at high occlusions and far-away objects, limited visibility in camera plane and limited groundtruth supervision at far-away depth also result in more noise/ambiguous depth estimate leading to erroneous $3D$ object detections.

Range	Depth Density	3D AP			BEV AP		
in meters		Easy	Medium	Hard	Easy	Medium	Hard
$0-30$	Raw	**94.32**	**92.67**	**89.51**	95.85	**96.1**	**95.19**
	Dense	89.52	87.88	77.61	**95.94**	93.87	81.40
$30-70$	Raw	**45.62**	**55.75**	**54.27**	53.65	**70.56**	**68.94**
	Dense	39.38	42.35	36.83	**54.07**	58.37	51.56

Table 5.2: AP Comparison of raw (64R LiDAR) and dense depth (TWISE) at different depth ranges in *meters*. Numerical results also show dense depth performs worse compared to raw LiDAR at all categories; with the gap increasing at higher depth range and more difficult categories respectively.

5.3.2 Noise Modelling

From the previous section, we notice degradation of object detection performances due to erroneous high-density depth estimates. In this section, we model this error with some noise characteristics that is consistent with these depth estimates. Assume there is an underlying GT surface to be estimated at each pixel. The LiDAR samples a subset of these pixels exactly or with small noise. The depth completion method estimates all the pixels with some error. These estimated pixel depths generate a point cloud with a noise term that models the errors in the depth for each pixel. For notational convenience, we the terms D and $\hat{D}$ for original and estimated depth respectively.

We consider two kinds of errors in these estimates; errors coming from mixed depth and depth misalignment error respectively. *Mixed depth*, or floating depth pixels between foreground and

background can be introduced by using a spatial gaussian blur. The gaussian blur function is a 2D spatial filter kernel given by:

$$G(x, y) = \frac{1}{\sqrt{2\pi\rho^2}} \exp - \frac{x^2 + y^2}{2\rho^2} \tag{5.1}$$

where x and y are the spatial positions in the grid and ρ is the standard deviation of this filter. The estimated depth can be obtained by convolution of original depth with this kernel as:

$$\begin{aligned} \hat{D} &= D * G(x, y) \\ &= \sum_{s=-a}^{a} \sum_{t=-b}^{b} G(s, t) D(x + s, y + t) \end{aligned}$$

where s and t defines the kernel size of the filter. *Depth Misalignment Error* replicates larger errors in depth as distance increases. We model an additive noise model for this error. The noise η comes from a zero mean gaussian distribution with standard deviation ν such that $\eta \sim M(0, \nu)$, whose uncertainty (standard deviation) increases linearly with distance. The linear relationship of uncertainty vs distance is given by:

$$\nu = 0.05 + aD \tag{5.2}$$

where a is the slope of the linear function. The additive noise model is then defined by:

$$\hat{D} = D + \eta \tag{5.3}$$

Although other errors are possible, the above-mentioned errors are most prevalent in depth

completion methods. In the context of $3D$ object detection, these errors can be further classified as three different types as seen in Fig.5.3.

- Noisy depth points within *foreground*; Depth points on the *foreground* object can spread out within and around the 3D bounding box, as a result, object shapes and size can appear distorted (see Fig. 5.3(c)), we can label it as *noise-fg*.

- Smeared depth points inbetween FG and BG; Floating depth points between *foreground* and *background*; we can label it as *noise-inbet*.

- Noise from Background clutter; There can be structures in the background that can resemble similar objects; e.g. bushes, telephone boxes, es, etc which might look alike to objects' surface as shown in Fig. 5.3(i). We can label it as *noise-bg*.

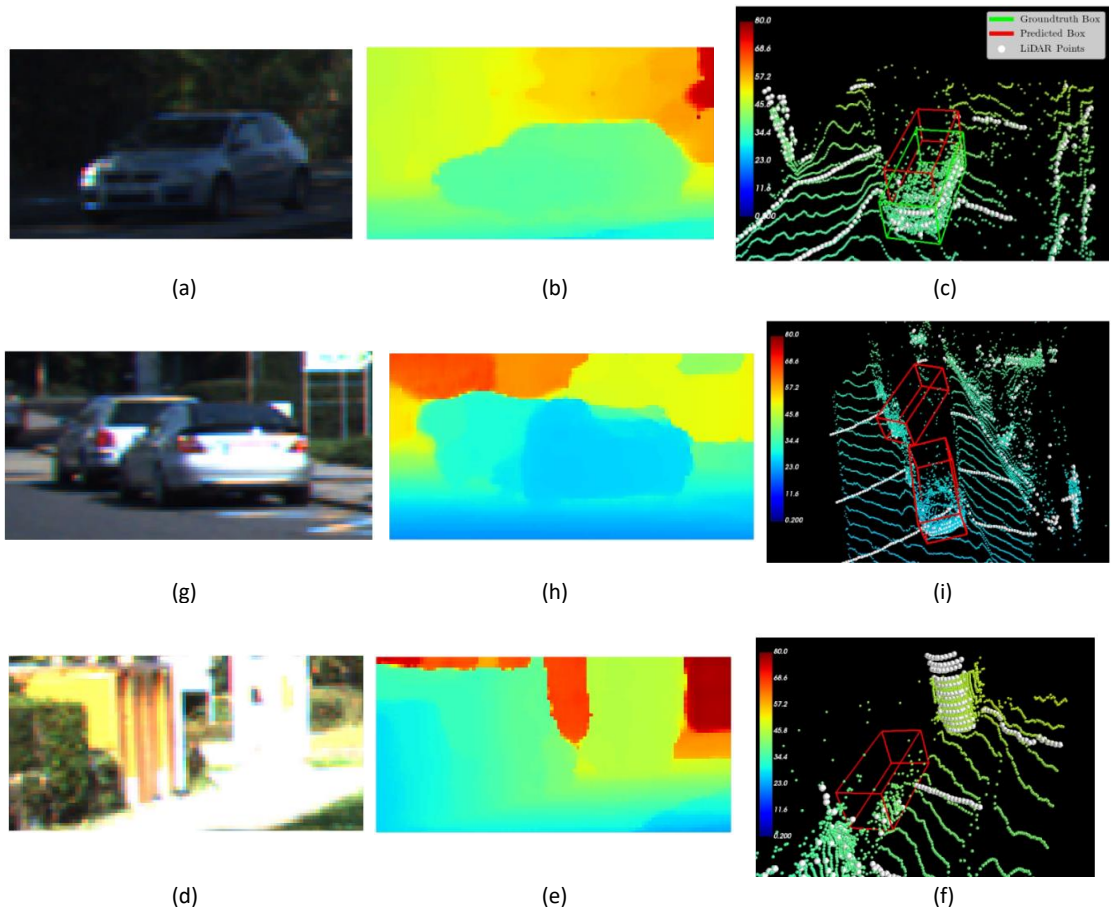


Figure 5.3: Different types of depth noise on estimated depth; the first column shows color images, the second column shows depth images, and the third column shows 3D pointclouds; (a), (b) and (c) show *noise-fg*, indicating smeared points within the car resulting in misalignment of the predicted and GT bounding boxes as shown in red and blue respectively; (d), (e) and (f) show *noise-inbet*, indicating smeared pointcloud between two objects resulting in the wrong orientation of the 2nd car; and (g), (h) and (i) show *noise-bg*, the background points (phone-booth) wrongly classified as a car since its outer surface looks similar to a car shape.

Note that noisy pointclouds also arise due to the limitation of acquisition equipments. However, the properties of noise from estimated depth are very different from the noise that can come from acquisition equipments [141, 142]. Sensor noise typically comes in the form of sensor imperfections, temperature noise, etc and the sensors are typically in cm level accuracy, which is an order of magnitude lower than noise from estimated depth. In addition to noisy dense depth estimate, SoTA backbone networks of $3D$ detection are not conducive to high-density noisy depth points which we describe next.

5.3.3 Architectures

Voxel-net and point-net based architectures are increasingly used as backbones in 3D object detection problems. All these architectures rely on subsampling depth points. Voxel-net based architectures divide the $3D$ space into voxels and subsample depth points based on voxel regions. Voxelized pointclouds are either projected to different views such as bird-eye views (BEV), range view (RV) to be processed by $2D$ convolution, or kept in $3D$ coordinates to be processed by sparse $3D$ convolutions. The point-based method can preserve precision in localization and detailed $3D$ structure information, while voxel-based methods are fast due to aggressive subsampling of points in defined grids.

Deng et al. [9] argue that the $3D$ structure is of significant importance for object detections. Although time-consuming and memory-intensive, point-based methods have the potential to encode detailed $3D$ information quite well. However, these methods also rely on the downsampling of pointclouds at each encoder layer for efficiency and enlarged receptive field. A very important component of the point-based architecture is the furthest point sampling (FPS) which uniformly samples depth points from low-density (far-away pixels) and high-density regions (close-by pixels) in $3D$. It ensures diversity in the selection of points. However, it also promotes noisy pixels and outliers which we describe next.

5.3.3.1 Promotion of Noisy Pixels using FPS

FPS is used to select representative points at each encoding layer of point-net architectures. Downsampling pointcloud is essential for the efficient encoding of points. However, by preferentially selecting points that are spatially far apart from others, it promotes background points and outlier points more compared to relevant object pixels. Fig. 5.4 shows FPS selecting background (trees

and vegetation) and outlier points (at boundaries) from the dense depth estimate of TWISE.

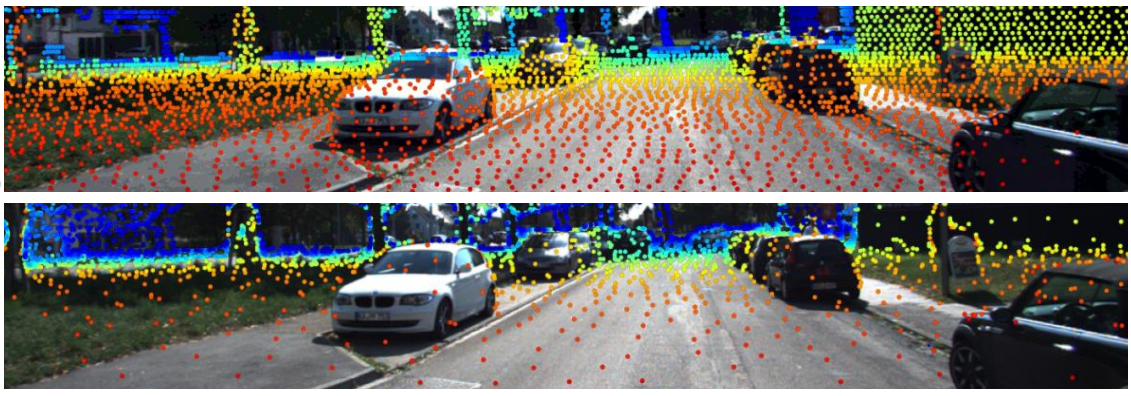


Figure 5.4: Selecting 4096 points from (a) 64R raw LiDAR and (b) dense depth (noisy environments) of TWISE. FPS samples more background and outlier points at boundaries of tree trunks and buildings from noisy dense depth.

5.3.3.2 Downsampling in Point-Based Architecture

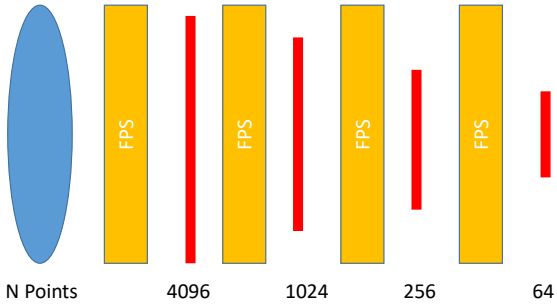


Figure 5.5: Subsampling in point-based architecture. The dense depth points get subsampled to 64 encoded points at the final encoder level.

Downsampling is essential for encoding the points and making memory and computation reasonable. A typical backbone architecture of pointnet downsamples N (input) points into 4096, 1024, 256, 64 points at each encoder stage respectively, see Fig. 5.5. This architecture bottleneck and FPS constrict relevant object points to move deep into the encoder layers.

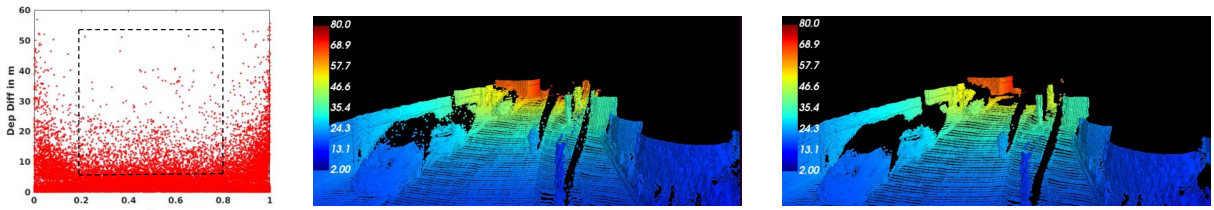
Having studied the architecture limitations of the point-based 3D backbone network, we now explore some remedial steps that can be used to filter out noisy pixels from the estimated depth and

bypass some architecture limitations of 3D object detection networks. The next section describes these phenomena.

5.3.4 Remedies to Tackle Noisy Depth

We investigate some potential ways to filter out noisy depth from dense depth estimates and bypass architecture limitations of the backbone networks. This section explains the analysis.

5.3.4.1 Filtering Smeared Points



(a) (b) (c)

Figure 5.6: Removing smeared points by TWISE. (a) shows the rectangular region $0.2 < \sigma < 0.8$ and depth difference (difference between BG and FG in TWISE) $>= 3m$ as smeared depth points. The σ parameter refers to σ parameter learned in TWISE. (b) and (c) show the unfiltered and filtered depth points in $3D$ space respectively. As shown, most of the floating depth pixels at the scene in (b) is filtered out at (c).

Although TWISE can reduce depth smearing considerably, the σ parameter can still allow depth mixing at regions which are ambiguous. These floating/smeared depth pixels along depth discontinuity and far-away depth can trouble object detection algorithms. Qiu et al. [85] liken smeared points as salt and pepper noise and rely on traditional $2D$ median filtering to remove this noise. Another way to remove it is by checking consistent depth points between estimated depth and depth points created by morphological filtering on sparse depth [143]. However, the ability of morphological filtering decreases as input raw LiDAR points become more and more sparse at far-away regions. There are some natural ways to filter out floating depth pixels between foreground and background based on TWISE estimate. We realize that hard ambiguous regions in TWISE

are often characterized by higher depth discrepancies between foreground and background depth channel. Depth mixing typically occurs at these regions when the σ parameter of TWISE does a soft averaging of the FG and BG estimates instead of selecting either of the depth. This motivates us to design an effective filter for removing smeared points. We use a depth difference threshold between *foreground* and *background* depth and sigma threshold to design the filter (see 5.6 (a)). It shows that by pruning out depth from this rectangular band region, it is possible to get rid of floating depth pixels or *noise-inbet* around boundaries and long-range depth to a great extent. We name it *sigma* filter for the ease of explanation in the next sections.

5.3.4.2 Filtering Background Clutter

We use grid sampling on 3D to downsample dense depth uniformly on close and far away points. Grid sampling prunes points that fall within t m (in our case, 0.2m) of a grid and replaces it with a representative point. Additionally, we use semantic map information of objects (e.g. vehicles, pedestrians, cyclists) generated from the SoTA method [8] to filter out most of the background depth clutter. It ensures that only relevant object pixels are used for subsampling the pointcloud. Some background context of the object can still be gathered by collecting points that fall within the r m (in our case, 0.3m) radius of any relevant car pixels (see Fig 5.7). Using this strategy, it is possible to reduce the number of points significantly by ensuring relevant pixels are always considered as input to the network. It is also possible to bypass the architecture limitations of point-based architecture.

5.3.4.3 Filtering Pixels within r distance from raw LiDAR

Instead of using the estimated depth, we can also leverage the presence of raw LiDAR which is available as input to any depth completion problem. The LiDAR points select from the dense depth

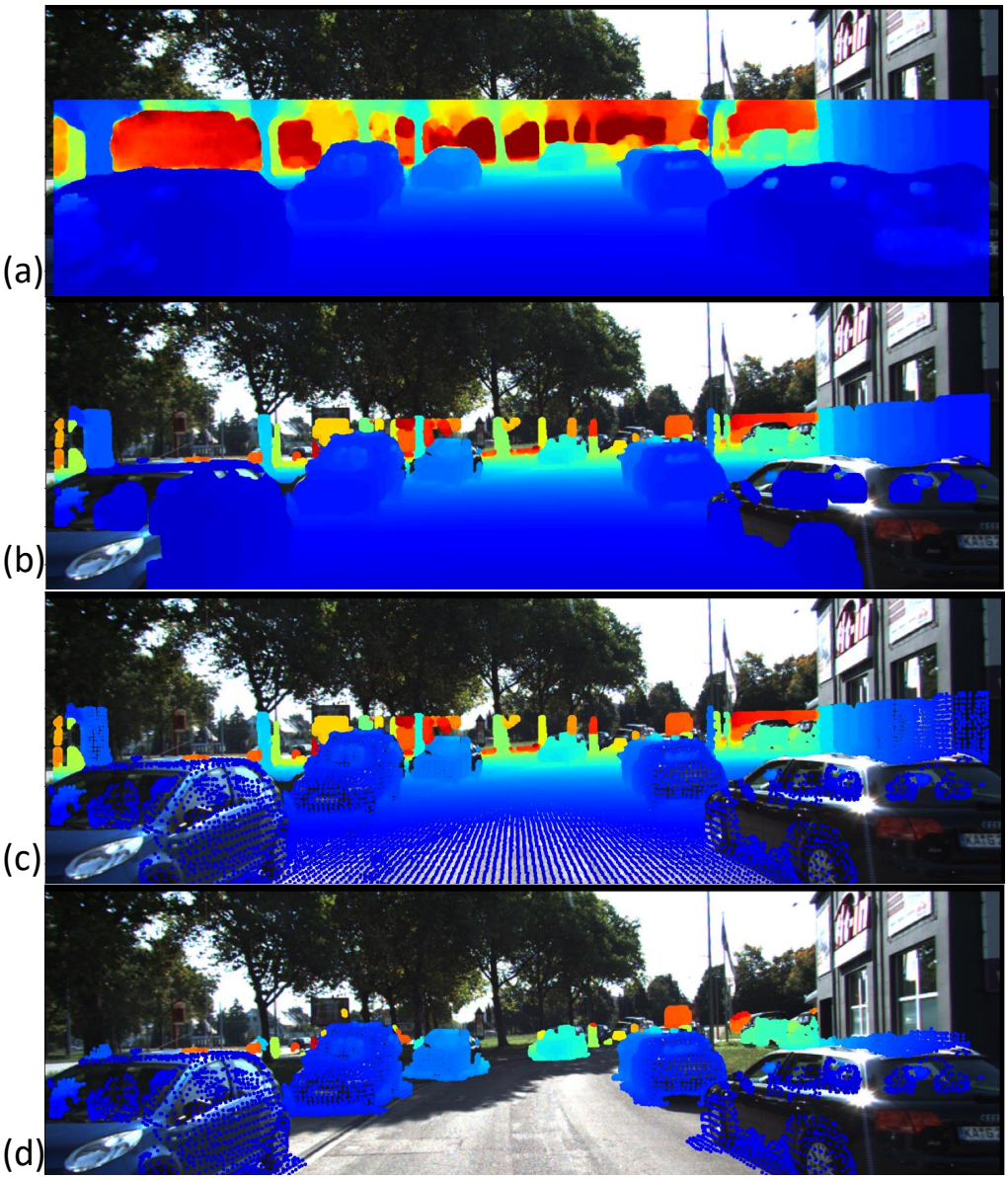


Figure 5.7: Sampling to select relevant pixels from dense depth. $(a) \approx 315k$ with dense depth, $(b) \approx 200k$ after pruning ambiguous pixels based on sigma filter 5.3.4.1, $(c) \approx 35k$ after grid sampling at 0.2m, $(d) \approx 15k$ points after using object level semantic mask and gathering neighboring points 0.3m from the semantic pixels in 3D.

any points that fall within the r m radius from them. In this way, it is possible to avoid smeared points and gather only meaningful background points for context. Due to perspective distortion, closer and far away depth points are unevenly distributed in the image plane. Any depth points that are within d m from the camera are termed as close pixels and pixels beyond d m are far-away pixels. We can use two different r values; $r1$ and $r2$ for depths $<= d$ and $> d$ respectively. This

new depth can be termed as augmented LiDAR since LiDAR is now augmented with dense depth estimate.

5.4 Experiments and Results

We evaluate the proposed algorithm on the standard KITTI Depth Completion dataset [1], a real-world outdoor scene, and Virtual KITTI [100], a synthetic dataset with photo-realistic images and dense ground-truth depth.

5.4.1 Dataset

5.4.1.1 KITTI

KITTI's 3D/BEV object detection framework contains 7481 training samples and 7518 testing samples. The training set is further divided into a training set (3712 samples), and validation samples (3769 samples). The dataset has difficulty levels (easy, medium, and hard) based on the object size, occlusion, and truncation level respectively. There are three major object categories; car, pedestrian, and cyclists respectively. We use the 'Cars' category for training and evaluation.

5.4.1.2 Virtual KITTI

we evaluate VKITTI 2.0, a synthetic dataset with noise-free and dense GT depth at depth discontinuities, accurate 3D object bounding boxes, and noise-free semantic labels. The VKITTI 2.0, created by the Unity game engine, contains 5 different camera locations (15^o left, 15^o right, 30^o left, 30^o 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 driv-

ing sequences 01, 02, 06, and 018 respectively. For testing, we use sequence 020 of both the left and right stereo cameras, and choose all other frames, with a total of 1.6k images. We subsample the dense GT depth in azimuth-elevation space to simulate a LiDAR-like pattern as sparse inputs. We found out that, unlike KITTI, VKITTI does not label easy, medium, and hard categories on its own. We have to label the raw VKITTI 3D object detection boxes into easy, medium, and hard cases depending on the truncation and occlusion levels provided. Object categories are classified as easy if the height of $2D$ bounding boxes are greater than 40 pixels, truncation, and occlusion level is less than 0.1. Similarly, medium categories are referred as objects with height ≥ 20 and ($0.1 \leq$ truncation ≤ 0.5 or and $0.1 \leq$ occlusion ≤ 0.6) respectively. Similarly, for hard categories, height ≥ 15, and ($0.5 \leq$ truncation ≤ 0.9, or occlusion ≥ 0.6) respectively.

5.4.2 Metrics

We use average precision as the common metric to evaluate 3D object detections. Recently, the KITTI dataset applies a new evaluation protocol that uses 40 recall positions instead of 11 recall positions to make a fairer evaluation. All the methods are compared with the new evaluation metric.

5.4.3 Architecture

We use various backbone architectures (i.e. point-based, voxel-based, or mix of points and voxels) for our study. We choose a variant of PointRCNN (see [5]) architecture as our backbone architecture in all ablation studies unless noted.

5.4.4 Implementation Details

Our implementation of PointRCNN2 is similar to the implementation of EPNet [5], except no color fusion network is used. This is done to eliminate the contribution of the color fusion network and investigate the effect of noisy pointcloud on object detection. The range of the pointcloud is restricted to $[-40, 40]$, $[-1, 3]$, and $[0, 70.4]$m along X (right), Y (down) and Z (forward) axis in camera coordinate respectively. And the orientation of $theta$ is in the range of $[-\pi, \pi]$. The number of input pointcloud is fixed at 16384 for raw LiDAR to facilitate batch training. For dense depth, we randomly subsample 50k points as input to the network. Afterward, FPS sampling is used to subsample points to 4096, 1024, 256, 64 respectively similar to point CNN. We keep focal loss for *foreground/background* classification and consistency loss between localization and classification confidence as proposed by the paper. During inference, the top 8000 proposal boxes generated by RPN are selected based on classification confidence. NMS threshold of 0.8 is then used to filter out redundant boxes and obtain 64 positive candidate boxes refined by the refinement network.

Throughout the whole training process, no data augmentation is used, and we kept the training epochs to 50 unless otherwise noted. For all the other networks, we use the public implementation of OpenPcDet [144] and stick to their training protocol unless otherwise noted.

5.4.5 Results

Tab.5.3 refers to object detection results in the KITTI dataset. All the architecture uses LIDAR only for $3D$ object detection, and the improvement results with dense depth are shown with PointRCNN2 as backbone architecture. Filtering floating pixels by sigma filter 5.3.4.1improves performance significantly compared with baseline (D). In D and SF in this table, we first grid

sample dense depth similar to 5.7, and then semantic mask generated by [8] is used for filtering background points. Some background context is captured by gathering r m depth pixels around the semantic depth pixels. Finally, we find the dense depth estimate gives the best results when augmented with raw LiDAR (L and D and SF) filtered by semantic mask and sigma filter.

Archi.	Res.	Car 3D AP			Car BEV AP		
		Easy	Medium	Hard	Easy	Medium	Hard
PointRCNN	L	88.4	76.5	74.4	87.3	85.3	83.3
PointPillars	L	86.0	76.6	70.1	89.8	86.7	83.1
Second	L	88.8	78.4	76.9	90.3	87.7	79.7
PVRCNN	L	89.2	79.1	78.4	90.2	87.8	87.3
VoxelRCNN	L	89.6	79.4	78.7	90.4	88.2	87.8
PointRCNN2	L	90.0	78.7	76.7	93.1	87.1	85.0
	D	82.0	59.8	52.8	89.4	68.3	61.0
	D and F	86.6	69.9	60.8	92.8	78.3	69.0
	D and SF	88.9	75.3	70.3	95.3	84.3	79.4
	L and D and SF	**92.5**	**79.5**	73.2	**96.3**	**88.5**	82.3

Table 5.3: $3D$ Object Detection with augmented LiDAR. L refers to 64R LiDAR, D refers to Dense Depth with TWISE, F refers to sigma filter 5.3.4.1, SF refers to filter used with semantic mask, and Aug.F refers to LiDAR augmented with dense depth estimate using the semantic mask and filter.

We evaluate dense depth estimates on different 3D-based architectures (point-based and voxel-based) and realize that in all these architectures, dense depth estimate reduces detection performance, more significantly in moderate and hard cases (see Tab. 5.4). This indicates that at moderate to heavy occlusions depth estimate is noisier. Although point-based methods are more heavily affected by noise and outliers, sigma filtering recovers performance in medium and hard cases and outbeat voxel-based method.

In order to analyze the $3D$ detection performance on KITTI (real-world dataset) using noisy depth estimate. The figure shows a bird-eye view of predictions and ground truth for different configurations of depth points. Bird-eye views are chosen to show a number of false positives which affect detection performance.

We ablate the effect of several remedial measures to mitigate depth noise on $3D$ object de-

Catgry	Archi.	Res.	Car 3D AP			Car BEV AP		
			Easy	Medium	Hard	Easy	Medium	Hard
Point-Based	PointRCNN	D	74.4	51.9	43.5	80.2	61.2	52.3
	PointPillars	D	74.7	57.5	54.4	83.4	71.4	68.1
	PointRCNN2	D	82.0	59.8	52.8	89.4	68.3	61.0
		D and F	**86.6**	**69.9**	**60.8**	**92.8**	**78.3**	69.0
Voxel-Based	Second	D	71.2	55.7	53.2	79.5	67.5	65.8
	Voxel-RCNN	D	72.8	58.4	56.7	84.0	68.9	67.7
		D and F	79.3	61.6	58.7	87.8	72.8	**70.3**
Mix	PVRCNN	D	73.5	57.8	54.1	83.0	68.5	66.3

Table 5.4: Performance evaluation of different architectures on a validation set of KITTI with dense depth from TWISE. D refers to Dense Depth with TWISE, F refers to sigma filter 5.3.4.1. It shows the filter designed with TWISE output can improve detection performance significantly by pruning floating and ambiguous depth pixels.

tection performance. In Tab 5.5, we show how local region proposals in the form of estimated semantic maps can help improve object detection by reducing background clutters and bypassing architecture bottlenecks. We also try sampling on dense depth in azimuth and elevation space similar to [138, 140] to make pseudo-LiDAR representation. However, the key is to select relevant object pixels as input to the network. To remove redundancy and increase relevant object pixels, we apply grid sampling and semantic mask on the dense depth. Additionally, floating depth pixels are reduced by the sigma filter. The results show all these steps are crucial in improving $3D$ object detection by reducing background clutter, floating depth noise, and bypassing architecture limitations. To gather some background context, pixels around $r = 0.3$m are gathered around the relevant object pixels.

In Tab 5.6, we ablate on different ways of gathering background context by gathering neighboring pixels around the relevant object pixels. Due to uneven distribution of depth pixels at close and far-away regions, we use two different r values for gathering neighboring pixels. The conclusion is that raw LiDAR augmented with dense depth estimate improves object detection accuracy considerably, although at hard regions, the raw LiDAR performance still performs the best.

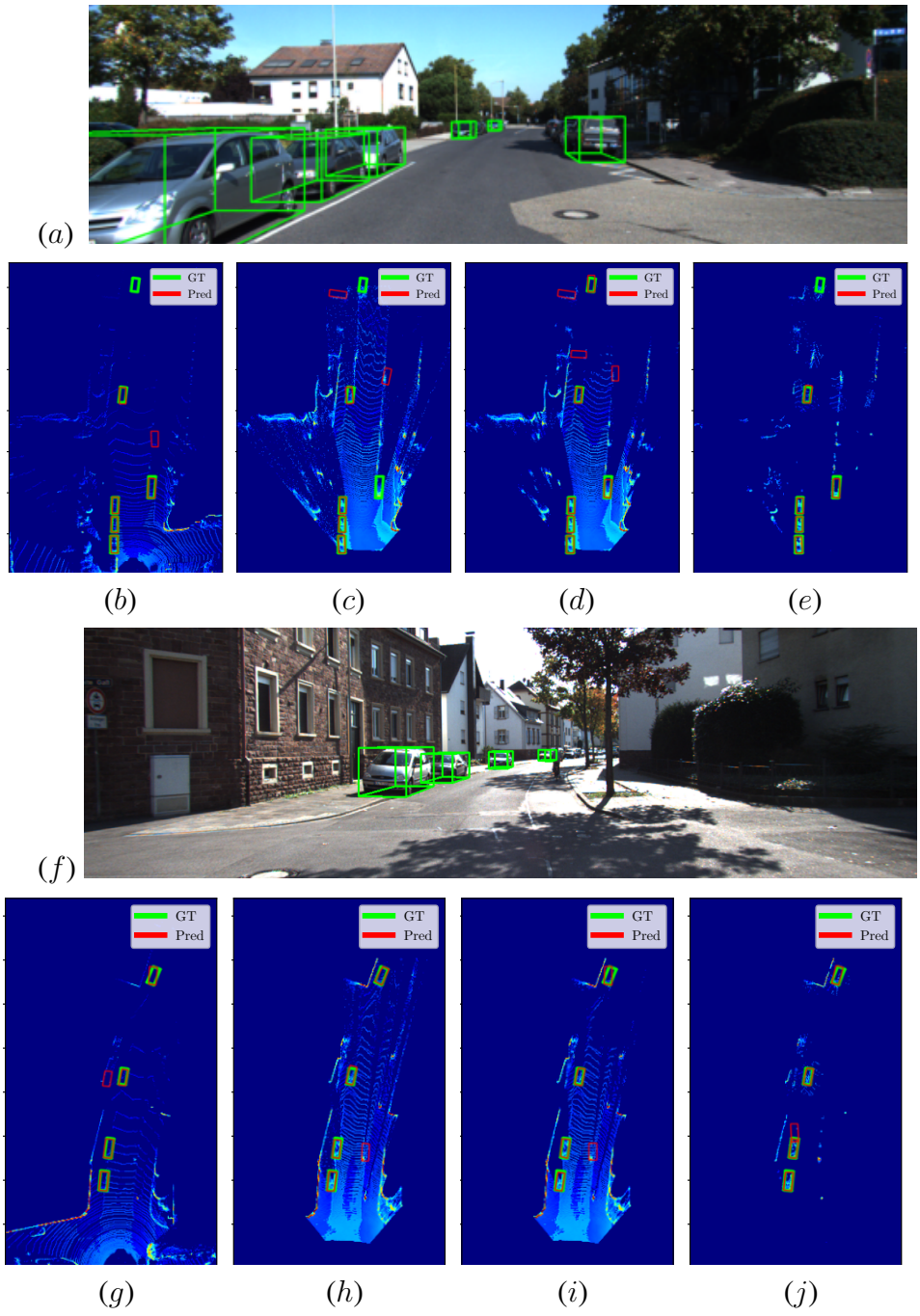


Figure 5.8: Object detection performance comparison in bird eye view with different depth inputs. (a) show color image, and BEV detection on (b) sparse depth, (c) dense depth, (d) depth filterd with sigma filter (5.3.4.1), (e) augmented LiDAR with variable radius respectively. (f), (g), (h), and (i) show another example of a scene with color image and bird eye view of all the methods subsequently. In both the examples, augmented LiDAR has best performance compared to all other methods.

Does high-resolution depth really help in object detection? We set up VKITTI experiments to determine the effect of whether high-resolution 3D pointcloud helps in object detection. In

Archi.	Input	Sampling Region	Car 3D AP			Car BEV AP		
			Easy	Med.	Hard	Easy	Med.	Hard
PointRCNN2	Lidar	N/A	**89.0**	**78.69**	**76.68**	**93.08**	**87.12**	**85.0**
MultiStack + PointRCNN2	Filtered Depth	N/A	86.6	69.9	60.82	92.77	78.34	68.97
MultiStack + PointRCNN2	64R Filtered Depth	azimuth and elevation space	87.85	73.59	69.14	92.27	82.44	79.49
MultiStack + PointRCNN2	128R Filtered Depth	azimuth and elevation space	88.91	74.45	67.41	92.95	83.11	75.90
MultiStack + PointRCNN2	Sem Filtered Depth	Grid Sampling 0.1m	88.91	75.10	70.26	95.30	84.3	79.44

Table 5.5: 3D Object detection results with defined sampling regions to reduce background clutter and bypass architecture limitations. Filtered depth refers to the TWISE filter 5.3.4.1, 64R and 128R are depth sampled in azimuth and elevation space to simulate a LiDAR, while semantic filtered depth refers to the estimated semantic mask created by [8], and used to filter dense depth in image plane based on vehicle pixels. The dense depth pixels are reduced further by grid sampling in 3D at grid spacing 0.1m.

Table 5.7, we classify pixels as All (foreground and background pixels), and Semantic (foreground) pixels which are selected using ground truth semantic masks. We use only *noise-free* depth points in this setting to understand the effect of high-resolution on 3D object detection. We use two types of architectures (point-based and voxel-based) to understand its effect.

For PointRCNN2, We see that when 'All' pixels are considered, the 3D detection performance tops at $256R$ resolution, but falls down with higher resolution ($512R$, Alldep). This indicates that at higher resolution, the FPS component in the architecture selects more non-relevant pixels and hurts the performance. If relevant object pixels ('Semantic') are selected as input to the network, performance improves for all resolutions significantly. The plateauing out at 'AllDep' could be due to memory constraints or the architecture bottleneck of the pointnet structure.

For VoxelRCNN, the findings are interestingly different. We see the performance steadily increases for higher resolution for 'All' pixels, with 'Alldep' giving the best performance. Since no FPS is present in its pipeline, it ensures that relevant object pixels are not thrown away in preference of background pixels. Providing target-relevant pixels improves the performance only

Train Strat.	Archi.	Input Dep	Sampling Region	Car 3D AP			Car BEV AP		
				Easy	Med.	Hard	Easy	Med.	Hard
1-stage	PointNet	Lidar	N/A	89.0	78.69	76.68	93.08	87.12	85.0
1-stage	PointNet	Sem. Lidar	N/A	90.53	79.33	**77.02**	95.1	88.2	**86.22**
2-stage	MultiStack & PointNet	Sem Filtered	Grid Sampling, $r = 0.3$m around sampled pts	88.91	75.10	70.26	95.30	84.30	79.44
2-stage	MultiStack & PointNet	Sem Filtered	Grid Sampling $r = 1$m around sampled pts	89.28	76.88	71.56	93.2	83.82	80.53
2-stage	MultiStack & PointNet	Sem Filtered	Grid Sampling $r = 0.3$m around sampled pts < 40m, $r = 1.5$m around sampled pts $>= 40$m	92.34	78.08	73.14	95.95	84.85	81.93
2-stage	MultiStack & PointNet	Sem Filtered & Sem-Mask	Grid Sampling $r = 0.3$m around sampled pts < 40m, $r = 1.5$m around sampled pts $>= 40$m	**92.51**	**79.50**	73.20	**96.25**	**88.5**	82.25

Table 5.6: 3D Object Detection results with the apriori sampling region and different radius configurations used to gather background pixels around the relevant object pixels. Sem filtered refers to sigma filter and semantic map used to filter out background and floating depth points. Sem-Mask in last row refers to binary semantic information which is concatenated with the input pointcloud as additional information to the detection network.

PixelType	Res. Type / Resol.	3D Bounding Box Easy	3D Bounding Box Med.	3D Bounding Box Hard	Bird's Eye View Box Easy	Bird's Eye View Box Med.	Bird's Eye View Box Hard
All	$64R$	81.4/80.5	80.1/71.0	71.8/69.0	89.7/90.1	81.6/80.9	72.6/71.7
	$128R$	85.5/80.8	79.7/75.0	71.8/70.2	90.6/90.2	83.4/81.0	75.7/75.9
	$256R$	**89.9**/81.3	**81.4**/80.0	**79.7**/70.8	**90.8**/90.4	81.6/81.2	81.3/80.01
	$512R$	**89.9**/81.2	80.7/80.4	72.0/71.4	90.6/90.4	81.4/81.5	78.3/80.2
	Alldep	81.7/89.7	**81.4**/80.7	72.4/**79.7**	90.6/90.4	81.7/**89.8**	72.6/**81.5**
Semantic	$64R$	88.1/–	82.0/–	74.1/–	91.5/–	88.3/–	80.7/–
	$128R$	88.8/–	85.4/–	77.9	91.7	88.8	81.3/–
	$256R$	89.9/–	88.3/–	80.7/–	92.0/–	89.3/–	84.0/–
	$512R$	90.1/–	**88.5**/–	80.9/–	**92.1**/–	89.5/–	84.1/–
	Alldep	**90.2**/89.9	**88.5**/80.5	**81.0**/79.3	**92.1**/90.5	**89.5**/89.3	**84.2**/80.0

Table 5.7: Average precision (%) for 3D detection and pose estimation of cars on VirtualKITTI using PointRCNN2[[5]]/VoxelRCNN[9]. Alldep refers to depth pixels grid sampled at 0.1m, while semantic pixel refers to selected pixels using GT semantic masks. Note that pixels within $r = 0.3$m radius of semantics are also taken into consideration.

slightly, indicating that selecting the relevant pixels apriori has no major impact on the performance in this kind of architecture.

In Tab. 5.8, we show the effect of noise on object detection performance on PointRCNN2 by simulating different kinds of noise; Gaussian Blur refers to the gaussian smoothing along the depth discontinuities and simulating floating points along the boundary. Depth Misalignment refers to the depth noise along the optical ray. We simulate this noise by increasing gaussian noise linearly with distance on the valid pixels. The results substantiate the claim that noisy pointcloud decreases

Noise:	3D Bounding Box			Bird's Eye View Box		
	Easy	Med.	Hard	Easy	Med.	Hard
$64R$ + Noise-free	**81.4**	**80.1**	**71.8**	**89.7**	**81.6**	**72.6**
$64R$ + Gaussian Blur	67.4	54.4	47.1	73.4	62.4	54.8
$64R$ + Depth Misalign.	70.9	56.3	47.4	77.4	63.6	55.9

Table 5.8: 3D detection and pose estimation of cars on VirtualKITTI [1] using different types of depth noises. Gaussian Blur smooths depth along boundaries and simulate smeared depth; Depth Misalign. simulates depth error along the optical ray.

Target Pix.	3D Bounding Box			Bird's Eye View Box		
	Easy	Med.	Hard	Easy	Med.	Hard
All	81.4	80.1	71.8	89.7	81.6	72.6
$2D$Bounding Box	82.0	79.1	71.4	84.8	82.1	76.9
Semantic	**88.1**	**82.0**	**74.1**	**91.5**	**88.3**	**80.7**

Table 5.9: Average precision (%) for 3D detection and pose estimation of cars on VirtualKITTI [1] using PointRCNN2. 2D Bounding box refers to the GT bounding boxes within which depth pixels are sampled, while Semantic refers to GT semantics. Note that pixels within $r = 0.3$m radius of semantic pixels are also taken into consideration.

object detection performance significantly.

In Tab.5.9, We also experimented on finding an effective sampling region to select relevant object pixels; it shows that semantic regions, compared to $2D$ bounding boxes are most effective in improving the object detection performance in point-based backbone networks.

5.5 Conclusion

Existing 3D architectures for object detection are sensitive to noise from the pointclouds generated from depth estimates in image plane. Learning accurate depth in image plane is non-trivial since often real-life groundtruth data is noisy. We design a $sigma$ filter from TWISE that can minimize the *noise-inbet* pixels to a great extent. We also found out that estimated semantic masks from an external CNN network can suppress *noise-bg* pixels significantly. These filters can prune out noise from dense depth estimate quite effectively. We conclude that raw LiDAR, once augmented with pruned dense depth estimate, can contribute to improving object detection by gathering important details and context of *relevant* object pixels. In our future work, we plan to improve detection accuracy on foreground objects by designing a 3D pointcloud loss constraint, introducing 3D priors i.e. shapes and geometry knowledge of the scene.

Chapter 6

Conclusion and Future Works

6.1 Conclusions

Depth completion aims to recover dense depth maps from sparse depth measurements, guided by color image because of its high-resolution imagery. We introduce two novel representation of depth called multi-channel and dual channel representation; which, when incorporated effectively into a learning framework, are capable of improving depth completion performance significantly by recovering depth discontinuities. We also study an application of high-resolution depth in $3D$ object detection problem and show some potential advantages and pitfalls in using high-resolution depth estimates using SoTA $3D$ object detection problems. We conclude the thesis with our contributions, and possible avenues of future work in depth completion problems.

6.1.1 Contributions

In this section we list several contributions from the thesis.

6.1.1.1 Multi-Channel and Dual Channel Representation

We propose two novel representations of depth; multi-channel and dual channel representation that can model ambiguity at object boundaries. Multichannel representation can be expressed by probability distribution of each depth which can model ambiguity by multiple peaks. By using finite number of coefficients for depth reconstruction, it is possible to prevent floating pixels around

boundary completely. However, it needs to accommodate many channels for wide range of depths to preserve depth precision, resulting in high memory and computational requirements. We then introduce a new dual representation of depth, called twin surface, that learns a foreground and background depth for each pixel. Ideally, the foreground and background depth only differ around object boundaries when there is ambiguity and are same at object surfaces. It is possible to reconstruct depth by selecting foreground and background at object boundaries and select or mix any depths within object surfaces. Although some mixed depth pixels can still be present due to incorrect learning, it can drastically reduce memory requirement compared to depth coefficients. Both representation can be learned using standard neural network architectures.

6.1.1.2 Loss Functions

We propose some effective loss functions to learn the multichannel and dual channel representations. We showed that the cross-entropy loss is an effective measure to learn multichannel depth (DC), which is also a discrete probability distribution of depth. In order to learn DC estimate by cross-entropy loss, the groundtruth depth also needs to be converted to groundtruth DC. To learn dual channel representation, we designed two asymmetric loss functions; ALE and RALE, that can be used as biased estimator of foreground (FG) and background (BG) depth respectively. Additionally, we learn a sigma that is a weighted combination of FG and BG depth. It is possible to select a FG and BG depth at ambiguous region by adopting this strategy, and as a result, the final depth map can still preserve depth discontinuity at ambiguous regions.

6.1.1.3 Noisy GT and Metrics

RMSE is the standard and most common metric to evaluate depth completion methods since it favors far-away depth pixel errors compared to close-by depth pixels. With clean GT depth, it is

an effective mechanism to evaluate depth pixels at all ranges. However, real-life world has noisy GT due to different approaches in creating GT. For example, accumulation of lidar points from multiple frames causes depth noise in moving objects, and depth inconsistency from stereo depth cause several long-range depth pixels to omit in ground-truth. For indoor scenes, colorization techniques create smeared points in NYU2 groundtruth. We discovered in our analysis that MAE is the least susceptible metric to use in presence of GT outliers. We also propose two evaluation measures, TMAE and TRMSE metrics, which can reward better solutions at boundaries and within object surfaces by reducing the preference of outlier points and mixed depth pixel errors in GT and estimated depths respectively.

6.1.1.4 3D Object Detection from Noisy Depth

We apply depth estimates from depth completion in perception problems like $3D$ object detection and study how high-resolution noisy depth extended to pointcloud can affect object detection performances. Our analysis showed several depth noise and architecture limitations in SoTA method that are potential factors in degraded object detection performances. We propose a $sigma$ filter, and some additional remedies to handle noises common in depth estimate and conclude that depth estimates can be leveraged more effectively if it augments the sparse lidar by gathering dense depth points around local regions.

6.1.2 Future Improvements

Even though our proposed representation and learning framework can reduce significant smearing at object boundaries, we realize some potential limitations of our depth completion methods. Mixed depth pixels and depth noise still remain a concern for long-range depth where GT supervision is sparse and for heavily occluded regions which are not visible by the camera. Also

real-world GT like KITTI and NYU2 contains enough outlier noise resulting in noisy supervision. We also realize that since supervision is done in image plane, there are very few pixels that represent long-range depth compared to close-range pixels as a result of perspective distortion.

Creating clean high-resolution depth is paramount for quality depth completion and depth estimation. There are number of avenues that can be explored for improving depth completion. Weaker labels like $3D$ bounding boxes, pointcloud supervision by chamfer distance or $3D$ geometric constraints can also be used for additional supervision to tackle errors in long-range depth and highly occluded regions in image plane. We also realize that SoTA architectures do not necessarily support some high-resolution dense depth due to large number of depth points at high resolution. So our work can be extended to designing a more conducive network architecture for supporting dense depth. There are opportunities to develop applications of dense depth estimates in perception, navigation, generation of high-definition mapping problem to utilize the true potential of high-resolution depth.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] 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)*, 2012.

[2] 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)*, 2018.

[3] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from Lidar and monocular camera," *arXiv preprint arXiv:1807.00275*, 2018.

[4] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. European Conf. Computer Vision (ECCV)*, 2012.

[5] T. Huang, Z. Liu, X. Chen, and X. Bai, "Epnet: Enhancing point features with image semantics for 3d object detection," in *European Conference on Computer Vision*. Springer, 2020, pp. 35–52.

[6] A. Li, Z. Yuan, Y. Ling, W. Chi, s. zhang, and C. Zhang, "A multi-scale guided cascade hourglass network for depth completion," in *IEEE Workshop Application Computer Vision (WACV)*, March 2020.

[7] S. Imran, X. Liu, and D. Morris, "Depth completion with twin surface extrapolation at occlusion boundaries," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2583–2592.

[8] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8856–8865.

[9] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel r-cnn: Towards high performance voxel-based 3d object detection," *arXiv preprint arXiv:2012.15712*, 2020.

[10] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: self-supervised depth completion from lidar and monocular camera," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3288–3295.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[12] S. Imran, Y. Long, X. Liu, and D. Morris, "Depth coefficients for depth completion," *arXiv preprint arXiv:1903.05421*, 2019.

[13] J. Park, K. Joo, Z. Hu, C.-K. Liu, and I. S. Kweon, "Non-local spatial propagation network for depth completion," in *Proc. European Conf. Computer Vision (ECCV)*, 2020.

[14] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 1, no. 2, 2017, p. 3.

[15] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[16] B. Fortin, R. Lherbier, and J. Noyer, "A model-based joint detection and tracking approach for multi-vehicle tracking with Lidar sensor," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1883–1895, Aug 2015.

[17] 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.*, vol. 35, no. 5, pp. 1039–1050, 2013.

[18] T. Hassner and R. Basri, "Example based 3d reconstruction from single 2d images," in *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*. IEEE, 2006, pp. 15–15.

[19] A. Tuan Tran, T. Hassner, I. Masi, and G. Medioni, "Regressing robust and discriminative 3d morphable models with a very deep neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5163–5172.

[20] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. D. Stefano, "Real-time self-adaptive deep stereo," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 195–204.

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

[22] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 66–75.

[23] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 270–279.

[24] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep Ordinal Regression Network for Monocular Depth Estimation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2002–2011.

[25] J. Hu, M. Ozay, Y. Zhang, and T. Okatani, "Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1043–1051.

[26] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6647–6655.

[27] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4009–4018.

[28] Z. Cheng, Y. Zhang, and C. Tang, "Swin-depth: Using transformers and multi-scale fusion for monocular-based depth estimation," *IEEE Sensors Journal*, vol. 21, no. 23, pp. 26 912–26 920, 2021.

[29] J. Watson, M. Firman, G. J. Brostow, and D. Turmukhambetov, "Self-supervised monocular depth hints," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2162–2171.

[30] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838.

[31] K. Wang and S. Shen, "Mvdepthnet: Real-time multiview depth estimation neural network," in *2018 International conference on 3d vision (3DV)*. IEEE, 2018, pp. 248–257.

[32] X. Long, L. Liu, W. Li, C. Theobalt, and W. Wang, "Multi-view depth estimation using epipolar spatio-temporal networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8258–8267.

[33] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5667–5675.

[34] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *Int. Conf. 3D Vision (3DV)*, 2017.

[35] 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)*, Nov 2011, pp. 1623–1630.

[36] 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)*, 2018.

[37] Y. Long, D. Morris, X. Liu, M. Castro, P. Chakravarty, and P. Narayanan, "Radar-camera pixel depth association for depth completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 507–12 516.

[38] J. Diebel and S. Thrun, "An application of Markov Random Fields to range sensing," in *Advances in Neural Information Processing Systems (NIPS)*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds. MIT Press, 2006, pp. 291–298. [Online]. Available: http://papers.nips.cc/paper/2837-an-application-of-markov-random-fields-to-range-sensing.pdf

[39] 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)*, June 2007, pp. 1–8.

[40] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 1, June 2003.

[41] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[42] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 567–576.

[43] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[44] G. Riegler, M. Rüther, and H. Bischof, "ATGV-Net: Accurate depth super-resolution," in *Proc. European Conf. Computer Vision (ECCV)*. Springer, 2016, pp. 268–284.

[45] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[46] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2650–2658.

[47] 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)*. IEEE, 2018, pp. 1–8.

[48] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich, "Estimating depth from RGB and sparse sensing," in *Proc. European Conf. Computer Vision (ECCV)*, 2018.

[49] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *Int. Conf. 3D Vision (3DV)*. IEEE, 2017, pp. 11–20.

[50] 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)*. IEEE, 2017, pp. 5059–5066.

[51] X. Cheng, P. Wang, and R. Yang, "Depth estimation via affinity learned with convolutional spatial propagation network," in *Proc. European Conf. Computer Vision (ECCV)*, 2018.

[52] J. Huang, A. B. Lee, and D. B. Mumford, "Statistics of range images." Institute of Electrical and Electronics Engineers, 2000.

[53] S. Kalkan, F. Wörgötter, and N. Krüger, "Statistical analysis of local 3D structure in 2D images," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*. Citeseer, 2006, pp. 1114–1121.

[54] Y. Zhang and T. Funkhouser, "Deep depth completion of a single RGB-D image."

[55] P. Hu, J. Ziglar, D. Held, and D. Ramanan, "What you see is what you get: Exploiting visibility for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 001–11 009.

[56] 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)*. Springer, 2008, pp. 44–57.

[57] N. J. Mitra and A. Nguyen, "Estimating surface normals in noisy point cloud data," in *Proceedings of the nineteenth annual symposium on Computational geometry*. ACM, 2003, pp. 322–328.

[58] 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)*, 2018.

[59] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3577–3586.

[60] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2088–2096.

[61] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: a network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[62] M. Fey, J. Eric Lenssen, F. Weichert, and H. Müller, "Splinecnn: Fast geometric deep learning with continuous b-spline kernels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 869–877.

[63] L. Shao, Y. Tian, and J. Bohg, "Clusternet: Instance segmentation in RGB-D images," *arXiv preprint arXiv:1807.08894*, 2018.

[64] 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)*. Springer, 2014, pp. 345–360.

[65] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *In Proceeding of IEEE Computer Vision and Pattern Recognition*, 2017.

[66] Y. Tai, J. Yang, X. Liu, and C. Xu, "MemNet: A persistent memory network for image restoration," in *In Proceeding of International Conference on Computer Vision*, 2017.

[67] L. Sevilla-Lara and E. Learned-Miller, "Distribution fields for tracking," in *2012 IEEE Conference on computer vision and pattern recognition*. IEEE, 2012, pp. 1910–1917.

[68] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," in *European conference on computer vision*. Springer, 2006, pp. 568–580.

[69] M. Felsberg, P.-E. Forssén, and H. Scharr, "Channel smoothing: Efficient robust smoothing of low-level signal features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 2, pp. 209–222, 2005.

[70] E. G. Learned-Miller, "Data driven image models through continuous joint alignment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 2, pp. 236–250, 2005.

[71] G. Riegler, D. Ferstl, M. Rüther, and H. Bischof, "A deep primal-dual network for guided depth super-resolution," 2016.

[72] 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)*, 2013, pp. 993–1000.

[73] J. Lu and D. Forsyth, "Sparse depth super resolution," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2245–2253.

[74] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4340–4349.

[75] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[76] D. Hernandez-Juarez, L. Schneider, A. Espinosa, D. Vazquez, A. Lopez, U. Franke, M. Pollefeys, and J. C. Moure, "Slanted stixels: Representing san francisco's steepest streets," 2017.

[77] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[78] A. Patil, S. Malla, H. Gang, and Y.-T. Chen, "The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes," in *International Conference on Robotics and Automation*, 2019.

[79] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft level 5 av dataset 2019," urlhttps://level5.lyft.com/dataset/, 2019.

[80] 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.

[81] 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)*. IEEE, 2015, pp. 922–928.

[82] J. T. Barron and B. Poole, "The fast bilateral solver," in *Proc. European Conf. Computer Vision (ECCV)*, 2016.

[83] 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.*, vol. 38, no. 12, pp. 2457–2471, 2016.

[84] Y. Yang, A. Wong, and S. Soatto, "Dense depth posterior (ddp) from single image and sparse range," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3353–3362.

[85] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, B. Zeng, and M. Pollefeys, "Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3313–3322.

[86] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia, "Unsupervised learning of geometry from videos with edge-aware depth-normal consistency," 2018. [Online]. Available: https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16421

[87] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, "Depth completion from sparse lidar data with depth-normal constraints," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2811–2820.

[88] X. Cheng, P. Wang, C. Guan, and R. Yang, "Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion," in *Proc. AAAI Conf. Artificial Intelligence (AAAI)*, 2020.

[89] Z. Xu, H. Yin, and J. Yao, "Deformable spatial propagation networks for depth completion," in *Proc. Int. Conf. Image Processing (ICIP)*. IEEE, 2020, pp. 913–917.

[90] Y. Chen, B. Yang, M. Liang, and R. Urtasun, "Learning joint 2d-3d representations for depth completion," in *Proc. Int. Conf. Computer Vision (ICCV)*, 2019, pp. 10 023–10 032.

[91] R. Xiang, F. Zheng, H. Su, and Z. Zhang, "3ddepthnet: Point cloud guided depth completion network for sparse depth and single color image," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020.

[92] X. Xiong, H. Xiong, K. Xian, C. Zhao, Z. Cao, and X. Li, "Sparse-to-dense depth completion revisited: Sampling strategy and graph construction⋆."

[93] 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)*, 2017, pp. 3147–3155.

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

[95] J. Shade, S. Gortler, L.-w. He, and R. Szeliski, "Layered depth images," 1998, pp. 231–242.

[96] S. Tulsiani, R. Tucker, and N. Snavely, "Layer-structured 3d scene inference via view synthesis," in *Proc. European Conf. Computer Vision (ECCV)*, 2018, pp. 302–317.

[97] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf, "Casual 3D Photography," vol. 36, no. 6, pp. 234:1–234:15, 2017.

[98] T. Vogels, F. Rousselle, B. McWilliams, G. Röthlin, A. Harvill, D. Adler, M. Meyer, and J. Novák, "Denoising with kernel prediction and asymmetric loss functions," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–15, 2018.

[99] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[100] Y. Cabon, N. Murray, and M. Humenberger, "Virtual kitti 2," 2020.

[101] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

[102] X. Pan, Z. Xia, S. Song, L. E. Li, and G. Huang, "3d object detection with pointformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7463–7472.

[103] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, "Voxel transformer for 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3164–3173.

[104] Y. Wang, T. Shi, P. Yun, L. Tai, and M. Liu, "Pointseg: Real-time semantic segmentation based on 3d lidar point cloud," *arXiv preprint arXiv:1807.06288*, 2018.

[105] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[106] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, "Polarnet: An improved grid representation for online lidar point clouds semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9601–9610.

[107] C. Goodin, M. Doude, C. R. Hudson, and D. W. Carruth, "Enabling off-road autonomous navigation-simulation of lidar in dense vegetation," *Electronics*, vol. 7, no. 9, p. 154, 2018.

[108] T. Ort, I. Gilitschenski, and D. Rus, "Autonomous navigation in inclement weather based on a localizing ground penetrating radar," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3267–3274, 2020.

[109] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr, "Ga-net: Guided aggregation net for end-to-end stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 185–194.

[110] S. Zhu, G. Brazil, and X. Liu, "The edge of depth: Explicit constraints between segmentation and depth," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 116–13 125.

[111] X. Wang, M. H. Ang Jr, and G. H. Lee, "Cascaded refinement network for point cloud completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 790–799.

[112] G. Qian, A. Abualshour, G. Li, A. Thabet, and B. Ghanem, "Pu-gcn: Point cloud upsampling using graph convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 683–11 692.

[113] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: Point cloud upsampling network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2790–2799.

[114] B.-U. Lee, K. Lee, and I. S. Kweon, "Depth completion using plane-residual representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 916–13 925.

[115] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 040–11 048.

[116] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, "Structure aware single-stage 3d object detection from point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 873–11 882.

[117] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "Std: Sparse-to-dense 3d object detector for point cloud," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1951–1960.

[118] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[119] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[120] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[121] B. Li, "3d fully convolutional network for vehicle detection in point cloud," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1513–1518.

[122] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 7074–7082.

[123] L. Liu, J. Lu, C. Xu, Q. Tian, and J. Zhou, "Deep fitting degree scoring network for monocular 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1057–1066.

[124] A. Naiden, V. Paunescu, G. Kim, B. Jeon, and M. Leordeanu, "Shift r-cnn: Deep monocular 3d object detection with closed-form geometric constraints," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 61–65.

[125] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9287–9296.

[126] J. Fang, L. Zhou, and G. Liu, "3d bounding box estimation for autonomous vehicles by cascaded geometric constraints and depurated 2d detections using 3d results," *arXiv preprint arXiv:1909.01867*, 2019.

[127] Y. Chen, L. Tai, K. Sun, and M. Li, "Monopair: Monocular 3d object detection using pairwise spatial relationships," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 093–12 102.

[128] G. Brazil, G. Pons-Moll, X. Liu, and B. Schiele, "Kinematic 3d object detection in monocular video," in *European Conference on Computer Vision*. Springer, 2020, pp. 135–152.

[129] X. Ma, S. Liu, Z. Xia, H. Zhang, X. Zeng, and W. Ouyang, "Rethinking pseudo-lidar representation," in *European Conference on Computer Vision*. Springer, 2020, pp. 311–327.

[130] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[131] D. Park, R. Ambrus, V. Guizilini, J. Li, and A. Gaidon, "Is pseudo-lidar needed for monocular 3d object detection?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3142–3152.

[132] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2020.

[133] A. Paigwar, D. Sierra-Gonzalez, Ö. Erkent, and C. Laugier, "Frustum-pointpillars: A multi-stage approach for 3d object detection using rgb camera and lidar," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2926–2933.

[134] J. H. Yoo, Y. Kim, J. Kim, and J. W. Choi, "3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*. Springer, 2020, pp. 720–736.

[135] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 641–656.

[136] S. Pang, D. Morris, and H. Radha, "Clocs: Camera-lidar object candidates fusion for 3d object detection," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 386–10 393.

[137] X. Weng and K. Kitani, "Monocular 3d object detection with pseudo-lidar point cloud," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[138] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8445–8453.

[139] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," *arXiv preprint arXiv:1906.06310*, 2019.

[140] R. Qian, D. Garg, Y. Wang, Y. You, S. Belongie, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, "End-to-end pseudo-lidar for image-based 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5881–5890.

[141] S. Luo and W. Hu, "Score-based point cloud denoising," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4583–4592.

[142] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, and A. Yuille, "Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding," *arXiv preprint arXiv:1810.06125*, 2018.

[143] J. Gu, Z. Xiang, Y. Ye, and L. Wang, "Denselidar: A real-time pseudo dense depth guided depth completion network," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, p. 1808–1815, Apr 2021. [Online]. Available: http://dx.doi.org/10.1109/LRA.2021.3060396

[144] O. D. Team, "Openpcdet: An open-source toolbox for 3d object detection from point clouds," https://github.com/open-mmlab/OpenPCDet, 2020.