Flexible Depth Completion for Sparse and Varying Point Densities

Jinhyung Park[1] Yu-Jhe Li[2] Kris Kitani[1]
[1] Carnegie Mellon University [2] Microsoft Research

Abstract

While recent depth completion methods have achieved remarkable results filling in relatively dense depth maps (e.g., projected 64-line LiDAR on KITTI or 500 sampled points on NYUv2) with RGB guidance, their performance on very sparse input (e.g., 4-line LiDAR or 32 depth point measurements) is unverified. These sparser regimes present new challenges, as a 4-line LiDAR increases the distance between pixels without depth and their nearest depth point sixfold from 5 pixels to 30 pixels compared to 64 lines. Observing that existing methods struggle with sparse and variable distribution depth maps, we propose an Affinity-Based Shift Correction (ASC) module that iteratively aligns depth predictions to input depth based on predicted affinities between image pixels and depth points. Our framework enables each depth point to adaptively influence and improve predictions across the image, leading to largely improved results for fewer-line, fewer-point, and variable sparsity settings. Further, we show improved performance in domain transfer from KITTI to nuScenes and from random sampling to irregular point distributions. Our correction module can easily be added to any depth completion or RGB-only depth estimation model, notably allowing the latter to perform both completion and estimation with a single model.

1. Introduction

Recent advances in depth completion have focused on densifying depth maps created by projecting 64-line LiDAR onto the RGB image in the KITTI dataset [14, 49] or by randomly sampling 500 depth points from a Kinect RGBD camera in the NYUv2 dataset [44]. While demonstrating strong performance on dense input regimes, these methods remain largely untested on sparser depth maps produced by affordable, few-line LiDAR and by sparse valid points ($\sim$ 0.04% pixels) from SfM pipelines [41, 42]. To support various sensors and applications, models must also be evaluated on more difficult sparse and out-of-distribution settings.

Such sparser regimes pose new challenges. As shown in Figure 1, while most image pixels are within 5 pixels of input depth with projected 64-line LiDAR, this distance increases *sixfold* with cheaper LiDAR sensors. In Figure 2, we show that prediction error dramatically increases with larger such pixel distances. Although this deterioration is expected, some methods that perform well for dense input are comparatively weaker for sparse input. For more reliable real-world deployment and wider applicability, completion models should demonstrate strong performance under a wide range of input conditions.

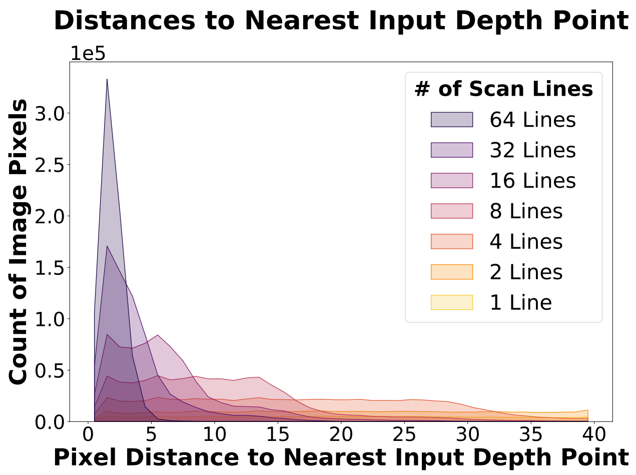

Figure 1. The distance from each pixel to its nearest depth point is less than 5 pixels for the 64-line LiDAR most commonly used for depth completion on KITTI. When switching to a more affordable and deployable 4-line LiDAR, this distance increases dramatically to over 30 pixels, posing new difficulties for depth completion.

In this regime, existing methods' primarily convolutional processing of depth points is not suitable for input of variable sparsity and distributions [16, 25, 36, 47]. For sparse depth maps with a larger variance in pixel-to-point distance, each depth point must contribute to depth predictions for large, or even global, regions of the image, adapting its region of influence based on the layout of input depth points and its feature proximity to different locations in the image. Towards this end, we develop a novel depth completion framework that leverages predicted affinity between image locations and depth points to iteratively improve depth predictions. Our proposed Affinity-Based Shift Correction (ASC) module determines the depth error of related regions and shifts the depth predictions of these groups to align with input depth points. Coupled with a correction confidence weighting step, our module substantially improves depth completion performance.

We extensively benchmark representative models and

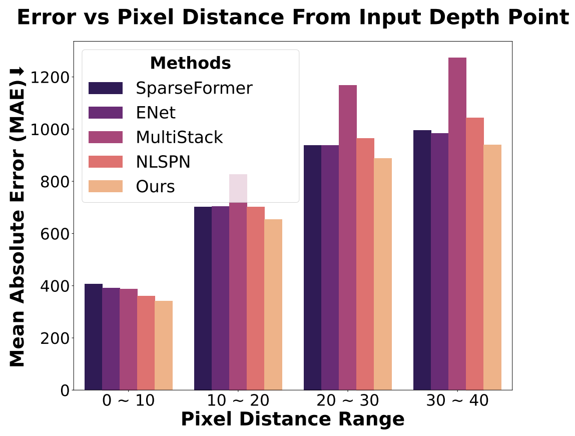

Figure 2. Performance of depth completion methods over increasing distances to nearest depth point in KITTI. MAE↓ is in mm.

our pipeline on a battery of settings. To support other applications [41, 55] and datasets [3], completion models should perform well over different scan-lines and sampling patterns. For instance, using models trained on 64-line KITTI to generate depth maps on 32-line nuScenes [3] results in significant artifacts [55] and cannot directly be used for NeRF applications [35, 55]. Similarly, models trained with random points on NYUv2 fail when using SIFT [32] or SfM keypoints [53]. As such, beyond training and evaluating at individual sparsity levels (64 $\sim$ 1-line LiDAR, 500 $\sim$ 2 points), we also examine training a single model for variable sparsity levels, evaluating on SIFT [32] keypoints, testing on unseen point patterns, and cross-dataset transfer. Further, our ASC module naturally enables joint depth estimation and completion with a single model, and we demonstrate wide applicability to various RGB-only estimation models.

Our main contributions are as follows:

- We propose a novel depth correction module that iteratively refines predictions using pixel-point affinity.
- We benchmark and outperform prior work on an extensive suite of real-world settings.
- We verify that training with variable sparsity produces transferable depth completion models by evaluating completion on nuScenes and on SIFT keypoints.
- Our correction module can easily be applied to any depth completion or RGB-only estimation model.

2. Related Work

2.1. Depth Estimation and Completion

Depth estimation seeks to produce dense depth maps using a single image [2, 13, 24, 37, 64], multiple views [1, 21, 60, 61], or additional modalities [7, 9, 15, 22, 33, 36, 38, 39]. Depth completion [10, 15, 19, 36, 47] uses an RGB image and a sparse depth map from a LiDAR sensor, a depth camera, or an SfM pipeline. The early work S2D [33] uses an encoder-decoder CNN, and followup methods use coarse-to-fine refinement [18, 25, 27] and multi-scale models [16, 25, 59]. Some methods use geometric constraints such as surface normals [38, 58] or planes [23], while others do processing in 3D space [5, 17, 20, 29, 56, 65]. To generate more locally consistent depth, some works build on spatial propagation networks [28] to predict the affinity of each pixel with its local [7, 8, 16], deformable [26, 36], or 3D [29] neighbors. While these methods demonstrate good performance for fixed, semi-dense input, their strong local bias in propagating depth information hinders their application to fewer-line, fewer-point, and variable sparsity settings. Our module allows points to interact with regions across the image and is applicable to any completion model, making it complementary to these methods.

While SparseFormer [52] considers the relationship between points and pixels, they take a simple weighted sum over input depths for each pixel. This results in low-quality intermediate predictions and constrains the region that each depth point can affect. In contrast, we propose a shift correction module using affinities between pixels and depth points and demonstrate stronger performance in all settings.

Also relevant to our work is monocular depth estimation [2, 13, 24, 40, 54]. As predicting depth from a single image is fundamentally ambiguous, existing works predict scale [13] or scale-shift invariant depth maps [40] and globally align predictions to ground truth depth maps for evaluation. One method [57] locally adjusts monocular predictions using sparse points using a fixed gaussian kernel on distance to compute affinities. Our method instead adapts affinities based on semantic features and input point distributions, making it more flexible without bandwidth tuning.

2.2. Depth Completion for Various Sparse Settings

While most depth completion methods evaluate on 64-line KITTI [14, 48] and 500-point NYUv2 [43], some works consider other settings. SparseSPN [53] proposes to train on SIFT keypoints [42]. SpAgNet [10] develops a sparsity-agnostic model, training on 500 points and testing on fewer points. Our method outperforms these works under similar conditions, and we further find that training on a variable number of points transfers well even to unique point distributions. A few methods [17, 18, 52] train and evaluate on individual sparsity settings. We compare with these works and improve the benchmark by carefully selecting the line configuration for fewer lines on KITTI and by *re-training* baselines on each setting. Another method considers developing domain-agnostic models [63] using monocular models trained on many datasets [40, 62]. Our model does not use pretrained monocular models, and we also consider different LiDAR scan-lines and extremely sparse depth maps.

3. Method

3.1. Base Depth Estimation Architecture

Our Affinity-based Shift Correction (ASC) module can be applied to any depth estimation method with an encoder and decoder as shown in Figure 3(a-b). Deserving careful con-

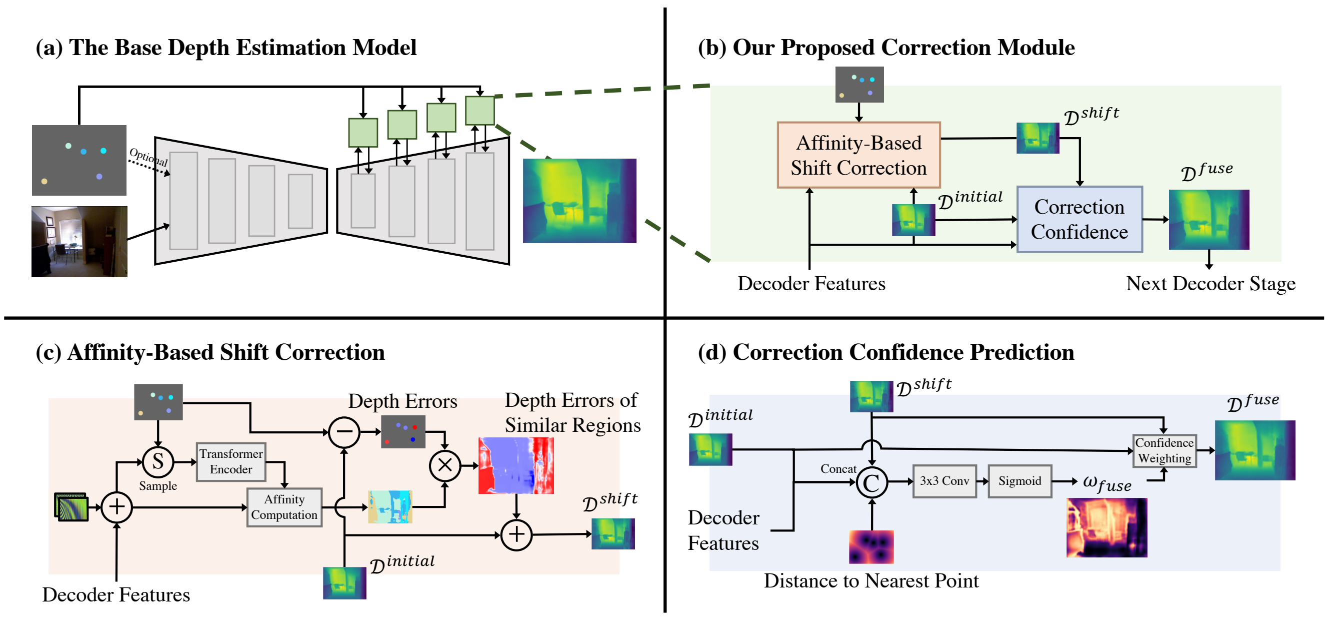

Figure 3. **Overall Framework. (a)** Our correction module can be applied to the decoder of any depth estimation model. **(b)** Each module adaptively propagates depth information by refining initial depth predictions and fusing them back into decoder features. **(c)** The Affinity-Based Shift Correction aligns predictions to input depth using depth errors of similar regions. **(d)** Correction Confidence Prediction enables selection between context-based and input depth corrected predictions.

sideration are the input modalities to our encoder. Leading depth completion methods generally must input both the RGB image and the sparse depth map into the encoder [16, 25, 33, 36, 65] as they use this input-level fusion as the main source of depth information. However, our module fuses depth at the decoder stage, allowing the model to perform completion even only with RGB input to the encoder. We find that while RGBD input outperforms RGB input for denser inputs and matching train/test distributions, RGB input is better at handling unseen sparsity levels and distributions. In this work, we apply our ASC module to a basic depth completion model, with or without NLSPN [36], as well as two representative monocular depth estimation architectures, BTS [24] and AdaBins [2] (Sec. 3.4).

3.2. Affinity-Based Shift Correction

Given a feature map of a decoder stage, our ASC module shown in Figure 3(c) adaptively propagates depth information from each input point to across the entire image. For a single decoder stage, let $\mathcal{F}$ denote the image feature map of shape $H' \times W' \times C$ and let $\mathcal{P} = \{(p_j, d_j)\}_{j=1}^{N}$ denote the list of N input depth points, where p_j and d_j are the 2D projection and the depth of the j-th input depth point, respectively. We first predict an initial depth map $\mathcal{D}^{initial} \in \mathbb{R}^{H' \times W' \times 1}$ from $\mathcal{F}$ using an MLP. At a high level, our module will align this initial prediction to the input depth points and fuse it back into $\mathcal{F}$ for the next decoder stage.

Affinity Computation. Suppose we are given a depth point on a cabinet as input. To align our depth predictions $\mathcal{D}^{initial}$ with the input depth points, we cannot merely correct our prediction for that single pixel. Instead, we utilize the single input point as a reference to which we align our depth predictions for the entire relevant region. To achieve this, we identify the regions in the image for which each depth point should act as a reference point. We compute the affinity between each pair of image pixels and depth points, where the affinity represents the extent to which each depth point should contribute to the alignment of each image pixel.

We observe that the range of influence of each depth point must also depend on the distribution and number of input points. For example, between 64-line and 4-line LiDAR, the distance between each image pixel and its nearest depth point varies from 5 to 30 pixels. To allow points to calibrate their regions of influence similar to object queries [4, 6, 30, 51], we generate features for each depth point by adding 2D positional embeddings [12, 45, 50] to the image features, denoted $\mathcal{F}'$, sampling image features at each depth point projection, and leveraging a single transformer layer:

$$\{f_j^P\}_{j=1}^{N} = \text{TransformerEncoder}\left(\{[\mathcal{F}'[p_j], d_j]\}_{j=1}^{N}\right) \tag{1}$$

where $\mathcal{F}'[p_j]$ indicates bilinearly sampling $\mathcal{F}'$ at position p_j and $[\cdot, \cdot]$ denotes concatenation. Then, employing the cross-attention mechanism, the affinity between image pixel feature f_i^I and input depth point (p_j, d_j) is:

$$\mathcal{A}_{ij} = \frac{\exp((W_q f_i^I)^T (W_k f_j^P))}{\sum_{m=1}^{N} \exp\left((W_q f_i^I)^T (W_k f_m^P)\right)} \tag{2}$$

Shift Correction. Using these affinities, we create a shift-corrected depth map $\mathcal{D}^{shift}$, which corrects each pixel in $\mathcal{D}^{initial}$ using depth errors of similar pixels. Specifically,

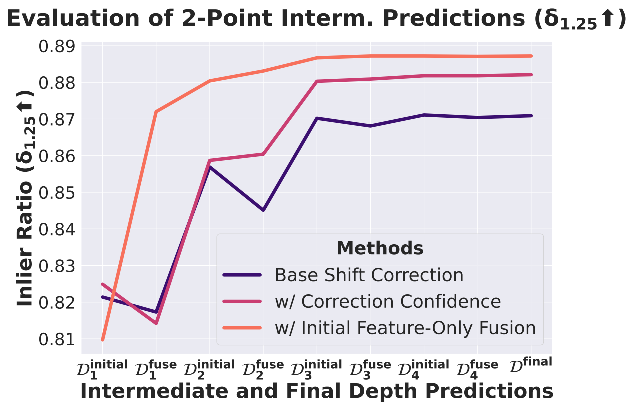

Figure 4. Evaluation of intermediate depth predictions through decoder stages of the model. Using Correction Confidence and Initial Feature-Only Fusion enables more consistent improvement.

we find the shift-corrected depth of pixel i:

$$\mathcal{D}_i^{shift} = \mathcal{D}_i^{initial} + \sum_{j=1}^{N} \mathcal{A}_{ij}(d_j - \mathcal{D}_j^{initial}) \tag{3}$$

The second term is the weighted average of depth errors in the initial depth prediction for pixels j that have input depth points, where the weights are each pixel j's affinity, to pixel i. Intuitively, if pixel i is in a region of the image generally predicted to be too close, the depth prediction for pixel i will be shifted accordingly. By supervising $\mathcal{D}^{shift}$, each depth point learns to adaptively influence regions for which it can serve as an effective reference for alignment. In addition to shift correction, we also use the affinities to take a weighted sum over the point features [52] to get a feature map $\mathcal{F}^{point}$. This feature map and the shift corrected depth map are fused with the initial decoder features and are used as input to the next decoder stage.

Initial Feature-Only Fusion. Although applying the ASC module to each decoder layer improves performance, the shift-corrected predictions of the first, coarsest decoder layer have poor quality when using RGB input. This is because the initial predictions for this coarsest decoder layer are purely monocular and thus suffer from scale ambiguity. Equation 3 can be re-written as:

$$\mathcal{D}_i^{shift} = \sum_{j=1}^{N} \mathcal{A}_{ij}(\mathcal{D}_i^{initial} - \mathcal{D}_j^{initial}) + \sum_{j=1}^{N} \mathcal{A}_{ij} d_j \tag{4}$$

The first term is the relative depth prediction of pixel i w.r.t similar pixels j. This relative depth is applied to the actual input depths of similar pixels j to get the depth prediction for pixel i. Due to inherent scale ambiguity in the initial depth predictions and thus these relative depths, the shift correction yields poor results for the first decoder stage. Instead, just fusing the weighted sum of depth point features for the first decoder stage improves results and yields scale-consistent predictions for subsequent decoder stages, which can then effectively do shift correction as shown in Fig. 4.

3.3. Correction Confidence Prediction

While the ASC module allows each depth point to adaptively rectify initial predictions, we find that with fewer than five input depth points, shift-corrected predictions are sometimes worse for certain regions than the initial predictions. The causes are twofold. First, if the initial depth predictions for the pixels with input depth are poor, their errors are propagated to other regions as well. This problem is more significant with fewer points, where each pixel cannot reference many input depth points for correction. Second, certain depth points cannot benefit depth prediction for the entire image. For instance, although a point on a cabinet may improve predictions for the cabinet, the wall behind it, and the floor beneath it, it has little relationship to objects in an adjacent room. For these regions, the initial depth prediction based on surrounding context can be better.

Motivated by these observations, as shown in Figure 3(d), we combine the initial and corrected depth predictions and fuse only the best from each depth map. Specifically, the depth map fused into the decoder feature is:

$$\mathcal{D}^{fuse} = (1 - \omega_{fuse}) \circ \mathcal{D}^{initial} + \omega_{fuse} \circ \mathcal{D}^{shift} \tag{5}$$

$$\omega_{fuse} = \sigma(\phi_\theta([\mathcal{F}', \mathcal{F}^{point}, \mathcal{D}^{initial}, \mathcal{D}^{shift}, \mathcal{F}^{dist}])) \tag{6}$$

where $\mathcal{F}^{dist}$ is the normalized distance of each pixel to its nearest depth point, ϕ_θ is a lightweight CNN head, σ is the sigmoid function, and ω_{fuse} is a 1-channel confidence map. This predicted confidence significantly improves performance for few-point regimes as shown in Figure 4.

3.4. Joint Depth Estimation and Completion

As discussed in Section 3.1, the ASC module allows for depth completion with an RGB-input encoder. Thus, our method is easily applicable to monocular depth estimation models, enabling a single model to flexibly do both monocular depth estimation and completion depending on availability of depth points. We train this unified model by sampling from estimation and completion tasks and by bypassing the ASC modules for the depth estimation task. The final model demonstrates good performance on both tasks.

3.5. Losses

In total, our pipeline produces nine depth maps - four initial predictions $\mathcal{D}^{initial}$, four fused predictions $\mathcal{D}^{fuse}$, and one final prediction $\mathcal{D}^{final}$. The overall loss is:

$$\mathcal{L} = \mathcal{L}_{final} + \sum_{i=1}^{4} \alpha_i \left(\mathcal{L}_{initial}^i + \mathcal{L}_{fuse}^i\right) \tag{7}$$

where for instance $\mathcal{L}_{final}$ denotes the depth loss on $\mathcal{D}^{final}$ and α_i weights the losses of each scale.

4. Experiments

4.1. Datasets and Evaluation Metrics

We closely follow previous work [5, 34, 36, 47, 59] for general dataset settings. We refer readers to Section B of the supplementary for full details on the datasets.

KITTI. We simulate fewer-line LiDAR by subsampling 64-line LiDAR. Notably, for 1 to 8-line LiDAR, the pitch of the simulated sensor significantly impacts completion results. Unlike prior work [18], we select the pitch maximizing performance to mirror realistic scenarios where the sensor is placed to maximize scene coverage. To avoid bias, a separate MiDAS-like [40] RGBD model is used to select pitch.

NYUv2. By default, we randomly sample 2, 8, 32, 200, and 500 input points over the depth map. To further model real-world point distributions, we also benchmark with SIFT keypoints, which are often clustered around corners.

VOID. We use the official 150, 500, and 1500 depth points tracked by a real-world EKF-based VIO SLAM system.

Evaluation. In the main paper, we focus on quantitative results. Extensive visualizations and qualitative analysis are in the supplementary. As in prior work [16, 20, 36, 59], for KITTI & VOID, we report RMSE↓ and MAE↓ (in mm), and for NYUv2, we report RMSE↓ (meters), REL↓, and $\delta_{1.25}$↑.

4.2. Overview of Experiments

We extensively benchmark on a suite of real-world settings:

- In Section 4.3, we ablate the components of our method.
- In Sections 4.4 and 4.5, we test on KITTI and NYUv2. Notably, we consider two settings. In the "**Fixed**" sparsity setting, a separate model is trained & evaluated for each input depth distribution. In the "**Variable**" sparsity setting, a single model is jointly trained over all input distributions by randomly sampling # of lines for KITTI or # of points for NYUv2 in each training iteration. By evaluating a single model on multiple input distributions, this "Variable" setting models a realistic setup where depth sparsities may vary during deployment.
- In Section 4.6, we test on real-world indoor distributions: SIFT points on NYUv2 and SLAM-tracked points on VOID. On VOID, we show that our method is more robust when testing on sparser inputs than during training.
- In Section 4.7, we focus on unseen sparsity levels and patterns. We show that 1) our model is more robust under input distribution shifts, and 2) training on "Variable" sparsity is a simple training scheme that yields transferable models, validating our main experimental setting.
- In Section 4.8, we apply our ASC module to various RGB-only depth estimators for joint estimation & completion, and we also demonstrate a downstream application for 3D detection on variable scan lines.

Implementation Details. We emphasize that, for fair comparison, we carefully **re-train all methods** for sparser regimes, instead of applying 64-line or 500-point checkpoints. Due to the size of the KITTI dataset, we train on 6x temporally sampled frames [18]. The partially scale-invariant loss [13] is used for $\mathcal{D}^{initial}$. For $\mathcal{D}^{fuse}$ and $\mathcal{D}^{final}$, we use ℓ_1 for NYUv2 and both ℓ_1 and ℓ_2 for KITTI [36]. More details are in Section C of the supplementary.

Components	# of Sampled Points				
	2	8	32	200	500
Features (f_j^P)	0.106	0.076	0.050	0.024	0.017
Depths (d_j)	0.136	0.099	0.064	0.039	0.033
Depth Errors ($d_j - \mathcal{D}_j^{initial}$)	0.130	0.084	0.050	0.023	0.016
Features + Depths	0.108	0.078	0.050	0.025	0.018
Features + Depth Errors	0.115	0.078	0.049	0.023	0.016

Table 1. Ablation on different uses of affinity. Metric is REL↓.

4.3. Ablation Study

We first validate design choices in our Affinity-based Shift Correction module. To verify how effectively the module propagates depth information, the base model in the first two ablations is an RGB-input encoder-decoder with an EfficientNetB5 [46] backbone. We train with variable sparsity, and we evaluate on **REL↓**. Full tables with all metrics and visualizations are in Section D.1 of the supplementary.

Different Uses of Affinity. In our final model, we use affinities from Equation 2 to take a weighted sum over features and depth errors for correction. We ablate each component and also test a simpler weighted sum over just depths. We start with a baseline encoder-decoder using our module without correction confidence and initial feature-only fusion and with $\ell_1 + \ell_2$ supervision. We then consider fusion of an affinity-weighted sum of depth features, depths, or depth errors. Results are in Table 1.

When using a single component, taking a weighted sum over features is best for few points (2 & 8) while weighted sum over the depth errors is best for more points (200 & 500). A simple weighted sum over input depths performs poorly especially with more points as it blurs depth boundaries. With two components, using features with depth errors is better for 32, 200, and 500 points and using features with input depths is better for 2 points. As per analysis in Section D.2 of the supplementary, when fusing weighted sum over features and depths, the model just uses the low-quality and blurry intermediate depth map as a vague depth prior. On the other hand, fusing features and depth errors causes the model to be over-confident in the corrected depth map $\mathcal{D}^{shift}$ for fewer points as discussed in Section 3.3.

Correction Confidence and Initial Feature-Only Fusion. To address this issue, in Table 2, we add confidence prediction ω_{fuse} and find that predicting ω_{fuse} with knowledge of the distance from each pixel to the nearest depth point improves few-point regimes. This corroborates our analysis in Section 3.3 that depth of some regions of the image in the extreme sparsity setup are best predicted just by using contextual information. Supervising initial direct depth predictions with the partially scale-invariant loss [13] for

Components	# of Sampled Points				
	2	8	32	200	500
Features + Depth Errors	0.115	0.078	0.049	0.023	0.016
+ ω_{fuse} w/o $\mathcal{F}^{dist}$	0.113	0.078	0.049	0.023	0.016
+ $\mathcal{F}^{dist}$	0.108	0.078	0.049	0.023	0.016
+ Partial SI-Loss for $\mathcal{D}^{initial}$	0.108	0.077	0.047	0.023	0.016
+ Initial Feature-only fusion	0.102	0.073	0.047	0.023	0.016
+ ℓ_1 for $\mathcal{D}^{final}$ and $\mathcal{D}^{fuse}$	0.098	0.069	0.044	0.022	0.015

Table 2. Ablation on correction confidence, initial feature-only fusion, and loss functions. Metric is REL↓.

Backbone	w/ Depth Enc. Input	NLSPN	# of Sampled Points				
			2	8	32	200	500
Res34	✗	✗	0.110	0.078	0.047	0.022	0.015
Res34	✓	✗	0.134	0.086	0.048	0.022	0.016
Res34	✓	✓	0.132	0.084	0.047	0.021	0.014
Res34+	✗	✗	0.131	0.086	0.049	0.023	0.017
Res34+	✓	✗	0.133	0.081	0.044	0.020	0.014
Res34+	✓	✓	0.127	0.078	0.043	0.019	0.013
Original NLSPN Model			0.132	0.084	0.044	0.019	0.013
Effb5	✗	✗	0.098	0.069	0.044	0.022	0.015
Effb5	✓	✗	0.101	0.071	0.045	0.023	0.017
Effb5	✓	✓	0.098	0.069	0.043	0.021	0.014

Table 3. Ablating backbones, inputs, and NLSPN. Metric is REL↓.

easier optimization improves performance slightly. Then, switching the first correction module to feature-only fusion to address scale-ambiguity before correction improves performance, supporting our analysis in Section 3.2. Finally, using ℓ_1 loss for supervising $\mathcal{D}^{final}$ and $\mathcal{D}^{fuse}$ on NYUv2 following NLSPN [36] further improves performance.

Backbone, Input Modalities, and NLSPN Head. In Table 3, we ablate backbones, input modalities, and the NLSPN head. ResNet34+ is the backbone of NLSPN, without the stem's stride 2 convolution and max-pooling. Overall, the high-res ResNet34+ is better for denser input but worse for fewer points. Effb5 performs the best for few points, supporting our intuition that with fewer points, contextual information from the image is more important. RGB encoder input is best for few points, but is worse than RGBD + NLSPN for dense points. RGB input with NLSPN head is excluded as it generally performs worse than just RGB.

Next, the results of the original NLSPN model are in row 7. Adding our module (row 6) significantly outperforms the original NLSPN model for sparser settings. We exhaustively further verify the improvement from our module in more complex and realistic settings in the main results sections. Overall, our ASC module can be easily applied to models with different backbones and input modalities. Notably, our module is the only source of depth information for RGB-input settings, and they compare favorably even against strong RGBD-input depth completion models that run a deep CNN encoder on depth maps.

CSPN vs NLSPN Refinement. In Table 4, we show that performance of our pipeline improves by adding the CSPN [7] head, then further by adding the NLSPN head [36]. This demonstrates that our proposed ASC module is complementary to improvements in spatial propagation refinement.

Components	# of Sampled Points				
	2	8	32	200	500
R34+ RGBD	0.1327	0.0805	0.0435	0.0198	0.0140
w/ CSPN	0.1332	0.0802	0.0428	0.0193	0.0134
w/ NLSPN	0.1272	0.0783	0.0425	0.0190	0.0130

Table 4. Ablation on refinement head. Metric is REL↓.

4.4. Evaluation on the KITTI Dataset

In Table 5, we compare methods in both "Fixed" and "Variable" settings, as indicated on the left side of the table. Ours (NLSPN Base) has the same architecture as NLSPN but with our ASC modules. In the "Fixed" setting, our method outperforms existing work in for all scan lines, especially for fewer-line settings. The gap is even larger when training a single model on variable scan lines, and switching from Res34+ to Effb5 improves performance for sparser settings, supporting our analysis from ablating backbones. Finally, although not our focus, we include a full leaderboard comparison on the very dense 64-line LiDAR in the supplementary and find that our method performs competitively.

4.5. Evaluation on the NYUv2 Dataset

Results for NYUv2 are in Table 6. As in KITTI, we took paper results whenever possible and **re-trained baselines** [20, 25, 36, 52] **on each setting** otherwise. On NYUv2, we compare across fixed and variable settings because performance improves on sparser input (2, 8, 32 pts) when training a single model on variable # of points, and using a single model for multiple sparsities imposes no additional costs.

Our method achieves state-of-the-art performance, with especially notable improvements in sparser settings over NLSPN, our base model, and prior work. We see a similar trade-off as in KITTI for sparse vs. dense performance when switching from Res34+ to Effb5, which we explore further in Section 4.8. A full leaderboard comparison for the largely saturated 500-point setting is in the supplementary. In all, we verify that our proposed ASC module improves performance remarkably for more difficult fewer-point and variable-sparsity settings.

4.6. Evaluation on Real-World Indoor Distributions

SIFT Keypoints. To evaluate our model on uneven distributions similar to SfM points [32, 42], we follow SparseSPN settings in training and testing on SIFT keypoints on NYUv2 as shown in Table 7. Our pipeline outperforms CSPN and NLSPN and also achieves superior results to SparseSPN without a surface normal ViT-B network.

SLAM Keypoints. In Table 8 we test with points tracked by a VIO system. We verify 1) our ASC module improves NLSPN for uneven, real-world distributions, 2) our ASC module also *transfers* better to unseen sparsity levels (e.g., 150,

	Method	1 Line		2 Line		4 Line		8 Line		16 Line		32 Line		64 Line	
		RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Fixed Scan Line	MultiStack [25]	2505.9	1146.0	2456.2	1009.7	2180.0	766.6	1725.4	519.0	1363.3	377.5	1065.6	280.2	869.7	220.9
	GuideNet [47]	2252.7	1007.9	2223.3	900.9	2038.7	752.0	1628.2	541.7	1293.1	411.7	1011.9	300.5	850.0	250.5
	NLSPN [36]	2069.3	873.9	2020.1	782.5	1851.2	627.1	1496.9	446.7	1215.3	333.8	990.3	258.0	833.5	208.8
	ENet [16]	2077.9	868.7	2023.5	773.8	1847.7	622.5	1522.0	452.7	1236.5	351.5	1004.7	303.3	846.9	229.3
	SparseFormer [52]†	2065.9	900.2	2009.1	801.7	1836.8	639.6	1500.9	472.6	1228.0	365.4	1017.3	287.6	891.5	238.3
	Ours (NLSPN Base)	**2015.1**	**849.9**	**1954.4**	**738.6**	**1798.1**	**599.9**	**1458.3**	**435.2**	**1202.1**	**324.8**	**970.2**	**253.9**	**818.1**	**205.7**
Variable Scan Lines	MultiStack [25]	2547.3	1181.4	2487.3	1058.6	2224.7	806.7	1792.3	567.1	1440.9	419.9	1139.7	319.0	956.0	258.4
	GuideNet [47]	2385.4	1099.8	2314.6	974.5	2205.3	876.0	1940.9	741.9	1634.6	622.7	1297.7	475.4	1027.9	346.6
	NLSPN [36]	2195.5	968.0	2150.0	874.8	1963.3	690.2	1605.4	505.2	1319.9	387.4	1074.6	305.0	923.2	256.1
	ENet [16]	2126.0	937.0	2084.6	831.6	1918.2	676.2	1589.8	514.2	1310.3	409.1	1074.4	337.2	931.7	300.2
	SparseFormer [52]†	2128.6	988.8	2067.7	862.9	1893.2	691.5	1571.9	525.7	1297.4	420.9	1080.4	352.7	947.0	308.8
	Ours (NLSPN Base)	2051.9	913.2	2015.7	788.4	**1829.0**	633.9	**1529.2**	**475.1**	**1258.9**	**367.6**	**1033.2**	**290.6**	**884.8**	**237.1**
	Ours (NLSPN Effb5 Base)	**2032.7**	**830.1**	**1997.3**	**750.9**	1873.8	**627.9**	1611.5	481.6	1333.1	373.4	1090.4	295.2	932.3	244.5

Table 5. Evaluation on the KITTI dataset. For fixed scan lines, we train and test on same scan lines. For variable scan lines, we train sampling from 1 to 64 line LiDAR. **Best** and second best results are emphasized. †Model re-implemented by us.

	Method	2 Points			8 Points			32 Points			200 Points			500 Points		
		$\delta_{1.25}\uparrow$	REL↓	RMSE↓	$\delta_{1.25}\uparrow$	REL↓	RMSE↓	$\delta_{1.25}\uparrow$	REL↓	RMSE↓	$\delta_{1.25}\uparrow$	REL↓	RMSE↓	$\delta_{1.25}\uparrow$	REL↓	RMSE↓
Fixed # of Points	S2D [34]	-	-	-	-	-	-	-	-	-	0.971	0.044	0.230	-	-	-
	GuideNet [47]	-	-	-	-	-	-	-	-	-	0.988	0.024	0.142	0.995	0.015	0.101
	MultiStack [25]	0.7451	0.1677	0.6014	0.8825	0.0997	0.4212	0.9578	0.0501	0.2626	0.9890	0.0200	0.1403	0.9949	0.0126	0.0972
	NLSPN [36]	0.7736	0.1461	0.5572	0.9022	0.0892	0.3982	0.9658	0.0441	0.2469	0.9904	0.0185	0.1332	**0.9955**	0.0117	0.0924
	PointFusion [17]	0.875	0.109	0.470	-	-	-	0.963	0.057	0.319	**0.995**	**0.015**	**0.112**	**0.996**	0.014	**0.090**
	SparseFormer [52]	0.740	0.161	0.626	-	-	-	0.962	0.050	0.255	0.989	0.022	0.142	0.994	0.014	0.104
	SparseFormer [52]†	0.7638	0.1549	0.5807	0.8918	0.0988	0.4109	0.9619	0.0500	0.2575	0.9898	0.0218	0.1371	0.9951	0.0151	0.0991
	CostDCNet [20]	0.8042	0.1421	0.5454	0.9077	0.0884	0.3970	0.964	0.048	0.258	0.9889	0.0209	0.1429	0.995	0.013	0.096
	Ours (NLSPN Base)	0.8080	0.1353	0.5232	0.9166	0.0811	0.3647	0.9680	0.0433	0.2407	0.9905	0.0182	0.1315	**0.9956**	**0.0115**	0.0917
Variable # of Points	MultiStack [25]	0.7253	0.1838	0.6395	0.8604	0.1140	0.4598	0.9531	0.0552	0.2772	0.9874	0.0222	0.1482	0.9934	0.0150	0.1093
	NLSPN [36]	0.8220	0.1321	0.4973	0.9117	0.0844	0.3664	0.9668	0.0444	0.2399	0.9899	0.0194	0.1349	0.9949	0.0131	0.0980
	SparseFormer [52]†	0.7716	0.1583	0.5610	0.8931	0.0966	0.3978	0.9632	0.0503	0.2545	0.9896	0.0230	0.1398	0.9946	0.0172	0.1052
	CostDCNet [20]	0.8249	0.1300	0.5119	0.9101	0.0859	0.3810	0.9629	0.0482	0.2582	0.9884	0.0218	0.1469	0.9939	0.0150	0.1079
	Ours (NLSPN Base)	0.8350	0.1272	0.4749	0.9215	0.0783	0.3473	**0.9690**	**0.0425**	**0.2335**	0.9903	0.0190	0.1325	0.9951	0.0130	0.0970
	Ours (NLSPN Effb5 Base)	**0.8934**	**0.0977**	**0.3994**	**0.9363**	**0.0687**	**0.3234**	0.9679	**0.0425**	0.2386	0.9881	0.0208	0.1461	0.9939	0.0140	0.1062

Table 6. Evaluation on the NYUv2 dataset. †Model re-implemented by us. We compare across both fixed and variable # of training points here because for some sparsities and models, variable training improves performance.

Method	$\delta_{1.02}\uparrow$	$\delta_{1.05}\uparrow$	$\delta_{1.10}\uparrow$	REL↓	RMSE↓
CSPN [7]	0.535	0.783	0.889	0.043	0.220
NLSPN [36]	0.603	0.808	0.909	0.036	0.179
SparseSPN [53]	0.648	0.844	0.929	0.031	0.159
SparseSPN [53]†	0.702	0.876	0.944	0.026	0.147
Ours (NLSPN Base)	**0.671**	**0.862**	**0.939**	**0.028**	**0.148**

Table 7. Methods are trained and evaluated on 800 SIFT keypoints on NYUv2. †Additional ViT-B used for normals.

Method	150 Points		500 Points		1500 Points	
	RMSE↓	MAE↓	RMSE↓	MAE↓	RMSE↓	MAE↓
NLSPN	217.0	104.5	146.9	59.2	79.2	26.3
Ours (NLSPN Base)	181.5	83.7	131.5	**52.0**	78.3	**25.5**
Ours (R34 RGB)	**163.9**	**79.8**	**122.7**	53.1	**72.6**	25.8

Table 8. Evaluation on VOID. Models are trained with 1500 points *without* point-dropping augmentation. Metrics are in mm.

500 points), and 3) using an RGB input encoder with our ASC module is best for such unevenly distributed points.

4.7. Inference on Unseen Depth Point Distributions

Here, we focus on point distributions unseen during training to evaluate wide applicability to other tasks and datasets.

	Method	# of SIFT Points			
		8 Points	16 Points	32 Points	64 Points
2-500	NLSPN	0.113	0.098	0.079	0.063
	Ours (NLSPN Base)	0.110	0.094	**0.075**	**0.060**
	Ours (R34 RGB)	**0.102**	**0.091**	0.078	0.064
8-64 SIFT	NLSPN	0.125	0.109	0.090	0.074
	Ours (NLSPN Base)	0.106	0.094	**0.079**	**0.066**
	Ours (R34 RGB)	**0.097**	**0.089**	**0.079**	0.068

Table 9. Evaluation on sparse SIFT keypoints. Metric is REL↓.

Transfer to SIFT Keypoints. We consider SIFT keypoints as a realistic, uneven target distribution of input depth points. We test models trained with variable random 2∼500 points and variable SIFT 8∼64 points in Table 9. The results confirm that our main evaluation setting of training on a variable # of randomly sampled points yields transferable models. Furthermore, we doubly verify that our ASC module again improves over NLSPN for uneven input and that RGB encoder variants are best for sparse and uneven input.

Comparison to Sparsity Agnostic Depth Completion. SpAgNet [10] proposes a sparsity-agnostic completion

	Method	8 Lines		16 Lines		32 Lines	
		RMSE↓	MAE↓	RMSE↓	MAE↓	RMSE↓	MAE↓
64	NLSPN	14099	9698	9542	5957	6371	3403
	Ours (NLSPN Base)	11197	6652	7829	4296	5704	2784
	Ours (R34 RGB)	**10877**	**5666**	**6612**	**3015**	**4855**	**2058**
1~64	NLSPN	7693	3982	5685	2880	4901	2393
	Ours (NLSPN Base)	7423	3849	5475	2595	**4774**	**2175**
	Ours (R34 RGB)	**6538**	**3176**	**5409**	**2419**	5072	2220

Table 10. Evaluation on nuScenes when training with 64-lines or random 1 to 64 lines on KITTI. Metrics are in mm.

Method	0 Points		2 Points		32 Points		500 Points	
	REL↓	RMSE↓	REL↓	RMSE↓	REL↓	RMSE↓	REL↓	RMSE↓
BTS R50 Mono-only [24]	0.119	0.419	-	-	-	-	-	
BTS R50 + Ours	0.117	0.411	0.095	0.365	0.043	0.215	0.017	0.108
AdaBins Mono-only [2]	0.103	0.364	-	-	-	-	-	
AdaBins + Ours	0.107	0.368	0.087	0.328	0.044	0.206	0.026	0.133

Table 11. Joint depth estimation & completion with our module.

model trained on 500 points and tested on various point patterns including Livox. The full comparison is in Section G of the supplementary. When similarly trained on 500 points, our pipeline with an RGB-input encoder outperforms SpAgNet in almost all settings. Evaluating methods trained with variable points, performance further improves, even for unique shifted grid and Livox patterns. This further demonstrates that simply training on a variable # of randomly sampled points transfers to unseen distributions.

Transfer Performance on nuScenes. To test wider applicability of completion methods, we benchmark KITTI-trained models on domain adaptation to nuScenes depth completion. We evaluate using the dense depth maps generated by [31]. The default nuScenes LiDAR is 32 lines, and we generate 16 and 8 lines by subsampling. The core results are in Table 10, and extensive analyses and visualizations are in Section I of the supplementary. We note that this is a very difficult setting with domain shift in capture conditions, RGB resolution, and sparse depth. RGBD models trained on just 64-line LiDAR transfer poorly and generates artefacts as noted in previous work [55]. However, using our pipeline with RGB input generates cleaner results. Training on variable lines improves performance, and in all settings, our pipeline outperforms NLSPN.

4.8. More Applications of the ASC module

Application to Monocular Depth Estimation Models. As our ASC module propagates depth information in the decoder stage, it can easily be applied to *any* RGB-only model to yield a *single model* for both depth estimation and completion. We verify this with monocular architectures: BTS [24] and AdaBins [2]. We train these models on the NYUv2 depth estimation split prepared by BTS for fair comparison. In Table 11, our pipeline maintains performance in the 0 point regime while consistently improving with more points. Interestingly, we find that AdaBins performs better for fewer points and BTS for more points, and we hypothesize AdaBins' use of a transformer to predict global depth distribution of the image may not be helpful when depth

Method	Backbone	Pre-train	-D input	2points	32 points	500 points
NLSPN	R34+	IN1k	✓	0.1321	0.0444	0.0131
Ours (NLSPN)	R34+	IN1k	✓	0.1272	0.0425	**0.0130**
Ours	R34	IN1k	✗	0.1100	0.0470	0.0151
Ours	BEiT-B	MiDaS	✗	0.0866	0.0416	0.0154
Ours	BEiT-L	MiDaS	✗	**0.0744**	**0.0385**	0.0158

Table 12. NYUv2 completion w/ variable # of points. REL↓ used.

BTS [24] + Our ASC w/ various RGB backbones	0 Points		2 Points		32 Points		500 Points	
	REL↓	RMSE↓	REL↓	RMSE↓	REL↓	RMSE↓	REL↓	RMSE↓
ResNet50 (ImageNet1k)	0.117	0.411	0.095	0.365	0.043	0.215	0.017	**0.108**
BEiT-B (MiDaS pretrained)	0.106	0.369	0.081	0.317	0.037	0.201	0.016	0.115
BEiT-L (MiDaS pretrained)	**0.088**	**0.317**	**0.063**	**0.264**	**0.030**	**0.170**	**0.016**	0.115

Table 13. Joint depth estimation & completion with standard mono-depth settings (416x544 resolution). RMSE is in meters.

distribution is known with many input depths. Our module allows any depth estimation method to also perform completion, connecting two largely separated lines of work.

Scaling Monocular Depth Estimation Models. As our ASC module can easily be applied to any mono-depth architecture, we demonstrate that it is complementary to advancements in mono-depth results. We show completion-only variants in Table 12 and joint estimation & completion in Table 13. Using a mono-depth MiDaS [40] backbone improves results for 0-32 points. However, there are trade-offs at very dense (500-point) settings, where a simple R34+ backbone with RGB**<u>D</u>** input performs better. A more extensive discussion is in Section J of the supplementary.

Single Pipeline for 3D Detection on Variable Scan Lines. Other than domain adaptation, another advantage of a single completion model for variable scan lines is ease of deployment to different sensors and settings. To test our model for downstream tasks, we train 3D detection on point clouds out-projected from completed depth maps. The full results are in Section K of the supplementary. First, our completion model consistently improves 3D detection [11], especially in few-line settings where the LiDAR-only model collapses. Our completion-then-detection pipeline also outperforms LiDAR-only for both "Fixed" 64-line and "Variable" 1-64 line settings. In all, we confirm that our pipeline is effective for downstream 3D detection and can be used for a completion-then-detection pipeline on variable sparsities.

5. Conclusion

In this work, we proposed a novel depth completion framework that leverages predicted affinity between image locations and depth points to enable each depth point to adaptively influence and improve depth predictions across the image. Our method improves performance for both sparser and variable distribution regimes on the KITTI and NYUv2 datasets, as well as for when transferred to the nuScenes dataset and to SIFT keypoints. Further, our proposed shift-correction module can be applied to any monocular depth estimation model. We believe our work bridges the gap between monocular estimation and depth completion and will enable the development of depth completion methods applicable to various sensors and applications.

References

[1] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Multi-view depth estimation by fusing single-view depth probability with multi-view geometry. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[2] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021. 2, 3, 8

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 2

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing. 3

[5] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning joint 2d-3d representations for depth completion. In *ICCV*, pages 10023–10032, 2019. 2, 5

[6] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021. 3

[7] Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–119, 2018. 2, 6, 7

[8] Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10615–10622, 2020. 2

[9] Nathaniel Chodosh, Chaoyang Wang, and Simon Lucey. Deep convolutional compressed sensing for lidar depth completion. In *ACCV*, pages 499–513, 2018. 2

[10] Andrea Conti, Matteo Poggi, and Stefano Mattoccia. Sparsity agnostic depth completion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5871–5880, 2023. 2, 7

[11] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. *arXiv:2012.15712*, 2020. 8

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3

[13] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014. 2, 5

[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 1, 2

[15] Vitor Guizilini, Rares Ambrus, Wolfram Burgard, and Adrien Gaidon. Sparse auxiliary networks for unified monocular depth prediction and completion. In *CVPR*, pages 11078–11088, 2021. 2

[16] Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. Penet: Towards precise and efficient image guided depth completion. *arXiv preprint arXiv:2103.00783*, 2021. 1, 2, 3, 5, 7

[17] Lam Huynh, Phong Nguyen, Jiří Matas, Esa Rahtu, and Janne Heikkilä. Boosting monocular depth estimation with lightweight 3d point fusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12767–12776, 2021. 2, 7

[18] Saif Imran, Xiaoming Liu, and Daniel Morris. Depth completion with twin surface extrapolation at occlusion boundaries. In *CVPR*, pages 2583–2592, 2021. 2, 5

[19] Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. Sparse and dense data with cnns: Depth completion and semantic segmentation. In *2018 International Conference on 3D Vision (3DV)*, pages 52–60. IEEE, 2018. 2

[20] Jaewon Kam, Jungeon Kim, Soongjin Kim, Jaesik Park, and Seungyong Lee. Costdcnet: Cost volume based depth completion for a single rgb-d image. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pages 257–274. Springer, 2022. 2, 5, 6, 7

[21] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *Advances in neural information processing systems*, 30, 2017. 2

[22] Jason Ku, Ali Harakeh, and Steven L Waslander. In defense of classical image processing: Fast depth completion on the cpu. In *CRV*, pages 16–22, 2018. 2

[23] Byeong-Uk Lee, Kyunghyun Lee, and In So Kweon. Depth completion using plane-residual representation. In *CVPR*, pages 13916–13925, 2021. 2

[24] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019. 2, 3, 8

[25] Ang Li, Zejian Yuan, Yonggen Ling, Wanchao Chi, Chong Zhang, et al. A multi-scale guided cascade hourglass network for depth completion. In *WACV*, pages 32–40, 2020. 1, 2, 3, 6, 7

[26] Yuankai Lin, Tao Cheng, Qi Zhong, Wending Zhou, and Hua Yang. Dynamic spatial propagation network for depth completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1638–1646, 2022. 2

[27] Lina Liu, Xibin Song, Xiaoyang Lyu, Junwei Diao, Mengmeng Wang, Yong Liu, and Liangjun Zhang. Fcfr-net: Feature fusion based coarse-to-fine residual learning for depth completion. In *AAAI*, pages 2136–2144, 2021. 2

[28] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. Learning affinity via spatial propagation networks. *arXiv preprint arXiv:1710.01020*, 2017. 2

[29] Xin Liu, Xiaofei Shao, Bo Wang, Yali Li, and Shengjin Wang. Graphcspn: Geometry-aware depth completion via dynamic gcns. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, pages 90–107. Springer, 2022. 2

[30] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. *ArXiv*, abs/2104.00678, 2021. 3

[31] Yunfei Long, Daniel Morris, Xiaoming Liu, Marcos Castro, Punarjay Chakravarty, and Praveen Narayanan. Radar-camera pixel depth association for depth completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12507–12516, 2021. 8

[32] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004. 2, 6

[33] Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *ICRA*, pages 4796–4803. IEEE, 2018. 2, 3

[34] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *ICRA*, 2019. 5, 7

[35] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2

[36] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. Non-local spatial propagation network for depth completion. In *ECCV*, 2020. 1, 2, 3, 5, 6, 7

[37] Vaishakh Patil, Christos Sakaridis, Alexander Liniger, and Luc Van Gool. P3depth: Monocular depth estimation with a piecewise planarity prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1610–1621, 2022. 2

[38] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3313–3322, 2019. 2

[39] Chao Qu, Ty Nguyen, and Camillo Taylor. Depth completion via deep basis fitting. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 71–80, 2020. 2

[40] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 2, 5, 8

[41] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12892–12901, 2022. 1, 2

[42] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 1, 2, 6

[43] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision – ECCV 2012*, pages 746–760, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. 2

[44] N. Silberman, Derek Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1

[45] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021. 3

[46] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6105–6114. PMLR, 2019. 5

[47] Jie Tang, Fei-Peng Tian, Wei Feng, Jian Li, and Ping Tan. Learning guided convolutional network for depth completion. *IEEE Transactions on Image Processing*, 30:1116–1129, 2020. 1, 2, 5, 7

[48] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *3DV*, pages 11–20, 2017. 2

[49] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 international conference on 3D Vision (3DV)*, pages 11–20. IEEE, 2017. 1

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 3

[51] Yue Wang, Vitor Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. *arXiv preprint arXiv:2110.06922*, 2020. 3

[52] Frederik Warburg, Michael Ramamonjisoa, and Manuel López-Antequera. Sparseformer: Attention-based depth completion network. *arXiv preprint arXiv:2206.04557*, 2022. 2, 4, 6, 7

[53] Yuqun Wu, Jae Yong Lee, and Derek Hoiem. Sparse spn: Depth completion from sparse keypoints. *arXiv preprint arXiv:2212.00987*, 2022. 2, 7

[54] Ke Xian, Jianming Zhang, Oliver Wang, Long Mai, Zhe Lin, and Zhiguo Cao. Structure-guided ranking loss for single image depth prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 611–620, 2020. 2

[55] Ziyang Xie, Junge Zhang, Wenye Li, Feihu Zhang, and Li Zhang. S-nerf: Neural radiance fields for street views. *arXiv preprint arXiv:2303.00749*, 2023. 2, 8

[56] Xin Xiong, Haipeng Xiong, Ke Xian, Chen Zhao, Zhiguo Cao, and Xin Li. Sparse-to-dense depth completion revis-

ited: Sampling strategy and graph construction. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[57] Guangkai Xu, Wei Yin, Hao Chen, Chunhua Shen, Kai-Sheng Cheng, Fengyu Wu, and Fengshang Zhao. Towards 3d scene reconstruction from locally scale-aligned monocular video depth. 2022. 2

[58] Yan Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, Hujun Bao, and Hongsheng Li. Depth completion from sparse lidar data with depth-normal constraints. In *ICCV*, pages 2811–2820, 2019. 2

[59] Zhiqiang Yan, Kun Wang, Xiang Li, Zhenyu Zhang, Jun Li, and Jian Yang. Rignet: Repetitive image guided network for depth completion. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVII*, pages 214–230. Springer, 2022. 2, 5

[60] Jiayu Yang, Wei Mao, Jose M Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4877–4886, 2020. 2

[61] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018. 2

[62] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 204–213, 2021. 2

[63] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Simon Chen, and Chunhua Shen. Towards domain-agnostic depth completion. *arXiv preprint arXiv:2207.14466*, 2022. 2

[64] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. NeW CRFs: Neural Window Fully-connected CRFs for Monocular Depth Estimation. *arXiv e-prints*, art. arXiv:2203.01502, 2022. 2

[65] Shanshan Zhao, Mingming Gong, Huan Fu, and Dacheng Tao. Adaptive context-aware multi-modal network for depth completion. *IEEE Transactions on Image Processing*, 2021. 2, 3