2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

JetsonCompletion: Real-Time Depth Completion on Resource-Constrained Edge Devices

Kailin Wang[1,2], Xiaozhou Zhu[2,3], Benyi Yang[2,3], Tian Zhang[2,3], Haoxin Zhang[2,4], Fei Xie[†,1], Shuaixin Li[*,2,3]

***Abstract*— Depth completion from sparse LiDAR points and images is a key perception task for autonomous robots, enabling dense 3D understanding in challenging environments. However, most recent researches achieve accuracy gains by greatly enlarging network size, making them unsuitable for real-time deployment on power- and compute-constrained platforms. This paper proposes an ultra-lightweight depth completion framework optimized for embedded systems. Our approach integrates a re-parameterized encoder–decoder with fewer than 5 M parameters and a two-stage hybrid distillation strategy. The first stage progressively densifies sparse depth supervision, while the second preserves edge fidelity through a combination of metric and structural losses. A full TensorRT FP16 pipeline further ensures efficient deployment. Extensive experiments on KITTI Depth Completion, NYU-v2 . demonstrate that our method achieves competitive accuracy while maintaining high efficiency. On a Jetson Xavier NX, the system runs at over 30 FPS with sub-33 ms latency within a 20 W power envelope, showing strong potential for real-world micro-robotic platforms. We will open-source the code to benefit the community. Our open source website: https://github.com/2463450186Q/JetsonCompletion.git**

I. Introduction

Depth completion is essential for autonomous navigation and 3-D reconstruction. However, measurements from the typical used LiDAR are inherently sparse: beyond $50\,\mathrm{m}$ or on low-albedo surfaces, valid returns often drop to only 5–30% of points. To recover dense maps, most existing researches fuse sparse depth with RGB images. Yet the prevailing paradigm has pursued ever-higher accuracy (e.g., MAE, RMSE) by continually enlarging model size and branching complexity. As a result, state-of-the-art networks now even exceed $100M$ weights and require power-hungry desktop GPUs, incurring latencies $\geq 150ms$—far above the $\leq 20W$ TDP (Thermal Design Power) and real-time demand of micro-UAVs and mobile robots.

Recent works [7], [8], [10] have primarily concentrated on architectural refinements rather than model compression or knowledge distillation, leaving resulting networks computationally intensive and ill-suited for embedded deployment. As illustrated in Fig.2, another bottleneck arises from dataset supervision: the widely-used KITTI-DC benchmark provides only semi-dense ground truth (around 30% labelled pixels).

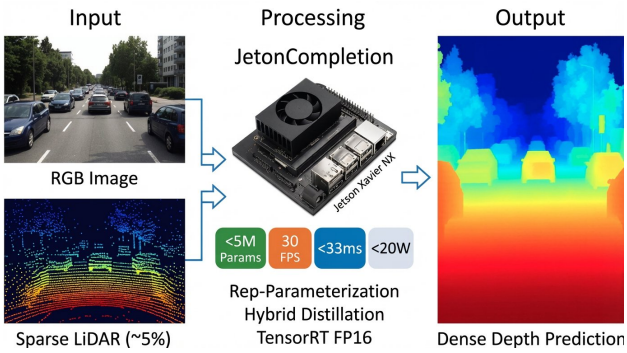


Fig. 1: Our Method Enables Real-Time Inference on Jetson Embedded Platforms

Consequently, the loss function supplies insufficient signal in distant, occluded or low-reflectance regions, often yielding over-smoothed boundaries. While methods, e.g., [12], exploit powerful monocular priors to inject structured geometric knowledge, and methods, e.g., [1], distill structural cues from monocular depth estimators into completion networks, these strategies are still evaluated on desktop GPUs whose peak power consumption still far exceeds the thermal and energy budget of embedded platforms.

Moreover, the mobile- and robotics-oriented literature remains surprisingly sparse:no prior work to date has simultaneously achieved all three requirements— decimeter-level accuracy, $\geq 30\,\mathrm{FPS}$ throughput, and sustained $\leq 20\,\mathrm{W}$ operation—on an off-the-shelf embedded computer.

To address these challenges, we propose **JetsonCompletion**, an ultra-lightweight depth completion framework that elevates parameter count, power and latency as co-equal objectives alongside accuracy. By combining a re-parameterised slim encoder–decoder with a progressive dual-teacher distillation scheme, our method delivers real-time dense depth estimation on the Jetson Xavier NX (21 TOPS INT8) within a strict $20\,\mathrm{W}$ envelope. Our main contributions are as follows:

1) **Ultra-lightweight real-time depth completion under edge constraints.** We design a re-parameterized encoder–decoder with fewer than 5M parameters, capable of inferring 1216×352 dense depth at $\geq 30\,\mathrm{FPS}$ on the Jetson Xavier NX while remaining within a $\leq 20\,\mathrm{W}$ power budget, providing the first sustained real-time solution for micro-UAVs and nano-robots.

[1] Nanjing Normal University, Nanjing, China.
[2] Defense Innovation Institute, Chinese Academy of Military Science, Beijing, China.
[3] Intelligent Game and Decision Laboratory, Beijing, China.
[4] Sun Yat-sen University, Guangzhou, China.
[*] Corresponding author: Shuaixin Li (lsx_navigation@sina.com).
[†] Co-corresponding author: Fei Xie (xiefei@njnu.edu.cn).

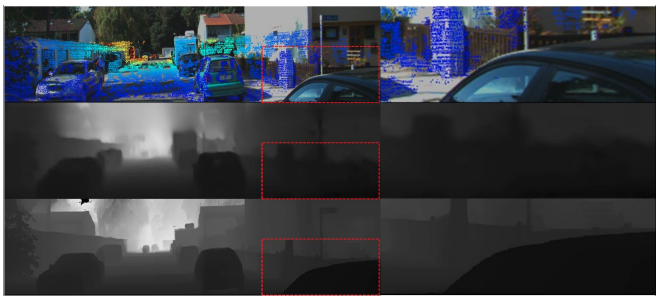


Fig. 2: Limitations in depth-completion supervision (top-to-bottom): (a) Sparse official GT projection; (b) depth-completion network produces blurred car boundaries (red dashed boxes) because hardly no LiDAR rays hit the vehicle; (c) the same scene estimated by a monocular branch, which recovers sharper object contours thanks to learned photometric priors. In (a) and (b) the dashed boxes highlight areas where no valid LiDAR returns exist; the absence of ground-truth depth supervision in these regions leads to failure in predicting the car depth under standard depth-completion pipelines.

2) **Hybrid two-stage dual-teacher distillation.** A state-of-the-art depth completion teacher densifies supervision from 30% to full pixel coverage, while a monocular depth-estimation teacher injects Structural Similarity Index Measure (SSIM) and edge-aware losses. Joint training reduces root mean square error (RMSE) by $110.0\,\text{mm}$ compared to sparse-only supervision while preserving sub-pixel boundary precision.
3) **End-to-end deployment and validation** Our PyTorch$\rightarrow$TensorRT FP16 pipeline runs continuously for over one hour without frame drops, achieving $< 33\,\text{ms}$ latency and sub-metre accuracy on KITTI-DC within the $20\,\text{W}$ TDP envelope.

II. Related Works

A. Depth Completion

Depth completion from sparse LiDAR and RGB has been extensively studied over the past few years. Early CNN-based methods such as NLSPN [4] and PENet [6], [9], [21] refined depth via non-local propagation or dual-branch fusion, but remained heavily dependent on RGB cues, making them vulnerable to low-texture regions and illumination changes. Later designs introduced semantic or attention modules [21], improving feature interaction at the cost of higher inference latency and reliance on external models.

To capture long-range context, recent works incorporate Transformers, as in GuideFormer [8] and CompletionFormer [7]. While these achieve state-of-the-art accuracy, the quadratic complexity of self-attention leads to large model sizes and poor real-time performance, e.g., CompletionFormer runs below $10\,\text{FPS}$ even on high-end GPUs, and becomes impractical for embedded deployment under a $20\,\text{W}$ power budget.

Another challenge stems from supervision. The open-source datasets, e.g., KITTI Depth Completion benchmark, only provide semi-dense labels ($\sim$30% pixel coverage), with severe sparsity beyond $80\,\text{m}$. This weak signal causes blurred boundaries and unreliable depth in distant or low-reflectance regions. To enrich supervision, recent studies distill priors from monocular depth estimators [1], [11], [12], or design architectures robust to diverse sparse patterns, e.g., OMNI-DC [2]. While effective in improving generalization, these models still require large backbones and remain unverified on edge hardware.

In summary, although depth completion has advanced rapidly, existing methods either emphasize accuracy with heavy architectures or improve supervision using monocular priors, yet none has demonstrated the ability to achieve *all three* requirements simultaneously: decimeter-level accuracy, $\geq 30\,\text{FPS}$ throughput, and $\leq 20\,\text{W}$ power consumption on commodity embedded devices. This gap motivates our work.

B. Lightweight Visual Backbones

The performance of depth completion is closely tied to backbone design. Classic CNNs such as ResNet enabled very deep networks via residual mapping, but quickly grow to hundreds of layers and $> 100\,\text{M}$ parameters. Vision Transformers [5], [13], [15], [22] offer strong global modeling, yet their $\mathcal{O}(N^2)$ complexity and memory cost make them difficult to deploy on embedded devices.

To reduce compute consumption, lightweight CNNs (e.g., MobileNet [16], ShuffleNet [17], GhostNet [18]) exploit depth-wise separable convolutions or reparameterization, while efficient Transformers (e.g., DeiT-Ti, MobileViT [23]) employ distillation or factorized attention. Hybrid designs such as ConvNeXt and CoAtNet combine convolutional locality with window-based attention. Despite these advances, most lightweight backbones are developed for classification or detection, and few have been systematically tailored to the specific challenges of LiDAR-guided depth completion.

To the best of our knowledge, no prior work simultaneously achieves high accuracy, real-time throughput, and low-power operation for depth completion on embedded platforms. **JetsonCompletion** directly addresses this gap by coupling a reparameterized lightweight backbone with a dual-teacher distillation strategy, enabling dense depth prediction under strict edge constraints.

III. Methodology

We design a depth completion backbone specifically for *embedded deployment*, where three aforementioned constraints must be satisfied simultaneously: strict computational budget, ultra-low inference latency, and accuracy *vs.* state-of-the-art multi-branch architectures.

we replace the conventional encoder–decoder architecture augmented with a refinement head with a single-stack, reparameterizable encoder–decoder network that requires no additional refinement stage.

During training, the model employs *multiple parallel branches* to strengthen gradient propagation and improve

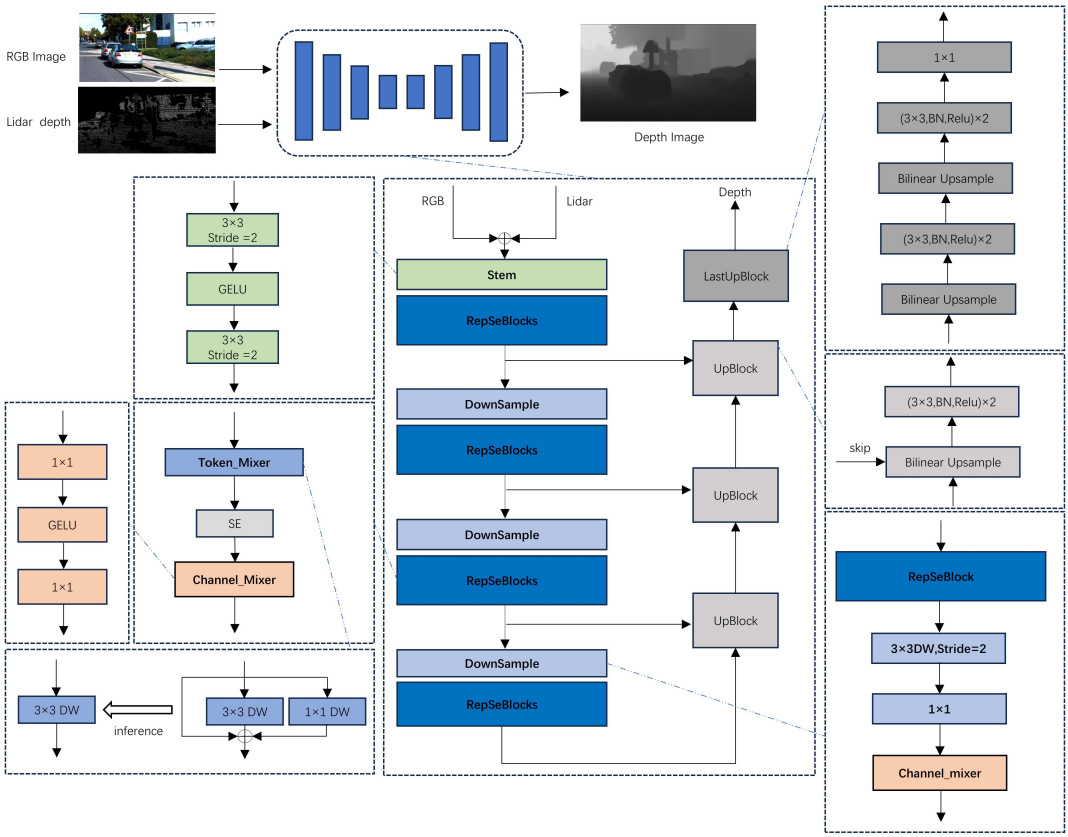


Fig. 3: Overall architecture of the proposed depth-completion network, including its encoder decoder structure and three branch design. In the inference stage of the network, the three branch structure is integrated into a single output through a 3x3 convolutional layer to generate the final depth prediction map.

representational capacity. After training, all branches are *algebraically merged* into a *pure single-branch ConvNet* through structural reparameterization. This strategy provides high expressivity during training while ensuring maximum efficiency at inference, resulting in an identical lightweight architecture with no additional overhead.

As illustrated in Fig.3, the entire architecture is constructed solely from a unified lightweight operator family—RepSeBlock—which is trained once and fused before deployment. This design enables real-time performance on embedded devices under stringent speed and memory constraints, without compromising accuracy.

A. Reparameterizable Encoder

1) Macro architecture: Following RepViT [19], [20], we adopt a *four-stage hierarchical encoder* tailored for embedded deployment. The stem downsamples the input by $4\times$ through two successive 3×3 convolutions. Each subsequent stage performs a stride-2 downsampling and then stacks N_k RepSeBlocks, producing a feature pyramid $\{\mathbf{F}_k\}_{k=1}^{4}$ with spatial strides $\{4, 8, 16, 32\}$. Downsampling is realised without additional operators: the first RepSeBlock in each stage conducting $2\times$ resolution reduction by simply setting its depth-wise convolution stride to 2, while the residual path is aligned by a 1×1 convolution. This yields an efficient *parameter-free downsampling strategy*.

2) RepSeBlock: a dual-purpose unit: RepSeBlock is the *only* building block used throughout the encoder. During training it behaves as a multi-branch token–channel mixer, while at inference it collapses into a single 3×3 convolution with an affine transformation (i.e., learned scale and bias).

a) Token-mixing: employs a parallel formulation:

$$\mathbf{Y}_{\text{token}} = \text{DW}_{3\times3}(\mathbf{X}) + \text{DW}_{1\times1}(\mathbf{X}) + \mathbf{X}, \tag{1}$$

followed by an squeeze-and-excitation (SE) module (ratio 4) for channel recalibration.

b) Channel-mixing: is a conventional 1×1 expand–contract MLP with GELU and stochastic depth captures cross-channel dependencies. After training, all BatchNorm layers are folded into their preceding convolutions, so the block is algebraically equivalent to a single convolution with affine bias. (see Sec. III-A.4). Thus, RepSeBlock provides training-time expressivity but inference-time efficiency identical to a plain CNN.

3) Parameter-Free Downsampling Strategy : When spatial down-sampling is required, only the stride of the 3 × 3 DWConv in the Token-Mixer of the first RepSeBlock within the corresponding stage is set to 2; the remaining branches (1 × 1 DWConv and the residual shortcut) retain stride = 1, achieving channel- and spatial-alignment through the 1 × 1 DWConv and the residual connection. This design obviates

the need for additional down-sampling operators or bypasses, realizing a "zero-parameter" 2× down-sampling.

4) Structural Reparameterization: During training, the Token-Mixer aggregates three parallel branches:

$$\mathbf{Y}_{\text{train}} = \text{BN}(\mathbf{W}_3 * \mathbf{X}) + \text{BN}(\mathbf{W}_1 * \mathbf{X}) + \text{BN}(\mathbf{X}), \quad (2)$$

where $*$ denotes depth-wise convolution, $\mathbf{W}_3 \in \mathbb{R}^{C\times1\times3\times3}$ represents the 3×3 depth-wise kernel, $\mathbf{W}_1 \in \mathbb{R}^{C\times1\times1\times1}$ denotes the 1×1 depth-wise kernel, and $\mathbf{X} \in \mathbb{R}^{B\times C\times H\times W}$ is the input tensor. γ_i, β_i, μ_i, and σ_i are the learned scale, bias, running mean, and running standard deviation of the Batch-Norm layer following branch i, respectively. Batch-Norm can be linearly merged into its preceding convolution:

$$\text{BN}(\mathbf{Y}) = \frac{\gamma}{\sigma}(\mathbf{Y} - \mu) + \beta \triangleq \mathbf{W}' * \mathbf{X} + \mathbf{b}'. \quad (3)$$

Folding all branches yields a single equivalent kernel and bias:

$$\mathbf{W}_{\text{fuse}} = \frac{\gamma_3}{\sigma_3}\mathbf{W}_3 + \text{pad}_{3\times3}\Big(\frac{\gamma_1}{\sigma_1}\mathbf{W}_1\Big) + \frac{\gamma_0}{\sigma_0}\mathbf{I}, \quad (4)$$

$$\mathbf{b}_{\text{fuse}} = \sum_i \Big(\beta_i - \frac{\gamma_i\mu_i}{\sigma_i}\Big), \quad (5)$$

where $\mathbf{I}$ is the 3×3 identity kernel and $\text{pad}_{3\times3}(\cdot)$ zero-pads 1×1 kernels to 3×3. During inference, the output is given by

$$\mathbf{Y}_{\text{infer}} = \mathbf{W}_{\text{fuse}} * \mathbf{X} + \mathbf{b}_{\text{fuse}}, \quad (6)$$

reducing computation with zero accuracy drop.

5) Network Architecture: Decoder Design: To recover full resolution, we design a lightweight decoder chain (Fig. 3). Each UpBlock performs: (i) bilinear $2\times$ up-sampling, (ii) channel-wise concatenation with the encoder skip, (iii) two 3×3 Conv-BN-ReLU layers. Formally,

$$\mathbf{F}_{\text{up}} = \text{Conv}_{3\times3}\Big(\text{Concat}\big[\text{Up}_{2\times}(\mathbf{F}_{\text{low}}),\ \mathbf{F}_{\text{skip}}\big]\Big). \quad (7)$$

All convolutions use *depth-wise separable* kernels to keep computation low. A final $4\times$ up-sampling (bilinear or learnable deconv) restores the original resolution before the regression head.

B. Progressive Mixed-Distillation Training

We aim to train an ultra-lightweight network $f_{\boldsymbol{\theta}}$ that regresses dense depth $\hat{\mathbf{D}} \in \mathbb{R}^{H\times W}$ from an RGB image $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$ and a sparse LiDAR map $\mathbf{D}^{\text{sparse}} \in \mathbb{R}^{H\times W}$, while only 30% of the pixels $\mathcal{V} \subset \Omega$ possess ground-truth labels $\mathbf{D}^{\text{gt}}$. Directly minimising the $\mathcal{L}_1$ error on $\mathcal{V}$ leads to *supervision starvation* outside the sparse set, which in turn yields excessive filter redundancy and blurred object boundaries .

To inject pixel-dense supervisory signals without introducing any extra annotation cost, we propose a **Progressive Mixed Distillation** framework that distils knowledge from two complementary teachers into the student $f_{\boldsymbol{\theta}}$ (sub 5 M parameters). As illustrated in Fig. 4, the optimisation proceeds in two stages: (1) metric distillation that transfers absolute depth values from a SOTA completion network [1]; (2) joint metric-structure distillation that further incorporates monocular geometric cues [3].

1) Teacher Generation: Let $\mathbf{D}^{(1)}$ denote the prediction of an off-the-shelf *depth-completion* network (SOTA-C) and $\mathbf{D}^{(2)}$ the output of a *monocular depth-estimation* network (SOTA-M). $\mathbf{D}^{(1)}$ is metrically accurate yet occasionally over-smoothed, whereas $\mathbf{D}^{(2)}$ provides complete shape cues up to an unknown scale. Both teachers are frozen during the whole training stage.

2) Stage-1: Metric Distillation: In the first stage $t \in [0, T_1)$, we treat $\mathbf{D}^{(1)}$ as pseudo ground-truth and enforce dense regression by

$$\mathcal{L}_{\text{Metric}} = \frac{1}{|\Omega|}\sum_{p\in\Omega}\Big[\big|\hat{D}_p - D_p^{(1)}\big| + \big|\hat{D}_p - D_p^{(1)}\big|^2\Big], \quad (8)$$

where $\hat{\mathbf{D}} = f_{\boldsymbol{\theta}}(\mathbf{I}, \mathbf{D}^{\text{sparse}})$. Compared with the vanilla sparse loss, Eq. (8) raises the effective sample density from 30% to 100%, yielding a $\mathbf{3.3}\times$ increase in supervised gradients and significantly accelerating convergence.

3) Stage-2: Joint Metric-Structure Distillation: Starting from $t = T_1$, we continue to exploit $\mathbf{D}^{(1)}$ for absolute depth while simultaneously transferring geometric details from $\mathbf{D}^{(2)}$. The overall objective is a weighted combination of three terms:

$$\mathcal{L}_2 = \underbrace{\mathcal{L}_{\text{Metric-C}}}_{\text{metric}} + \alpha\,\underbrace{\mathcal{L}_{\text{Sparse}}}_{\text{scale}} + \lambda_{\text{Struct}}\,\underbrace{\mathcal{L}_{\text{Struct}}}_{\text{geometry}}. \quad (9)$$

Metric supervision $\mathcal{L}_{\text{Metric-C}}$. Identical to Eq. (8), we keep the dense $\mathcal{L}_1 + \mathcal{L}_2$ loss against $\mathbf{D}^{(1)}$ to maintain metric accuracy.

Sparse ground-truth supervision $\mathcal{L}_{\text{Sparse}}$. To recover the global scale and offset, we additionally minimise

$$\mathcal{L}_{\text{Sparse}} = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\Big[\big|\hat{D}_v - D_v^{\text{gt}}\big| + \big|\hat{D}_v - D_v^{\text{gt}}\big|^2\Big]. \quad (10)$$

Structure–edge consistency $\mathcal{L}_{\text{Struct}}$. Inspired by monocular estimation literature, we adopt SSIM and gradient alignment to transfer fine boundary cues from $\mathbf{D}^{(2)}$:

$$\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(\hat{\mathbf{D}}, \mathbf{D}^{(2)}), \quad (11)$$

$$\mathcal{L}_{\text{edge}} = \|\nabla_x\hat{\mathbf{D}} - \nabla_x\mathbf{D}^{(2)}\|_1 + \|\nabla_y\hat{\mathbf{D}} - \nabla_y\mathbf{D}^{(2)}\|_1, \quad (12)$$

$$\mathcal{L}_{\text{Struct}} = \lambda_{\text{SSIM}}\mathcal{L}_{\text{SSIM}} + \lambda_{\text{edge}}\mathcal{L}_{\text{edge}}. \quad (13)$$

The relative weights α, λ_{SSIM} and λ_{edge} are *dynamically* adjusted according to the validation MAE and RMSE.

4) Overall Training Pipeline: Putting everything together, the optimisation objective at training step t is given by

$$\mathcal{L}_t = \begin{cases} \mathcal{L}_{\text{Metric}}, & 0 \le t < T_1, \\ \mathcal{L}_{\text{Metric-C}} + \alpha\mathcal{L}_{\text{Sparse}} + \lambda_{\text{Struct}}\mathcal{L}_{\text{Struct}}, & T_1 \le t < T_2. \end{cases} \quad (14)$$

We employ the Adam optimiser with an initial learning rate of 1×10^{-3} and cosine annealing. The overall objective is summarised in Eq. (14). Batch size is set to 15 on a single RTX-4090D GPU.

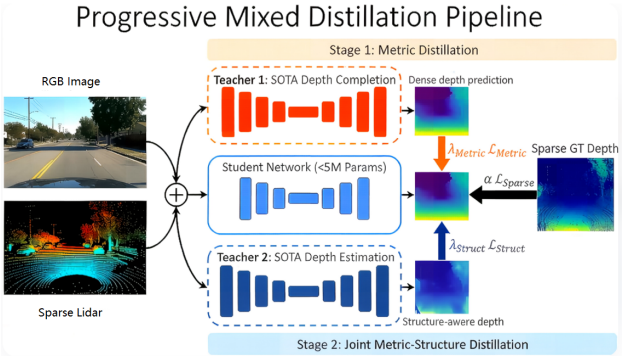


Fig. 4: Overview of the progressive mixed distillation pipeline. The student is first supervised by the completion teacher only (Stage-1), and then jointly optimised with both teachers (Stage-2).

IV. Experiments

A. Implementation Details

Our method is implemented in PyTorch. Compared with prior work that typically relies on multi-GPU setups, we conduct the entire distillation and training phase on a single NVIDIA RTX 4090D GPU, and subsequently fine-tune on the real dataset, achieving competitive results. See the supplementary video for the end-to-end demonstration on Jetson Xavier NX at 30 FPS within the 20 W power envelope.

The embedded test platform is summarised in Table I.

TABLE I: Jetson Xavier NX embedded platform specifications.

Module	**Specification**
SoC	NVIDIA Jetson Xavier NX
CPU	6-core Carmel ARMv8.2 @ 1.9 GHz
GPU	384-core Volta + 48 Tensor Cores
Peak Compute	21 TOPS (INT8) / 6 TFLOPS (FP16)
Memory	16 GB 128-bit LPDDR4x @ 59.7 GB/s
Power Envelope	20 W nominal

B. Dataset

We evaluate the proposed method on two widely-used benchmarks: outdoor KITTI Depth Completion (DC) and indoor NYU-v2. KITTI provides 86 898 training frames, 1 000 for validation and 1 000 for testing; the sparse depth (5.9% pixels) is captured by Velodyne HDL-64e, while dense GT is obtained by fusing 11 consecutive LiDAR sweeps. NYU-v2 contains RGB-D pairs from 464 indoor scenes; following [6], we train on 50 000 randomly-sampled frames and test on the official 654 images after half-down-sampling to 320×240 and center-cropping to 304×228. Dataset-specific loss weights are searched individually on the corresponding validation split; all experiments are conducted on the same single-RTX-4090D GPU. Quantitative results are summarised in Table II and III .

TABLE II: Comparison of Different Methods on KITTI-DC.

Method	MAE (mm)	iMAE (1/km)	iRMSE (1/km)	RMSE (mm)	REL	Runtime (ms) PC	Runtime (ms) Jetson	Params (M)
CSPN [25]	279.5	1.15	2.93	1019.6	0.117	1000	>100	–
DeepLiDAR [26]	226.5	1.15	2.56	758.4	0.115	70	>100	144.06
GuideNet [27]	218.8	0.99	2.25	736.2	0.101	140	>100	–
NLSPN [4]	199.6	0.84	1.99	741.7	0.092	220	–	26.25
PENet [6]	210.6	0.94	2.17	730.1	–	65	>100	132.1
ENet [6]	216.26	0.95	2.14	741.3	–	19	87	131.7
ACMNet [28]	206.1	0.90	2.08	744.9	0.105	44	–	–
TWISE [29]	195.6	**0.82**	2.08	840.2	0.097	20	–	–
GuideFormer [8]	207.8	0.97	2.14	721.5	–	–	–	–
DySPN [30]	**192.7**	**0.82**	**1.88**	**709.1**	0.090	160	–	–
Ours	212.6	0.94	1.98	742.1	**0.08**	**3**	**31**	**4.11**

TABLE III: Quantitative results on the NYU-v2 test set, evaluated on a single RTX 4090D GPU. ↓ denotes that lower values are better.

Method	RMSE↓(m)	REL↓
CSPN [25]	0.117	0.016
GuideNet [27]	0.101	0.015
NLSPN [4]	0.092	**0.012**
ACMNet [28]	0.105	0.015
DySPN [30]	**0.089**	**0.012**
Ours	0.106	**0.012**

C. Evaluation

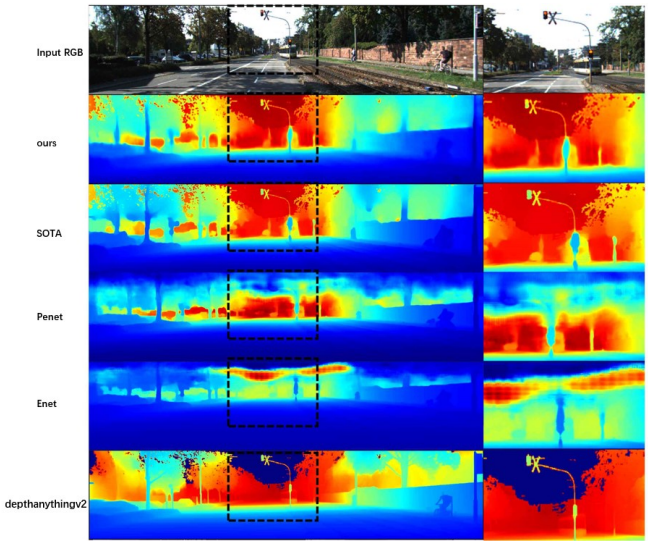
1) Quantitative Results : As shown in Table II, the proposed method demonstrates strong overall performance on the KITTI-DC benchmark. It consistently runs ≥30 FPS on both desktop GPUs and the Jetson Xavier NX, outperforming most prior work in deployment efficiency. This speed advantage is essential for real-time robotics applications, especially on edge devices where many competing methods fail owing to library incompatibility or excessive resource demands.

Among the entries that can still be evaluated, Our sub 5 M parameter model achieves accuracy competitive with architectures possessing 30–130 M parameters, maintaining top-tier iRMSE, RMSE and REL performance. These results indicate that the distilled lightweight architecture retains the predictive capacity of much heavier teachers without sacrificing edge-deployability. **Note:** Runtime and parameter entries for several baselines are marked "–" because the original open-source repositories are no longer accessible; thus the corresponding figures could not be re-measured.

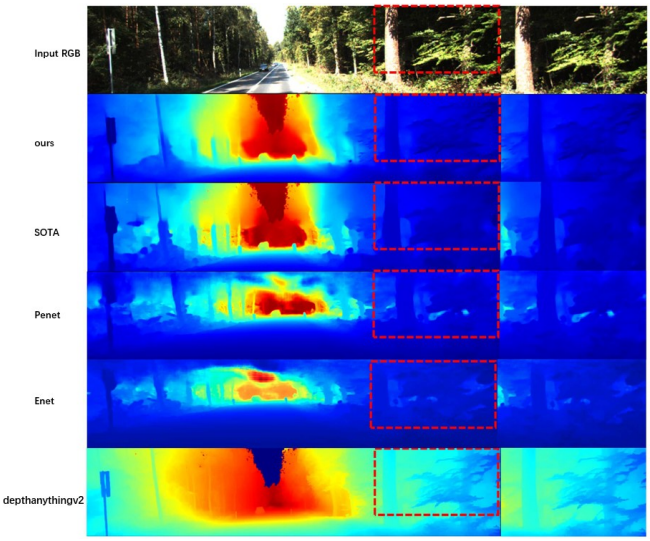
TABLE IV: Cross-platform comparison on a randomly sampled subset of the KITTI validation set.

Method	Platform	MAE↓(mm)	RMSE↓(mm)	FPS↑
PENet	RTX-4090D	213.4	730.1	15.4
PENet	Jetson NX	213.4	730.1	1.8
ENet	RTX-4090D	245.7	785.3	55.6
ENet	Jetson NX	245.7	785.3	8.1
Ours	**RTX-4090D**	219.8	732.1	**275**
Ours(TensorRT)	**Jetson NX**	229.9	736.1	**30.3**

2) Qualitative Results : As illustrated in Fig. 5, due to space constraints, we present a representative qualitative comparison across two scenes. Scene 1 (urban scenario): From top to bottom, the rows show (1) raw RGB, (2) depth



(a) Scene 1



(b) Scene 2

Fig. 5: Qualitative results on KITTI-DC.

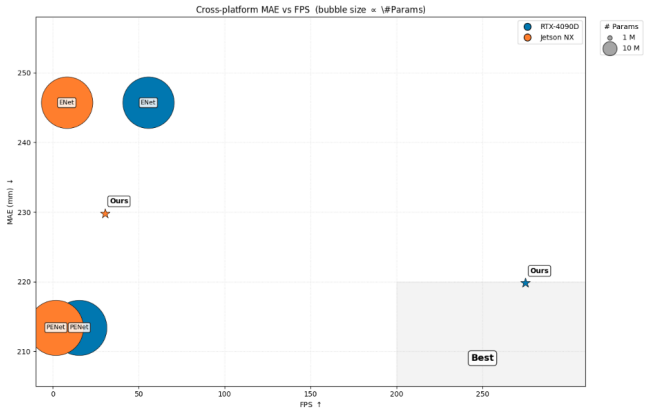


Fig. 6: Inference results obtained by our algorithm on both PC and Jetson platforms using the KITTI dataset..

predicted by our method, (3) depth from the current SOTA completion network [1] (i.e., the teacher used in distillation), (4) depth from PENet [6], (5) depth from ENet [6], and (6) monocular depth estimate (our second teacher [3]; geometrically coherent but metrically unscaled). Black dashed boxes highlight regions where both our student network and the completion teacher yield predictions that are visually consistent with the RGB image. Overall, the distilled model faithfully inherits—and in some local areas even refines—the teacher's knowledge. In contrast, PENet and ENet exhibit noticeable artifacts, such as depth discontinuities at distant utility poles and blurred reconstructions in LiDAR-sparse or unlabeled regions. Residual errors in our approach are mainly localized along object boundaries, which we consider acceptable given the inherent uncertainty of semi-dense ground truth in these regions.

Scene 2 (suburban scenario with dense vegetation): This scene presents a greater challenge, as LiDAR returns are sparse within tree foliage, and official ground truth labels are particularly scarce. From top to bottom, the six rows again correspond to: (1) raw RGB, (2) our predicted depth, (3) SOTA completion [1] network output, (4) PENet [6], (5) ENet [6], and (6) monocular depth estimate [3]. Red boxes highlight that our model preserves fine structural details of tree branches, closely resembling the geometrically plausible structures in the monocular depth estimate. In contrast, rows 3–5 (i.e., the comparison methods) fail to capture such fine-grained morphology. The road-side sign and tree trunks on the left also preserve sharper contours in our depth. Notably, although the SOTA completion method (row 3) theoretically provides the best metric scale, our result closely aligns with it in absolute depth values, while also retaining superior structural fidelity. This suggests that our method achieves a strong balance between geometric accuracy and structural coherence, even in challenging, LiDAR-sparse vegetated environments.

3) Cross-Platform Deployment on Jetson Xavier NX: As illustrated in Fig. 6, To further validate the practical value of our framework, we conduct a cross-platform study on the NVIDIA Jetson Xavier NX—a 15-W edge device with only 384 CUDA cores. Most prior networks either (i) contain heavy, unoptimised CUDA kernels or (ii) rely on custom layers that are incompatible with JetPack , so they cannot be deployed on the NX without extensive re-engineering. We therefore select PENet and ENet, the two reference models whose official code can at least be run for Jetson, and compare them with our distilled model.

We randomly sample 500 frames from the official KITTI-DC validation set and report the average scores in Table IV. On the desktop RTX-4090D, our network yields 275 FPS (3.6 ms) while keeping RMSE on par with PENet (732.1 mm vs 730.1 mm). After direct deployment on Jetson NX (TensorRT-FP16, no INT8 quantisation or pruning), we still achieve 30.3 FPS—×17 faster than PENet (1.8 FPS) and ×3.7 faster than ENet (8.1 FPS)—while maintaining an RMSE of 736.1 mm and an MAE of only 229.8 mm. Peak power remains < 20 W, so the device never triggers thermal

throttling; by contrast, both competitors must drop clocks to stay alive.

These results confirm that the proposed lightweight encoder–decoder, trained with the hardware-aware distillation pipeline, is the first depth-completion network that delivers real-time, high-precision inference on resource-constrained robots and autonomous vehicles without any network-level compression.

D. Ablation Studies

To verify the contribution of each component in our proposed two-stage hybrid distillation strategy, we conduct ablative experiments on the KITTI-DC validation set. All compared models are built upon **our designed network model** (sub 5 M parameters) and share identical training data, augmentation protocols, and inference settings. The experimental results are presented in Table V.Standard metrics RMSE, MAE, iRMSE and iMAE are reported.

1) Teacher Model Ablation: SOTA-C vs SOTA-M: We first distill the student using each teacher individually to demonstrate their complementarity.

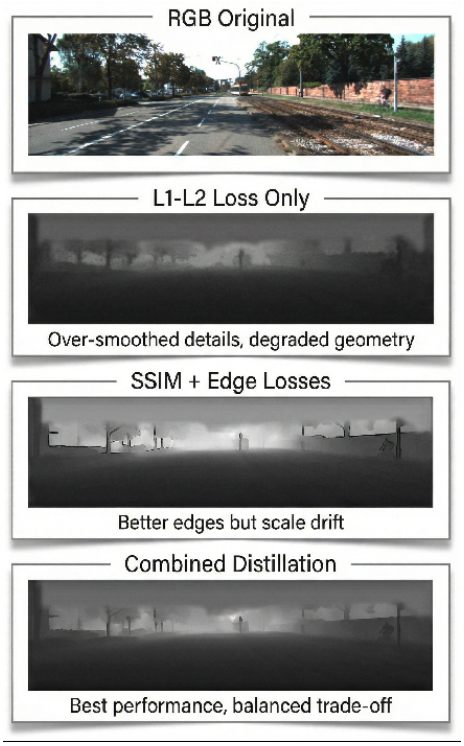


Fig. 7: From top to bottom, the four panels represent: 1. the RGB original, 2. the visual result supervised only by the L1-L2 loss, 3. the visual result supervised only by the SSIM and edge losses, and 4. the visual result obtained via distillation with both teachers using the combined L1-L2 loss.

Observations: As illustrated in Fig. 7,

(1) Supervision with the SOTA-C teacher alone using only the L1–L2 loss reduces RMSE but yields over-smoothed details; the global geometric structure is severely degraded, preventing the recovery of the original object shapes (e.g., the head of the cyclist is collapsed and distorted). (2) Employing the SOTA-M teacher alone with SSIM and edge losses better preserves edges and fine structures; however, it suffers from scale drift, leading to noticeable biases in absolute depth values. (3) The proposed two-stage joint distillation combines both teachers: the L1–L2 loss recovers metric accuracy, while the SSIM and edge losses maintain structural fidelity. Progressive optimization achieves a balanced trade-off between absolute scale and geometric detail, verifying the necessity of integrating both teacher models.

2) Training-Stage Schedule: Single-Stage vs Two-Stage: We merge the two stages into a single-stage multi-task training to examine whether the progressive schedule is necessary.

Conclusion: The two-stage strategy significantly outperforms the single-stage joint alternative, indicating that the "metric-first, structure-second" progressive optimization avoids task conflict and reaches a better Pareto optimum.

3) Effective Supervision Density: We finally compare the percentage of pixels that actually contribute gradients (effective supervision density) among different strategies.

Without any extra labels, our dense distillation raises the supervision density by $3.3\times$ and reduces RMSE by $110.0\,\text{mm}$, demonstrating its cost-effectiveness.

4) Summary: (1) Both teachers are indispensable: SOTA-C provides metric reference while SOTA-M offers structural priors. (2) The progressive two-stage schedule is superior to single-stage joint optimization. (3) Dense distillation significantly improves overall accuracy without extra parameters or annotations.

V. Conclusion

We present JetsonCompletion, an ultra-lightweight depth-completion framework tailored for power-constrained mobile platforms. By integrating a $< 5\,\text{M}$ parameter reparameterizable network with a two-stage Metric+Structure dual-teacher distillation scheme, the system delivers $> 30\,\text{FPS}$ dense depth at 1216×352 resolution on Jetson Xavier NX while peak power remains $\leq 20\,\text{W}$. Extensive experiments on KITTI-DC show that our approach maintains competitive accuracy with a $3.3\times$ increase in supervised gradients . The complete PyTorch$\rightarrow$TensorRT FP16 toolchain is released, offering an out-of-the-box low-power perception paradigm for micro-UAVs and embedded robots. Future work will extend the framework to higher-resolution panoramic sensors and explore on-the-fly domain adaptation for cross-scene flights.

Acknowledgments

The authors acknowledge the use of multiple large language models—including Kimi, Qwen3-Max, Grok, GPT-4o, Gemini series, and Claude—for language editing, grammatical refinement, and stylistic polishing of the manuscript text, as well as for code debugging and bug checking. All scientific content, including research conception, methodology, experimental analysis, and conclusions, was independently developed by the authors.

TABLE V: Ablation on teacher sources (RMSE / MAE). Loss terms are identical to the main two-stage formulation.

Teacher Source	Stage-1 Loss	Stage-2 Loss	RMSE↓ / MAE↓
Sparse-GT only	–	$\alpha\mathcal{L}_{\text{Sparse}}$	852.1 / 293.3
SOTA-C only	$\mathcal{L}_{\text{Metric-C}}$	$\mathcal{L}_{\text{Metric-C}} + \alpha\mathcal{L}_{\text{Sparse}}$	795.4 / 237.2
SOTA-M only	–	$\lambda_{\text{Struct}}\mathcal{L}_{\text{Struct}} + \alpha\mathcal{L}_{\text{Sparse}}$	824.6 / 260.4
Both teachers (full)	$\mathcal{L}_{\text{Metric-C}}$	$\mathcal{L}_{\text{Metric-C}} + \alpha\mathcal{L}_{\text{Sparse}} + \lambda_{\text{Struct}}\mathcal{L}_{\text{Struct}}$	**742.1 / 212.6**

Regarding figures: all images in the initial submission were manually created without any AI assistance. In the final camera-ready version, Figures 1, 4, and 7 have been reformatted and visually enhanced using *Gemini 3 Pro Image* exclusively for layout optimization and aesthetic improvement. The underlying scientific data and visual representations in these figures remain entirely author-generated and unaltered. All other qualitative comparison figures presenting experimental results are presented in their original form without AI-based modification.

Concerning references: all cited works were personally read, selected, and curated by the authors based on scholarly judgment. AI tools were used only to verify citation formatting consistency and apply IEEE reference style standards; they played no role in the selection, interpretation, or academic evaluation of the referenced literature.

REFERENCES

[1] Y. Liang, Y. Hu, W. Shao, and Y. Fu, "Distilling monocular foundation model for fine-grained depth completion," arXiv:2503.16970, 2025.

[2] Y. Zuo, W. Yang, Z. Ma, and J. Deng, "OMNI-DC: Highly Robust Depth Completion with Multiresolution Depth Integration," arXiv preprint arXiv:2411.19278, 2024.

[3] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth Anything V2," arXiv:2406.09414, 2024.

[4] J. Park, K. Joo, Z. Hu, C.-K. Liu, and I. S. Kweon, "Non-local spatial propagation network for depth completion," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, ser. Lecture Notes in Computer Science, vol. 12358, Cham: Springer, 2020, pp. 120–136.

[5] L. Meng, H. Li, B.-C. Chen, S. Lan, Z. Wu, Y.-G. Jiang, and S.-N. Lim, "AdaViT: Adaptive vision transformers for efficient image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022, pp. 12309–12318.

[6] M. Hu, S. Wang, B. Li, S. Ning, L. Fan, and X. Gong, "PENet: Towards precise and efficient image guided depth completion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 13656–13662.

[7] Y. Zhang, X. Guo, M. Poggi, Z. Zhu, G. Huang, and S. Mattoccia, "CompletionFormer: Depth completion with convolutions and vision transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 18527–18536.

[8] K. Rho, J. Ha, and Y. Kim, "GuideFormer: Transformers for image guided depth completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022, pp. 6240–6249.

[9] 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. Artif. Intell.*, vol. 34, 2020, pp. 10615–10622.

[10] J. Tang, F.-P. Tian, B. An, J. Li, and P. Tan, "Bilateral propagation network for depth completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024.

[11] H. Lin, S. Peng, J. Chen, S. Peng, J. Sun, M. Liu, *et al.*, "Prompting Depth Anything for 4K resolution accurate metric depth estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2025, pp. 17070–17080.

[12] Z. Wang, S. Chen, L. Yang, J. Wang, Z. Zhang, H. Zhao, and Z. Zhao, "Depth anything with any prior," arXiv preprint arXiv:2505.10565, 2025.

[13] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929, 2021.

[14] H. Touvron et al., "Training data-efficient image transformers & distillation through attention," arXiv preprint arXiv:2012.12877, 2021.

[15] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," arXiv preprint arXiv:2103.14030, 2021.

[16] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[17] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," arXiv preprint arXiv:1807.11164, 2018.

[18] K. Han et al., "GhostNet: More features from cheap operations," arXiv preprint arXiv:1911.11907, 2020.

[19] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "RepVGG: Making VGG-style ConvNets great again," arXiv:2101.03697, 2021.

[20] A. Wang, H. Chen, Z. Lin, J. Han, and G. Ding, "RepViT: Revisiting mobile CNN from ViT perspective," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 15909–15920.

[21] D. Nazir, A. Pagani, M. Liwicki, D. Stricker, and M. Z. Afzal, "SemAttNet: Towards attention-based semantic aware guided depth completion," *IEEE Access*, vol. 10, pp. 118277–118290, 2022.

[22] Y. Li, G. Yuan, Y. Wen, J. Hu, G. Evangelidis, S. Tulyakov, Y. Wang, and J. Ren, "EfficientFormer: Vision transformers at MobileNet speed," arXiv:2206.01191, 2022.

[23] S. Mehta and M. Rastegari, "MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2022.

[24] S. N. Wadekar and A. Chaurasia, "MobileViTv3: Mobile-friendly vision transformer with simple and effective fusion of local, global and input features," arXiv:2209.15159, 2022.

[25] X. Cheng, P. Wang, and R. Yang, "Depth estimation via affinity learned with convolutional spatial propagation network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, ser. Lecture Notes in Computer Science, vol. 11214, Cham: Springer, 2018, pp. 1–17.

[26] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, and B. Zeng, "DeepLiDAR: Deep surface normal guided depth prediction for outdoor scene from sparse LiDAR data and single color image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 3385–3394.

[27] J. Tang, F.-P. Tian, W. Feng, J. Li, and P. Tan, "Learning guided convolutional network for depth completion," *IEEE Trans. Image Process.*, vol. 30, pp. 1116–1129, 2021.

[28] S. Zhao, M. Gong, H. Fu, and D. Tao, "Adaptive context-aware multi-modal network for depth completion," *IEEE Trans. Image Process.*, vol. 30, pp. 5264–5276, 2021.

[29] S. Imran, X. Liu, and D. Morris, "Depth completion with twin surface extrapolation at occlusion boundaries," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 2573–2582.

[30] Y. Lin, T. Cheng, Q. Zhong, W. Zhou, and H. Yang, "Dynamic spatial propagation network for depth completion," arXiv preprint arXiv:2202.09769, 2022.