Depth on Demand: Streaming Dense Depth from a Low Frame Rate Active Sensor

Supplementary Material

This manuscript provides additional insights about the ECCV paper "Depth on Demand: Streaming Dense Depth from a Low Frame Rate Active Sensor". We collect here additional experimental material, an in-depth description of the modules composing our framework, and qualitative results.

1 Additional Implementation Details

1.1 Architectural Details

In this section, we deeply detail the core components of our framework – described in Section 3. Table I provides implementation details of the main components of our architecture. The visual cues integration module encodes separately the epipolar correlation features and the current depth estimate $(D)_i^N$, then it fuses such data with monocular data $\tilde{\mathcal{F}}_8^t$ using two Gated Recurrent Units with kernel size of 1×5 and 5×1; this latter choice is done as it leads to a lighter model than using a single 5×5 kernel. The depth cues integration module, composed of only four convolutional layers, fuses different depth representations obtained by different sources, as described in Figure 2. The depth decoding module computes multi-scale depth maps. At each iteration a set of upsampling weights and features are computed, then the upsampling is performed using convex upsampling. Finally, the hidden state $(\mathcal{H})_{i=0}^N$ initialization is performed by means of a simple convolutional layer.

1.2 Training Details

We train our model on ScanNetV2 [3], TartanAir [11], and KITTI [4] with AdamW, 10^{-4} learning rate and 10^{-5} weight decay. We always perform 100K training steps, dividing the learning rate by 10 at 60K and 90K steps. We train on 2 RTX 3090 in mixed precision with (total) batch size 8. Moreover, we clip gradients with global norm 1 to stabilize the behavior of Gated Recurrent Units and we enforce the same random seed in each training to increase reproducibility. On ScanNetV2 [3] we train on the same split defined by [8] with a buffer of 7 source frames to enable consistent comparisons. However, we evaluate on the whole test video sequences subsampled by a factor of ten to mimic a fast-moving camera in an indoor environment; since the camera moves really slow with respect to its high frame rate. We test on 7Scenes [9] in the same way. On TartanAir [11] we train and test on the whole video sequences without any frames subsampling using a buffer of 7 source frames while training. Finally, on KITTI [4] we perform training with a buffer of 3 source frames sampled with a frequency of 2Hz.

Visual Cues Integration

Input Tensor	Layer	K	S	In	Out	Output Tensor
$\mathcal{C}$	Conv2D + ReLU	1	1	$3^2 \times 41$	256	corr0
corr0	Conv2D + ReLU	3	1	256	192	corr1
$(D)_i^N$	Conv2D + ReLU	7	1	1	128	depth0
depth0	Conv2D + ReLU	3	1	128	64	depth1
depth1, corr1	Conv2D + ReLU	3	1	192+64	128-1	conv0
conv0, $\bar{\mathcal{F}}_8^i$, $(\mathcal{H})_i^N$, $(D)_i^N$	ConvGRU2D	(1, 5)	1	128+128+128	128	hidden0
hidden0	ConvGRU2D	(5, 1)	1	128	128	$(\mathcal{H})_{i+1}^N$

Depth Cues Integration

Input Tensor	Layer	K	S	In	Out	Output Tensor
$(\mathcal{H})_{i+1}^N$	Conv2D + ReLU	3	1	128	64	conv0
conv0	Conv2D	3	1	64	1	ΔD_c
$(\mathcal{H})_{i+1}^N$, ΔD_c, ΔD_d	Conv2D + ReLU	3	1	128+1+1	64	conv1
conv1	Conv2D	3	1	64	1	ΔD_f

Depth Decoding

Input Tensor	Layer	K	S	In	Out	Output Tensor
$(\mathcal{H})_{i=N-1}^N$, $\bar{\mathcal{F}}_8^i$, $(D)_{i=N-1}^N$	Conv2D + ReLU	3	1	$128+128+1$	$3^2 \times 4 + 64$	conv0
conv0	Conv2D	3	1	$3^2 \times 4 + 64$	$3^2 \times 4 + 64$	upweights8,feats8
upweights8, $(D)_{i=N-1}^N$	ConvexUpsample	-	-	$3^2 \times 4 + 1$	1	D^4
$\bar{\mathcal{F}}_4^i$, D_4, feats8	Conv2D + ReLU	3	1	$64+64+1$	$3^2 \times 4 + 32$	conv1
conv1	Conv2D	3	1	$3^2 \times 4 + 32$	$3^2 \times 4 + 32$	upweights4,feats4
upweights4, D_4	ConvexUpsample	-	-	$3^2 \times 4 + 1$	1	D^2
$\bar{\mathcal{F}}_2^i$, D_2, feats4	Conv2D + ReLU	3	1	$64+32+1$	$3^2 \times 4$	conv2
conv2	Conv2D	3	1	$3^2 \times 4$	$3^2 \times 4$	upweights4
upweights4, D_2	ConvexUpsample	-	-	$3^2 \times 4$	1	D^1

Hidden State Initialization

Input Tensor	Layer	K	S	In	Out	Output Tensor
$\bar{\mathcal{F}}_8^i$	Conv2D + Tanh	3	1	128	128	$(\mathcal{H})_{i=0}^N$

Table I: Architecture Modules Description. Description of the main components of our architecture in terms of layers, input and output dimensions. Each line represents a layer of a module where "Input Tensor" is the name of the input, "Layer" the type of layer used, "K" the kernel size if the layer is convolutional, "S" the stride if the layer is convolutional, "In" the number of input channels, and "Out" the number of output channels. Finally, "Output Tensor" is the name associated to the output tensor. Name of input and output tensors may refer to intermediate outputs described in the main paper.

2 Additional Ablation Studies

2.1 Number of Iterations

Our framework is characterized by an iterative module for depth refinement and multi-modal integration, in this section we study the impact of applying a different number of iterations at testing time. During training, we always perform 10 iterations. Figure I shows the mean absolute error in meters on the test split of 7Scenes [9] performing a different number of iterations. As can be clearly seen the network stabilizes its performance starting from 8 iterations, demonstrating its capability to reach a point of convergence. The number of iterations applied affects the time-accuracy trade-off of our approach. Thus, this latter can be adapted to the deploying requirements modulating such a parameter.

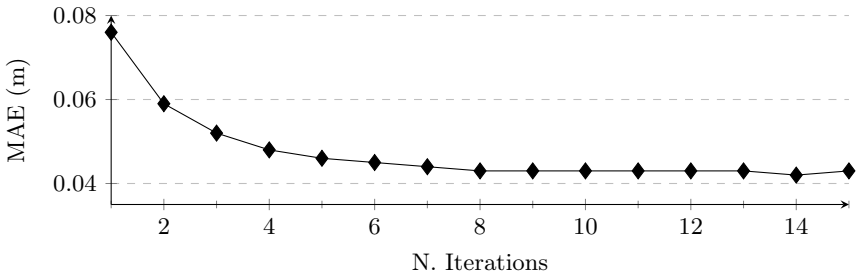

Fig. I: Number of Iterations. Performance using a different number of iterations on 7Scenes [9]. Our approach allows to change the number of iterations to adapt the time-accuracy trade-off required by the deploying environment.

200 Points

Method	MAE↓	RMSE↓	Abs Rel↓	Sq Rel↓	$\sigma < 1.05$↑
SpAgNet [2]	0.079	0.150	0.048	0.016	0.760
NLSPN [6]	0.075	0.152	0.046	0.017	0.794
CompletionFormer [12]	0.081	0.161	0.048	0.018	0.777
DoD (ours)	0.050	0.117	0.029	0.010	0.872

100 Points

Method	MAE↓	RMSE↓	Abs Rel↓	Sq Rel↓	$\sigma < 1.05$↑
SpAgNet [2]	0.094	0.165	0.056	0.018	0.696
NLSPN [6]	0.093	0.171	0.056	0.021	0.725
CompletionFormer [12]	0.099	0.181	0.059	0.023	0.703
DoD (ours)	0.061	0.129	0.035	0.011	0.832

50 Points

Method	MAE↓	RMSE↓	Abs Rel↓	Sq Rel↓	$\sigma < 1.05$↑
SpAgNet [2]	0.116	0.187	0.070	0.023	0.600
NLSPN [6]	0.118	0.198	0.071	0.026	0.616
CompletionFormer [12]	0.127	0.211	0.075	0.029	0.594
DoD (ours)	0.080	0.152	0.045	0.015	0.757

(a)

Method	Nr.	MAE↓	Time (ms)	Memory (GB)
Guided PatchMatch-Net [10]+ [7]	8	0.191	51.542	0.321
Guided CAS-MVSNet [5]+ [7]	8	0.120	95.556	1.100
Guided UCS-Net [1]+ [7]	8	0.141	104.63	1.064
Guided PatchMatch-Net [10]+ [7]	2	0.267	19.362	0.224
Guided CAS-MVSNet [5]+ [7]	2	0.250	50.750	0.983
Guided UCS-Net [1]+ [7]	2	0.228	55.668	1.063
SpAgNet [2]	1	0.068	46.249	0.817
NLSPN [6]	1	0.061	55.458	0.878
CompletionFormer [12]	1	0.067	89.741	1.141
DoD (ours)	2	0.043	32.625	0.263

(b)

Table II: Spatial Sparsification and Memory-Time Studies. On the left, Spatial sparsification study on 7Scenes [9]. We keep fixed the sparsification ratio $\tau = 0.2$ and progressively reduce the number of sparse depth points projected. Our approach is the most robust versus spatial sparsification since it enables multi-view cues exploitation. On the right, we study the memory and time impact of our approach. DoD provides the best trade-off performance, leading the accuracy ranking by a large margin and still being extremely lightweight in terms of memory and inference time.

2.2 Spatial Sparsification

In this section, we study the performance of our framework applying a variable *spatial* density to the sparse depth data. This is the case in which a different active sensor is used in the deploying environment. Moreover, it allows us to assess the effectiveness of our approach to exploiting multi-view cues. Indeed, when depth data sparsity increases, the unique information other than monocular features our method can leverage to retrieve accurate 3D reconstruction is the information extracted from the RGB source view. When training on ScanNetV2 [3], we always sample 500 sparse depth points. Table II(a) reports results obtained with variable density on 7Scenes [9], while keeping the temporal density fixed to $\tau = 0.2$. As the spatial density diminishes, depth completion methods struggle since they do not employ geometry cues. On the other hand, our approach is way more robust versus this kind of sparsity since it can recover useful information from the source view as well.

2.3 Memory and Time

We provide a detailed memory and time benchmark in Table II(b), to complete the overview provided in Figure 7 in the main paper, where only the methods providing the best trade-off between memory, time, and accuracy are highlighted. Notably, PatchMatchNet [10] using only 2 views is lighter and faster than our approach. However, with respect to our approach, it provides disastrous performance since the MAE error is 6× worse, with comparable memory occupancy and a small 1.7× speed boost.

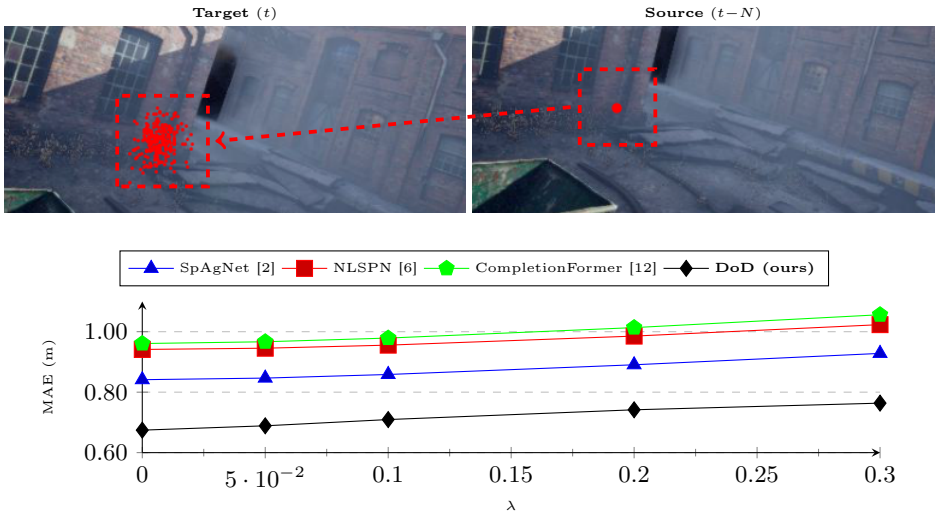

Fig. II: Pose Noise Sensitivity Study. We assess the impact of noisy pose in our and competitor frameworks perturbing pose information with Gaussian noise. At the top, we show qualitatively how such a noise affects depth point projection. At the bottom, we evaluate methods performance versus noise intensity on TartanAir [11]. Each model tested is trained with noise-free pose.

2.4 Pose Noise Sensitivity

We propose an additional experiment against test-time pose noise sensitivity, in the event that the pose estimator – *e.g.*, any visual-inertial odometry or SLAM pipeline – may introduce an error in the pose estimate. Let us represent a pose by a six degrees of freedom vector $\mathbf{q} := (\mathbf{t}, \mathbf{r}) \in \mathbb{R}^6$ with translation $\mathbf{t}$ and rotation $\mathbf{r}$ (in Euler angles). Given each ground truth pose $\mathbf{q}$, we draw random Gaussian noise on it, yielding the perturbed pose $\hat{\mathbf{q}} \sim \mathcal{N}(\mathbf{q}, \lambda^2 \operatorname{diag}(\mathbf{q})^2)$ where we dub λ the pose noise factor. We feed $\hat{\mathbf{q}}$ everywhere we would use $\mathbf{q}$ in the pipeline. At the top of Figure II, we report a visual example where we project a known sparse depth point with a set of 100 noisy poses drawn from the aforementioned distribution with pose noise factor $\lambda = 0.3$ for a given $\mathbf{q}$. Such noise not only affects our pipeline but any other depth completion method as well, in the assumption that the pose is used to reproject the sparse depth points since it corresponds to a significant uncertainty in the depth hints localization. In our case, the impact of pose errors is more subtle; first off, it affects the geometry cues, in that the epipolar correlation block samples along perturbed epipolar lines. Secondly, it affects the reprojected sparse depth points on the target view, as per the completion case. At the bottom of Figure II we report quantitative results sweeping $\lambda \in [0, 0.3]$ (where 0 is the noiseless case). Each model in this evaluation is trained with noise-free pose on TartanAir [11]. As it would seem that we could be more affected by pose errors, by this test we assess that the gap in performance between our method and depth completion methods remains

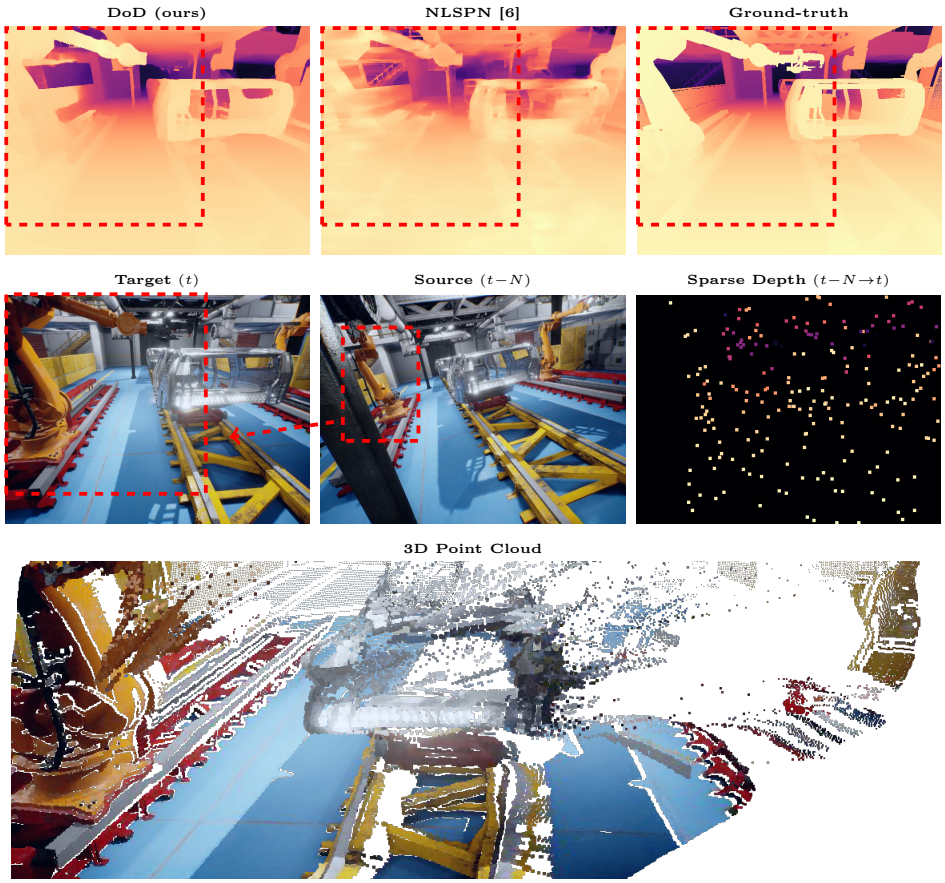

Fig. III: Moving Objects. Example of our framework behavior on moving objects in a video sequence from TartanAir [11]. On top, ours and [6] depth estimation. Below, are the target, source, and depth points used. In the dashed red bounding box is highlighted a robotic arm moving regardless of the camera. Last, is the 3D projection of our depth estimation. Our approach provides consistent monocular depth estimation where multi-view and depth cues are wrong.

fixed w.r.t. λ, *i.e.*, in a fair evaluation pose noise causes a degradation that increases gracefully with λ whilst keeping an almost fixed quality gap between all methods.

2.5 Moving Objects

Furthermore, we qualitatively assess the behavior of our approach on scenes with moving objects. Traditionally, multi-view methods work under the assumption of a static environment, to enable the use of multi-view geometry cues. Nonetheless, moving objects can occur in real use-cases. In the automotive scenario, other vehicles move [4]; in the indoor scenario people or objects can move. In this paper, we do not focus on the challenge of dealing with scene motion. Nonetheless,

we acknowledge the existence of this issue and thus we provide a qualitative study of how our framework behaves in moving areas. Figure III provides an example scene from TartanAir [11] where a robotic arm moves on an assembly line. When sparse depth points are projected from the source view I_{t-N} to the target view I_t sparse depth points gathered on the arm are wrongly projected. Moreover, multi-view cues are not useful in this case – *i.e.* even if the network predicts the correct depth on the target view for a moving object the projection on the source view leads to a wrong position. Thus, the monocular features are the unique useful information to estimate depth in the dashed red box. Our approach demonstrates to better exploit such information than NLSPN [6], effectively ignoring misleading multi-view and sparse depth information. Nonetheless, monocular depth estimated from a non-specialized approach is far from being fully accurate, as can be observed in the point cloud at the bottom of Figure III.

3 Qualitative Results

3.1 3D Reconstruction

We provide qualitative results about the final 3D reconstructions we obtain through our approach in indoor environments from ScanNetV2 [3] and 7Scenes [9]. To build each mesh we use 500 sparse depth points and sparsification ratio $\tau = 0.2$. Then, we integrate depth prediction in a TSDF volume using the same parameters as [8] and extract the mesh with the marching cubes algorithm. Qualitatives are showed in Figure IV and Figure V for ScanNetV2 and 7Scenes, respectively.

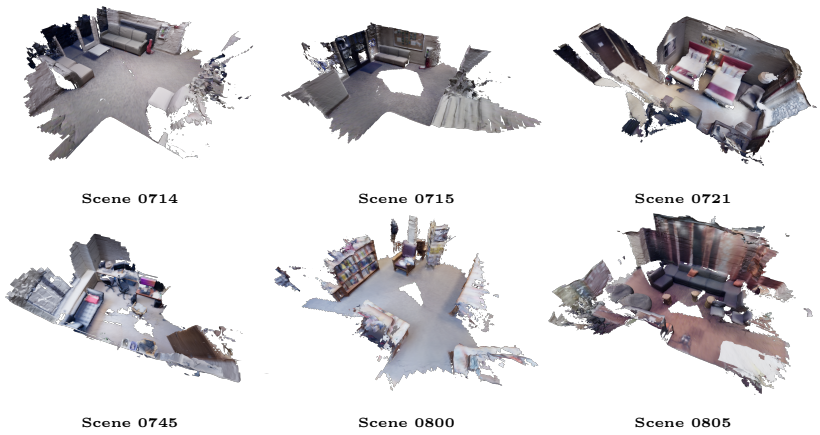

Fig. IV: ScanNetV2 3D Reconstruction. We provide different qualitative mesh reconstructions on ScanNetV2 [3] in different indoor scenarios. Our approach enables fine-grain effective RGB-D 3D reconstruction exploiting both high frame rate RGB cameras and slow yet accurate sparse active sensors.

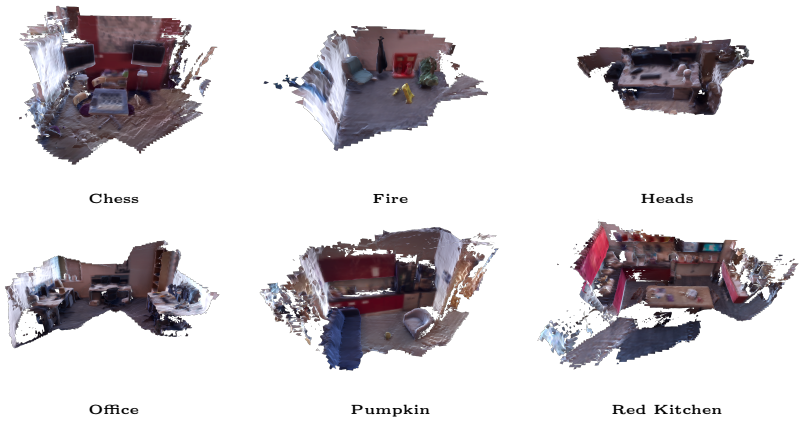

Fig. V: 7Scenes 3D Reconstruction. We provide qualitative mesh reconstructions on 7Scenes [9] in generalization – *i.e.* we train only on ScanNetV2 [3]. Our framework demonstrates the capability to easily generalize to new environments as assessed in the experiments on 7Scenes.

3.2 Generalization on Waymo

Finally, in Figure VI we provide a few inferences of DoD trained on KITTI in generalization on the Waymo Dataset. DoD provides reasonable reconstructions even in this scenario, despite the small size and repetitiveness of the KITTI dataset hamper good generalization.

Fig. VI: Qualitatives on Waymo. We test DoD trained on KITTI on the Waymo dataset in generalization. Although the KITTI's small size and repetitiveness hamper deployment in generalization DoD produces reasonable results predictions

References

1. Cheng, S., Xu, Z., Zhu, S., Li, Z., Li, L.E., Ramamoorthi, R., Su, H.: Deep stereo using adaptive thin volume representation with uncertainty awareness. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2524–2534 (2020)
2. Conti, A., Poggi, M., Mattoccia, S.: Sparsity agnostic depth completion. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 5871–5880 (January 2023)
3. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2017)
4. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
5. Gu, X., Fan, Z., Zhu, S., Dai, Z., Tan, F., Tan, P.: Cascade cost volume for high-resolution multi-view stereo and stereo matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2495–2504 (2020)
6. Park, J., Joo, K., Hu, Z., Liu, C.K., Kweon, I.S.: Non-local spatial propagation network for depth completion. In: Proc. of European Conference on Computer Vision (ECCV) (2020)
7. Poggi, M., Conti, A., Mattoccia, S.: Multi-view guided multi-view stereo. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2022), iROS
8. Sayed, M., Gibson, J., Watson, J., Prisacariu, V., Firman, M., Godard, C.: Simplerecon: 3d reconstruction without 3d convolutions. In: Proceedings of the European Conference on Computer Vision (ECCV) (2022)
9. Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A.W.: Scene coordinate regression forests for camera relocalization in rgb-d images. 2013 IEEE Conference on Computer Vision and Pattern Recognition pp. 2930–2937 (2013), https://api.semanticscholar.org/CorpusID:8632684
10. Wang, F., Galliani, S., Vogel, C., Speciale, P., Pollefeys, M.: Patchmatchnet: Learned multi-view patchmatch stereo. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14194–14203 (2021)
11. Wang, W., Zhu, D., Wang, X., Hu, Y., Qiu, Y., Wang, C., Hu, Y., Kapoor, A., Scherer, S.: Tartanair: A dataset to push the limits of visual slam (2020)
12. Zhang, Y., Guo, X., Poggi, M., Zhu, Z., Huang, G., Mattoccia, S.: Completionformer: Depth completion with convolutions and vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 18527–18536 (June 2023)