
Supplementary Material: Repurposing Marigold for Zero-Shot Metric Depth Estimation via Defocus Blur Cues

We organize the supplement as follows. In section A, we provide implementation details for our method. We outline the dataset capture parameters in section B, the calibration procedure to align RealSense and predicted depth in section C, and the per-scene quantitative metrics for the scenes in Fig.4 of the main paper in section D. We report some additional ablation studies in section E. In section F, we visualize how incorporating defocus cues through our optimization affects the relative depth predicted by Marigold. In section G, we demonstrate the results from using blurry images for all the F-stops (besides F/8 used in the paper) captured as part of the real dataset. In section H, we discuss ground truth data capture and show error maps for our predictions and MLPro to validate the ground truth data quality. We visualize predictions from our method with GeoWizard [1] as the backbone in section I and provide more implementation details for the experiments with GeoWizard. We compare our method with the MMDE baselines on the NYUv2 test set in section J, and with fine-tuning-based baselines in section K. We also provide the hyperparameters and additional details for fine-tuning the baselines in section K. We outline the comparison with methods that use multi-aperture inputs [9] on the NYUv2 test set in section L.

A Implementation details

We use RGB images of size 750×1126 , and depth maps captured at 480×640 resolution for evaluation in our method. In the CUDA implementation of the Disc PSF, we use a maximum window size (extent of i, j in equation 3 of the main paper) of 63 pixels.

B Real Dataset details

RGB Camera details We used a Canon EOS 5D Mark II as the RGB camera. This camera features a 21MP sensor with dimensions of 5616×3744 pixels and a pixel pitch of $6.41 \mu\text{m}$. The sensor uses an RGGB Bayer pattern. We used a Canon lens with a focal length (f in main text) of 50mm. This lens allows adjusting the F-stop from $N = 1.4$ to $N = 22$. We set the focus distance (F') to 80cm for all the captures. For our captured real dataset, we use the raw images (without any non-linear corrections) as inputs for our method. While providing the AIF as input to Marigold, we apply gamma correction ($x^{(1/2.2)}$) to it, but use the linear (raw) images for the loss function and forward model.

Depth Camera details We used an Intel RealSense D435 to capture ground truth depth. This camera has a stereoscopic depth sensor (with FOV 91.2 degrees) and a 2MP RGB camera. The depth sensor has less than 2% error at 2 meters (i.e., up to 4cm error at 2 meters). To reduce flying pixel errors in the captured depth maps (due to low texture/illumination, multipath interference, etc), we average the depth maps over 60 frames. We have uploaded the dataset at this link. For the THORDOG scene, F/11 and F/16 images are unavailable (due to image corruption), but all the other F-stops are present in the dataset. Please see the README.md file in the dataset for further details on usage.

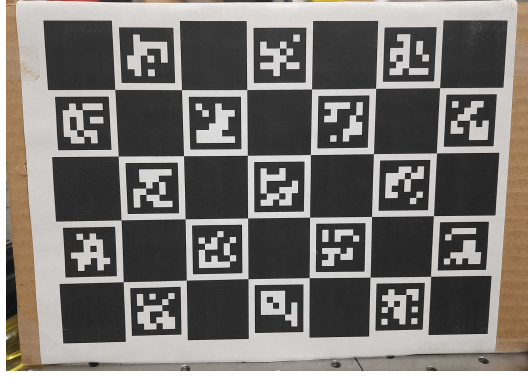


Figure 1: 7x5 ChArUco Board

C Calibration process to align RealSense and Predicted Depth

Our capture setup consists of the RealSense mounted rigidly to the flash mount of the DSLR (Fig. 2c, main text). Hence, for using the ground truth depth map from RealSense for evaluation, we estimate the pose of the DSLR with respect to the RealSense using camera calibration. We capture 10-15 images of a charuco board (fig. 1), sufficiently covering the depth and XY range. using the DSLR and the RGB camera of RealSense. Note that we use full-resolution images from the DSLR for calibration, and then later rescale the camera intrinsics appropriately. Using OpenCV functions, we estimate the relative pose transformation between the DSLR and the RealSense Camera. We use the *pyrealsense2* library for transforming the captured depth map from the RealSense Depth camera to the RealSense RGB camera pose. We will release the calibration code post acceptance. The above procedure provides us the pose transformation from DSLR to RealSense, denoted as $\mathcal{T}_{\text{DSLR}}^{\text{RS}} = \begin{bmatrix} R_{\text{DSLR}}^{\text{RS}} & T_{\text{DSLR}}^{\text{RS}} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4}$, where $R_{\text{DSLR}}^{\text{RS}}, T_{\text{DSLR}}^{\text{RS}}$ correspond to the 3×3 rotation matrix and 3×1 translation vector.

Aligning depth maps for evaluation Using the camera intrinsic matrix K_{DSLR} for the DSLR (estimated during calibration), we transform our predicted depth map (in DSLR coordinate system) to a point cloud \mathbf{P}_{DSLR} in the DSLR coordinate system.

$$\mathbf{P}_{\text{DSLR}}(x, y, z) = D(u, v) \cdot K_{\text{DSLR}}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad (1)$$

where $D(u, v)$ is the predicted depth at pixel (u, v) . We then transform $\mathcal{P}_{\text{DSLR}}$ to the RealSense coordinate system:

$$\mathbf{P}_{\text{RS}} = R_{\text{DSLR}}^{\text{RS}} \mathbf{P}_{\text{DSLR}} + T_{\text{DSLR}}^{\text{RS}}. \quad (2)$$

We then project the transformed point cloud \mathbf{P}_{RS} to the RealSense Camera plane using the camera projection equation to get the predicted depth map from the perspective of the RealSense camera. Note that since the DSLR has a much shorter FOV than the RealSense camera, the projected depth map (in RealSense coordinate system) will be sparser. We thus compute the metrics only for the pixels at which both ground truth and predicted depth maps have non-zero values. Note that we also undistort the RealSense depth map using the `cv2.undistort` function in OpenCV before evaluation, where the distortion coefficients are estimated during the camera calibration procedure by the OpenCV API.

D Per-scene metrics

We report the per-scene metrics in tables 1 to 7. While some competing methods (e.g., Metric3D, MLPro) perform marginally better on a few scenes such as THORDOG and SHOERACK, they exhibit significant failures on others. In contrast, our method demonstrates consistent performance across all

Method	RMSE ↓	REL ↓	log10 ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	CD ↓	FA ↑
Ours	0.505	0.205	0.075	0.772	0.940	0.984	0.201	0.726
MLPro	0.633	0.284	0.106	0.598	0.921	0.969	0.242	0.660
UniDepth	0.557	0.244	0.121	0.384	0.913	0.967	0.198	0.730
Metric3D	0.530	0.237	0.113	0.478	0.902	0.973	0.174	0.771

Table 1: Depth estimation metrics for the THORDOG scene.

Method	RMSE ↓	REL ↓	log10 ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	CD ↓	FA ↑
Ours	0.312	0.157	0.067	0.805	0.968	0.985	0.162	0.811
MLPro	0.302	0.174	0.080	0.733	0.976	0.990	0.153	0.791
UniDepth	0.415	0.235	0.118	0.492	0.861	0.971	0.156	0.803
Metric3D	0.287	0.191	0.092	0.548	0.966	0.994	0.095	0.894

Table 2: Depth estimation metrics for the BOOKS scene.

scenes, with substantial improvements in the average metrics reported in Table 1 of the main paper. For our real dataset, we observe that applying gamma correction to the raw all-in-focus (AIF) image improves UniDepth’s performance but not MLPro or Metric3D. To ensure a fair comparison, we therefore report results using gamma-corrected AIF input for UniDepth, and linear raw inputs for MLPro and Metric3D.

E Ablations

In table 8 and table 9, we demonstrate the robustness of our approach to different initializations of $\mathbf{z}_T^{(d)}$, and the number of sampling steps (value of T). Optimizing for consistency with defocus cues ensures that we obtain consistent metrics that are sufficiently robust (with small standard deviations, see table 8) to the initial noise latents. We observe that while increasing the number of inference steps leads to visually better results in some cases, it also increases texture-depth coupling.

F Visualizing Marigold predictions pre and post optimization

We quantitatively showed in Table 2 of the main paper that optimizing $\mathbf{z}_T^{(d)}$, i.e., refining both the relative depth and affine parameters, yields better results than optimizing only the affine scaling while keeping the Marigold-predicted depth fixed. In fig. 2, we visualize how incorporating defocus cues affects the relative depth predicted by Marigold.

In some regions of the SHOERACK scene (blue boxes/insets), our method struggles to correct a poor initial estimate for one of the shoes. However, across most other scenes (THORDOG, PLANE, KITCHEN, BOOKS), it visibly improves relative depth in several regions (see yellow insets). Recall that the final metric depth output depends on both the refined relative depth and the learned affine parameters. For example, in the KITCHEN scene, our method sharpens relative depth for objects like the bottles and stove (yellow inset), and compensates for upper-scene errors by adjusting the affine offset α , leading to a more accurate metric depth reconstruction (see Fig. 4 in the main paper).

Method	RMSE ↓	REL ↓	log10 ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	CD ↓	FA ↑
Ours	0.261	0.108	0.042	0.933	0.990	0.992	0.066	0.917
MLPro	0.750	0.399	0.218	0.005	0.210	0.995	0.262	0.381
UniDepth	0.452	0.216	0.101	0.519	0.992	0.994	0.161	0.774
Metric3D	0.297	0.130	0.047	0.919	0.989	0.992	0.090	0.902

Table 3: Depth estimation metrics for the STAIRS scene.

Method	RMSE ↓	REL ↓	log10 ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	CD ↓	FA ↑
Ours	0.064	0.049	0.022	0.996	0.998	0.999	0.050	0.924
MLPro	0.096	0.085	0.035	0.995	0.998	0.999	0.085	0.914
UniDepth	1.045	0.978	0.295	0.002	0.003	0.488	0.645	0.006
Metric3D	0.344	0.319	0.120	0.162	0.996	0.998	0.277	0.484

Table 4: Depth estimation metrics for the PLANE scene.

Method	RMSE ↓	REL ↓	log10 ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	CD ↓	FA ↑
Ours	0.346	0.125	0.058	0.827	0.951	0.990	0.085	0.905
MLPro	0.956	0.454	0.157	0.162	0.670	0.986	0.468	0.410
UniDepth	0.721	0.299	0.162	0.200	0.748	0.904	0.294	0.519
Metric3D	0.479	0.191	0.095	0.701	0.873	0.935	0.127	0.858

Table 5: Depth estimation metrics for the TOYS scene.

Method	RMSE ↓	REL ↓	log10 ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	CD ↓	FA ↑
Ours	0.251	0.121	0.053	0.903	0.996	0.998	0.101	0.888
MLPro	0.329	0.176	0.069	0.894	0.992	0.998	0.170	0.797
UniDepth	0.492	0.270	0.138	0.069	0.888	0.998	0.252	0.611
Metric3D	0.297	0.135	0.065	0.753	0.996	0.998	0.126	0.866

Table 6: Depth estimation metrics for the SHOERACK scene.

Method	RMSE ↓	REL ↓	log10 ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑	CD ↓	FA ↑
Ours	0.173	0.109	0.044	0.919	0.982	0.991	0.057	0.916
MLPro	0.206	0.154	0.070	0.791	0.980	0.993	0.052	0.922
UniDepth	0.337	0.263	0.132	0.174	0.897	0.995	0.111	0.844
Metric3D	0.209	0.164	0.077	0.712	0.983	0.994	0.056	0.920

Table 7: Depth estimation metrics for the KITCHEN scene.

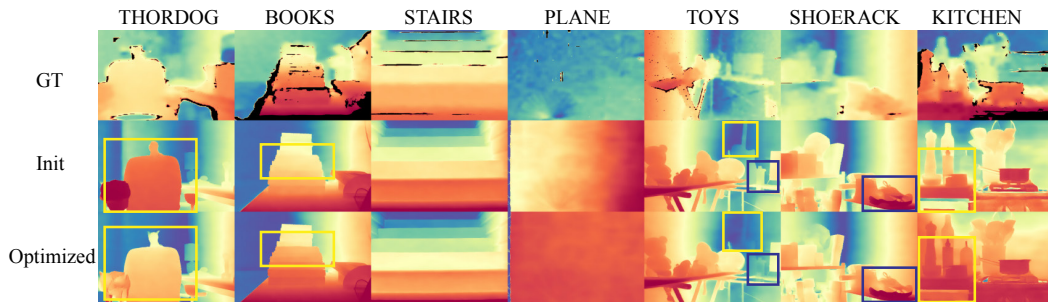


Figure 2: Marigold relative depth after optimization. We show regions (yellow boxes) where we observe improvements from our method (bottom row) compared to the initial relative depth prediction (middle row) from Marigold. We observe failure cases (blue boxes) in the SHOERACK scene, where the optimization fails to correct for the wrong initial prediction by Marigold. In the TOYS scene, while our model reduces the detail on the bottle (blue box), it recovers the correct relative depth region for the guitar region (yellow box).

Scene	δ_1	δ_2	δ_3	\log_{10}	REL	RMSE(m)	Runtime (s)
STAIRS	0.87 (0.03)	0.99 (0.0006)	0.99 (0.0004)	0.049 (0.0048)	0.12 (0.0089)	0.27 (0.01)	229.93
SHOERACK	0.90 (0.017)	0.98 (0.01)	0.99 (0.005)	0.052 (0.0036)	0.12 (0.0086)	0.25 (0.02)	229.91

Table 8: Mean (std. dev) for depth metrics from our method over 10 runs initialized with different noise latents $\mathbf{z}_T^{(d)}$ for the STAIRS and SHOERACK scenes. We observe considerable robustness to initial $\mathbf{z}_T^{(d)}$ initialization as demonstrated by low standard deviations.

Sampling steps	δ_1	δ_2	δ_3	\log_{10}	REL	RMSE (m)	Runtime (s)
1	0.8971	0.9807	0.9926	0.0477	0.1114	0.2345	228.07
2	0.8739	0.9799	0.9926	0.0500	0.1150	0.2352	328.68
3	0.8578	0.9666	0.9933	0.0569	0.1293	0.2501	430.27
4	0.8913	0.9796	0.9928	0.0514	0.1217	0.2410	530.25

Table 9: Varying the number of sampling steps in Marigold-LCM. We observe that increasing the number of sampling steps leads to higher optimization runtime, with only marginal improvements in quantitative metrics. We thus opt for a single sampling step in our method.

G Performance across different apertures on real data

Recall that in Fig. 5 (Left) of the main paper, we evaluate our method on blurred images simulated at different apertures. This synthetic experiment allows us to simulate a wider range of apertures for comparison, while isolating the effect of forward model mismatch from the effect of aperture size. In table 10, we show that the trend seen in Fig. 5 main paper – optimal performance at a specific F-stop with reduced accuracy at larger or smaller apertures, holds for real-world data as well. We obtain optimal performance at $N = 8$ on the real dataset, with degraded results at both smaller and larger F-stops. The optimal F-stop may differ between real and synthetic datasets, as performance in real data is influenced by both forward model mismatch (which depends on aperture size) and the strength of the blur cue.

H Ground truth data quality and error map visualizations

Acquiring very high-quality depth maps from real hardware is non-trivial and often requires complex structured light setups[7]. While we acknowledge that RealSense is not as precise as LiDAR or structured-light scanners, it is a much cheaper, hence academically viable alternative. It provides dense, metrically accurate depth within its operating range (0.2–3.8m) using stereo cues, and has been used as ground truth in prior monocular depth from defocus work [2]. We clarify that our goal is not to achieve sub-millimeter accurate ground truth, but rather to evaluate relative and metric consistency in practical indoor scenarios. RealSense has a much larger field of view compared to the DSLR camera used to capture the all-in-focus (AIF) image. As a result, while plotting in 2D, the details in the depth map look slightly blurry. We have visualized the error maps for our method and MLPro against the ground truth in fig. 4 and fig. 3, respectively. The visualization shows that MLPro (which has sharp boundaries in its predictions) struggles to predict correct depth scales for some areas in the

F-stop (N)	δ_1	δ_2	δ_3	\log_{10}	REL	RMSE (m)
4.0	0.8578	0.9499	0.9747	0.0536	0.1216	0.2722
8.0	0.8971	0.9807	0.9926	0.0477	0.1114	0.2345
11.0	0.8971	0.9802	0.9929	0.0507	0.1164	0.2335
13.0	0.6571	0.9088	0.9858	0.0905	0.1912	0.3682
16.0	0.5472	0.8539	0.9884	0.1107	0.2226	0.4172

Table 10: Depth estimation performance across different F-stop values. We report numbers averaged over all scenes, excluding THORDOG since it doesn’t have images for $N = 11, 16$. We observe that $N = 8$ is optimal across most metrics, with performance degrading for higher or lower F stops. We thus report results with $N = 8$ in the main paper.

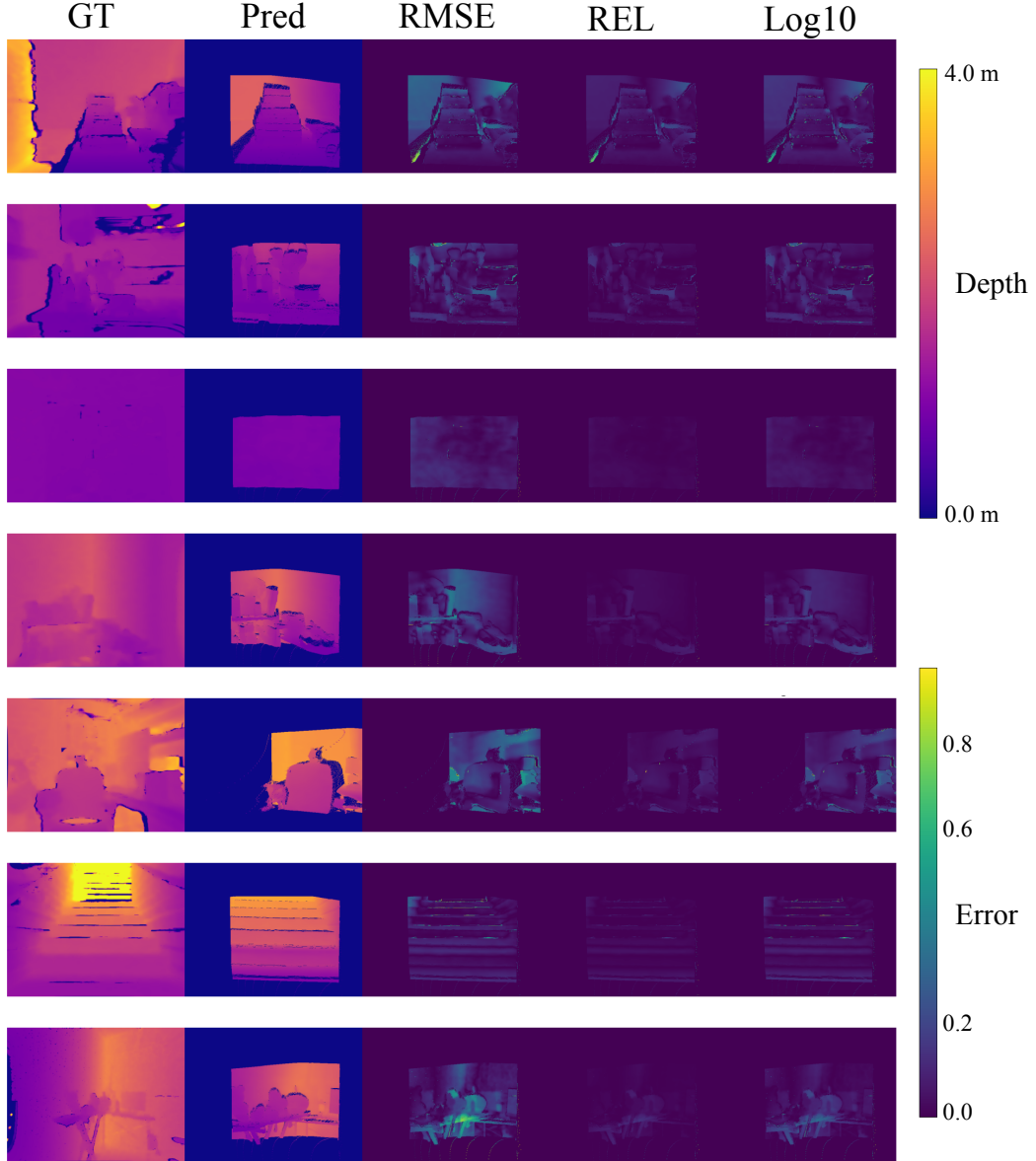


Figure 3: Error visualizations of our predictions on our dataset. We plot the per-pixel error for three metrics (RMSE, REL, and \log_{10}) between predictions from our model and the ground truth from RealSense. We observe that the failure regions are uniformly spread across edges (Row 6), constant texture backgrounds (Row 1, 4, 5), and certain texture-depth coupled patches (Row 4,5,7).

scene, resulting in significant errors on planar regions. Our visualizations indicate that there is no particular bias towards high error on sharp boundaries in our collected data, making it suitable as ground truth for comparison.

I Our method with Geowizard backbone

We visualize the results from our method with Geowizard as the backbone in fig. 5. While using Geowizard produces quantitatively similar results to using Marigold (Tab 1, main paper), we observe that the depth obtained from Geowizard after inference time optimization exhibits minor patch-level artifacts (last 2 rows in fig. 5), potentially due to stronger texture-depth coupling. Unlike Marigold, Geowizard does not provide a latent consistency model (LCM) checkpoint; hence, it requires more

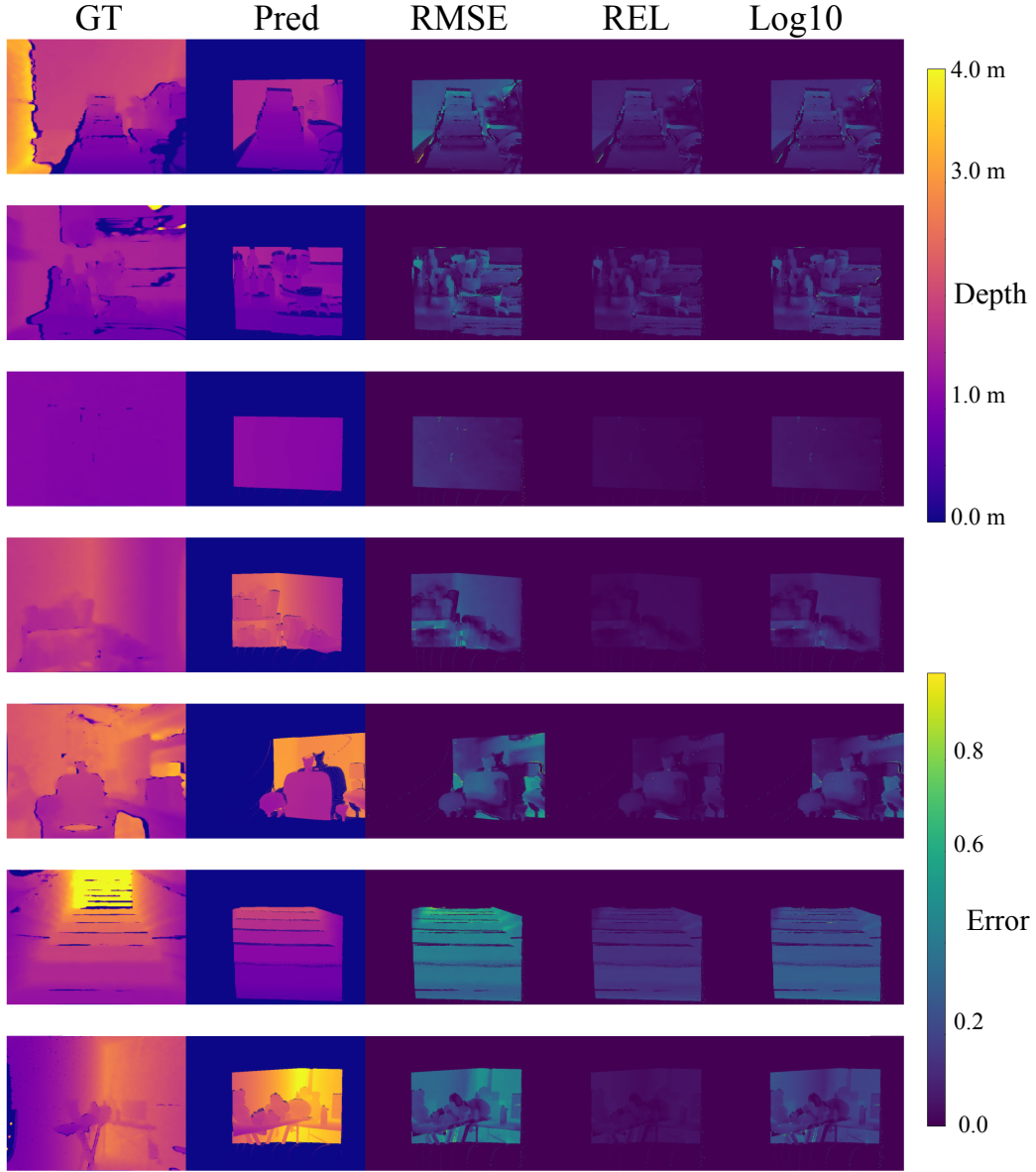


Figure 4: Error visualizations of MLPro predictions on our dataset. We plot the per-pixel error for three metrics (RMSE, REL, and log10) in columns 3-5, comparing predictions (column 2) from MLPro with the ground truth from RealSense. The Depth color bar corresponds to the first two columns, and the Error colorbar for the remaining 3 columns. MLPro produces outputs with sharper depth edges, but fails at accurate scale estimation (Row 5), resulting in higher error, as evident from brighter RMSE error maps for MLPro compared to our method. MLPro also struggles with depth estimation for the scene background walls (Row 4, 7), while our method performs qualitatively and quantitatively better than MLPro (fig. 3) due to incorporating physical depth cues in the form of defocus blur.

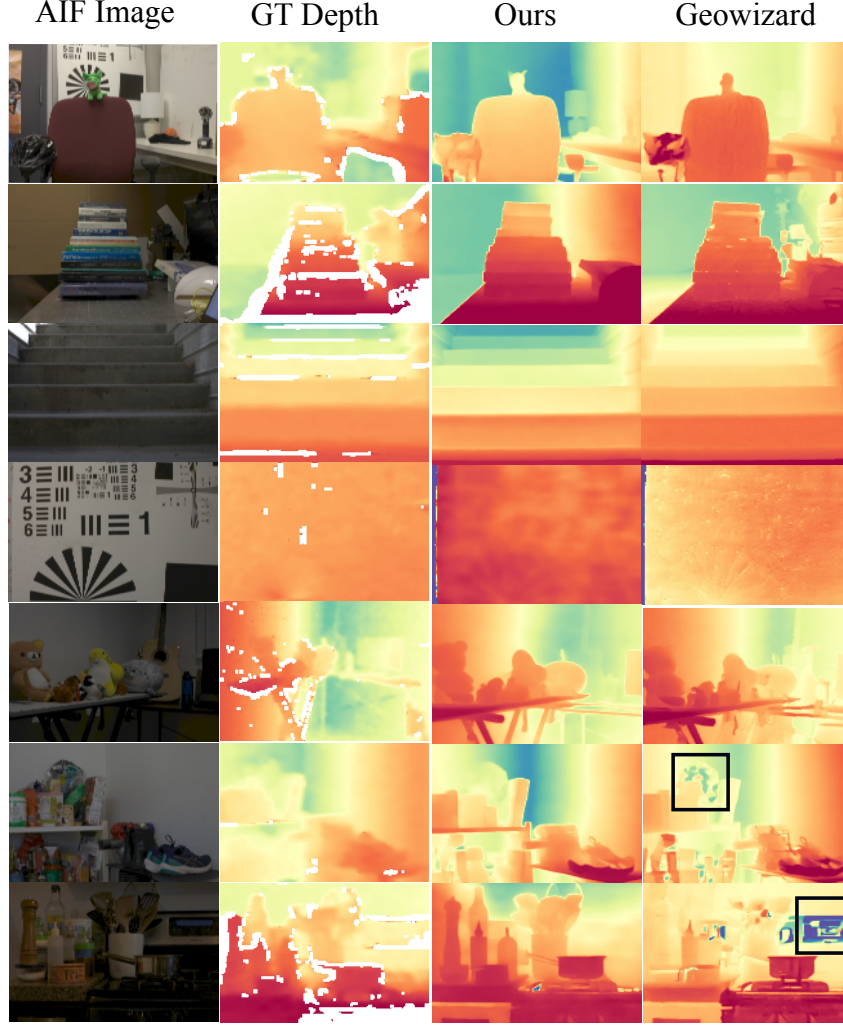


Figure 5: **Results from our method with Geowizard as the backbone.** While Geowizard produces visually similar results to our method, it exhibits stronger texture-depth coupling (highlighted in the last 2 rows), resulting in minor patch-based artifacts.

sampling steps during inference. The metrics for Geowizard obtained in Table 2 of the main paper are with 50 optimization iterations, and 5 sampling steps for Geowizard per iteration. We use a learning rate of 1.0×10^{-3} for all of the learnable parameters. Geowizard is 4x slower than the Marigold-LCM checkpoint on an A-40 GPU, but still achieves comparable results. This highlights the flexibility of our approach: as stronger or faster diffusion models become available, our method can seamlessly leverage them for improved performance. We report per-scene metrics for our dataset computed with the Geowizard backbone in table 11.

J Evaluations on NYUv2 Test set

We also compare our method and the aforementioned baselines on the test set (654 AIF-depth image pairs) of the NYUv2 dataset [5] in table 12. For our evaluation, we synthesize the blurred input at an aperture of F/8 and a focus distance of 1.5 m using our forward model for a simulated comparison. We emphasize that the leading methods are trained on Millions of Image-depth pairs to predict metric depth (Metric3D: 8M, UniDepth: 3M). In contrast, our method performs comparably to them without any re-training using defocus cues with a scale-invariant monocular depth estimator (Marigold). We

Scene	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	$\log_{10} \downarrow$	REL \downarrow	RMSE \downarrow	FA \uparrow	CD \downarrow
THORDOG	0.6800	0.9267	0.9837	0.0810	0.1733	0.4063	0.8361	0.1391
BOOKS	0.8850	0.9678	0.9814	0.0610	0.1501	0.2946	0.8367	0.1368
STAIRS	0.7280	0.9904	0.9931	0.0793	0.1737	0.3963	0.8320	0.1556
PLANE	0.9935	0.9969	0.9985	0.0317	0.0768	0.1048	0.9093	0.0598
TOYS	0.7483	0.9123	0.9890	0.0712	0.1443	0.4111	0.8777	0.1084
SHOERACK	0.8893	0.9869	0.9976	0.0458	0.0990	0.2110	0.9126	0.0702
KITCHEN	0.8469	0.9805	0.9892	0.0538	0.1406	0.2127	0.9116	0.0651

Table 11: Per Scene Metrics from our method with GeoWizard Backbone.

Method	REL \downarrow	RMSE \downarrow	$\log_{10} \downarrow$	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	CD \downarrow	FA \uparrow
MLPro	0.115	0.507	0.054	0.880	0.960	0.980	0.246	0.848
UniDepth	0.077	0.323	0.034	0.943	0.988	0.996	0.161	0.887
Metric3D	0.106	0.423	0.045	0.900	0.969	0.987	0.204	0.87
Ours	0.150	0.535	0.045	0.899	0.949	0.969	0.271	0.867

Table 12: **Comparison with learning based MMDE methods on NYUv2 dataset.** We achieve comparable performance to SOTA foundation model baselines (UniDepth, Metric3D, MLPro) across all metrics, while previous methods (bottom 2 rows) perform poorly. The metrics are computed by directly comparing each method’s metric depth output with the NYUv2 ground truth. The results are averaged over all 654 test images in the NYUv2 test split [3].

visualize some of the predictions on the NYUv2 test set from our method in fig. 6, demonstrating comparable performance to learning based MMDE methods trained on large metric depth datasets.

K Fine-tuning based baselines and NYUv2 test set results

Comparing our method with fine-tuning baselines on the NYUv2 test set On the NYUv2 dataset (table 13), we either outperform or are comparable to all the baselines except DepthAnythingv2 [14] fine-tuned on NYUv2, which is expected, as the fine-tuning dataset is in the same distribution as the test dataset.

Fine-tuning details We outline here the fine-tuning details for the baselines in Table 1 of the main paper.

- DepthAnythingv2 finetuned on HyperSim dataset – For this baseline, we use the checkpoint provided by [14] paper in their official codebase.
- DepthAnythingv2 fine-tuned on NYUv2 – We used the official fine-tuning code in the DepthAnythingv2 repository to fine-tune the base model for 25 epochs on the NYUv2 training set, with a learning rate of 5.0×10^{-5} on a NVIDIA A40 GPU.
- Marigold Fine-tuned on NYUv2 – We fine-tuned Marigold for 60 epochs using the official training code, with a learning rate of 2.0×10^{-5} on a Nvidia A-40 GPU. To obtain both

Method	REL \downarrow	RMSE \downarrow	$\log_{10} \downarrow$	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
[14] FT Hypersim	0.2136	0.6548	0.0822	0.6824	0.9685	0.9930
[14] FT NYUv2	0.0790	0.2990	0.0340	0.9510	0.9930	0.9980
Marigold FT NYUv2	0.1108	0.3730	0.0511	0.8689	0.9834	0.9965
Ours, No FT	0.150	0.535	0.045	0.899	0.949	0.969

Table 13: **Comparison with fine-tuning (FT) based methods on the NYUv2 test set.** Our method (row 4) with no fine-tuning (FT) outperforms DepthAnythingV2 FT on HyperSim (row 1), and is comparable to Marigold FT on NYUv2 (row 2). This highlights that fine-tuning-based methods can be sensitive to the domain gap between the FT dataset and the target test set, as DepthAnything only improves when fine-tuned on NYUv2 (row 2). We’d like to note that it is well established [6] that a training-based method, when trained on the same dataset, can be better than inference time methods. Our results show the key advantages of inference time methods: trading speed / slight performance for increased adaptability and no training.

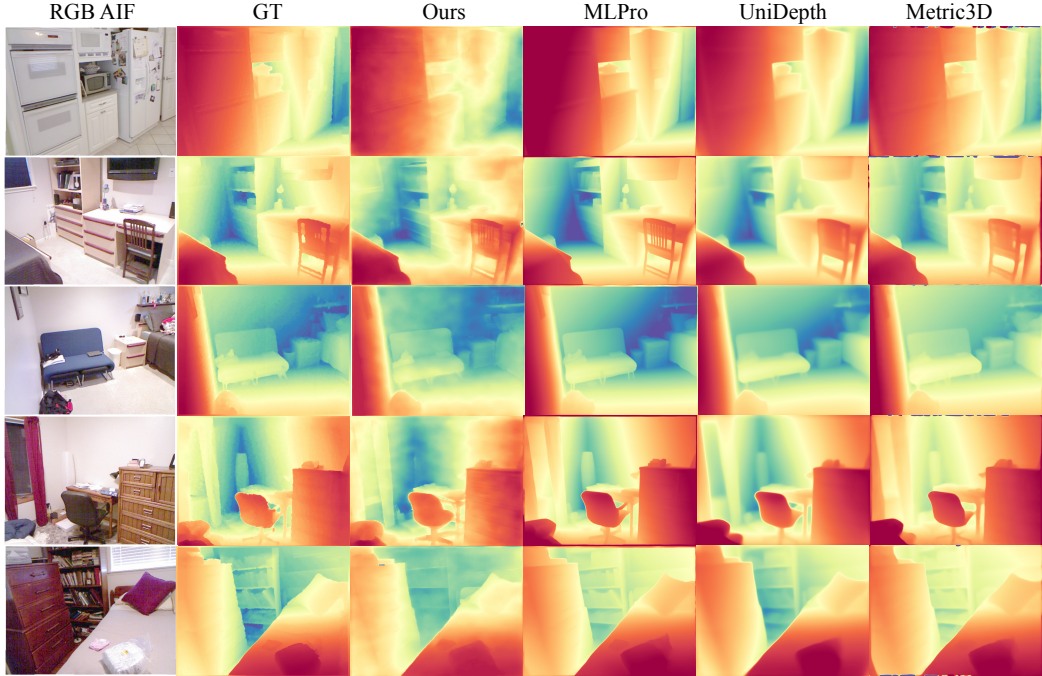


Figure 6: Qualitative comparison between our method and competing MMDE baselines on the NYUv2 test set. Our method performs comparably to existing SOTA MMDE methods, without being trained for metric depth estimation by leveraging defocus cues and inference time optimization. Leveraging defocus cues enables our method to recover correct depth scales, and also correct for certain depth monocular depth ambiguities (pillow in row 5), which learning based MMDE methods might struggle with. In some instances, however (row 4 wall bottom corner), our method may sacrifice local depth accuracy for global scale accuracy.

depth and RGB latents, Marigold uses the same encoder, requiring inputs normalized to $[0,1]$. This makes direct fine-tuning with metric depth supervision non-trivial. As a practical proxy, we normalize the GT depth maps (for training supervision) using a fixed global minimum and maximum depth for the full NYUv2 dataset (0.1m, 10m), rather than the originally used per-image normalization. Developing optimal fine-tuning strategies for Marigold with true metric depth supervision is beyond the scope of this work; our setup here serves solely as a baseline for comparison and context.

L Comparing with multi-aperture methods

While our work indeed falls under the category of multi-shot (two-image) monocular metric depth estimation, it differs from depth-from-defocus (DfD)–based methods [10, 4] by requiring only two images captured with different apertures but at the same focus distance. In contrast, most existing DfD approaches [8, 10, 11, 12, 13] rely on 4–5 or more images captured at varying focus distances.

Our setup is more similar to [9], which also relies on two input images: an AIF and a blurred image, captured at the same focus distance but different F-stops (aperture). They propose a differentiable forward model that takes an AIF and simulates defocus blur by interpolating pre-blurred AIFs at discrete depth levels, with weights predicted by a CNN that implicitly encodes scene depth. The network is trained to match the synthesized image to the captured (ground truth) blurred image. We implemented the method in [9], and also a ground truth depth-supervision-based baseline mentioned in their work, which involves directly regressing metric depth from a network with the AIF and blurred images (multi-aperture stack, captured at the same focus distance) as inputs. Following their training-based framework, we trained and tested these baselines on the NYU-v2 dataset. We observe in table 14 that both the baselines perform poorly compared to our method on the NYU-v2 test

set. The Depth supervision baseline produces blurry results, as observed in [9], while the method proposed in [9] struggles with strong texture-depth coupling, resulting in inaccurate metric depth.

Method	REL ↓	RMSE ↓	log10 ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
Ours	0.150	0.535	0.045	0.899	0.949	0.969
Multi-Aperture Depth Supervision	0.394	1.2667	0.1626	0.3815	0.6668	0.8401
Srinivasan et. al 2018	0.5221	1.7415	0.2571	0.2411	0.4592	0.6369

Table 14: **Comparison with multi-aperture depth estimation methods on the NYUv2 dataset.** Our method outperforms prior multi-aperture and camera-physics-based deep learning approaches across all metrics. All methods use both an all-in-focus (AIF) image and a blurred image as input. Chamfer Distance and FA metrics are omitted for the baselines due to their large gap from our method on the other 3D metrics.

References

- [1] Xiao Fu, Wei Yin, Mu Hu, Kaixuan Wang, Yuexin Ma, Ping Tan, Shaojie Shen, Dahua Lin, and Xiaoxiao Long. Geowizard: Unleashing the diffusion priors for 3d geometry estimation from a single image. In *European Conference on Computer Vision*, pages 241–258. Springer, 2024.
- [2] Bhargav Ghanekar, Salman Siddique Khan, Pranav Sharma, Shreyas Singh, Vivek Boominathan, Kaushik Mitra, and Ashok Veeraraghavan. Passive snapshot coded aperture dual-pixel rgb-d imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 25348–25357, 2024.
- [3] Shir Gur and Lior Wolf. Single image depth estimation trained via depth from defocus cues. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7683–7692, 2019.
- [4] Hayato Ikoma, Cindy M Nguyen, Christopher A Metzler, Yifan Peng, and Gordon Wetzstein. Depth from defocus with learned optics for imaging and occlusion-aware depth estimation. In *2021 IEEE International Conference on Computational Photography (ICCP)*, pages 1–12. IEEE, 2021.
- [5] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.
- [6] Zachary Novack, Julian McAuley, Taylor Berg-Kirkpatrick, and Nicholas J. Bryan. DITTO: Diffusion inference-time t-optimization for music generation. In *International Conference on Machine Learning (ICML)*, 2024.
- [7] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 195–202. IEEE, 2003.
- [8] Haozhe Si, Bin Zhao, Dong Wang, Yunpeng Gao, Mulin Chen, Zhigang Wang, and Xuelong Li. Fully self-supervised depth estimation from defocus clue. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9140–9149, 2023.
- [9] Pratul P Srinivasan, Rahul Garg, Neal Wadhwa, Ren Ng, and Jonathan T Barron. Aperture supervision for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6393–6401, 2018.
- [10] Huixuan Tang, Scott Cohen, Brian Price, Stephen Schiller, and Kiriakos N Kutulakos. Depth from defocus in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2740–2748, 2017.
- [11] Chao Wang, Krzysztof Wolski, Xingang Pan, Thomas Leimkühler, Bin Chen, Christian Theobalt, Karol Myszkowski, Hans-Peter Seidel, and Ana Serrano. An implicit neural representation for the image stack: Depth, all in focus, and high dynamic range. Technical report, 2023.
- [12] Ning-Hsu Wang, Ren Wang, Yu-Lun Liu, Yu-Hao Huang, Yu-Lin Chang, Chia-Ping Chen, and Kevin Jou. Bridging unsupervised and supervised depth from focus via all-in-focus supervision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12621–12631, 2021.
- [13] Changyeon Won and Hae-Gon Jeon. Learning depth from focus in the wild. In *European Conference on Computer Vision*, pages 1–18. Springer, 2022.
- [14] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv preprint arXiv:2406.09414*, 2024.