MVSAnywhere: Zero-Shot Multi-View Stereo

Sergio Izquierdo[3,*] Mohamed Sayed[1] Michael Firman[1] Guillermo Garcia-Hernando[1]
Daniyar Turmukhambetov[1] Javier Civera[3] Oisin Mac Aodha[2] Gabriel Brostow[1,4] Jamie Watson[1,4]
[1]Niantic [2]University of Edinburgh [3]I3A, Universidad de Zaragoza [4]UCL
https://nianticlabs.github.io/mvsanywhere/

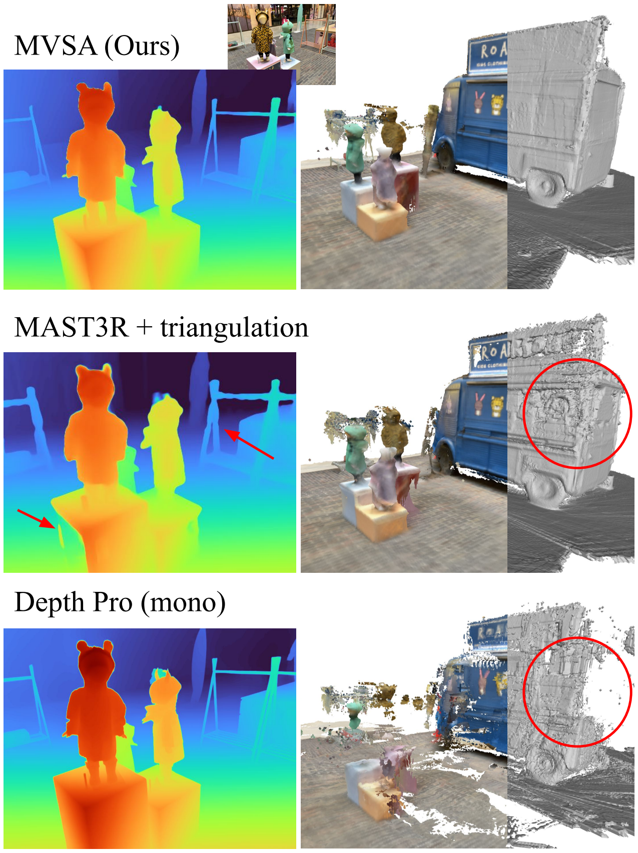
Abstract

*Computing accurate depth from multiple views is a fundamental and longstanding challenge in computer vision. However, most existing approaches do not generalize well across different domains and scene types (*e.g. *indoor* vs. *outdoor). Training a general-purpose multi-view stereo model is challenging and raises several questions,* e.g. *how to best make use of transformer-based architectures, how to incorporate additional metadata when there is a variable number of input views, and how to estimate the range of valid depths which can vary considerably across different scenes and is typically not known a priori? To address these issues, we introduce MVSA, a novel and versatile Multi-View Stereo architecture that aims to work Anywhere by generalizing across diverse domains and depth ranges. MVSA combines monocular and multi-view cues with an adaptive cost volume to deal with scale-related issues. We demonstrate state-of-the-art zero-shot depth estimation on the Robust Multi-View Depth Benchmark, surpassing existing multi-view stereo and monocular baselines.*

1. Introduction

Estimating accurate depth from multiple RGB images is a core challenge in 3D vision, and a building block for downstream applications like 3D reconstruction and autonomous driving. Recent approaches in learning-based multi-view stereo (MVS) are capable of generating accurate depths [6, 95, 96]. However, existing methods typically struggle to generalize to scene and camera setups that differ significantly from those in their training data. As a result, there is a pressing need for general-purpose MVS methods that are more robust to differences between the training and test distributions.

We take inspiration from the recent explosion in scene-agnostic *single-view* depth models, which predict plausible metric [2, 26, 30, 56, 81, 98] or up-to-scale [39, 59, 93, 94] depth using only a single image as input. These models are typically trained on large curated sets of synthetic and/or real RGB-D data, endowing them with impressive generalization performance on previously unseen data. Single-view models are, however, inherently limited by their input.



MVSA (Ours)

MAST3R + triangulation

Depth Pro (mono)

Figure 1. Our **MVSA** model results in high-quality reconstructions from posed images, and is superior to existing monocular and MVS methods. Here we compare with Depth Pro [3], a recent monocular method which produces sharp and good looking depth maps, but can have inconsistent scaling of depths, which are required for good meshes. We also include a variant of MAST3R [43] that we have augmented with ground truth camera poses. Our model gives sharp depth maps which are also accurate and 3D consistent, producing high-quality meshes in zero-shot environments.

[*] Work done during an internship at Niantic.

For our specific depth prediction target, constraining the model's input to just one image forces it to use single-view geometry cues (*e.g.* vanishing points) and learned patterns [15], while losing the stronger multi-view signal. While there are temporal extensions of these single view models [55, 68, 80, 85, 90], their focus is on temporal perceptual consistency, and not necessarily multi-view consistency. In application contexts where multiple views are available at inference time, it stands to reason that these lead to significantly more accurate depth estimates [36, 49, 70].

Developing a general-purpose MVS method, however, raises two significant challenges. Firstly, it should be able to deal with arbitrary depth ranges. Existing MVS methods typically require a known range of depths to 'search' over along epipolar lines, corresponding to a discrete set of depth bins used to build a cost volume. These depths are typically either fixed (and chosen from the range of depths in the training data) [62] or are provided at test time for each image [76, 95, 96]. Secondly, the many emerging benefits of ViTs [17] motivate us to find a way to 'upgrade' parts of standard MVS architectures that are still CNNs.

To address these challenges, we introduce a new general-purpose MVS method named Multi-View Stereo Anywhere (**MVSA**). Similarly to recent performant monocular methods, it is trained on a large and diverse set of data, spanning diverse depth ranges. Along with harmonizing these training signals, our main technical contributions are:

- A novel transformer-based architecture that processes the multi-view cost volume, while *also* incorporating monocular features. We propose a Cost Volume Patchifier that tokenizes the cost volume without loosing its details, while also incorporating features from a monocular ViT.
- We propose a view-count-agnostic and scale-agnostic mechanism to construct the cost volume using geometric metadata given any number of input source frames. This is in contrast to the established practice [62] of concatenating geometric metadata from a fixed number of frames to build the cost volume.

MVSA predicts highly accurate and 3D-consistent *depths*, obtaining state-of-the-art results on the Robust Multi-View Depth Benchmark [67], which contains a variety of challenging held-out datasets. We also report scores for some new single- and multi-view methods for comparison. Our better depths result in improved 3D mesh *reconstruction* compared to alternative depth-based reconstruction methods (Fig. 1). Code and pretrained models are available at https://github.com/nianticlabs/mvsanywhere.

2. Related Work

Multi-view stereo (MVS). MVS algorithms estimate depth from posed multi-view images using epipolar geometry [77]. Given calibrated cameras, early methods estimated depth by matching image patches [22, 65]. Subsequently, deep learning approaches were introduced, first for stereo matching [100] and later improved via end-to-end learning, typically using plane-sweep cost volumes [11, 25, 34, 35, 40, 73, 79, 96, 101]. Subsequent methods introduced advances in architectures [6, 7, 16], increased robustness to occlusion and moving objects [47, 88, 103], integrated temporal information [18], improved model efficiency [62, 99], jointly estimated camera pose [43, 82] and ingested prior geometry estimates to improve depths [63].

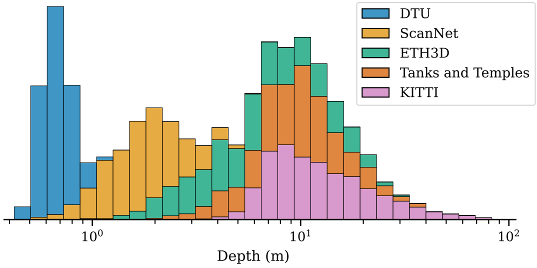


Figure 2. **MVS datasets cover a wide range of depth values.** Here we show the distribution of % depths in the DTU [37], ScanNet [13], ETH3D [66], Tanks and Temples [42], and KITTI [24] datasets, as a stacked bar chart. Note the log x-axis. This wide range of depth values can be challenging when it comes to constructing meaningful cost volumes and predicting the final depths.

With some exceptions [102], earlier stereo and MVS methods were traditionally both trained and tested on the same dataset/domain, and were limited in their ability to generalize to out-of-distribution data. This domain generalization issue is a consequence of most performant learning-based MVS methods being data-hungry. Approaches such as a training on synthetic [50, 104] or pseudo-labeled depth [31] can be effective, but so far, struggle to span a diverse range of scene types and scales. Self-supervised approaches can be trained without depth supervision, but current methods produce inferior depths compared to fully supervised approaches [14, 41, 92]. Concurrent with our work, [27, 86] trained large *binocular* stereo models on large synthetic datasets.

Adaptive cost volumes. One of the challenges in developing a general-purpose domain-agnostic MVS method is that different scenes can contain wildly different depth ranges, *e.g.* indoor scenes are limited to a few meters, while outdoor ones can span much larger distances. This is a problem as conventional cost volumes require a known depth range, which is typically just estimated based on the minimum and maximum depth values in the training set. As a result, there is a need for cost volumes that are not restricted to a pre-defined range or bins, and instead are adaptive. In the context of self-supervised learning with unscaled poses, [85] estimated bin ranges at training time via an exponential moving average of the depth predictions. Another approach is to predict bin centers iteratively in a coarse-to-fine manner, where the outputs from the previous iteration are used to seed the range in the next [25, 51, 105]. Alternatively, the bin offsets can be predicted by a learned network [12] or from estimated depth uncertainty [10, 44]. We estimate cost volume depth ranges to enable us adapt to any range of depths, while prior work has done this when the test time range is known, but they wish to reduce computation or enhance detail.

Single-view depth. Monocular depth methods, trained using supervised learning, do not have access to multi-view images at inference time so instead aim to estimate a depth map from a *single* image [19–21, 46]. Building on highly advanced and effective image backbones [54], more recent monocular methods have focused on making *general-purpose* depth estimation models, which aim to work on arbitrary scenes [9, 59]. Further works have scaled up the size of models and datasets, training on combinations of real and/or synthetic data [93, 94], and have used stronger image-level priors [28, 39]. One of the limitations of models trained from stereo-image-derived supervision without known baselines [59] or human annotations [9] is that these only enable a relative, and not metric (*e.g.* in meters), depth prediction.

Other monocular models predict *metric* depth [2, 3, 30, 81, 98]. Not only does this rely on appropriate training data, but also requires an understanding of camera intrinsics, which are often a required additional input to the network. Conventional monocular methods are inherently limited by only incorporating information from single views at inference time, even when multi-view information is available [85]. On the other hand, with recent advances, they can still provide a very valuable signal when only one image is available. As in [1, 84, 89], we combine features extracted from a monocular depth model with a multi-view cost volume to better leverage monocular and multi-view cues.

3. General-purpose Multi-View Stereo

Our model takes as input a $H \times W$ reference image I_r together with neighboring source frames $I_{i \in \{1 \ldots N\}}$, each with their relative poses and intrinsics. At test time we aim to predict a dense depth map $\hat{D}_r$ for I_r. For ours to be a general-purpose MVS method, we seek to:

1. **Generalize to any domain**. Most current MVS methods are typically trained on and tested on data from similar domains, *e.g.* indoor only or driving only.
2. **Generalize to any range of depths**. Predicted depth maps need to be accurate for nearby surfaces (*e.g.* for robotics) or for more distant ones (*e.g.* for drones and autonomous driving). In some scenarios like SfM, the depths and camera poses are in a non-metric up-to-scale coordinate system. Hence, general-purpose MVS should be robust to the scale of the coordinate system.
3. **Be robust to the number and selection of source frames**. Traditional MVS systems can struggle when there is little overlap between source and reference frames. We also want MVS methods to be agnostic to the number of source frames available at test time.

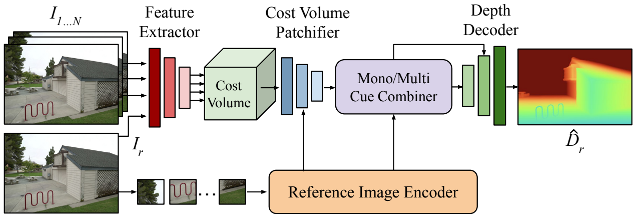


Figure 3. **Our general-purpose multi-view depth estimation model**. We start with a cost-volume based architecture, which matches deep features between views at different hypothesized depths. Key for performance are our Cost Volume Patchifier and Mono/Multi Cue Combiner. These also fuse single-view information coming from the Reference Image Encoder and source views.

4. **Predict 3D-consistent depths.** Depths from one viewpoint should be consistent with those predicted from different viewpoints. Fusion of consistent depth maps will produce a mesh with accurate estimates of 3D surfaces.

While prior works have tackled these problems in turn, we are the first model, to the best of our knowledge, to tackle all four problems in a single system.

3.1. MVSAnywhere

We introduce **MVSAnywhere** (MVSA), a novel general-purpose MVS system which is designed to embody each of the previous properties. To help us learn from diverse datasets and hence **generalize to any domain**, we use a large transformer-based architecture, which takes as input: (1) multi-view information from the reference and source images, and (2) single-view information, which is extracted directly from the reference image via a monocular *reference image encoder*. The overall architecture (Fig. 3 and supplementary) is broadly inspired by recent MVS approaches, *e.g.* [62]. It comprises five key components:

Feature extractor. This encodes the source and reference images into deep feature maps $\mathcal{F}_r$ and $\mathcal{F}_{i \in \{1 \ldots N\}}$, that will be processed via a cost volume. We use the first two blocks of a ResNet18 [29] for this encoder, producing feature maps at resolution $H/4 \times W/4$.

Cost volume. Following *e.g.* [7, 34, 40, 79], we warp feature maps $\mathcal{F}_i$ from each source view to the reference one using a set of hypothesized depth values (*i.e.* bins) $\mathcal{D}$. We then concatenate these warped features and $\mathcal{F}_r$ with appropriate *metadata*, following [62]. See Sec. 3.2 for our specific novel contributions in this matter.

Reference image encoder. This extracts powerful deep monocular features for I_r. We use the ViT Base [17] encoder from Depth Anything V2 [94], with their pretrained weights for relative monocular depth estimation, which help us to be **robust to limited overlaps** between source

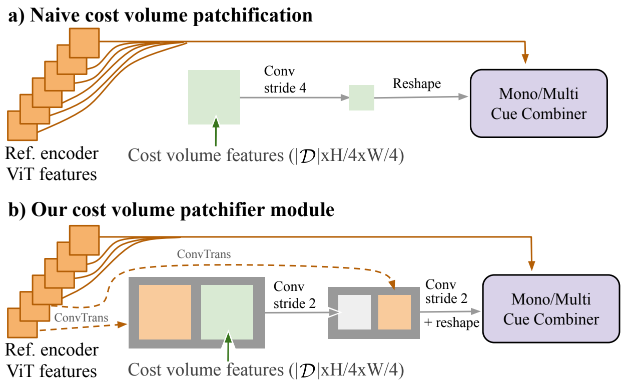

a) Naive cost volume patchification
Conv
stride 4
Reshape
Mono/Multi
Cue Combiner
Ref. encoder
ViT features
Cost volume features (|$\mathcal{D}$|xH/4xW/4)
b) Our cost volume patchifier module
ConvTrans
ConvTrans
Conv
stride 2
Conv
stride 2
+ reshape
Mono/Multi
Cue Combiner
Ref. encoder
ViT features
Cost volume features (|$\mathcal{D}$|xH/4xW/4)

Figure 4. **Our cost volume patchifier** enables high-quality information to be extracted from a $|\mathcal{D}| \times \frac{H}{4} \times \frac{W}{4}$ cost volume, ready for input to the Mono/Multi Cue Combiner ViT. (a) Shows the naive approach to patchification. (b) Our approach makes better use of the reference image features.

and reference frames. As ViT Base operates on 14×14 patches, the reference image is resized to $\frac{14H}{16} \times \frac{14W}{16}$ resolution before feeding to ViT Base, such that the extracted features are size $\frac{H}{16} \times \frac{W}{16}$.

Mono/Multi Cue Combiner. This converts the "patchified" features of the cost volume and reference image into a sequence of features which go to our depth decoder. Monocular and multi-view cues are combined by a novel component described in Sec. 3.3.

Depth Decoder. Based on the decoder from [58], MVSA progressively upsamples and processes features from the Mono/Multi Cue Combiner module to produce the final depth map at the reference image resolution.

3.2. Metadata Agnostic to View Count and Scale

SimpleRecon [62] demonstrated that readily available metadata, *e.g*. geometric and camera pose information, can be incorporated into the cost volume to improve depths. For each pixel location (u_r, v_r) in I_r and depth bin k in $\mathcal{D}$, we backproject the pixel to a 3D point P and then reproject it into every source view I_i. The specific metadata for the bin with coordinates (u_r, v_r, k) in the cost volume includes: the dot product between feature vector $\mathcal{F}_r(u_r, v_r)$ and corresponding pixels in $\mathcal{F}_i(u_i, v_i)$, ray directions from source and reference origin to P, depths in reference and source views, angle between rays from reference and source views, relative poses between the reference and source cameras, and depth validity masks (in case P is outside a source frame frustum). See our supplementary for full details.

SimpleRecon [62]'s cost volume concatenates metadata from all eight source frames and runs an MLP to produce one single cost (matching score) per spatial location and depth hypothesis. While this gives good scores, its limitation is that it requires *exactly* eight source frames for every training and test reference image, limiting the model's flexibility (note though that traditional MVS methods are typically already view-count agnostic). To address this limitation, we introduce a *view-count-agnostic metadata* component which enables a single model to **generalize to any number of source views**. For each source frame, we run an MLP that ingests the metadata from the reference frame and the source frame and predicts two values: a score and a weight. This results in N scores and N weights. A weighted sum of the N scores is computed after the N weights go through softmax. This weighted sum is used as the value in the cost volume at every pixel location (u, v) and depth hypothesis k. Our novel module enables aggregation of the matching score and confidence for each source frame, while allowing for a variable number of source frames for each I_r.

The source camera poses may be close to the reference, or far from it. To be more invariant to this possible range of scales, we also make the metadata **scene scale-agnostic**. To this end, we normalize the relative pose measures of the metadata using a maximum across all the source frames for a given reference frame. We also normalize the depth hypothesis metadata using the maximum and minimum of $\mathcal{D}$.

As the scene scale information is not provided to the rest of our network, we rescale the depth predictions to match the scale of the input poses. Our depths are predicted with a sigmoid function σ over the logit x. To align the prediction of the network with the cost volume, the sigmoid output is scaled by the depth range of the cost volume, so

$$\hat{D}_r = \exp\left(\log(d_{\min}) + \log(d_{\max}/d_{\min}) \cdot \sigma(x)\right). \quad (1)$$

3.3. Mono/Multi Cue Combiner

Given the cost volume of shape $|\mathcal{D}| \times \frac{H}{4} \times \frac{W}{4}$, and the reference image encoder features of shape $C \times \frac{H}{16} \times \frac{W}{16}$ (outputs of different blocks of the reference image encoder), we pose the question: how can we best combine these features to provide the strongest signal for the decoder? Motivated by the recent success of transformer architectures in single-view depth prediction [3, 58, 94], we use a ViT-Base network to process these features in a *Mono/Multi Cue Combiner* network, which produces a sequence of tokens for the decoder to transform into a depth prediction.

To effectively achieve this we need to i) convert the cost volume into a token sequence without sacrificing information and ii) incorporate the monocular cues to help in decoding sharp depth. For i), a naive approach would be to apply a strided convolution projecting to the ViT token dimensions, resembling how RGB images are patchified. However this is suboptimal, for it lacks contextual information on how to achieve this downsampling. Instead, we propose a **cost volume patchifier** module. This guides the downsampling process with information from the first two blocks of the

Name	Scenes	#total scenes	# total images	# training tuples	Metric poses?	Moving objects?
Hypersim [61]	Indoor	461	77K	45K	Yes	No
TartanAIR [83]	Indoor, Outdoor	30	1M	92K	Yes	Yes
BlendedMVG [97]	Indoor, Outdoor, Aerial	389	110K	97K	No	No
MatrixCity [45]	Outdoor, Aerial	1	519K	40K	Yes	No
VKITTI2 [5, 23]	Outdoor	5	21K	40K	Yes	Yes
Dynamic Replica [38]	Indoor	484	145K	70K	Yes	Yes
MVSSynth [34]	Outdoor	117	12K	3K	No	Yes
SAIL-VOS 3D [32, 33]	Indoor, Outdoor	6807	237K	21K	Yes	Yes

Table 1. We train on eight MVS datasets from a variety of domains. All these datasets are synthetically rendered, giving them perfect ground truth depths and camera calibration. However, BlendedMVG uses real textures on their assets.

reference image encoder. We convert the cost volume into tokens using two strided convolutions, but first, concatenate each of them with the monocular features of the first two blocks, transposed and projected at 1/4 and 1/8 of the input resolution, respectively. The output of this module is a sequence of $\frac{H}{16} \times \frac{W}{16}$ tokens, matching the sequence length of the monocular features. These tokens are then fed into a ViT-B initialized with DINOv2 weights (see Fig. 4).

For ii) we add the tokens from the cost volume with the ones from the reference image encoder after projecting the latter with a linear layer. We repeat this process at blocks 2, 5, 9, and 11 of the ViT to incorporate multiple levels of monocular cues. This simple mechanism allows the network to refine and regularize the cost volume with the help of the reference image structure.

3.4. Generalizing to Any Range of Depths

When building a cost volume, a set of depth hypotheses (*i.e.* bins) $\mathcal{D}$ are used to warp feature maps $\mathcal{F}_i$ to I_r. This raises the question: How do we choose $\mathcal{D}$ to **generalize to any range of depths**? Depth ranges vary hugely across datasets (see Fig. 2), so using the same fixed range is suboptimal.

We address this with a cascaded cost volume approach, first introduced in 3D stereo matching [25, 51, 105]. While these works start from a known 'ground truth' depth range, we use the known intrinsics and extrinsics to infer the minimum and maximum depths that could be matched between I_r and each I_i. We space our initial depth bins logarithmically within this range, then make an initial depth prediction. The min and max values of this initial estimate are then used to rebuild the cost volume for a final depth prediction. This iterative process occurs only at test time; during training, we use the known depth range. Full details are in the supplementary. Importantly, previous methods that are provided with an exact depth range learn to predict depths that cover all the depth hypotheses. Thus, when using a rough estimate of the range, these methods fail to align the prediction to the actual valid depths. To further mitigate this issue, we augment the ground truth ranges via a random perturbation during training.

3.5. Implementation Details

Losses. We use the supervised losses from [62]. These comprise an L1 loss between the log of the ground truth and the log of the predicted depth values, and a gradient and normals loss. Training losses are applied to four output scales of the decoder. At inference, only the final largest-scale prediction is used. We take as input 640 × 480 images, and output depth maps at the same resolution. We use 64 depth bins in $\mathcal{D}$ sampled in log space.

Keyframes. For datasets with dense sequences, we choose reference and source frames with the strategy of [18, 62]. To be robust to sparser sets of frames, we also select tuples based on geometry overlap, obtaining tuples of not necessarily consecutive frames. Full details on our architecture and training strategy are provided in the supplementary.

Training data. For MVSA to **generalize across domains**, we train on a large and diverse set of synthetic datasets, as listed in Tab. 1. A subset of these training datasets contain moving objects. Our reference image encoder is initialized from Depth Anything V2 (DAV2) [94], which uses a teacher network trained on synthetic datasets similar to ours, and a student network distilled using various real images that do not overlap with our evaluation benchmarks. DAV2 was initialized from a pretrained DINOv2 [54] network, in turn trained on internet images.

4. Experiments

We evaluate MVSA on both depth estimation and 3D reconstruction tasks. We also implement and report scores for a set of new baselines, which have never before been evaluated on the benchmarks we use.

4.1. Baselines

Where possible, we obtain results directly from prior works [67, 82]. We also evaluate and implement other strong baselines that did not previously report performance on diverse MVS benchmarks. These include: (i) A strong monocular baseline in the form of DAV2 [94]. To account for the unknown affine transform, we align its predictions to the ground truth using least squares. (ii) MAST3R [43] (raw depth estimate) which involves passing the reference and one other source image as input and taking the z component of the point cloud as the depth prediction. (iii) MAST3R (plus our triangulation) which is a novel extension of MAST3R so that it can use provided extrinsics and intrinsics, when available. For each of the available source images, we use MAST3R descriptors to match points with the reference image. We then triangulate points from such matches, rescale the raw depth predictions, and aggregate the point clouds from the different views using a sum weighted by the predicted confidences. Note, this method requires one forward pass and thousands of triangulations

Approach	GT Poses	GT Range	Align	KITTI rel ↓	KITTI τ ↑	ScanNet rel ↓	ScanNet τ ↑	ETH3D rel ↓	ETH3D τ ↑	DTU rel ↓	DTU τ ↑	T&T rel ↓	T&T τ ↑	Average rel ↓	Average τ ↑	Average time [s] ↓
a) Depth from frames (w/o poses)																
DeMoN [75]	✗	✗	‖t‖	15.5	15.2	**12.0**	**21.0**	17.4	15.4	21.8	16.6	13.0	23.2	16.0	18.3	0.08
DeepV2D KITTI [73]	✗	✗	med	**(3.1)**	**(74.9)**	23.7	11.1	27.1	10.1	24.8	8.1	34.1	9.1	22.6	22.7	2.07
DeepV2D ScanNet [73]	✗	✗	med	10.0	36.2	(4.4)	(54.8)	11.8	29.3	7.7	33.0	**8.9**	**46.4**	8.6	39.9	3.57
MAST3R [43] (raw output)	✗	✗	med	**3.3**	**67.7**	**(4.3)**	**(64.0)**	**2.7**	**79.0**	**3.5**	**66.7**	**(2.4)**	**(81.6)**	**3.3**	**71.8**	**0.07**
MAST3R [43] (raw output)	✗	✗	✗	61.4	0.4	(12.8)	(19.4)	43.8	3.1	145.8	0.5	(66.9)	(0.0)	66.1	4.7	**0.07**
b) Depth from frames and poses (with per-image range provided)																
MVSNet [96]	✓	✓	✗	22.7	36.1	24.6	20.4	35.4	31.4	(1.8)	(86.0)	8.3	73.0	18.6	49.4	**0.07**
MVSNet Inv. Depth [96]	✓	✓	✗	18.6	30.7	22.7	20.9	21.6	35.6	(1.8)	(86.7)	6.5	74.6	14.2	49.7	0.32
Vis-MVSNet [103]	✓	✓	✗	9.5	55.4	8.9	33.5	10.8	43.3	(1.8)	(87.4)	4.1	87.2	7.0	61.4	0.70
PatchmatchNet [76]	✓	✓	✗	10.8	45.8	8.5	35.3	19.1	34.8	(2.1)	(82.8)	4.8	82.9	9.1	56.3	0.28
MVSFormer++ DTU+BlendedMVG [7]	✓	✓	✗	4.4	65.7	7.9	39.4	7.8	50.4	**(0.9)**	**(95.3)**	3.2	88.1	4.8	67.8	0.78
MVSFormer++ CE our data [7]	✓	✓	✗	4.4	63.9	6.4	43.3	6.7	56.4	(1.2)	(90.7)	2.6	88.3	4.3	68.5	0.78
MVSFormer++ regression our data [7]	✓	✓	✗	**3.7**	**67.2**	**5.7**	**45.1**	**5.3**	**57.5**	(1.2)	(90.5)	**2.2**	**88.6**	**3.6**	**69.8**	0.78
c) Single-view depth																
Depth Pro [3] †	✗	✗	med	6.1	39.6	(4.3)	(58.4)	6.1	53.5	5.6	49.6	5.6	57.5	5.6	51.7	5.16
Depth Pro [3] †	✗	✗	✗	13.6	14.3	9.2	19.7	28.5	8.7	161.8	3.5	38.3	4.4	50.3	10.1	5.16
Metric3D [30] †	✗	✗	med	5.1	44.1	**2.4**	**78.3**	4.4	54.5	10.1	39.5	6.2	48.0	5.6	52.9	0.46
Metric3D [30] †	✗	✗	✗	8.7	13.2	6.2	19.3	12.7	13.0	890.5	1.4	16.7	13.7	187.0	12.1	0.46
UniDepthV2 [56] †	✗	✗	med	**4.0**	**55.3**	(2.1)	(82.6)	**3.7**	**66.2**	3.2	72.3	**3.6**	**68.4**	**3.3**	**68.9**	0.29
UniDepthV2 [56] †	✗	✗	✗	13.7	4.8	(3.2)	(61.3)	15.4	11.9	964.8	1.3	16.7	12.7	202.7	18.4	0.29
UniDepthV1 [56] †	✗	✗	med	4.4	51.6	**(1.9)**	**(84.3)**	5.4	48.4	9.3	31.8	9.6	38.7	6.1	51.0	0.21
UniDepthV1 [56] †	✗	✗	✗	5.2	39.5	(2.7)	(69.4)	48.2	1.8	583.3	1.0	30.7	4.2	134.0	23.2	0.20
DepthAnything V2 (ViT-B) [94]	✗	✗	lstsq †	6.6	38.6	4.0	58.6	4.7	56.5	**2.6**	**74.7**	4.5	57.5	4.8	54.1	**0.05**
d) Depth from frames and poses (w/o per-image range)																
Fast-MVSNet [99]	✓	✗	✗	12.1	37.4	287.1	9.4	131.2	9.6	(540.4)	(1.9)	33.9	47.2	200.9	21.1	0.35
MVS2D ScanNet [95]	✓	✗	✗	73.4	0.0	(4.5)	(54.1)	30.7	14.4	5.0	57.9	56.4	11.1	34.0	27.5	**0.05**
MVS2D DTU [95]	✓	✗	✗	93.3	0.0	51.5	1.6	78.0	0.0	**(1.6)**	(92.3)	87.5	0.0	62.4	18.8	0.06
Robust MVD Baseline [67]	✓	✗	✗	7.1	41.9	7.4	38.4	9.0	42.6	2.7	82.0	5.0	75.1	6.3	56.0	0.06
MAST3R (plus our triangulation)	✓	✗	✗	3.4	66.6	(4.5)	**(63.0)**	**3.1**	**72.9**	3.4	67.3	(2.4)	(83.3)	3.4	70.1	0.72
MVSA (Ours)	✓	✗	✗	**3.2**	**68.8**	**3.7**	**62.9**	3.2	68.0	**1.3**	**95.0**	**2.1**	**90.5**	**2.7**	**77.0**	0.12

Table 2. **We set a new SOTA in depth estimation on the RMVDB**. See Sec. 4 for details of the metrics, baselines and groupings. A full version of this table, including older baselines, appears in the supplementary. Monocular methods with † are given ground truth intrinsics. The best result for each section appears in **bold**, and (parentheses) indicate results where the evaluation dataset is in the training set.

per source view, significantly reducing its speed. MAST3R trains on ScanNet [13] and MegaDepth [46] (which contains a subset of the Tanks and Temples dataset [42]). Baseline implementation details are in the supplementary.

Benchmark. We evaluate 'zero-shot' depth estimation performance on the five multi-view datasets from the RMVDB benchmark [67], which are not included in our training data. It contains the KITTI [24] ScanNet [13], ETH3D [66], DTU [37], and Tanks and Temples [42] datasets and represents a diverse set of evaluation scenarios, *e.g.* driving sequences, room scans, building scans, and tabletop objects, among others. We use the evaluation procedure and source view selection procedure from [67], allowing direct comparison to previous approaches.

Methods are grouped into four different types (a-d) depending on the information they are provided, *e.g.* if they are given GT cameras, GT depth ranges, *etc.* MVSA naturally fits into type (d), where all methods are given GT poses, so need to predict depth directly in metric scale and hence do not need any alignment or knowledge of the GT depth range. Note, some methods train on the training splits of one, or more, of the benchmark datasets, thus achieving very high scores in those cases. We denote these in Tab. 2 with a parenthesis around them.

Metrics. We report two commonly used metrics to compare the predicted $\hat{d}$ and GT depth d. The absolute relative depth (rel) is computed per-pixel as $|\hat{d} - d|/d$, while the inlier percentage τ, with threshold 1.03, is computed per-pixel as $[\max(\hat{d}/d, d/\hat{d}) < 1.03]$, where $[]$ is the Iverson bracket. Both metrics are averaged over all valid GT pixels in each test image, before averaging over all images.

Results. Tab. 2 depicts the quantitative results, where we outputperform all baselines across most metrics. Qualitative results in Fig. 7 demonstrate that our MVSA model produces depth maps with superior edge detail and consistent scaling across a variety of scenes, visually outperforming prior methods. MVSA also performs well on moving objects, *e.g.* as found in KITTI; see also Fig. 6. Both **MAST3R triangulated** and the **Robust MVD Baseline** exhibit poor edge quality, limiting their suitability for applications such as single-image novel view synthesis [69], which requires sharp depth boundaries. While Depth Pro produces

Approach	ScanNet		ETH3D	
	rel ↓	τ ↑	rel ↓	τ ↑
Robust MVD Baseline [67]	6.02	47.83	5.75	71.64
MAST3R Triangulated	(3.88)	(68.68)	2.37	84.90
MVSA (Ours)	**3.22**	**69.45**	**1.27**	**93.24**

Table 3. **Our variant of RMVDB**. We use better test-time tuples for ScanNet, and for ETH3D we use the undistorted test images.

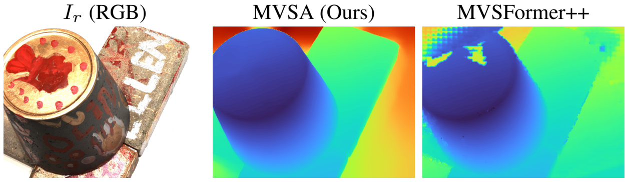


Figure 5. **Many MVS models fail in areas of poor frame overlap**. Here we show how MVSFormer++ (right) fails to recover geometry in areas of the image where there are no matching pixels between source and target views (see the top left corner). Our model (middle) handles this situation gracefully.

sharp edges, it frequently displays incorrect depth scaling. In contrast, our MVSA model combines competitive quantitative performance with sharper edges, making it ideal for tasks that demand both visual and depth accuracy. Finally, GT depth-based median and least squares scaling of monocular methods and depth from frames methods (w/o poses) is crucial for good scores, while MVSA consistently predicts high-quality and metric depths.

Alternative Variant of RMVDB. We further evaluate some of the leading models on a RMVDB variant, in which we change some conditions to better reflect real-world scenarios. In this variant, for ScanNet we use keyframes using the strategy of [18], rather than the temporally sequential keyframes provided by the benchmark. For ETH3D we undistort both the images and the ground truth using their provided Thin-Prism [87] camera parameters. Results are shown in Tab. 3. On this revised benchmark, we more comprehensively outperform the baselines.

4.2. Ablations Study

In Tab. 4 we validate our design decisions by turning on and off sections of our system. We train all ablations at a smaller resolution (512 × 384 input), and without using metadata, for efficiency. At this resolution, Row **A** is 'ours' and all other rows are ablations relative to this. Row **B** replaces our standard ViT-B with the smaller ViT Small, both for the cost volume ViT and the reference image encoder. The reference image encoder is initialized from Depth Anything v2 (small). Row **C** uses our training data and pipeline, but with the fully-convolutional architecture from SimpleRecon [62] (without metadata). Row **D** is our system but without adding noise to the ground truth range at training time (Section 3.4). Although this method can excel when the initial range is accurate, it can fail to generalize (see DTU). Row **E** is our full architecture but without the pretrained encoder weights from [94]. Instead we initialize with DINOv2 weights. Row **F** is our system without a cascaded cost volume, and instead uses a fixed set of depth bins, losing the ability to refine depth bins and work with arbitrary scales or scene sizes. Row **H** is our full model, but where we take the first depth prediction from the model as our final output, without re-building the cost volume. Even though these bins capture the full range of depths in the test datasets (Fig. 2), we see that performance degrades. Row **G** uses CNN layers instead of ViT to combine mono/multi features. Row **I** uses naive patchification to preprocess the cost volume for input to the mono/multi cue combiner ViT, as outlined in Fig. 4. These results confirm our design decisions, and full implementation details are in the supplementary.

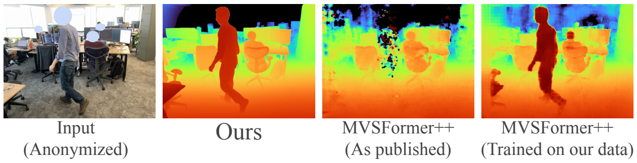


Figure 6. We handle **dynamic objects** significantly better than traditional MVS *e.g.* MVSFormer++. See supplementary for details of the MVSFormer++ variants shown here.

		KITTI		ScanNet		ETH3D		DTU		T&T		Average	
		rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑
A	Ours (no metadata, low res.)	3.39	**66.88**	3.86	60.82	**3.11**	**70.17**	2.43	92.05	**2.23**	**88.38**	**3.00**	**75.66**
B	w/ ViT Small	3.57	64.34	4.40	56.57	3.63	64.61	2.69	91.52	2.71	84.51	3.40	72.31
C	w/ [62]'s architecture	3.63	65.94	5.03	51.76	3.74	63.09	**1.77**	90.97	2.78	87.90	3.39	71.93
D	w/o noise on GT range	**3.33**	66.83	5.14	52.77	3.53	66.32	13.45	89.21	2.34	87.64	5.32	74.89
E	w/o DAV2 weights	3.45	65.42	4.58	57.14	3.48	65.59	2.14	**92.48**	2.56	86.01	3.24	73.33
F	w/ fixed bins [0-100m]	3.41	64.33	**3.80**	**61.62**	3.15	67.20	4.11	65.64	2.36	85.72	3.37	68.90
G	no MMCC ViT	3.54	65.32	4.39	56.94	3.56	65.27	3.07	90.49	2.46	87.54	3.41	73.11
H	w/o bin refinement	3.57	63.39	5.18	51.05	3.50	67.93	6.80	82.12	2.27	87.04	4.26	70.31
I	Naive patchify	3.66	62.61	4.27	58.86	3.18	67.27	1.95	91.69	2.46	86.52	3.11	73.39

Table 4. **Ablation Study.** Here we validate our design decisions on RMVDB [67] by ablating various components. See Sec. 4.2 for details. **First** and second best scores are indicated.

Robustness to pose rescaling. In the supplementary we include results where we scale ScanNet poses ×100, and show our depths are robust to this scaling (*vs.* ours w/o normalization of the metadata, which performs badly).

4.3. Meshing and 3D Reconstruction

To judge the 3D-consistency of our predictions we evaluate our model on ScanNet Mesh Evaluation benchmark using the protocol defined in [4] which also uses source frame selection from [18]. The benchmark uses a GT mesh collected with an active RGBD sensor captured in a video. The evaluation computes point-to-point vertex error from GT to predicted (as accuracy), from predicted to GT (as completion), and the average of the two (as chamfer). Additionally, 200k points are sampled uniformly over each mesh and point-to-point errors thresholded at 5cm distance are used to compute precision, recall and F-score. Almost all competing methods are trained on ScanNet, however our method that was not trained on ScanNet performs comparatively, outperforming many of the methods in Table 5. See Fig. 1 and the supplementary for qualitative meshing results.

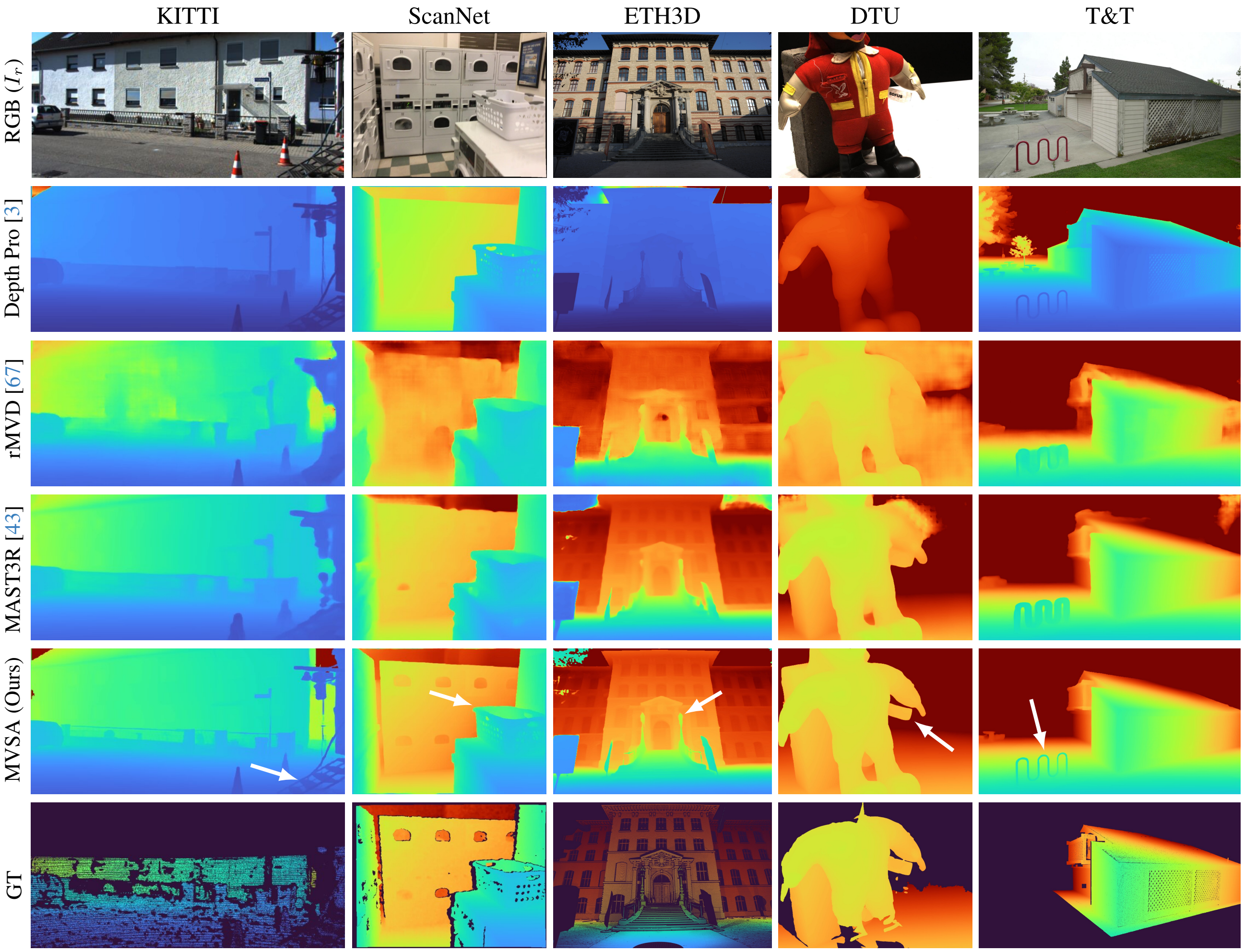


Figure 7. **Qualitative comparison of depth prediction results across multiple datasets** (KITTI, ScanNet, ETH3D, DTU, and Tanks & Temples). Rows show different methods: Depth Pro [67], rMVD baseline [67], MAST3R (Triangulated) [43], and our MVSA model, along with RGB inputs (I_r) and ground-truth depths (GT). Depth Pro provides sharp edges but often misestimates depth scale, while our MVSA model captures finer details than MAST3R and rMVD. Depth maps are normalized to ground truth depth range for consistent visualization; see the supplementary material for unnormalized results, especially highlighting scale discrepancies in Depth Pro.

		Comp↓	Acc↓	Chamfer↓	Prec↑	Recall ↑	F-Score ↑
DeepVideoMVS [18]	**S**	10.68	6.90	8.79	0.541	0.592	0.563
ATLAS [53]	**S**	7.16	7.61	7.38	0.675	0.605	0.636
NeuralRecon [72]	**S**	5.09	9.13	7.11	0.630	0.612	0.619
3DVNet [60]	**S**	7.72	6.73	7.22	0.655	0.596	0.621
TransformerFusion [4]	**S**	5.52	8.27	6.89	0.728	0.600	0.655
VoRTX [71]	**S**	**4.31**	7.23	**5.77**	**0.767**	0.651	**0.703**
SimpleRecon [62]	**S**	5.53	**6.09**	5.81	0.686	**0.658**	0.671
COLMAP [65]		10.22	11.88	11.05	0.509	0.474	0.489
MAST3R [43] (raw depth)	**S+**	12.35	12.69	12.52	0.265	0.283	0.272
MAST3R [43] (+ triangulation)	**S+**	5.38	6.78	6.08	0.572	0.655	0.608
SimpleRecon [62] (trained on our data)		8.07	6.67	7.37	0.501	0.597	0.544
MVSA (Ours)		**4.93**	**6.39**	**5.66**	**0.616**	**0.696**	**0.652**

Table 5. **ScanNet Mesh Evaluation [4]**. Scores adapted from [4, 62]. Rows marked with **S** were trained on ScanNet only, while those marked **S+** were trained on ScanNet and other datasets. Our MVSA model, which was not trained on ScanNet, outperforms many models which were, *e.g.* [53, 60, 72].

Limitations. While we use multi-view information to generate depths, we do not enforce or encourage temporal consistency. Techniques for this [31, 49, 90] could work with MVSA. Also, like traditional MVS, our method requires known camera intrinsics and poses; recent works suggest this requirement could be relaxed [52, 78].

5. Conclusions

We introduced MVSAnywhere, a new general-purpose MVS depth estimation approach. We addressed challenges associated with training on diverse MVS datasets, such as how to best leverage ViT-based architectures, how to incorporate geometric metadata, and how to handle variable depth ranges. Through extensive experimentation, we compare to numerous existing and new baselines. Our contributions result in state-of-the-art zero-shot performance on a range of challenging reconstruction and depth estimation test datasets, in some cases even outperforming models trained on the test domains.

Acknowledgements

We are extremely grateful to Saki Shinoda, Jakub Powierza, and Stanimir Vichev for their invaluable compute infrastructure support. S Izquierdo and J Civera are funded by the Spanish projects PID2021-127685NB-I00 and TED2021-131150B-I00, Aragón project T45_23R, and the scholarship FPU20/02342.

References

[1] Luca Bartolomei, Fabio Tosi, Matteo Poggi, and Stefano Mattoccia. Stereo anywhere: Robust zero-shot deep stereo matching even where either stereo or mono fail. In *CVPR*, 2025. 3

[2] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. ZoeDepth: Zero-shot transfer by combining relative and metric depth. *arXiv:2302.12288*, 2023. 1, 3

[3] Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second. In *ICLR*, 2025. 1, 3, 4, 6, 8, 13, 17, 20, 21, 22, 23, 24, 25, 26, 27, 28

[4] Aljaz Bozic, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. TransformerFusion: Monocular RGB scene reconstruction using transformers. *NeurIPS*, 2021. 7, 8

[5] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. *arXiv:2001.10773*, 2020. 5

[6] Chenjie Cao, Xinlin Ren, and Yanwei Fu. MVSFormer: Multi-view stereo by learning robust image features and temperature-based depth. *TMLR*, 2023. 1, 2

[7] Chenjie Cao, Xinlin Ren, and Yanwei Fu. MVSFormer++: Revealing the devil in transformer's details for multi-view stereo. In *ICLR*, 2024. 2, 3, 6, 15, 16, 17

[8] Hanlin Chen, Fangyin Wei, Chen Li, Tianxin Huang, Yunsong Wang, and Gim Hee Lee. Vcr-gaus: View consistent depth-normal regularizer for gaussian surface reconstruction. *Advances in Neural Information Processing Systems*, 37:139725–139750, 2024. 15

[9] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *NeurIPS*, 2016. 3

[10] Shuo Cheng, Zexiang Xu, Shilin Zhu, Zhuwen Li, Li Erran Li, Ravi Ramamoorthi, and Hao Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *CVPR*, 2020. 3

[11] Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation network. *PAMI*, 2019. 2

[12] Andrea Conti, Matteo Poggi, Valerio Cambareri, and Stefano Mattoccia. Range-agnostic multi-view depth estimation with keyframe selection. In *3DV*, 2024. 3

[13] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017. 2, 6, 13

[14] Yuchao Dai, Zhidong Zhu, Zhibo Rao, and Bo Li. Mvs2: Deep unsupervised multi-view stereo with multi-view symmetry. In *3DV*, 2019. 2

[15] Duolikun Danier, Mehmet Aygün, Changjian Li, Hakan Bilen, and Oisin Mac Aodha. DepthCues: Evaluating monocular depth perception in large vision models. In *CVPR*, 2025. 2

[16] Yikang Ding, Wentao Yuan, Qingtian Zhu, Haotian Zhang, Xiangyue Liu, Yuanjiang Wang, and Xiao Liu. Transmvsnet: Global context-aware multi-view stereo network with transformers. In *CVPR*, 2022. 2

[17] 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. In *ICLR*, 2021. 2, 3

[18] Arda Duzceker, Silvano Galliani, Christoph Vogel, Pablo Speciale, Mihai Dusmanu, and Marc Pollefeys. DeepVideoMVS: Multi-view stereo on video with recurrent spatio-temporal fusion. In *CVPR*, 2021. 2, 5, 7, 8, 14

[19] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 3

[20] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014. 14

[21] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018. 3

[22] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2015. 2

[23] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016. 5

[24] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012. 2, 6

[25] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, 2020. 2, 3, 5

[26] Vitor Guizilini, Pavel Tokmakov, Achal Dave, and Rares Ambrus. GRIN: Zero-shot metric depth with pixel-level diffusion. *arXiv:2409.09896*, 2024. 1

[27] Xianda Guo, Chenming Zhang, Youmin Zhang, Dujun Nie, Ruilin Wang, Wenzhao Zheng, Matteo Poggi, and Long Chen. Stereo anything: Unifying stereo matching with large-scale mixed data. *arXiv:2411.14053*, 2024. 2

[28] Jing He, Haodong Li, Wei Yin, Yixun Liang, Leheng Li, Kaiqiang Zhou, Hongbo Liu, Bingbing Liu, and Ying-Cong Chen. LOTUS: Diffusion-based visual foundation model for high-quality dense prediction. In *ICLR*, 2025. 3

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[30] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3D v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *PAMI*, 2024. 1, 3, 6, 13, 15, 17

[31] Wenbo Hu, Xiangjun Gao, Xiaoyu Li, Sijie Zhao, Xiaodong Cun, Yong Zhang, Long Quan, and Ying Shan. DepthCrafter: Generating consistent long depth sequences for open-world videos. In *CVPR*, 2025. 2, 8

[32] Y.-T. Hu, H.-S. Chen, K. Hui, J.-B. Huang, and A. G. Schwing. SAIL-VOS: Semantic Amodal Instance Level Video Object Segmentation – A Synthetic Dataset and Baselines. In *CVPR*, 2019. 5

[33] Y.-T. Hu, J. Wang, R. A. Yeh, and A. G. Schwing. SAIL-VOS 3D: A Synthetic Dataset and Baselines for Object Detection and 3D Mesh Reconstruction from Video Data. In *CVPR*, 2021. 5

[34] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *CVPR*, 2018. 2, 3, 5

[35] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. DPSNet: End-to-end deep plane sweep stereo. *ICLR*, 2019. 2

[36] Sergio Izquierdo and Javier Civera. SfM-TTR: Using structure from motion for test-time refinement of single-view depth networks. In *CVPR*, 2023. 2

[37] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 2, 6

[38] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. DynamicStereo: Consistent dynamic depth from stereo videos. *CVPR*, 2023. 5

[39] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *CVPR*, 2024. 1, 3

[40] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017. 2, 3

[41] Tejas Khot, Shubham Agrawal, Shubham Tulsiani, Christoph Mertz, Simon Lucey, and Martial Hebert. Learning unsupervised multi-view stereopsis via robust photometric consistency. In *CVPR Workshops*, 2019. 2

[42] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *Transactions on Graphics*, 2017. 2, 6

[43] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3D with MASt3R. In *ECCV*, 2024. 1, 2, 5, 6, 8, 13, 17, 20, 21, 22, 23, 24, 25, 26, 27, 28

[44] Jingliang Li, Zhengda Lu, Yiqun Wang, Jun Xiao, and Ying Wang. Nr-mvsnet: Learning multi-view stereo based on normal consistency and depth refinement. *TIP*, 2023. 3

[45] Yixuan Li, Lihan Jiang, Linning Xu, Yuanbo Xiangli, Zhenzhi Wang, Dahua Lin, and Bo Dai. MatrixCity: A large-scale city dataset for city-scale neural rendering and beyond. In *ICCV*, 2023. 5

[46] Zhengqi Li and Noah Snavely. MegaDepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 3, 6, 13, 14

[47] Xiaoxiao Long, Lingjie Liu, Christian Theobalt, and Wenping Wang. Occlusion-aware depth estimation with adaptive normal constraints. In *ECCV*, 2020. 2

[48] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Seminal graphics: pioneering efforts that shaped the field*, 1998. 14

[49] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *Transactions on Graphics*, 2020. 2, 8

[50] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 2

[51] Zhenxing Mi, Chang Di, and Dan Xu. Generalized binary search network for highly-efficient multi-view stereo. In *CVPR*, 2022. 3, 5

[52] Riku Murai, Eric Dexheimer, and Andrew J. Davison. MASt3R-SLAM: Real-time dense SLAM with 3D reconstruction priors. In *CVPR*, 2025. 8

[53] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3D scene reconstruction from posed images. In *ECCV*, 2020. 8

[54] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *TMLR*, 2024. 3, 5

[55] Vaishakh Patil, Wouter Van Gansbeke, Dengxin Dai, and Luc Van Gool. Don't forget the past: Recurrent depth estimation from monocular video. *Robotics and Automation Letters*, 2020. 2

[56] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. UniDepth: Universal monocular metric depth estimation. In *CVPR*, 2024. 1, 6, 13, 17

[57] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3D dataset (HM3D): 1000 large-scale 3D environments for embodied AI. In *NeurIPS Datasets and Benchmarks Track*, 2021. 13

[58] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 4, 13

[59] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *PAMI*, 2022. 1, 3, 14

[60] Alexander Rich, Noah Stier, Pradeep Sen, and Tobias Höllerer. 3DVNet: Multi-view depth prediction and volumetric refinement. In *3DV*, 2021. 8

[61] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021. 5

[62] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. SimpleRecon: 3D reconstruction without 3D convolutions. In *ECCV*, 2022. 2, 3, 4, 5, 7, 8, 13, 14, 15, 18

[63] Mohamed Sayed, Filippo Aleotti, Jamie Watson, Zawar Qureshi, Guillermo Garcia-Hernando, Gabriel Brostow, Sara Vicente, and Michael Firman. Doubletake: Geometry guided depth estimation. In *ECCV*, 2024. 2

[64] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, 2016. 17

[65] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 2, 8, 17

[66] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, 2017. 2, 6

[67] Philipp Schröppel, Jan Bechtold, Artemij Amiranashvili, and Thomas Brox. A benchmark and a baseline for robust multi-view depth estimation. In *3DV*, 2022. 2, 5, 6, 7, 8, 13, 15, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28

[68] Jiahao Shao, Yuanbo Yang, Hongyu Zhou, Youmin Zhang, Yujun Shen, Matteo Poggi, and Yiyi Liao. Learning temporally consistent video depth from video diffusion priors. In *CVPR*, 2025. 2

[69] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3D photography using context-aware layered depth inpainting. In *CVPR*, 2020. 6

[70] Nikolai Smolyanskiy, Alexey Kamenev, and Stan Birchfield. On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach. In *CVPR*, 2018. 2

[71] Noah Stier, Alexander Rich, Pradeep Sen, and Tobias Höllerer. VoRTX: Volumetric 3D reconstruction with transformers for voxelwise view selection and fusion. In *3DV*, 2021. 8

[72] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. In *CVPR*, 2021. 8

[73] Zachary Teed and Jia Deng. DeepV2D: Video to depth with differentiable structure from motion. In *ICLR*, 2020. 2, 6, 17

[74] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. Dn-splatter: Depth and normal priors for gaussian splatting and meshing. *arXiv preprint arXiv:2403.17822*, 2024. 15

[75] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and motion network for learning monocular stereo. In *CVPR*, 2017. 6, 17

[76] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. PatchmatchNet: Learned multi-view patchmatch stereo. In *CVPR*, 2021. 2, 6, 17

[77] Fangjinhua Wang, Qingtian Zhu, Di Chang, Quankai Gao, Junlin Han, Tong Zhang, Richard Hartley, and Marc Pollefeys. Learning-based multi-view stereo: A survey. *arXiv:2408.15235*, 2024. 2

[78] Hengyi Wang and Lourdes Agapito. 3D reconstruction with spatial memory. In *3DV*, 2025. 8

[79] Kaixuan Wang and Shaojie Shen. MVDepthNet: Real-time multiview depth estimation neural network. In *3DV*, 2018. 2, 3

[80] Rui Wang, Stephen M Pizer, and Jan-Michael Frahm. Recurrent neural network for (un-)supervised learning of monocular video visual odometry and depth. In *CVPR*, 2019. 2

[81] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. MoGe: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. *arXiv:2410.19115*, 2024. 1, 3

[82] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, 2024. 2, 5, 13

[83] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. TartanAir: A dataset to push the limits of visual SLAM. In *IROS*, 2020. 5

[84] Xiaofeng Wang, Zheng Zhu, Guan Huang, Fangbo Qin, Yun Ye, Yijia He, Xu Chi, and Xingang Wang. MVSTER: Epipolar transformer for efficient multi-view stereo. In *ECCV*, 2022. 3

[85] Jamie Watson, Oisin Mac Aodha, Victor Prisacariu, Gabriel Brostow, and Michael Firman. The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth. In *CVPR*, 2021. 2, 3

[86] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. FoundationStereo: Zero-shot stereo matching. In *CVPR*, 2025. 2

[87] Juyang Weng, Paul Cohen, Marc Herniou, et al. Camera calibration with distortion models and accuracy evaluation. *PAMI*, 1992. 7

[88] Felix Wimbauer, Nan Yang, Lukas Von Stumberg, Niclas Zeller, and Daniel Cremers. MonoRec: Semi-supervised dense reconstruction in dynamic environments from a single moving camera. In *CVPR*, 2021. 2

[89] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting gaussian splatting and depth. In *CVPR*, 2025. 3

[90] Honghui Yang, Di Huang, Wei Yin, Chunhua Shen, Haifeng Liu, Xiaofei He, Binbin Lin, Wanli Ouyang, and Tong He. Depth any video with scalable synthetic data. In *ICLR*, 2025. 2, 8

[91] Jiayu Yang, Wei Mao, Jose M. Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *CVPR*, 2020. 17

[92] Jiayu Yang, Jose M Alvarez, and Miaomiao Liu. Self-supervised learning of depth inference for multi-view stereo. In *CVPR*, 2021. 2

[93] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 1, 3

[94] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything V2. In *NeurIPS*, 2024. 1, 3, 4, 5, 6, 7, 13, 17

[95] Zhenpei Yang, Zhile Ren, Qi Shan, and Qixing Huang. MVS2D: Efficient multi-view stereo via attention-driven 2D convolutions. In *CVPR*, 2022. 1, 2, 6, 17

[96] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018. 1, 2, 6, 17

[97] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A large-scale dataset for generalized multi-view stereo networks. In *CVPR*, 2020. 5

[98] Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3D: Towards zero-shot metric 3D prediction from a single image. In *ICCV*, 2023. 1, 3, 13

[99] Zehao Yu and Shenghua Gao. Fast-MVSNet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *CVPR*, 2020. 2, 6, 17

[100] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, 2015. 2

[101] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. GA-Net: Guided aggregation net for end-to-end stereo matching. In *CVPR*, 2019. 2

[102] Feihu Zhang, Xiaojuan Qi, Ruigang Yang, Victor Prisacariu, Benjamin Wah, and Philip Torr. Domain-invariant stereo matching networks. In *ECCV*, 2020. 2

[103] Jingyang Zhang, Yao Yao, Shiwei Li, Zixin Luo, and Tian Fang. Visibility-aware multi-view stereo network. In *BMVC*, 2020. 2, 6, 17

[104] Jiawei Zhang, Jiahe Li, Lei Huang, Xiaohan Yu, Lin Gu, Jin Zheng, and Xiao Bai. Robust synthetic-to-real transfer for stereo matching. In *CVPR*, 2024. 2

[105] Song Zhang, Wenjia Xu, Zhiwei Wei, Lili Zhang, Yang Wang, and Junyi Liu. ARAI-MVSNet: A multi-view stereo depth estimation network with adaptive depth range and depth interval. *Pattern Recognition*, 2023. 3, 5

[106] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. DeepTAM: Deep tracking and mapping. In *ECCV*, 2018. 17

[107] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 14

Appendix

A. Additional baseline details

Most of the scores in Tab. 2 in the main paper (and Table A5) are directly taken from prior works [67, 82]. For completeness, we additionally evaluate other works which did not evaluate on RMVDB. These were described briefly in the main paper, but here we provide full details.

Single-view baselines. First we include general-purpose monocular depth models via the publicly available checkpoints, using the reference frame as input (*i.e.* no multi-view signal). We include monocular models which predict metric depth [3, 30, 56, 98], both with and without median scaling to align the predictions to the GT. We use the publically available code and pretrained models for these baselines. For each of them, we use the mode which allows input of known intrinsics, and we provide the intrinsics given with each of the test datasets. We additionally include Depth Anything V2 [94], which is a large ViT-based model which predicts depths up to an unknown affine transform. To account for the unknown affine transform, we align the predictions to the GT using least squares. Without median scaling the performance of many of these monocular models is poor, even though they are trained to predict in metric space. Even with median scaling, they rarely outperform our model, which benefits from multi-view information.

MAST3R [43] (raw depth estimate). MAST3R takes a pair of images as input and predicts a metric-scaled point cloud in the reference frame of the first image. The simplest method to extract depth is to input I_r, and one other frame, and take the z component of the point cloud as the depth prediction. When selecting the source frame to use, we use the 'best' single source frame as provided by the benchmark. We include this baseline as *MAST3R (raw output)* in (b). We clarify here that MAST3R is trained on the Habitat dataset [57], which itself contains ScanNet [13]. So **MAST3R trains on ScanNet**. They also train on MegaDepth [46], which contains a subset of the Tanks and Temples dataset. Their scores are therefore in parentheses for these datasets.

MAST3R [43] (plus our triangulation). The provided version of MAST3R is not able to make use of camera poses and extrinsics. For an additional comparison with MAST3R, we extend their method so that it can use provided extrinsics and intrinsics, when available, to boost their results. For each of the available source images, we use MAST3R descriptors to match points with the reference frame. Then we use these matches and the known camera extrinsics and intrinsics to triangulate points, which results in a sparse depth map from the viewpoint of the reference image. We then use these sparse depths to rescale the MAST3R raw depth predictions into a more 'correct' range. Then, following DUST3R [82], we aggregate the point clouds X^i from the different views, i, using a weighted sum and the predicted confidences, C^i

$$D = \frac{\sum_{i=0}^{N} X_z^i C^i}{\sum_{i=0}^{N} C^i}$$

Note, that this method requires one forward pass and thousands of triangulations per each source view, making it slow. We denote this as *MAST3R (plus our triangulation)* in part (d) of the results table.

B. Additional model and training details

B.1. Model architecture

Feature extractor. We use the first two blocks of a pretrained ResNet-18 network and apply an instance normalization to the features, following [62].

Reference image encoder. We use a ViT-B initialized from the Depth Anything V2 relative depth weights.

Mono/Multi Cue Combiner. For the cost volume patchifier we use three residual blocks with stride 1, 2, and 2 that progressively downsample the cost volume to 1/16 of the image resolution. After the first and the second of these blocks we concatenate the features with the output of the first two blocks of the reference image encoder, respectively. These outputs are first upsampled using transposed convolutions with kernel and stride size 4 and 2, and projected into the desired number of channels using a linear layer. The concatenated features are processed by two residual blocks. Each of the resulting pixels at 1/16 resolution after the cost volume patchifier is a token which is then processed by a ViT-B initialized from DINOv2 weights. After blocks 2, 5, 9, and 11, we add these with their respective blocks from the reference image encoder. We linearly project the reference features before adding them.

Depth decoder. Based on the decoder from [58], it reassembles the tokens into feature maps, and it fuses these to obtain fine-grained predictions. We adapt this module to produce outputs at full, 1/2, 1/4 and 1/8 resolutions.

B.2. Metadata details

In our full model we use 'metadata' as components of the cost volume, following [62]. For completeness, here we briefly re-state what these comprise. Given pixel location u_r, v_r in the reference image, and depth bin k, we include the following items of metadata in the cost volume:

Feature dot product: The dot product of the reference image features and each warped source image features, expressed for each source image i as $\mathcal{F}^r \cdot \langle \mathcal{F} \rangle^i$, where $\langle\rangle$ is the warping operation which warps features from the source image to the reference viewpoint at the depth corresponding to bin k. This value is often used as the sole matching affinity in cost volumes.

Ray directions $\mathbf{r}^0_{k,u_r,v_r}$ and $\mathbf{r}^i_{k,u_r,v_r} \in \mathbb{R}^3$: This is the normalized directions pointing from the camera origins to the 3D location of a point (k, u_r, v_r) in the plane sweep cost volume. We create rays for the reference and all the source images.

Reference plane depth z^r_{k,u_r,v_r}: This is the depth of the point at position (k, u_r, v_r) in the cost volume, measured perpendicularly from the reference camera. We normalize these values with the min and max depth.

Reprojected depths z^i_{k,u_r,v_r}: This is the perpendicular depth of the 3D point at position (k, u_r, v_r) in the cost volume, relative to the source camera n. We normalize these values with the min and max depth.

Relative ray angles $\theta^{r,i}$: This is the angle between the ray directions $\mathbf{r}^r_{k,u_r,v_r}$ and $\mathbf{r}^i_{k,u_i,v_i}$.

Relative pose distance $p^{r,i}$: This is the relative pose distance between the reference camera and a source frame, as defined in [18]:

$$p^{r,i} = \sqrt{\|\mathbf{t}^{r,i}\| + \frac{2}{3}\text{tr}(\mathbb{I} - \mathbf{R}^{r,i})}, \tag{2}$$

where $\mathbf{t}^{r,i}$ and $\mathbf{R}^{r,i}$ are the relative translation and rotation between views i and r. The translation, $\mathbf{t}^{r,i}$, is normalized by the source frame with the biggest pose distance.

Depth validity masks m^i_{k,u_r,v_r}: This is a binary mask indicating whether the point (k, u_r, v_r) in the cost volume projects in front of the source camera i or not.

B.3. Training details

We train on two A100 GPUs for a total of 145k steps, which takes approximately 2.5 days. We use a batch size of eight on each GPU for our 448×336 models, and a batch size of six for our 640×480 models. We accumulate gradients over two consecutive batches, so we only backpropogate after two batches have been seen.

We use different learning rates for different components. The matching encoder and cost volume MLPs have an initial learning rate of $1\text{e}-4$ until step 140k, and $1\text{e}-5$ afterwards. The decoder and multi/mono cue combiner have an initial learning rate of $5\text{e}-5$ which is linearly decayed during training to $5\text{e}-8$. The reference image encoder has an initial learning rate of $5\text{e}-6$ which is linearly decayed during training to $5\text{e}-9$. We use weight decay of $1e-4$.

B.4. Training losses

Log depth loss. We follow [20, 62] and supervise predictions using log-depth across all pixels. At each decoder output scale s, we apply an absolute error on the log-depth:

$$\mathcal{L}_{\text{depth}} = \frac{1}{HW} \sum_{s=1}^{4} \sum_{i,j} \frac{1}{s^2} \left| \uparrow_{gt} \log \hat{D}^{i,j}_r - \log D^{i,j} \right|, \tag{3}$$

where $\hat{D}_r$ is the network output depth map, D is the ground truth depth map and i, j superscripts indicate pixel indices. Here each decoder output is aligned with the full-size ground truth depth map via nearest-neighbor upsampling [18] (denoted by $\uparrow_{gt}$). We average this loss across pixels, scales, and batches.

Gradient loss. Following *e.g.* [46, 59, 62], we apply a multi-scale gradient loss to the highest-resolution network output. In contrast to those works, we apply the loss to $\frac{1}{D}$ rather than to D directly, to help prevent this loss from dominating the overall loss. Our gradient loss is therefore:

$$\mathcal{L}_{\text{grad}} = \frac{1}{HW} \sum_{s=1}^{4} \sum_{i,j} \left| \nabla \downarrow_s \frac{1}{\hat{D}^{i,j}_r} - \nabla \downarrow_s \frac{1}{D^{i,j}} \right|, \tag{4}$$

where ∇ computes first-order spatial gradients, and $\downarrow_s$ represents downsampling to scale s.

Normals loss. We also use a simplified normal loss.

$$\mathcal{L}_{\text{normals}} = \frac{1}{2HW} \sum_{i,j} \left(1 - \hat{N}^{i,j} \cdot N^{i,j}\right), \tag{5}$$

where N is the normal map derived from depth and camera intrinsics.

Final loss. Our final loss is

$$\mathcal{L} = \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{grad}} + \mathcal{L}_{\text{normals}}. \tag{6}$$

C. Constructing 3D meshes

In Fig. 1 and Sec. 4.4 in the main paper we convert our posed depth maps into 3D meshes for visualization and evaluation. To do this, we broadly follow the procedure from prior work *e.g.* [62]. We fuse depths into a sparse TSDF volume using Open3D [107], use a voxel size of 4cm, and avoid fusing depth values over 3.5m. We extract a final mesh from the TSDF using marching cubes [48].

D. Additional results

Tab. 2 in the main paper omitted some earlier methods for space reasons. In Tab. A5 we present a more complete version of the table, with scores from the earlier methods reinstated.

In Tab. A6 we present an extended version of the ablations from Tab. 4 from the main paper. We divide the ablations into subsections to validate the different design choices in MVSA. Except those experiments specifically ablating the metadata, all ablations are done without metadata for efficiency reasons. In Tab. A7 we report results of these ablations on the improved tuples for ScanNet and undistorted images for ETH3D, as in Tab. 3 of the main paper.

Ablating the Mono/Multi Cue Combiner. We show how both, using a CNN instead of a ViT (no MMCC ViT) and using a naive way to convert the cost volume into tokens (naive patchify) result in significantly degraded results.

Ablating metadata and normalization. Using readily available metadata is overall beneficial for our model. The model without metadata can sometimes lead to better results, like in ScanNet, where the baseline between frames is many times too small and relying on the cost volume does not provide as much benefit. However the average results indicate that the using metadata is superior. Although using metadata without scale normalization can report better results on this metric benchmark, we show in Tab. A1 how this model struggles when using an arbitrary scale. We exemplify this by scaling the ScanNet poses by an arbitrary factor where we rescale ScanNet poses and depths (which are in metric scale) by a factor of 100. This simulates the type of 'non-metric scene scale' we might see if, for example, we reconstructed a scene using SfM package such as COLMAP which does not provide metric scaled reconstructions.

Ablating the cost volume. Not using noise on the cost volume range during training significantly impacts performance in cases where our range estimate using camera intrinsics and extrinsics is poor, like in DTU or ScanNet. Unsurprisingly these are the datasets where not doing a refinement has the biggest impact. A fixed 0-100m range can be beneficial in some cases where the test data overlaps this range. However, the model fails in the case of very different ranges like on DTU, and would not work with non-metric poses.

Ablating the overall architecture. We compare our model without starting from DAv2 pretrained weights, using a smaller ViT and using a previous state-of-the-art architecture from [62]. Each of these variants obtain worse results than using our architecture.

Effect of our training datasets. We show in Table A3 head-to-head network comparisons trained on the same data. In the first two rows, we show our network *vs*. MVSFormer++ [7], both of which were trained from scratch on DTU+BlendedMVS. In the bottom two rows, we fine-tuned[7] for 40K steps with all MVSAnywhere datasets. We use their 'regression loss option as we found that this gave them better scores when training with our data. However, this model is subpar to ours, particularly in casually captured datasets such as ScanNet.

Evaluating our depth range estimation. In A4, we show how our heuristic to estimate the depth range also works well in conjunction with MVSFormer++ [7]'s architecture even though it was not trained to handle noisy ranges (but still it does not beat our full system).

Approach	**ScanNet**		**ScanNet** ×100	
	rel ↓	τ ↑	rel ↓	τ ↑
Ours (low res) [67]	4.22	61.80	4.22	61.83
Ours w/o norm. metadata w/o view count agnostic	3.97	61.23	4.34	57.59

Table A1. **Results on an arbitrary scale**. We scale the ScanNet poses by a factor of 100 to evaluate robustness to arbitrary scales.

	DTU (mm)		
	Acc.↓	Compl.↓	Overall↓
Direct depth regression + cloud fusion			
Ours	0.845	0.625	0.735
Robust MVD	1.330	1.029	1.180
MAST3R + triangulation	3.969	3.548	3.758
DUST3R	2.677	0.805	1.741
MVSFormer++ [6]	(0.309)	(0.252)	(0.281)
Sub-pixel matching + triangulation			
MAST3R [39]	0.403	0.344	0.374

Table A2. **DTU Reconstruction.** We evaluate our method and other direct depth regression methods on DTU reconstruction.

Results on DTU. Results using the point cloud evaluation on DTU are shown in Table A2. We follow [7] to fuse our depth maps using their confidence predictions and their default fusion hyperparameters.

E. Gaussian Splat Regularization using MVSAnywhere

We show a use case of our method as a regularizer for Gaussian Splats. We follow a similar approach as VCR-GauS [8] and DN-Splatter [74] regularizing depth and normals during training as additional losses. In Figure A1, we show qualitative results of the meshes obtained after using raw Splatfacto without regularization, and with normal and depth supervision from Metric3D [30] and from MVSA. Note that we used an scale-invariant loss for Metric3D given that the used scenes lay in arbitrary scales. More details on this regularization are available on the code.

F. Additional qualitative results

We present additional qualitative results in Figs. A2 to A4, A6 and A7 to A10. Figs. A2, A5 and A8 depicts depth maps where the range has been normalized using the ground truth range. It demonstrates how our MVSA approach is able to predict very similar scale to the ground truth, whereas Depth Pro struggles to do so. In Figs. A3, A6 and A9 the depth maps are normalized per-image. Our model and Depth Pro produce the sharpest results. In Figs. A4,

Approach	**GT**	**GT**	**Align**	**KITTI**		**ScanNet**		**ETH3D**		**DTU**		**T&T**		**Average**	
	Poses	**Range**		rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑
MVSFormer++ DTU+BlendedMVG [7]	✓	✓	✗	4.4	65.7	7.9	39.4	7.8	50.4	(0.9)	(95.3)	3.2	88.1	4.8	67.8
MVSA DTU+BlendedMVG [7]	✓	✓	✗	4.1	57.8	4.8	53.7	3.6	61.5	(1.4)	(90.7)	2.7	80.1	3.3	68.8
MVSFormer++ MVSA data	✓	✓	✗	3.7	67.8	5.4	46.5	4.9	59.1	1.3	90.1	2.0	89.7	3.5	70.6
MVSA MVSA data	✓	✗	✗	3.2	68.8	3.7	62.9	3.2	68.0	1.3	95.0	2.1	90.5	2.7	77.0

Table A3. **Effect of training data.** We run a head-to-head comparison with MVSFormer++[7], first training our architecture on MVSFormer++ data, and then training MVSFormer++ on our data. In both cases, our architecture is better suited for the task of multi-view depth estimation.

Approach	**GT**	**GT**	**Align**	**KITTI**		**ScanNet**		**ETH3D**		**DTU**		**T&T**		**Average**	
	Poses	**Range**		rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑
MVSFormer++ [7]	✓	✓	✗	4.4	65.7	7.9	39.4	7.8	50.4	(0.9)	(95.3)	3.2	88.1	4.8	67.8
MVSFormer++ [7] (0.25 - 100m)	✓	✗	✗	47.9	13.0	21.3	33.3	39.4	18.7	(25.6)	(51.5)	40.6	30.0	35.0	29.3
MVSFormer++ [7] (our heuristic)	✓	✗	✗	8.3	49.3	48.7	11.3	21.7	31.8	(45.4)	(59.7)	6.5	75.6	26.12	45.5
MVSFormer++ [7] (our heuristic + refinement)	✓	✗	✗	5.1	64.6	23.5	27.3	10.4	48.6	(25.5)	(62.6)	3.7	87.2	13.6	58.1
MVSA	✓	✗	✗	3.2	68.8	3.7	62.9	3.2	68.0	1.3	95.0	2.1	90.5	2.7	77.0

Table A4. **Depth range estimation.** We show how our heuristic to estimate a depth range can also work in other methods like MVSFormer++ [7]. This effectively removes the GT Range input requirement.

A7 and A10 the error maps reveal how our approach achieves the best alignment with the ground truth. Although MAST3R's scale is good, it has high errors around object boundaries.

Approach	GT Poses	GT Range	**Align**	**KITTI** rel ↓	τ ↑	**ScanNet** rel ↓	τ ↑	**ETH3D** rel ↓	τ ↑	**DTU** rel ↓	τ ↑	**T&T** rel ↓	τ ↑	**Average** rel ↓	τ ↑	time [s] ↓
Classical SfM approaches																
COLMAP [64, 65]	✓	✗	✗	**12.0**	**58.2**	**14.6**	**34.2**	**16.4**	**55.1**	**0.7**	**96.5**	**2.7**	**95.0**	**9.3**	**67.8**	≈ 180
COLMAP Dense [64, 65]	✓	✗	✗	26.9	52.7	38.0	22.5	89.8	23.2	20.8	69.3	25.7	76.4	40.2	48.8	≈ 180
a) Depth from frames (w/o poses)																
DeMoN [75]	✗	✗	$\|\mathbf{t}\|$	15.5	15.2	**12.0**	**21.0**	17.4	15.4	21.8	16.6	13.0	23.2	16.0	18.3	**0.08**
DeepV2D KITTI [73]	✗	✗	med	**(3.1)**	**(74.9)**	23.7	11.1	27.1	10.1	24.8	8.1	34.1	9.1	22.6	22.7	2.07
DeepV2D ScanNet [73]	✗	✗	med	10.0	36.2	(4.4)	(54.8)	11.8	29.3	7.7	33.0	**8.9**	**46.4**	8.6	39.9	3.57
MAST3R [43] (raw output)	✗	✗	med	**3.3**	**67.7**	**(4.3)**	**(64.0)**	**2.7**	**79.0**	**3.5**	**66.7**	**(2.4)**	**(81.6)**	**3.3**	**71.8**	0.07
MAST3R [43] (raw output)	✗	✗	✗	61.4	0.4	(12.8)	(19.4)	43.8	3.1	145.8	0.5	(66.9)	(0.0)	66.1	4.7	0.07
b) Depth from frames and poses (with per-image range provided)																
MVSNet [96]	✓	✓	✗	22.7	36.1	24.6	20.4	35.4	31.4	(1.8)	(86.0)	8.3	73.0	18.6	49.4	**0.07**
MVSNet Inv. Depth [96]	✓	✓	✗	18.6	30.7	22.7	20.9	21.6	35.6	(1.8)	(86.7)	6.5	74.6	14.2	49.7	0.32
CVP-MVSNet [91]	✓	✓	✗	156.7	2.2	137.1	15.9	156.4	13.6	(4.0)	(68.4)	24.7	52.9	95.8	30.6	0.49
Vis-MVSNet [103]	✓	✓	✗	9.5	55.4	8.9	33.5	10.8	43.3	(1.8)	(87.4)	4.1	87.2	7.0	61.4	0.70
PatchmatchNet [76]	✓	✓	✗	10.8	45.8	8.5	35.3	19.1	34.8	(2.1)	(82.8)	4.8	82.9	9.1	56.3	0.28
Fast-MVSNet [99]	✓	✓	✗	14.4	37.1	17.0	24.6	25.2	32.0	(2.5)	(81.8)	8.3	68.6	13.5	48.8	0.30
MVS2D ScanNet [95]	✓	✓	✗	21.2	8.7	(27.2)	(5.3)	27.4	4.8	17.2	9.8	29.2	4.4	24.4	6.6	**0.04**
MVS2D DTU [95]	✓	✓	✗	226.6	0.7	32.3	11.1	99.0	11.6	(3.6)	(64.2)	25.8	28.0	77.5	23.1	0.05
MVSFormer++ DTU [7]	✓	✓	✗	26.3	42.8	16.7	28.0	30.3	40.1	**(0.8)**	**(95.7)**	7.2	82.3	16.3	57.8	0.78
MVSFormer++ DTU+BlendedMVG [7]	✓	✓	✗	**4.4**	**65.7**	**7.9**	**39.4**	**7.8**	**50.4**	(0.9)	(95.3)	**3.2**	**88.1**	**4.8**	**67.8**	0.78
c) Single-view depth																
Depth Pro [3] †	✗	✗	med	6.1	39.6	(4.3)	(58.4)	6.1	53.5	5.6	49.6	5.6	57.5	5.6	51.7	5.16
Depth Pro [3] †	✗	✗	✗	13.6	14.3	9.2	19.7	28.5	8.7	161.8	3.5	38.3	4.4	50.3	10.1	5.16
Metric3D [30] †	✗	✗	med	5.1	44.1	**2.4**	**78.3**	4.4	54.5	10.1	39.5	6.2	48.0	5.6	52.9	0.46
Metric3D [30] †	✗	✗	✗	8.7	13.2	6.2	19.3	12.7	13.0	890.5	1.4	16.7	13.7	187.0	12.1	0.46
UniDepthV2 [56] †	✗	✗	med	**4.0**	**55.3**	(2.1)	(82.6)	**3.7**	**66.2**	3.2	72.3	**3.6**	**68.4**	**3.3**	**68.9**	0.29
UniDepthV2 [56] †	✗	✗	✗	13.7	4.8	(3.2)	(61.3)	15.4	11.9	964.8	1.3	16.7	12.7	202.7	18.4	0.29
UniDepthV1 [56] †	✗	✗	med	4.4	51.6	**(1.9)**	**(84.3)**	5.4	48.4	9.3	31.8	9.6	38.7	6.1	51.0	0.21
UniDepthV1 [56] †	✗	✗	✗	5.2	39.5	(2.7)	(69.4)	48.2	1.8	583.3	1.0	30.7	4.2	134.0	23.2	0.20
DepthAnything V2 (ViT-B) [94]	✗	✗	lstsq †	6.6	38.6	4.0	58.6	4.7	56.5	**2.6**	**74.7**	4.5	57.5	4.8	54.1	**0.05**
d) Depth from frames and poses (w/o per-image range)																
DeMoN [75]	✓	✗	✗	16.7	13.4	75.0	0.0	19.0	16.2	23.7	11.5	17.6	18.3	30.4	11.9	0.08
DeepTAM [106]	✓	✗	✗	68.7	0.4	(6.7)	(39.7)	20.4	19.8	58.0	9.1	40.0	12.9	38.8	16.4	0.85
DeepV2D KITTI [73]	✓	✗	✗	(20.4)	(16.3)	25.8	8.1	30.1	9.4	24.6	8.2	38.5	9.6	27.9	10.3	1.43
DeepV2D ScanNet [73]	✓	✗	✗	61.9	5.2	**(3.8)**	(60.2)	18.7	28.7	9.2	27.4	33.5	38.0	25.4	31.9	2.15
MVSNet [96]	✓	✗	✗	14.0	35.8	1568.0	5.7	507.7	8.3	(4429.1)	(0.1)	118.2	50.7	1327.4	20.1	0.15
MVSNet Inv. Depth [96]	✓	✗	✗	29.6	8.1	65.2	28.5	60.3	5.8	(28.7)	(48.9)	51.4	14.6	47.0	21.2	0.28
CVP-MVSNet [91]	✓	✗	✗	158.2	1.2	2289.0	0.1	1735.3	1.2	(8314.0)	(0.0)	415.9	9.5	2582.5	2.4	0.50
Vis-MVSNet [103]	✓	✗	✗	10.3	54.4	84.9	15.6	51.5	17.4	(374.2)	(1.7)	21.1	65.6	108.4	31.0	0.82
PatchmatchNet [76]	✓	✗	✗	29.0	16.3	70.1	16.7	99.4	3.5	(82.6)	(5.6)	39.4	19.3	64.1	12.3	0.18
Fast-MVSNet [99]	✓	✗	✗	12.1	37.4	287.1	9.4	131.2	9.6	(540.4)	(1.9)	33.9	47.2	200.9	21.1	0.35
MVS2D ScanNet [95]	✓	✗	✗	73.4	0.0	(4.5)	(54.1)	30.7	14.4	5.0	57.9	56.4	11.1	34.0	27.5	**0.05**
MVS2D DTU [95]	✓	✗	✗	93.3	0.0	51.5	1.6	78.0	0.0	**(1.6)**	(92.3)	87.5	0.0	62.4	18.8	0.06
Robust MVD Baseline [67]	✓	✗	✗	7.1	41.9	7.4	38.4	9.0	42.6	2.7	82.0	5.0	75.1	6.3	56.0	0.06
MAST3R (plus our triangulation)	✓	✗	✗	3.4	66.6	(4.5)	**(63.0)**	**3.1**	**72.9**	3.4	67.3	(2.4)	(83.3)	3.4	70.1	0.72
MVSA	✓	✗	✗	**3.2**	**68.8**	**3.7**	**62.9**	3.2	68.0	**1.3**	**95.0**	**2.1**	**90.5**	**2.7**	**77.0**	0.12

Table A5. **Full results on the RMVDB**. See Sec. 4 in the main paper for details of the metrics, baselines and groupings. Monocular methods with † are given ground truth intrinsics. The best result for each section appears in **bold**, and (parentheses) indicates a given model is trained on data from the same dataset.

	Architecture	Resolution	Pretrn. DAV2 weights	Metadata	Noise on GT range	Bin refinement†	KITTI		ScanNet		ETH3D		DTU		T&T		Average	
							rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑	rel ↓	τ ↑
Ours full, high-res	Ours	640 ×480	✓	✓	✓	✓	3.2	68.8	3.7	62.9	3.2	68.0	1.3	95.0	2.1	90.5	2.7	77.0
Ours w/o metadata, high-res	Ours	640 ×480	✓		✓	✓	3.3	68.3	3.7	63.8	3.1	69.3	1.9	93.5	2.1	89.9	2.8	77.0
a) Ablate Mono/Multi Cue Combiner																		
Ours w/o metadata	Ours	448 × 336	✓		✓	✓	3.39	66.88	3.86	60.82	3.11	70.17	2.43	92.05	2.23	88.38	3.00	75.66
” no MMCC ViT	w/o MMCC ViT	448 × 336	✓		✓	✓	3.54	65.32	4.39	56.94	3.56	65.27	3.07	90.49	2.46	87.54	3.41	73.11
” w/ naive patchify	Naive patchify	448 × 336	✓		✓	✓	3.66	62.61	4.27	58.86	3.18	67.27	1.95	91.69	2.46	86.52	3.11	73.39
b) Ablate metadata and normalization																		
Ours (low res.)	Ours	448 × 336	✓	✓	✓	✓	3.31	67.63	4.05	61.27	3.29	66.66	2.51	93.15	2.26	88.66	3.08	75.47
w/o normalized metadata w/o view count agnostic	Ours	448 × 336	✓	✓	✓	✓	3.39	66.23	3.97	61.23	3.10	68.84	2.25	91.98	2.24	88.20	2.99	75.30
Ours w/o metadata	Ours	448 × 336	✓		✓	✓	3.39	66.88	3.86	60.82	3.11	70.17	2.43	92.05	2.23	88.38	3.00	75.66
c) Ablate cost volume																		
Ours w/o metadata	Ours	448 × 336	✓		✓	✓	3.39	66.88	3.86	60.82	3.11	70.17	2.43	92.05	2.23	88.38	3.00	75.66
” w/o noise on GT range	Ours	448 × 336	✓			✓	3.33	66.83	5.14	52.77	3.53	66.32	13.45	89.21	2.34	87.64	5.32	74.89
” w/o bin refinement	Ours	448 × 336	✓		✓		3.57	63.39	5.18	51.05	3.50	67.93	6.80	82.12	2.27	87.04	4.26	70.31
” w/ fixed bins [0-100m]	Ours	448 × 336	✓		✓	F	3.41	64.33	3.80	61.62	3.15	67.20	4.11	65.64	2.36	85.72	3.37	68.90
d) Ablate overall architecture																		
Ours w/o metadata	Ours	448 × 336	✓		✓	✓	3.39	66.88	3.86	60.82	3.11	70.17	2.43	92.05	2.23	88.38	3.00	75.66
” w/o DAV2 weights	Ours	448 × 336			✓	✓	3.45	65.42	4.58	57.14	3.48	65.59	2.14	92.48	2.56	86.01	3.24	73.33
” w/ ViT Small	ViT Small	448 × 336	✓		✓	✓	3.57	64.34	4.40	56.57	3.63	64.61	2.69	91.52	2.71	84.51	3.40	72.31
” w/ [62]'s architecture	From [62]	512 × 384	✓		✓	✓	3.63	65.94	5.03	51.76	3.74	63.09	1.77	90.97	2.78	87.90	3.39	71.93

Table A6. **Ablation on RMVDB.** We validate our design decisions on RMVDB [67] by ablating various components. Except otherwise indicated, our ablations are run on a low-resolution version of our network, trained without metadata (for tractability reasons). The row with **F** uses a fixed set of bins (0-100m) for all images.

	Architecture	Resolution	Pretrained DAV2 weights	Metadata	Noise on GT range	Bin refinement†	**ScanNet**		**ETH3D**	
							rel ↓	τ ↑	rel ↓	τ ↑
Ours full, high-res	Ours	640 ×448	✓	✓	✓	✓	3.22	69.45	1.27	93.24
Ours w/o metadata, high-res	Ours	640 ×448	✓		✓	✓	3.28	69.31	1.25	93.45
a) Ablate cost volume patchifier										
Ours w/o metadata	Ours	448 × 336	✓		✓	✓	3.22	69.16	1.52	91.02
” no MMCC ViT	w/o MMCC ViT	448 × 336	✓		✓	✓	3.42	67.06	1.75	89.62
” w/ naive patchify	Naive patchify	448 × 336	✓		✓	✓	3.40	68.05	1.59	90.06
b) Ablate metadata and normalization										
Ours (low res)	Ours	448 × 336	✓	✓	✓	✓	3.30	69.70	1.46	91.65
” w/o normalized metadata w/o view count agnostic	Ours	448 × 336	✓	✓	✓	✓	3.28	68.87	1.47	91.30
c) Ablate cost volume										
Ours w/o metadata	Ours	448 × 336	✓		✓	✓	3.22	69.16	1.52	91.02
” w/o noise on GT range	Ours	448 × 336	✓			✓	3.93	64.47	1.86	90.36
” w/o bin refinement	Ours	448 × 336	✓		✓		4.13	59.53	1.92	86.63
” w/ fixed bins [0-100m]	Ours	448 × 336	✓		✓	**F**	3.27	69.03	1.96	86.78
d) Ablate overall architecture										
Ours w/o metadata	Ours	448 × 336	✓		✓	✓	3.22	69.16	1.52	91.02
” w/o DAV2 weights	Ours	448 × 336			✓	✓	3.33	67.94	1.71	89.13
” w/ ViT Small	ViT Small	448 × 336	✓		✓	✓	3.43	66.98	1.89	87.90
” w/ [62]'s architecture	From [62]	512 × 384	✓		✓	✓	4.02	62.37	1.57	91.57

Table A7. **Ablation RMVDBVariant**. We use better test-time tuples for ScanNet, and for ETH3D we use the undistorted test images.

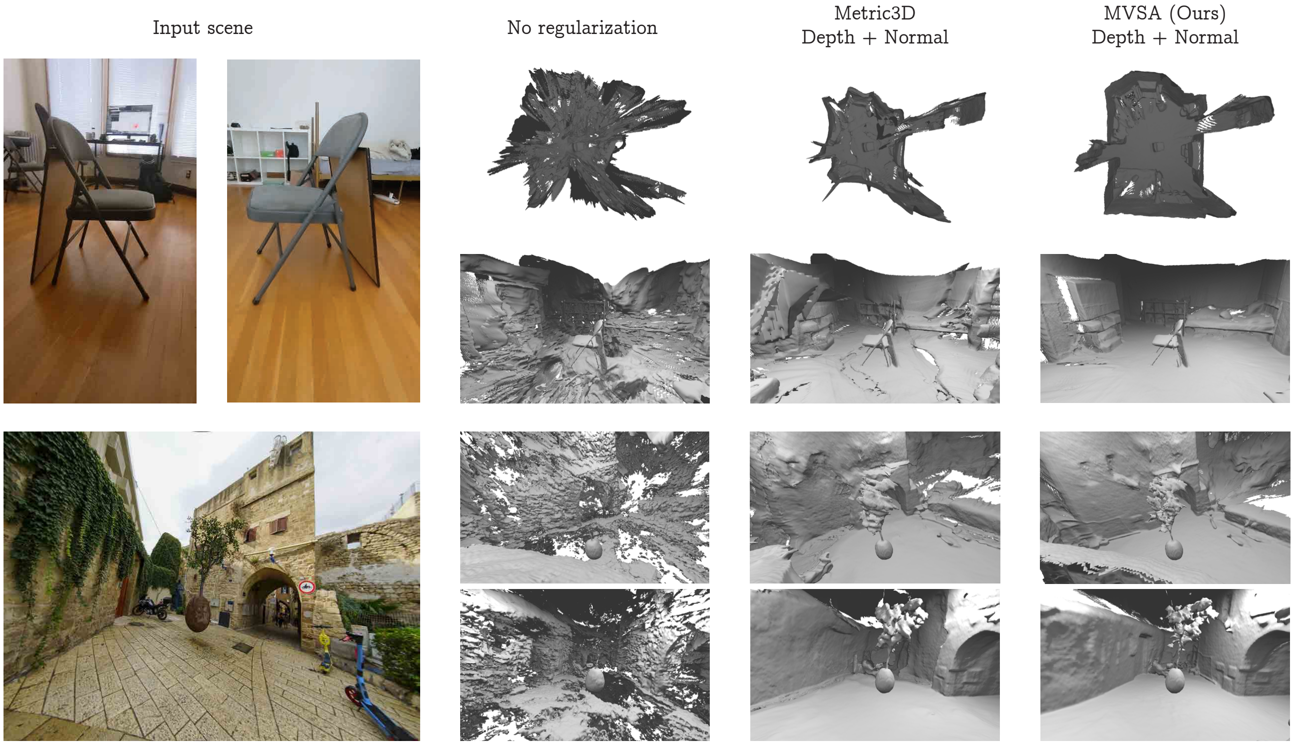


Figure A1. **Qualitative comparison of Gaussian Splat meshes using Metric3D and MVSA as regularizers.**

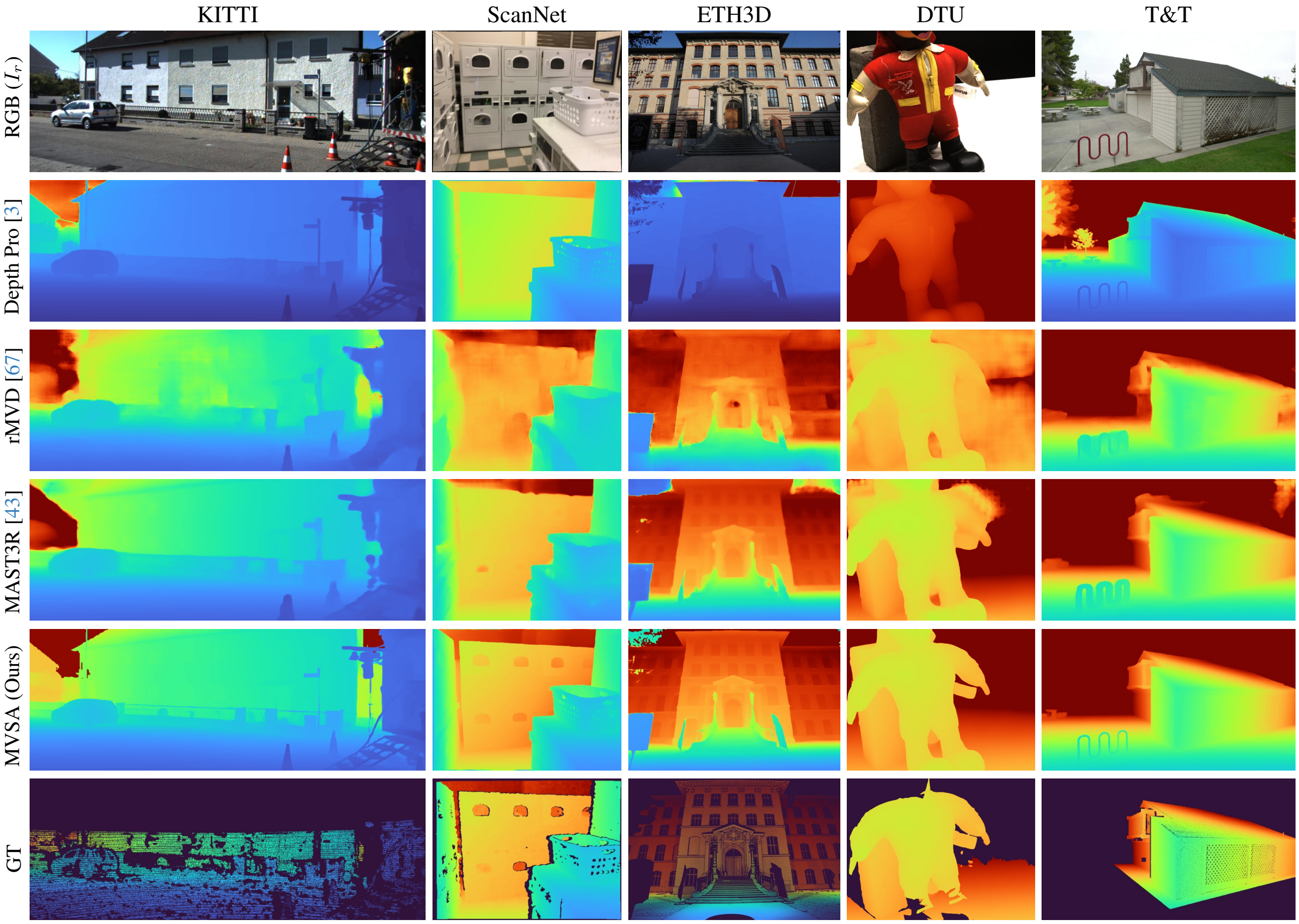


Figure A2. **Qualitative comparison of depth prediction results across multiple datasets (Normalized). This is a duplicate from main paper, for completeness.** (KITTI, ScanNet, ETH3D, DTU, and Tanks & Temples). Rows show different methods: Depth Pro [67], rMVD baseline [67], MAST3R (Triangulated) [43], and our MVSA model, along with RGB inputs (I_r) and ground-truth depths (GT). Depth Pro provides sharp edges but often misestimates depth scale, while our MVSA model captures finer details than MAST3R and rMVD. Depth maps are normalized to ground truth depth range for consistent visualization.

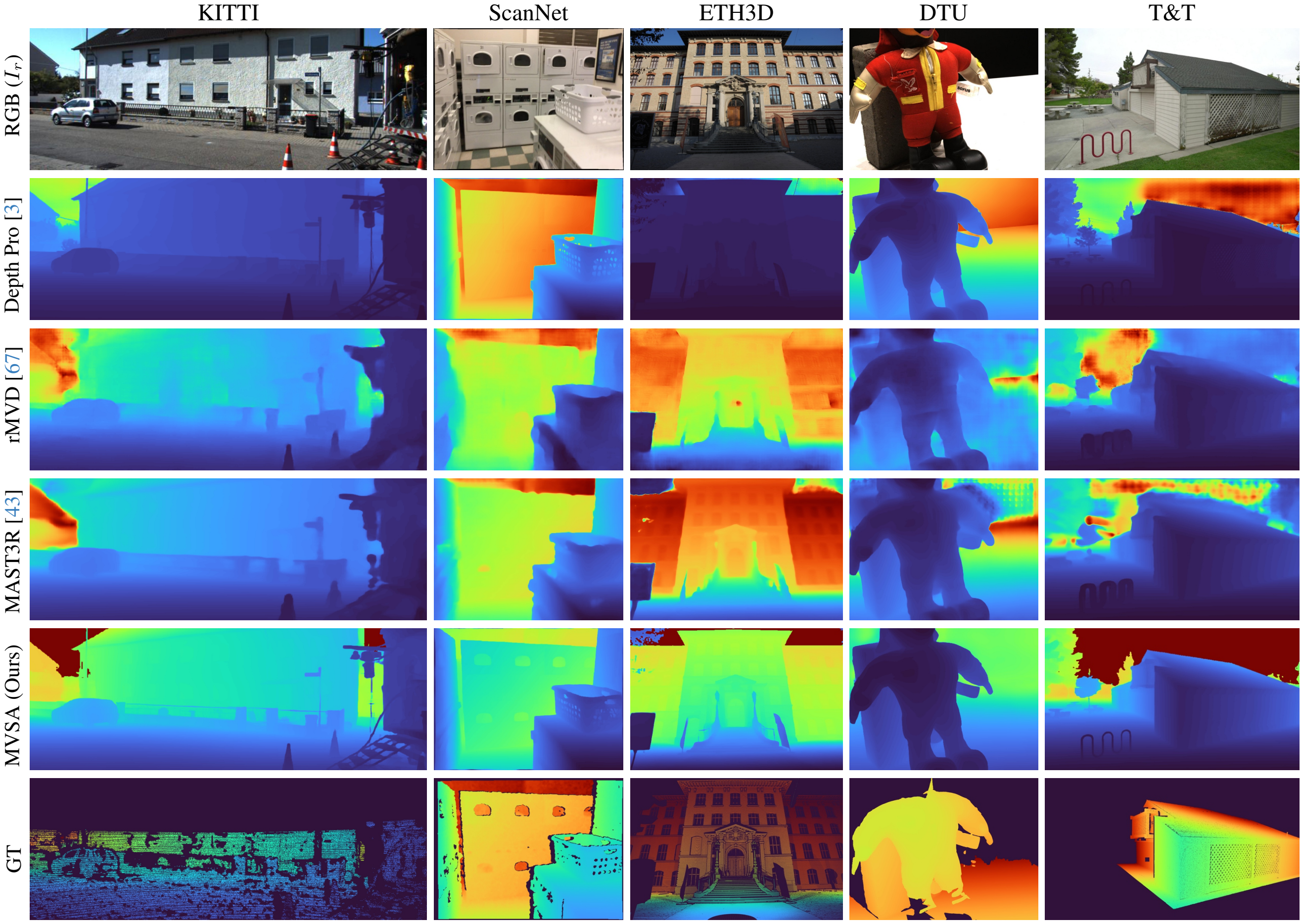


Figure A3. **Qualitative comparison of depth prediction results across multiple datasets (Unnormalized)** (KITTI, ScanNet, ETH3D, DTU, and Tanks & Temples). Rows show different methods: Depth Pro [67], rMVD baseline [67], MAST3R (Triangulated) [43], and our MVSA model, along with RGB inputs (I_r) and ground-truth depths (GT). Depth maps are normalized per image.

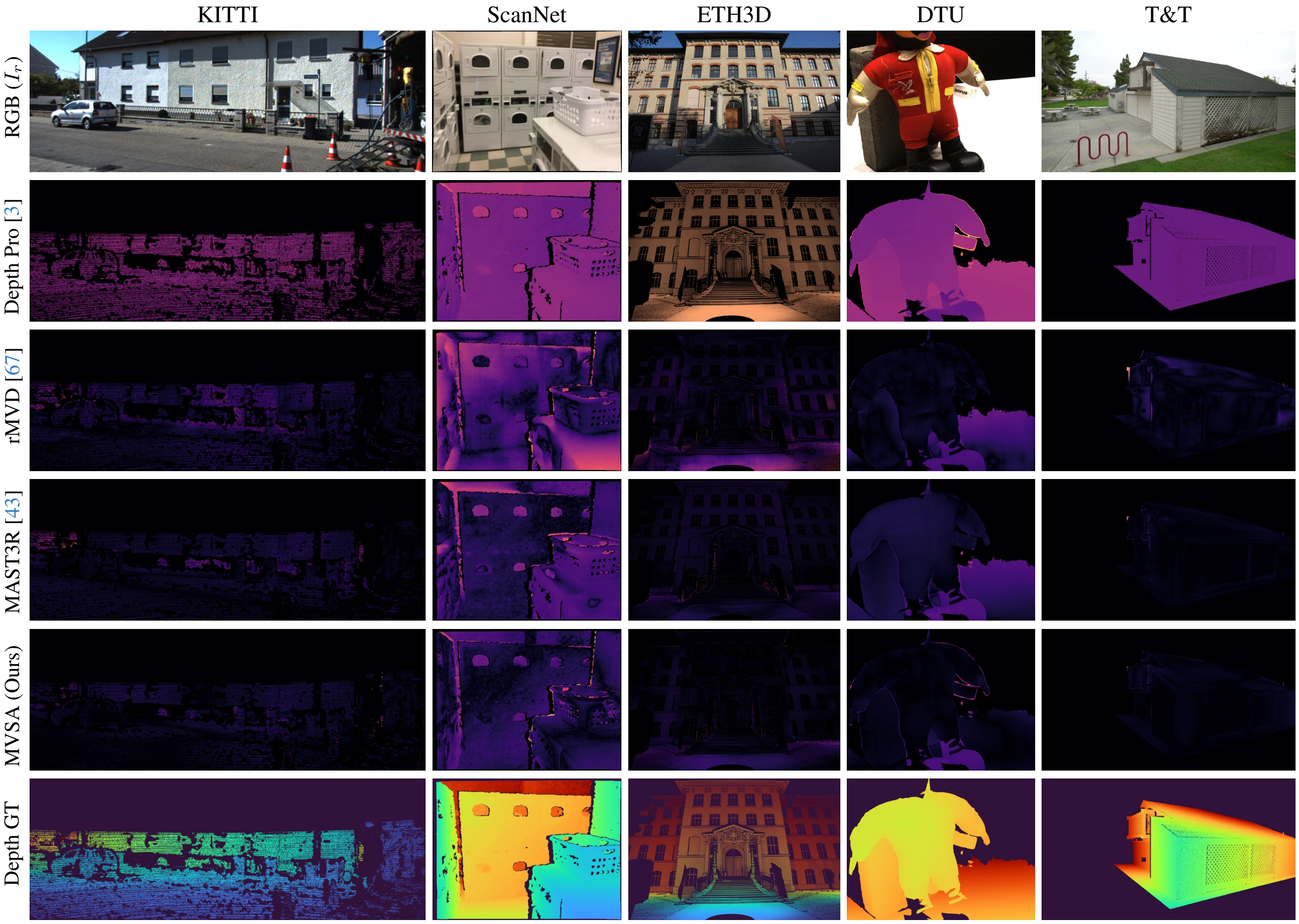


Figure A4. **Qualitative comparison of depth prediction errors across multiple datasets** (KITTI, ScanNet, ETH3D, DTU, and Tanks & Temples). Rows show different methods: Depth Pro [67], rMVD baseline [67], MAST3R (Triangulated) [43], and our MVSA model, along with RGB inputs (I_r) and ground-truth depths (GT). Depth Pro provides sharp edges but often misestimates depth scale, while our MVSA model captures finer details than MAST3R and rMVD. Error maps are normalized to maximum error among methods for each scene.

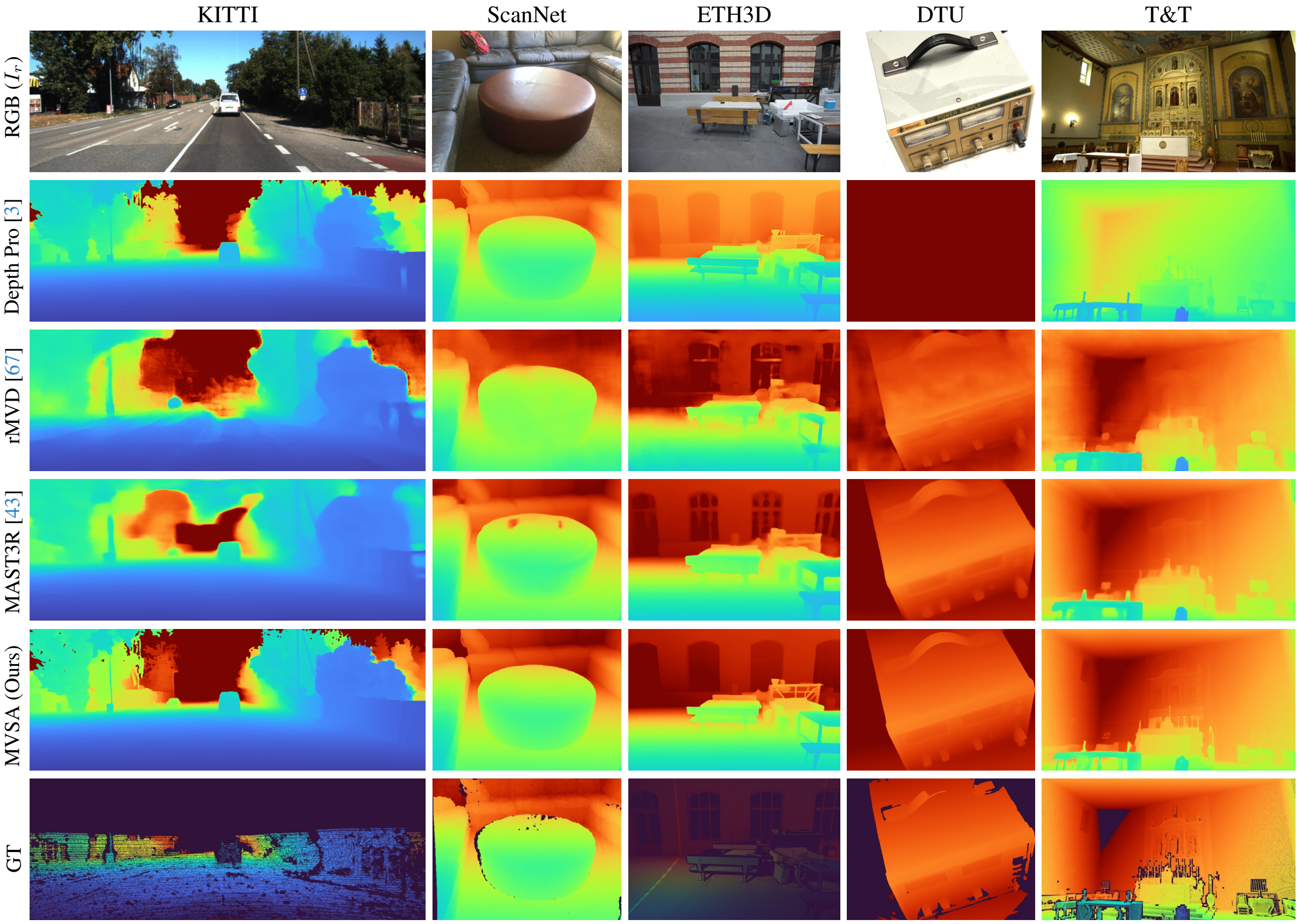


Figure A5. **Qualitative comparison of depth prediction results across multiple datasets (Normalized)** (KITTI, ScanNet, ETH3D, DTU, and Tanks & Temples). Rows show different methods: Depth Pro [67], rMVD baseline [67], MAST3R (Triangulated) [43], and our MVSA model, along with RGB inputs (I_r) and ground-truth depths (GT). Depth Pro provides sharp edges but often misestimates depth scale, while our MVSA model captures finer details than MAST3R and rMVD. Depth maps are normalized to ground truth depth range for consistent visualization.

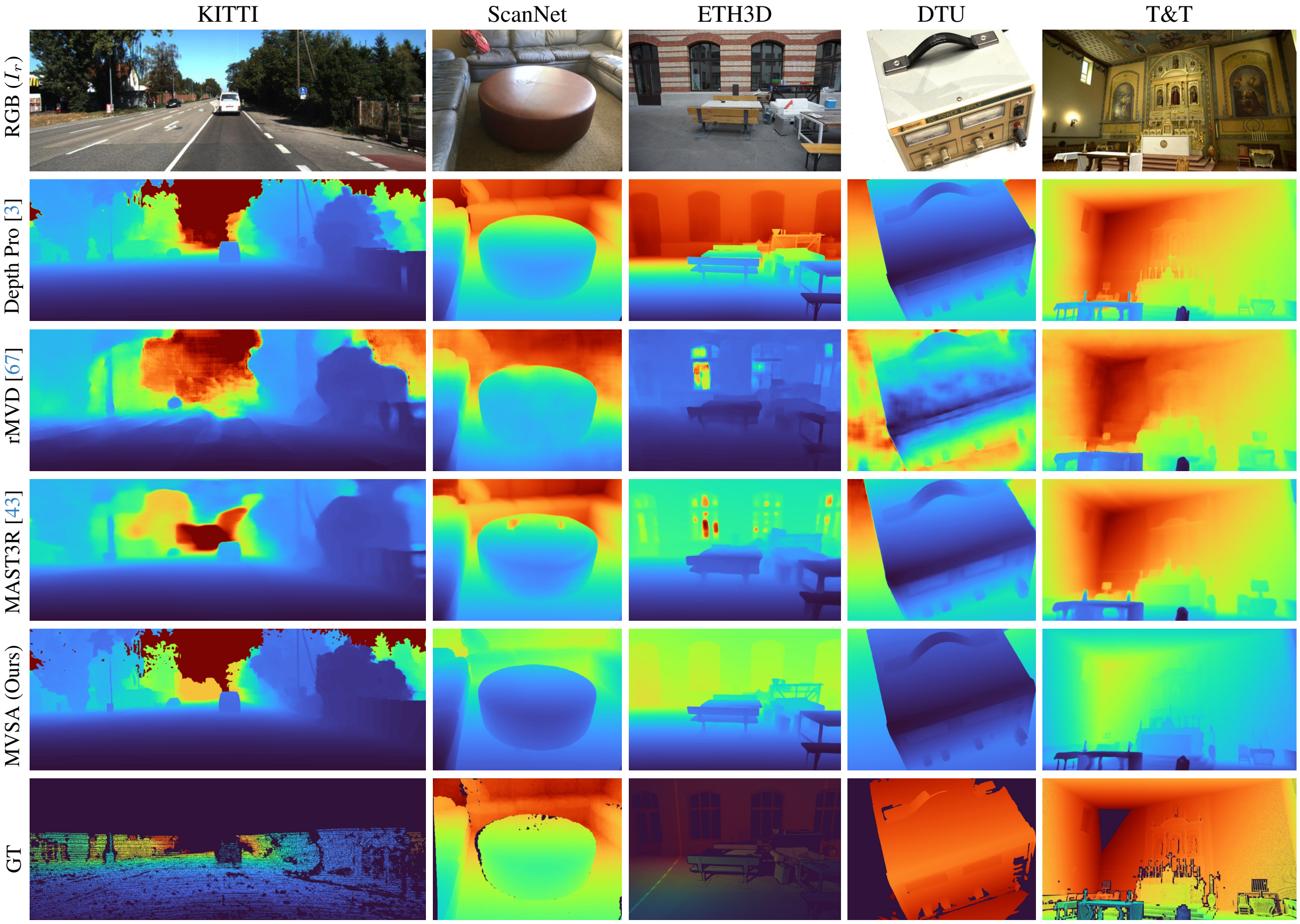


Figure A6. **Qualitative comparison of depth prediction results across multiple datasets (Unnormalized)** (KITTI, ScanNet, ETH3D, DTU, and Tanks & Temples). Rows show different methods: Depth Pro [67], rMVD baseline [67], MAST3R (Triangulated) [43], and our MVSA model, along with RGB inputs (I_r) and ground-truth depths (GT). Depth maps are normalized per image.

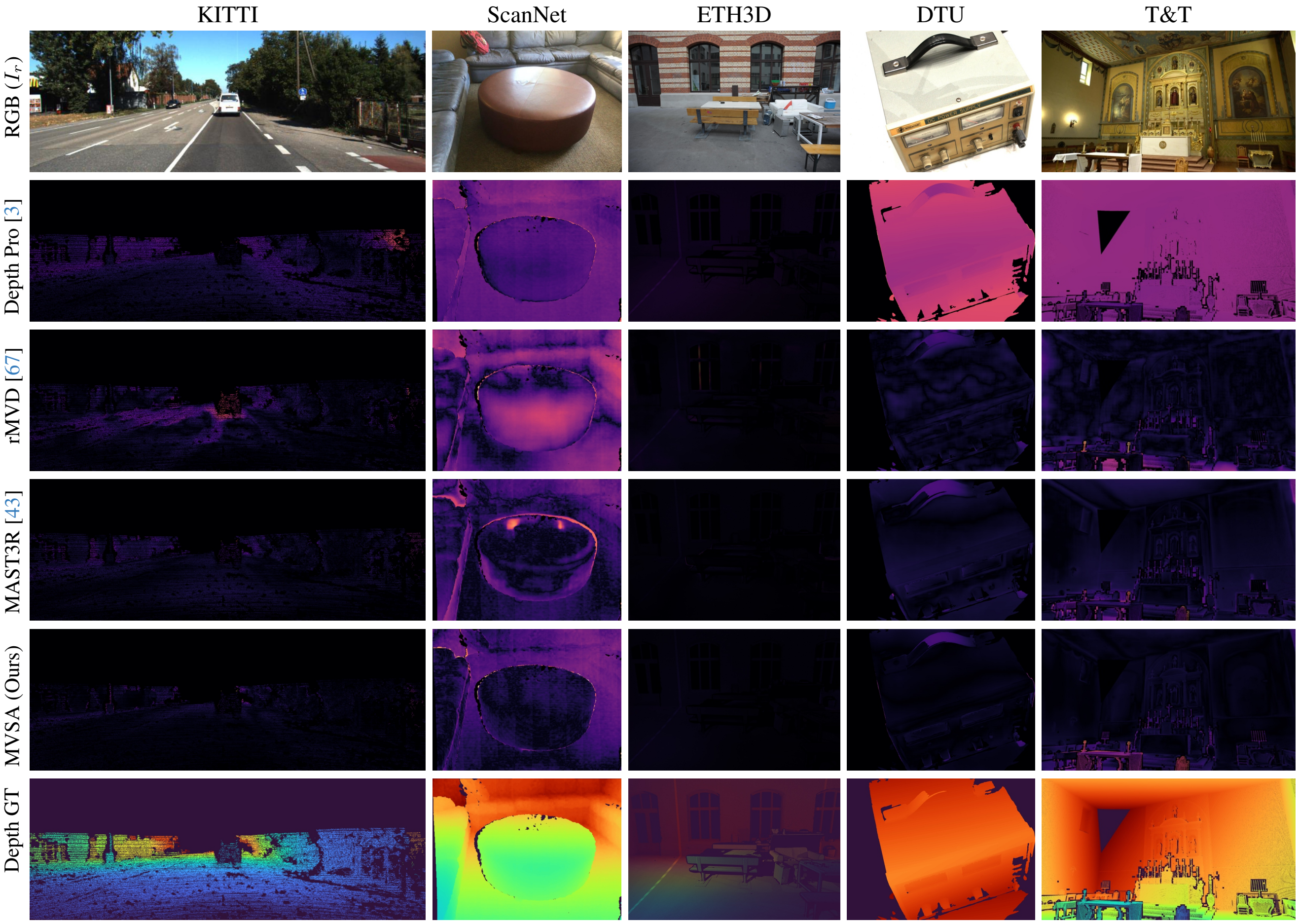


Figure A7. **Qualitative comparison of depth prediction errors across multiple datasets** (KITTI, ScanNet, ETH3D, DTU, and Tanks & Temples). Rows show different methods: Depth Pro [67], rMVD baseline [67], MAST3R (Triangulated) [43], and our MVSA model, along with RGB inputs (I_r) and ground-truth depths (GT). Depth Pro provides sharp edges but often misestimates depth scale, while our MVSA model captures finer details than MAST3R and rMVD. Error maps are normalized to maximum error among methods for each scene.

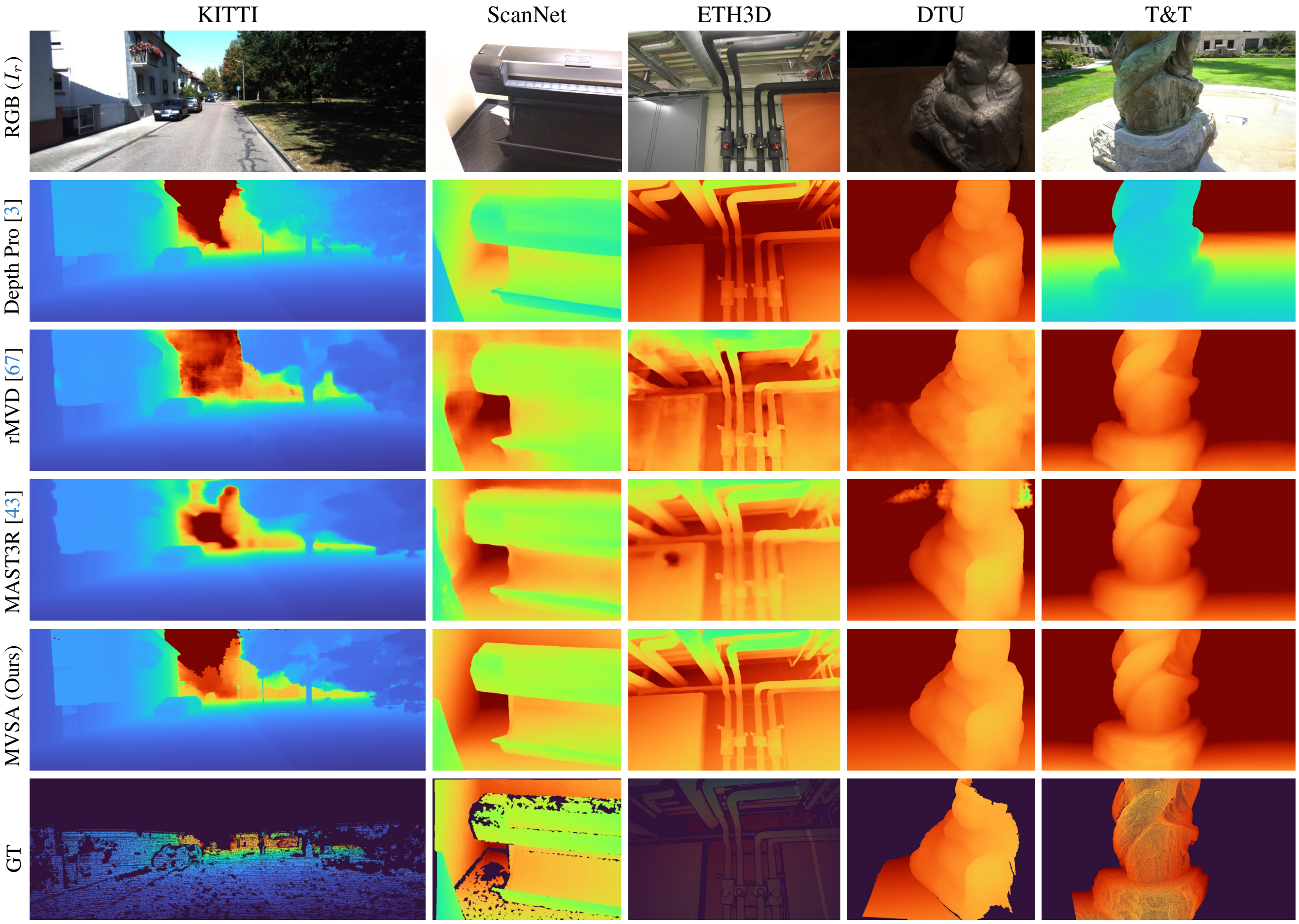


Figure A8. **Qualitative comparison of depth prediction results across multiple datasets (Normalized)** (KITTI, ScanNet, ETH3D, DTU, and Tanks & Temples). Rows show different methods: Depth Pro [67], rMVD baseline [67], MAST3R (Triangulated) [43], and our MVSA model, along with RGB inputs (I_r) and ground-truth depths (GT). Depth Pro provides sharp edges but often misestimates depth scale, while our MVSA model captures finer details than MAST3R and rMVD. Depth maps are normalized to ground truth depth range for consistent visualization.

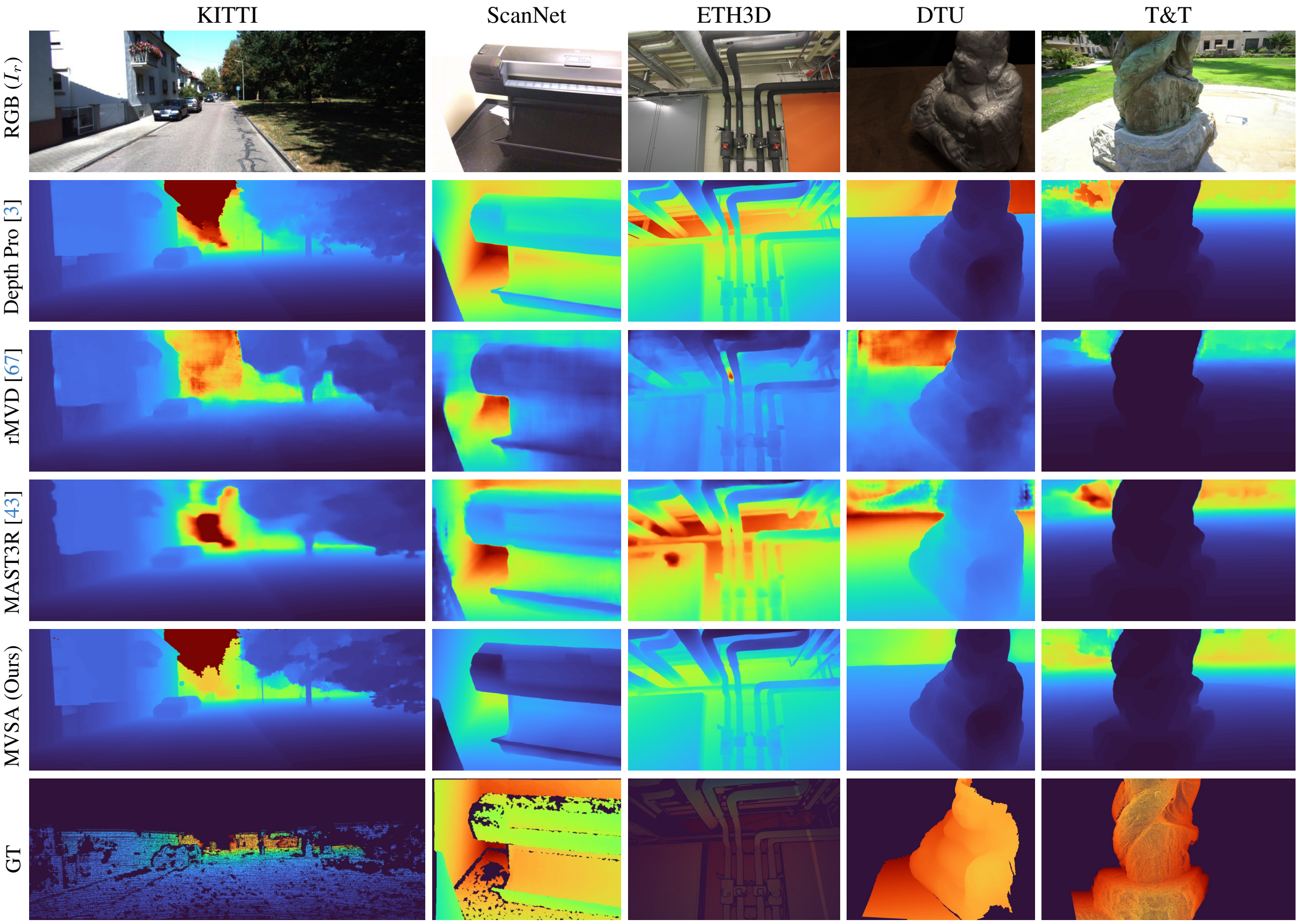


Figure A9. **Qualitative comparison of depth prediction results across multiple datasets (Unnormalized)** (KITTI, ScanNet, ETH3D, DTU, and Tanks & Temples). Rows show different methods: Depth Pro [67], rMVD baseline [67], MAST3R (Triangulated) [43], and our MVSA model, along with RGB inputs (I_r) and ground-truth depths (GT). Depth maps are normalized per image.

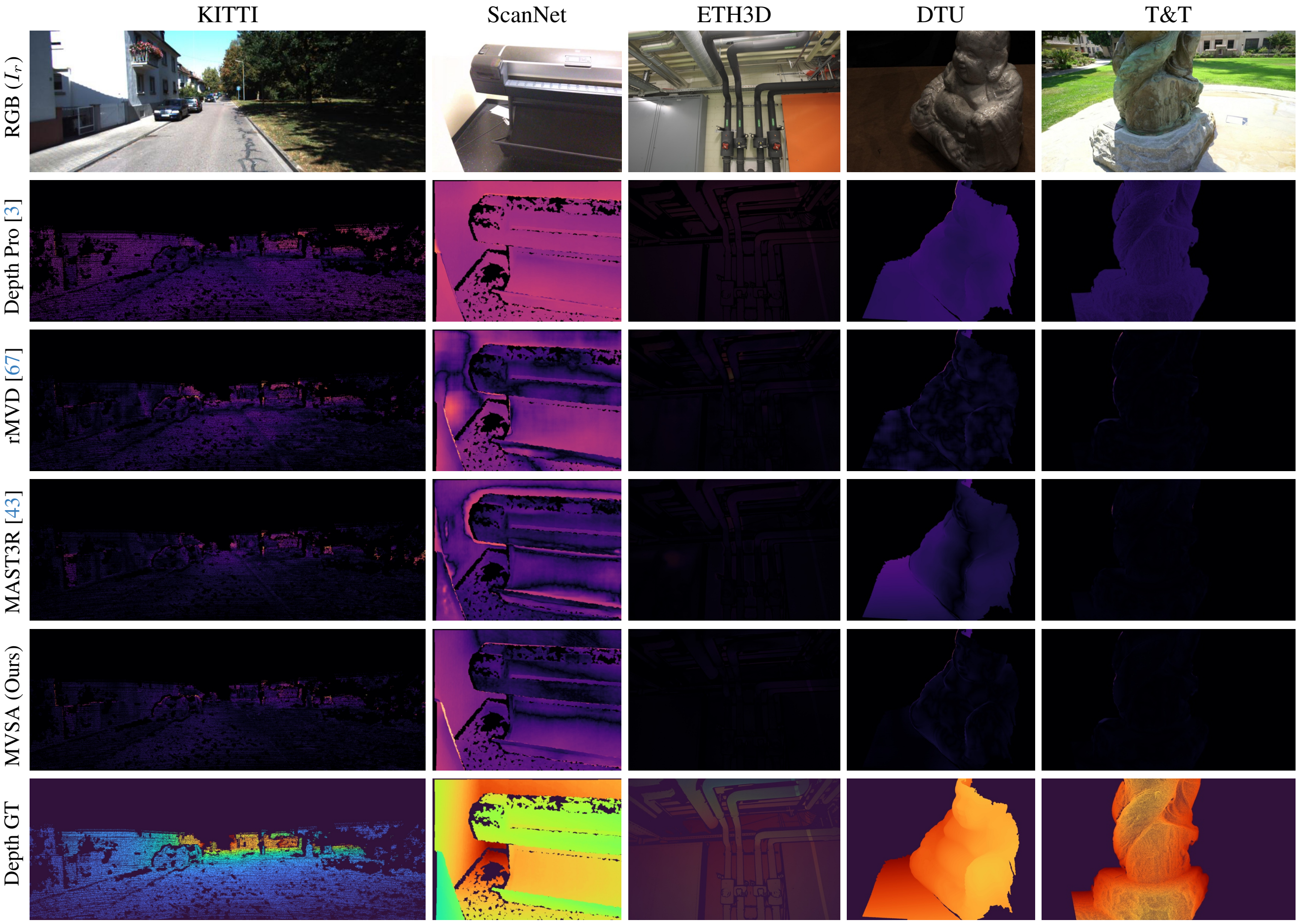


Figure A10. **Qualitative comparison of depth prediction errors across multiple datasets** (KITTI, ScanNet, ETH3D, DTU, and Tanks & Temples). Rows show different methods: Depth Pro [67], rMVD baseline [67], MAST3R (Triangulated) [43], and our MVSA model, along with RGB inputs (I_r) and ground-truth depths (GT). Depth Pro provides sharp edges but often misestimates depth scale, while our MVSA model captures finer details than MAST3R and rMVD. Error maps are normalized to maximum error among methods for each scene.