DoubleTake: Geometry Guided Depth Estimation Supplementary Material

Mohamed Sayed[1] Filippo Aleotti[1] Jamie Watson[1,2] Zawar Qureshi[1]
Guillermo Garcia-Hernando [1] Gabriel Brostow[1,2] Sara Vicente[1] Michael Firman[1]
[1]Niantic [2]UCL
https://nianticlabs.github.io/doubletake/

Contents

1. New evaluation masks

Metrics for mesh evaluation. We follow existing works [1, 8] and report reconstruction metrics based on point to point distances on point clouds sampled from the ground truth and predicted meshes. The first metric, *Accuracy*, is the mean of the distances between every point in the predicted point cloud and its nearest neighbor in the ground truth point cloud. The second metric, *Completion* is the reverse, the mean of the distances between every point in ground truth point cloud and its nearest neighbor in the predicted point cloud. *Chamfer* is the mean of *Accuracy* and *Completion*. *Precision* and *Recall* are the ratio of points whose *Accuracy* and *Completion* are less than 5cm respectively. And finally *F-Score* is the harmonic mean of *Precision* and *Recall*. Methods may under-predict geometry (*i.e.* predict a mesh which is not complete enough relative to the ground truth) and achieve better *Accuracy* and *Precision* at the expense of worse *Completion* and *Recall*. For example, a predicted mesh with a single point perfectly placed on a ground truth point will yield perfect *Precision*. Over-predicting geometry, for example placing a point everywhere in the volume, will yield the perfect *Completion* and *Recall* at the expense of *Accuracy* and *Precision*. We therefore consider *Chamfer* and *F-Score* as the best representations of overall prediction quality.

Incomplete ground truth and masking. The ground truth meshes in ScanNetV2 are only as complete as the scan sequences allow. This means that there are gaps in the ground truth geometry in areas the camera hasn't seen. In practice, methods may predict geometry not present in the ground truth, and get unfairly punished on *Accuracy* and *Precision*. To prevent this, previous work applied a mask to the predicted meshes when computing *Accuracy* and *Precision*, attempting to ensure that only parts of the scene which are also present in the ground truth mesh are evaluated.

Previous masking strategies. ATLAS [8] rendered depth maps from a predicted mesh, invalidated rendered depths using ground-truth depth map validity, re-fused it using TSDF fusion, and finally computed point-to-point distances on the re-fused meshes. The evaluation, however, is limited by the accuracy of the TSDF fusion step, and previous work [13] has shown that the ground truth mesh scores poorly on this benchmark. TransformerFusion [1] iterate on this and instead use a visibility volume mask to trim predictions when computing *Accuracy*. We visualize such a mask in Fig 1. It is likely these masks were computed using ground-truth depth maps. Ground truth meshes do not have the same extent as the ground truth depth maps, therefore not all predicted points are correctly masked out when computing *Accuracy*. This problem is present in both the ATLAS [8] and TransformerFusion [1] benchmarks. To illustrate this, we use the Open3D [19] TSDF fuser to fuse ground truth depth maps from the ScanNetV2 test set into a mesh. We then evaluate these meshes using the evalua-

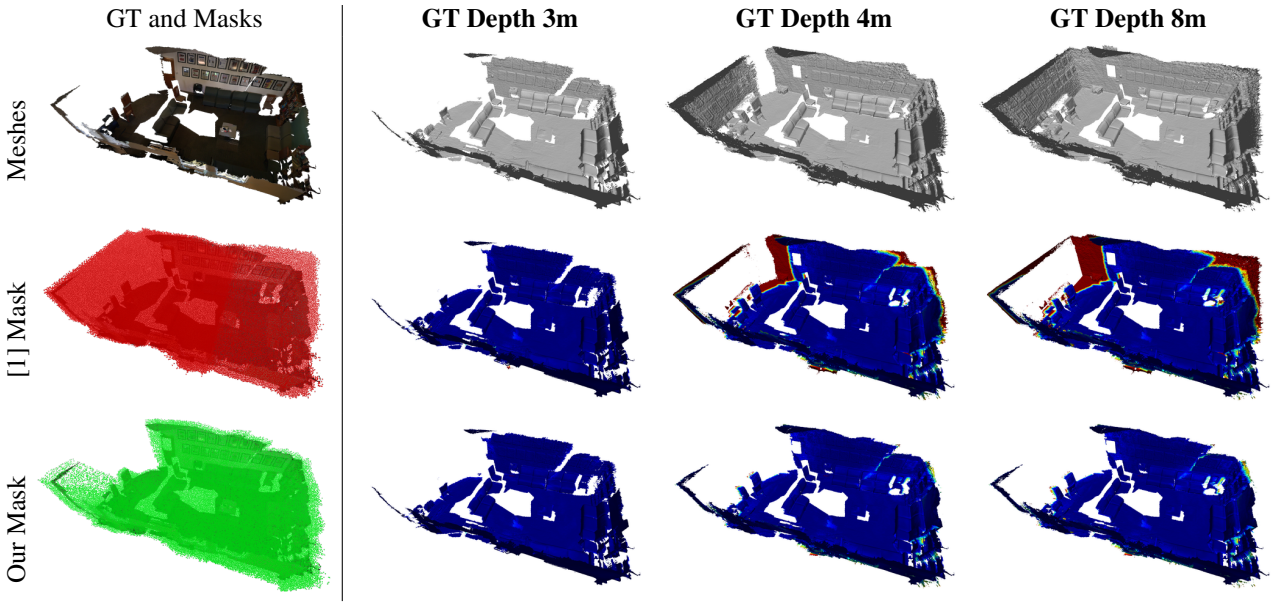

Figure 1. **Evaluation Mask Comparison.** The left-most column shows the ground truth mesh, with masks from [1] and 'ours' overlaid (in red and green respectively). We note that our mask is a tighter fit to the ground truth mesh. The remaining columns in the **top row** show meshes generated using the ground truth depth maps, fused at 3m, 4m and 8m maximum depth values. Notice that these meshes include geometry not present in the ground truth. The remaining cells in the table show the meshes at the top of each column masked using either the mask from [1] (row 2) or 'our' masks (row 3). Each mask's mesh is colored with an error map evaluating the error of each of the fused meshes. The error goes from blue (low error) to red (high error). Row two, which uses masks from [1], has many areas of high error shown in red. The final row shows the error maps when using our masks. Our masks more accurately reflect the geometry in the ground truth, which means that there are fewer areas of high (red) error in the meshes when using our masks.

tion protocol and masks from [1] in Table 1 (top). As we increase the maximum depth we fuse to, the mesh scores worse on *Accuracy* and *Precision* as there are now more predicted points outside the range of the ground truth mesh that don't have a close neighboring ground truth point.

Our proposed masks. We instead build visibility volumes using rendered depth maps of the ground-truth meshes. We initialize a volume using a ground truth mesh's extents with a 0.5m buffer in all axes. For every view in a sequence, we render the ground truth mesh into a depth map. We then project each voxel in the volume down to the current view, check if the projected pixel lands in the image, and z-test the projected pixel against the rendered mesh depth map. If the projected depth for a point is less than the mesh render's depth plus a buffer of 30cm, then that voxel is marked as visible. We visualize these masks and compare them with those from [1] in Fig 1. We also use these masks when computing the *Accuracy* error on meshes from fused ground depth maps in Table 1. The *Accuracy* and *Precision* scores do not fall as drastically with increased maxed depth as with the masks from [1], indicating that these masks allow for a fairer protocol for evaluation.

		Acc↓	Comp↓	Chamfer↓	Prec↑	Recall↑	F-Score↑
[1] Masks	GT Depth Fused 3m 2cm	0.80	1.76	1.28	.986	.954	.968
	GT Depth Fused 4m 2cm	3.87	1.16	2.52	.899	.986	.938
	GT Depth Fused 5m 2cm	5.90	1.27	3.59	.861	.980	.911
	GT Depth Fused 8m 2cm	6.00	1.35	3.67	.855	.977	.906
Our Masks	GT Depth Fused 3m 2cm	0.46	1.76	1.11	.995	.954	.973
	GT Depth Fused 4m 2cm	0.84	1.16	1.00	.970	.986	.978
	GT Depth Fused 5m 2cm	1.11	1.27	1.19	.953	.980	.966
	GT Depth Fused 8m 2cm	1.34	1.35	1.34	.940	.977	.957

Table 1. **Fused GT Mesh Evaluation Comparison on ScanNetV2**. Here we fuse the ground truth depth maps into a volume and evaluate the mesh against the ground truth meshes. We do this with two approaches to masking non-visible regions: at top, the approach from [1], and at bottom, our proposed approach. The scores from [1] degrade as we increase the maximum depth that depths are fused at. In contrast, our scores remain more constant. Furthermore, our approach gives overall significantly better scores, which suggests that our masks are more accurate representations of the ground truth visibility. See Section 1 for more details.

2. Additional results

2.1. Video – robustness to moving objects

In Figure 2 we show some frames from our video of our system estimating depth where there are moving objects. We show how we can reliably estimate depth for both the

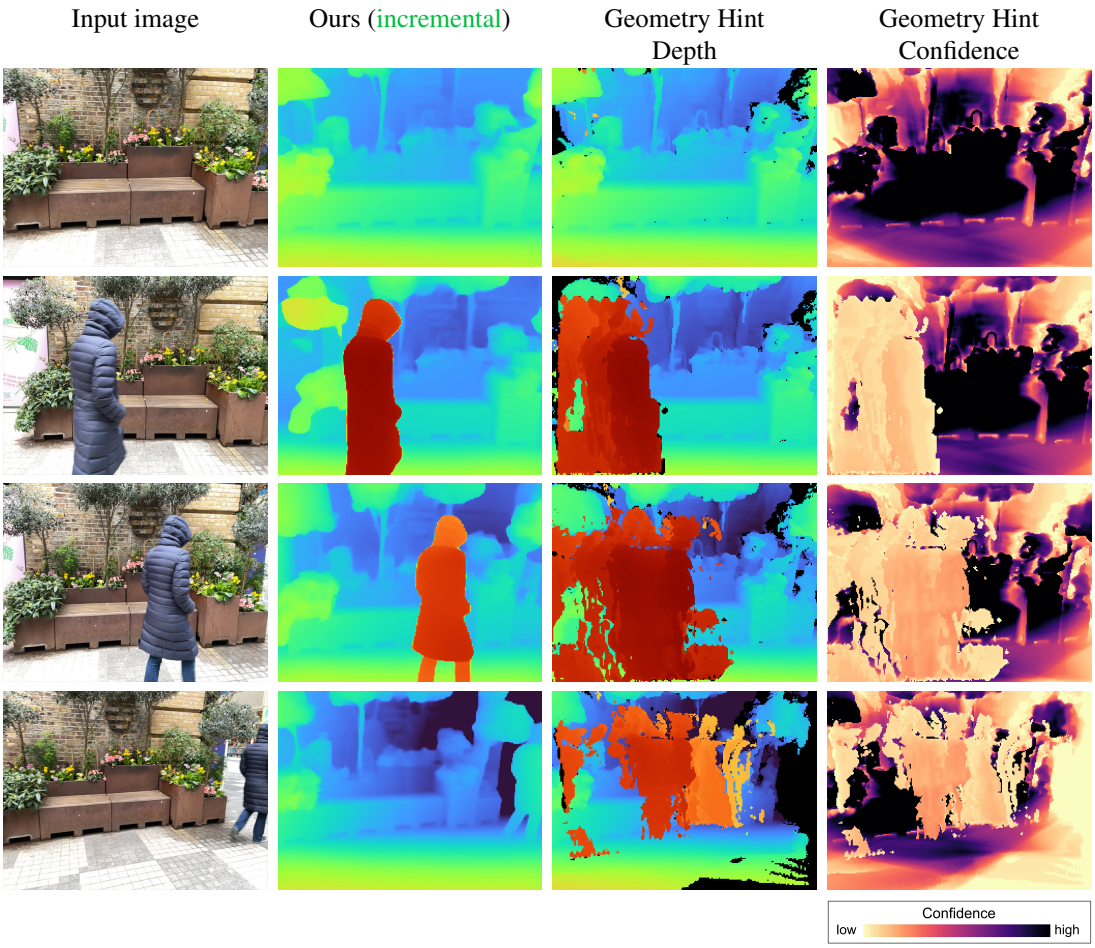

Figure 2. **Qualitative results for a scene with moving objects**

static scene and the moving person in the scene. As the person moves through the scene, the rendered depth map from the geometry hint has some artefacts and is unreliable in areas the moving object has recently visited. This unreliability is reflected in the confidence we also provide to the network. Our final prediction from our network ignores the poor quality regions of the hint and produces good quality depths. Please see our supplementary video for the full sequence.

2.2. Additional ablation results and details

Table 2 extends the equivalent ablation table in the main paper, with additional rows and columns. We include a new row , giving an additional ablation in the spirit of [10]. We also include rows K, L and M, ablating our 'offline' method. We also include an additional column, giving logRMSE scores (omitted from the main paper due to space and readability constraints).

Here we give more details of our 'in the sprit of' ablations:

Ablation details for row I – Ours (incremental, fast). For this version of our model, we substitute some of the components with lighter and faster networks. We use a slimmed down decoder with simpler skip connections and a ResNet18 for the image-prior encoder.

Ablation details for row – [10]. [10] describe a method for modulating a cost volume given access to ground truth point clouds at 3% sparsity when rendered down. We use the same modulation strategy in place of out *Hint MLP* in our model and train it on the same corpus of rendered mesh data. We use the same parameters for gaussian modulation along a ray as described in the paper and codebase, and apply it on every ray on output from the matching MLP. We test this model in both offline and online modes. We observe that given no access to a confidence on the rendered depth map, the model does not learn to maximize the use of the incoming hint, falling behind our model in both modes.

Ablation details for row E – [18]. In this ablation experiment, we replace our *Hint MLP* with a strategy inspired by SimpleMapping [18]. Specifically, the proposed hint MLP **is removed**, and our rendered geometry hint depth maps are first processed by a depth completion network [17], which

			Abs Diff↓	Abs Rel↓	Sq Rel↓	RMSE↓	logRMSE↓	$\delta < 1.05$↑	$\delta < 1.25$↑
Online	A	**Ours** without Hint MLP (as in SimpleRecon [13])	.0873	.0430	.0128	.1483	.0685	74.12	98.05
	B	**Ours** w/ hint, without confidence	.0863	.0392	.0129	.1529	.0665	77.81	98.02
	C	**Ours** w/ hint & confidence to cost volume encoder	.0890	.0438	.0130	.1506	.0695	73.39	97.95
	D	**Ours** w/ warped depth as hint	.0889	.0436	.0130	.1492	.0689	73.26	98.06
	E	**Ours** w/ hint-based variable depth planes [18]	.0821	.0405	.0121	.1432	.0664	76.67	98.19
	E	SimpleRecon [13] w/ TOCD [6]	.0880	.0437	.0134	.1505	.0702	74.19	97.83
		Ours w/ hint cost volume modulation [10]	.0773	.0372	.0112	.1381	.0632	79.52	98.34
	F	**Ours** w/ single MLP for matching and hints	.0773	.0371	.0112	.1381	.0629	79.56	98.35
	G	**Ours** (no hint on inference)	.0870	.0428	.0128	.1477	.0683	74.35	98.02
	H	**Ours** (incremental, fast)	.0826	.0400	.0125	.1473	.0679	77.14	98.05
	I	**Ours** (incremental)	.0767	.0369	.0112	.1377	.0629	79.93	98.35
Offline	J	SimpleRecon [13] (offline)	.0812	.0399	.0118	.1413	.0652	77.02	98.26
	K	**Ours** w/ hint, without confidence	.0635	.0309	.0092	.1228	.0566	86.08	98.64
	L	**Ours** w/ hint cost volume modulation [10]	.0651	.0320	.0094	.1242	.0575	84.92	98.61
	M	**Ours** (offline)	.0627	.0306	.0092	.1225	.0565	86.46	98.62

Table 2. **Ablation evaluation.** Scores are depth metrics on ScanNetV2. This table is a copy of the table in the main paper but with the addition of the logRMSE metric, plus rows K, L, M, and N. and . (We use for the additional row with cost volume modulation so letters remain consistent between the main paper and supplementary). See the text for descriptions of these variants.

Training data generation strategy	Abs Diff↓	Abs Rel↓	Sq Rel↓	RMSE↓	logRMSE↓	$\delta < 1.05$↑	$\delta < 1.25$↑
Ground truth as hint	.0954	.0488	.0176	.1637	.0760	72.13	97.26
Full mesh renders	.0789	.0382	.0113	.1391	.0641	78.29	98.27
Full and partial mesh renders	.0767	.0369	.0112	.1377	.0629	79.93	98.35

Table 3. **Comparison of different strategies to generate training data for the geometry hint.**

generates dense depth maps. Then, a SimpleRecon model exploits this depth map to build a cost volume: for each pixel, 64 depth plane hypotheses are linearly spaced around the depth value indicated by the dense depth map, using a 0.04m space interval. Moreover, features from the depth completion network are concatenated to SimpleRecon as described in [18]. The two networks are trained sequentially: first, we train the depth completion network from [17] to complete depth maps rendered from the mesh (which generally contain holes due to occlusions, camera motion etc); then we train SimpleRecon keeping the depth completion network frozen. At training time, we train both the model adopting the same hints and augmentation strategies used to train our proposed model, enforcing the losses described in [18] and [13] to the depth completion network and SimpleRecon respectively. It is worth noting that, differently from SimpleMapping, at inference time our depth hints are not inferred by a sparsified ground truth; instead they are directly rendered from the predicted mesh. Moreover, compared to our proposed strategy, this solution requires an additional encoder-decoder network.

Ablation details for row F – SimpleRecon [13] w/ TOCD [6]. Recall that TOCD [6] uses a point cloud of the scene to process per frame depth estimates into temporally consistent depth estimates. In this ablation experiment we use the reference implementation of TOCD [6] and the provided scene point clouds. We use SimpleRecon depth maps as input to their pipeline instead of the original DPT [11] depth maps, which are less accurate since DPT is a monocular depth prediction system. We use the pretrained models provided by the authors along with predictions from SimpleRecon. In the original implementation, the input depths to the temporal model are scaled and shifted using the ground-truth depths. However, we use predicted depths for such scaling and shifting instead.

Ablation on different strategies to generate training data. In table 3 we present results for an additional comparison of different strategies to generate training data. In particular, we compare the performance of different models that were trained using different depth hints. *Ground truth as hint* receives the ground truth depth map as hint at training time. *Full mesh renders* uses depth renders of meshes covering the full scenes and *Full and partial mesh renders* uses both depth renders of full scenes and depth renders of partial scenes. For visualizations of full and partial renders see figure 8. All models are evaluated in incremental mode. The results validate our choice of using both full and partial depth renders as our training data generation strategy.

2.3. Additional results and explanations for 3RScan

Table 4 is a copy of Table 5 in the main paper, but with the additional metrics of Abs Rel and logRMSE. The first two rows in this table present simple baselines we devised to generate depth for the current camera viewpoint. Like 'ours', these baselines assume we have previously visited a location and generated geometry at that location. Also like

	Abs Diff↓	Abs Rel↓	Sq Rel↓	RMSE↓	logRMSE↓	$\delta < 1.05$↑	$\delta < 1.25$↑
Rendered depth from TSDF	.2506	.2021	.1293	.3338	.2711	23.53	67.97
Densified rendered depth	.1763	.1499	.0629	.2264	.1706	30.61	81.69
SimpleRecon [13]	.1350	.1025	.0437	.1879	.1308	46.88	89.32
Ours (no hint)	.1346	.1026	.0449	.1879	.1312	47.28	89.39
Ours (incremental)	.1255	.0948	.0395	.1787	.1258	48.76	90.47
Ours (revisit)	.1182	.0892	.0368	.1710	.1208	50.24	91.78
Ours (revisit, pose noise)	.1199	.0908	.0372	.1725	.1222	49.46	91.56

Table 4. **Long-term hints using 3RScan – full metrics**. This table is a copy of Table 5 in the main paper, with the addition of Abs rel and logRMSE metrics.

	Abs Diff↓	Sq Rel↓	RMSE↓	$\delta < 1.05$↑	$\delta < 1.25$↑
Ours (revisit)	.0911	.0138	.1518	70.13	97.58
Ours (revisit Ace [2] poses)	.0912	.0137	.1510	69.80	97.64
Ours (no hint)	.1088	.0166	.1678	61.02	97.07

Table 5. **Long-term hints using 7Scenes**. Here we compare Ours (revisit) with a variant where alignment between scans is estimated using a the relocalization algorithm Ace [2]. Notice that there is only a very slight degradation in depth metrics, highlighting the robustness of our method to noisy estimates of pose.

'ours', the baselines assume the current sequence is captured in the same camera coordinate frame as the previous visit to the location. We now describe these baselines in more details:

Rendered depth is a baseline which simply renders a depth map from the previous geometry from the current camera viewpoint, and uses this as the final depth prediction. Some pixels may not have a valid depth generated, for example if a surface wasn't viewed in the previous visit to this location. For these pixels, we interpolate new values based on neighboring valid pixels.

Densified rendered depth solves the missing pixel problem in a more involved way, where we complete these missing pixels via a machine-learned densification network. For fairness with 'ours', we implement a densification network based on [13]. This network is trained to take as input an incomplete depth map, where missing pixels are give a label -1, plus an RGB image. The network is trained to output a complete depth map. This network is trained and evaluated in monocular mode, so does not include a cost volume. This network is trained on ScanNetV2 using the same set of partial and full mesh depth renders we use train our models.

Further details on 'Ours (revisit, pose noise)' experiment. For this experiment, we add $\epsilon_t \sim \mathcal{N}(0, 4\text{cm})$ to each component of translation, and $\epsilon_r \sim \mathcal{N}(0, 0.02^2)$ to each component of rotation.

2.4. Revisit on 7Scenes with relocalization

In Table 5 we show further results for the revisit variant of our method, this time using the 7Scenes dataset. Here, we simulate a real use case whereby a user has previously scanned a scene, and then returns to the same scene and relocalizes using [2]. In practice, we first build a TSDF using a training sequence, before running [2] on each test image to a estimate per frame pose. These predicted poses are then used to render our hints for input to our model. Notice that even with noisy estimated poses from a relocalization algorithm (row 2), our method still performs very competitively when compared to using ground truth poses (row 1).

Note that for this experiment we use the test split from [2], which differs from test split used for the results in the main paper.

2.5. Ours offline using alternative methods as hints.

In Table 6 we show results for the offline variant of our model, where the initial TSDF is built using different methods, and the *second pass* is run as normal using our network, where we render the extracted mesh into each frame to generate a hint. In each case our model performs well, greatly improving over the initial predictions. This demonstrates that our method is robust to where the hints come from, handling hints which differ to those seen during training.

The best performing hybrid method, **Ours** + FineRecon [14] as hint, surpasses all other baselines on all metrics. While in practice this system may be difficult to deploy and is computationally expensive, we are excited that our system can work in combination with FineRecon [14] to generate a new state-of-the-art. Since FineRecon [14] estimates the TSDF directly using a 3D CNN there are no TSDF confidences available, and so we set confidence to 1.0 for pixels with a valid rendered depth value, and 0 otherwise. Additionally, depth scores are obtained by rendering the final predicted mesh into each frame and comparing to ground

	Abs Diff↓	Abs Rel↓	Sq Rel↓	$\delta < 1.05$↑	Chamfer↓	F-Score↑
DeepVideoMVS[4]	.1186	.0583	.0190	60.20	6.73	.579
Ours + DeepVideoMVS[4] as hint	.0667	.0330	.0098	84.09	5.12	.700
SimpleRecon[13] (offline)	.0812	.0399	.0118	77.02	5.05	.687
Ours + SimpleRecon[13] as hint	.0627	.0306	.0092	86.46	4.50	.738
FineRecon[14]	.0812	.0389	.0265	85.60	4.99	.710
Ours + FineRecon[14] as hint	.0574	.0279	.0089	89.03	4.15	.772

Table 6. **Ours offline with alternative hints.** We evaluate our model in offline mode, using alternative methods to generate our hints. In all cases, we greatly improve over the initial predictions.

Figure 3. **Qualitative Comparison of NeRFs on a ScanNetV2 scene.** A NeRF trained using our depths as input produces higher quality novel views when compared with the vanilla baseline. Note the fixed halo artifacts on the wall in the first row, on the couch in the second row, and on the wall to the right of the bookshelf in the third row.

truth depth maps.

Note that all the '**Ours** +' rows in the table use the same model used for **Ours** in the main paper, without retraining. The only difference vs. the main paper is the geometry used for the hint.

2.6. Benefits of our depths for NeRF reconstructions

In this section we present an interesting application of our depth maps for improving NeRF renders. NeRFs [7] are a ubiquitous representation for image-based rendering. NeRFs can though suffer from artefacts. To help to reduce artefacts or to improve the underlying geometry, depths can be used in training [3, 12]. Here, we use a nerf variant 'nerfacto' (from NerfStudio [16]) to train a NeRF on a ScanNetV2 sequence, and show that using our depths can help to reduce artefacts in the final renders.

Approach First, we cropped the ScanNetV2 images (and their corresponding intrinsics) to remove the black border pixels from the rectification. We create a dataset from the images where we have predictions from our offline (two pass method). We then trained a NeRF with 90% of the images as training images. We trained three NeRF variants, each using the same RGB images:

- **Vanilla** NeRF, using the nerfacto default arguments.
- **w/ GT Depth**, where we pass ground truth depth maps to enable a better NeRF representation.
- **w/ Ours offline**, where we pass depth maps estimated from our offline (two-pass) reconstruction.

For all variants, we render novel views using 5% of the images, sampled from the images not in the training set.

Results Some example renders are shown in Figure 3, where we observe that our novel views from the NeRF regularized using our depths yields the same qualitative improvement as from the ground truth depth.

2.7. Additional qualitative depth map comparisons

Figure 4 shows additional depth results comparing our incremental approach to depth maps from [13] and [4]. As with Figure 5 in the main paper, here we see the better fidelity to the ground truth, better small details, and better overall scene accuracy using our method. Figure 5 shows the benefit that ours (offline) brings over ours (incremental).

2.8. Additional qualitative results on out-of-distribution scenes

Figure 6 gives additional results on out-of-distribution sequences. These results use our ScanNetV2-trained model, and show results when evaluated in offline mode. We observe that our model transfers well to these new scenes. Figure 7 compares our meshes with those produced from FineRecon [14]. We note that our meshes are cleaner, more accurate and have less unwanted background noise vs. [14].

3. Full architecture details

As described in the main paper, with the exception of our 'Hint MLP' our network follows the architecture of [13]. For completeness, we give a brief overview of this architecture here:

The decoder is based on UNet++ [20]. Each block is a residual `BasicBlock` from [5], each comprising 256, 128, 64, and 64 channels. Activations are LeakyRelus with a slope of 0.2. The image prior encoder uses an EfficientNetV2 [15], while the matching feature encoder uses the first two blocks of ResNet18 [5]. The cost volume and image feature encoder is based on [4]. These also use `BasicBlocks`, with 64, 128, 256 and 384 channels per layer.

The **matching MLP** has an input size of 202 dimensions, followed by two hidden layers of 128 dimensions each, then an output neuron of dimension 1. Our **hint MLP** accepts an input with three dimensions, followed by two hidden layers of 12 dimensions each, then an output neuron of dimension 1.

Importantly, both of our modes (incremental and offline) use the same network weights.

4. Runtime and memory use

The main paper gives the overall timings of our approach vs. some competing depth and reconstruction systems. Table 7 gives a more detailed breakdown of the timings for various parts of our system in incremental mode. Table 8 gives a more detailed breakdown of the timings for various parts of our system in offline mode.

Per frame incremental operations	Time per frame (ms)
Marching cubes	9.4
Mesh depth render	9.2
Confidence sampling	0.69
Model forward pass	52.8
TSDF fusion	4.54
Total time	**76.63**

Table 7. **Incremental timings** Average per frame timings for the ScanNetV2 test set on an Nvidia A100, when run in incremental mode with batch size 1.

Per scene offline operations	Time per scene (s)
First stage model forward pass	8.75
First stage TSDF fusion (4cm)	0.66
Marching cubes (run once)	0.01
Second stage mesh depth render	1.34
Second stage confidence sampling	0.17
Second stage model forward pass (with caching)	1.88
Second stage TSDF Fusion (2cm)	1.04
Total time	**13.9**

Table 8. **Offline timings** Average per scene timings for the ScanNetV2 test set on an Nvidia A100, when run offline with batch size 8.

Memory use The additional memory required of our method compared to SimpleRecon [13] is relatively small. For our TSDF reconstructions we store two float16s per voxel, one for the TSDF value and one for the confidence. At 2cm voxel resolution, the TSDF for the largest scene in Scannet is 148MB of memory, and the extracted mesh is 9MB.

For very large scenes we use voxel hashing [9], based on the implementation in Open3D [19]. Voxel hashing is highly memory efficient, only storing data for voxels which lie inside the observed truncation band (we use a band of 6cm). On average, voxel hashing requires roughly 6% of the memory of dense volumes [9].

5. Training Data

Section 3.5 in the main paper gives an overview of the different hints we generate for training data. Figure 8 shows examples of the full and partial training hints we generate at training time.

6. Introspection – cost volume visualizations

Figure 9 gives some introspection into why our method outperforms SimpleRecon [13]. Like Figure 5 in the main paper, Figure 9 shows that our predictions are better than [13]. However, in this figure we additionally visualize the winner-takes-all depth map from the cost volume. To generate this visualization, we take the final $D \times H \times W$ cost

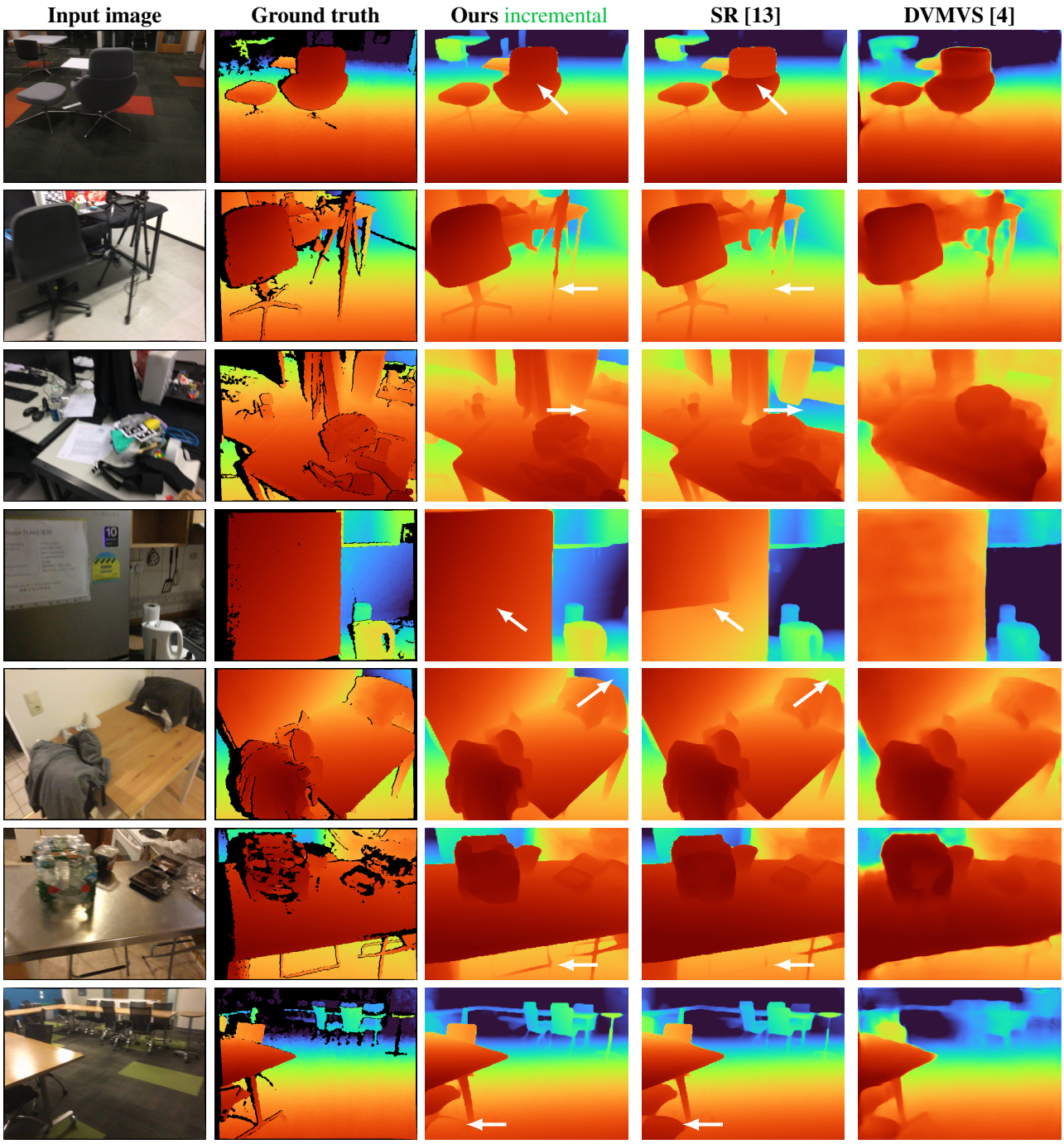

Figure 4. **Additional qualitative depth results on ScanNetV2 for incremental**

volume (see *e.g.* Figure 3 in the main paper), and take an `argmax` along the depth dimension. This gives an $H \times W$ image, indicating the depth bin in the cost volume which had the 'best' match between source and target views. We notice that our winner-takes-all depth map, which is boosted by our geometry hint, is significantly more clear and similar to the ground truth than the winner-takes-all depth map from [13]. Figure 10 shows the equivalent visualization for offline mode; again, we see the benefits that our cost volume construction brings.

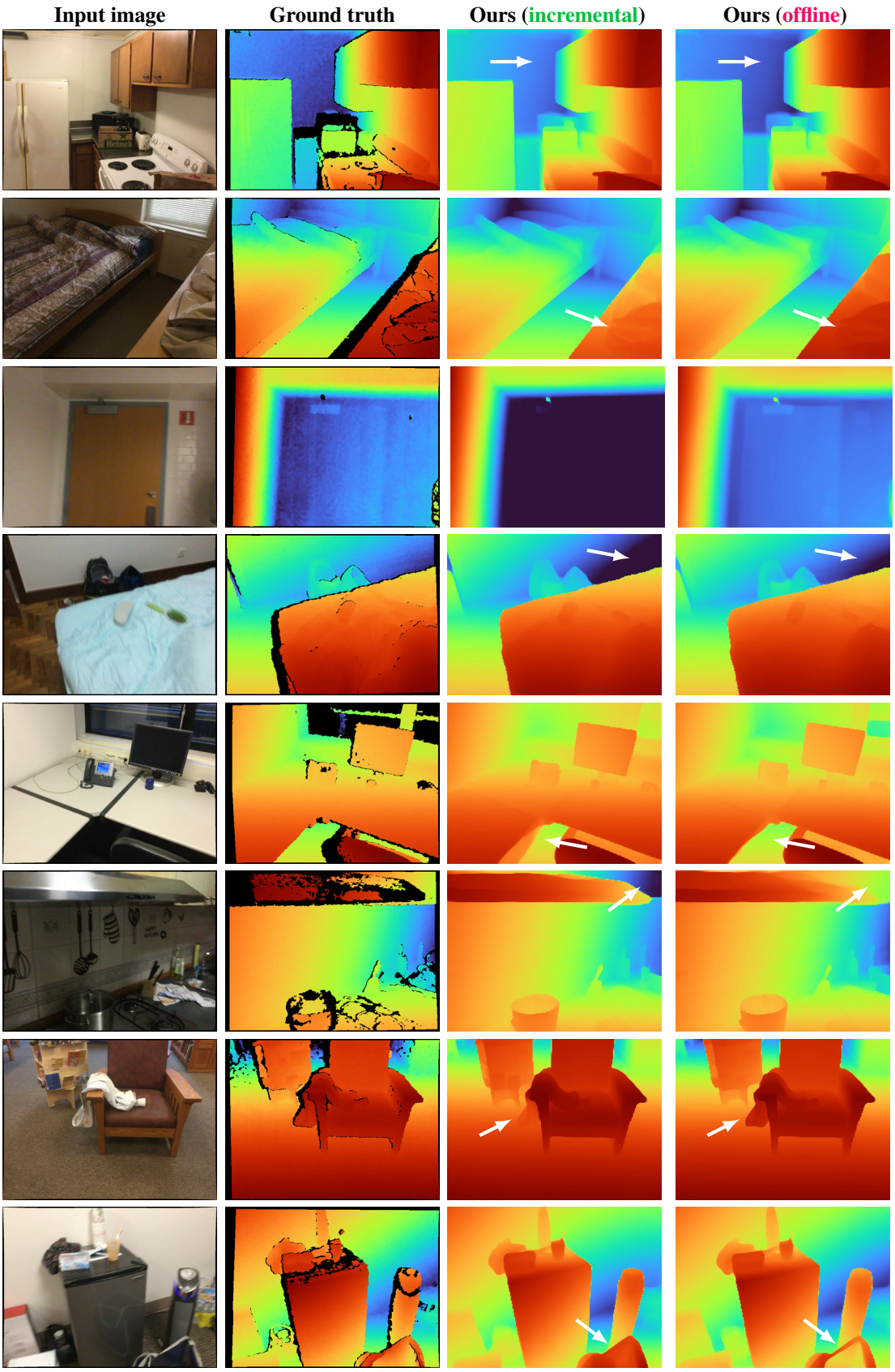

Figure 5. **Qualitative depth results ours incremental vs ours offline.** Note, most frames are at the very beginning of the sequence. The first four images show examples where ours (offline) gives overall better scale prediction. The last four images show examples where ours (offline) improves small details.

Figure 6. **Other examples of out-of-distribution scenes.** On the left we show some example frames from the input sequence, which was casually captured with a smartphone. On the right we show renders of the resultant mesh. As we use TSDF fusion, we can easily encode extra channels such as RGB as shown here.

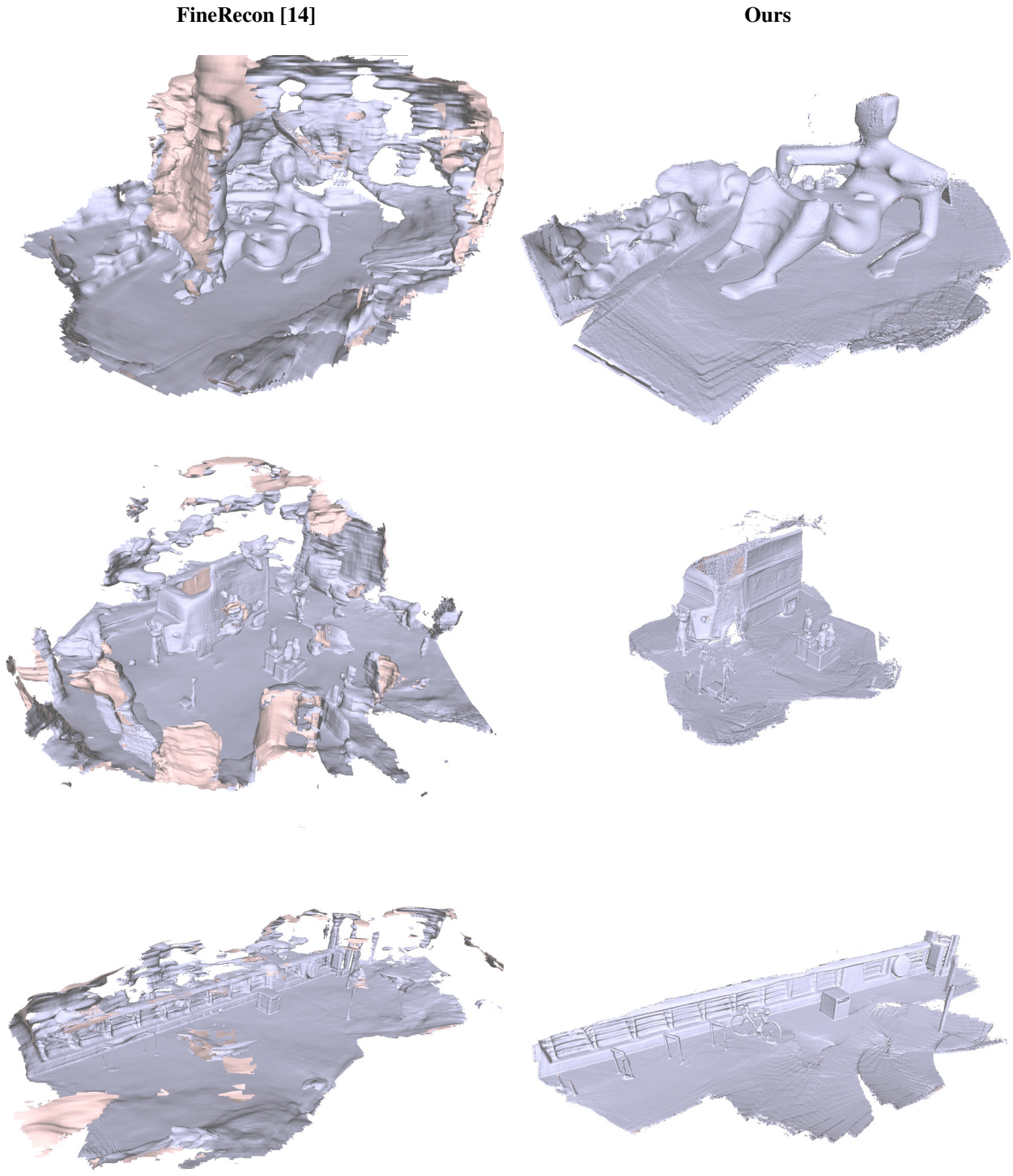

Figure 7. **Comparing our performance on out-of-distribution scenes against [14]**. We show meshes here **untextured**, as this allows easier inspection of the geometry quality. Please see Figure 6 here and Figure 9 in the main paper for some example RGB input images for these scenes. We ran FineRecon at a 6cm voxel size. When we tried running at an effective resolution of 1cm as in the paper, we ran out of memory with a request of 205GBs above what our A100 GPUs capacity of 40GBs.

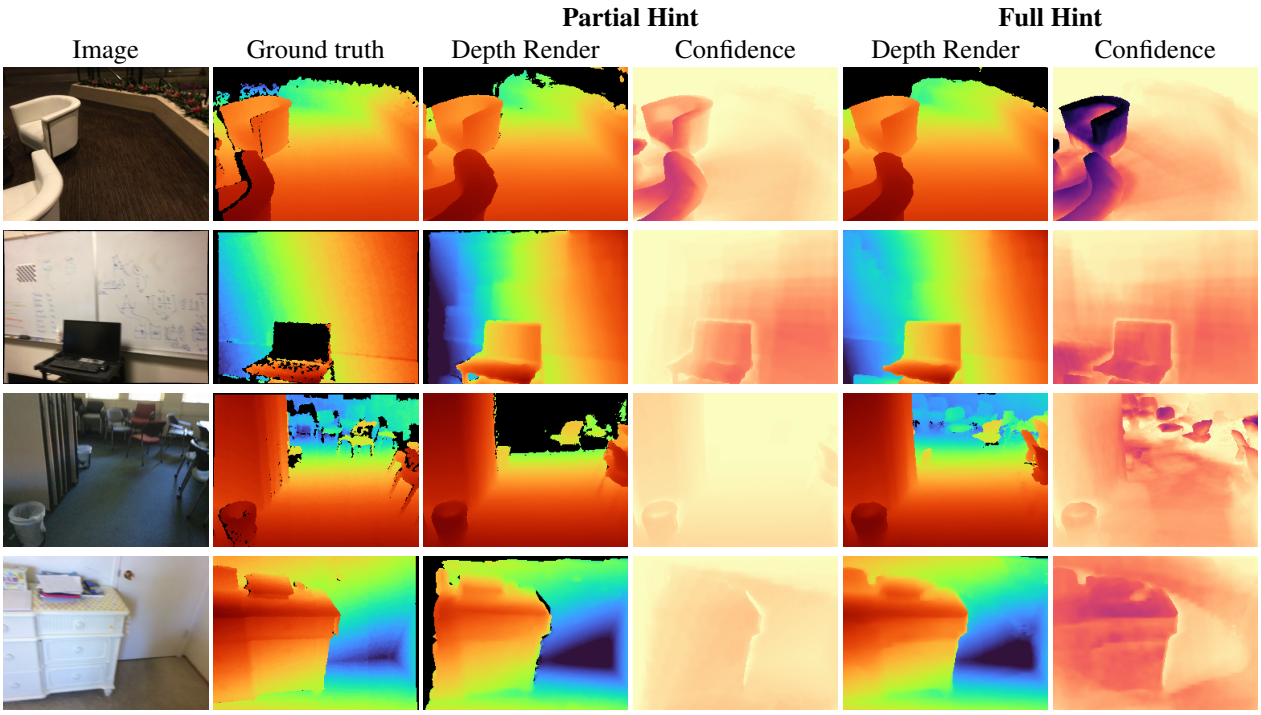

Figure 8. **Examples of training data.** *Partial hints* are generated from partial TSDFs, generated only using frames up to the current training frame. *Full hints* are generated from a full TSDF, assuming all frames in the scene have been observed.

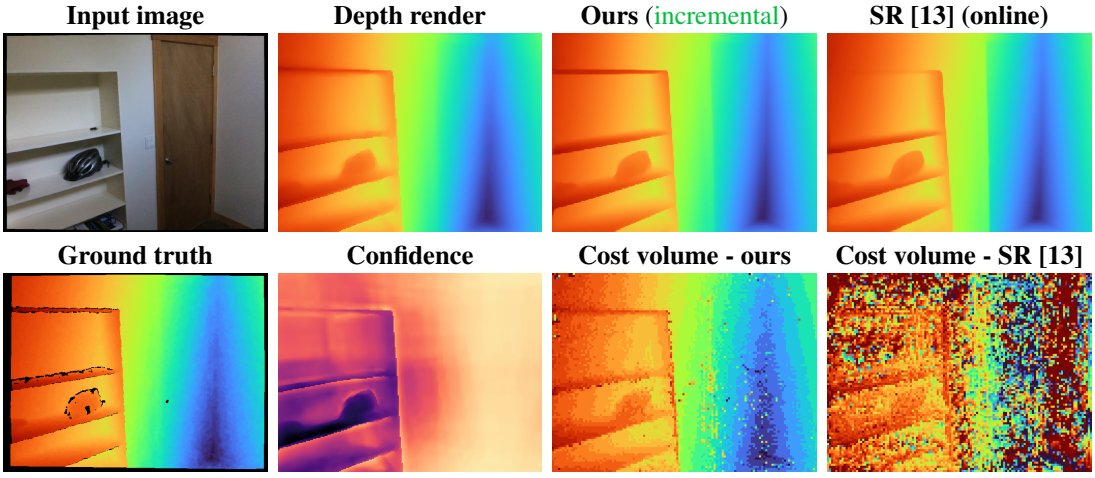

Figure 9. **Cost volume visualizations for online methods.** We compare the winner-takes-all depth map from the cost volume obtained by our method and SimpleRecon [13], both in online mode. Ours is less noisy and looks more similar to the ground truth depth map.

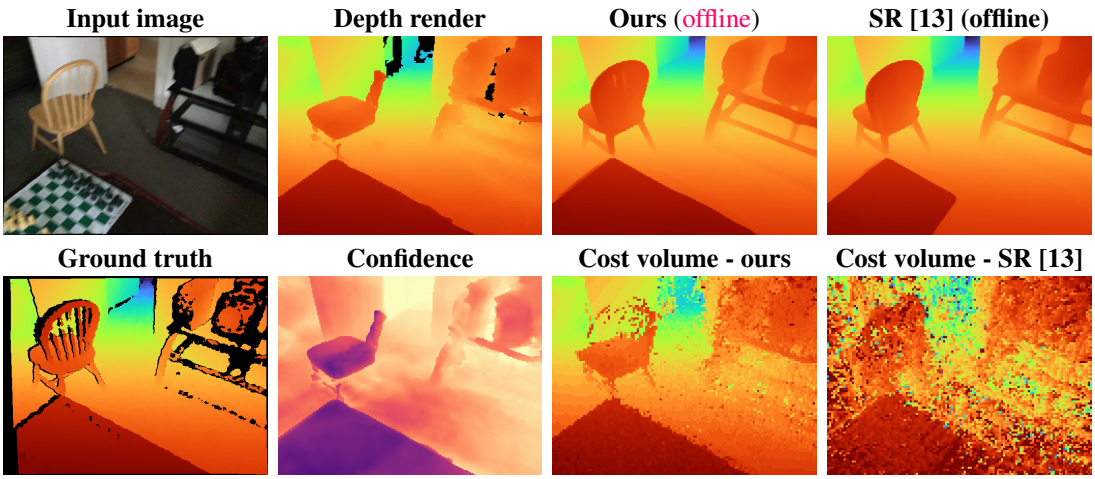

Figure 10. **Cost volume visualizations for offline methods.**

References

[1] Aljaz Bozic, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. TransformerFusion: Monocular RGB scene reconstruction using transformers. *NeurIPS*, 2021. 1, 2

[2] Eric Brachmann, Tommaso Cavallari, and Victor Adrian Prisacariu. Accelerated coordinate encoding: Learning to relocalize in minutes using RGB and poses. In *CVPR*, 2023. 5

[3] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*, 2022. 6

[4] 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. 6, 7, 8

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[6] Numair Khan, Eric Penner, Douglas Lanman, and Lei Xiao. Temporally consistent online depth estimation using point-based fusion. In *CVPR*, 2023. 4

[7] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 6

[8] 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. 1

[9] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 2013. 7

[10] Matteo Poggi, Andrea Conti, and Stefano Mattoccia. Multi-view guided multi-view stereo. In *IROS*, 2022. 3, 4

[11] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021. 4

[12] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, 2022. 6

[13] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. SimpleRecon: 3D reconstruction without 3D convolutions. In *ECCV*, 2022. 1, 4, 5, 6, 7, 8, 12, 13

[14] Noah Stier, Anurag Ranjan, Alex Colburn, Yajie Yan, Liang Yang, Fangchang Ma, and Baptiste Angles. Finerecon: Depth-aware feed-forward network for detailed 3D reconstruction. In *ICCV*, 2023. 5, 6, 7, 11

[15] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *ICML*, 2021. 7

[16] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 6

[17] Alex Wong and Stefano Soatto. Unsupervised depth completion with calibrated backprojection layers. In *ICCV*, 2021. 3, 4

[18] Yingye Xin, Xingxing Zuo, Dongyue Lu, and Stefan Leutenegger. SimpleMapping: Real-Time Visual-Inertial Dense Mapping with Deep Multi-View Stereo. In *ISMAR*, 2023. 3, 4

[19] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 1, 7

[20] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. UNet++: A nested U-Net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, 2018. 7