AirPlanes: Accurate Plane Estimation via 3D-Consistent Embeddings

Supplementary Material

Contents

A. Additional results

A.1. Full geometry metrics

In Table A1 we display scores from the main paper with the complete set of geometry metrics from [4]. As mentioned in the paper, our geometry scores are comparable with *SR [5] + RANSAC*, which uses the same underlying geometry. *FineRecon [7] + RANSAC* is a more computationally expensive method which scores better in the completion and recall metrics.

A.2. Results using PlanarRecon evaluation code

Test split we use in the main paper. We train and evaluate on ScanNetV2 [1], because [3] provided ground truth plane annotations for most of it. However, plane annotations are unavailable for the ScanNetV2 test set. For our **main paper results** we therefore split the official ScanNetV2 validation set into new *plane evaluation* validation and test splits, dubbed $\text{val}^{\text{planes}}$ and $\text{test}^{\text{planes}}$, with 80 and 100 scenes respectively. For a fair comparison with prior work, in the main paper we re-evaluate baselines on our new test split, using the methodology as described in the main paper e.g., removing unseen voxels to avoid penalizing methods such as [4].

Evaluation from [12]. For full transparency, we additionally adapted the evaluation code from [12] and evaluated a selection of models using their code and their data splits. Table A2 shows these results, i.e., models evaluated on the evaluation code and dataset split from [12]. This code uses the full ScanNetV2 validation set of 312 scenes for the test set. We make three observations from these results:

1. In some cases, the scores presented in the paper [12] do not match the scores we achieved from their online code implementation[1]. We highlight these discrepancies in Table A2. For example, the segmentation results for the 'NeuralRecon + RANSAC' baseline result in superior scores using the author's code compared to those in the paper.
2. Our RANSAC implementation is slightly better than but generally comparable to the results from [12]'s code, e.g., comparing the row 'NeuralRecon + RANSAC (Rerunning [12]'s GitHub code)' vs. the row NeuralRecon + RANSAC (Our RANSAC implementation)'.
3. Our method is the top-performing method across all metrics using their benchmarking protocol. For example we score 2.713 VOI vs the closest competitor (NeuralRecon + RANSAC) which scores 3.512. For transparency we note though that the hyperparameters of our method were tuned on $\text{val}^{\text{planes}}$, the scenes from which are present in the test set used in the Table A2 scores. The results in our main paper are reported on a held out test set. Nonetheless, the difference in scores between our method and the closest competitors here are large.

A.3. Ablation of the embedding dimension

An important component of our method is the MLP that learns 3D consistent plane embeddings. For all results in the main paper, we use an embedding space with 3 dimensions. In Table A3 we compare results using different dimensions for the embedding space. The results show that the embedding MLP and resulting performance is not strongly impacted by the number of dimensions, meaning we are robust to this hyperparameter. We note though that our method performs slightly better for a higher number of dimensions at the cost of a higher time to train the MLP.

A.4. Scalability of meanshift clustering to large scenes

To evaluate the scalability of our current clustering approach on large scenes, we created a large 3D scene by concatenating all 100 test scenes from ScanNet. This cre-

[1] https://github.com/neu-vi/PlanarRecon

	Geometry						Segmentation			Planar		
	comp ↓	acc ↓	chamfer↓	prec↑	rec↑	f1 ↑	voi↓	ri↑	sc↑	fidelity ↓	accuracy↓	chamfer↓
PlaneRecTR [6] + aggregation	5.71	11.92	8.82	32.32	64.31	42.53	4.028	0.924	0.268	22.58	15.71	19.14
Atlas [4] † + RANSAC	19.98	5.31	12.65	65.26	46.74	53.71	2.868	0.932	0.465	22.60	17.71	20.16
NeuralRecon [8] + RANSAC	10.16	7.84	9.00	44.09	50.80	46.91	3.176	0.929	0.391	16.76	13.08	14.92
FineRecon [7] + RANSAC	**4.59**	6.53	5.56	59.00	**70.55**	64.10	2.377	0.950	0.531	**7.74**	11.71	9.72
SR [5] + RANSAC	6.09	**4.71**	5.40	**70.31**	61.38	**65.45**	2.507	0.946	0.515	9.42	10.13	9.78
PlanarRecon [12]	12.29	7.49	9.89	47.39	40.50	43.47	3.201	0.919	0.405	18.86	16.21	17.53
Ours	5.74	4.86	**5.30**	67.79	62.42	64.92	**2.268**	**0.957**	**0.568**	8.76	**7.98**	**8.37**

Table A1. **ScanNetV2 evaluation.** These results are equivalent to Table 1 in the main paper but we additionally report the complete set of geometry metrics from [4].

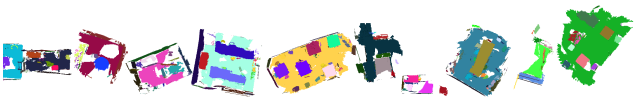
		Geometry					Segmentation		
		comp ↓	acc ↓	prec ↑	rec ↑	fscore ↑	voi↓	ri↑	sc↑
PlanarRecon	Scores as published in [12]	0.154	0.105	0.398	0.355	0.372	3.622	0.897	0.248
PlanarRecon	Re-running [12]'s GitHub code	0.147	0.100	0.404	0.360	0.378	3.598	0.897	0.250
NeuralRecon + RANSAC	Scores as published in [12]	0.144	0.128	0.306	0.296	0.296	8.087	0.828	0.066
NeuralRecon + RANSAC	Re-running [12]'s GitHub code	0.143	0.119	0.317	0.278	0.294	4.405	0.902	0.212
NeuralRecon + RANSAC	Our RANSAC implementation	0.126	0.094	0.396	0.435	0.410	3.512	0.917	0.237
Ours	Our method	**0.072**	**0.067**	**0.626**	**0.554**	**0.586**	**2.713**	**0.935**	**0.352**

Table A2. **Results using PlanarRecon's [12] evaluation code on the entire validation set.** We display scores reported in the original paper from [12] and the scores we obtain when running the public code from [12]. Finally, we also report our implementation of sequential RANSAC and our method. All the scores here are computed using the 312 validation sequences defined in [1], using the benchmark code provided by [12]. **Best** and second best overall scores are highlighted in the table. Some results are highlighted in red to draw attention to cases with the largest discrepancies between the numbers reported in [12] and the results from running the public code.

Dimension of embedding space	Geometry		Segmentation			Planar		
	chamfer ↓	f1 ↑	voi↓	ri↑	sc↑	fidelity↓	accuracy↓	chamfer↓
2	0.054	0.645	2.288	0.957	0.566	8.74	8.02	8.38
3 ← in main paper	0.053	0.649	2.268	0.957	0.568	8.76	7.98	8.37
4	0.053	0.649	2.257	0.958	0.572	8.59	8.02	8.31
5	0.053	0.650	2.253	0.958	0.574	8.60	7.82	8.21

Table A3. **Embedding dimension ablation.** We compare the performance of our method when using a different number of dimensions for the MLP embedding space.

ates an approx. 2500 square meter scene. The figure below shows a small portion of this scene. Running our meanshift based clustering on this large scene takes 360ms, indicating our clustering can cope with reasonably large scenes. We observed that the number of anchors does not need to be increased linearly with the size of the scene, as the meanshift step is followed by connected components.



A.5. Qualitative results on ScanNetV2

Figs. A1 and A2 illustrate further qualitative comparisons between our method, our Sequential RANSAC implementation, and PlanarRecon [12]. We again note how our approach has closer fidelity to the ground truth compared to the other methods. In the same figures we also show comparisons between our method and results from other ablations and baselines from main paper.

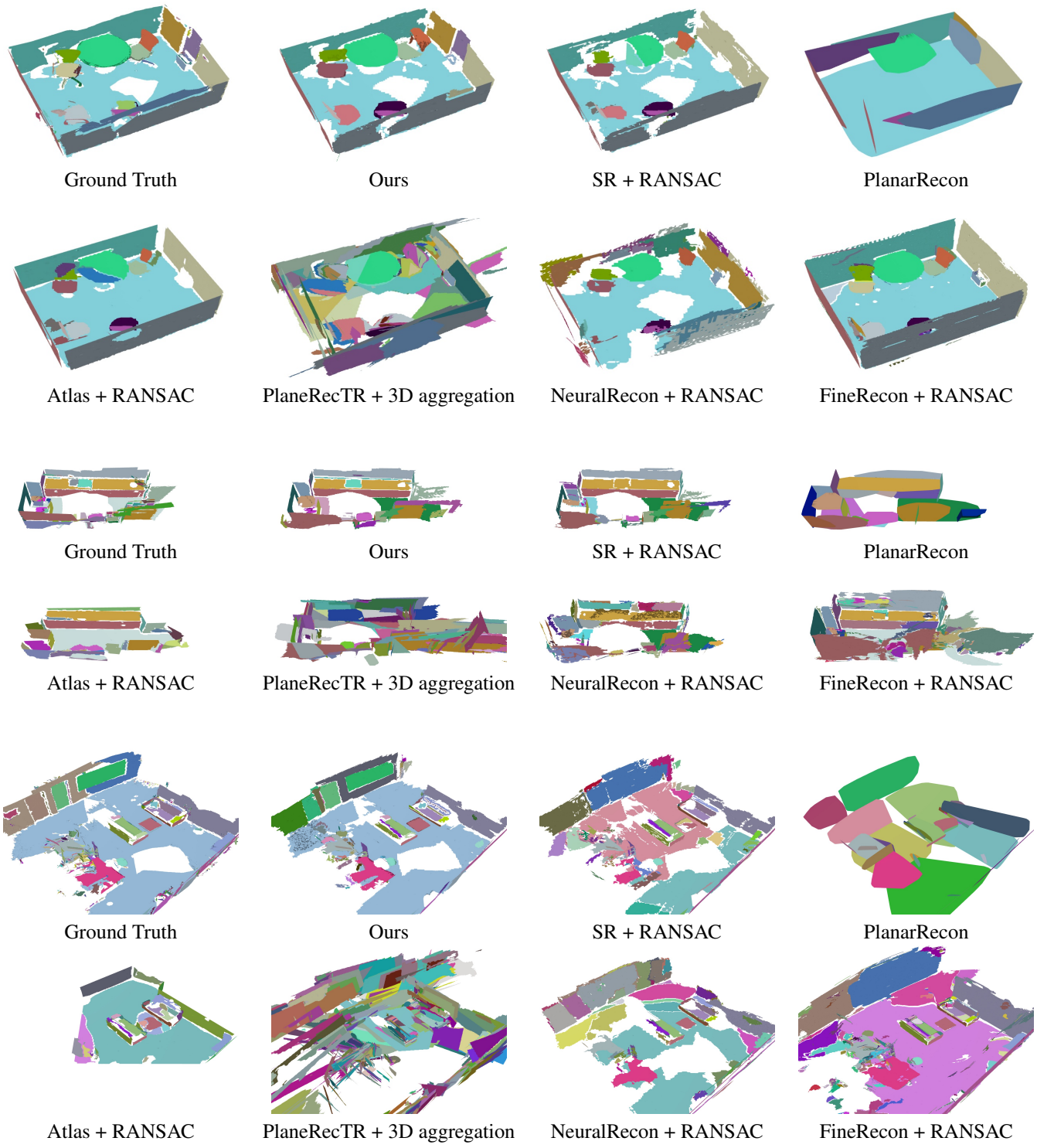


Figure A1. **Comparisons with more baselines and ablations on ScanNetV2.**

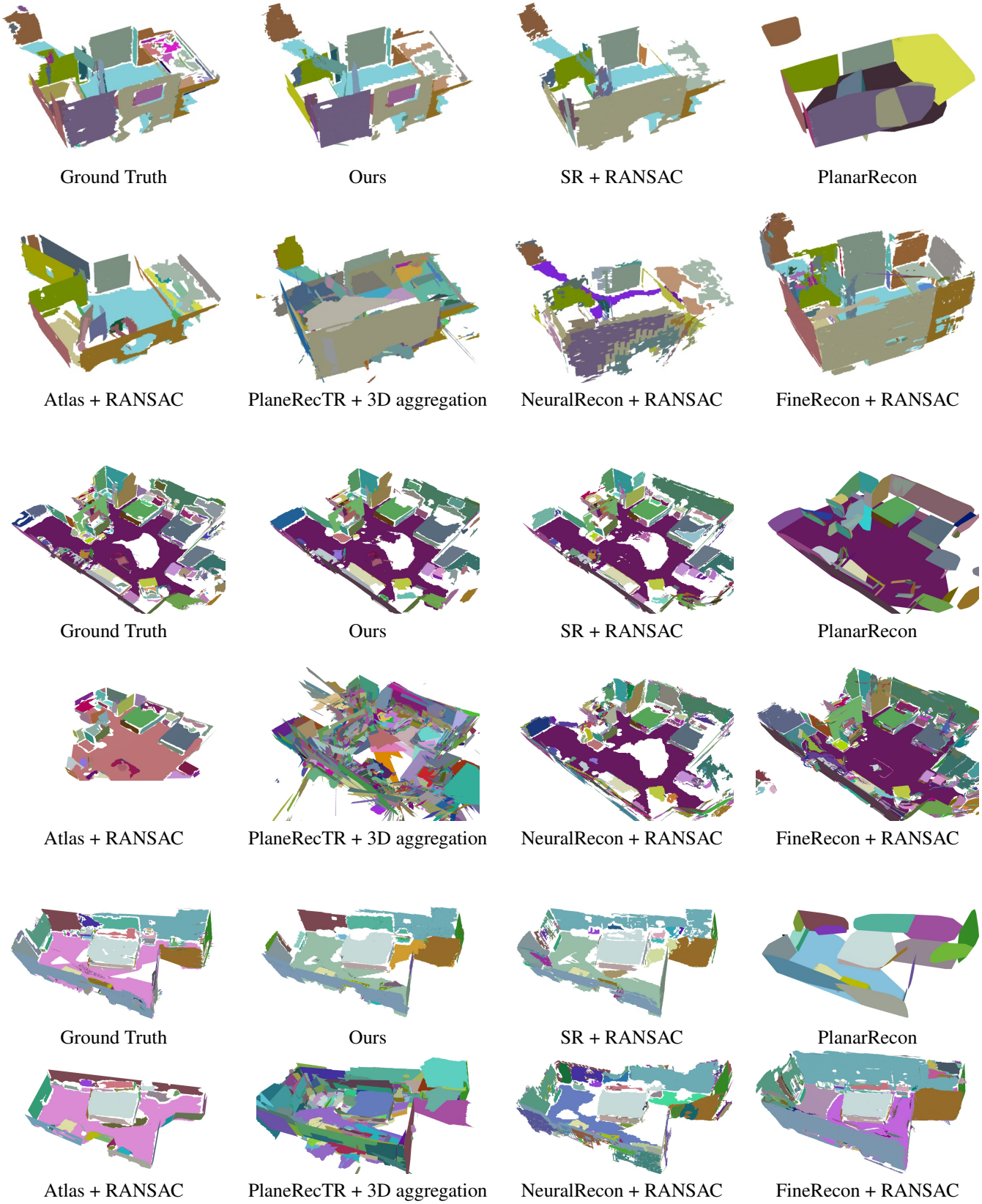


Figure A2. **Comparisons with more baselines and ablations on ScanNetV2.**

B. Additional implementation details

The input image resolution for all our networks is 512×384 and the output resolution for all the per-pixel 2D networks is 256×192.

B.1. Depth prediction and geometry reconstruction

We use the depth network from SimpleRecon [5] utilizing the publicly available version of the authors' implementation. This depth method uses a plane sweep stereo cost volume and a U-Net++ [14] to regularize the cost volume output using features from an image prior encoder to produce the final depth map. A shallow feature extractor captures features from the target frame and frames from source views. The features from the source views are warped to match the viewpoint of the target frame at various hypothesis depth planes using known camera intrinsics and extrinsics. Subsequently, a cost volume is formed by passing these features through a learned Multi-Layer Perceptron (MLP) as proposed in [5]. The resulting cost volume, along with deep image features extracted from the target frame, is then fed into a cost volume encoder for additional refinement. This is followed by a decoder, which adheres to the architecture introduced in [2, 10, 11].

We train for $110k$ steps with a batch size of 16 on two Nvidia A100s GPUs, with a learning rate drop at $60k$ and $70k$ steps. We select a model checkpoint after $100k$ steps based on the validation set.

B.2. Networks for planar masks and 2D embeddings

We use two small skip-decoders from [2] to predict planar masks and 2D embeddings. The input to these decoders are features from the SimpleRecon [5] depth backbone, specifically the output of the cost volume and image-prior encoders. We weight the 2D push/pull embeddings and planar mask losses equally. We use the same training schedule and hyperparameters as the depth network, but we only train the planar mask and 2D embeddings decoders and keep the depth encoders' weights fixed. We pick the best checkpoint based on the 2D embeddings loss on the validation set.

B.3. Online inference

When running the online version of our method, we use the mean-shift clustering implementation of [13]. The clustering is based on the embeddings only (i.e., no normals or offsets are used) and uses a bandwidth of 0.25. After convergence, we use the same two post-processing steps that are used for our RANSAC implementation, i.e., connected components to separate non-contiguous planes and removal of planes with fewer then 100 vertices.

As discussed in the paper, we run a simple Hungarian matching algorithm to match planes across time, to give color consistency in our visualizations. We compute an average plane embedding for each plane by averaging all the embeddings for points assigned to that plane. The cost matrix for the assignment problem is given by the euclidean distance between the average plane embeddings, where assignments are additionally set as invalid if: (a) the distance between the two plane centroids is higher than 2 meters, (b) the euclidean distance between normals is higher than 1, and (c) the number of vertices assigned to the plane goes down by more than 66%. All these thresholds are empirically chosen.

Note that more sophisticated algorithms could have been used for the matching step. For example, the clustering step could be informed by previous plane assignments and use those to initialize the clustering algorithm.

C. Implementation details for the baselines

C.1. 2D Semantic baseline

Our 2D semantic predictor has been trained to predict semantic maps given a single input image. Features are extracted from the image with an EfficientNet v2 [9] backbone and passed to a small decoder with skip connection, similar to the one used for estimating planar masks. The network predicts 20 classes and we use a cross entropy loss to train it. At test time, pixel-wise class probabilities predicted for each frame are fused into the TSDF volume, using depth maps from an off-the-shelf SimpleRecon model [5]. After this step, the final class for each voxel is selected by taking the class with highest probability. For completeness, the chosen classes are reported in Table A4.

wall	floor	cabinet	bed	chair
sofa	table	door	window	bookshelf
picture	counter	desk	curtain	refrigerator
shower curtain	toilet	sink	bathtub	otherfurniture

Table A4. **Semantic classes.** Semantic classes used to train the 2D semantic predictor.

C.2. PlaneRecTR [6] + aggregation baseline

PlaneRecTR [6] is a state-of-the-art method for single image 3D plane estimation. In order to be compared with the other methods for the task of plane estimation for an entire *scene*, we aggregate frame predictions into a single coherent representation. We follow a procedure similar to the one detailed in PlanarRecon [12] for their single frame baseline comparisons. A plane for the current frame is matched and aggregated with a world plane if: (a) the angle between the plane normals is smaller than 8.1 degrees; (b) the absolute difference between plane offsets is smaller than 0.5, and; (c) the absolute difference between plane centroids is smaller than 0.5 meters. If several world planes satisfy this condition, we choose the plane with smaller angle between the normals as the matching plane. After aggregation, we

update the world plane parameters (normal, offset, and centroid) using a running average. All thresholds were tuned on the validation set.

C.3. Sequential RANSAC baseline

Implementation details. Our sequential RANSAC baseline implementation consists of the the following steps: We first sample $N = 1000$ points from the mesh, where each of these is a candidate hypothesis. For each hypothesis, we estimate the plane equation using the point and its associated mesh normal. We then count the number of inlying vertices for each hypothesis. A point is considered an inlier if: (i) the dot product between its normal and the normal of the sampled point is > 0.8 and (ii) the distance from the point to the plane hypothesis is < 0.3m. We then select the plane hypothesis with the most inliers, and remove these inliers from the mesh. We then repeat all these steps until either: (i) we have found more than 100 planes or (ii) the largest plane we have found has fewer than 100 inliers. As in our method, for this RANSAC baseline, we finally run a connected components step at the end to divide up planes which are not contiguous on the mesh.

Differences to [12]. We compare our implementation of Sequential RANSAC on meshes against the variant used in [12]. A qualitative comparison can be seen in Fig. A3. We also compare the versions numerically in Table A2.

C.4. Speed comparison with PlanarRecon

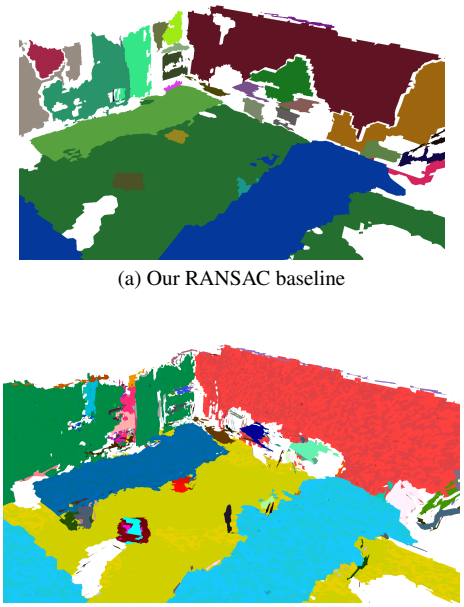
PlanarRecon [12] reports a running time of 40ms *per keyframe* on a NVIDIA V100 GPU, while our method runs in 152ms per keyframe. However, fairly comparing speeds is difficult. PlanarRecon constructs a scene *a fragment at a time*, where a fragment is composed of 9 keyframes. In practice, this means that PlanarRecon only updates the geometry and plane estimate every 9 keyframes, while we update both for every single keyframe. We have timed PlanarRecon on a NVIDIA T4 GPU and got the following breakdown of average timings *for a 9 keyframes fragment*:

Component	Running time in ms
Fragment net	328.8
Tracking	183.2
Features	117.2
Mean shift	29.0
Total time for a fragment	658.1

D. Additional evaluation details

D.1. Our dataset split

Our test set test$^{\text{planes}}$ was formed by selecting 100 unique scenes (i.e., only including those with suffixes `_00`) from the official ScanNetV2 validation set, since the test set does not have the necessary ground truth for plane evaluation. We select those scenes by choosing the scans with the least amount of missing or broken camera poses. We use the rest of the validation set as a separate validation set for tuning and checkpoint selection. The resulting scans selected for test$^{\text{planes}}$ are listed in Table A5. We will release our code for evaluation on this test split to ease further comparison by future researchers.



(a) Our RANSAC baseline

(b) RANSAC baseline from [12]

Figure A3. **RANSAC comparison.** Our RANSAC baseline (top) produces more accurate planes compared with the RANSAC baseline from [12] (bottom). See also Table A2. Both RANSAC methods are run using the same input mesh, which in this case is from [8]. Note the speckled artefacts in (b) are due to some extraneous faces in the mesh from [12], and do not impact the scores as we only evaluate the vertex positions not orientations.

D.2. Planar metrics

Before computing the planar metrics, we use a similar pre-processing step to the one used for geometric evaluation. We first sample 200,000 points from the ground truth and predicted meshes and compute metrics on those point clouds. We then apply a visibility mask to the point clouds and we mask out the 3D points sampled on faces that connect two or more planes. In contrast with the geometry metrics, we apply the visibility mask before computing the fidelity metric.

References

[1] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet:

scene0664_00	scene0064_00	scene0304_00	scene0338_00	scene0678_00	scene0593_00
scene0518_00	scene0251_00	scene0651_00	scene0011_00	scene0278_00	scene0207_00
scene0462_00	scene0086_00	scene0696_00	scene0647_00	scene0221_00	scene0693_00
scene0671_00	scene0316_00	scene0377_00	scene0663_00	scene0599_00	scene0435_00
scene0565_00	scene0406_00	scene0231_00	scene0307_00	scene0356_00	scene0578_00
scene0535_00	scene0558_00	scene0591_00	scene0030_00	scene0621_00	scene0652_00
scene0684_00	scene0088_00	scene0488_00	scene0187_00	scene0354_00	scene0217_00
scene0314_00	scene0633_00	scene0164_00	scene0549_00	scene0685_00	scene0084_00
scene0357_00	scene0100_00	scene0131_00	scene0644_00	scene0643_00	scene0458_00
scene0423_00	scene0300_00	scene0670_00	scene0144_00	scene0461_00	scene0686_00
scene0574_00	scene0081_00	scene0653_00	scene0552_00	scene0690_00	scene0378_00
scene0607_00	scene0629_00	scene0658_00	scene0608_00	scene0353_00	scene0575_00
scene0606_00	scene0046_00	scene0208_00	scene0050_00	scene0430_00	scene0025_00
scene0568_00	scene0474_00	scene0329_00	scene0553_00	scene0246_00	scene0389_00
scene0427_00	scene0343_00	scene0609_00	scene0660_00	scene0701_00	scene0598_00
scene0441_00	scene0595_00	scene0494_00	scene0146_00	scene0648_00	scene0580_00
scene0559_00	scene0550_00	scene0645_00	scene0426_00		

Table A5. **Test sequences.** The ScanNetV2 scans which form our test$^{\text{planes}}$ test set.

Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017. 1, 2

[2] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019. 5

[3] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. PlaneRCNN: 3D plane detection and reconstruction from a single image. In *CVPR*, 2019. 1

[4] 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, 2

[5] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. Simplerecon: 3D reconstruction without 3D convolutions. In *ECCV*, 2022. 1, 2, 5

[6] Jingjia Shi, Shuaifeng Zhi, and Kai Xu. PlaneRecTR: Unified query learning for 3D plane recovery from a single view. In *ICCV*, 2023. 1, 2, 5

[7] 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. *arXiv:2304.01480*, 2023. 1, 2

[8] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. In *CVPR*, 2021. 2, 6

[9] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021. 5

[10] Jamie Watson, Michael Firman, Gabriel J. Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *ICCV*, 2019. 5

[11] Jamie Watson, Oisin Mac Aodha, Victor Prisacariu, Gabriel J. Brostow, and Michael Firman. The temporal opportunist: Self-supervised multi-frame monocular depth. In *CVPR*, 2021. 5

[12] Yiming Xie, Matheus Gadelha, Fengting Yang, Xiaowei Zhou, and Huaizu Jiang. PlanarRecon: Real-time 3D plane detection and reconstruction from posed monocular videos. In *CVPR*, 2022. 1, 2, 5, 6

[13] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3D reconstruction via associative embedding. In *CVPR*, 2019. 5

[14] 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. 5