AirPlanes: Accurate Plane Estimation via 3D-Consistent Embeddings

Jamie Watson[1,3] Filippo Aleotti[1] Mohamed Sayed[1] Zawar Qureshi[1]
Oisin Mac Aodha[2] Gabriel Brostow[1,3] Michael Firman[1] Sara Vicente[1]
[1]Niantic [2]University of Edinburgh [3]UCL
https://nianticlabs.github.io/airplanes/

Abstract

Extracting planes from a 3D scene is useful for downstream tasks in robotics and augmented reality. In this paper we tackle the problem of estimating the planar surfaces in a scene from posed images. Our first finding is that a surprisingly competitive baseline results from combining popular clustering algorithms with recent improvements in 3D geometry estimation. However, such purely geometric methods are understandably oblivious to plane semantics, which are crucial to discerning distinct planes. To overcome this limitation, we propose a method that predicts multi-view consistent plane embeddings that complement geometry when clustering points into planes. We show through extensive evaluation on the ScanNetV2 dataset that our new method outperforms existing approaches and our strong geometric baseline for the task of plane estimation.

1. Introduction

While only parts of the real world are perfectly planar, a 3D reconstruction made out of planes is a useful parameterization for many downstream tasks. A planar scene reconstruction is a common representation for applications in robotics [20, 51], path planning [25], and augmented reality (AR) [75]. For example, both ARKit [2] and ARCore [21], two of the most used AR platforms, provide 3D plane estimation from scenes as part of their frameworks.

Broadly, there are two families of approaches for 3D plane extraction from images: geometric versus learning-based methods. Geometry-based pipelines assume access to a point cloud or mesh of the scene, e.g., as estimated from multi-view stereo or LIDAR. This geometry is then partitioned into planes using geometric cues, e.g., using RANSAC [17]. The disadvantages of these approaches are that they can be sensitive to noisy data and they do not typically encode learned priors to facilitate robust plane estimation. In contrast, learning-based methods make use of supervised data to develop models that can predict plane parameters from raw images. Many prior works have focused on the task of extracting planes from *single* input images [37, 38, 63, 71, 80, 82]. In practice though, it is more common to have a *sequence* of input images of the scene of interest, e.g., in AR applications where the user is interacting with new parts of the scene in real-time. There is however, only limited work that extends these learning-based single image methods to the multi-image setting [79].

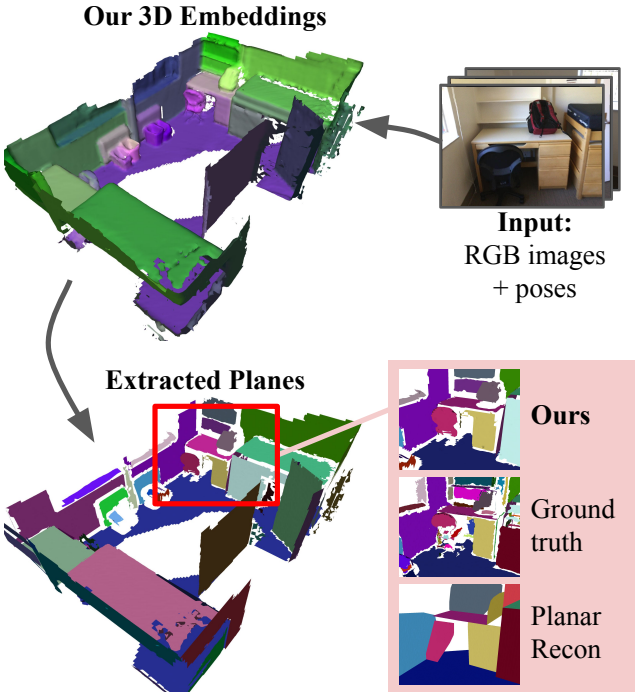


Figure 1. **We create planar scene representations using only posed RGB images as input**. Existing systems can predict *per-pixel* planar embeddings for each image, but these are not 3D consistent. We learn a per-scene function which maps points on the same plane to nearby positions in an embedding space. Clustering these embeddings, using strong geometrical priors, gives accurate planar reconstructions.

Inspired by recent work in interactive labeling [87], we propose an alternative approach to discovering planes in 3D. We train a small MLP network for each scene, which maps any 3D location in that scene to an embedding vec-

tor. Using various 2D and 3D cues, we train the MLP to produce embeddings which are 3D consistent and can be easily clustered to uncover distinct and accurate planar regions. By exploiting learned cues when decomposing a scene into planes, our method can adapt to different definitions of *what* constitutes a plane. This is important because the concept of what counts as a plane is application dependent. For example, a painting on the wall can be considered either a distinct plane or part of the wall plane, depending on the application. Unlike purely geometric definitions, our method learns what is considered a plane based on what is "encoded" in the training data.

Our core contribution is a new method that estimates 3D-consistent plane embeddings from a sequence of posed RGB images, and then groups them into planar instances. We demonstrate via extensive evaluation that our method is more accurate than recent end-to-end learning-based approaches, and can run at interactive speeds. We also make a surprising observation by proposing an additional strong 'geometry plus RANSAC' baseline. It can achieve impressive accuracy, outperforming existing baselines, ranking second place behind our proposed method.

2. Related work

Planes from single images. Although estimating 3D planes from single images is an ill-posed problem, multiple deep learning solutions have been proposed. Top-down approaches [37, 38, 80] directly predict a mask and the parameters of each plane. In contrast, bottom-up approaches [82] first map pixels into embeddings, which can subsequently be clustered into planes (e.g., via clustering methods [11]). More recent works [63, 71] leverage the query learning mechanism of Vision Transformers [15] to achieve state-of-the-art single-image results. These methods process frames independently, so are unable to produce temporally and 3D-consistent planes. As a result, they would require non-trivial plane tracking mechanisms to match the same plane across different frames over time. In contrast, we leverage multi-view image sequences, which enables planes to be estimated in 3D rather than just from single images.

We note that some works use planarity assumptions to regularize depth maps [19, 35, 62, 83] or to improve 3D scenes [3, 81] and poses [72]. In contrast, our aim is to find a high quality planar decomposition of the scene, rather than to use planarity for regularization in downstream tasks.

Planes from 3D and multi-view images. The extraction of geometric primitives, such as planes, from 3D point clouds is an established problem [78]. RANSAC [17] and the Hough transform [22] are popular strategies to help fit planes, and other 3D shapes [5, 29, 54, 60], to 3D data.

While a small number of works start from multi-view stereo estimated point clouds [4, 7, 34], the vast majority of plane extraction methods assume access to higher-quality 3D LIDAR scans [8, 33, 44, 46, 74]. These methods can be slow, i.e., not suitable for real-time AR applications, and they cannot easily cope with non-trivial amounts of noise in the input point clouds. To address noise, existing methods have attempted to enforce simple to define priors during reconstruction such as a Manhattan-world assumption [76], object/scene symmetry [43], or via user interaction [65]. Methods that only use geometry are fundamentally limited by the quality of the 3D information provided to them. In contrast, learning-based methods can learn to compensate for such issues and can also generate planar decompositions that better align with the semantic content of the scene.

Learning-based methods have been proposed for estimating planes from a limited number of input images [1, 27, 39]. However, extending these methods to entire videos is not trivial. Most related to us, PlanarRecon [79] is one of the first learning-based methods to predict a planar representation of entire 3D scenes. They incrementally detect and reconstruct 3D planes from posed RGB sequences, where 3D planes are detected in video fragments before the fragments are fused into a consistent planar reconstruction. The pipeline is somewhat complex and contains expensive operations such as 3D convolutions, recurrent units, and differentiable matching. In contrast, we trade such complexity for an efficient non-plane-based 3D scene reconstruction method [58], which provides reliable scene geometry estimates suitable for input to our plane estimation method. Finally, room layout estimation can also leverage multiple images [23, 56, 68]. However, their extreme scene simplification is only suitable for a limited number of applications.

2D and 3D segmentation. Our task of dividing a 3D scene into planes has some similarities with 3D semantic [9, 55] or panoptic instance [47, 59, 61] segmentation. These methods are less applicable to our problem because they aim to segment objects or semantic regions without special regard for geometric properties. Recent works have leveraged NeRFs to obtain a consistent semantic [86] or panoptic [32, 64] scene representation. Our method follows this direction by also using test time optimization. However, unlike [64] we use an online rather than offline reconstruction method, and do not need to perform linear assignment for every frame.

Scene-level embeddings. Our key innovation is to use per-scene 3D embeddings to represent planes. We are inspired by previous works, e.g., iMap [69] and iLabel [87], who showed how emergent embeddings can be used for interactive reconstruction and labeling. We are inspired by these works, but instead of encoding scene geometry or semantic labels, we encode plane embeddings trained to be multi-view consistent. Related, there are works that optimize 3D embeddings from 2D supervision, e.g., to ground 2D vision-language features [53] in 3D [28, 30, 52, 73]. However, unlike our reconstruction focus, their aim is to ground open-vocabulary semantic queries in 3D.

Representations for 3D reconstruction. The focus of our work is planar scene representations, but there are many alternatives to planes. For example, TSDFs encode shape volumetrically. They can be generated by estimating depth, e.g., from multi-view stereo [10, 16, 18, 24, 58], or directly via more expensive 3D convolutions [45]. Subsequent methods [6, 70] have extended neural TSDF estimation to the online setting. Implicit functions are an alternative representation which have been used to map from 3D space to occupancy [40, 41, 50, 57]. In the context of SLAM, implicit neural strategies have been developed [69, 85, 88] that are able to encode scene geometry using a multi-layer perceptron (MLP). Finally, further from our task, the recent success of NeRFs [42] for realistic novel view synthesis has paved the way for methods that apply volume rendering to represent a scene using a neural network [36, 49, 77, 84].

3. Method

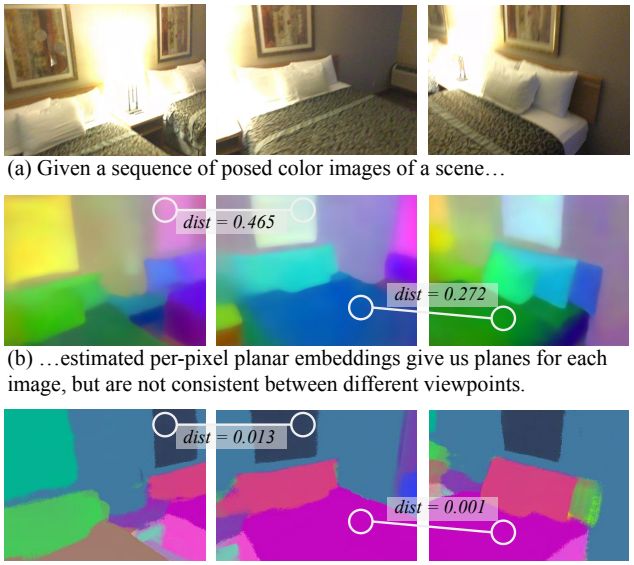
We take as input a sequence of color images, each associated with a known camera pose. We aim to predict a representation of the imaged 3D scene, where surfaces are segmented into constituent planes. We follow the definition of planes from previous work [38, 39, 79] where there can be semantic separation between parts, e.g., nearby table-tops should each have a different plane and a closed door should have a different plane to the wall enclosing it.

Our approach estimates planes by first reconstructing the 3D geometry of the scene using a mesh representation. We then train a network that maps each point on the mesh to a 3D-consistent embedding space, such that points on the same plane map to nearby places in the space. These embeddings *implicitly* encode semantic instance information and geometric cues. Semantics complement the 3D geometry information provided by the mesh computed via a lightweight multi-view stereo system. We then use a clustering algorithm on the geometry and embeddings to compute accurate plane assignments. All the steps in our method support online inference. An overview of our approach is shown in Fig. 3.

3.1. Learning 3D planar embeddings

Our key innovation is to learn a mapping from each 3D point $\mathbf{p}$ in a reconstructed scene to an embedding $\mathbf{e_p}$, such that points on the same plane map to nearly the same place in embedding space, while points on different planes map to different places. We denote these as '3D embeddings', where 3D refers to the fact that the embeddings encode *per-scene*, and not *per-image*, planar information. We first review how existing *single image* embedding networks are trained, before describing how we distill these pixel-wise embeddings into a 3D-consistent embedding.

Single image embeddings. In the case of plane estimation from monocular images, [82] train a feedforward network to map a single color image to per-pixel embeddings. Pixels i and j in the same image are mapped to embeddings $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively, where $\mathbf{x}_i$ is similar to $\mathbf{x}_j$, if and only if i and j are in the same plane. This is achieved by training a network which takes as input a *single* image and outputs a per-pixel embedding, using two losses: a *pull* loss penalizing pixel embeddings $\mathbf{x}_i$ that are different from the mean embedding of their corresponding plane; and a *push* loss encouraging mean embeddings for each plane to be different from each other. One option to obtain 3D embeddings could be to find all pixels that correspond to the reprojection of a 3D point across multiple views and average their per-pixel embeddings. The issue with this approach can be seen in Fig. 2. Here, the per-pixel embeddings are not consistent across views, despite encoding valuable planar instance information for each individual view. This is ablated in Sec. 5.3 as 'embeddings w/o test-time optimization'.



(a) Given a sequence of posed color images of a scene…

(b) …estimated per-pixel planar embeddings give us planes for each image, but are not consistent between different viewpoints.

(c) Our per-scene embeddings are consistent across 3D space, enabling us to extract accurate planes for the whole scene.

Figure 2. **Per-image planar embeddings are not temporally consistent.** While they can segment planes within a single image, plane embeddings in (b) from [82] do not result in 3D consistent embeddings for a full scene. Our method (c) gives a per-scene embedding which is consistent across many views of that scene.

Consistent 3D embeddings. Our goal is to learn embeddings that preserve the properties of the per-pixel embeddings, while being consistent across views. We achieve this goal by learning a *per-scene* mapping function ϕ, which is parameterized as an MLP and is optimized at test time, following recent work [52, 87]. Our network takes as input a 3D point $\mathbf{p}$ and predicts its '3D' embedding $\mathbf{e_p} = \phi(\mathbf{p})$.

Single image embeddings distillation loss. Our network ϕ is trained to distill information contained in the per-pixel embeddings $\mathbf{x}$. For a pair of pixels i and j in a single image, we take their embeddings $\mathbf{x}_i$ and $\mathbf{x}_j$. We also know

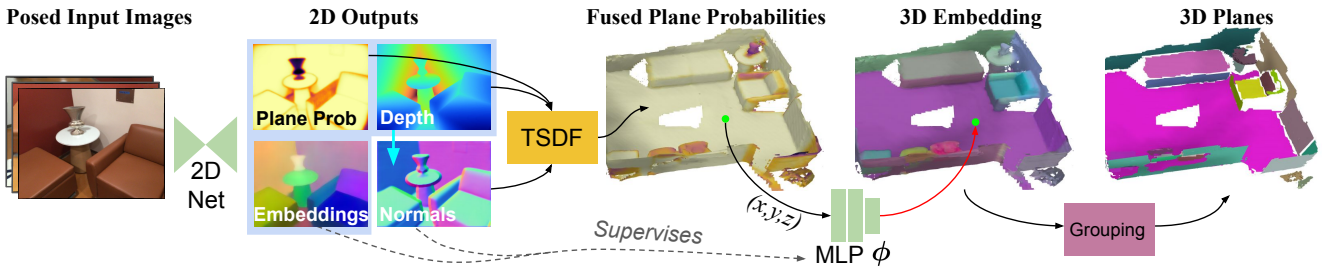


Figure 3. **Our method for 3D plane estimation.** For each RGB keyframe we estimate per-pixel depth, planar probability and planar embedding following [82]. We fuse the depths and planar probabilities into a TSDF and extract a mesh. We then train a per-scene MLP to distill the per-pixel embeddings into 3D-consistent embeddings. These are finally grouped via clustering into 3D planes.

their corresponding 3D positions $\mathbf{p}_i$ and $\mathbf{p}_j$ and their image-space normals $\mathbf{n}_i$ and $\mathbf{n}_j$. We can then train the network ϕ such that $\phi(\mathbf{p}_i)$ is similar to $\phi(\mathbf{p}_j)$, if and only if their corresponding embeddings in image space $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar and their normals ($\mathbf{n}_i$ and $\mathbf{n}_j$) are also similar. Inspired by the *push-pull* loss used for the single image embeddings [82], we use the following loss to encourage this:

$$L_\phi = \begin{cases} \|\phi(\mathbf{p}_i) - \phi(\mathbf{p}_j)\|, \\ \quad \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < t_e \text{ and } \mathbf{n}_i \cdot \mathbf{n}_j > t_n \\ \max(0, t_p - \|\phi(\mathbf{p}_i) - \phi(\mathbf{p}_j)\|), \\ \quad \text{otherwise,} \end{cases} \tag{1}$$

where t_e is a *pull* threshold on embeddings, t_n is a threshold on normals, and t_p is a *push* threshold. This loss is applied to sampled pairs of points on the same image.

3.2. 3D geometry estimation

To estimate planes, we use our 3D embeddings alongside an initial estimate of scene geometry. To estimate an accurate 3D mesh we use SimpleRecon [58], a state-of-the-art 3D reconstruction system that requires posed images as input. In it, depth maps are estimated using a multi-view stereo net, then fused into a 3D mesh via a truncated signed distance function (TSDF) [12].

We adapt their network to additionally predict a planar/non-planar probability, assigning a per-pixel value indicating if that pixel belongs to a planar or non-planar region, trained equivalently to the single-image plane estimator of [82]. Our novelty is to then combine these per-pixel predictions into 3D as an additional channel in the TSDF. When extracting the mesh, we exclude voxels that have an aggregated non-planar value of less than $p = 0.25$, so that non-planar regions are not part of the final mesh. This extracted mesh is one of the inputs to the next steps.

3.3. Plane grouping

Given an embedding for each vertex in our 3D mesh, our next step is to cluster vertices into plane instances based on those embeddings and on geometry information defined by the mesh. For this clustering step we rely on sequential RANSAC [17]. RANSAC works by randomly sampling plane instance proposals, checking the inlier count for each proposal, and selecting the plane instance with the most inliers. This process is done sequentially, where at each iteration the points associated with the last predicted plane are removed from the pool. Each plane instance proposal is created by sampling a single mesh vertex, which together with its associated normal, defines a plane. A different mesh vertex is considered an inlier to this plane proposal if: (i) the distance to the plane is smaller than a threshold r_d and (ii) the euclidean difference between embeddings is smaller than a threshold r_e. After convergence, we merge planes with highly similar embeddings and normals, i.e., where the distance between average embeddings is < 0.2 and the dot product between average normals is > 0.6. Next, we run a connected components algorithm on the mesh representation of each discovered plane in turn, to separate out non-contiguous planes.

Since the non-planar vertices have already been removed as explained in Sec. 3.2, we expect all remaining vertices to be assigned a plane instance label. RANSAC, however does not guarantee this. For this reason, we run a post-processing step that iteratively propagates labels to connected unlabeled points from the RANSAC step. Finally, we remove planes with fewer than 100 vertices.

3.4. Online inference

All components of our method are designed so that they can run online with little adaptation. The 3D geometry estimation steps, i.e., depth estimation, fusion into TSDF, and mesh extraction, are commonly used in online systems [48, 58]. Our per-scene embedding network is always updated in an online fashion, similar to [69, 87]. Given the current 3D mesh and the current embedding network, embeddings can be predicted for all mesh vertices. We then perform clustering to extract plane instances. To achieve interactive speeds for our online method, we replace our RANSAC clustering method, which takes 131ms on aver-

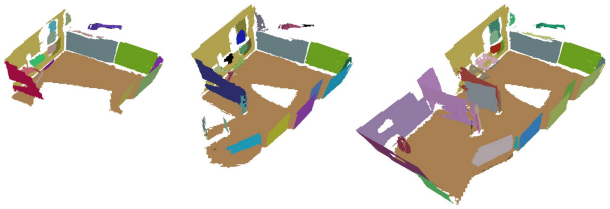


Figure 4. **Planes can be estimated online at interactive rates.** As new RGB frames are acquired, we can update the weights of our MLP and recompute plane assignments. See Sec. 5.5 for timings.

age per scene, with the mean-shift algorithm [11] using the efficient implementation from [82], which takes 25ms. We evaluate this alternative clustering algorithm in the experimental section. Finally, each time we recompute planes, we use Hungarian matching [31] between the previous and current plane assignments to encourage consistency of planes across time (visible in the figure as stability of colors over time, while new planes are computed). Fig. 4 shows an online reconstruction obtained with our method for a ScanNetV2 scene.

3.5. Sequential RANSAC: A strong baseline

Given recent advances in 3D scene reconstruction from image inputs, e.g., [58], the question arises: How good a planar decomposition can we achieve if we run RANSAC on the mesh only, without the contribution of our 3D embeddings? Surprisingly, later results show this simple baseline performs very well. However, while this naive approach takes geometry into account, it does not leverage semantic or appearance-based cues, leading to plane over- and under-segmentation issues (see Fig. 5). Our method, using 3D plane embeddings, addresses these problems.

4. Implementation details

Depth, plane probabilities, and per-pixel embedding network architecture. We use the SimpleRecon [58] architecture for depth estimation. Encoder features are shared between the depth estimation, plane probabilities, and per-pixel embedding tasks, though they have separate decoders. Full architecture details are in the supplementary material.

Embedding MLP network. We use a three-layer MLP with 128 dimensions for each hidden layer. Following [66], we lift the input to the MLP to 48 periodic activation functions before it is input to the first linear layer. Our final embedding has three dimensions. We use $t_e = 0.9$, $t_n = 0.8$ and $t_p = 1.0$, tuned on the validation set. Similarly to [87], the MLP is always trained in an online fashion. For each new keyframe we sample 400 pixels from it and apply Eqn. (1) to each pair of points, together with the pairs from the 10 most recent keyframes. We then run backpropagation ten times to optimize the MLP.

Grouping thresholds. For RANSAC we set $r_e = 0.5$ and $r_d = 0.1$. We set the mean-shift bandwidth to 0.25.

(a) RGB view (b) Ground truth planes

(c) Planes from geometry + RANSAC do not respect object instance boundaries

(d) Our predictions give a separate plane for each instance

Figure 5. **Sequential RANSAC alone is not enough to segment planar instances.** Sequential RANSAC (with geometry from [58]) does well, but fails to segment adjacent co-planar instances. Our method can segment these, e.g., this picture frame.

Mesh planarization. Given our final assignment of points to planes, we perform *mesh planarization* to convert our 3D mesh into a planarized mesh. First, we estimate the plane equation for each plane. Next, each point is moved along the normal of its assigned plane such that it lies on the plane it is assigned to. This is the mesh which is geometrically evaluated against the ground truth planarized mesh.

5. Experiments

We train and evaluate on ScanNetV2 [13], because [38] provided ground truth plane annotations for most of it. Plane annotations are unavailable for the ScanNetV2 test set. 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, we re-evaluate baselines on our new test split. The new splits and our evaluation code are available at https://nianticlabs.github.io/airplanes/.

5.1. Evaluation metrics

Geometric evaluation. Here, we evaluate how well the predicted planar mesh approximates the geometry of the ground truth planar mesh. Following [79] we adopt conventional 3D metrics [6, 45]. To compare a predicted mesh with the ground truth mesh, we first sample $N = 200,000$ points from each mesh. We then compare the two sampled point clouds to each other using chamfer distance and f1 score. See [6] for details.

Fully volumetric methods such as [45] predict geometry for the whole scene, including unobserved regions. To prevent such methods from being penalized unfairly, we en-

	Geometry		Segmentation			Planar		
	chamfer ↓	f1 ↑	voi↓	ri↑	sc↑	fidelity↓	accuracy↓	chamfer↓
PlaneRecTR [63] + aggregation	8.82	42.53	4.028	0.924	0.268	22.58	15.71	19.14
Atlas [45] † + RANSAC	12.65	53.71	2.868	0.932	0.465	22.60	17.71	20.16
NeuralRecon [70] + RANSAC	9.00	46.91	3.176	0.929	0.391	16.76	13.08	14.92
FineRecon [67] + RANSAC	5.56	64.10	2.377	0.950	0.531	**7.74**	11.71	9.72
SR [58] + RANSAC	5.40	**65.45**	2.507	0.946	0.515	9.42	10.13	9.78
PlanarRecon [79]	9.89	43.47	3.201	0.919	0.405	18.86	16.21	17.53
Ours	**5.30**	64.92	**2.268**	**0.957**	**0.568**	8.76	**7.98**	**8.37**

Table 1. **Quantitative evaluation.** We report geometry scores using the test^{planes} split of ScanNetV2 [13]. Here, **bold** indicates best, underline second best. We use publicly available checkpoints unless indicated. † indicates that we use fewer voxels to fit in memory.

	Geometry chamfer↓	Segmentation voi↓	Planar chamfer↓
Atlas [45] † + RANSAC	12.65	2.868	20.16
+ our 3D embeddings	**12.30**	**2.673**	**18.92**
NeuralRecon [70] + RANSAC	9.00	3.176	14.92
+ our 3D embeddings	**8.54**	**2.713**	**13.42**
FineRecon [67] + RANSAC	5.56	2.377	9.72
+ our 3D embeddings	**4.80**	**2.159**	**7.76**
SR [58] + RANSAC	5.40	2.507	9.78
+ our 3D embeddings **(Ours)**	**5.30**	**2.268**	**8.37**

Table 2. **Our 3D embeddings** can be used in combination with a variety of different 3D geometry estimators, leading to improved results for all methods.

force a visibility mask to handle unseen points differently when computing metrics, following [6, 58]. This visibility mask is applied to all methods for fair comparison. We also mask out 3D points sampled on faces that connect two or more planes, as these points have ambiguous labeling. For full transparency, we report numbers in the supplementary material using the evaluation method from [79] without our additions.

Plane segmentation evaluation. Following previous work on plane estimation [63, 79, 80, 82], we also report the following clustering metrics: Variation of Information (VOI), Rand Index (RI), and Segmentation Covering (SC). Given a predicted mesh, we use the protocol proposed in [79] to map the plane ID of each vertex to the closest vertex in the ground truth mesh. See [79] for full details.

Planar metrics. To better evaluate how well the main, i.e., large planes in the ground truth scene, are reconstructed we additionally propose the following protocol. We select the $k = 20$ largest planes from each ground truth mesh. For each such plane $\mathbf{q}_j$, we find the predicted plane $\mathbf{p}_i$ that most closely matches according to the completion metric. We report the fidelity between $\mathbf{q}_j$ and $\mathbf{p}_i$ as $completion(\mathbf{q}_j, \mathbf{p}_i)$, where *completion* is the completion metric from [45]. The average of this score over all k ground truth planes over all scenes is our *planar fidelity* score. We also report the geometric accuracy between $\mathbf{q}_j$ and $\mathbf{p}_i$ as *planar accuracy*, and the average of the two as *planar chamfer*.

5.2. Comparisons with baselines

We evaluate our 3D plane estimation method against various baselines (Table 1). **PlanarRecon** [79] is the existing state-of-the-art method for 3D plane estimation from posed RGB images. We outperform their approach in geometry, segmentation, and planar metrics. We also compare with the leading baseline for 3D plane estimation from a single image, **PlaneRecTR** [63]. For each scene, we run this single image predictor for selected keyframes. Planes from each incoming image are matched to the closest world planes by comparing planar normals, offsets, and plane positions.

We compare with our own implementation of sequential **RANSAC**, applied to meshes from SimpleRecon [58]. SimpleRecon (SR in tables) is the same method we use for geometry estimation, as detailed in Sec. 3.2, making this the closest baseline to our method, but without using the benefits of our 3D consistent embeddings. In addition, we also apply the sequential RANSAC method to geometry from [45, 67, 70]. See supplementary material for implementation details of the baselines.

Our method outperforms all other methods on the segmentation metrics. While the results for the geometric metrics are comparable with the *SR [58] + RANSAC* baseline, we significantly outperform this baseline on the segmentation and planar metrics, clearly demonstrating the benefit of using our 3D consistent embeddings. Surprisingly, PlanarRecon [79] is outperformed by several of our sequential RANSAC baselines. This is in contrast with the results presented in [79], and we discuss this difference in more detail in the supplementary material.

Our embeddings benefit other geometry methods. To validate the usefulness of our 3D embeddings, we use them in combination with different geometry estimation methods [45, 58, 67, 70]. We compare using only 3D geometry versus using 3D geometry plus the embeddings derived from our test-time optimized MLPs without retraining. We show the results for this experiment in Table 2. For all methods, we observe that the additional information encoded in the embeddings improves over the baseline of using geometry + RANSAC only.

	Geometry		Segmentation			Planar		
	chamfer ↓	f1 ↑	voi↓	ri↑	sc↑	fidelity↓	accuracy↓	chamfer↓
Fused per-pixel embeddings w/o test time optimization	5.76	61.83	2.670	0.949	0.485	11.63	8.30	9.97
Fused per-pixel embeddings w. train. time m-v consist.	5.72	61.91	2.672	0.950	0.485	11.90	8.01	9.96
Ours without planar probability	5.39	64.04	2.257	0.958	0.570	8.67	8.12	8.39
Ours (RANSAC)	5.30	64.92	2.268	0.957	0.568	8.76	7.98	8.37
Ours (Mean-shift)	5.70	62.73	2.344	0.954	0.556	9.56	8.19	8.88
SR [58] + RANSAC + predicted semantic labels	5.56	64.79	2.483	0.948	0.525	10.20	7.81	9.01
SR [58] + RANSAC + g.t. semantic labels	5.41	65.71	2.262	0.956	0.568	9.96	6.13	8.04
SR [58] + RANSAC + g.t. instance labels	5.68	65.57	2.257	0.954	0.584	9.94	4.90	7.42

Table 3. **Our contributions result in the best performing model.** In these ablations, we turn parts of our method off in turn, or replace them with alternatives, to show the benefit our contributions bring.

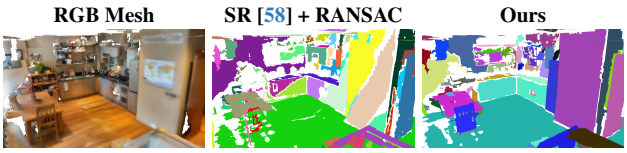


Figure 6. **Our method generalizes well to other scenes.** Here we show results on some non-ScanNetV2 casually-captured footage.

5.3. Ablations

We ablate our method to validate that our contributions lead to higher scores. These results are in Table 3. **Fused per-pixel embeddings w/o test time optimization** is our method but using the embeddings directly from [82], without our 3D distillation. These embeddings are fused as additional channels into the TSDF. **Fused per-pixel embeddings with training time multi-view consistency** is a variant of our method, where we attempt to train a single feed-forward embedding network which predicts multi-view consistent 3D embeddings directly, without performing test-time optimization. **Ours without planar probability** is our method but all points are assigned a planar probability of 1, meaning non-planar points are still part of the mesh. For this reason, we do not run the post-processing step that assigns unlabeled points after RANSAC.

The first two ablations show that it is not trivial to predict 3D consistent embeddings using a feed-forward network applied to each frame independently. This motivates our use of a per-scene MLP optimized at test time to achieve consistent embeddings. The last of these ablations shows that our fusion of planar probabilities into the TSDF improves geometry metrics. We note that some computational savings could be made, at the price of $\sim$1% drop in geometry scores, without this.

We also compare the two different plane grouping algorithms. Using the Mean-shift variant of our method leads to only a small degradation of results versus RANSAC, while achieving interactive speeds (see Sec. 5.5 for timings).

RANSAC oracle methods are variants of RANSAC which have access to ground truth semantic and instance information. **SR + RANSAC + ground truth semantic labels** uses ground truth semantic labels (transferred to the closest vertex in the predicted mesh) in the sequential RANSAC loop to separate planes. Specifically, points can be associated with a plane candidate only if they are geometrically consistent and have the same label. We additionally compare with a RANSAC variant with ***predicted* semantic labels**, where we predict $N = 20$ semantic classes and we fuse their probabilities in the TSDF. It is worth noting that explicitly predicting semantics is beneficial and leads to better planar scores compared to its geometry-only counterpart in Table 1. However, our method provides better results across all metrics, and requires fusion of only planar probabilities instead of N semantic classes, which might be challenging as N increases. Finally, we also show an oracle with **ground truth instance labels**, which presents an upper bound for plane estimation.

5.4. Qualitative results

Fig. 7 shows results of our method compared to the closest published competitor *PlanarRecon [79]* and our *SR [58] + RANSAC* baseline. We can see that our method has closer fidelity to the ground truth versus [79], and avoids oversimplification of geometry. By more closely adhering to the geometry of the real scene, our planes can appear to have 'jagged' edges when compared to the more simplified outputs from [79]. Our planar meshes have gaps where planes intersect because we remove triangles that connect vertices from different planes. If needed for a specific application, our outputs could be further post-processed, e.g., using [46]. A qualitative comparison of our method with the *SR [58] + RANSAC* baseline shows that we are able to recover separate semantic planes that have a common planar geometry. Finally, Fig. 6 shows more results of our method, with images and camera poses from an iPhone running ARKit [2].

5.5. Planes at interactive speeds

The online variant of our method, which uses mean-shift clustering, takes a total of 152ms per keyframe on average, on an RTX A6000 GPU. This comprises 65ms to obtain the per-pixel depth, planar probability, and 2D planar embedding and separately 1ms for TSDF fusion, 61ms to update the MLP, and 25ms to run the clustering. As the av-

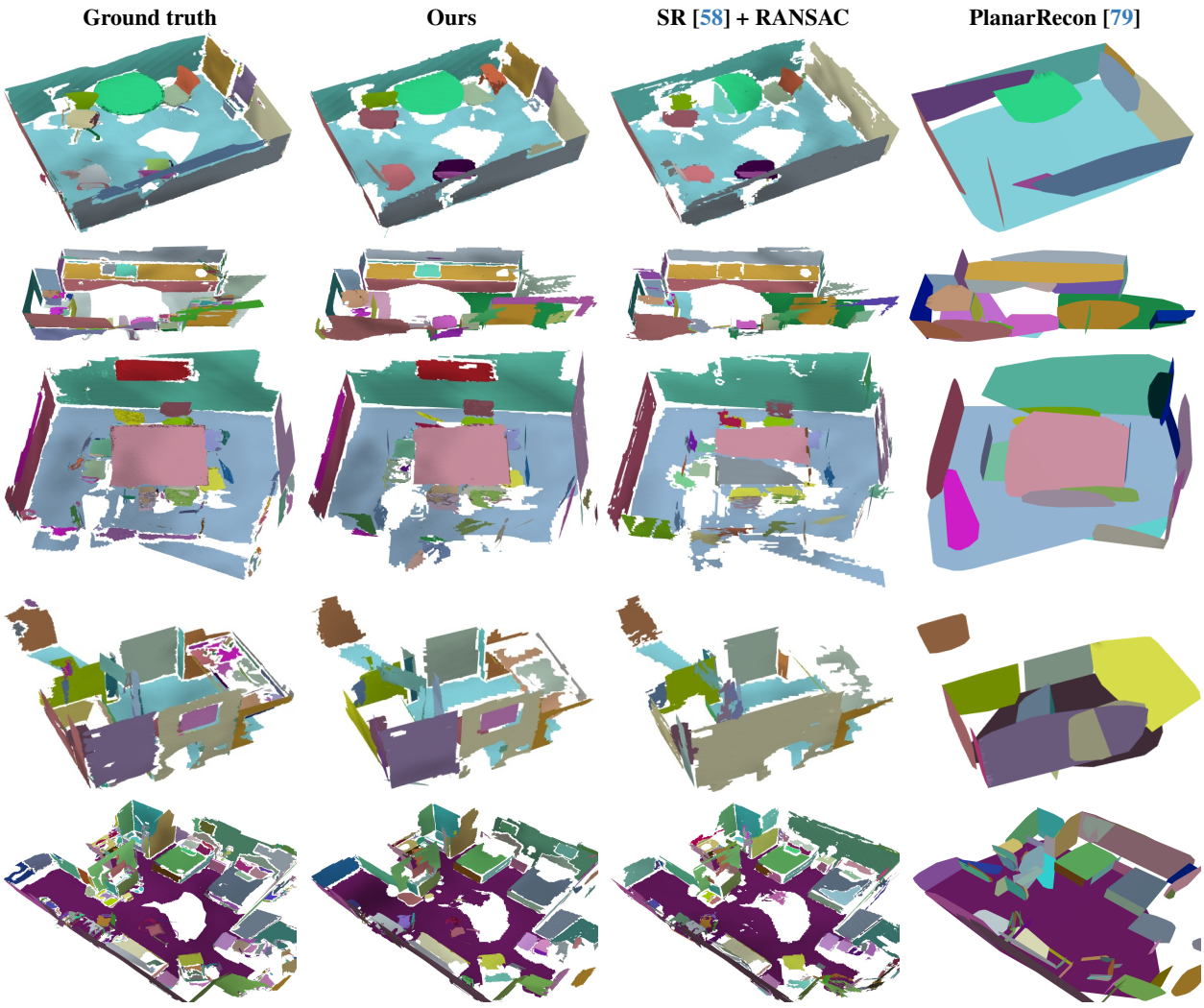


Figure 7. **Qualitative results on ScanNetV2.** Our predictions are more faithful to the ground truth both in terms of geometry and segmentation. Here we have removed ceiling planes (i.e., those with normals facing downwards) for visualization. In the first row, we recover the size and shape of the table well when compared to the baselines. In the second row, the sink is well separated from the countertop in our results. The bottom row shows a failure case where we do not recover small pillows on the beds.

erage interval between keyframes in ScanNetV2 is 272ms, our method runs at interactive speeds. Alternatively, for the RANSAC variant, the clustering step takes 131ms for an entire scene.

5.6. Limitations

Our method shows notable improvements compared to other 3D plane estimation methods, but limitations remain. Errors in the geometry from our MVS system might have severe consequences when extracting 3D planes. We also fit planes in a greedy manner. Instead, global optimization e.g., [26] may further improve results. Unlike [79], we only estimate planes for visible geometry. Completing unobserved regions, like [14, 45, 67], could be a useful extension for some applications.

6. Conclusion

We propose a new approach which takes a sequence of posed color images as input, and outputs a planar representation of the 3D scene. Surprisingly, we demonstrate that a strong baseline for this task is to simply run sequential RANSAC on a lightweight 3D reconstruction. However, this baseline is likely too limited for AR and robotics use-cases. Our approach addresses this, by training a 3D embedding network to map 3D points to 3D-consistent and meaningful plane embeddings, which can then be clustered into 3D planes. Our approach gives state-of-the-art plane estimation performance on the ScanNetV2 dataset.

Acknowledgements. We are extremely grateful to Saki Shinoda, Jakub Powierza, and Stanimir Vichev for their invaluable infrastructure support.

References

[1] Samir Agarwala, Linyi Jin, Chris Rockwell, and David F Fouhey. Planeformers: From sparse view planes to 3D reconstruction. In *ECCV*, 2022. 2

[2] Apple. ARKit, 2023. Accessed: 5 October 2023. 1, 7

[3] Charlotte Arndt, Reza Sabzevari, and Javier Civera. Do planar constraints improve camera pose estimation in monocular SLAM? In *ICCV*, 2023. 2

[4] András Bódis-Szomorú, Hayko Riemenschneider, and Luc Van Gool. Fast, approximate piecewise-planar modeling based on sparse structure-from-motion and superpixels. In *CVPR*, 2014. 2

[5] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. The 3D hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2011. 2

[6] Aljaz Bozic, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. TransformerFusion: Monocular RGB scene reconstruction using transformers. *NeurIPS*, 2021. 3, 5, 6

[7] Anne-Laure Chauve, Patrick Labatut, and Jean-Philippe Pons. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *CVPR*, 2010. 2

[8] Jie Chen and Baoquan Chen. Architectural modeling from sparsely scanned range data. *IJCV*, 2008. 2

[9] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 2

[10] Robert T Collins. A space-sweep approach to true multi-image matching. In *CVPR*, 1996. 3

[11] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 2002. 2, 5

[12] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Conference on Computer graphics and interactive techniques*, 1996. 4

[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. 5, 6

[14] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3D scans. In *CVPR*, 2018. 8

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2

[16] 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. 3

[17] Martin A Fishler. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 1, 2, 4

[18] David Gallup, Jan-Michael Frahm, Philippos Mordohai, Qingxiong Yang, and Marc Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *CVPR*, 2007. 3

[19] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *CVPR*, 2010. 2

[20] Patrick Geneva, Kevin Eckenhoff, Yulin Yang, and Guoquan Huang. Lips: Lidar-inertial 3D plane SLAM. In *IROS*, 2018. 1

[21] Google. ARCore, 2023. Accessed: 5 October 2023. 1

[22] Paul VC Hough. Method and means for recognizing complex patterns, 1962. US Patent 3,069,654. 2

[23] Zhihua Hu, Bo Duan, Yanfeng Zhang, Mingwei Sun, and Jingwei Huang. MVLayoutNet: 3D layout reconstruction with multi-view panoramas. In *International Conference on Multimedia*, 2022. 2

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

[25] Britta Hummel, Soeren Kammel, Thao Dang, Christian Duchow, and Christoph Stiller. Vision-based path-planning in unstructured environments. In *Intelligent Vehicles Symposium*, 2006. 1

[26] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *IJCV*, 2012. 8

[27] Linyi Jin, Shengyi Qian, Andrew Owens, and David F Fouhey. Planar surface reconstruction from sparse views. In *ICCV*, 2021. 2

[28] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *ICCV*, 2023. 2

[29] Florian Kluger, Hanno Ackermann, Eric Brachmann, Michael Ying Yang, and Bodo Rosenhahn. Cuboids revisited: Learning robust 3D shape fitting to single RGB images. In *CVPR*, 2021. 2

[30] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *NeurIPS*, 2022. 2

[31] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955. 5

[32] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation. In *CVPR*, 2022. 2

[33] Florent Lafarge and Pierre Alliez. Surface reconstruction through point set structuring. In *Computer Graphics Forum*, 2013. 2

[34] Florent Lafarge, Renaud Keriven, Mathieu Brédif, and Hoang-Hiep Vu. A hybrid multiview stereo algorithm for modeling urban scenes. *PAMI*, 2012. 2

[35] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv:1907.10326*, 2019. 2

[36] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin.

Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023. 3

[37] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single RGB image. In *CVPR*, 2018. 1, 2

[38] 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, 2, 3, 5

[39] Jiachen Liu, Pan Ji, Nitin Bansal, Changjiang Cai, Qingan Yan, Xiaolei Huang, and Yi Xu. PlaneMVS: 3D plane reconstruction from multi-view stereo. In *CVPR*, 2022. 2, 3

[40] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3D supervision. In *NeurIPS*, 2019. 3

[41] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019. 3

[42] 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. 3

[43] Niloy J Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3D geometry: Extraction and applications. In *Computer Graphics Forum*, 2013. 2

[44] Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. RAPter: rebuilding man-made scenes with regular arrangements of planes. *ACM Transactions on Graphics*, 2015. 2

[45] 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. 3, 5, 6, 8

[46] Liangliang Nan and Peter Wonka. PolyFit: Polygonal surface reconstruction from point clouds. In *ICCV*, 2017. 2, 7

[47] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. In *IROS*, 2019. 2

[48] Richard A Newcombe, Shahram Izadi, and Otmar Hilliges. KinectFusion: Real-time dense surface mapping and tracking. In *UIST*, 2011. 4

[49] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *ICCV*, 2021. 3

[50] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 3

[51] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius, Max Pfingsthorn, Sören Schwertfeger, and Jann Poppinga. Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation. *Journal of Field Robotics*, 2010. 1

[52] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. OpenScene: 3D scene understanding with open vocabularies. In *CVPR*, 2023. 2, 3

[53] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2

[54] Michaël Ramamonjisoa, Sinisa Stekovic, and Vincent Lepetit. Monteboxfinder: Detecting and filtering primitives to fit a noisy point cloud. In *ECCV*, 2022. 2

[55] Dario Rethage, Johanna Wald, Jurgen Sturm, Nassir Navab, and Federico Tombari. Fully-convolutional point networks for large-scale point clouds. In *ECCV*, 2018. 2

[56] Denys Rozumnyi, Stefan Popov, Kevis-Kokitsi Maninis, Matthias Nießner, and Vittorio Ferrari. Estimating generic 3D room structures from 2D annotations. *NeurIPS*, 2023. 2

[57] Martin Runz, Kejie Li, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, Ian Reid, Lourdes Agapito, Julian Straub, Steven Lovegrove, et al. FroDO: From detections to 3D objects. In *CVPR*, 2020. 3

[58] 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, 6, 7, 8

[59] Lukas Schmid, Jeffrey Delmerico, Johannes Schönberger, Juan Nieto, Marc Pollefeys, Roland Siegwart, and Cesar Cadena. Panoptic multi-TSDFs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency. In *ICRA*, 2022. 2

[60] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer Graphics Forum*, 2007. 2

[61] Daniel Seichter, Benedict Stephan, Söhnke Benedikt Fischedick, Steffen Mueller, Leonard Rabes, and Horst-Michael Gross. PanopticNDT: Efficient and Robust Panoptic Mapping. In *IROS*, 2023. 2

[62] Shuwei Shao, Zhongcai Pei, Weihai Chen, Xingming Wu, and Zhengguo Li. NDDepth: Normal-distance assisted monocular depth estimation. In *ICCV*, 2023. 2

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

[64] Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulò, Norman Müller, Matthias Nießner, Angela Dai, and Peter Kontschieder. Panoptic lifting for 3D scene understanding with neural fields. In *CVPR*, 2023. 2

[65] Sudipta N Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. Interactive 3D architectural modeling from unordered photo collections. *Transactions on Graphics*, 2008. 2

[66] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020. 5

[67] 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. 6, 8

[68] Jheng-Wei Su, Chi-Han Peng, Peter Wonka, and Hung-Kuo Chu. GPr-Net: Multi-view layout estimation via a geometry-aware panorama registration network. In *CVPR*, 2023. 2

[69] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. iMAP: Implicit mapping and positioning in real-time. In *ICCV*, 2021. 2, 3, 4

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

[71] Bin Tan, Nan Xue, Song Bai, Tianfu Wu, and Gui-Song Xia. PlaneTR: Structure-guided transformers for 3D plane recovery. In *ICCV*, 2021. 1, 2

[72] Bin Tan, Nan Xue, Tianfu Wu, and Gui-Song Xia. NOPE-SAC: Neural one-plane RANSAC for sparse-view planar 3D reconstruction. *PAMI*, 2023. 2

[73] Nikolaos Tsagkas, Oisin Mac Aodha, and Chris Xiaoxuan Lu. Vl-fields: Towards language-grounded neural implicit spatial representations. In *ICRA Workshops*, 2023. 2

[74] Eric Turner and Avideh Zakhor. Watertight planar surface meshing of indoor point-clouds with voxel carving. In *3DV*, 2013. 2

[75] Yuko Uematsu and Hideo Saito. Multiple planes based registration using 3D projective space for augmented reality. *Image and Vision Computing*, 2009. 1

[76] Carlos A Vanegas, Daniel G Aliaga, and Bedrich Benes. Automatic extraction of Manhattan-world building masses from 3D laser range scans. *Transactions on Visualization and Computer Graphics*, 2012. 2

[77] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 3

[78] Shaobo Xia, Dong Chen, Ruisheng Wang, Jonathan Li, and Xinchang Zhang. Geometric primitives in lidar point clouds: A review. *Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2020. 2

[79] 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, 3, 5, 6, 7, 8

[80] Fengting Yang and Zihan Zhou. Recovering 3D planes from a single image via convolutional neural networks. In *ECCV*, 2018. 1, 2, 6

[81] Botao Ye, Sifei Liu, Xueting Li, and Ming-Hsuan Yang. Self-supervised super-plane for neural 3D reconstruction. In *CVPR*, 2023. 2

[82] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3D reconstruction via associative embedding. In *CVPR*, 2019. 1, 2, 3, 4, 5, 6, 7

[83] Zehao Yu, Lei Jin, and Shenghua Gao. P^2net: Patch-match and plane-regularization for unsupervised indoor depth estimation. In *ECCV*, 2020. 2

[84] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction. *NeurIPS*, 2022. 3

[85] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. GO-SLAM: Global optimization for consistent 3D instant reconstruction. In *ICCV*, 2023. 3

[86] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021. 2

[87] Shuaifeng Zhi, Edgar Sucar, André Mouton, Iain Haughton, Tristan Laidlow, and Andrew J. Davison. iLabel: Revealing objects in neural fields. *Robotics and Automation Letters*, 2021. 1, 2, 3, 4, 5

[88] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. NICE-SLAM: Neural implicit scalable encoding for SLAM. In *CVPR*, 2022. 3