Planar Surface Reconstruction from Sparse Views

Linyi Jin   Shengyi Qian   Andrew Owens   David F. Fouhey
University of Michigan
{jinlinyi,syqian,ahowens,fouhey}@umich.edu
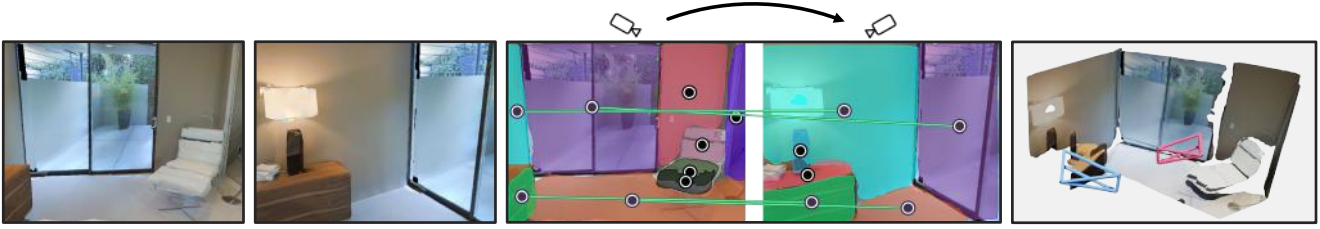https://jinlinyi.github.io/SparsePlanes

Figure 1: Given two RGB images with an unknown relationship, our system produces a single, coherent planar surface reconstruction of the scene in terms of 3D planes and relative camera poses. Our method succeeds despite the wide baseline and the relatively small amount of overlap in structure (here: $64°$ rotation and 1.7m translation and 27% overlap). We show this reconstruction with the inferred left and right cameras in **Blue** and **Red**.

Abstract

The paper studies planar surface reconstruction of indoor scenes from two views with unknown camera poses. While prior approaches have successfully created object-centric reconstructions of many scenes, they fail to exploit other structures, such as planes, which are typically the dominant components of indoor scenes. In this paper, we reconstruct planar surfaces from multiple views, while jointly estimating camera pose. Our experiments demonstrate that our method is able to advance the state of the art of reconstruction from sparse views, on challenging scenes from Matterport3D.

1. Introduction

Consider the two photos in Figure 1, as humans, we can infer that they were taken from the same scene: there is a chair on one side, a bedside table on the other, and a large glass wall and floor in the middle. We perceive the scene correctly despite the fact that they are *sparse views* [37]: they were taken from very different camera poses, and very little of the scene structure within them overlaps. There are also challenges in grouping: while each photo might, at first glance, seem to contain its own glass wall and floor, they are each "slices" of the same planar objects. Despite these challenges, humans readily understand spaces like these from only a few ordinary photos, such as when they share collections of photos from the same event or look for housing.

Yet this setting poses challenges for today's computer vision methods. Traditional tools from multi-view geometry [18, 1] largely rely on correspondence for reconstruction and are fundamentally limited to the small part of the scene that directly overlaps, even when the camera pose is known. Learning-based single view 3D [63, 28], offers ways of reconstructing each image, but produces two messy piles of partial reconstructions due to the unknown viewpoint change. In Figure 1, the floor is fragmented across both views and the back of the chair is present in only one view. While humans can associate these pieces and infer their relative positions to produce a *coherent* reconstruction, it is not trivial for today's reconstruction algorithms.

We believe the ease with which humans solve the two unknown camera reconstruction problem, coupled with the difficulty it poses for computers marks it as an important task on the path to human-level 3D perception. Indeed, it poses challenges to existing work in deep learning for multiview reconstruction, which typically requires known camera poses [23, 31] as opposed to unknown poses, many views [21] as opposed to two, additional depth information at test time [65, 62] as opposed to RGB images, or works only on synthetic data [37]. Typically the extra information used is fundamental to the algorithm (e.g., using poses for triangulation) and cannot be removed to produce a method that works in the two unknown view reconstruction settings.

We propose a learning-based approach that constructs a coherent 3D reconstruction from two views with an un-

known relationship. Our insight, supported empirically, is that progress can be made by jointly tackling three related challenges: per-view reconstruction, inter-view correspondence, and inter-view 6DOF (rotation and translation) pose. Throughout, we use plane segments as our representation since they have simple parameters, are often good approximations [14], and there is a strong line of work for estimating them [29, 28] or related properties [13, 9] from images.

Our method, described in Section 3, combines a deep neural network architecture and an optimization problem to jointly estimate planes, their relationships, and camera transformations. Our architecture builds on PlaneR-CNN [28] to produce, per-input, plane segments and parameters, per-plane embeddings for correspondences, as well as a probability distribution over relative cameras. This information is used in a discrete-continuous optimization problem (along with optional point features) to produce a coherent reconstruction across views. This reasoning across views enables our approach to, for instance in Figure 1, produce a single floor rather than a set of inconsistent floor fragments, and jointly infer the distance from the images to the scene boundaries as well as the relative camera pose.

We validate our approach on realistic renderings from the Matterport3D [4] dataset using pairs with limited overlap (average $53°$ rotation, 2.3m translation, 21% overlap). We report experimental results in Section 4 for three tasks: producing a single coherent reconstruction from the two views, matching planes across views, and estimating the full 6DOF relative camera pose. We compare extensively with a variety of baselines (e.g., adding an independent network to estimate relative pose followed by fusion of the two scene layouts) and ablations that test the contributions of our method and design choices. Our results demonstrate the value of joint consideration of the interrelated problems: our approach substantially outperforms the fusion of existing approaches to the independent problems of camera pose estimation and scene structure estimation.

2. Related Work

The goal of this paper is to produce a coherent 3D plane surface reconstruction of the scene given two images with an unknown relationship between the cameras. To solve the problem, our approach needs to both reconstruct 3D shapes from 2D images and establish correspondence and identify the relative camera pose between views. Our work therefore touches on many topics in 3D computer vision, ranging from single-view reconstruction to correspondence to relative camera view prediction to two-view stereo.

Much of our signal comes from reconstruction methods that map 2D images to 3D structure. This has long been a goal of computer vision with methods that aim to extract normals [9, 57], voxels [6, 16], and depth [9, 39] from 2D images. Our approach builds most heavily on a work aim-

ing to produce a planar reconstruction [29, 63, 28, 67, 5, 22]. We build upon the work in this area, in particular PlaneR-CNN [28], but we use it to build a planar reconstruction from two perspective images (i.e., what an ordinary person's cell phone might capture casually). This focus on perspective images separates our work from approaches that use panorama images [71, 70, 52, 64].

In the process, we find correspondence between planar regions of the scene. Correspondence is, of course, one of the long-standing problems in computer vision. A great deal of work aims to describe patch-based regions, ranging from classic SIFT descriptors [30] to learned descriptors [32, 2, 43, 8], which are often paired with projective geometry [18]; other work finds matches across object-level correspondence [3] or planar-level correspondence [40]. We see our work of jointly reconstructing and identifying correspondence as complementary to this work; a core component is an embedding network that aims to describe the plane, like [3] and the last component of our system uses VIP-like features [60] to constrain plane parameters. Our approach, however, solves a superset of these problems, since it produces a reconstruction as well.

We additionally predict the relative transformation between the cameras. This has been studied as an output of correspondence methods from both a classical and learning-based perspective. There are, however, methods that directly try to predict these transformations, including by deep networks [37, 12], aligning RGBD data [41, 65, 11], or learning to predict the fundamental matrix [38, 35, 55]. Like these approaches, we aim to estimate the relative transformation between the images as a component (although unlike some, we assume only ordinary RGB images); improving this component is complementary to our goals.

The relationship between reconstruction, correspondence, and camera pose, and the value of planes for inference has long been well-understood in the stereo community. While we use two views, these are well-separated and unknown, so our approach is different compared to standard two-view stereo [45] or visual SLAM [40, 59, 47] and more similar to wide-baseline stereo [36, 33]. Unlike wide-baseline stereo systems, however, we also produce reconstructions for portions of the scene that are seen in only one camera. Nonetheless, we draw inspiration from works in this community that use planes as a useful unit of inference [49, 14, 15, 17].

The most similar work in this direction is Associative3D [37], which solves a related problem of reconstructing volumetric objects. Our approach is inspired by [37], but overcomes several methodological limitations, which we experimentally show. Associative3D uses one network for detecting objects and another for relative camera pose, and fuses the two via a heuristic RANSAC-like scheme. While effective on a six-object subset of the clean synthetic
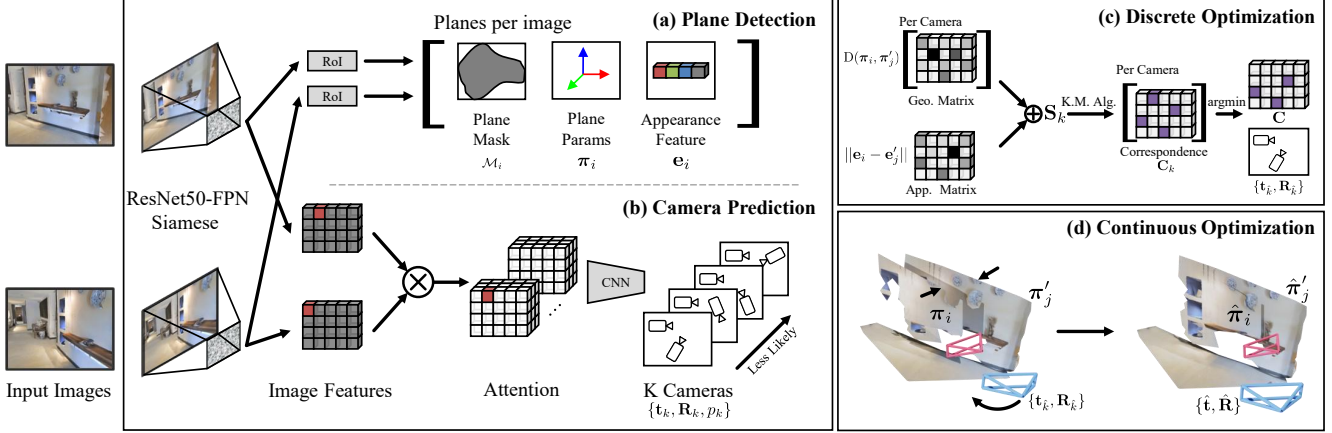
Figure 2: **Our Approach.** Given a pair of images, we use a ResNet50-FPN to detect planes and predict probabilities of relative camera poses, and use a two-step optimization to generate a coherent planar reconstruction. (a) For each plane, we predict a segmentation mask, plane parameters, and an appearance feature. (b) Concurrently, we pass image features from the detection backbone through the attention layer and predict the camera transformation between views. (c) Our discrete optimization fuses the prediction of the separate heads to select the best camera pose and plane correspondence. (d) Finally, we use continuous optimization to update the camera and plane parameters.

SUNCG dataset [50], these components fall short on the realistic Matterport3D dataset [4] that our approach uses. Meeting the challenge of handling with planar segments (which cover objects *and* layouts) in realistic data requires a more principled optimization strategy and better signal for relative camera estimation. As another benefit, our approach has one backbone network forward pass per image.

3. Approach

Our approach, depicted in Figure 2, aims to map two images with an unknown relationship to a set of globally consistent planes and relative camera poses. This task requires estimating plane parameters, reasoning about their relationship (to avoid, e.g., a reconstruction with two floors), and inferring the relative pose. These subproblems are related since, for instance, plane parameters and correspondence constrain relative camera pose. We approach this with a network that predicts parameters and embeddings for planes (Sec. 3.1) and a distribution over relative camera pose (Sec. 3.2), followed by a joint optimization (Sec. 3.3) to produce a final coherent scene via joint reasoning. At training time, our system depends on RGBD for supervision, but can run on ordinary RGB images at test time.

3.1. Plane Prediction Module

Our plane prediction module produces, per-image, a set of plane segments that also have an embedding for cross-view matching. Each plane has: a segment $\mathcal{M}_i$; plane parameters $\boldsymbol{\pi}_i = [\mathbf{n}_i, o_i]$ (where $\mathbf{n}_i$ is a unit vector normal and o_i the offset) giving the plane equation $\boldsymbol{\pi}_i^T[x, y, z, -1] = 0$; and a unit-norm embedding $\mathbf{e}_i$ for cross-view matching.

Throughout, we denote planes in view 2 as $\mathcal{M}'_j, \boldsymbol{\pi}'_j, \mathbf{e}'_j$, etc.
Plane detection. We extend PlaneRCNN [28] to detect planes in each image. During inference, the system produces backbone features using ResNet50-FPN [26]. We use a region proposal network to propose boxes and then infer plane masks and normals from features from RoIAlign [19]. At the same time, a decoder maps the backbone feature to a depthmap, which is used to compute the plane offset.

Appearance embedding. We additionally predict a cross-view embedding by training the network on *pairs* of images via triplet loss. Given plane correspondence, we form cross-view triplets $\mathbf{e}_a, \mathbf{e}'_p, \mathbf{e}'_n$ where **a**nchor $\mathbf{e}_a$ corresponds with the **p**ositive match $\mathbf{e}'_p$ and not the **n**egative match $\mathbf{e}'_n$. We minimize a standard triplet loss [48] $\max(||\mathbf{e}_a - \mathbf{e}'_n||_2 - ||\mathbf{e}_a - \mathbf{e}'_p||_2 + \alpha, 0)$, which gives a loss if the anchor and positive are not closer than the anchor and the negative by a margin $\alpha = 0.2$. We use online triplet mining and randomly pick negative matches with positive loss.

3.2. Camera Pose Module

Our camera pose module estimates a distribution over the relative camera pose between views. This enables joint reasoning between a holistic estimate of the camera as well as the valuable cues in the estimated geometry: plane correspondences and parameters can constrain relative poses. We produce a distribution $p_k > 0, \sum_k p_k = 1$ over a set of discrete pairs of rotations and translations $\{(\mathbf{t}_k, \mathbf{R}_k)\}$).

Our network outputs two independent multinomial distributions over translation and rotation using attention-style features. The joint translation/rotation distribution is then their product. Our attention features follow recent literature

that find similarity between images using attention [58, 56, 42] and capture, at each pixel in a feature map, the relative similarity between that pixel and the pixels in the other image. We use the backbone network to extract c-dimensional feature maps F_1, F_2 of size $c \times h_1 \times w_1$ and $c \times h_2 \times w_2$. Suppose p_1 and p_2 index pixels in feature F_1 and F_2 over both rows and columns (i.e., $F_1(p_1)$ is a c-dimensional column), we then compute the $(h_2 w_2) \times (h_1 w_1)$ attended feature A:

$$A(p_1, p_2) = \frac{\exp(F_2(p_2)^\intercal F_1(p_1))}{\Sigma_{p_2} \exp(F_2(p_2)^\intercal F_1(p_1))}. \quad (1)$$

We reshape this to a $h_2 w_2$-channel $h_1 \times w_1$ feature map where each pixel represents the normalized correlation with each of the other pixels in feature map 2. We apply a convolutional network to the reshaped A with six layers of 3×3 convolutions and two fully connected layers that predicts the camera distribution. We found this attention to be superior compared to the strategy of concatenating average-pooled vector outputs in [37, 12]. as well as other alternate strategies in experiments in Section 4. We train the camera pose module on top of the ResNet50-FPN's P_3 feature [26].

3.3. Optimization

Our final step is an optimization that serves two purposes. First, it propagates information between the related problems: the parameters of a mutually visible plane can inform relative camera poses, and a better pose of a camera can help disambiguate matches. Second, it ensures the coherence: naively concatenating two single-view parses of the scene (as we show empirically) yields a collection of possibly overlapping, inconsistent planar fragments showing the same object from different vantage points.

We cast this as an optimization problem over camera-to-camera transformation $\hat{\mathbf{R}}, \hat{\mathbf{t}}$ and set of plane parameters $\hat{\boldsymbol{\pi}}_i, \hat{\boldsymbol{\pi}}'_j$ as well as a plane correspondence matrix $\mathbf{C} \in \{0,1\}^{m \times n}$ where $\mathbf{C}_{i,j}$ is 1 if and only if plane i corresponds to plane j. As input, the optimization has a series of m planes for the view 1 $\{\mathcal{M}_i, \boldsymbol{\pi}_i, \mathbf{e}_i\}$ and n planes for view 2 $\{\mathcal{M}'_j, \boldsymbol{\pi}'_j, \mathbf{e}'_j\}$, and a distribution over relative camera transformations $\{\mathbf{t}_k, \mathbf{R}_k, p_k\}$. The optimization tightly couples discrete and continuous variables and has many degenerate solutions. We thus follow a two stage approach: we hold plane parameters fixed and solve for plane correspondence and rough camera location; we then continuously optimize camera and plane parameters.

Discrete Problem: We first solve a discrete problem that selects a camera from the K options and the plane correspondence matrix $\mathbf{C}$. The most important term is expressed via a $m \times n$ cost matrix $\mathbf{S}_k$ encoding the quality of a plane correspondence $\mathbf{C}$ assuming camera k has been selected. Assuming the second view's plane parameters have been transformed into the first view by camera k (omitted for clarity), the cost matrix $\mathbf{S}_k$ (with trade-off parameters λ)

is

$$(\mathbf{S}_k)_{i,j} = \lambda_e ||\mathbf{e}_i - \mathbf{e}'_j|| + \lambda_n \mathrm{acos}(|\mathbf{n}_i^\intercal \mathbf{n}'_j|) + \lambda_o |o_i - o'_j| \quad (2)$$

which includes terms for the embedding, normal, and offset which are 0 with perfect matches.

This cost term is combined with two regularizing terms. One, $-\log(p_k)$, penalizes unlikely cameras via the negative log-likelihood of camera k; the other rewards matching objects: $-\sum_{i,j} \mathbf{C}_{i,j}$. We pick the best correspondence and camera that optimize the objective

$$\arg\min_{\mathbf{C},k} -\lambda_c \log(p_k) + \sum_{i,j} (\mathbf{C} \circ \mathbf{S}_k)_{i,j} - \sum_{i,j} \mathbf{C}_{i,j}, \quad (3)$$

which encourages selecting a likely camera, as many planes as possible, and correspondence that is consistent in both appearance and geometry assuming the camera is correct.

With a fixed camera k, the objective $\sum_{i,j} (\mathbf{C} \circ \mathbf{S}_k)_{i,j}$ can be efficiently solved using the Hungarian algorithm, and the number of matches handled via thresholding. Since there are a finite number (K) of camera hypotheses, this amounts to solving K independent matching problems.

Continuous Problem: Having selected a camera $\hat{k}$ and planar correspondences $\mathbf{C}$, we can then refine the predictions from the deep networks for the camera and planes. We optimize over camera transformations $\hat{\mathbf{R}}, \hat{\mathbf{t}}$ and plane parameters $\hat{\boldsymbol{\pi}}_i, \hat{\boldsymbol{\pi}}'_j$ to minimize geometric distance between the corresponding planes (assuming the same coordinate frame) and pixel alignment error based on pixel-level features:

$$\arg\min_{\hat{\mathbf{R}}, \hat{\mathbf{t}}, \hat{\boldsymbol{\pi}}_i, \hat{\boldsymbol{\pi}}'_j} \sum_{i,j} \mathbf{C}_{i,j}(||\hat{\boldsymbol{\pi}}_i - \hat{\boldsymbol{\pi}}'_j|| + d_{\mathrm{pixel}}(\hat{\boldsymbol{\pi}}_i, \hat{\boldsymbol{\pi}}'_j)) + d_{\mathrm{cam}}(\hat{\mathbf{R}}, \hat{\mathbf{t}})$$
$$(4)$$

where d_{pixel} measures the Euclidean distance of back-projected points that match across corresponding planes. Inspired by [60], we warp texture to viewpoint normalized cameras using the plane parameters to extract viewpoint-invariant SIFT [30] features. The term d_{cam} regulates the deviation from the selected camera bin. We initialize the optimization at $\mathbf{R}_{\hat{k}}, \mathbf{t}_{\hat{k}}, \boldsymbol{\pi}_i, \boldsymbol{\pi}'_j$, and optimize with a trust-region reflective minimizer (see supplemental for details). We parameterize rotations as 6D vectors following [69].

Merging Planes: Given planes in correspondence and camera transformations, we merge them in the global frame. We merge offsets by averaging and normals $\{\mathbf{n}_i\}$ by solving for the $\hat{\mathbf{n}}$ maximizing $\sum_i (\hat{\mathbf{n}}^\intercal \mathbf{n}_i)^2$ via an eigenvalue problem.

3.4. Implementation Details

A full detailed description appears in the supplemental. We use Detectron2 [61] to implement our network. The plane detection backbone uses ResNet50-FPN pretrained on COCO [27]. We directly regress normals instead of using classification in [28]. The camera branch and the plane embedding are trained using a Siamese network whose backbone is the plane detection backbone. During training, we

Image 1	Image 2	Prediction	Ground Truth	Prediction	Ground Truth

Figure 3: Results on Matterport3D test set. **Blue** and **Red** frustums show cameras for image 1 and 2. More results are in the supplemental.

first train the plane prediction backbone on single images and then freeze the network. We train the plane appearance embedding and camera pose module on the frozen backbone. We fit all trade-off parameters (*e.g.*, λ_n) on the validation set using randomized search. We run k-means clustering and spherical k-means clustering on the training set to produce 32 bins for translation and rotation respectively.

4. Experiments

We evaluate our approach using renderings of real-world scenes. Our approach generates a new, rich and coherent output in terms of a planar reconstruction plus a geometric relationship between the two views. In the process of producing this reconstruction, it solves two other problems: predicting object correspondences and relative camera pose estimation. We therefore evaluate our model in three ways, which each naturally have their own metrics and baselines: the full model (Section 4.2), the plane correspondence (Section 4.3) and the relative camera poses (Section 4.4).

4.1. Experimental Setup

We use renderings of real scenes from Habitat [44] and Matterport3D [4]. Habitat enables the rendering of realistic

images and provides ground truth pose and depth that can be used for evaluation. We stress that our image pairs have far less overlap compared to other settings with an average overlap of just 21% vs 64% used by DeMoN [55] on Sun3D [62] or 80% by the plane odometry method [41] (see supplement for more statistics).

Dataset: Our training, validation, and test set consist of 31932, 4707, and 7996 image pairs respectively. We fit our ground truth planes on Matterport3D using a standard RANSAC approach, using semantic labels to constrain the planes, following [29]. We fit these planes on the 3D meshes and render them to 2D, which provides us with plane correspondence even when the planes have limited overlap. We start with a pool of single of images following [68] and select pairs with ≥ 3 common planes and ≥ 3 unique planes like [37].

4.2. Full Scene Evaluation

We begin by evaluating our approach's ability to produce full scene reconstruction in terms of a set of 3D plane segments in the scene in a common coordinate frame.

Metrics: We follow other approaches that reconstruct the scene factored into components [53, 24, 25, 34, 37] and treat
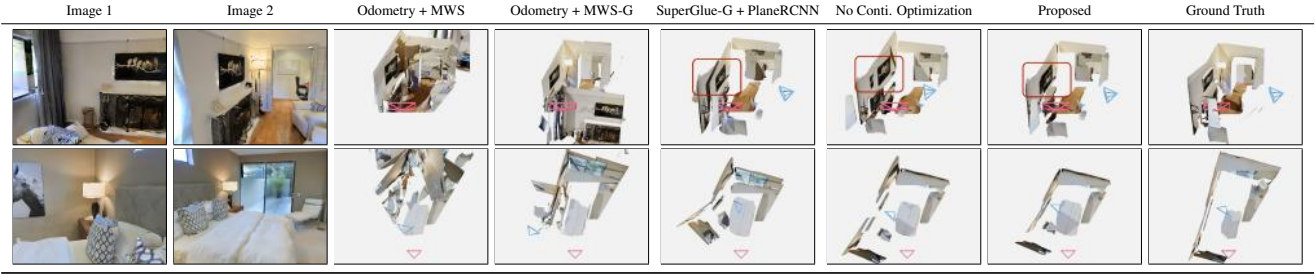
Image 1	Image 2	Odometry + MWS	Odometry + MWS-G	SuperGlue-G + PlaneRCNN	No Conti. Optimization	Proposed	Ground Truth

Figure 4: Comparison between baselines. **Row 1:** Shown in red boxes, SuperGlue + PlaneRCNN does not predict the plane correspondence; our discrete optimization merges the same wall from two views into one plane; our proposed method further aligns the texture on the wall. **Row 2:** Plane Odometry + MWS-G aligns plane normals but with incorrect correspondence.

the full problem like a detection problem, evaluated using average precision (AP). Given a ground-truth decomposition into components (in this case, planes), we evaluate how well the approach detects and successfully reconstructs each one. We define a true positive as a detection satisfying three criteria: (i) (*Mask*) mask intersection-over-union ≥ 0.5; (ii) (*Normal*) surface normal distance, $\mathrm{acos}(|\mathbf{x}_1^\mathsf{T}\mathbf{x}_2|) \leq 30°$; and (iii) (*Offset*) offset distance, $|o_1 - o_2| \leq 1\mathrm{m}$.

Baselines: We compare with alternate approaches and ablations that test the system's contributions. To the best of our knowledge, no existing work solves our task, so we test fusions of existing systems. These baselines estimate a relative camera pose and per-image planes.

Plane Odometry [41] + MWS [14] / PlaneRCNN [28]: [41] extracts plane-primitives from successive RGB-D video frames and uses plane-to-plane registration to solve for the camera pose. For the 3D representation, we either predict planes using PlaneRCNN in our plane prediction module or run Manhattan-world Stereo [14] on predicted depth (MWS) or ground truth depth (MWS-G). We estimate depthmaps with [39], a state of the art system trained on ≈2M images. We found that MWS gave better plane fits on depthmaps without semantic labels compared to our RANSAC approach, also shown in [28, 29]. Plane Odometry mainly reasons with geometric information when producing its final output and tests a simple fusion of visual odometry and plane extraction systems.

SuperGlue-G [43] + MWS [14] / PlaneRCNN [28]: [43] is a point-based matching algorithm that uses deep features to find pixel correspondence across images to estimate an essential matrix. The scale of the translation is not determined by the essential matrix so we use the *ground truth* translation scale for this method. Approaches like [43] are complementary to our camera branch since they have finer resolution but no object-size prior; our approach could integrate [43] in its continuous optimization. Like the previous baseline, we try each of MWS/MWS-G/Plane-RCNN.

RPNet [12] + PlaneRCNN [28]: We apply the PlaneRCNN module used by our system and our improved variant of RP-Net [12] – see Sec. 4.4 for more details. This is equivalent

to our system, but ignoring our optimization or reasoning over plane parameters. This tests the value of joint reasoning while holding base networks fixed.

Appearance Embedding Only: We run the optimization with only the appearance embedding $\mathbf{e}_i$ predicted for each plane. This gives a fusion of planes based on learned embeddings, which tests the value of our geometric reasoning.

Associative3D [37] Optimization: We apply a heuristic RANSAC-style optimization scheme from the most closely related paper [37]. We adapt it to our setting for fair comparison (see supplemental for detail). This tests the value of our optimization, which casts camera selection and matching in terms that are solvable via principled algorithms.

No Continuous Optimization: We perform the full method without the continuous optimization in Section 3.3.

Proposed: We apply the full proposed method.

Qualitative Results: We show qualitative results in Figure 3. Prediction and ground truth are shown in two novel views to see all planes and camera poses in the scene. Our method predicts accurate camera poses from inputs with small overlap, merges planes across different views, and aligns their textures to produce a coherent reconstruction. More examples are in the supplement.

We show comparison between our proposed approach and baselines in Figure 4. *Plane Odometry* fails on predicted depth due to distortion and rarely finds the correct correspondences between planes, *e.g.* incorrect walls are aligned. *SuperGlue-G* can usually obtain a reasonable camera pose given oracle translation scale but does not merge planes across views, *e.g.*, non-parallel walls. *No Continuous Optimization* finds and merges the corresponding planes using the Hungarian algorithm in our discrete optimization. However, simply merging the corresponding planes still generates erroneous results, *e.g.*, unaligned texture of pictures on the wall in row 1, disconnected walls in row 2, which are caused by discretization bins of the camera transformation and inaccurate single-view prediction. Our proposed approach refines the camera and planes further, so that it fixes the inconsistency of plane detection and camera modules and generates reconstructions that are significantly

Table 1: Average Precision, treating reconstruction as 3D plane detection with different definitions of true positive. *All:* Mask IoU ≥ 0.5, Normal error $\leq 30°$, Offset error ≤ 1m. *-Offset* and Normal ignore the offset and normal condition respectively.

Methods	All	-Offset	-Normal
Odometry [41] + MWS [14, 39]	4.80	6.72	6.12
Odometry [41] + MWS-G [14]	16.42	19.57	18.81
Odometry [41] + PlaneRCNN [28]	21.33	27.08	24.99
SuperGlue-G [43] + MWS [14, 39]	7.56	9.06	8.64
SuperGlue-G [43] + MWS-G [14]	23.01	23.89	25.22
SuperGlue-G [43] + PlaneRCNN [28]	30.06	33.24	33.52
RPNet [12] + PlaneRCNN [28]	29.44	35.25	31.67
Appearance Embedding Only	33.04	39.78	36.85
Associative3D [37] Optimization	33.01	39.43	35.76
No Continuous Optimization	35.87	**42.13**	38.80
Proposed	**36.02**	42.01	**39.04**

closer to the ground truth.

Quantitative Results: We show results in Table 1. Fitting planes on predicted depthmaps (*MWS*) fails poorly due to distorted predictions. This is a known challenge (*e.g.*, addressed by concurrent work [66] without public code at time of submission). Methods using *PlaneRCNN* outperform *MWS* even with ground truth (GT) depth, also found in [29, 28]; this is because of PlaneRCNN's stronger plane labels improved by semantics. *Plane Odometry* finds pose primarily with plane normals. Its correspondence is often wrong because our low-overlap setting limits the information in geometric cues alone (in contrast to the combined normal, offset, and texture used by our method). SuperGlue with GT translation scale (*SuperGlue-G*) slightly outperforms the *RPNet* with PlaneRCNN, but falls short of the proposed full method by 6 AP. Some of this gap is because there are duplicate copies of planes found in both images. One contribution of our method is preventing this case. While merging based on appearance only partially solves the problem, the full method, which incorporates geometry, does the best. The particular optimization for this merging is moreover important: the heuristic randomized optimization of [37] does worse. The continuous optimization makes little difference in planes but improves the camera considerably (Sec. 4.4). Additional experiments on AP vs. overlap are in the supplement.

4.3. Plane Correspondence

One core challenge for generating a single coherent reconstruction from two images is ensuring that each visible plane appears exactly once rather than being split. We therefore evaluate how well we can identify correspondences between planes across views.

Baselines: We compare the proposed approach with three

Table 2: Plane correspondence on our test set. IPAA-X [3] measures the fraction of pairs with no less than X% of planes associated correctly. Ground truth bounding boxes are used.

	IPAA-100	IPAA-90	IPAA-80
Appearance Only	6.8	23.5	**55.7**
ASNet [3] on ROI	6.9	21.7	52.1
Associative3D [37] Opt.	0.9	10.7	44.0
Proposed	**16.2**	**28.1**	55.3

baselines that test the contribution of each component of our method. We measure performance using ground-truth bounding boxes during evaluation so that we measure only errors due to correspondence and not detection.

Appearance Only: We apply the Hungarian algorithm to the appearance embedding and do not use geometry. This tests how important geometry is to identifying correspondence.

MessyTable [3] ASNet on ROI: We adapt the ASNet [3] framework to our setting and architecture for fair comparison (see supplemental for details). This network improves matching by adding an additional pathway that has more context. This tests whether our matching is constrained by looking only inside a proposal region.

Associative3D [37] Optimization: We again apply the optimization method of [37], following Section 4.2.

Proposed: We apply the full approach.

Metrics: We aim to measure how well methods associate planes in a pair of images. We therefore evaluate performance directly with Image Pair Association Accuracy (IPAA) metric by Cai *et al*. [3], which represents the fraction of image pairs with no less than X% of the planes associated correctly (written as IPAA-X). We use ground truth bounding boxes in this section for evaluation.

Quantitative Results: Table 2 shows that our method outperforms all other benchmarks on IPAA-100 and IPAA-90 and has a slight drop of 0.4% in IPAA-80. Our full optimization improves IPAA-100 by a large margin (9.4%) compared to using the appearance embedding matrix only. We hypothesize our approach outperforms [37] due to the factorial growth of the search space, which the heuristic approach struggles with. We suspect that [3] struggles since it is designed for table-top matching, where the planar table constrains how much context can change from view to view (in contrast to our strongly non-planar scenes).

Qualitative Results: *Appearance Only* and *ASNet* do not consider geometric information; therefore, they have difficulty distinguishing planes of similar texture, *e.g.*, two sides of the bed in Figure 5. *Associative3D* does not find all the correspondence due to its random search scheme. Our method detects all the correspondence and distinguishes similar texture planes using geometric information.
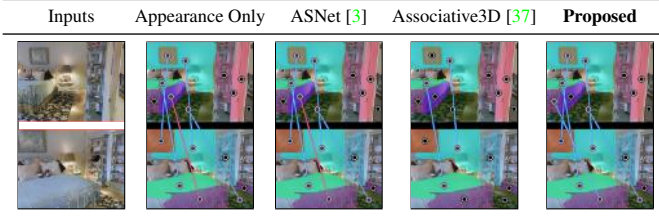
Figure 5: Representative correspondence prediction, showing true positive matches in **Blue** and false positives in **Red**.

Figure 6: Representative failure modes. **Row 1:** The kitchen countertop has little overlap across views. Though our method stitches them, there is no correct pixel correspondence to constrain the alignment, shown in the **red** box. **Row 2:** The system fails on multiple coplanar segments with very similar texture.

Table 3: Evaluation of relative camera pose. Rows 1-4 compare with other baselines; Rows 5-6 analyze increasing amounts of optimization. We show median, mean error and % error $\leq$ 1m or 30° for translation and rotation.

Method	Trans.			Rot.		
	Med.	Mean	($\leq$1m)	Med.	Mean	($\leq$30°)
[41] + GT Depth	3.20	3.87	16.0	50.43	55.10	40.9
[41] + [39]	3.34	4.00	8.30	50.98	57.92	29.89
Assoc.3D [37]	2.17	2.5	14.8	42.09	52.97	38.1
SuperGlue [43]	-	-	-	**3.88**	24.17	77.84
No Optimization	0.90	1.40	55.5	7.65	24.57	81.9
No Continuous	0.88	1.36	56.5	7.58	22.84	**83.7**
Proposed	**0.63**	**1.25**	**66.6**	7.33	**22.78**	83.4

4.4. Relative Camera Pose Estimation

Our final experiment tests the effectiveness of the predicted relative camera pose. We compare our method with a number of alternate approaches and baselines. Additional experiments testing the use of attention features rather than concatenation features appear in the supplement.

Metrics: Following prior works [51, 54, 37, 10], we measure the camera rotation and translation by geodesic rotation distance and Euclidean distance separately. We report the mean and median error, as well as the fraction below a fixed threshold (30° and 1m following [37]).

Baselines: We compare our full system with four baselines and two ablations.

Plane Odometry [41] + GT Depth: We apply the RGBD plane odometry system from [41] to the ground-truth depth. As described in Section 4.2, this approach mainly reasons geometrically, without learned priors or deep features.

Plane Odometry [41] + [39]: We apply the same approach, using estimated depth from [39] for a more fair comparison.

Associative3D [37] camera branch: We apply an improved version of the camera branch of Associative3D, which is an improved version of the RPNet [12]. This method average pools ResetNet50 features [20] followed by a MLP. For fair comparison, we upgrade the MLP to match the number of layers and parameters of our approach. This tests the value of our improved attention features.

[43]: We apply the SuperGlue system. Although as a pixel-correspondence-based method, we see this as a complementary line of work to our method. SuperGlue estimates camera pose via the essential matrix and thus cannot estimate the translation's scale. We note that the approach in Section 4.2 used the ground-truth camera translation scale to give an upper bound if the true translation scale were used. The approach fails on 11.6% of our pairs due to insufficient correspondence; in this case, we assign the identity matrix for rotation. We use the only available model, which was trained on ScanNet [7].

No Optimization: We apply our camera branch and take the most likely camera without any joint reasoning.

No Continuous: We apply our approach but do not apply continuous optimization.

Proposed: We apply the full approach.

Quantitative Results: We report results in Table 3. The full method has low median errors of 0.63m/7.33°. Comparing with other approaches, we find that our approach also substantially outperforms the Plane Odometry [41] method regardless of whether ground truth or predicted depth is used. Just as described in Section 4.2, this is due to the limited amount of information that is available in geometry alone. SuperGlue can be extraordinarily accurate in rotation when textures provide strong correspondences across views, but also can fail to make inferences on hard scenes.

The most likely camera from our approach (*No Optimization*) substantially outperforms the previous method of [37], which follows a simpler average pooling. This is largely due to our use of attention features; additional more detailed ablations are in the supplement. Discrete optimization consistently slightly improves results, but continuous optimization substantially improves translation with marginal gains and losses in rotation performance.

View 1	View 2	View 3	Prediction

Figure 7: Extending the system to > 2 views. We stitch two views together and stitch the third view incrementally.

5. Conclusion

We've presented a learning-based system to produce a coherent planar surface reconstruction from two unknown views. Our results suggest that jointly considering correspondence and reconstruction improves both, although our experiments suggest room for growth (*e.g.*, see typical failure modes in Figure 6, including low-overlap planes and repeated coplanar and similar segments). Future directions include using our framework to reconstruct a more complete scene with fewer frames than traditional SfM [46]. Figure 7 shows examples that extend our system to > 2 views by incrementally stitching new views on reconstructed views.

References

[1] Sid Yingze Bao, Mohit Bagra, Yu-Wei Chao, and Silvio Savarese. Semantic structure from motion with points, regions, and objects. In *CVPR*, 2012. 1

[2] Axel Barroso-Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key.net: Keypoint detection by handcrafted and learned cnn filters. In *ICCV*, 2019. 2

[3] Zhongang Cai, Junzhe Zhang, Daxuan Ren, Cunjun Yu, Haiyu Zhao, Shuai Yi, Chai Kiat Yeo, and Chen Change Loy. Messytable: Instance association in multiple camera views. In *ECCV*, 2020. 2, 7, 8, 17, 19

[4] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, 2017. 2, 3, 5, 18

[5] Weifeng Chen, Shengyi Qian, David Fan, Noriyuki Kojima, Max Hamilton, and Jia Deng. Oasis: A large-scale dataset for single image 3d in the wild. In *CVPR*, 2020. 2

[6] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 2

[7] 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. 8

[8] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. *CVPR*, 2019. 2

[9] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 2

[10] Mohamed El Banani, Jason J Corso, and David F Fouhey. Novel object viewpoint estimation through reconstruction alignment. In *CVPR*, 2020. 8

[11] Mohamed El Banani, Luya Gao, and Justin Johnson. Unsupervisedr&r: Unsupervised point cloud registration via differentiable rendering. In *CVPR*, 2021. 2

[12] Sovann En, Alexis Lechervy, and Frédéric Jurie. Rpnet: An end-to-end network for relative camera pose estimation. In *ECCV*, 2018. 2, 4, 6, 7, 8

[13] David F. Fouhey, Abhinav Gupta, and Martial Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013. 2

[14] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Manhattan-world stereo. In *CVPR*, 2009. 2, 6, 7

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

[16] R. Girdhar, D.F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 2

[17] Simon Hadfield and Richard Bowden. Exploiting high level scene cues in stereo reconstruction. In *ICCV*, 2015. 2

[18] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2004. 1, 2

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 3, 18

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 8

[21] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *CVPR*, 2018. 1

[22] Ziyu Jiang, Buyu Liu, Samuel Schulter, Zhangyang Wang, and Manmohan Chandraker. Peek-a-boo: Occlusion reasoning in indoor scenes with plane representations. In *CVPR*, 2020. 2

[23] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NeurIPS*, 2017. 1

[24] Nilesh Kulkarni, Ishan Misra, Shubham Tulsiani, and Abhinav Gupta. 3d-relnet: Joint object and relational network for 3d prediction. In *ICCV*, 2019. 5

[25] Lin Li, Salman Khan, and Nick Barnes. Silhouette-assisted 3d object instance reconstruction from a cluttered scene. In *ICCV Workshops*, 2019. 5

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 3, 4, 14

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 4, 18

[28] 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, 4, 6, 7, 18

[29] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *CVPR*, 2018. 2, 5, 6, 7, 18

[30] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 2, 4, 19

[31] 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. 1

[32] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *NeurIPS*, 2017. 2

[33] Dmytro Mishkin, Jiri Matas, Michal Perdoch, and Karel Lenc. Wxbs: Wide baseline stereo generalizations. *BMVC*, 2015. 2

[34] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *CVPR*, 2020. 5

[35] Omid Poursaeed, Guandao Yang, Aditya Prakash, Qiuren Fang, Hanqing Jiang, Bharath Hariharan, and Serge Belongie. Deep fundamental matrix estimation without correspondences. In *ECCV*, 2018. 2

[36] Phil Pritchett and Andrew Zisserman. Wide baseline stereo matching. In *ICCV*, 1998. 2

[37] Shengyi Qian, Linyi Jin, and David F. Fouhey. Associative3d: Volumetric reconstruction from sparse views. In *ECCV*, 2020. 1, 2, 4, 5, 6, 7, 8, 14, 17, 18, 20

[38] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *ECCV*, 2018. 2

[39] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020. 2, 6, 7, 8

[40] Carolina Raposo and Joao P Barreto. πMatch: Monocular vslam and piecewise planar reconstruction using fast plane correspondences. In *ECCV*, 2016. 2

[41] Carolina Raposo, Miguel Lourenço, Michel Antunes, and João Pedro Barreto. Plane-based odometry using an rgb-d camera. In *BMVC*, 2013. 2, 5, 6, 7, 8, 12, 13

[42] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. In *NeurIPS*, 2018. 4

[43] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 2, 6, 7, 8, 12

[44] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 5, 18

[45] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002. 2

[46] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 9

[47] Thomas Schops, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *CVPR*, 2019. 2

[48] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 3

[49] Sudipta N Sinha, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *ICCV*, 2009. 2

[50] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017. 3

[51] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *ICCV*, 2015. 8

[52] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *CVPR*, 2019. 2

[53] Shubham Tulsiani, Saurabh Gupta, David F Fouhey, Alexei A Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2d image of a 3d scene. In *CVPR*, 2018. 5

[54] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *CVPR*, 2015. 8

[55] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *CVPR*, 2017. 2, 5, 12, 13

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

[57] Xiaolong Wang, David F. Fouhey, and Abhinav Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015. 2

[58] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. 4

[59] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J Leonard, and John McDonald. Robust real-time visual odometry for dense rgb-d mapping. In *ICRA*, 2013. 2

[60] Changchang Wu, Brian Clipp, Xiaowei Li, Jan-Michael Frahm, and Marc Pollefeys. 3d model matching with viewpoint-invariant patches (vip). In *CVPR*, 2008. 2, 4

[61] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 4, 18

[62] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013. 1, 5, 12

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

[64] Shang-Ta Yang, Fu-En Wang, Chi-Han Peng, Peter Wonka, Min Sun, and Hung-Kuo Chu. Dula-net: A dual-projection network for estimating room layouts from a single RGB panorama. In *CVPR*, 2019. 2

[65] Zhenpei Yang, Siming Yan, and Qixing Huang. Extreme relative pose network under hybrid representations. In *CVPR*, 2020. 1, 2

[66] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image. In *CVPR*, 2021. 7

[67] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. In *CVPR*, 2019. 2

[68] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *CVPR*, 2017. 5, 18

[69] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019. 4

[70] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *CVPR*, 2018. 2

[71] Chuhang Zou, Jheng-Wei Su, Chi-Han Peng, Alex Colburn, Qi Shan, Peter Wonka, Hung-Kuo Chu, and Derek Hoiem. 3d manhattan room layout reconstruction from a single 360 image. *arXiv preprint arXiv:1910.04099*, 2019. 2

A. Additional Results

A.1. Dataset

To give additional context for the dataset, we transform the point cloud of view 1 to view 2. The average percentage of overlapping points of our dataset is 21%. For context, Sun3D [62] image pairs used by DeMoN [55] had an average overlap of 64%; FR2_desk and FR3_structure used by Plane Odometry [41] had an overlap of 78%. Figure 9 shows the histogram of overlap ratio of our dataset versus the other datasets.

Random examples are shown in Figure 10. Our dataset is much more sparse than datasets used in DeMoN and Plane Odometry, because they sample adjacent frames from a video, while we render a sparse view dataset ourselves.

A.2. Performance vs. Overlapping Ratio

Figure 8 shows the performance of our method and other baselines with images of different overlap ratios. Our method has higher AP over other baselines across all overlap ratios in our dataset. Overlap ratio of around 0.05 is an extremely hard setting, our method (29 AP) and improved RPNet (27 AP) perform better than matching based methods (<24 AP) because SuperGlue [43] rarely finds enough correspondence. Deep networks, on the other hand, can use other cues and prior to make predictions. When the overlap ratio is higher, our method can also outperform other baselines (increase by over 5 AP when overlap ratio is around 0.25 and 11 AP when overlap is about 0.55). We can merge planes based on correspondence to produce a coherent reconstruction while other baselines produce inconsistent and duplicated planes.

A.3. Rank by Single-view Prediction

We further investigate how the quality of single-view prediction (outputs from plane prediction module on each image) will affect our results. Table 4 shows results on top 25%, 50%, 75% and 100% of the test examples ranked by single-view AP. On top 25% examples where the single-view prediction are more accurate, our proposed methods have a higher AP (increase by over 10 points compared to 100% data). Moreover, our proposed method has a higher gain over ablation baselines *Associative3D optimization* and *Appearance Embedding only* by above 4.9 points on top 25% examples and even higher (> 10 points) over the other external baselines. It is still significant but slightly lower when we have worse single-view predictions.
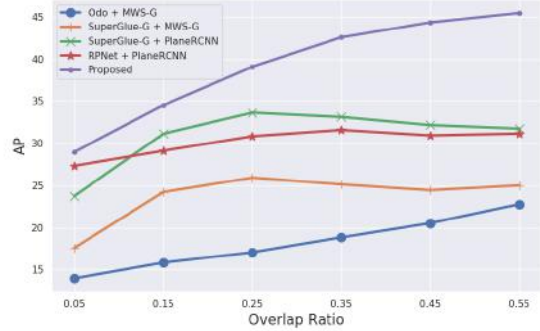

Figure 8: AP vs. different overlap ratios.

Table 4: We further investigate how the quality of single-view prediction will affect our results. We include results on top 25%, 50%, 75% and 100% of the test examples ranked by single-view prediction AP. Similar to Table 1 in our paper, a prediction is a true positive only if Mask IoU ≥ 0.5, Normal error $\leq 30°$ and Offset error ≤ 1m.

| Methods	Top 25%	Top 50%	Top 75%	100%
PlaneOdometry + MWS	7.26	6.22	5.56	4.80
PlaneOdometry + MWS-G	20.84	18.94	17.79	16.42
PlaneOdometry + PlaneRCNN	30.33	27.12	24.48	21.33
SuperGlue-G + MWS	11.46	9.76	8.73	7.56
SuperGlue-G + MWS-G	28.80	26.33	24.81	23.01
SuperGlue-G + PlaneRCNN	41.84	37.58	34.31	30.06
RPNet + PlaneRCNN	41.64	37.04	33.71	29.44
Appearance Embedding only	47.83	42.17	38.06	33.04
Associative3D Optimization	47.51	41.93	38.01	33.01
No Continuous Optimization	52.15	46.07	41.47	35.87
Proposed	**52.75**	**46.45**	**41.76**	**36.02**

(a) Ours. (b) DeMoN. (c) PlaneOdometry.
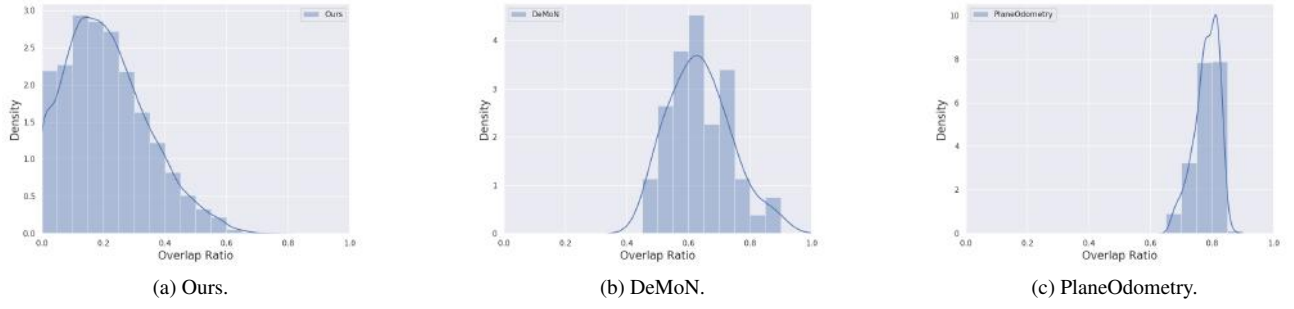
Figure 9: Histograms of overlap ratio for different datasets.

Dataset	Image 1	Image 2	Overlapping	Image 1	Image 2	Overlapping
Ours						
[55]						
[41]						

Figure 10: Overlapping regions of random examples of Our dataset (row 1-4), DeMoN [55] (row 5-6) and Plane Odometry [41] (row 7). We transform the point cloud of image 1 to image 2 and show it as a green mask.

Table 5: Ablation study of our camera pose module.

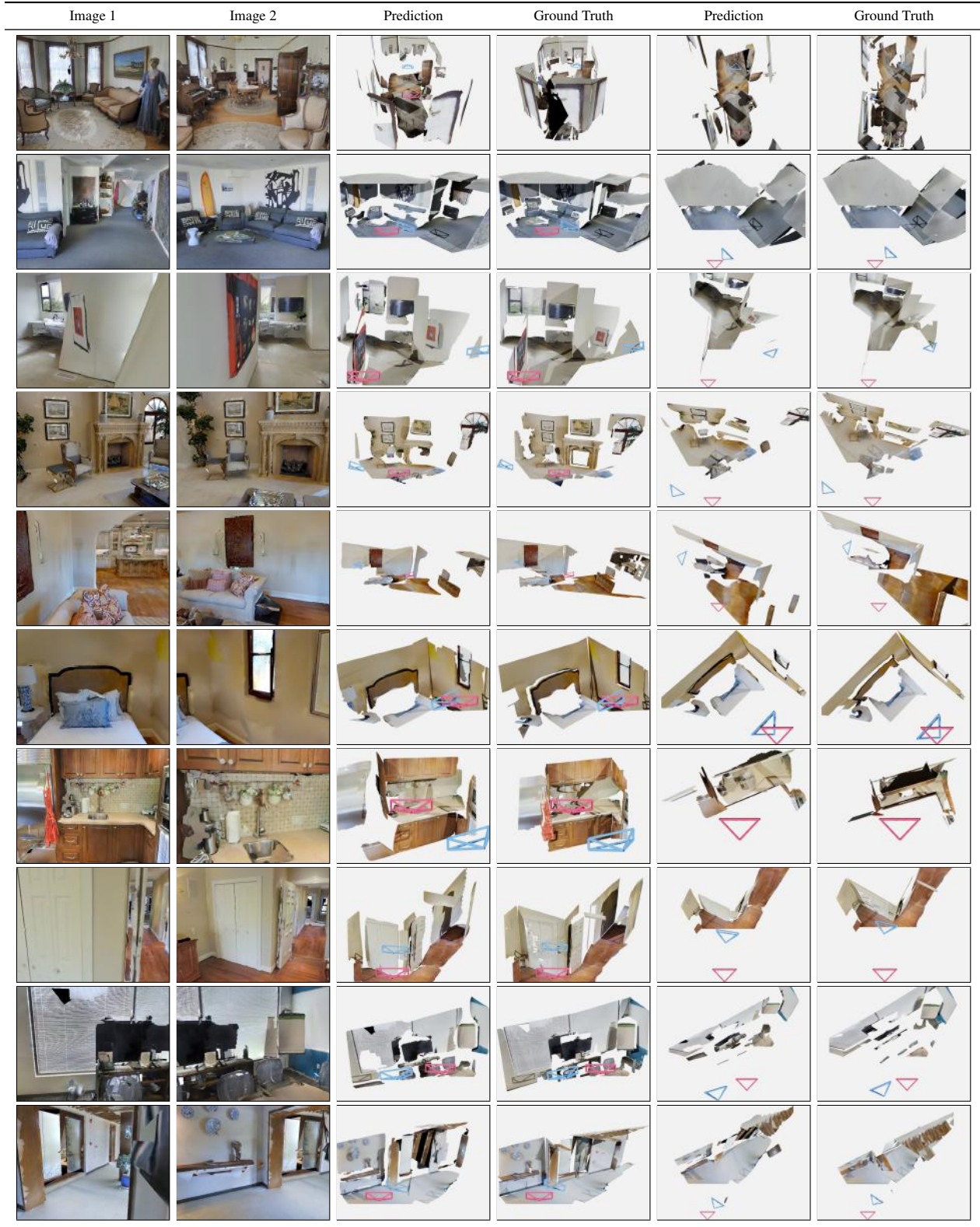| Method	Translation (meters)			Rotation (degrees)		
	Median ↓	Mean ↓	(Err ≤ 1m)% ↑	Median ↓	Mean ↓	(Err ≤ 30°)% ↑
Assoc.3D [37] cam.	2.17	2.46	14.8	42.09	52.97	38.1
ResNet50-CatConv	1.55	1.98	29.4	29.33	46.08	50.6
ResNet50-Attention	0.98	1.41	51.1	12.78	24.23	77.3
Proposed	**0.63**	**1.15**	**66.6**	**7.33**	**22.78**	**83.4**

A.4. Ablation Study for Camera Pose Module

We include detailed architectures for baselines in Section 4.4 of our paper in Table 8, 9 and 10. ReLU is used between all Linear and Conv layers. We use ResNet50 pretrained on COCO as the backbone to predict camera pose. We do not freeze the backbone for the baselines since they are standalone networks. Table 5 shows the ablation experiments. With the ability to explicitly calculate the relationship of features across views, our proposed attention module (ResNet50-Attention) outperforms other standalone architectures by a large margin on all the metrics. Running the attention module on the plane detection backbone further improves results, but due to switching to a FPN [26], it is difficult to directly ascribe performance changes.

A.5. Qualitative Results

We first present our reconstruction results on selected examples in Figure 11, extending Figure 3 in our paper. We show our prediction and ground truth from two novel views to see all planes in the whole scene – a slightly raised view and a top down view. We then present results automatically evenly spaced in the test set in Figure 12, according to single-view AP. As we quantitatively show in Section A.3, higher single-view AP means much better results. Therefore, we hope the evenly spaced results can represent the overall performance of our approach better. When single-view plane prediction is accurate, our reconstructions from sparse views are typically reasonable.

We also show more selected qualitative examples on plane correspondence prediction in Figure 13, extending Figure 5 in our paper. Those examples use *predicted* bounding boxes. To determine the ground truth correspondence, we assign each ground truth box with a predicted box whose mask IoU is greater than 0.5. We also randomly choose examples in Figure 14 to reflect the overall performance of our approach and baselines. Those examples use *ground truth* boxes which are the same as the ones used in correspondence evaluation. As shown in the random results, there are many false positives and false negatives in challenging cases; therefore, there is still much space to improve in predicting plane correspondence.

Image 1	Image 2	Prediction	Ground Truth	Prediction	Ground Truth

Figure 11: More results on Matterport3D test set, extending Figure 3 in our paper, **Blue** and **Red** frustums show cameras for image 1 and 2.

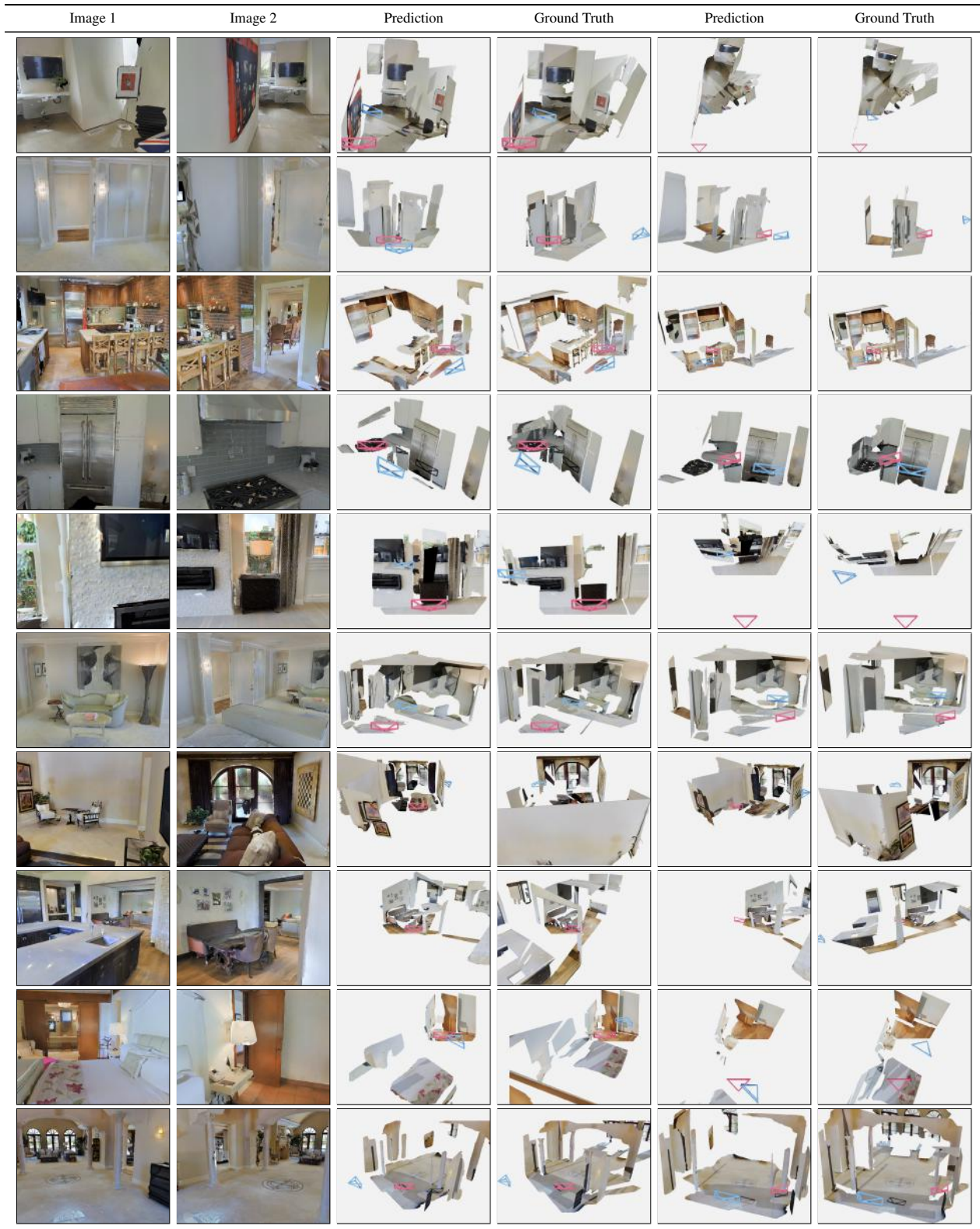| Image 1	Image 2	Prediction	Ground Truth	Prediction	Ground Truth

Figure 12: Random results on Matterport3D test set, automatically evenly spaced according to single-view AP. The first row has the highest single-view AP. **Blue** and **Red** frustums show cameras for image 1 and 2.
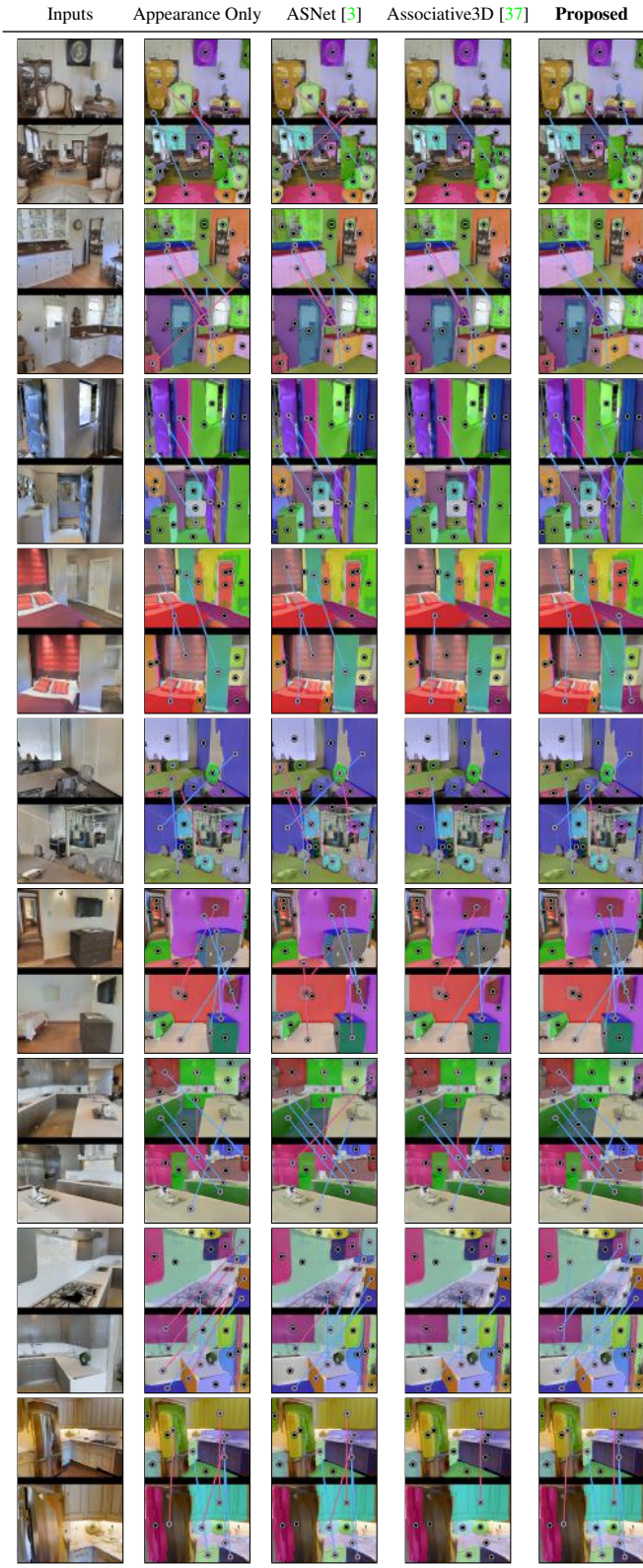
Figure 13: Selected correspondence predictions on *predicted* boxes, extending Figure 5 in our paper, showing true positive matches in **Blue** and false positives in **Red**. Correct matches are determined by assigning each ground truth box with a predicted box whose Mask IoU is greater than 0.5.
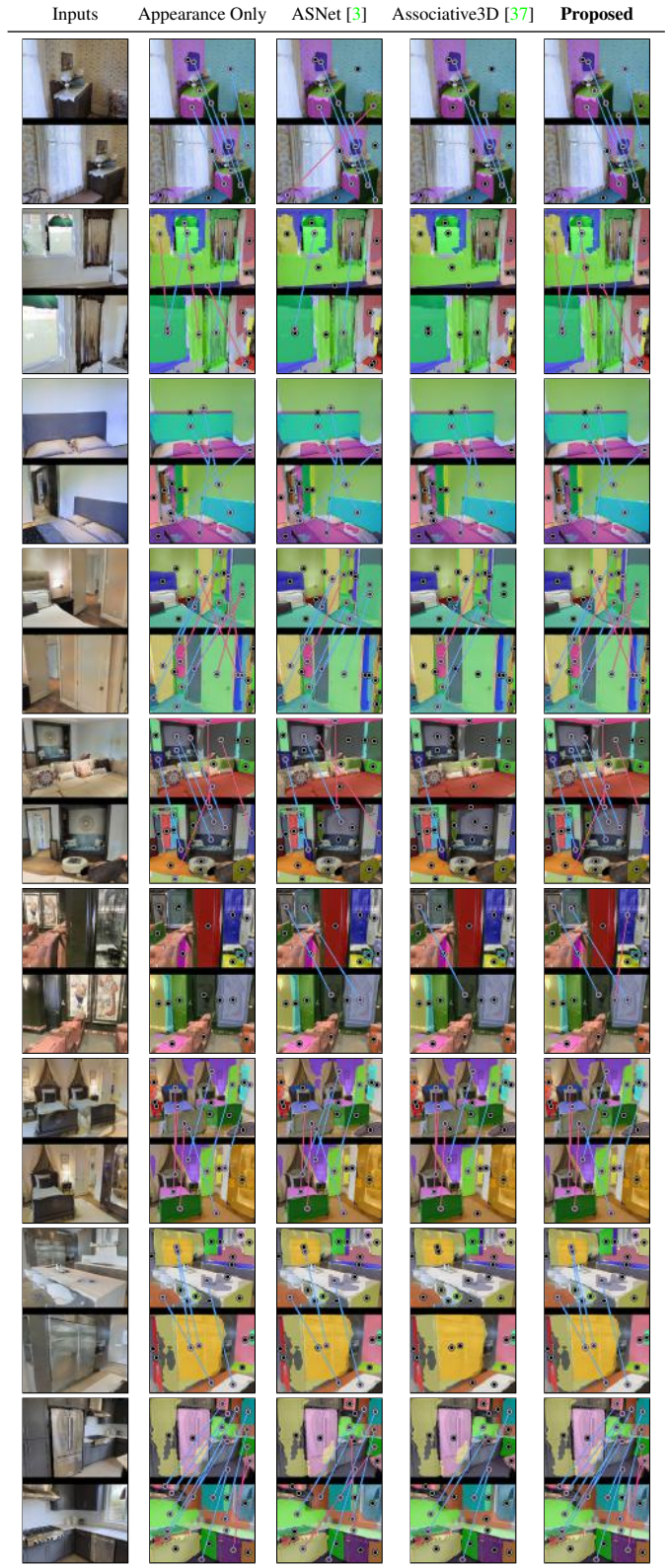
Figure 14: Random correspondence predictions on *ground truth* boxes, showing true positive matches in **Blue** and false positives in **Red**.

B. Implementation

B.1. Dataset Preprocessing

We augment Matterport3D [4] with plane segmentation annotations on the original mesh. This enables consistent plane instance ID and plane parameters through rendering, and automatically establishes plane correspondences across different images. We fit planes using RANSAC, following [29], on mesh vertices within the same object instance (using the annotation provided by the original Matterport3D dataset). We then render the per-pixel plane instance ID, along with the RGB-D images using AI Habitat [44]. Since the original instance segmentation mesh has "ghost" objects and holes due to artifacts, we further filter out bad plane annotations by comparing the depth information and the plane parameters. Figure 15 shows examples of plane annotations on the mesh and rendered plane segmentations. Small plane masks with area less than 1% of the total image pixels are removed.

We follow [68] to generate random camera poses. To include more data, we keep all valid cameras in each horizontal sector. The camera is of a random height 1.5-1.6m above the floor and a downward tilt angle of 11 degrees to simulate human's view. The same camera intrinsic are used to render all the images. To generate image pairs, we randomly sample cameras within each room, and enforce that there are at least three common planes and at least three unique planes in each image. Our ground truth data includes a depthmap and plane segmentation for each image, as well as plane correspondences and camera transformations for the image pair.

B.2. Network Architectures

We use Detectron2 [61] to implement our network. The overall architecture of our network is shown in Table 6. The backbone, RPN, box branch and mask branch are identical to Mask R-CNN [19]. The depth branch is the same as the depthmap decoder in [28]. Table 7 shows the exact architecture of the camera pose module in our model.

Training details. The plane detection backbone uses ResNet50-FPN pretrained on COCO [27]. We first train the plane prediction module without the plane embedding. Then we freeze the network and train the embedding head using the predicted bounding boxes. Finally, we freeze the whole network and train the camera pose module. We put in as large of a batch as possible to fit in the available GPUs. The number of iterations are picked based on evaluation results on the validation set. We use batch size 8, 16, 32 on 4, 4, and 2 GPUs for 37k, 36k and 35k iterations to train the plane prediction module, embedding head and camera pose module respectively. We used 2080Ti GPUs. We use SGD with momentum of 0.9, a weight decay of 0.0001, multistep learning rate with base $\mathrm{lr} = 0.001$, multiplied by 0.1 after

Figure 15: Example plane annotations of our dataset.

30k iterations, and warmup across all the training jobs.

B.3. Optimization

Associative3D [37] optimization. We re-build the heuristic optimization on the same network branches our approach uses to ensure apples-to-apples comparisons; we also optimize their search to accommodate for differences in plane-vs-object matching by using Eqn. 3 as the objective.

Discrete optimization. Our discrete optimization searches through all K camera options, ($K = 32 \times 32 = 1024$), and outputs the best camera $\{\mathbf{t}_{\hat{k}}, \mathbf{R}_{\hat{k}}\}$ as well as a binary correspondence matrix $\mathbf{C}$ for planes across images using the Hungarian algorithm, shown in Algorithm 1. cam2world converts plane parameters $[\mathbf{n}, o]$ from the camera frame to the world frame,

$$
\begin{aligned}
\mathbf{P} &= \left(1 + \frac{\mathbf{t}_k \cdot \mathbf{R}_k(o \cdot \mathbf{n})}{||\mathbf{R}_k(o \cdot \mathbf{n})||^2}\right) \mathbf{R}_k(o \cdot \mathbf{n}), \\
\mathbf{n}^w &= \mathbf{P}/||\mathbf{P}||, \\
o^w &= ||\mathbf{P}||.
\end{aligned}
\tag{5}
$$

In practice, in Eqn. 3 of the paper, we normalize the offset error and normal error. Since the offset error is unbounded, we clamp it by O_{clamp}. We use threshold after Hungarian algorithm to reject matches with large distance error. The hyperparameters are determined using random search on the validation set: threshold $= 0.7$, $\lambda_e = 0.47$, $\lambda_n = 0.25$, $\lambda_o = 0.28$, $O_{\mathrm{clamp}} = 4$, $\lambda_N = 0.311$, $\lambda_h = 0.432$, $\lambda_t = 0.166$, $\lambda_R = 0.092$.

Continuous optimization. Continuous optimization further outputs refined camera $\{\hat{\mathbf{t}}, \hat{\mathbf{R}}\}$ and plane parameters $\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\pi}}'$ based on the discrete optimization results by using non-linear optimization to minimize both the geometric distance between the corresponding planes and the pixel alignment error based on pixel-level feature. We use scipy.optimize.least_squares with default settings for the optimization. For the pixel alignment error, we

Algorithm 1 Discrete optimization. It takes as input the plane embeddings and parameters as well as the camera poses and probabilities, it outputs the best binary correspondence $\mathbf{C}$ and the best camera pose $\{\mathbf{t}_{\hat{k}}, \mathbf{R}_{\hat{k}}\}$. cam2world converts plane parameters from the camera frame to the world frame as described in Eqn. 5. $\mathbf{S}_k$ is the same term in Eqn. 2 of the main paper, terms in view 2 are denoted as $\mathbf{e}', [\mathbf{n}', o']$.

Require: $\mathbf{e}, \mathbf{e}', \mathbf{n}, \mathbf{n}', o, o', \{p_{\mathbf{t}_k}, p_{\mathbf{R}_k}, \mathbf{t}_k, \mathbf{R}_k\}_{k=0,\cdots,K}$
Ensure: $\mathbf{C}, \{\mathbf{t}_{\hat{k}}, \mathbf{R}_{\hat{k}}\}$
  1: min_cost $\leftarrow$ Inf
  2: $\hat{k} \leftarrow -1$
  3: $\Delta\mathbf{e}_{ij} = ||\mathbf{e}_i - \mathbf{e}'_j||$
  4: **for** $k \in \{1, \cdots, K\}$ **do**
  5:     $\mathbf{n}^w, o^w \leftarrow$ cam2world$(\mathbf{n}, o, \mathbf{t}_k, \mathbf{R}_k)$
  6:     $\mathbf{n}'^w, o'^w \leftarrow$ cam2world$(\mathbf{n}', o', \mathbf{0}, \mathbf{I}(3))$
  7:     $\Delta\mathbf{n}_{ij} \leftarrow \mathrm{acos}(|\mathbf{n}_i^{w\mathsf{T}}\mathbf{n}'^w_j|)/\pi$
  8:     $\Delta\mathbf{o}_{ij} \leftarrow \min(|o_i^w - o'^w_j|/O_{\mathrm{clamp}}, 1)$
  9:     $\mathbf{S}_k \leftarrow \lambda_e\Delta\mathbf{e} + \lambda_n\Delta\mathbf{n} + \lambda_o\Delta\mathbf{o}$
 10:     $\mathbf{C}_k \leftarrow$ Hungarian$(\mathbf{S}_k)$
 11:     $\mathbf{C}_k(\mathbf{S}_k < \text{threshold}) = 0$
 12:     cost $\leftarrow \lambda_h \sum_{i,j}(\mathbf{C}_k \circ \mathbf{S}_k)_{i,j} - \lambda_t\log(p_{\mathbf{t}_k}) - \lambda_R\log(p_{\mathbf{R}_k}) - \lambda_N \sum_{i,j}\mathbf{C}_{k,i,j}$
 13:     **if** cost $<$ min_cost **then**
 14:         min_cost $\leftarrow$ cost
 15:         $\hat{k} = k$
 16:     **end if**
 17: **end for**
 18: **return** $\mathbf{C}_{\hat{k}}, \{\mathbf{t}_{\hat{k}}, \mathbf{R}_{\hat{k}}\}$

---

measure the distance between back-projected corresponding pixels on the corresponding planes. We first warp the texture of each plane segment to viewpoint normalized camera frame using the predicted plane parameters. We then extract SIFT [30] features and match keypoints between corresponding planes. Incorrect matches are filtered out using RANSAC with an affine transformation. Theoretically the predicted keypoints of the two plane segments in a correctly normalized camera frame should satisfy a stronger relationship (2D rotation+translation) if plane parameters are accurate. However, since our plane parameters are not perfect, we relax to an affine transformation. Finally, the corresponding pixels are back-projected to 3D using the camera intrinsics and extrinsics to calculate point-wise Euclidean distance. We use Eqn. 4 in our paper as the objective. We set d_{cam} to be the geodesic rotation distance between $\hat{\mathbf{R}}$ and $\mathbf{R}_{\hat{k}}$ to regulate the deviation from the selected camera bin.

B.4. ASNet [3] Baseline.

Following the Appearance-Surrounding Network in [3], in addition to the appearance branch, we add a surrounding branch for feature extraction to augment our embedding head. We combine both appearance features and surrounding features to determine plane correspondences. Instead of cropping the original image which causes one inference per detection, we increase the receptive field of RoIAlign by a factor of 2 to extract larger ROI features. After RoIAlign,

the features are cropped and passed to a surrounding extractor and an appearance extractor. Both the extractors use the same number and size of Conv layers and linear layers as our embedding head uses (4 Conv layers and 1 FC layer). We train using cosine similarity weighted loss as described in Equation 1 and 2 of [3]. We ensure apples-to-apples comparisons by re-training the embedding head on the same freezed backbone our approach uses.

Table 6: Overall architecture for our proposed network. The backbone, RPN, box, and mask branches are identical to Mask R-CNN. The RPN predicts a bounding box for each of A anchors in the input feature map. C is the number of categories (here = 1 because we only have one "plane" class). TConv is a transpose convolution with stride 2. ReLU is used between all Linear, Conv and TConv operations. Outputs of the normal branch and the embedding branch are normalized. Depth branch uses Conv and Deconv layers to generate a depthmap with the same resolution as the input image. The camera pose module is detailed in Table 7, it takes features from both images and outputs 32 logits for translation and rotation bins respectively.

Index	Inputs	Operation	Output shape
(1)	Inputs	Input Image	$H \times W \times 3$
(2)	(1)	Backbone: ResNet50-FPN	$h \times w \times 256$
(3)	(2)	RPN	$h \times w \times A \times 4$
(4)	(2),(3)	`RoIAlign`	$14 \times 14 \times 256$
(5)	(4)	Box: 2×downsample, Flatten, Linear($7 \times 7 \times 256 \rightarrow 1024$), Linear($1024 \rightarrow 5C$)	$C \times 5$
(6)	(4)	Mask: $4 \times$ Conv($256 \rightarrow 256, 3 \times 3$), TConv($256 \rightarrow 256, 2 \times 2, 2$), Conv($256 \rightarrow C, 1 \times 1$)	$28 \times 28 \times C$
(7)	(4)	Normal: $4 \times$ Conv($256 \rightarrow 256, 3 \times 3$), Linear($14 \times 14 \times 256 \rightarrow 1024$), Linear($1024 \rightarrow 3$)	$C \times 3$
(8)	(4)	Embedding: $4 \times$ Conv($256 \rightarrow 256, 3 \times 3$), Linear($14 \times 14 \times 256 \rightarrow 1024$), Linear($1024 \rightarrow 128$)	$C \times 128$
(9)	(2)	Depth	$H \times W \times 1$
(10)	(2), (2)'	Camera pose module	$1 \times 32 + 1 \times 32$

Table 7: The architecture for **Proposed** camera pose module. It takes as input the P_3 features of the ResNet50-FPN backbone from both images ((2) in Table 6). Then six Conv layers are used to learn an appropriate image features for matching ((2)-(4)). `Maxpool` is used to reduce the feature size. We then compute the attention of the image features and reshape the result to $300 \times 15 \times 20$. After six Conv layers (stride alternates between 1 and 2 to reduce feature size) and two Linear layers, we predict two multinomial distributions over 32 bins for translation and rotation respectively. ReLU is used between all Linear, Conv layers.

Index	Inputs	Operation	Output shape
(1)	Input	P_3 features from ResNet50-FPN	$2 \times 60 \times 80 \times 256$
(2)	(1)	$2 \times$ Conv($256 \rightarrow 256, 3 \times 3$), Maxpool	$2 \times 30 \times 40 \times 256$
(3)	(2)	$2 \times$ Conv($256 \rightarrow 256, 3 \times 3$), Maxpool	$2 \times 15 \times 20 \times 256$
(4)	(3)	Conv($256 \rightarrow 256, 3 \times 3$), Conv($256 \rightarrow 512, 3 \times 3$)	$2 \times 15 \times 20 \times 512$
(5)	(4)	Attention	$300 \times 15 \times 20$
(6)	(5)	Conv($300 \rightarrow 128, 3 \times 3$), $5 \times$ Conv($128 \rightarrow 128, 3 \times 3$),	$128 \times 2 \times 3$
(7)	(6)	Linear($128 \times 2 \times 3 \rightarrow 64$)	1×64
(8)	(7)	Translation: Linear($64 \rightarrow 32$)	1×32
(9)	(7)	Rotation: Linear($64 \rightarrow 32$)	1×32

Table 8: The architecture for benchmark *Associative3D* [37] *camera branch* in Section 4.4 of our paper. Note this is a larger network compared to the original paper because we add as many nonlinearities as our architecture. It takes as input the features of the ResNet50 backbone from both images, then average pools the features and passes through eight Linear layers to predict two multinomial distributions over 32 bins for translation and rotation respectively.

Index	Inputs	Operation	Output shape
(1)	Input	Image features	$2 \times 15 \times 20 \times 2048$
(2)	(1)	Avgpool	$2 \times 1 \times 1 \times 2048$
(3)	(2)	Concat	$1 \times 1 \times 4096$
(4)	(3)	Linear($4096 \rightarrow 512$), $5 \times$ Linear($512 \rightarrow 512$), Linear($512 \rightarrow 64$)	1×64
(5)	(4)	Translation: Linear($64 \rightarrow 32$)	1×32
(6)	(4)	Rotation: Linear($64 \rightarrow 32$)	1×32

Table 9: The architecture for benchmark *ResNet50-CatConv* camera branch in Section 4.4 of our paper. It takes as input the features of the ResNet50 backbone from both images, concatenates the image features, and then uses six Conv layers (stride alternates between 1 and 2 to reduce feature size) and two Linear layers to predict two multinomial distributions over 32 bins for translation and rotation respectively.

Index	Inputs	Operation	Output shape
(1)	Input	Image features	$2 \times 15 \times 20 \times 2048$
(2)	(1)	Concat	$15 \times 20 \times 4096$
(3)	(2)	Conv($4096 \rightarrow 128, 3 \times 3$), $5 \times$ Conv($128 \rightarrow 128, 3 \times 3$),	$128 \times 2 \times 3$
(4)	(3)	Linear($128 \times 2 \times 3 \rightarrow 64$)	1×64
(5)	(4)	Translation: Linear($64 \rightarrow 32$)	1×32
(6)	(4)	Rotation: Linear($64 \rightarrow 32$)	1×32

Table 10: The architecture for benchmark *ResNet50-Attention* camera branch in Section 4.4 of our paper. It takes as input the features of the ResNet50 backbone from both images, computes the attention of the image features and reshapes the result to $300 \times 15 \times 20$. After six Conv layers (stride alternates between 1 and 2 to reduce feature size) and two Linear layers, it predicts two multinomial distributions over 32 bins for translation and rotation respectively.

Index	Inputs	Operation	Output shape
(1)	Input	Image features	$2 \times 15 \times 20 \times 2048$
(2)	(1)	Attention	$300 \times 15 \times 20$
(3)	(2)	Conv($300 \rightarrow 128, 3 \times 3$), $5 \times$ Conv($128 \rightarrow 128, 3 \times 3$),	$128 \times 2 \times 3$
(4)	(3)	Linear($128 \times 2 \times 3 \rightarrow 64$)	1×64
(5)	(4)	Translation: Linear($64 \rightarrow 32$)	1×32
(6)	(4)	Rotation: Linear($64 \rightarrow 32$)	1×32