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

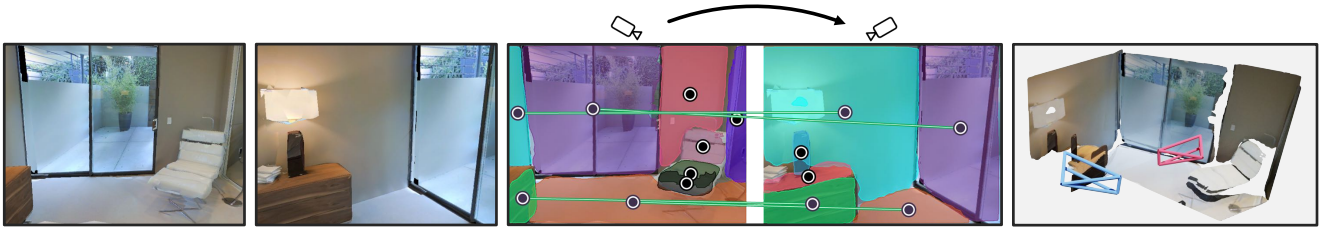

Figure 1. Given two RGB images with an unknown relationship, our system produces a coherent planar surface reconstruction of the scene in terms of 3D planes and relative camera pose. We show this reconstruction with the inferred left and right cameras in **Blue** and **Red**.

Abstract

The paper studies planar surface reconstruction of indoor scenes from two views with unknown camera pose. 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

Humans have a remarkable ability to infer the 3D structure of a scene from only a few images even when these photos are taken from drastically different views, and use this ability for tasks ranging from understanding photos of an event to looking for apartments. Our goal is to create a vision system that has similar capabilities. Such a system should be able to take perspective images as input, without *a priori* knowledge of camera pose or depth information, and produce a single, coherent reconstruction of a scene. This problem is challenging because it requires the system to both predict 3D information from each 2D image and estimate geometric relationships between views.

One might hope that this problem could be solved in a purely geometric fashion via methods such as structure from motion [18, 7, 2] or by simply predicting a per-view reconstruction [55, 28], but there are several challenges. Classic methods fail from the start since planes are often only visible in one view, and the large viewpoint change makes correspondence hard; naïve fusing single view predictions is difficult since the predictions are made independently. While there is increasing interest in multiview deep learning, they require known camera poses [23, 31], many views [21], synthetic data [35], or additional depth information [57, 54], or do not reconstruct [11, 6].

We address this problem by estimating and fusing the reconstructions while taking 3D geometry into consideration. In the process, we jointly tackle three core problems: correspondence with repetitive objects and weak texture; inferring relative camera pose; and reconstructing a scene from as little as one view. Our method and experimental results suggest that jointly solving these problems is important and that the overall solution is more than the sum of its parts.

We use plane segments as our base representation for reconstruction and correspondence, since they have several advantages: they have a simple parameterization as a mask and plane parameter vector, can often approximate scenes well, and there is a rich line of work in extracting them (or similar information) from ordinary 2D images [29, 28, 12, 50, 10]. They also aid us in obtaining correspondences, since plane geometry can give cues for correspondence [52, 13] even with limited-to-no overlap.

We propose an architecture and an associated optimization approach that jointly estimates planar segments and parameters, correspondence, and camera transformations from a pair of RGB images. Based on prior work PlaneRCNN [28], we detect planes from a pair of images, embeddings for correspondence, and a distribution over possible camera poses in a single forward pass. This information is then used in a discrete and continuous optimization problem

that jointly reasons over these predictions to predict a single, coherent, texture-aligned reconstruction across views. We evaluate this approach on realistic renderings from the Matterport3D [4] dataset, which consists of reality captures. Our results show that reasoning to solve a joint problem is important, and that jointly solving reconstruction and correspondence is important for both problems.

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 be able 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 [50], voxels [16], and depth [10] from 2D images. Our approach builds most heavily on a recent line of work aiming to produce a planar reconstruction or variants thereof [29, 55, 28, 58, 5, 22]. We build upon the work in this area, in particular PlaneRCNN [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 [62, 61, 45, 56].

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 [38, 9, 51], which are often paired with projective geometry [18]; other work finds matches across object-level correspondence [3]. 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 [52] 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 [35], aligning RGBD data [57], or learning to predict the fundamental matrix [36, 33]. 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 [40] and more similar to wide-baseline stereo [34]. 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 [42, 14, 15, 17].

The most similar work in this direction is Associative3D [35], which solves a related problem of reconstructing volumetric objects. Our approach is inspired by [35], 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 SUNCG dataset [43], 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 take as input a pair of images with an unknown relationship and produce a reconstruction of the scene in the form of a set of globally consistent planes and relative camera pose. Our first stage is trained to predict (a) for each image, a set of hypothetical planes (composed of a segment, plane parameters, and an appearance embedding vector), described in Section 3.1; and (b) a distribution over the relative camera pose between images, described in Section 3.2. These are combined to produce a coherent 3D interpretation by an optimization over camera location, plane correspondence, and plane parameters, described in Section 3.3. At training time, we assume access to ordinary RGBD images with relative camera poses, but at test time do not require depth.

3.1. Plane Prediction Module

The goal of our plane prediction module is to produce, per-image, a set of plane segments in the camera view that also have an embedding that enables cross-view matching. Each plane consists of: a segment $\mathcal{M}_i$, giving the extent of the plane; plane parameters $\boldsymbol{\pi}_i = [\mathbf{n}_i, o_i]$ (where $\mathbf{n}_i$ is a unit vector normal and o_i the offset) describing the plane

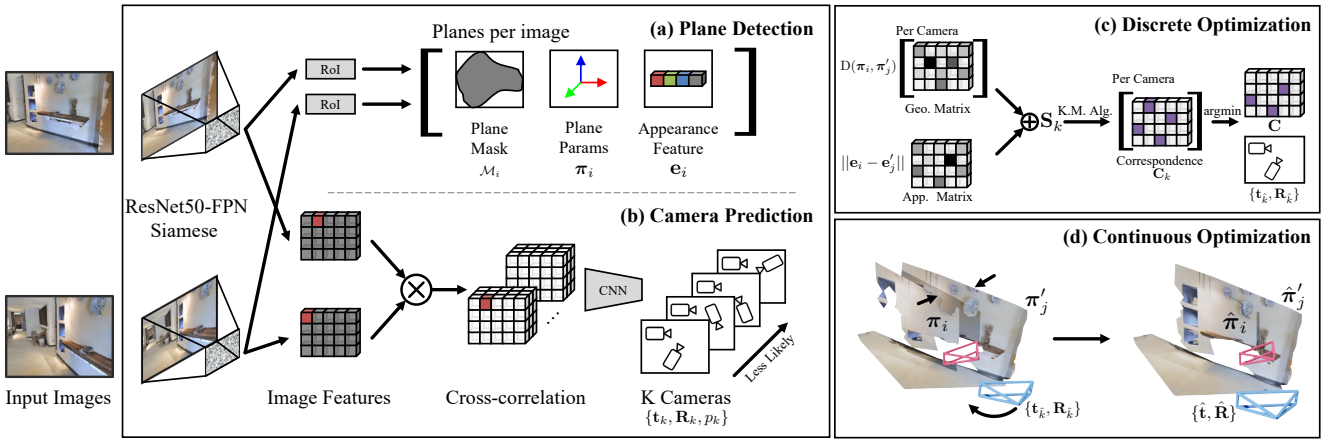

Figure 2. **Our Approach.** Given a pair of images, we use a ResNet50-FPN to detect planes and predict probabilities of relative camera poses, and use a two-step optimization to generate a coherent planar reconstruction. (a) For each plane, we predict a segmentation mask, plane parameters and an appearance feature. (b) Concurrently, we pass image features from the detection backbone through the cross-correlation 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 continuously optimize the camera and plane parameters.

equation $\boldsymbol{\pi}_i^T[x, y, z, -1] = 0$; and a unit-sphere embedding $\mathbf{e}_i$ ($||\mathbf{e}_i|| = 1$) for cross-view matching. Throughout, we will denote planes in view 2 as $\mathcal{M}'_j, \boldsymbol{\pi}'_j, \mathbf{e}'_j$, etc.

Plane detection backbone. We cast the problem as detection and extend the module from PlaneRCNN [28] to extract planar pieces from each image. During inference, the system produces a set of backbone features using ResNet50-FPN [26]. We use a region proposal network to propose boxes and use RoIAlign [19] to extract plane mask and normal. At the same time, a decoder maps the backbone feature to a depthmap, which is used to compute offset for each plane. We refer readers to [28] for more details.

Learning appearance feature. In addition to the components in [28], we learn a cross-view embedding by training the network on *pairs* of images via triplet loss. We assume knowledge of plane correspondence and we can 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 [41] $\max(||\mathbf{e}_a - \mathbf{e}'_n||_2 - ||\mathbf{e}_a - \mathbf{e}'_p||_2 + \alpha, 0)$, 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 model aims to estimate the relative camera pose between the two views. This problem is difficult given the relative viewpoints that appear in our data, and so the goal of this stage is to provide an uncertainty distribution over possible views, rather than solving the camera pose problem in one go. We thus cast the problem as a prediction over the a set of discrete bins for translation and rotation. The output of the model is a distribution over a fixed codebook of translation and rotation pairs $\{\mathbf{t}_k, \mathbf{R}_k, p_k\}$ (i.e., $p_k > 0, \sum_k p_k = 1$).

Our network outputs two multinomial distributions, one over translation and the other over rotation using cross-correlational features. We treat these distributions independently, so the joint translation/rotation distribution is then their cartesian product. Our cross-correlational features follow recent literature that find similarity between images using attention [51, 49, 37] and capture, at each pixel in a feature map, the relative similarity between that pixel and the pixels in the other image. Specifically, we use a 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 $(h_2 w_2) \times (h_1 w_1)$ cross-correlation feature A:

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

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 cross-correlation to be superior compared to the strategy of concatenating pre-pooled vector outputs in [35] 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], with six convolutional layers on top to learn an appropriate feature for matching.

3.3. Optimization

Our goal of obtaining a single reconstruction makes optimization necessary and challenging: choosing the most

likely camera and accepting each image's planes does not work due to duplicate planes and imperfect measurement. While duplicate planes would pose no issue with perfect parameters, our ability to estimate plane parameters is limited, leading to multiple planes in the same rough location.

We therefore optimize over the final set of planes given the evidence presented by the deep networks to produce a single coherent reconstruction of the scene that is consistent (*e.g.*, planes in correspondence are in the same location). We cast this as an optimization problem over a final 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 $\mathbf{C} \in \{0,1\}^{m \times n}$. Each entry in the correspondence matrix $\mathbf{C}_{i,j}$ is 1 if and only if plane i corresponds to plane j. The sums over rows and columns are at most 1, and are zero if the plane has no match. As input, the optimization has a series of n planes for the view one $\{\mathcal{M}_i, \boldsymbol{\pi}_i, \mathbf{e}_i\}$ and m planes for view two $\{\mathcal{M}'_j, \boldsymbol{\pi}'_j, \mathbf{e}'_j\}$, and a distribution over relative camera transformations $\{\mathbf{t}_k, \mathbf{R}_k, p_k\}$.

The optimization variables are both discrete and continuous and are tightly connected (*e.g.*, optimizing plane parameters depends on the correspondence). Moreover, due to our use of two *unknown views*, there are many degenerate solutions. We therefore solve the problem with a two-stage approach: we first hold plane parameters fixed, and solve for plane correspondence and rough camera location; we use the initialization and then solve the continuous problem to produce the final plane and camera parameters.

Discrete Problem: We first solve a discrete minimization 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 $\mathbf{S}_k$ that encodes 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 $\mathbf{S}_k$ is

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

which includes terms for the appearance embedding, normal similarity, and offset distance. All have associated tradeoff parameters λ. The offsets and normals incur zero penalty if the predictions are perfect.

This cost term is combined with two other regularizing terms. One term, $-\log(p_k)$, 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 (while incorporating the camera).

With a fixed camera k, the objective $\sum_{i,j} (\mathbf{C} \circ \mathbf{S}_k)_{i,j}$ can be efficiently solved for 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 the camera $\hat{k}$ and planar correspondences $\mathbf{C}$, we can then continuously refine the predictions from the deep networks for the camera and plane. We optimize over camera transformations $\hat{\mathbf{R}}, \hat{\mathbf{t}}$ and plane parameters $\hat{\boldsymbol{\pi}}_i, \hat{\boldsymbol{\pi}}'_j$ to minimize both geometric distance between the corresponding planes (assuming the same coordinate frame) and pixel alignment error based on pixel-level feature:

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

where d_{pixel} measures the Euclidean distance of back-projected points that match across corresponding planes. Inspired by [52], 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 solve it with the trust region reflective minimizer from Scipy with default options . We parameterize rotations as 6D vectors following [60].

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 [53] 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 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 tradeoff 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 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

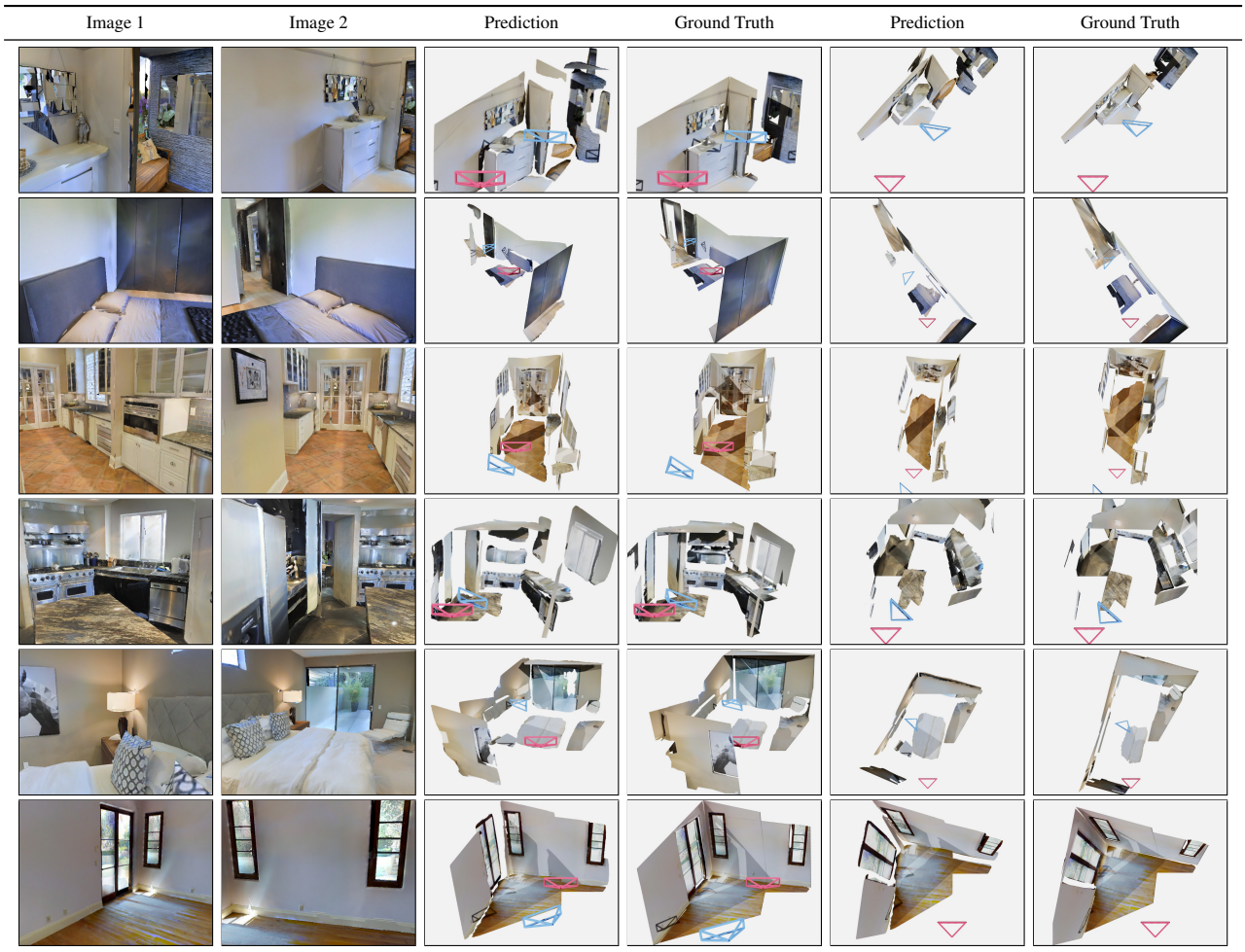

Figure 3. Results on Matterport3D test set. **Blue** and **Red** frustums show cameras for image 1 and 2. More results are in the supplemental.

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 transformation (Section 4.4). For each task, we compare with methods that use the strategy taken by Associative3D [35], as well ablations of our model.

4.1. Experimental Setup

We use re-renderings of real scenes using the AI Habitat simulator [39] and Matterport3D dataset [4]. This simulator enables the rendering of realistic images from arbitrary viewpoints, and provides us with ground truth pose and depth that can be used for evaluation.

Dataset: Our training, validation, and test set consist of 31932, 4707, and 7996 pairs of images respectively. We fit our ground truth planes on Matterport3D using a standard RANSAC approach. 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 follow the camera sampling strategy in [59] to generate single images. We then pick pairs that have at least three planes in common and at least three unique planes. These are generated from randomly sampled pairs among single images, following a similar strategy as [35], which aims to generate pairs that are challenging but feasible. These planes and their correspondences are used to train the plane prediction module. We obtain the camera pose supervision from the simulator. All parameters are tuned on the validation set.

Representative examples of pairs can be seen throughout the paper, but to give quantitative context for the dataset, we computed some statistics on 200 random image pairs from the dataset. We transform the point cloud of view 1 to view 2. The average percentage of overlapping points was 21%. For context, we did the same for the Sun3D [54] image pairs used by DeMoN [48]: these had an average overlap of 64%.

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 and in this paper we use the second camera frame.

Table 1. We evaluate performance by treating the reconstruction as 3D plane detection and report the average precision. **All** means a prediction is a true positive only if Mask IoU ≥ 0.5, Normal error $\leq 30^\circ$ and Offset error ≤ 1m. **(Normal + Mask)** and **(Offset + Mask)** mean a prediction is a true positive when satisfying both thresholds.

Methods	All	Normal + Mask	Offset + Mask
Top 1 camera, no merge	29.44	35.25	31.67
Top 1 camera, appearance only	33.04	39.78	36.85
Associative3D [35] Optimization	33.01	39.43	35.76
Discrete Optimization, Appearance + Geometry	35.87	**42.13**	38.80
Proposed	**36.02**	42.01	**39.04**

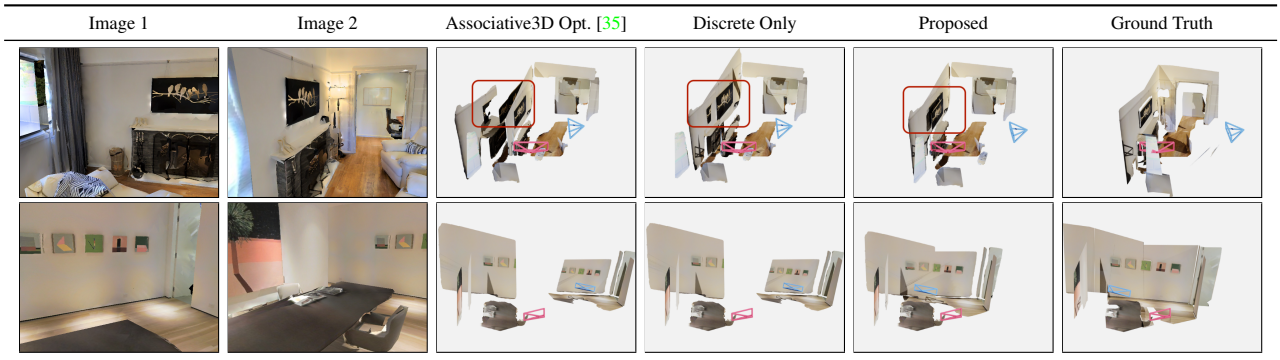

Figure 4. Comparison between Associative3D Opt., Discrete Only and Proposed approach. **Row 1:** As shown in **red** boxes, Associative3D Opt. fails to predict the plane correspondence, while our discrete optimization manages to merge it. **Row 2:** Our continuous optimization adjusts cameras and merges planes based on Discrete results.

Baselines: We compare the full system against four baselines that test contributions of our system.

Top 1 Camera, no merge: This system combines the two plane reconstructions with the most likely camera and is the simplest approach predicting components independently.

Top 1 Camera, Appearance Only: This system solves for the best correspondence between planes using the optimization procedure, but uses only appearance embedding $\mathbf{e}_i$ predicted alongside each plane. This approach does not reason about the set of possible cameras: without the geometric term in the cost matrix, the only evidence about which camera best explains the scene is from the camera pose module.

Associative3D Optimization: The most closely related paper [35] solves a similar problem using a RANSAC-style method due to speed considerations. They use an appearance-based object embedding vector to construct an affinity matrix to generate many correspondence proposals and use a scoring function that incorporates geometry to rank the proposals. We apply their technique to our objective. We use the embedding vector for each plane to construct the affinity matrix. Then we randomly generate 128 matching proposals for each possible camera to search for the best correspondence and camera. For a fair comparison, we use our camera pose module and scoring function Eqn. 3. This baseline tests the contribution of our optimization.

Discrete Only, Appearance+Geometry: This incorporates the appearance, normal, and offset terms in the cost matrix, enabling joint reasoning over plane and cameras. It does not include the continuous optimization described in Section 3.3. This tests the contribution of jointly reasoning over planes and camera compared to just appearance features.

Proposed: This system is all components of the proposed method. Compared to the method above, this tests the contribution of the continuous optimization.

Qualitative Results: We show qualitative results in Figure 3, as well as the comparison between our proposed approach and baselines in Figure 4. *Associative3D Optimization* can usually obtain a reasonable camera transformation using our improved camera pose module but usually does not find good plane correspondence across views. The first row shows Associative3D fails to merge the walls and pictures across images. *Discrete Only* improves the plane correspondence using the Hungarian algorithm instead of randomly searching. 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 closer to the ground truth.

Metrics: We reconstruct the scene in terms of a set of plane segments that are placed into the scene and therefore evaluate our approach following other approaches that reconstruct the scene factored into components [46, 24, 25, 32, 35]. In particular, we treat the full problem like a detection problem, and use average precision (AP). Given a ground-

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 [35] Opt.	0.9	10.7	44.0
Proposed	**16.2**	**28.1**	55.3

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*) evaluates the surface normal distance, $\text{acos}(|\mathbf{x}_1^\intercal \mathbf{x}_2|) \leq 30°$; and (iii) (*Offset*) measures the offset distance, $|o_1 - o_2| \leq 1$.

Quantitative Results: We show results in Table 1. Compared to all four baseline methods, our approach obtains higher overall AP. Using the top-1 (i.e., most likely) camera without merging performs the worst (29.4 AP), because planes visible in both views are duplicated. Merging planes based on appearance features improves results, yielding an AP of 33.0. Incorporating geometric features (*Discrete*) improves AP by another 2.8 points. While the Associative3D optimization also uses geometric features, its AP is comparable to merging planes based on appearance features. This is because Associative3D's random search of the space generates suboptimal correspondence, as also shown in Section 4.3. Finally, continuous optimization improves results further, primarily by improving the camera estimations. The improvement is detailed in Section 4.4.

4.3. Plane Correspondence

While we indirectly evaluate correspondence via duplicates in AP in the prior section, we analyze it independently.

Baselines: We compare the proposed approach with three 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: The simplest approach is to use only the appearance embedding and no geometry. We apply the Hungarian algorithm directly to the appearance features.

MessyTable [3] ASNet on ROI: An alternative to using geometry in the reasoning is to simply incorporate more context information by applying another network that sees a larger context. We follow the approach of ASNet in [3] and add a second network that looks at the surroundings of a box, besides the first network that looks at a box. The ASNet approach was originally applied on image crops, leading to one backbone forward pass per box. To make the method comparable, we crop ROI features and add convolutional layers to extract appearance and surrounding features.

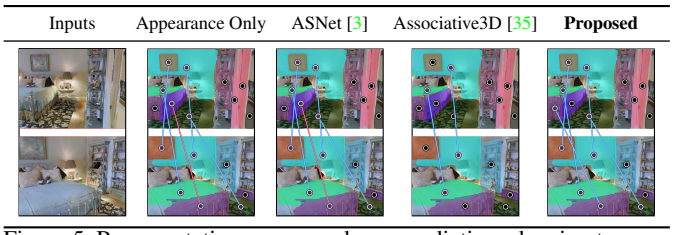

Figure 5. Representative correspondence prediction, showing true positive matches in **Blue** and false positives in **Red**.

For a fair comparison, we use our plane detection backbone and the same number of convolutional layers for the appearance and surrounding features as our method uses.

Associative3D [35] Optimization: We again apply the optimization method of [35], again using appearance to generate proposals and the full score to select correspondence.

Proposed: This is 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. We found metrics like AUROC to be misleading because they measure a correct ordering both within and *across* image pairs. This is naturally satisfied when one predicts an embedding on a hypersphere, but not easily so when there are terms measuring scene distance (e.g., Eqn. 2). We found our appearance-only features to have slightly better performance on AP/AUROC compared to [3] but do not report it because it does not only measure *within-pair* distance.

Quantitative Results: The results are shown in Table 2. Our method outperforms all other benchmarks on IPAA-100 and IPAA-90 and has a slight drop of 0.4% in IPAA-80. Our continuous optimization improves IPAA-100 by a large margin (9.4%) compared to using the appearance embedding matrix only. Although Associative3D's optimization incorporates geometric information, our method outperforms it. We hypothesize its heuristic optimization works well with the small search space in the original object-centric setting, but the far larger number of planes compared to objects causes issues due to the factorial grown in search space. Finally, the ASNet approach does not improve results in our setting. We believe this is because ASNet was designed for tabletop object matching where the surroundings of the same instance will be similar in another view due to the scene being roughly planar. Our dataset is highly non-planar and so the surroundings of an object may be very different if viewed from other angles.

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

Table 3. Evaluation of relative camera pose. Rows 1-3 analyze architectures; Rows 4-6 analyze increasing amounts of optimization.

Method	Translation (meters)			Rotation (degrees)		
	Median ↓	Mean ↓	(Err ≤ 1m)% ↑	Median ↓	Mean ↓	(Err ≤ 30°)% ↑
Associative3D cam.	2.17	2.46	14.77	42.09	52.97	38.09
ResNet50-CatConv	1.55	1.98	29.40	29.33	46.08	50.64
ResNet50-CrossCorr	0.98	1.41	51.06	12.78	24.23	77.30
Ours-CrossCorr	0.90	1.40	55.45	7.65	24.57	81.88
Ours-Discrete only	0.88	1.36	56.50	7.58	22.84	**83.69**
Proposed	**0.63**	**1.15**	**66.57**	**7.33**	**22.78**	83.39

the correspondence due to its random search scheme. Our method detects all the correspondence and distinguishes similar texture planes using geometric information.

4.4. Relative Camera Pose Estimation

Our last experiment tests the effectiveness of the predicted relative camera pose to do the reconstruction.

Metrics: Following prior works [44, 47, 35, 1], 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 percentage of examples below a fixed threshold. Here we use 30° for rotation and 1m for translation, which are the same as [35].

Baselines: Our goal is to evaluate the effectiveness of our camera pose module and validate our design decisions. The first three are independent branches that test ways to fuse data from two backbone networks; the last use our FPN backbone and test the contribution of reasoning over planes.

Associative3D [35] camera branch: Associative3D average pools ResNet50 features [20], and uses a 2-layer multi-layer perceptron on their concatenation. We report results with a deeper MLP (as many layers, parameters as our approach), which we found to give better results.

ResNet50-CatConv: We concatenate the ResNet50 layers without average pooling, followed by the same number of convolutions as in the cross-correlation network.

ResNet50-CrossCorr: Our proposed network, trained using a ResNet50 as backbone.

Ours-CrossCorr: This is our network *without optimization.*

Ours-Discrete Only: This is our approach without continuous optimization and tests jointly reasoning about planes and cameras with a fixed set of potential camera locations.

Proposed: This is our proposed approach.

Alternate Approaches: There has been considerable work in deep learning for correspondence estimation (e.g., SuperGlue [38]), which can be plugged into classic methods for camera pose estimation. We see these approaches as complementary: the correspondences can be plugged into our final objective (Eqn. 4) instead of SIFT matches, and it does not estimate translation vectors (which our approach obtains due to learning). For context, though, we report rotation errors of the SuperGlue [38] model pretrained on the similar dataset ScanNet [8]: the model obtains rotation median and mean error 3.88°, 24.17° and accuracy 77.84%, and fails on about 11.6% of our scenes due to insufficient correspondence. This points to the complementary nature of the approaches: as a pixel-correspondence-based method, [38] is more accurate with lots of texture and overlap, but our approach can make reasonable inferences on harder scenes.

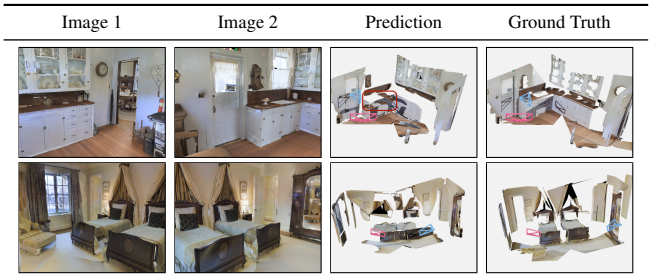

Figure 6. Representative failure modes. **Row 1:** The kitchen countertop has little overlap across views. Though our method stitches them, there is no correct pixel correspondence to constrain the alignment, shown in the **red** box. **Row 2:** The system fails on multiple coplanar segments with very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

Quantitative Results: We report results in Table 3. The full method has low median errors of 0.63m/7.33°. With the ability to explicitly calculate the relationship of features across views, our proposed cross correlation module (ResNet50-CrossCorr) outperforms other standalone architectures by a large margin on all the metrics. Running the cross-correlation module on the plane detection backbone further improves results, but due to switch to a FPN [26], it is difficult to directly ascribe performance changes. Discrete optimization consistently slightly improves results, but continuous optimization substantially improves translation with marginal gains and losses in rotation performance.

5. Conclusion

We 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 leveraging more natural forms of supervision.

Acknowledgments We thank Nilesh Kulkarni, Mohamed El Banani and Richard Higgins for helpful discussions. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

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

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

[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, 18

[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, 5, 11

[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, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. *NeurIPS*, 2016. 1

[7] David J Crandall, Andrew Owens, Noah Snavely, and Daniel P Huttenlocher. Sfm with mrfs: Discrete-continuous optimization for large-scale structure from motion. *IEEE transactions on pattern analysis and machine intelligence*, 2012. 1

[8] 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

[9] 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. *arXiv preprint arXiv:1905.03561*, 2019. 2

[10] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 1, 2

[11] Sovann En, Alexis Lechervy, and Frédéric Jurie. Rpnet: An end-to-end network for relative camera pose estimation. In *ECCV*, 2018. 1

[12] David F. Fouhey, Abhinav Gupta, and Martial Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013. 1

[13] David F Fouhey, Daniel Scharstein, and Amy J Briggs. Multiple plane detection in image pairs using J-linkage. In *ICPR*, 2010. 1

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

[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, second edition, 2004. 1, 2

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 3, 11

[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. 6

[25] Lin Li, Salman Khan, and Nick Barnes. Silhouette-assisted 3d object instance reconstruction from a cluttered scene. In *ICCV Workshops*, 2019. 6

[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, 8

[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, 11

[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, 11

[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. 1, 2, 11

[30] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 2, 4, 11

[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] 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. 6

[33] 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

[34] Phil Pritchett and Andrew Zisserman. Wide baseline stereo matching. In *IEEE International Conference on Computer Vision*, pages 754–760, 1998. 2

[35] Shengyi Qian, Linyi Jin, and David F. Fouhey. Associative3d: Volumetric reconstruction from sparse views. In *ECCV*, 2020. 1, 2, 3, 5, 6, 7, 8, 13, 14, 18

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

[37] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. In *NeurIPS*, 2018. 3

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

[39] 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, 11

[40] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1):7–42, 2002. 2

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

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

[43] 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. 2

[44] 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

[45] 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

[46] 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. 6

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

[48] 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. 5, 14, 15

[49] 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. 3

[50] Xiaolong Wang, David F. Fouhey, and Abhinav Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015. 1, 2

[51] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. 2, 3

[52] Changchang Wu, Brian Clipp, Xiaowei Li, Jan-Michael Frahm, and Marc Pollefeys. 3d model matching with viewpoint-invariant patches (vip). In *CVPR*, 2008. 1, 2, 4

[53] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 4, 11

[54] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013. 1, 5, 14

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

[56] 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

[57] Zhenpei Yang, Siming Yan, and Qixing Huang. Extreme relative pose network under hybrid representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2455–2464, 2020. 1, 2

[58] 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

[59] 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, 11

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

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

[62] 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. Implementation

A.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 [39]. 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 7 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 [59] 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.

A.2. Network Architectures

We use Detectron2 [53] to implement our network. The overall architecture of our network is shown in Table 4. 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 5 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 big 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, 2 2080Ti GPUs for 37k, 36k and 35k iterations to train the plane prediction module, embedding head and camera pose module respectively. We use SGD with momentum of 0.9, a weight decay of 0.0001, multistep learning rate with base $\text{lr} = 0.001$, multiplied by 0.1 after 30k iterations, and warmup across all the training jobs.

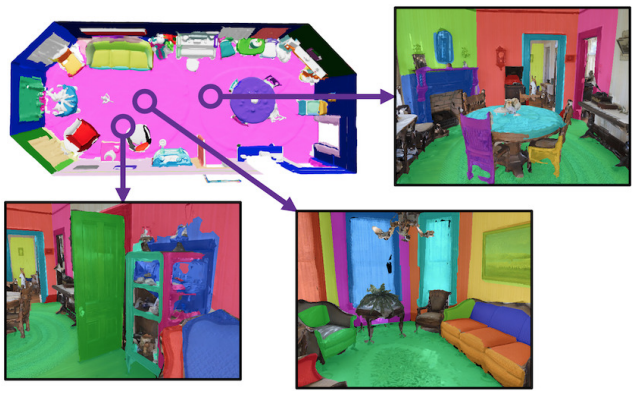

Figure 7. Example plane annotations of our dataset.

A.3. Optimization

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` $= 0.7$, $\lambda_e = 0.47$, $\lambda_n = 0.25$, $\lambda_o = 0.28$, $O_{\text{clamp}} = 4$, $\lambda_N = 0.311$, $\lambda_h = 0.432$, $\lambda_t = 0.166$, $\lambda_R = 0.092$.

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. For the pixel alignment error, we 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

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 $\mathbf{e}'$, $[\mathbf{n}', o']$.

Input: $\mathbf{e}, \mathbf{e}', \mathbf{n}, \mathbf{n}', o, o', \{p_{\mathbf{t}_k}, p_{\mathbf{R}_k}, \mathbf{t}_k, \mathbf{R}_k\}_{k=0,\cdots,K}$
Output: $\mathbf{C}, \{\mathbf{t}_{\hat{k}}, \mathbf{R}_{\hat{k}}\}$

`min_cost` $\leftarrow$ `Inf`
$\hat{k} \leftarrow -1$
$\Delta\mathbf{e}_{ij} = ||\mathbf{e}_i - \mathbf{e}'_j||$
for $k \in \{1, \cdots, K\}$ **do**
  $\mathbf{n}^w, o^w \leftarrow$ `cam2world`$(\mathbf{n}, o, \mathbf{t}_k, \mathbf{R}_k)$
  ${\mathbf{n}'}^w, {o'}^w \leftarrow$ `cam2world`$(\mathbf{n}', o', \mathbf{0}, \mathbf{I}(3))$
  $\Delta\mathbf{n}_{ij} \leftarrow \text{acos}(|\mathbf{n}_i^{w\intercal}{\mathbf{n}'_j}^w|)/\pi$
  $\Delta\mathbf{o}_{ij} \leftarrow \min(|o_i^w - {o'_j}^w|/O_{\text{clamp}}, 1)$
  $\mathbf{S}_k \leftarrow \lambda_e\Delta\mathbf{e} + \lambda_n\Delta\mathbf{n} + \lambda_o\Delta\mathbf{o}$
  $\mathbf{C}_k \leftarrow$ `Hungarian`$(\mathbf{S}_k)$
  $\mathbf{C}_k(\mathbf{S}_k <$ `threshold`$) = 0$
  `cost` $\leftarrow \lambda_h \sum_{i,j}(\mathbf{C}_k \circ \mathbf{S}_k)_{i,j} - \lambda_t \log(p_{\mathbf{t}_k}) - \lambda_R \log(p_{\mathbf{R}_k}) - \lambda_N \sum_{i,j} \mathbf{C}_{k,i,j}$
  if `cost` $<$ `min_cost` **then**
    `min_cost` $\leftarrow$ `cost`
    $\hat{k} = k$
  end if
end for
return $\mathbf{C}_{\hat{k}}, \{\mathbf{t}_{\hat{k}}, \mathbf{R}_{\hat{k}}\}$

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 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.

Camera pose prediction benchmark. We include detailed architectures for baselines in Section 4.4 of our paper in Table 6, 7 and 8. 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 4. 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 5, 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, $\text{Linear}(7 \times 7 \times 256 \to 1024)$, $\text{Linear}(1024 \to 5C)$	$C \times 5$
(6)	(4)	Mask: $4 \times \text{Conv}(256 \to 256, 3 \times 3)$, $\text{TConv}(256 \to 256, 2 \times 2, 2)$, $\text{Conv}(256 \to C, 1 \times 1)$	$28 \times 28 \times C$
(7)	(4)	Normal: $4 \times \text{Conv}(256 \to 256, 3 \times 3)$, $\text{Linear}(14 \times 14 \times 256 \to 1024)$, $\text{Linear}(1024 \to 3)$	$C \times 3$
(8)	(4)	Embedding: $4 \times \text{Conv}(256 \to 256, 3 \times 3)$, $\text{Linear}(14 \times 14 \times 256 \to 1024)$, $\text{Linear}(1024 \to 128)$	$C \times 128$
(9)	(2)	Depth	$H \times W \times 1$
(10)	(2), (2)'	Camera pose module	$1 \times 32 + 1 \times 32$

Table 5. 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 4). 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 cross correlation 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)	$2 \times \text{Conv}(256 \to 256, 3 \times 3)$, `Maxpool`	$2 \times 30 \times 40 \times 256$
(3)	(2)	$2 \times \text{Conv}(256 \to 256, 3 \times 3)$, `Maxpool`	$2 \times 15 \times 20 \times 256$
(4)	(3)	$\text{Conv}(256 \to 256, 3 \times 3)$, $\text{Conv}(256 \to 512, 3 \times 3)$	$2 \times 15 \times 20 \times 512$
(5)	(4)	`CrossCorrelation`	$300 \times 15 \times 20$
(6)	(5)	$\text{Conv}(300 \to 128, 3 \times 3)$, $5 \times \text{Conv}(128 \to 128, 3 \times 3)$,	$128 \times 2 \times 3$
(7)	(6)	$\text{Linear}(128 \times 2 \times 3 \to 64)$	1×64
(8)	(7)	Translation: $\text{Linear}(64 \to 32)$	1×32
(9)	(7)	Rotation: $\text{Linear}(64 \to 32)$	1×32

Table 6. The architecture for benchmark *Associative3D [35] 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)	$\text{Linear}(4096 \to 512), 5 \times \text{Linear}(512 \to 512), \text{Linear}(512 \to 64)$	1×64
(5)	(4)	Translation: $\text{Linear}(64 \to 32)$	1×32
(6)	(4)	Rotation: $\text{Linear}(64 \to 32)$	1×32

Table 7. 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)	$\text{Conv}(4096 \to 128, 3 \times 3), 5 \times \text{Conv}(128 \to 128, 3 \times 3)$,	$128 \times 2 \times 3$
(4)	(3)	$\text{Linear}(128 \times 2 \times 3 \to 64)$	1×64
(5)	(4)	Translation: $\text{Linear}(64 \to 32)$	1×32
(6)	(4)	Rotation: $\text{Linear}(64 \to 32)$	1×32

Table 8. The architecture for benchmark *ResNet50-CrossCorr* camera branch in Section 4.4 of our paper. It takes as input the features of the ResNet50 backbone from both images, computes the cross correlation 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)	`Cross_correlation`	$300 \times 15 \times 20$
(3)	(2)	$\text{Conv}(300 \to 128, 3 \times 3), 5 \times \text{Conv}(128 \to 128, 3 \times 3)$,	$128 \times 2 \times 3$
(4)	(3)	$\text{Linear}(128 \times 2 \times 3 \to 64)$	1×64
(5)	(4)	Translation: $\text{Linear}(64 \to 32)$	1×32
(6)	(4)	Rotation: $\text{Linear}(64 \to 32)$	1×32

B. Additional Results

B.1. Dataset

To give additional context for the dataset, we computed some statistics on 200 random image pairs from the dataset. We make sure 80% of their pixels have valid depth values. We transform the point cloud of view 1 to view 2. The average percentage of overlapping points was 21%. For context, we did the same for the Sun3D [54] image pairs used by DeMoN [48]: these had an average overlap of 64%.

Random examples are shown in Figure 8 and 9. We can see our dataset is much more sparse than DeMoN. It is because DeMoN samples next few frames of the video sequence, while we render a sparse view dataset ourselves.

B.2. 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 9, 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, all methods have a higher AP (increase by over 10 points compared to 100% data). Moreover, our proposed method has a higher gain over the *Associative3D optimization* and *Top 1 camera, appearance only* by above 4.9 points on top 25% examples. It is still significant but slightly lower when we have worse single-view predictions.

Table 9. 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, a prediction is a true positive only if Mask IoU ≥ 0.5, Normal error $\leq 30^\circ$ and Offset error $\leq$ 1m.

Methods	Top 25%	Top 50%	Top 75%	100%
Top 1 camera, no merge	41.64	37.04	33.71	29.44
Top 1 camera, appearance only	47.83	42.17	38.06	33.04
Associative3D [35] Optimization	47.51	41.93	38.01	33.01
Discrete Optimization, Appearance + Geometry	52.15	46.07	41.47	35.87
Proposed	**52.75**	**46.45**	**41.76**	**36.02**

B.3. Qualitative Results

We first present our reconstruction results on selected examples in Figure 10, 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 11, according to single-view AP. As we quantitatively show in Section B.2, 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 12, 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 13 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.

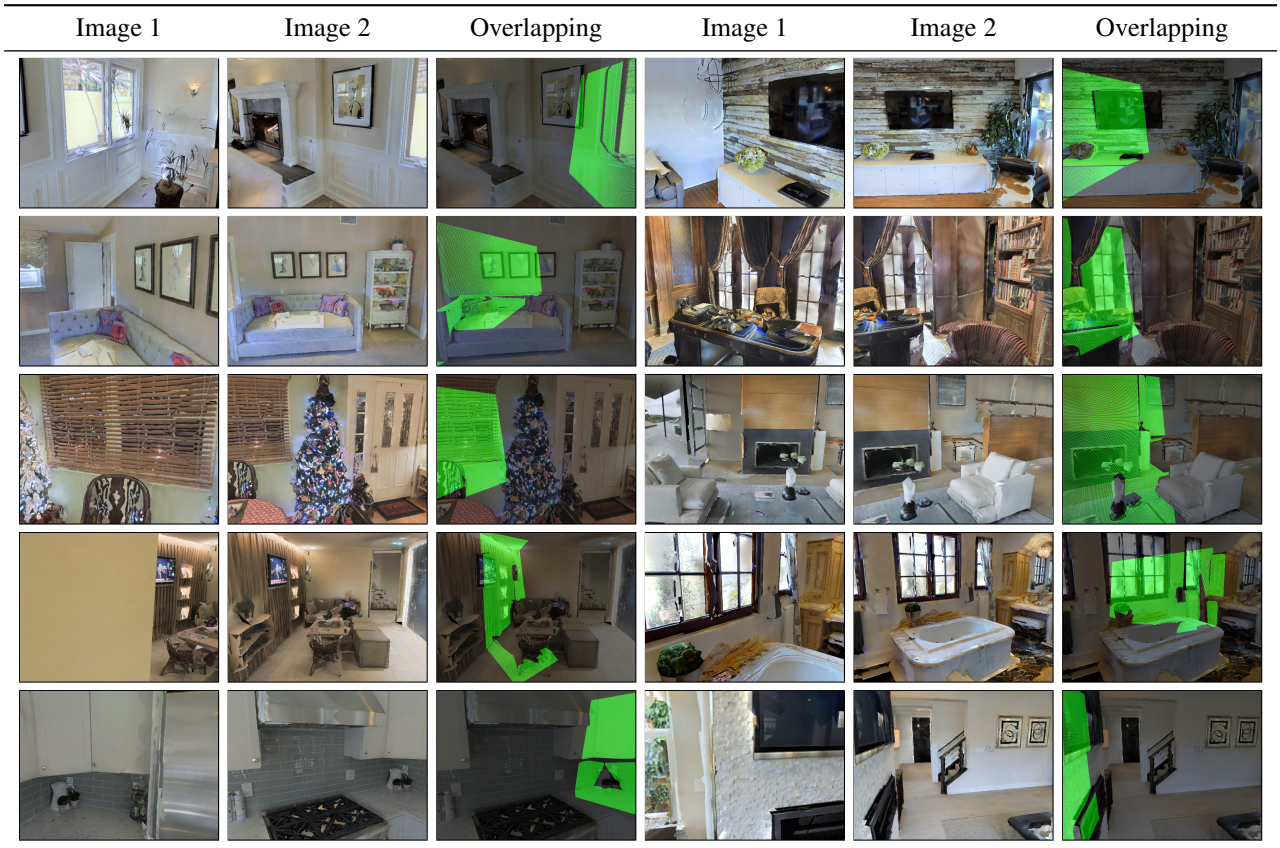

Figure 8. Overlapping regions of random examples of our dataset. We transform the point cloud of image 1 to image 2 and show it as a green mask.

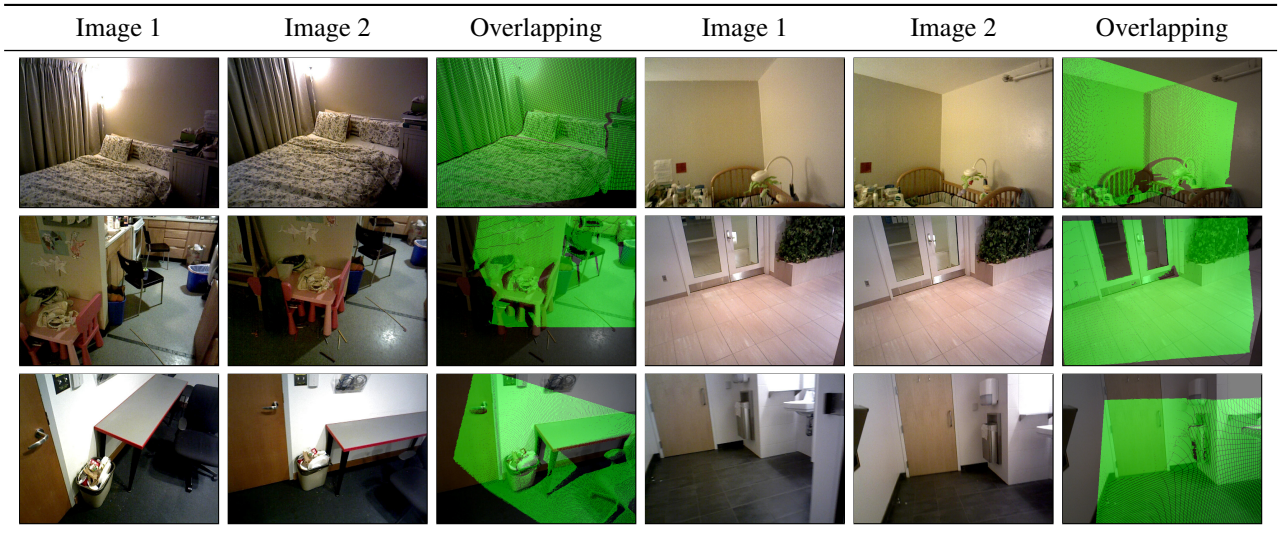

Figure 9. Overlapping regions of random examples of DeMoN [48]. We transform the point cloud of image 1 to image 2 and show it as a green mask.

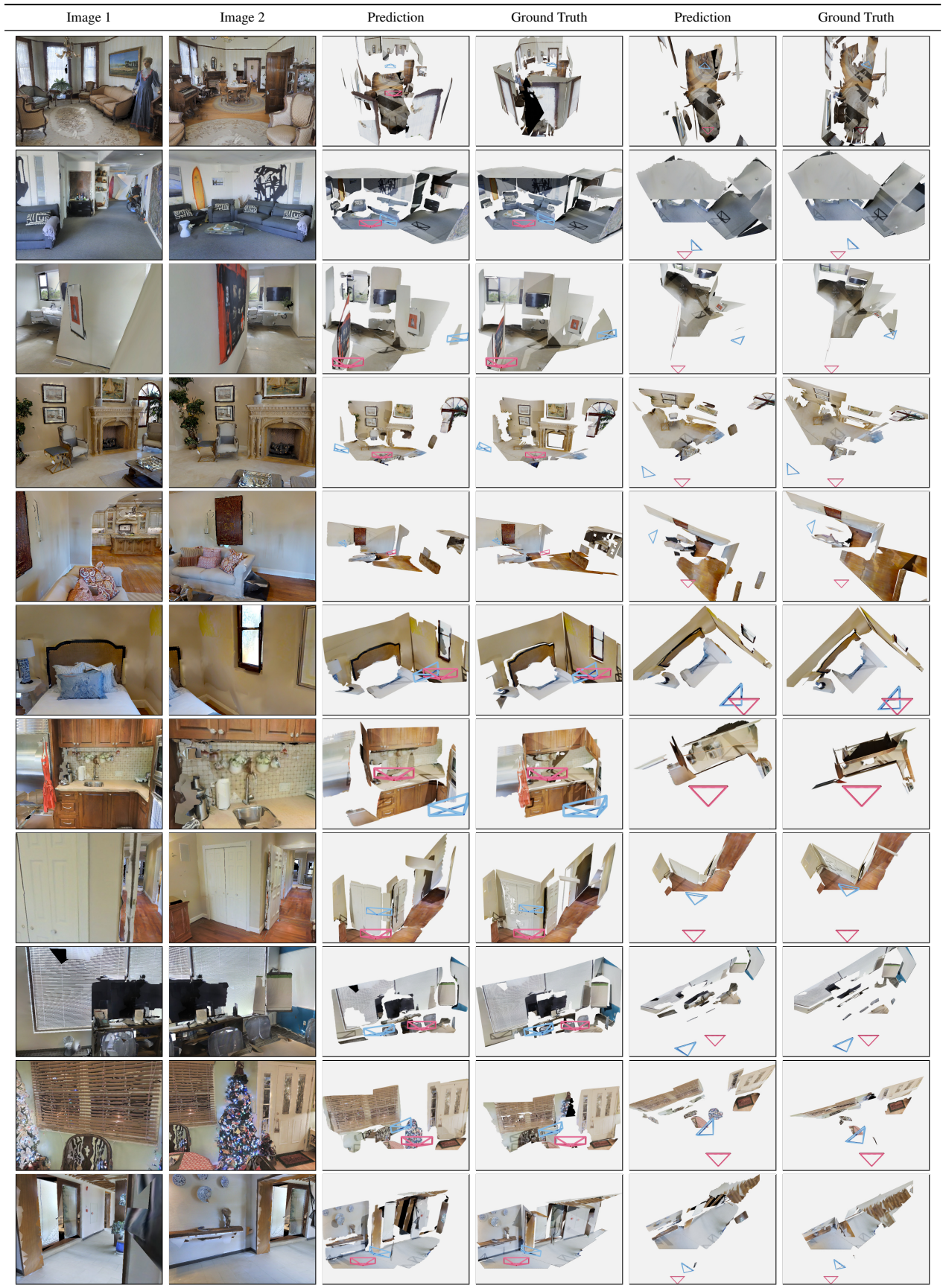

Figure 10. More results on Matterport3D test set, extending Figure 3 in our paper, **Blue** and **Red** frustums show cameras for image 1 and 2.

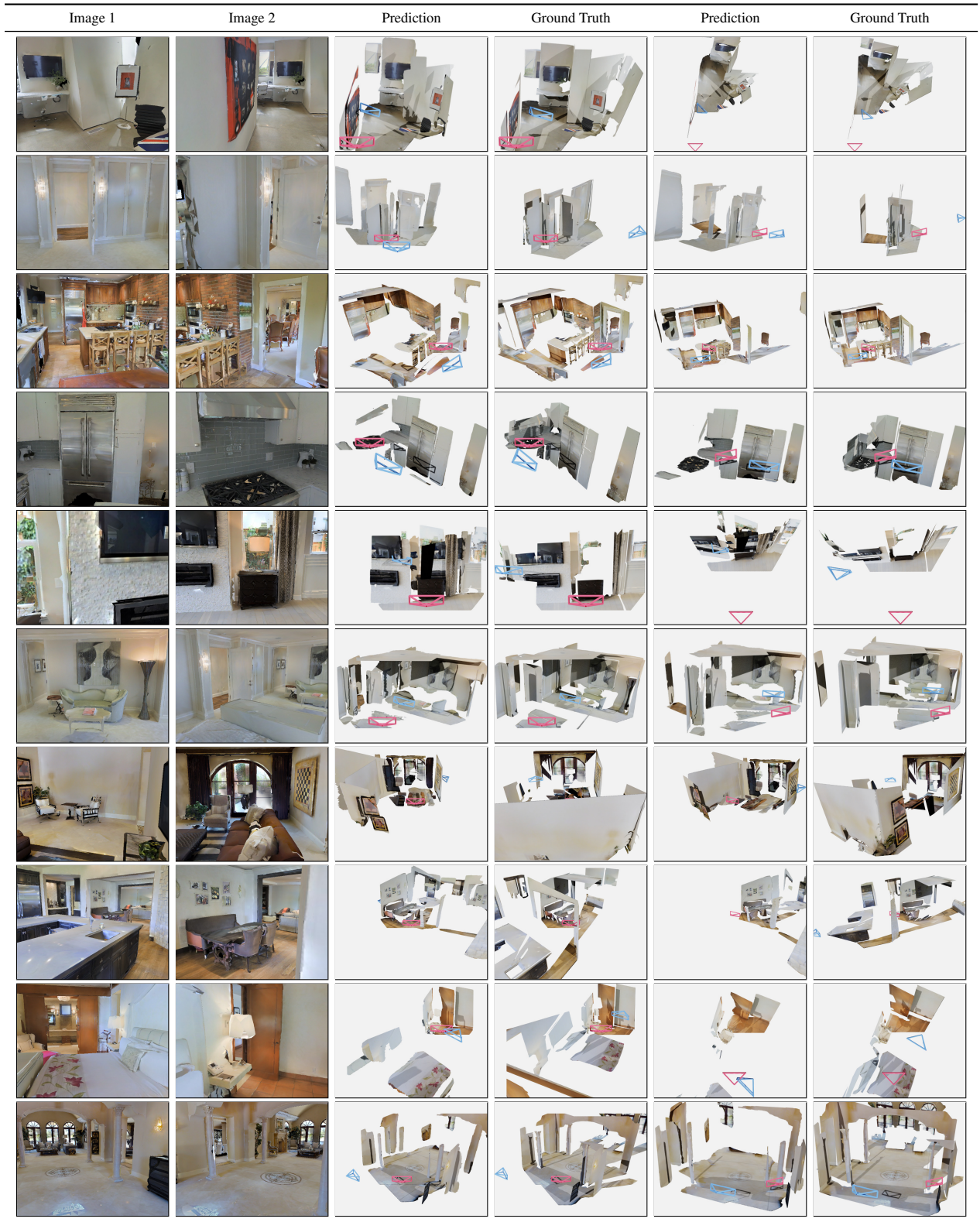

Figure 11. Random results on Matterport3D test set, automatically evenly spaced according to single-view AP. The first row has the highest single-view AP. **Blue** and **Red** frustums show cameras for image 1 and 2.

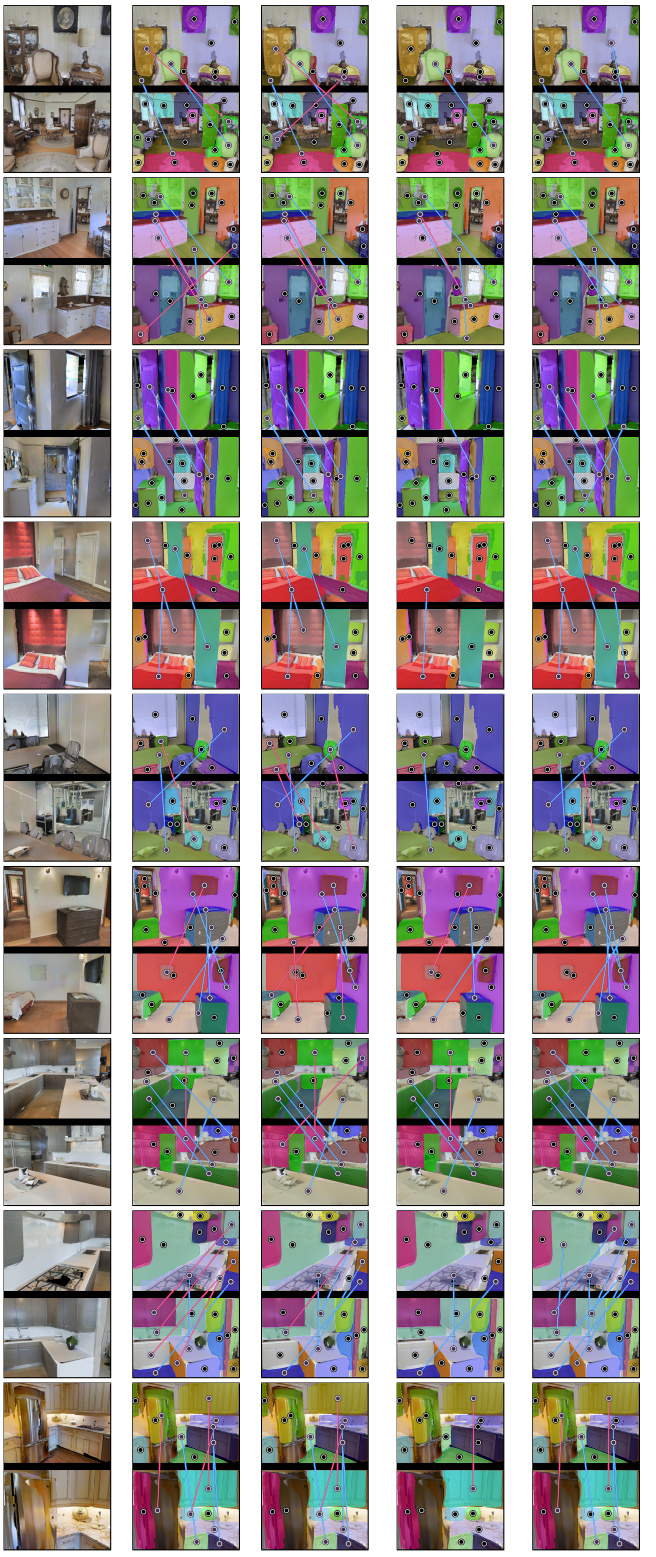

Figure 12. Selected correspondence predictions on *predicted* boxes, extending Figure 5 in our paper, showing true positive matches in **Blue** and false positives in **Red**. Correct matches are determined by assigning each ground truth box with a predicted box whose Mask IoU is greater than 0.5.

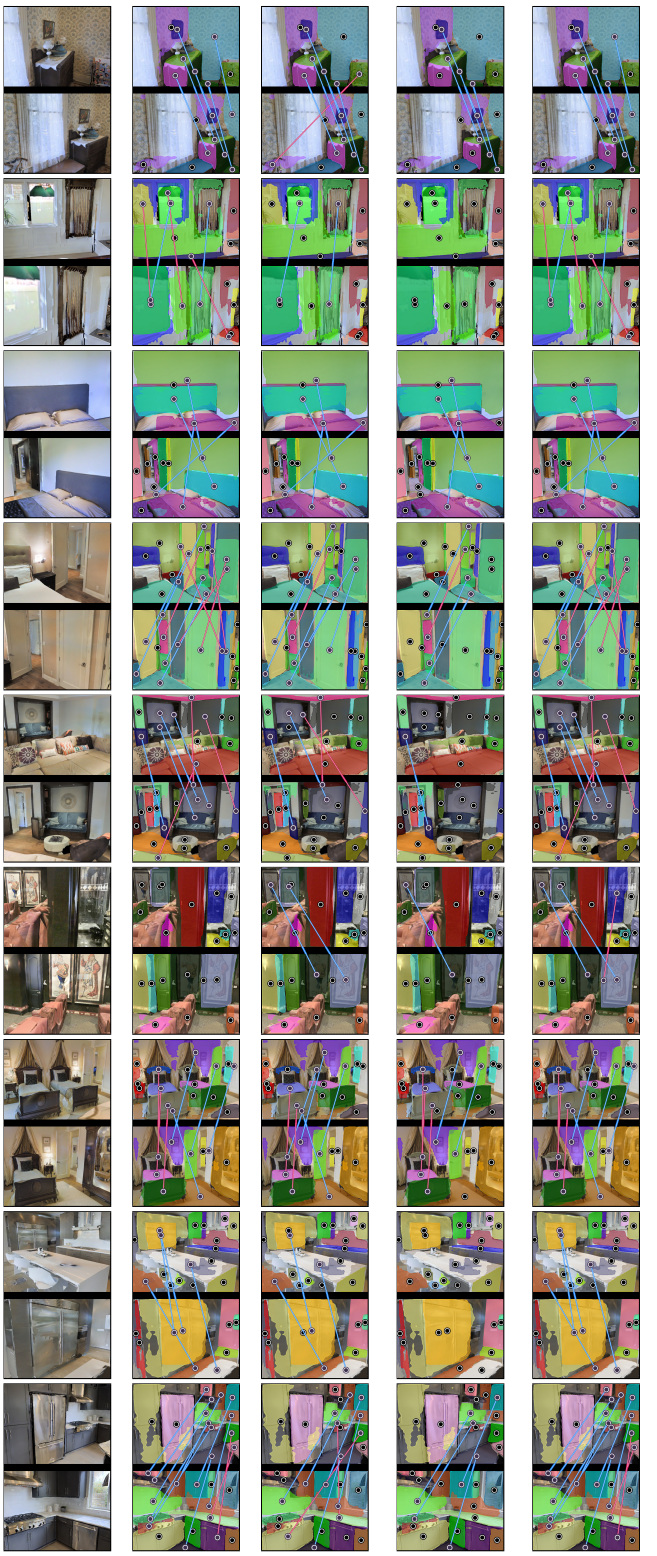

Figure 13. Random correspondence predictions on *ground truth* boxes, showing true positive matches in **Blue** and false positives in **Red**.