Supplementary Material
Indoor Panorama Planar 3D Reconstruction via Divide and Conquer

Contents

1. Ground-truth H&V-plane annotation

This section provides more details about the extraction of *HV-planes* from ground-truth depth.

1.1. Local analysis

The goal is to classify each pixel into 'H-pixel', 'V-pixel', or 'other' for later HV-plane extraction. The same rule is generally applied to all pixels, so we describe how we classify a pixel p for simplicity as follows. We relocate the world origin to the 3D coordinate of point p (same process applied to other points); the z-axis is aligned with gravity, and all the 3D points from the same image column are on the plane formed by y-axis and z-axis. Let $(0, y_t, z_t)$ and $(0, y_b, z_b)$ be the 3D coordinates from the top and bottom adjacent pixels respectively. Let d be the depth value of pixel p. We calculate the following information:

$T_H \leftarrow \min(d \cdot 5,\ 40)$ {threshold for 'H'}
$T_V \leftarrow 90 - T_H$ {threshold for 'V'}
$\theta_t \leftarrow \arctan2(|z_t|, |y_t|)$ {degree in range $[0°, 90°]$}
$\theta_b \leftarrow \arctan2(|z_b|, |y_b|)$ {degree in range $[0°, 90°]$}
$type_t \leftarrow \text{thresholdHVO}(\theta_t, T_H, T_V)$
$type_b \leftarrow \text{thresholdHVO}(\theta_b, T_H, T_V)$

T_H and T_V are depth-dependent thresholds (the farther, the more tolerant). The procedure 'thresholdHVO' is defined as

if $\theta < T_H$ **and** $\max(|z_t|, |z_b|) < 0.1$ **then**
  $type \leftarrow$ 'H'
else if $\theta > T_V$ **and** $\max(|y_t|, |y_b|) < 0.1$ **then**
  $type \leftarrow$ 'V'
else
  $type \leftarrow$ 'O'
end if

We then classify the pixel based on the states of $type_t$ and $type_b$ using the rule depicted in Table 1. The 'TBD' in Table 1 needs to be further determined by θ_t and θ_b:

if $type_t =$ 'V' **and** $type_b =$ 'H' **then**
  if $90 - \theta_t < \theta_b$ **then**
    return 'V-pixel'
  else
    return 'H-pixel'
  end if
else if $type_t =$ 'H' **and** $type_b =$ 'V' **then**
  if $\theta_t < 90 - \theta_b$ **then**
    return 'H-pixel'
  else
    return 'V-pixel'
  end if
end if

$type_t$ / $type_b$	'H'	'V'	'O'
'H'	'H-pixel'	TBD	'H-pixel'
'V'	TBD	'V-pixel'	'V-pixel'
'O'	'H-pixel'	'V-pixel'	'other'

Table 1: Type of a pixel according to the relationship with its top and bottom adjacent pixels

Finally, we use connected component (CC) analysis on 'H-pixel' and 'V-pixel', and re-assign small CC into 'other'.

1.2. Horizontal planes extraction

A horizontal plane has only one degree of freedom that can be parameterized by the signed distance (for up/down in 3D) to the camera height. We identify horizontal planes in a manner similar to non-maximum suppression:

1. Using linear search to find an H-plane covering the

largest number of remaining H-pixels within 5cm;

2. Removing any connected component (CC) on the image that contains fewer than 1,000 pixels (*i.e.* $\approx 0.2\%$ of the image size);
3. Masking out inlier H-pixels.

The process iterates until no H-plane can be found. The extracted H-planes are refined by reassigning all H-pixels to their nearest H-planes and recalculating the parameters of H-planes based on new inlier H-pixels.

1.3. Vertical planes extraction

A vertical plane has two degrees of freedom, which can be characterized by $\vec{n} = [x\ y\ 0] = o \cdot [\cos\theta\ \ \sin\theta\ 0]$—the unit surface normal $[\cos\theta\ \ \sin\theta\ 0]$ scaled by the plane offset o. Similar to H-plane extraction, we apply the following process:

1. RANSAC finding a plane $\vec{n}$ that covers the largest number of V-pixels within a threshold $T = \min(o \cdot 0.05, 0.2)$; A 3D point is said to be an inlier if the distance to the plane is less than the threshold;
2. Keeping only the largest CC as a V-plane instance since the detected plane $\vec{n}$ by RANSAC extends infinitely in 3D and could cover multiple non-connected planes.
3. Masking out inlier V-pixels.

The process iterates until the largest CC contains fewer than 1,000 pixels. The extracted V-planes are refined by reassigning border pixels of instances to their nearest V-planes. This step could cut an instance into different CCs; only the largest CC of the instance remains, and the other smaller-sized CCs are reassigned to the majority of their neighbors. Finally, V-plane parameters are refined by solving the least-squares based on the new inliers.

2. Relation with panorama room layout

The room layout reconstruct the highest-level structure of a room, which is consists of only a few facades and the result is up to a scale. Besides, most of the room layout reconstruction methods currently only consider one-floor-one-ceiling model. In plane evaluation metrics, many factors can even makes the detected layout facades to be counted as false positive, *e.g.*, planes of furniture or objects taking a large amount of pixels of a wall, beam, column, gates or doors to another area.

3. Detailed network architecture

The architecture for pixel-level feature extraction is depicted in Fig. 1. We employ ResNet101 as our backbone network, which provides features in four different output stride. We use ConvBlock to denote a sequence of

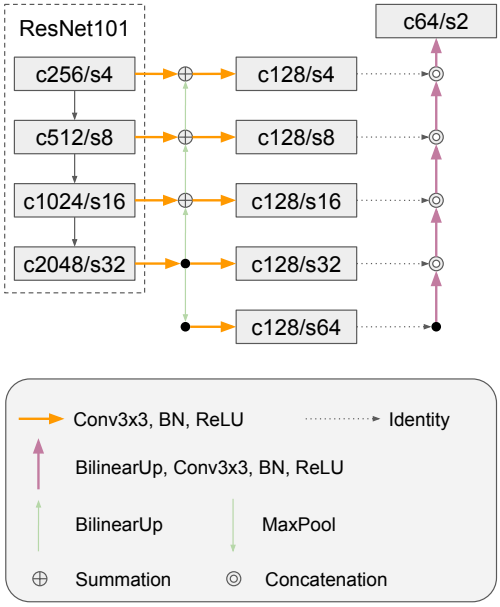

Figure 1: Detailed network architecture employed to extract per-pixel features. The "cX/sY" indicates that the feature tensor has X latent channels with spatial stride Y.

Conv3x3, BN, ReLU. Each backbone feature tensor is reduced to 128 latent channels by two ConvBlocks, where the intermediate features are summed with the upsampled coarser-scale features to capture a longer range of context. We also adding one extra level by MaxPool. Finally, from coarse to fine levels, the features are upsampled, refine by ConvBlock and concatenated with the finer level features. The overall encoder-decoder network produce a pixel-level feature tensor at output stride 2 with 64 latent channels, which is then followed by different Conv1x1 layer to estimate each modality for HV-planes reconstruction.

4. Visualization of yaw-rotation ambiguity in 360° images

We visualize the effect of camera yaw rotation on V-plane parameters in a 360° image in Fig 2, where planes are colored according to the angles of the plane orientations. We can see that values of the standard plane representation vary with different yaw rotations of 360° camera. This property, however, is inconsistent with the fact that the convolutional neural network is translation-invariant and therefore less aware of the yaw rotation.

To address the yaw ambiguity effectively, we propose to re-parameterize the ground-truth plane orientation θ_i^* of each pixel i into *residual form* with respect to the pixel yaw viewing angle u_i such that it is invariant to the 360° camera yaw rotation:

$$\theta'^*_i = \theta^*_i - u_i\,. \tag{1}$$

We show the same scene as Fig 2 but with our camera yaw invariant plane representation in Fig 3, where the ground-truth orientation is now camera yaw-invariant.

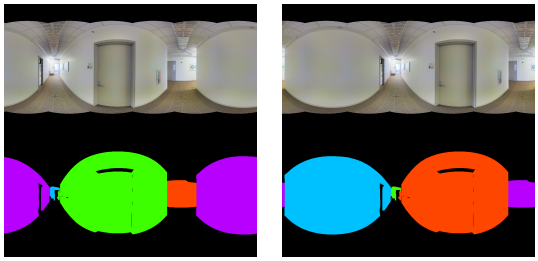

Figure 2: RGB and ground-truth V-planes colored by the angles of plane orientations. The left and right figures are two views of the exact same scene with different camera yaw rotations.

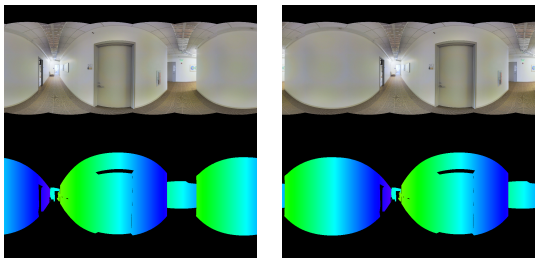

Figure 3: The proposed camera yaw-invariant plane representation.

5. V-planar orientation grouping

Our proposed plane instance segmentation module is geometry-aware. More specifically, we group all pixels with similar plane orientations in the first stage of the plane instance segmentation process. We describe below how we instantiate the grouping process for pixels classified as 'V-planar'. Let $\theta_i \in [-\pi, \pi]$ denote the reconstructed V-plane orientation in radian of a V-planar pixel with index i. We discretize all $\{\theta_i \mid \text{pixel } i \text{ is V-planar}\}$ into 360 bins. Note that the histogram is circular, where the first bin is adjacent to the last bin. We then find all prominent peaks on the histogram with two criteria: *i*) the vote is larger than any bins within the nearest 50 bins, and *ii*) it has at least 100 votes. V-pixels are assigned to their nearest peak to form surface orientation groups.

6. More quantitative comparison of intermediate version

We show more intermediate version of our method in Table 2. Our based method (w/o *yaw-inv*, w/o *ori-gp*) shows slightly better result on pixel-recall and plane-recall comparing to PlaneAE, which could be due to the difference in model architecture and the output modalities for representing plane parameter. By applying the *ori-gp*, we can see consistent improvement on segmentation quality, especially when it works with the *yaw-inv* (the quality of *ori-gp* depend on the quality of the planar yaw-angle estimation). Further, adding *yaw-inv* to our system leads to a significant improvement on pixel-recall and plane-recall which considering not only the segmentation quality but also geometry quality.

Method	Segmentation Quality			Recall↑	
	ARI↑	VI↓	SC↑	Pixel	Plane
PlaneAE [4]	0.765	1.536	0.733	0.520	0.341
w/o *yaw-inv* w/o *ori-gp*	0.762	1.555	0.734	0.542	0.360
w/o *yaw-inv*	0.764	1.542	0.738	0.545	0.369
w/o *ori-gp*	0.762	1.554	0.732	0.619	0.417
full	0.768	1.514	0.742	0.627	0.430

Table 2: More quantitative results on Stanford2d3d of our method's intermediate version. A strong baseline—PlaneAE—is also shown for comparison. *yaw-inv*: the yaw-invariant representation. *ori-gp*: the orientation grouping as a pre-filtering before mean shift.

7. More qualitative results reconstructed by our method.

We show more visual results on the two real-world dataset—Stanford2D3D [1] dataset and Matterport3D [2] dataset—in Fig. 4, Fig. 5 and Fig. 7.

8. Qualitative comparisons with baselines

We compare more qualitative results of the two baselines—PlaneRCNN [3] and PlaneAE [4]—and our method in Fig. 6. The first column shows input RGB and ground-truth plane instance segmentation. The second to the fourth columns respectively show results reconstructed by PlaneRCNN [3], PlanarReconstruct [4], and our method. For each scene, we show the planar depth error (clipped to 3 meters) in the top row and the predicted plane instance segmentation in the bottom row. We use red circles to highlight obvious errors.

Fig. 4 5 6 7 are in the next few pages.

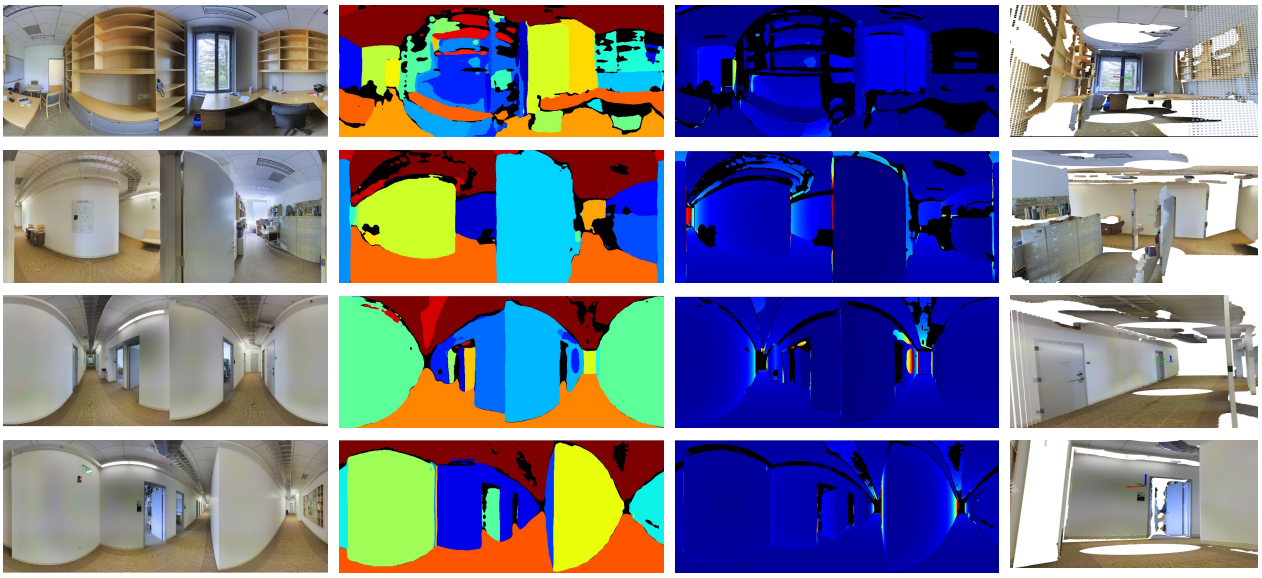

Figure 4: Visual results on Stanford2D3D [1] validation set. The first to the fourth columns show input RGB, plane instance segmentation detected by our method, planar depth error, and snapshot in the reconstructed 3D planar model.

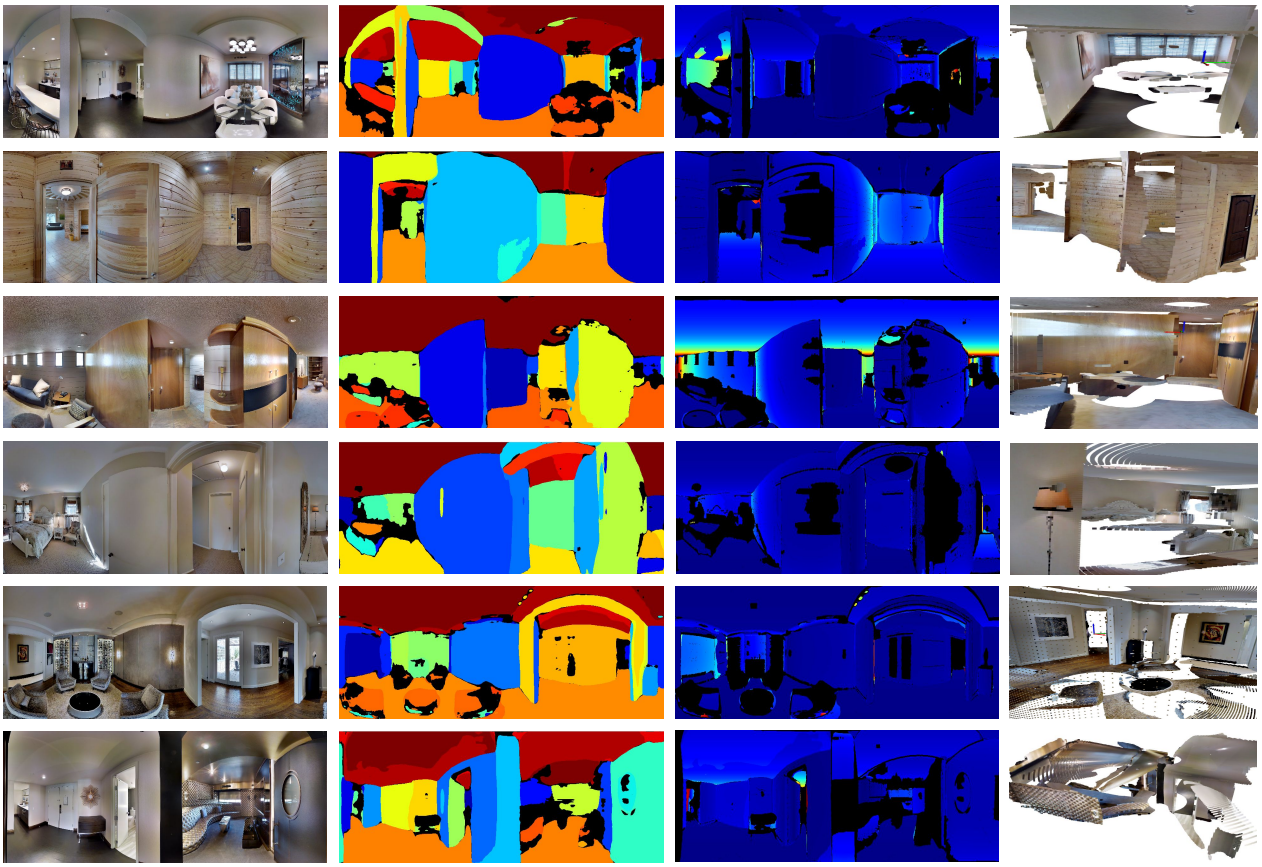

Figure 5: Visual results on Matterport3D [2] test set. The first to the fourth columns show input RGB, plane instance segmentation detected by our method, planar depth error, and snapshot in the reconstructed 3D planar model.

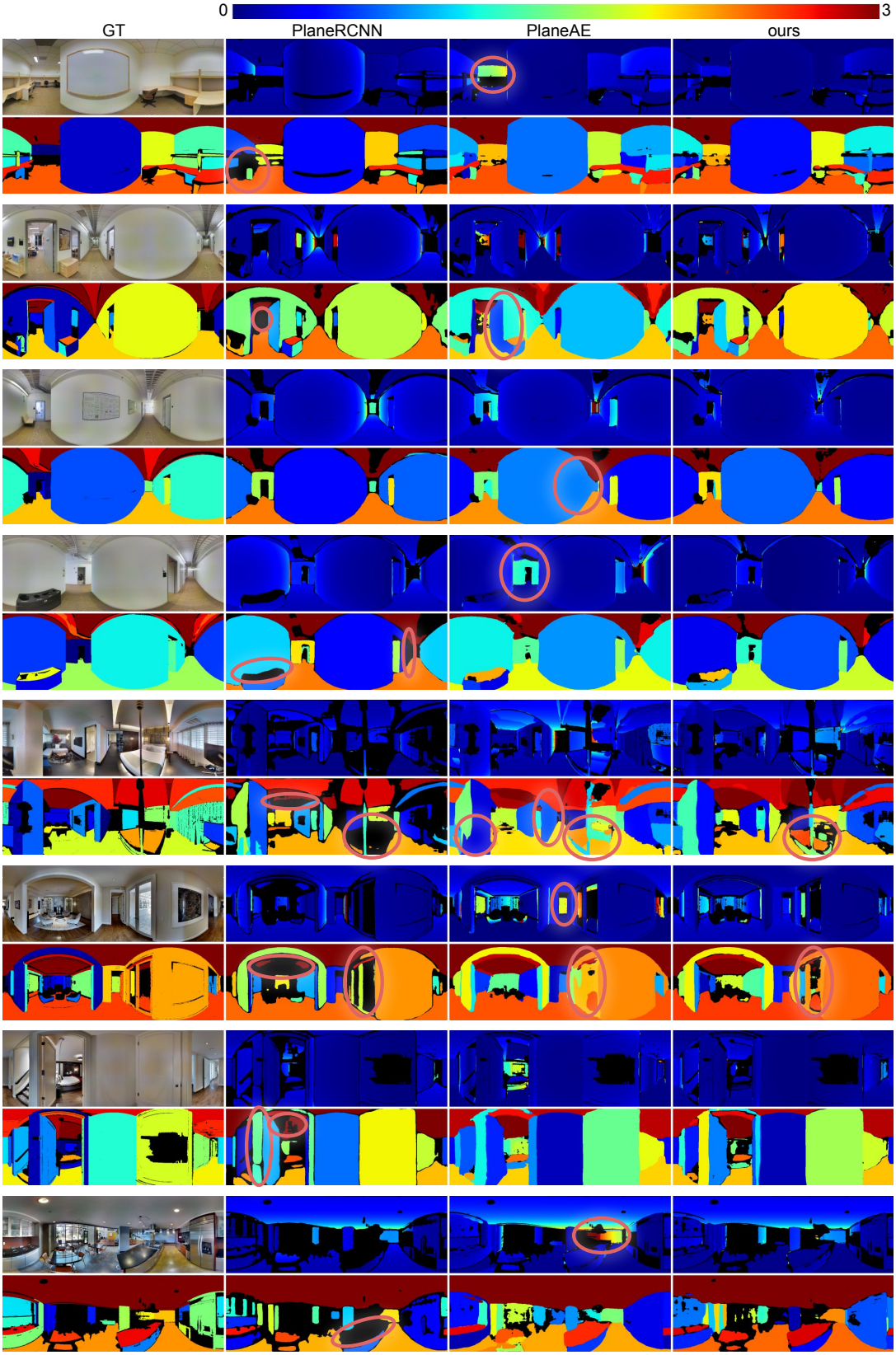

Figure 6: Qualitative comparisons with baselines. (See main text for details.)

Figure 7: 3D mesh visualization from our approach.

References

[1] Iro Armeni, Sasha Sax, Amir Roshan Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *CoRR*, abs/1702.01105, 2017. 3, 4

[2] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: learning from RGB-D data in indoor environments. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 667–676, 2017. 3, 4

[3] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. PlaneRCNN: 3d plane detection and reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4450–4459, 2019. 3

[4] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 1029–1037, 2019. 3