P2-Net: Joint Description and Detection of Local Features for Pixel and Point Matching

Supplementary Material

Bing Wang[1], Changhao Chen[2], Zhaopeng Cui[3], Jie Qin[4], Chris Xiaoxuan Lu[5], Zhengdi Yu[6], Peijun Zhao[1], Zhen Dong[7], Fan Zhu[8], Niki Trigoni[1], Andrew Markham[1]

[1]University of Oxford [3]Zhejiang University [5]University of Edinburgh [7]Wuhan University
[2]National University of Defense Technology [4]Nanjing University of Aeronautics and Astronautics [6]Durham [8]IIAI

In this supplementary material, we firstly provide the implementation details of our proposed P2-Net framework, the training strategy and dataset pre-processing. Then, the details of the evaluation metrics for pixel and point matching are presented. Finally, we report additional quantitative and qualitative matching results and analyze the changes of descriptor similarity under different network settings.

1. Implementation Details

Network architecture. Our proposed P2-Net for pixel and point matching consists of two fully convolutional networks for 2D and 3D feature extraction. In the image (2D) feature extraction network, we use nine 3×3 dilated convolutions with stride=1 in order to preserve the input resolution at all stages. The output feature dimensions and dilation rates for these layers are $(32, 32, 64, 64, 128, 128, 128, 128, 128)$ and $(1, 1, 2, 2, 4, 4, 4, 8, 16)$, respectively. In the former 6 layers, we apply LeakyReLU (0.1) after the convolution operation. In the point cloud feature extraction network, we exploit the *rigid* KP-FCNN [8], which is a fully convolutional network for segmentation. The output feature dimensions in the encoder part are set to $(64, 128, 256, 512, 1024, 2048)$. Skip connections are used between the corresponding layers of the encoder part and the decoder part. For each layer i, a cell size Δ_i is set to infer other parameters. Here, we adopt $\Delta_0 = 0.015$m as the initial cell size. Then, the convolution radius and the average kernel point radius are automatically set to $2.5\Delta_i$ and $1.5\Delta_i$. The output features are processed by a Linear Layer to get the final 128 dimensional descriptors. Besides, we disable batch normalization after all convolutions and replace ReLU with LeakyReLU (0.1). Other settings are kept the same as in the original paper [8].

Training strategy. Our P2-Net is trained with a two-stage training strategy. Specifically, we enable the detector to be jointly trained with the descriptor from the second epoch instead of from the beginning. Our implementation is trained on a single NVIDIA Tesla V100 card, and the training process finishes within 10 epochs. During training, the learning rate exponentially decays with every epoch, and is thus equal to one-tenth of the base learning rate at completion. We find that the two-stage training yields faster convergence. To be specific, one-stage training takes approximately 40 epochs to reach similar results achieved by two-stage training. Although we have tried different training strategies, e.g., changing the initial learning rate and adopting different decay rates, none of them have shown faster convergence results than the two-stage strategy that simply enables later training of the detector. This is because, for accurate keypoint detection, we provide the finest guidance for the gradient of the similarity. Such a design requires distinctive descriptors to provide explicit guidance for detection, which is hard to achieve at the beginning of training. Therefore, to speed up the training, we only focus on optimizing descriptors for the first epoch and then jointly train the descriptor and detector.

Dataset preprocessing. Our network requires a dataset of image and point cloud pairs (I, P) labelled with 2D-3D correspondences to train. Based on 7Scene dataset [4, 7], we firstly follow previous works [11, 10] to obtain the fused point cloud fragments with 5 consecutive RGB-D frames. Then we apply grid subsampling with a voxel size of 0.015m to control the number of points and ensure the spatial consistency of point clouds. To label the correspondences for (I, P), we transform pixels to 3D points by applying its transformation matrix, followed by mutually searching their nearest neighbors in the corresponding point cloud fragment lying in the camera frustum. Pixel and point pairs whose Euclidean distance lie within a threshold of $\Delta_0 = 0.015$m are considered as correspondences. Similar to [3], we use the (I, P) presenting at least 128 correspondences in order to obtain meaningful gradients. Finally, we generate a total number of $24,740$ image and point cloud pairs for training, and $1,591$ pairs for testing, with varying environmental settings. In addition, each point cloud frag-

ment is augmented by adding Gaussian noise with standard deviation 0.005 only during training. Meanwhile, each input image is standardized to be zero mean and unit norm for both training and testing.

2. Evaluation Metrics

In order to evaluate and compare the performance of various loss formulations and different networks on pixel and point matching, five commonly adopted evaluation metrics from [6, 3, 2, 5, 11, 1] are used in our evaluation. These metrics are formulated as:

Inlier ratio denotes the ratio of inliers to all matches, which is used to evaluate the correspondence quality. Given a partially overlapped image and point cloud pair $M(I, P)$, the correspondence set Ω_M for the pair M is obtained by mutually nearest neighbor search in feature space,

$$\Omega_M = \{x_i \in I, y_j \in P | F_I(x_i) = NN(F_P(y_j), F_I(I)), \\ F_P(y_j) = NN(F_I(x_i), F_P(P))\}, \tag{1}$$

where F_I and F_P are two descriptor networks that map the input (i.e. pixels x_i or points y_j) to feature descriptors, NN denotes the nearest neighbor search based on the Euclidean distance. Finally the inlier ratio of M (IR_M) is defined as,

$$IR_M = \frac{1}{|\Omega_M|} \sum_{(i,j)\in\Omega_M} \mathbb{1}(||\Gamma(x_i) - T^{-1}y_j|| < \tau_2), \tag{2}$$

in which Γ represents the mapping function that converts a pixel to its corresponding point under the current camera coordinate system. T is the ground truth transformation between the image and point cloud pair $M \in \mathcal{M}$. Here, the correspondence whose spatial distance is less than the inlier distance threshold $\tau_2 = 0.045$m, can be seen as an inlier.

Feature matching recall measures the percentage of image and point cloud pairs that can recover the pose with high confidence, reflecting the quality of features without using a RANSAC pipeline. Mathematically, it can be written as

$$FMR = \frac{1}{|\mathcal{M}|} \sum_{M=1}^{|\mathcal{M}|} \mathbb{1}\Big(IR_M > \tau_1\Big), \tag{3}$$

where τ_1 is the inlier ratio threshold of the pair M, which will be counted as one match if IR_M is above $\tau_1 > 0.5$.

Keypoint repeatability refers to the percentage of repeatable keypoints over all detected keypoints. For a pair M, the pixel and point pairs set Φ_M is:

$$\Phi_M = \{x_i \in I, y_j \in P | y_j = NN(\Gamma(x_i), P)\}, \tag{4}$$

then the keypoint repeatability of M (KR_M) is defined as,

$$KR_M = \frac{1}{|\Phi_M|} \sum_{(i,j)\in\Phi_M} \mathbb{1}(||\Gamma(x_i) - T^{-1}y_j|| < \tau_3). \tag{5}$$

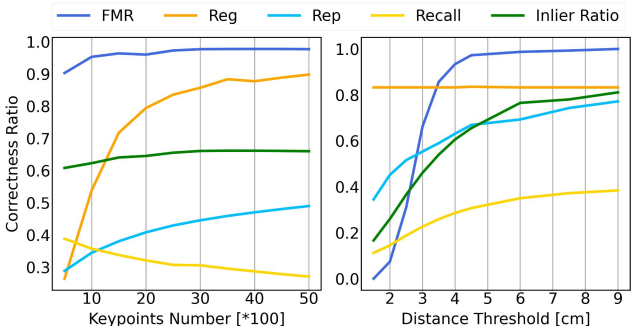

Figure 1: Performance in relation to the variation of keypoints number (left) and threshold (right).

This means that a keypoint in the image is considered repeatable if its spatial distance to the nearest keypoint in the point cloud is less than a threshold $\tau_3 = 0.02$m.

Registration recall represents how many overlapped image and point cloud pairs a matching algorithm can correctly recover, indicating the quality of features within a reconstruction system. Specifically, the registration recall (Reg) uses the following error metric between estimated pairs M, and corresponding pose $\hat{T}$ estimated by PnP+RANSAC to define a true positive:

$$Reg = \\ \frac{1}{|\mathcal{M}|} \sum_{M=1}^{|\mathcal{M}|} \mathbb{1}\Big(\sqrt{\frac{1}{|\Omega_M^*|} \sum_{(x^*,y^*)\in\Omega_M^*} ||\Gamma(x_*) - \hat{T}^{-1}y^*||^2} < \tau_4\Big), \tag{6}$$

where Ω_M^* is a set of corresponding ground truth pairs (x^*, y^*) in M:

$$\Omega_M^* = \{x^* \in I, y^* \in P\}, \tag{7}$$

and τ_4 is set to 0.05m.

Recall denotes the ratio of correct matches over all ground truth matches. Mathematically, it is expressed as:

$$R_M = \frac{1}{|\Omega_M^*|} \sum_{(i,j)\in\Omega_M} \mathbb{1}(||\Gamma(x_i) - T^{-1}y_j|| < \tau_2), \tag{8}$$

in which R_M represents the Recall for the pair M.

3. Additional Results

The impacts of keypoints and thresholds on performance. To better demonstrate the performance of a joint learning under different settings, we further report the results in the cases when increasing the sampled point number from 500 to 5000 and varying the thresholds from 1cm to 9cm. As shown in Fig. 1, with the increasing number of keypoints, all evaluation metrics, except *Recall*, grew accordingly. This is because, when more keypoints are selected, the ambiguity among them will also increase, which

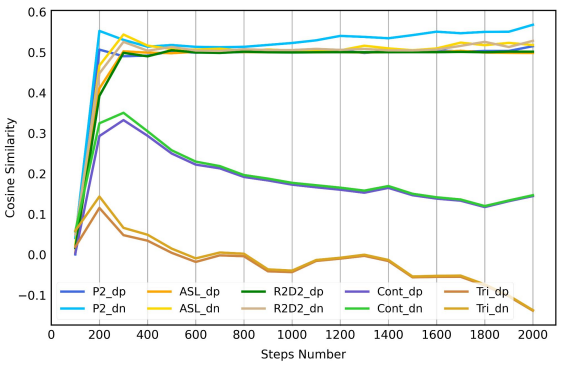

Figure 2: The positive similarity d_p and the most negative similarity d_n changes in different settings.

makes it harder to obtain correct matches. Such results also indicate that the detected keypoints are properly ranked, and the top points receive a higher probability to be matched. This is a desirable property that a reliable keypoint is expected to acquire. Similarly, if a loose threshold is set, only *Registration Recall* maintains the same level, while others show different improvements. This is because when calculating this metric, all possible matches are used, which is not related to the setting of threshold. Overall, our method shows consistent results with the change of keypoints number and threshold.

Qualitative examples. Here, we provide visualizations in Fig. 3 to present the sampled keypoints and a few matches from different scenes, under various camera motion status and conditions (e.g., motion blur), perceptual aliasing and textureless patterns in the room.

Similarity analysis For a better understanding of our proposed loss, we track the changes of the positive similarity d_p and the most negative similarity d_{n^*}=max(d_n) under various loss formulations and different networks. Here, d_p and d_n actually represent the average value of all s_p^i and all $(\max_j(s_n^{I_i^j}), \max_j(s_n^{P_i^j}))$ in Eq. 8, respectively. The similarity is calculated from the cosine distance that varies between $[-1, 1]$ (c.f. Sec. 3.2). For clarity, we use d_n to directly denote d_{n^*} in the following text. Here, we report the results of 1) P2-Net trained with our circle-guided descriptor loss (P2_d_p, P2_d_n); 2) P2-Net trained with the contrastive descriptor loss (Cont_d_p, Cont_d_n) [1]; 3) P2-Net trained with the triplet descriptor loss (Tri_d_p, Tri_d_n) [3, 6]; 4) P2-Net replaced with the ASL feature extractor (ASL_d_p, ASL_d_n) [6]; 4) P2-Net replaced with the R2D2 feature extractor (R2D2_d_p, R2D2_d_n) [6]. The last two networks are trained with our circle-guided descriptor loss. Other training or testing settings are kept the same.

As can be seen in Fig. 2, because pixel and point descriptors are heterogeneous, both d_p and d_n are extremely low (often around zero) in the initial phase. When P2-Net is trained with contrastive or triplet descriptor loss, Cont_d_p and Tri_d_p will quickly approach Cont_d_n and Tri_d_n. Under such cases, the network will lose the ability to distinguish d_p and d_n, resulting in ambiguous convergence. In the cases that only the 2D feature extractor is replaced, the loss minimization in network training will tend to synchronously increase the similarity for both positives and negatives, sacrificing the descriptor distinctiveness, which is consistent to the conclusion in Sec. 4.3. Moreover, because the design of our detector encourages higher relative scores for distinctive correspondence, such misleading distinctiveness from the descriptor cannot provide explicit and proper guidance for the detector at the beginning. In particular, our detector applies the hardest-in-batch sampling strategy in the global area, which further increases its dependence on distinctive descriptors. These also support the adoption of two-stage training discussed in Sec. 1.

	Volume	P2 [2D_Map]	P2 [3D_Map]
Floor 5a	38 m^3	72.5%	75.9%
Floor 5b	79 m^3	98.3%	99.3%
Average	-	85.4%	87.6%

Table 1: **Comparisons on selected scenes from 12Scenes [9].** Percentages of estimated camera poses falling within the threshold of ($5cm$, $5°$).

Quantitative results on visual localization. We also evaluate our framework with existing settings on sparser point clouds (3D projected depth maps) from 7Scenes, which only have 26,805 points on average for a single view at 640×480 resolution. It achieves 63.8% and 60.9% accuracy for P2[3D_Map] and P2[2D_Map], respectively, showing promise for handling sparser point clouds. However, we note that detailed evaluations on other types of point clouds, such as the ones obtained from LIDAR, might have different characteristics (e.g., much sparser), are a future area for further investigation.

Furthermore, we have tried our method on a more challenging 12Scenes dataset [9] with larger scale variation. As shown in Tab. 1, the scale of scenes varies from 38 m^3 to 79 m^3, significantly larger than 7Scenes (from 1 m^3 to 18 m^3). The promising results on 12Scenes also prove the capability of the proposed approach to handle more difficult datasets with larger levels of scale variation.

Significance of directly building 2D-3D matches. Our P2-Net enables directly localizing 2D images in a 3D model scanned by LIDAR or localizing 3D point clouds in a 2D Map, which reduces the restrictions for both sensors exploited for localization and types of pre-built maps. Thus, it is flexible and practical in real-world usages. Moreover, our approach would also be useful for interesting applications such as texture mapping (e.g., mapping pixels from a texture to a 3D surface) and robotic manipulation (e.g., matching images against 3D templates).

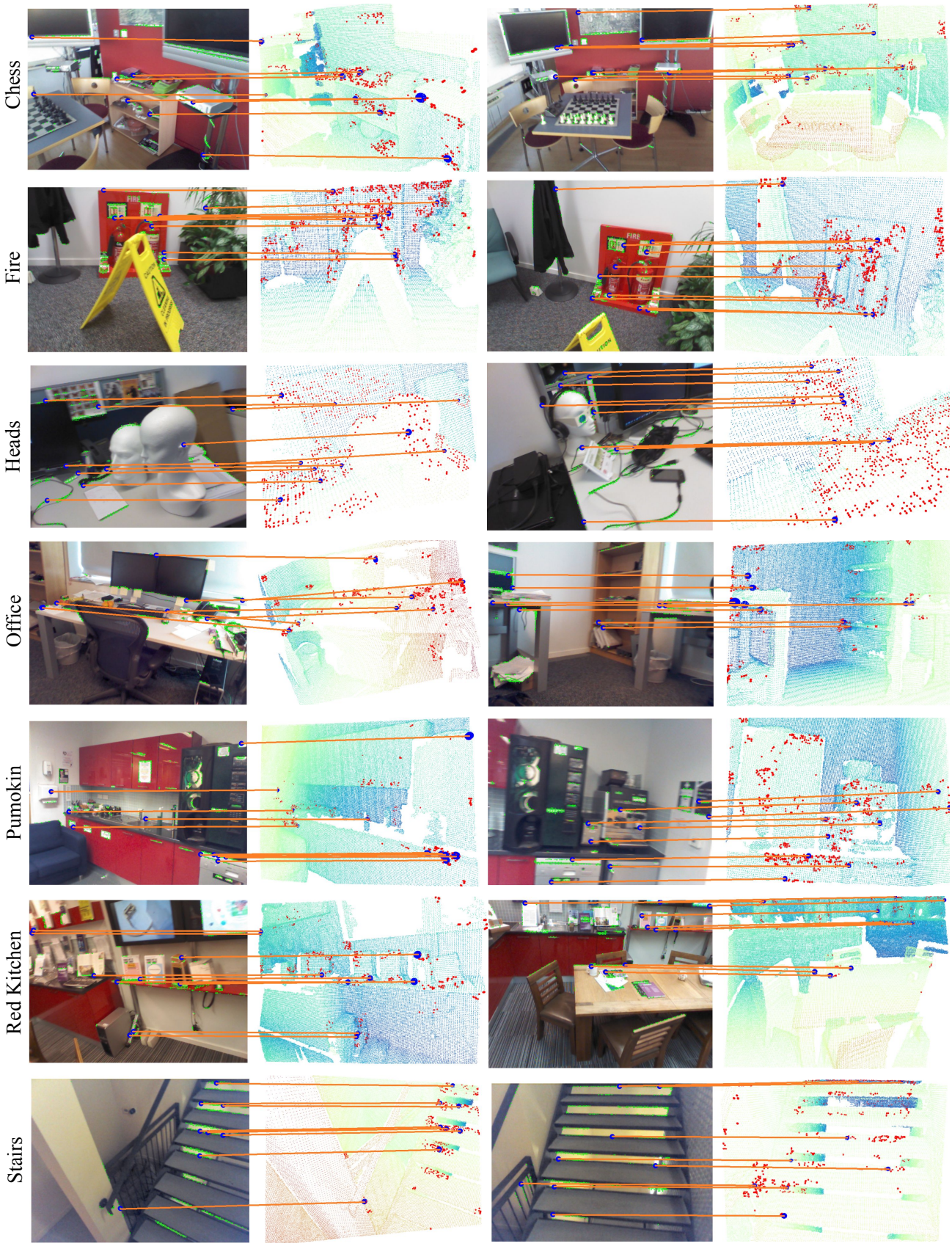

Figure 3: **Examples of good pixel and point matches from different scenes.**

References

[1] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. In *CVPR*, 2020.

[2] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, 2017.

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

[4] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time rgb-d camera relocalization. In *ISMAR*, 2013.

[5] Jiaxin Li and Gim Hee Lee. Usip: Unsupervised stable interest point detection from 3d point clouds. In *ICCV*, 2019.

[6] Zixin Luo, Lei Zhou, Xuyang Bai, Hongkai Chen, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Aslfeat: Learning local features of accurate shape and localization. In *CVPR*, 2020.

[7] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*, 2013.

[8] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019.

[9] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. In *3DV*, 2016.

[10] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. Sanet: Scene agnostic network for camera localization. In *ICCV*, 2019.

[11] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017.