Global-Aware Registration of Less-Overlap RGB-D Scans Supplementary Material

Che Sun, Yunde Jia, Yi Guo, and Yuwei Wu*
Beijing Laboratory of Intelligent Information Technology, School of Computer Science,
Beijing Institute of Technology, Beijing, 100081, China.
{sunche,jiayunde,guoyi,wuyuwei}@bit.edu.cn

1. Scene Inference Network

We construct the panorama $\mathbf{I}_{\mathrm{p}}$ via a scene inference network that includes two scan completion sub-networks and a panorama inference sub-network.

1.1. Network Structure

Scan Completion Sub-Network. The architecture of the scan completion sub-network is illustrated in Fig. 1. We follow the work of Yang *et al.* [4] to construct the sub-network and pre-process the scans to obtain their feature representations. The inputs are feature representations of two RGB-D scans $\mathbf{I}_1$ and $\mathbf{I}_2$, including specifying color, depth, and normal. Three independent encoders are first used to encode them into corresponding feature maps. The feature maps are then concatenated and fed into several convolutional and de-convolutional layers for encoding and decoding. Finally, five independent decoders are used to generate five feature representations of extrapolated scans.

Panorama Inference Sub-Network. The panorama inference sub-network has a similar encoder-decoder structure. The architecture of the encoder is the same as that of the scan completion sub-network (i.e., from $C1\_1, C1\_2, C1\_3$ to $C9$ in Fig. 1). Before decoding, two feature transformation modules are used to transform the extrapolated scans to the panorama at feature levels with extra inputs of transformation matrices $\mathcal{T}_1, \mathcal{T}_2 \in SE(3)$, which is shown in Fig. 2. The transformed features of two scans are concatenated at the channel dimension for decoding, obtaining the representation of the panorama in a decoder. The structure of the decoder is the same as that of the scan completion sub-network (i.e., from $D9$ to $D1\_1, D1\_2, D1\_3, D1\_4, D1\_5$ in Fig. 1).

1.2. Training

The scene inference network is trained via a supervised regression loss function. To this end, we need two input scans $\mathbf{I}_1, \mathbf{I}_2$, the ground truths of both the extrapolation scans $\mathbf{I}_1^*, \mathbf{I}_2^*$ and the panorama $\mathbf{I}_{\mathrm{p}}^*$, as well as the transformation matrices $\mathcal{T}_1, \mathcal{T}_2$ for training.

The representations of $\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_1^*$ and $\mathbf{I}_2^*$ are directly obtained from the datasets pre-processed by Yang *et al.* [4]. $\mathbf{I}_{\mathrm{p}}^*$ is rendered from the 3D scenes, where its pose (i.e., camera external matrix) is determined according to the poses of $\mathbf{I}_1$ and $\mathbf{I}_2$. We expect to align $\mathbf{I}_1$ and $\mathbf{I}_2$ with $\mathbf{I}_{\mathrm{p}}$ through symmetrical transformations (i.e., $\mathcal{T}_1^{-1} = \mathcal{T}_2$) for sufficient coverage. Specifically, we assume that the pose matrices of $\mathbf{I}_1$ and $\mathbf{I}_2$ are $\mathbf{P}_1^{\mathrm{W}}$ and $\mathbf{P}_2^{\mathrm{W}}$, respectively, at the world coordinate system. The matrices are

$$\mathbf{P}_1^{\mathrm{W}} = \begin{bmatrix} \mathbf{R}_1^{\mathrm{W}} & \mathbf{t}_1^{\mathrm{W}} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \mathbf{P}_2^{\mathrm{W}} = \begin{bmatrix} \mathbf{R}_2^{\mathrm{W}} & \mathbf{t}_2^{\mathrm{W}} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (1)$$

where $\mathbf{R}_1^{\mathrm{W}}, \mathbf{R}_2^{\mathrm{W}}$ are rotation matrices and $\mathbf{t}_1^{\mathrm{W}}, \mathbf{t}_2^{\mathrm{W}}$ are translation vectors at the world coordinate system. The pose matrix $\mathbf{P}_{\mathrm{p}}^{\mathrm{W}}$ of the panorama is computed by

$$\begin{aligned} &\mathbf{r}_1^{\mathrm{W}}, \mathbf{r}_2^{\mathrm{W}} = f_{\mathrm{m2q}}(\mathbf{R}_1^{\mathrm{W}}), f_{\mathrm{m2q}}(\mathbf{R}_2^{\mathrm{W}}), \\ &\mathbf{r}_{\mathrm{p}}^{\mathrm{W}} = (\mathbf{r}_1^{\mathrm{W}} + \mathbf{r}_2^{\mathrm{W}})/2, \\ &\mathbf{R}_{\mathrm{p}}^{\mathrm{W}} = f_{\mathrm{q2m}}(\mathbf{r}_{\mathrm{p}}^{\mathrm{W}}), \\ &\mathbf{t}_{\mathrm{p}}^{\mathrm{W}} = (\mathbf{R}_{\mathrm{p}}^{\mathrm{W}} + \mathbf{R}_1^{\mathrm{W}})((\mathbf{R}_{\mathrm{p}}^{\mathrm{W}})^{-1}\mathbf{t}_2^{\mathrm{W}} + (\mathbf{R}_1^{\mathrm{W}})^{-1}\mathbf{t}_1^{\mathrm{W}}), \\ &\mathbf{P}_{\mathrm{p}}^{\mathrm{W}} = \begin{bmatrix} \mathbf{R}_{\mathrm{p}}^{\mathrm{W}} & \mathbf{t}_{\mathrm{p}}^{\mathrm{W}} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \end{aligned} \quad (2)$$

where $f_{\mathrm{m2q}}(\cdot)$ denote converting rotation matrices $\mathbf{R}_1^{\mathrm{W}}$ and $\mathbf{R}_2^{\mathrm{W}}$ into rotation quaternion vectors $\mathbf{r}_1^{\mathrm{W}}$ and $\mathbf{r}_2^{\mathrm{W}}$, and $f_{\mathrm{q2m}}(\cdot)$ denote converting the vector $\mathbf{r}_{\mathrm{p}}^{\mathrm{W}}$ into the matrice $\mathbf{R}_{\mathrm{p}}^{\mathrm{W}}$. The ground-truth transformations matrices $\mathcal{T}_1, \mathcal{T}_2$ are computed by

$$\mathcal{T}_1 = \mathbf{P}_{\mathrm{p}}\mathbf{P}_1^{-1}, \ \ \mathcal{T}_2 = \mathbf{P}_{\mathrm{p}}\mathbf{P}_2^{-1}. \quad (3)$$

We train the scene inference network in an end-to-end manner, and the training batch is set to 4. We use the

*Corresponding author

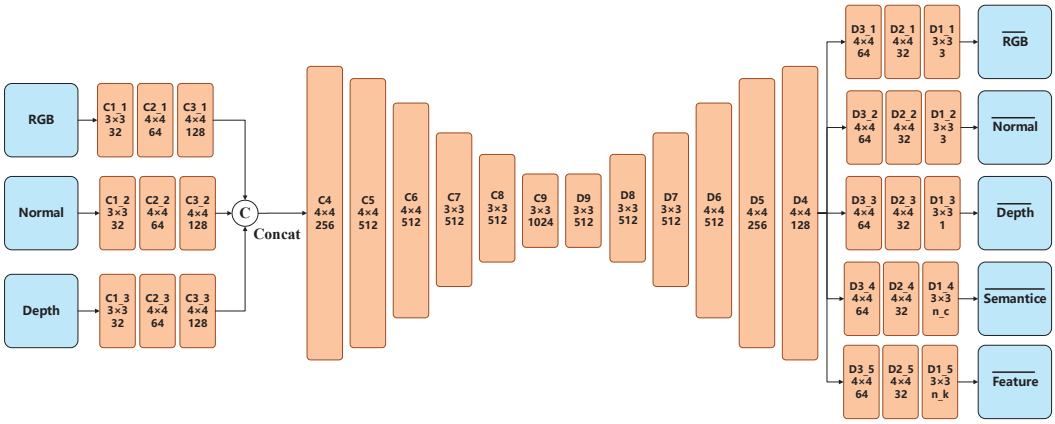

Figure 1. The architecture of the scan completion sub-network. In each block, the first line denotes the layer's name (e.g., $C1\_1$, $D1\_1$, etc.), where C and D indicate the convolutional layer and de-convolutional layer, respectively. The second line denotes the size of the convolutional kernels (e.g., 3×3 and 4×4), and the third line represents the number of kernels (e.g., 32, 64, etc.).

ADAM optimizer with a 0.001 learning rate for $200k$ iterations. During training, we add some noise to the transformation matrices $\mathcal{T}_1, \mathcal{T}_2$, and use them to transform the scans at feature levels.

2. Reinforcement Learning

2.1. Theoretical Analysis

The original objective function in Eq. (2) in the main submission is

$$\begin{aligned}\min_{\mathcal{T}_1,\mathcal{T}_2} & \sum_{\mathbf{m}_1 \in \mathcal{C}_1} \|\mathbf{I}_1(\mathbf{m}_1) - \mathbf{I}_{\mathrm{p}}^{(0)}(\mathbf{m}_{\mathrm{p}})\|_2^2 \\ & + \sum_{\mathbf{m}_2 \in \mathcal{C}_2} \|\mathbf{I}_2(\mathbf{m}_2) - \mathbf{I}_{\mathrm{p}}^{(0)}(\mathbf{m}_{\mathrm{p}})\|_2^2, \\ s.t., \ & [\mathrm{M}_{\mathrm{p}}; 1] = \mathcal{T}_1[\mathrm{M}_1; 1], [\mathrm{M}_{\mathrm{p}}; 1] = \mathcal{T}_2[\mathrm{M}_2; 1].\end{aligned} \tag{4}$$

Without considering uncertainty maps $\mathbf{U}$, the reward function in Eq. (4) in the main submission is

$$\begin{aligned} r_n = & \frac{1}{1+d_n}, \\ d_n = & \sum_{\mathbf{m}_1^{(n)} \in \mathcal{C}_1} \|\mathbf{F}_1^{(n)}(\mathbf{m}_1^{(n)}) - \mathbf{F}_{\mathrm{p}}^{(n)}(\mathbf{m}_1^{(n)})\|_2^2 \\ & + \sum_{\mathbf{m}_2^{(n)} \in \mathcal{C}_2} \|\mathbf{F}_2^{(n)}(\mathbf{m}_2^{(n)}) - \mathbf{F}_{\mathrm{p}}^{(n)}(\mathbf{m}_2^{(n)})\|_2^2, \end{aligned} \tag{5}$$

where the RGB values $\mathbf{I}_1(\cdot), \mathbf{I}_2(\cdot)$ and $\mathbf{I}_{\mathrm{p}}^{(0)}(\cdot)$ are replaced with the robust feature representations $\mathbf{F}_1^{(n)}(\cdot)$,$\mathbf{F}_2^{(n)}(\cdot)$ and $\mathbf{F}_{\mathrm{p}}^{(n)}(\cdot)$, respectively.

The supervised constraint in Eq. (7) and Eq. (8) in the main submission are

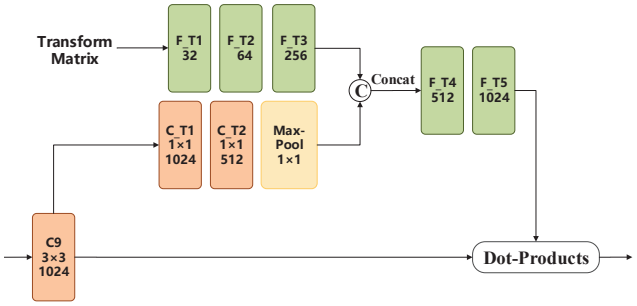

Figure 2. Illustration of the feature transformation. The transform matrix is converted into a vector (i.e., the concatenation of the rotation quaternion and translation vector), and is encoded by three fully-connected layers (F_T1, F_T2, F_T3). The feature maps of the $C9$ layer are also encoded by two convolutional layers (C_T1, C_T2) and one max-pooling layer ($MaxPool$). The two encoded features are concatenated to calculate the feature transformation via another two fully-connected layers (F_T4, F_T5). We perform dot-products between the feature transformation and all channel vectors of the feature maps of the $C9$ layer, generating transformed feature maps.

$$\begin{aligned} \mathcal{L}^{\mathrm{s}} = & \|(\mathbf{R}_1^{(n)})^{-1}(\mathbf{R}_1^{(n)})^* - \mathbf{1}\|_F^2 + \|\mathbf{t}_1^{(n)} - (\mathbf{t}_1^{(n)})^*\|_2^2 \\ & + \|(\mathbf{R}_2^{(n)})^{-1}(\mathbf{R}_2^{(n)})^* - \mathbf{1}\|_F^2 + \|\mathbf{t}_2^{(n)} - (\mathbf{t}_2^{(n)})^*\|_2^2, \end{aligned} \tag{6}$$

where,

$$\begin{aligned} \begin{bmatrix} (\mathbf{R}_1^{(n)})^* & (\mathbf{t}_1^{(n)})^* \\ \mathbf{0}^\top & 1 \end{bmatrix} &= (\mathcal{T}_1)^* \left(\prod_{i=1}^{n-1} \mathcal{T}_1^{(n-i)} \right)^{-1}, \\ \begin{bmatrix} (\mathbf{R}_2^{(n)})^* & (\mathbf{t}_2^{(n)})^* \\ \mathbf{0}^\top & 1 \end{bmatrix} &= (\mathcal{T}_2)^* \left(\prod_{i=1}^{n-1} \mathcal{T}_2^{(n-i)} \right)^{-1}. \end{aligned} \tag{7}$$

Theorem 1 *When the optimal outputs $\mathcal{T}_1^*, \mathcal{T}_2^*$ of the reinforcement learning satisfy both the supervised constraint in Eq. (6) and the maximum expected rewards in Eq. (5), they are exactly the solutions in Eq. (4)*[1].

Proof All the sequential actions are used to construct the optimal transformation matrices

$$\mathcal{T}_1^* = \prod_{i=1}^{n-1} \mathcal{T}_1^{(n-i)}, \ \mathcal{T}_2^* = \prod_{i=1}^{n-1} \mathcal{T}_2^{(n-i)}. \tag{8}$$

We use the actions to transform scans $\mathbf{I}_1$ and $\mathbf{I}_2$ into $\mathbf{I}_1^{(n)}$ and $\mathbf{I}_2^{(n)}$ at the n-th step, respectively, and the 3D coordinates are obtained by

$$[\mathrm{M}_1^{(n)}; 1] = \mathcal{T}_1^*[\mathrm{M}_1; 1], \ [\mathrm{M}_2^{(n)}; 1] = \mathcal{T}_1^*[\mathrm{M}_2; 1]. \tag{9}$$

The transformed scans $\mathbf{I}_1^{(n)}$ and $\mathbf{I}_2^{(n)}$ satisfy

$$\mathbf{F}_1^{(n)}(\mathbf{m}_1^{(n)}) = \mathbf{F}_1(\mathbf{m}_1), \ \mathbf{F}_2^{(n)}(\mathbf{m}_2^{(n)}) = \mathbf{F}_2(\mathbf{m}_2). \tag{10}$$

The supervised constraint in Eq. (6) enforces the scans $\mathbf{I}_1$ and $\mathbf{I}_2$ to align with the panorama $\mathbf{I}_\mathrm{p}$ at the n-th step. The optimal outputs mean the complete alignments, so the 3D/2D coordinates at the n-th step satisfy

$$\mathrm{M}_1^{(n)} = \mathrm{M}_2^{(n)} = \mathrm{M}_\mathrm{p}^{(n)}, \mathbf{m}_1^{(n)} = \mathbf{m}_2^{(n)} = \mathbf{m}_\mathrm{p}^{(n)}. \tag{11}$$

We assume that the position of the panorama is fixed during the reinforcement learning, so the coordinates are also fixed:

$$\begin{aligned} \mathrm{M}_1^{(n)} = \mathrm{M}_2^{(n)} = \mathrm{M}_\mathrm{p}^{(n)} = \mathrm{M}_\mathrm{p}, \\ \mathbf{m}_1^{(n)} = \mathbf{m}_2^{(n)} = \mathbf{m}_\mathrm{p}^{(n)} = \mathbf{m}_\mathrm{p}. \end{aligned} \tag{12}$$

Therefore, the following feature representations are the same:

$$\mathbf{F}_\mathrm{p}^{(n)}(\mathbf{m}_1^{(n)}) = \mathbf{F}_\mathrm{p}^{(n)}(\mathbf{m}_2^{(n)}) = \mathbf{F}_\mathrm{p}^{(n)}(\mathbf{m}_\mathrm{p}^{(n)}) = \mathbf{F}_\mathrm{p}^{(n)}(\mathbf{m}_\mathrm{p}). \tag{13}$$

Based on Eq. (9) and Eq. (12) , we obtain the same constraint in the original objective function in Eq. (4)

$$[\mathrm{M}_\mathrm{p}; 1] = \mathcal{T}_1^*[\mathrm{M}_1; 1], [\mathrm{M}_\mathrm{p}; 1] = \mathcal{T}_2^*[\mathrm{M}_2; 1]. \tag{14}$$

We substitute Eq. (13) and Eq. (10) into the reward function in Eq. (5):

$$\begin{aligned} d_n = & \sum_{\mathbf{m}_1 \in \mathcal{C}_1} \|\mathbf{F}_1(\mathbf{m}_1) - \mathbf{F}_\mathrm{p}^{(n)}(\mathbf{m}_\mathrm{p})\|_2^2 \\ & + \sum_{\mathbf{m}_2 \in \mathcal{C}_2} \|\mathbf{F}_2(\mathbf{m}_2) - \mathbf{F}_\mathrm{p}^{(n)}(\mathbf{m}_\mathrm{p})\|_2^2. \end{aligned} \tag{15}$$

[1]Both the uncertainty maps $\mathbf{U}$ and the panorama refinement are not considered in the theorem. $\mathbf{I}_\mathrm{p}^{(0)}$ and $\mathbf{I}_\mathrm{p}^{(n)}$ are treated equally.

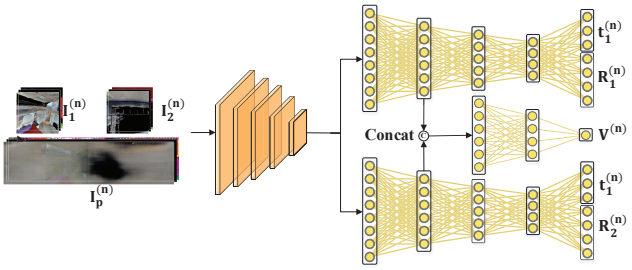

Figure 3. Illustration of the policy network f_π.

Maximizing the reward means minimizing the distance d_N in Eq. (15). This is equivalent to minimizing the error in the original objective function in Eq. (4) without considering the refinement of the panorama (i.e., $\mathbf{I}_\mathrm{p}^{(0)} = \mathbf{I}_\mathrm{p}^{(n)}$). Therefore, the sequential actions $\mathcal{T}_1^*, \mathcal{T}_2^*$ are also the optimal transformation matrices in Eq. (4).

□

2.2. Policy Network

We use a policy network f_π with a pre-trained embedding network e_ψ as the backbone to predict the action. We follow the work of Wang *et al.* [3] to construct the embedding network e_ψ by using a Siamese DGCNN to generate point embeddings. The embeddings are fed into a cascaded two-branch sub-network to predict distributions of the disentangled rotation $p(\mathbf{R}_1^{(n)}|s_n)$ and $p(\mathbf{R}_2^{(n)}|s_n)$ as well as the translation $p(\mathbf{t}_1^{(n)}|s_n)$ and $p(\mathbf{t}_2^{(n)}|s_n)$, as is shown in Fig. 3. A value approximator for optimization is also produced from the two concatenated branch representations, which shares the same parameters. The architecture of the sub-network for computing $p(\mathbf{R}_1^{(n)}|s_n)$ and $p(\mathbf{R}_2^{(n)}|s_n)$ is FC(1024,256,ReLU)-FC(256,128,ReLU)-FC(128,128,ReLU)-FC(128,4,None). The architecture of the sub-network for computing $p(\mathbf{t}_1^{(n)}|s_n)$ and $p(\mathbf{t}_2^{(n)}|s_n)$ is FC(1024,256,ReLU)-FC(256,128,ReLU)-FC(128,128,ReLU)-FC(128,3,None). The architecture of the sub-network for computing the value is FC(512,128,ReLU)-FC(128,1,None). FC(a, b, f) means a fully-connected layer with a trainable weight matrix $\mathbf{W} \in \mathbb{R}^{a \times b}$ and an activation function f. Each layer is followed by a layer normalization except for the last layer.

2.3. Proximal Policy Optimization

The policy network f_π is pre-trained and fine-tuned in a supervised manner by using the loss functions in Eq. (6) and Eq.(7), respectively, in the main submission. Besides, we also employ the proximal policy optimization algorithm [2] to optimize f_π to acquire the maximum reward.

We compute the action probabilities $p_\pi(a_n|s_n) = \{p(\mathbf{R}_1^{(n)}|s_n), p(\mathbf{R}_2^{(n)}|s_n), p(\mathbf{t}_1^{(n)}|s_n), p(\mathbf{t}_2^{(n)}|s_n)\}$ and estimate the value $V_\pi(s_n)$ from the policy network f_π at each time step n. The self-transformation matrices are samples from $p_\pi(a_n|s_n)$. We use $p_\pi(a_n|s_n)$ to represent the current output distributions, and use $p_{\pi_{\text{old}}}(a_n|s_n)$ to represent the output distributions when a_n was sampled. The expected cumulative reward can be used to construct the value function $V(s_n) = \mathbb{E}_{a_n, s_{n+1} \cdots}[\sum_{j=1}^{\infty} \gamma_j r_j]$, and we use $V_\pi(s_n)$ to approximate the value. The clipped surrogate objective is

$$\mathcal{L}^{\text{c}} = \hat{\mathbb{E}}\left[\min\left(\xi_n \hat{A}_n, \text{clip}(\xi_n, 1-\epsilon, 1+\epsilon)\hat{A}_n\right)\right], \quad (16)$$

where

$$\xi_n = \frac{p_\pi(a_n|s_n)}{p_{\pi_{\text{old}}}(a_n|s_n)}. \quad (17)$$

$\hat{A}_n$ is the advantage estimator:

$$\begin{aligned}\hat{A}_n = &- V(s_n) + r_n + \gamma r_{n+1} + \cdots \\ &+ \gamma^{N-n+1} r_{N-1} + \gamma^{N-n} V(s_n).\end{aligned} \quad (18)$$

The value loss function is

$$\mathcal{L}^{\text{v}} = \left(V_\pi(s_n) - V_{\text{targ}}\right)^2, \quad (19)$$

where

$$V_{\text{targ}} = r_n + \gamma r_{n+1} + \cdots + \gamma^{N-n} V_{\pi_{\text{old}}}(s_n). \quad (20)$$

The entropy regularization term for exploration is

$$\mathcal{L}^{\text{e}} = p_\pi \log p_\pi. \quad (21)$$

The on-policy update loss of the PPO algorithm is

$$\mathcal{L}^{\text{r}} = \mathcal{L}^{\text{c}} + \lambda^{\text{v}} \mathcal{L}^{\text{v}} + \lambda^{\text{e}} \mathcal{L}^{\text{e}}. \quad (22)$$

3. Additional Evaluations

3.1. Transformation Matrices

Besides the evaluation of transformation errors, we also calculate the success rate to evaluate our method on estimating the transformation matrices between less-overlap ($\leq 10\%$) RGB-D scans. If the rotation error or translation error of a transformed scan is below a certain threshold, we treat the transformation as a success. Tab. 1 shows the success rates under the thresholds of $3°$, $10°$ and $45°$ rotation errors as well as $0.1m$, $0.25m$ and $0.5m$ translation errors, respectively. The success rates demonstrate although our method performs better than the state-of-the-art methods on aligning less-overlap RGB-D scans, this task still needs to be studied for further improvements.

	Rotation (%)			Trans. (%)		
	3°	10°	45°	$0.1m$	$0.25m$	$0.5m$
ScanComp. [4]	0.9	8.8	45.3	0.4	2.3	18.7
HybridRep. [5]	1.1	9.5	50.4	**0.9**	3.0	22.6
Ours	**1.2**	**13.2**	**77.1**	0.8	**4.3**	**29.5**

Table 1. The success rates with different rotation and translation thresholds on the ScanNet dataset.

	True-Positve Rate (%)			Recall (%)		
	top-30	top-50	top-100	top-30	top-50	top-100
ScanComp. [4]	42.6	42.2	41.8	18.4	33.8	57.0
HybridRep. [5]	47.9	47.8	46.9	16.7	35.0	60.9
Ours	**65.9**	**65.6**	**65.0**	**24.5**	**40.6**	**80.8**

Table 2. Comparisons of the true-positive rate and recall of correspondences on the ScanNet dataset.

	Top-50 Precision (%)			Top-100 Precision (%)		
	5%	10%	15%	5%	10%	15%
ScanComp. [4]	6.6	15.3	25.8	7.7	17.2	31.3
HybridRep. [5]	7.0	18.2	24.5	8.5	20.6	33.4
Ours	**14.3**	**30.5**	**47.4**	**14.1**	**30.0**	**45.5**

Table 3. The top-50 and top-100 precision rates on the ScanNet dataset. The threshold is set to $5\%, 10\%, 15\%$ of the scene size that is defined as the maximum distance between 3D positions.

3.2. Point Correspondences

The true-positive rate and recall of point correspondences on the ScanNet dataset are shown in Tab. 2, where two scans have less than 10% overlap regions. We further compare the precision rate by using different thresholds instead of using the fixed threshold proposed in the main submission. The top-50 and top-100 precision rates are shown in Tab. 3. The threshold is set to $5\%, 10\%,$ and 15% of the scene size that is the maximum distance among all 3D points in the scene.

3.3. Visualization Results

Fig. 4 exhibits several examples of Mellado *et al.* [1], Yang *et al.* [4], and Yang *et al.* [5] for estimating transformation matrices, where the point clouds of the first scans (green) are transformed into the second scans (red) by using the transformation matrices $\mathcal{T}$.

We also visualize correspondences in Fig. 5. The threshold is set to $0.5m$, and green lines indicate correct correspondences and red lines denote incorrect ones. The correspondences are established on the noisy extrapolated scans, but here we show them on the ground-truth scans for better visualization.

Method	Speed (pps)	Rotation(°)	Trans.(m)
ScanComp. [4]	0.56	78.95	1.60
HybridRep. [5]	**2.44**	44.91	1.00
Ours	1.32	**33.73**	**0.61**

Table 4. Comparisons of the speed and relative errors on the ScanNet dataset. "pps" means pairs per second.

3.4. Analysis of Markov Reward Process

As is shown in Tab. 4, we compare the inference speed of our method with existing state-of-the-art methods [4, 5]. The inference speed is measured using a single NVIDIA RTX2080Ti GPU and an Intel i7-7800X CPU. We find that the speed of our method (1.32pps) is slightly slower than that (2.44pps) of [5], and faster than that (0.56pps) of [4]. Our iterative matching method takes more time, but is more efficient for alignments. A remained problem is that none of the existing methods (including ours) can achieve the real-time matching, which directs our future work towards real-time performances.

References

[1] Nicolas Mellado, Dror Aiger, and N. Mitra. Super 4pcs fast global pointcloud registration via smart indexing. *Computer Graphics Forum*, 33(5):205–215, 2014. 4, 6

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. 3

[3] Yue Wang, Yongbin Sun, Z. Liu, S. Sarma, M. Bronstein, and J. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38:1 – 12, 2019. 3

[4] Zhenpei Yang, Jeffrey Z Pan, Linjie Luo, Xiaowei Zhou, Kristen Grauman, and Qixing Huang. Extreme relative pose estimation for rgb-d scans via scene completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4531–4540, 2019. 1, 4, 5, 6, 7

[5] Zhenpei Yang, Siming Yan, and Qixing Huang. Extreme relative pose network under hybrid representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2455–2464, 2020. 4, 5, 6, 7

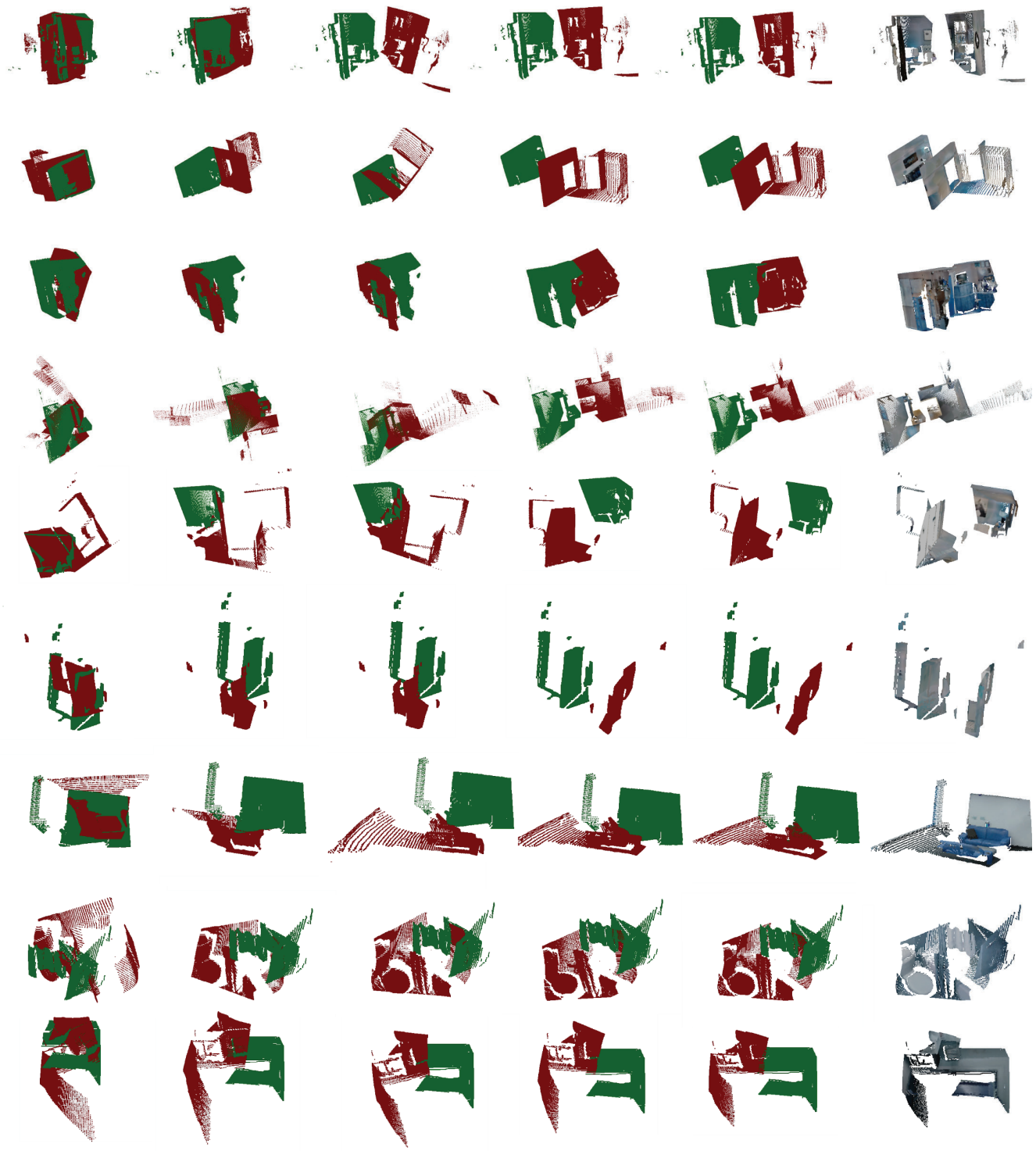

Figure 4. Qualitative results of Mellado *et al.* [1], Yang *et al.* [4], Yang *et al.* [5] and ours.

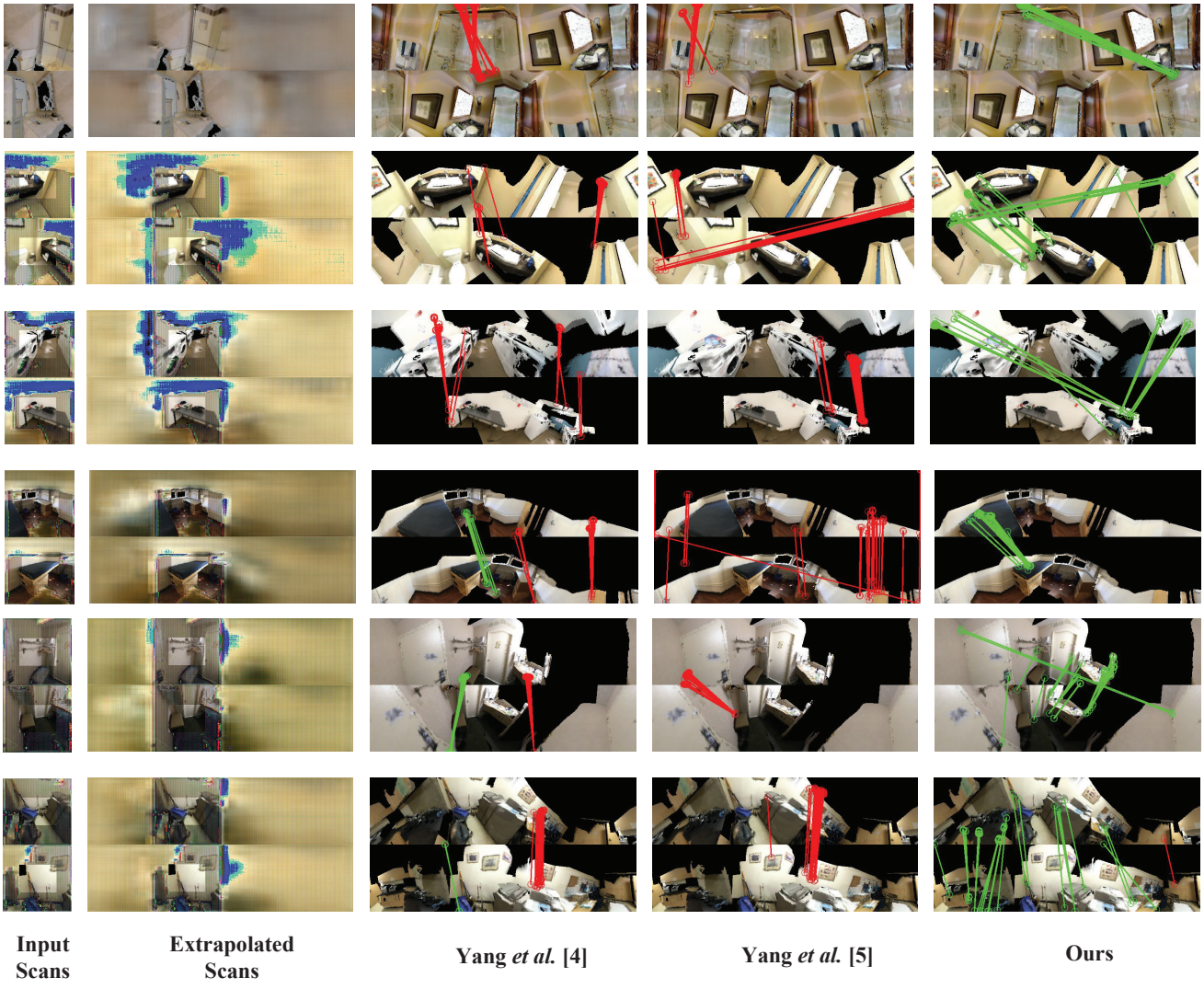

Figure 5. Visualization results of our method and baseline methods. From left to right, we show the input RGB-D scans, the extrapolated scans, as well as the correspondence results of Yang *et al.* [4], Yang *et al.* [5] and ours.