Learning multiview 3D point cloud registration – Supplementary material

Zan Gojcic*[§] Caifa Zhou*[§] Jan D. Wegner[§] Leonidas J. Guibas[†] Tolga Birdal[†]

[§]ETH Zurich [†]Stanford University

In this supplementary material, we provide additional information about the proposed algorithm (Sec. 1-2 and Alg. 1), network architectures and training configurations (Sec. 3), an extended ablation study (Sec. 5) as well as additional visualizations (Sec. 6). The source code and pretrained models are publicly available under https://github.com/zgojcic/3D_multiview_reg.

1. Closed-form solution of Eq. 4.

For the sake of completeness we summarize the closed-form differentiable solution of the weighted least square pairwise registration problem

$$\hat{\mathbf{R}}_{ij}, \hat{\mathbf{t}}_{ij} = \underset{\mathbf{R}_{ij},\mathbf{t}_{ij}}{\arg\min} \sum_{l=1}^{N} w_l ||\mathbf{R}_{ij}\mathbf{p}_l + \mathbf{t}_{ij} - \mathbf{q}_l)||^2. \quad (1)$$

Let $\overline{\mathbf{p}}$ and $\overline{\mathbf{q}}$

$$\overline{\mathbf{p}} := \frac{\sum_{l=1}^{N_{\mathbf{P}}} w_l \mathbf{p}_l}{\sum_{l=1}^{N_{\mathbf{P}}} w_l}, \quad \overline{\mathbf{q}} := \frac{\sum_{l=1}^{N_{\mathbf{Q}}} w_l \mathbf{q}_l}{\sum_{l=1}^{N_{\mathbf{Q}}} w_l} \quad (2)$$

denote weighted centroids of point clouds $\mathbf{P} \in \mathbb{R}^{N\times 3}$ and $\mathbf{Q} \in \mathbb{R}^{N\times 3}$, respectively. The centered point coordinates can then be computed as

$$\widetilde{\mathbf{p}}_l := \mathbf{p}_l - \overline{\mathbf{p}}, \quad \widetilde{\mathbf{q}}_l := \mathbf{q}_l - \overline{\mathbf{q}}, \quad l = 1,\dots,N \quad (3)$$

Arranging the centered points back to the matrix forms $\widetilde{\mathbf{P}} \in \mathbb{R}^{N\times 3}$ and $\widetilde{\mathbf{Q}} \in \mathbb{R}^{N\times 3}$, a weighted covariance matrix $\mathbf{S} \in \mathbb{R}^{3\times 3}$ can be computed as

$$\mathbf{S} = \widetilde{\mathbf{P}}^T \mathbf{W} \widetilde{\mathbf{Q}} \quad (4)$$

where $\mathbf{W} = \mathrm{diag}(w_1, \dots, w_N)$. Considering the singular value decomposition $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ the solution to Eq. 1 is given by

$$\hat{\mathbf{R}}_{ij} = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{V}\mathbf{U}^T) \end{bmatrix} \mathbf{U}^T \quad (5)$$

where $\det(\cdot)$ denotes computing the determinant and is used here to avoid creating a reflection matrix. Finally, $\hat{\mathbf{t}}_{ij}$ is computed as

$$\hat{\mathbf{t}}_{ij} = \overline{\mathbf{q}} - \hat{\mathbf{R}}_{ij}\overline{\mathbf{p}} \quad (6)$$

2. Closed-form solution of Eq. 5 and 6

In this section we summarize the closed form solutions to Eq. 5 and 6 from the main paper describing the rotation and translation synchronization, respectively.

The least squares formulation of the rotation synchronization problem

$$\mathbf{R}_i^* = \underset{\mathbf{R_i}\in SO(3)}{\arg\min} \sum_{(i,j)\in\mathcal{E}} c_{ij} ||\hat{\mathbf{R}}_{ij} - \mathbf{R}_i\mathbf{R}_j^T||_F^2 \quad (7)$$

admits a closed form solution under spectral relaxation as follows [1, 2]. Consider a symmetric matrix $\mathbf{L} \in \mathbb{R}^{3N_S\times 3N_S}$ resembling a block Laplacian matrix, defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \quad (8)$$

where $\mathbf{D}$ is the weighted degree matrix constructed as

$$\mathbf{D} = \begin{bmatrix} \mathbf{I}_3 \sum_i c_{i1} & & & \\ & \mathbf{I}_3 \sum_i c_{i2} & & \\ & & \ddots & \\ & & & \mathbf{I}_3 \sum_i c_{iN_S} \end{bmatrix} \quad (9)$$

*First two authors contributed equally to this work.

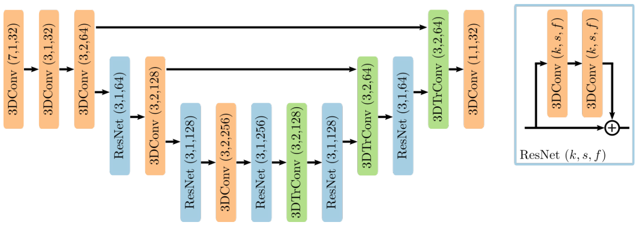


Figure 1. Network architecture of the FCGF [5] feature descriptor. Each convolutional layer (except the last one) is followed by batch normalization and ReLU activation function. The numbers in parentheses denote kernel size, stride, and the number of kernels, respectively.

and $\mathbf{A}$ is a block matrix of the relative rotations

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_3 & c_{12}\hat{\mathbf{R}}_{12} & \cdots & c_{1N_\mathcal{S}}\hat{\mathbf{R}}_{1N_\mathcal{S}} \\ c_{21}\hat{\mathbf{R}}_{21} & \mathbf{0}_3 & \cdots & c_{2N_\mathcal{S}}\hat{\mathbf{R}}_{2N_\mathcal{S}} \\ \vdots & & \ddots & \vdots \\ c_{N_\mathcal{S}1}\hat{\mathbf{R}}_{N_\mathcal{S}1} & c_{N_\mathcal{S}2}\hat{\mathbf{R}}_{N_\mathcal{S}2} & \cdots & \mathbf{0}_3 \end{bmatrix} \tag{10}$$

where the weights $c_{ij} := \zeta_{\text{init}}(\mathbf{\Gamma})$ represent the confidence in the relative transformation parameters $\hat{\mathbf{M}}_{ij}$ and $N_\mathcal{S}$ denotes the number of nodes in the graph. The least squares estimates of the global rotation matrices $\mathbf{R}_i^*$ are then given, under relaxed orthonormality and determinant constraints, by the three eigenvectors $\mathbf{v}_i \in \mathbb{R}^{3N_\mathcal{S}}$ corresponding to the smallest eigenvalues of $\mathbf{L}$. Consequently, the nearest rotation matrices under Frobenius norm can be obtained by a projection of the 3×3 submatrices of $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3] \in \mathbb{R}^{3N_\mathcal{S} \times 3}$ onto the orthonormal matrices and enforcing the determinant $\det(\mathbf{R}_i^*) = 1$ to avoid the reflections.

Similarly, the closed-form solution to the least squares formulation of the translation synchronization

$$\mathbf{t}_i^* = \arg\min_{\mathbf{t_i}} \sum_{(i,j)\in\mathcal{E}} c_{ij}||\hat{\mathbf{R}}_{ij}\mathbf{t}_i + \hat{\mathbf{t}}_{ij} - \mathbf{t}_j||^2 \tag{11}$$

can be written as [8]

$$\mathbf{t}^* = \mathbf{L}^+\mathbf{b} \tag{12}$$

where $\mathbf{t}^* = [\mathbf{t}_1^{*^T}, \ldots, \mathbf{t}_{N_\mathcal{S}}^{*^T}]^T \in \mathbb{R}^{3N_\mathcal{S}}$ and $\mathbf{b} = [\mathbf{b}_1^{*^T}, \ldots, \mathbf{b}_{N_\mathcal{S}}^{*^T}]^T \in \mathbb{R}^{3N_\mathcal{S}}$ with

$$\mathbf{b}_i := -\sum_{j\in\mathcal{N}(i)} c_{ij}\hat{\mathbf{R}}_{ij}^T\hat{\mathbf{t}}_{ij}. \tag{13}$$

where $\mathcal{N}(i)$ denotes all the neighboring vertices of $\mathbf{S}_i$ in graph $\mathcal{G}$.

3. Network architecture and training details

This section describes the network architecture as well as the training details of the FCGF [5] feature descriptor (Sec. 3.1) and the proposed registration block (Sec. 3.2). Both networks are implemented in Pytorch and pretrained using the *3DMatch* dataset [13].

3.1. FCGF local feature descriptor

Network architecture The FCGF [5] feature descriptor operates on sparse tensors that represent a point cloud in form of a set of unique coordinates $\mathbf{C}$ and their associated features $\mathbf{F}$

$$\mathbf{C} = \begin{bmatrix} x_1 & y_1 & z_1 & b_1 \\ \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & z_N & b_N \end{bmatrix}, \qquad \mathbf{F} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix} \tag{14}$$

where x_i, y_i, z_i are the coordinates of the i-th point in the point cloud and f_i is the associated feature (in our case simply 1). FCGF is implemented using the Minkowski Engine, an auto-differentiation library, which provides support for sparse convolutions and implements all essential deep learning layers [4]. We adopt the original, fully convolutional network design of FCGF that is depicted in Fig. 1. It has a UNet structure [11] and utilizes skip connections and ResNet blocks [7] to extract the per-point 32 dim feature descriptors. To obtain the unique coordinates $\mathbf{C}$, we use a GPU implementation of the voxel grid downsampling [4] with the voxel size $v := 2.5\,\text{cm}$.

Training details We again follow [5] and pre-train FCGF for 100 epochs using the point cloud fragments from the *3DMatch* dataset [13]. We optimize the parameters of the network using stohastic gradient descent with a batch size 4 and an initial learning rate of 0.1 combined with an exponential decay with $\gamma = 0.99$. To introduce rotation invariance of the descriptors we perform a data augmentation by randomly rotating each of the fragments along an arbitrary direction, by a different rotation, sampled from the $[0°, 360°)$ interval. The sampling of the positive and negative examples follows the procedure proposed in [5].

3.2. Registration block

Network architecture The architecture of the registration block (same for $\psi_{\text{init}}(\cdot)$ and $\psi_{\text{iter}}(\cdot)$)[2] follows [14] and is based on the PointNet-like architecture [10] where each of the fully connected layers ($\mathbf{P}$ in Fig. 2) operates on individual correspondences. The local context is then aggregated

[2]For $\psi_{\text{init}}(\cdot)$ the input dimension is increased from 6 to 8 (weights and residuals added).

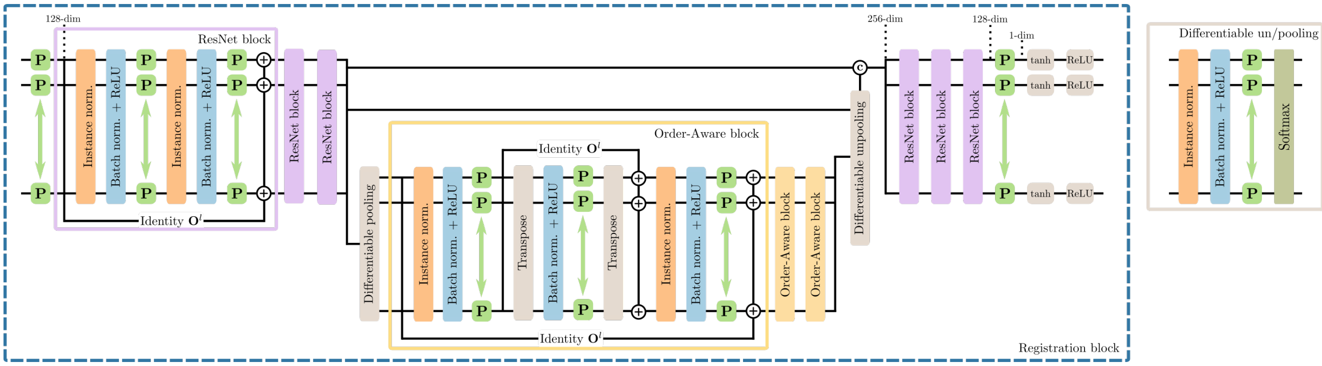


Figure 2. Network architecture of the registration block consists of two main modules: i) a PointNet-like ResNet block with instance normalization, and ii) an order-aware block. For each point cloud pair, putative correspondences are feed into three consecutive ResNet blocks followed by a differentiable pooling layer, which maps the N_c putative correspondences to M_c clusters $\mathbf{X}_{k+1}$ at the level $k+1$. These serve as input to the three order-aware blocks. Their output $\mathbf{X}'_{k+1}$ is fed along with $\mathbf{X}_k$ into the differentiable unpooling layer. The recovered features are then used as input to the remaining three ResNet blocks. The output of the registration block are the scores s_i indicating whether the putative correspondence is an outlier or an inlier. Additionally, the 128-dim features (denoted as $\mathbf{X}^{\text{conf}} := f_\theta^{-2}(\cdot)$) before the last perceptron layer $\mathbf{P}$ are used as input to the confidence estimation block.

using the instance normalization layers [12] defined as

$$\mathbf{y}_i^l = \frac{\mathbf{x}_i^l - \boldsymbol{\mu}^l}{\boldsymbol{\sigma}_l} \tag{15}$$

where $\mathbf{x}_i^l$ is the output of the layer l and $\boldsymbol{\mu}^l$ and $\boldsymbol{\sigma}_l$ are per dimension mean value and standard deviation, respectively. Opposed to the more commonly used batch normalization, instance normalization operates on individual training examples and not on the whole batch. Additionally, to reinforce the local context, the order-aware blocks [14] are used to map the correspondences to clusters using the learned soft pooling $\mathbf{S}_{\text{pool}} \in \mathbb{R}^{N_c \times M_c}$ and unpooling $\mathbf{S}_{\text{unpool}} \in \mathbb{R}^{N_c \times M_c}$ operators as

$$\mathbf{X}_{k+1} = \mathbf{S}_{\text{pool}}^T \mathbf{X}_k \quad \text{and} \quad \mathbf{X}'_k = \mathbf{S}_{\text{unpool}} \mathbf{X}'_{k+1} \tag{16}$$

where N_c is the number of correspondences and M_c is the number of clusters. $\mathbf{X}_k$ and $\mathbf{X}_{k+1}$ are the features at the level k (before clustering) and $k+1$ (after clustering), respectively (see Fig. 2). Finally, $\mathbf{X}'_{k+1}$ denotes the output of the last layer in the level $k+1$.

Training details We pre-train the registration blocks using the same fragments from the *3DMatch* dataset. Specifically, we first infer the FCGF descriptors and randomly sample $N_c = 5000$ descriptors per fragment. We use these descriptors to compute the putative correspondences for all fragment pairs (i, j) such that $i \leq j$. Based on the ground truth transformation parameters, we label these correspondences as inliers if the Euclidean distance between the points after the transformation is smaller than 7.5 cm. At the start of the training (first 15000 iterations) we supervise the learning using only the binary cross-entropy loss. Once a meaningful number of correspondences can already be classified correctly we add the transformation loss. We train the network for 500k iterations using Adam [9] optimizer with the initial learning rate of 0.001. We decay the learning rate every 1000 iterations by multiplying it with 0.999. To learn the rotation invariance we perform data augmentation, starting from the 25000th iteration, by randomly sampling an angle from the interval $[0°, n_a \cdot 20°)$ where n_a is initialized with zero and is then increased by 1 every 5000 iteration until the interval becomes $[0°, 360°)$.

4. Pseudo-code

Alg. 1 shows the pseudo-code of our proposed approach. We iterate $k = 4$ times over the network and transformation synchronization (i.e. *Transf-Sync*) layers and in each of those iterations we execute the *Transf-Sync* layer four times. Our implementation is constructed in a modular way (each part can be run on its own) and can accept a varying number of input point clouds with or without the connectivity information.

5. Extended ablation study

We extend the ablation study presented in the main paper, by analyzing the impact of edge pruning based on the local confidence (i.e. the output of the confidence estimation block) (Sec. 5.1) and of the weighting scheme (Sec. 5.2) on the angular and translation errors. The ablation study is performed on the point cloud fragments of the *ScanNet* dataset [6].

Algorithm 1 Pseudo-code of the proposed approach

Input: a set of potentially overlapping scans $\{\mathbf{S}_i\}_{i=1}^{N_\mathcal{S}}$
Output: globally optimized poses $\{\mathbf{M}_i^*\}_{i=1}^{N_\mathcal{S}}$
 # Compute the pairwise transformations
for *each pair of scans* $\mathbf{S}_i, \mathbf{S}_j \subset \mathcal{S}, i \neq j$ **do**
  # find the putative correspondences using $\phi(\cdot)$
  - $\mathbf{X}_{ij} = \text{cat}([\mathbf{S}_i, \phi(\mathbf{S}_i, \mathbf{S}_j)]) \in \mathbb{R}^{N_{\mathbf{S}_i} \times 6}$
  # compute the weights $\mathbf{w}_{ij} \in \mathbb{R}^{n_{\mathbf{S}_i}}$ using $\psi_{init}(\cdot)$
  - $\mathbf{w}_{ij} = \psi_{init}(\mathbf{X}_{ij}) \in \mathbb{R}^{N_{\mathbf{S}_i}}$
  - calculate $\mathbf{R}_{ij}, \mathbf{t}_{ij}$ using SVD according to (4)
 # Iterative network for transformation synchronization
$\mathbf{X}_{ij}^{(0)} \leftarrow \mathbf{X}_{ij}, \mathbf{w}_{ij}^{(0)} \leftarrow \mathbf{w}_{ij}, \mathbf{r}_{ij}^{(0)} \leftarrow \mathbf{r}_{ij}$
for $k = 1, 2, \cdots, \text{max\_iters}$ **do**
  for *each pairwise output from* ψ_{init} **do**
    - $\mathbf{R}_{ij}^{(k)}, \mathbf{t}_{ij}^{(k)}, \mathbf{w}_{ij}^{(k)} = \psi_{iter}([\mathbf{X}_{ij}^{(k-1)}, \mathbf{w}_{ij}^{(k-1)}, \mathbf{r}_{ij}^{(k-1)}])$
    - estimate *local*$\{c_{ij}^{(k)}\}$ using (16)
  - Gather the pairwise estimation as $\mathbf{R}^{(k)}, \mathbf{t}^{(k)}, \mathbf{c}^{(k)}$
  # Build the graph and perform the synchronization
  if $k = 1$ **then**
    - $\mathbf{c}^{(k)} :=$ *local*$\{\mathbf{c}^{(k)}\}$
  else
    - $\mathbf{c}^{(k)} := f_{HM}($*local*$\{\mathbf{c}^{(k)}\},$ *global*$\{\mathbf{c}^{(k-1)}\}$
  - $\mathbf{R}^{*(k)}, \mathbf{t}^{*(k)} = \text{Transf-Sync}(\mathbf{R}^{(k)}, \mathbf{t}^{(k)}, \mathbf{c}^{(k)})$
  # update step
  for *each pair of scans* $\mathbf{S}_i, \mathbf{S}_j \subset \mathcal{S}, i \neq j$ **do**
    - $\mathbf{X}_{ij}^{(k+1)} = \text{cat}([\mathbf{S}_i, \mathbf{M}_{ij}^{*(k)} \otimes \phi(\mathbf{S}_i, \mathbf{S}_j)]$
    - $\mathbf{w}_{ij}^{(k+1)} = \mathbf{w}_{ij}^{(k)}$
    - $\mathbf{r}_{ij}^{(k+1)} = \|\mathbf{S}_i - \mathbf{M}_{ij}^{*(k)} \otimes \phi(\mathbf{S}_i, \mathbf{S}_j)\|_2$

5.1. Impact of the edge pruning threshold

Results depicted in Fig. 3 show that the threshold value used for edge pruning has little impact on the angular and translation errors as long as it is larger than 0.2.

5.2. Impact of the harmonic mean weighting scheme

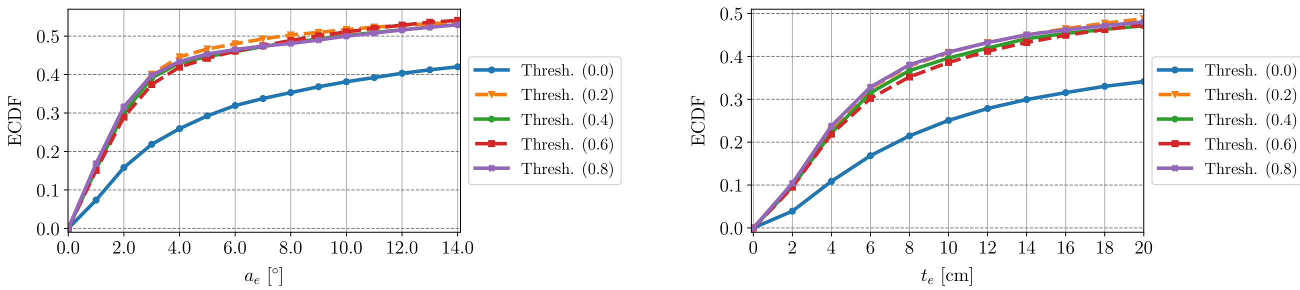
In this work, we have introduced a scheme for combining the local and global confidence using the harmonic mean (HM). In the following, we perform the analysis of this proposal and compare its performance to established methods based only on global information [3]. To this end, we again consider the scenario "Ours (Good)" as the input graph connectivity information. We compare the results of the proposed scheme (HM) to SE3 EIG [3], which proposes using the Cauchy function for computing the global edge confidence [3]. Note, we use the same pairwise transformation parameters, estimated using the method proposed herein, for all methods.

Without edge pruning It turns out that combining the local and global evidence about the graph connectivity is essential to achieve good performance. In fact, merely relying on local confidence estimates without HM weighting (denoted as ours; green) in Fig. 4) the *Transf-Sync* is unable to recover global transformations from the given graph connectivity evidence that is very noisy. Introducing the HM weighting scheme allows us to reduce the impact of noisy graph connectivity built solely using local confidence and can significantly improve performance after *Transf-Sync* block, which in turn enables us to outperform the *SE3 EIG*.

With edge pruning Fig. 5 shows that pruning the edges can help coping with noisy input graph connectivity built from the pairwise input. In principal, suppression of the edges with low confidence results in discarding the outliers that corrupt the l2 solution and as a result improves the performance of the *Transf-Sync* block.

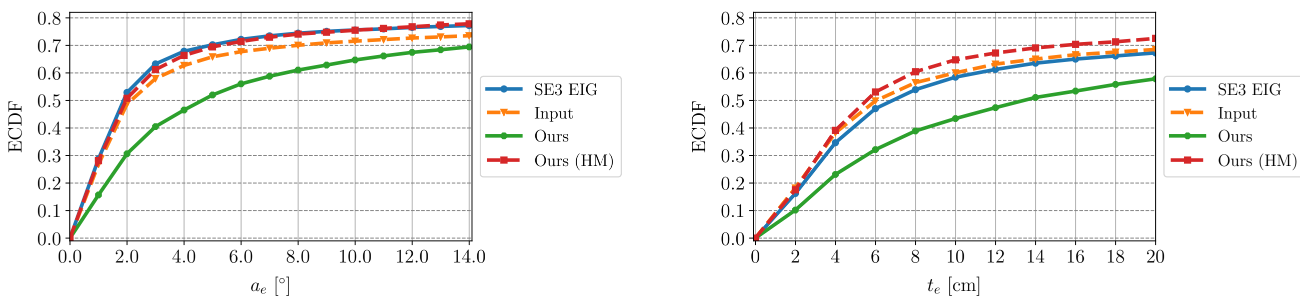
6. Qualitative results

We provide some additional qualitative results in form of success and failure cases on selected scenes of *3DMatch* (Fig. 6 and 7) and *ScanNet* (Fig. 8 and 9) datasets. Specifically, we compare the results of our whole pipeline *Ours (After Sync.)* to the results of *SE3 EIG* [3], pairwise registration results of our method from the first iteration *Ours* (1^{st} *iter.)*, and pairwise registration results of our method from the fourth iteration *Ours* (4^{th} *iter.)*. Both global methods (*Ours (After Sync.)* and *SE3 EIG*) use transformation parameters estimated by our proposed pairwise registration algorithm as input to the transformation synchronization. The failure cases of our method predominantly occur on point clouds with low level of structure (planar areas in Fig. 7 bottom) or high level of symmetry and repetitive structures (Fig. 9 top and bottom, respectively).



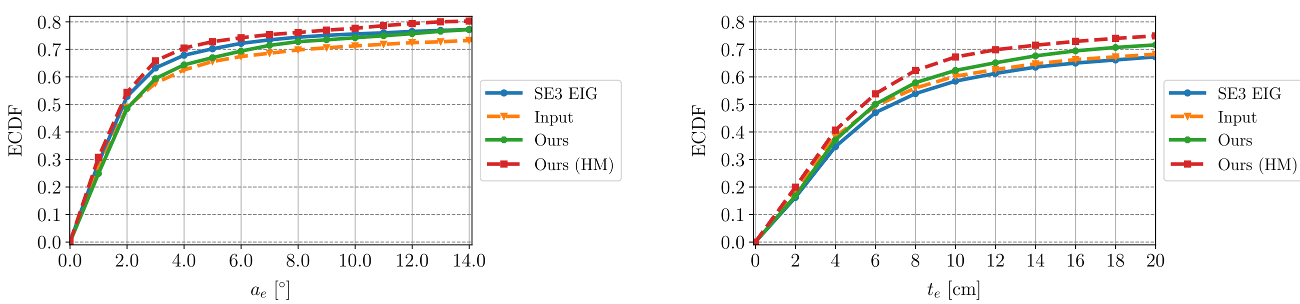
(a) (b)

Figure 3. Impact of the threshold value for edge pruning on the angular and translation errors. Results are obtained using the all pairs as input graph on *ScanNet* dataset [6]. (a) angular error and (b) translation error.



(a) (b)

Figure 4. Impact of the weighting scheme without edge cutting on the angular and translation errors. (a) angular and (b) translation errors.



(a) (b)

Figure 5. Impact of the weighting scheme combined with edge cutting, on the angular and translation errors. (a) angular error and (b) translation error.

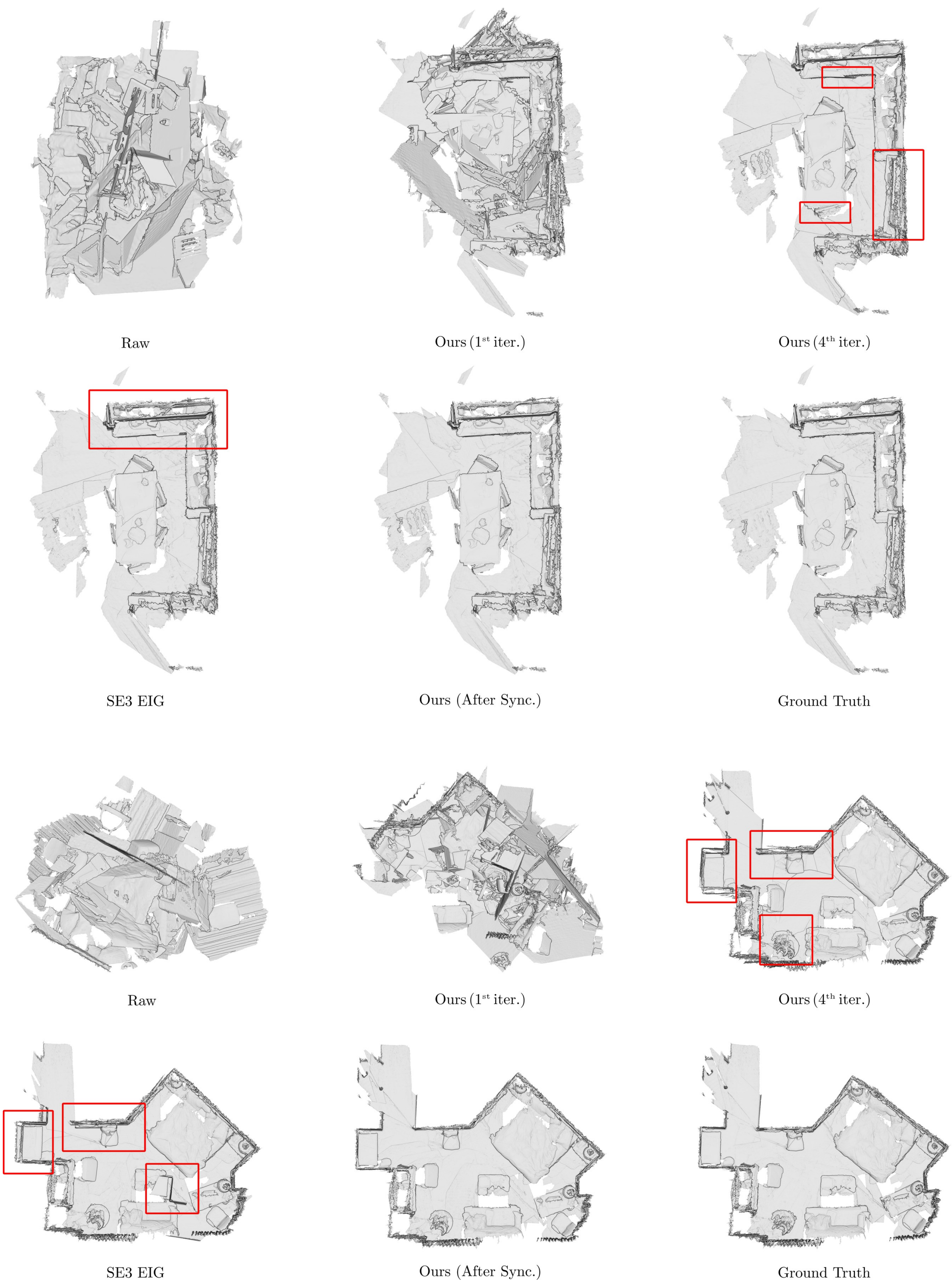


Figure 6. Selected **success cases** of our method on *3DMatch* dataset. Top: **Kitchen** and bottom: **Hotel 1**. Red rectangles highlight interesting areas with subtle changes.

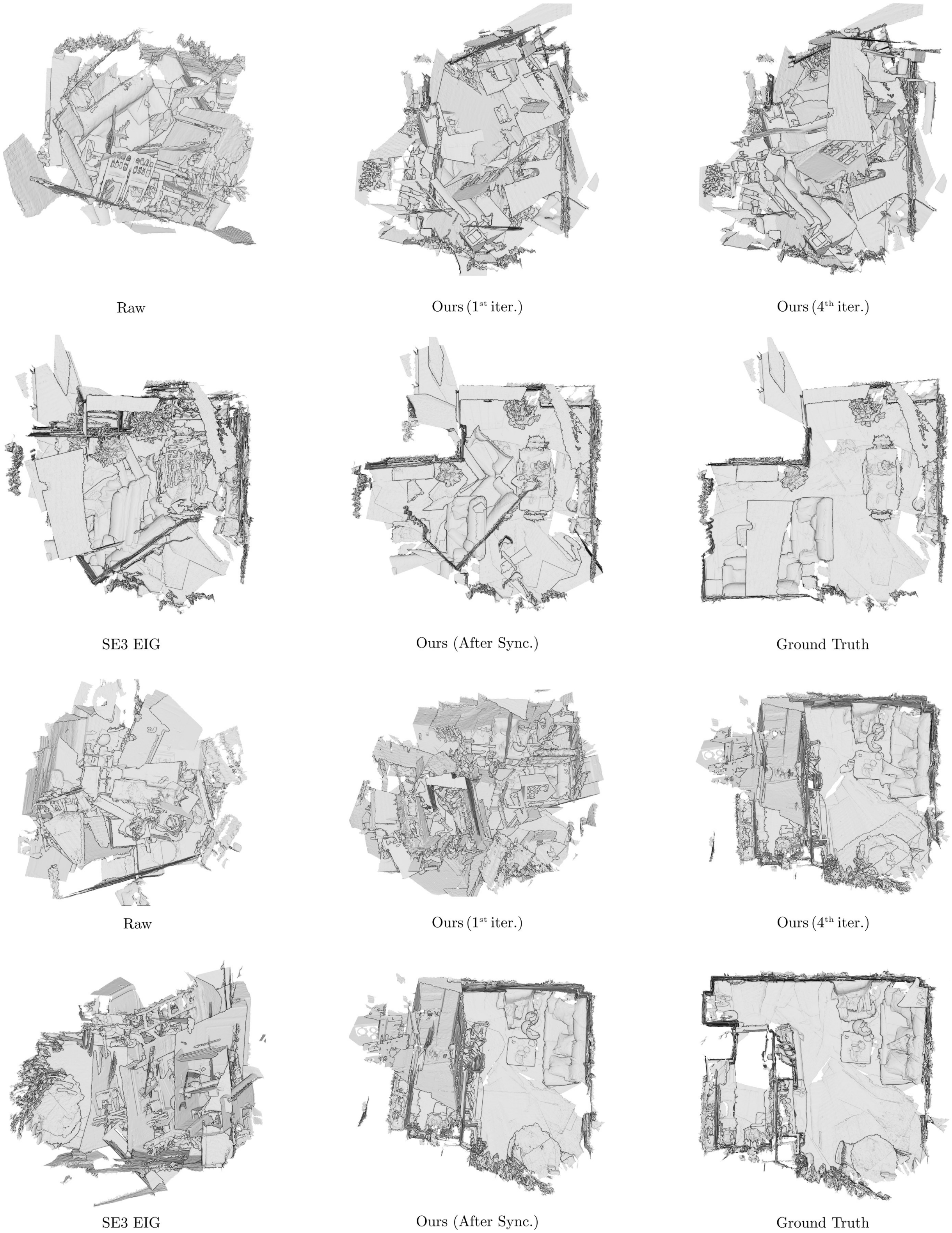


Figure 7. Selected **failure cases** of our method on *3DMatch* dataset. Top: **Home 1** and bottom: **Home 2**. Note that our method still provides qualitatively better results than state-of-the-art.

Figure 8. Selected **success cases** of our method on *ScanNet* dataset. Top: **scene0057_01** and bottom: **scene0309_00**. Red rectangles highlight interesting areas with subtle changes.

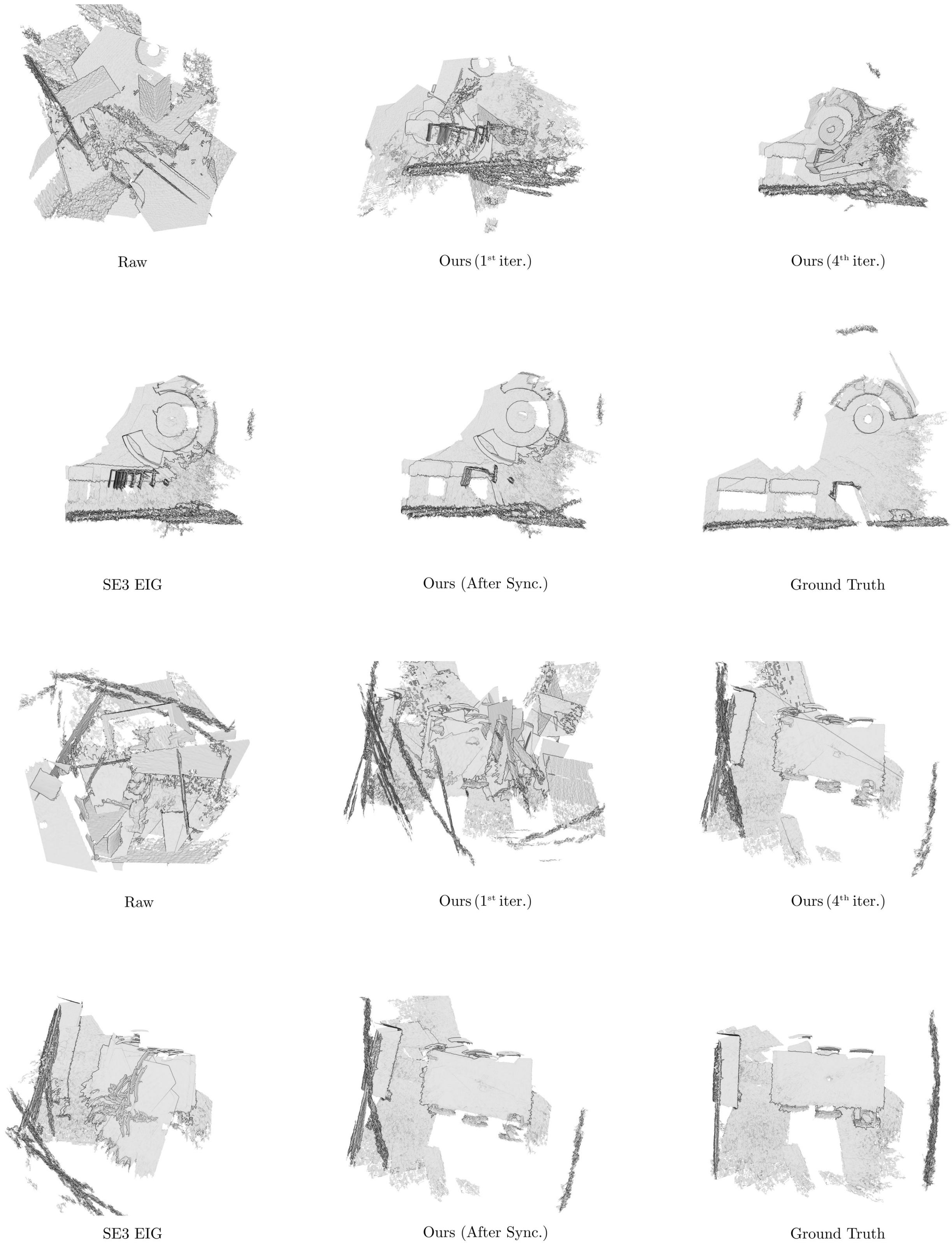


Figure 9. Selected **failure cases** of our method on *ScanNet* dataset. Top: **scene0334_02** and bottom: **scene0493_01**. Note that our method still provides qualitatively better results than state-of-the-art.

References

[1] Mica Arie-Nachimson, Shahar Z Kovalsky, Ira Kemelmacher-Shlizerman, Amit Singer, and Ronen Basri. Global motion estimation from point matches. In *International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 2012. 1

[2] Federica Arrigoni, Luca Magri, Beatrice Rossi, Pasqualina Fragneto, and Andrea Fusiello. Robust absolute rotation estimation via low-rank and sparse matrix decomposition. In *IEEE International Conference on 3D Vision (3DV)*, pages 491–498, 2014. 1

[3] Federica Arrigoni, Beatrice Rossi, and Andrea Fusiello. Spectral synchronization of multiple views in se(3). *SIAM Journal on Imaging Sciences*, 9(4):1963–1990, 2016. 4

[4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3075–3084, 2019. 2

[5] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 8958–8966, 2019. 2

[6] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 3, 5

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2

[8] Xiangru Huang, Zhenxiao Liang, Xiaowei Zhou, Yao Xie, Leonidas J Guibas, and Qixing Huang. Learning transformation synchronization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8082–8091, 2019. 2

[9] Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. In *International Conference on Learning Representations 2015*, 2015. 3

[10] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[12] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 3

[13] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[14] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. In *International Conference on Computer Vision (ICCV)*, 2019. 2, 3