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

\[
\hat{R}_{ij}, \hat{t}_{ij} = \arg \min_{R_{ij}, t_{ij}} \sum_{l=1}^{N} w_l ||R_{ij}p_l + t_{ij} - q_l||^2 \tag{1}
\]

Let \(p\) and \(q\)

\[
p := \frac{\sum_{l=1}^{N_P} w_l p_l}{\sum_{l=1}^{N_P} w_l}, \quad q := \frac{\sum_{l=1}^{N_Q} w_l q_l}{\sum_{l=1}^{N_Q} w_l} \tag{2}
\]

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

\[
\tilde{p}_l := p_l - p, \quad \tilde{q}_l := q_l - q, \quad l = 1, \ldots, N \tag{3}
\]

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

\[
S = \tilde{P}^T W \tilde{Q} \tag{4}
\]

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

\[
\hat{R}_{ij} = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \text{det}(VU^T) \end{bmatrix} U^T \tag{5}
\]

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

\[
t_{ij} = q_l - \hat{R}_{ij}p_l \tag{6}
\]

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

\[
\hat{R}_i = \arg \min_{R_i \in SO(3)} \sum_{(i,j) \in E} c_{ij} ||\hat{R}_{ij} - R_iR_j^T||^2_F \tag{7}
\]

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

\[
L = D - A \tag{8}
\]

where \(D\) is the weighted degree matrix constructed as

\[
D = \begin{bmatrix}
I_3 \sum_i c_{i1} & & \\
& I_3 \sum_i c_{i2} & \\
& & \ddots \\
& & & I_3 \sum_i c_{IN_S}
\end{bmatrix} \tag{9}
\]
and $A$ is a block matrix of the relative rotations

$$A = \begin{bmatrix}
0_3 & c_{12} \hat{R}_{12} & \cdots & c_{1N_S} \hat{R}_{1N_S} \\
c_{21} \hat{R}_{21} & 0_3 & \cdots & c_{2N_S} \hat{R}_{2N_S} \\
\vdots & \vdots & \ddots & \vdots \\
c_{N_S 1} \hat{R}_{N_S 1} & c_{N_S 2} \hat{R}_{N_S 2} & \cdots & 0_3
\end{bmatrix}$$

(10)

where the weights $c_{ij} := \zeta_{\text{ini}}(\Gamma)$ represent the confidence in the relative transformation parameters $M_{ij}$ and $N_S$ denotes the number of nodes in the graph. The least squares estimates of the global rotation matrices $\hat{R}^*_i$ are then given, under relaxed orthonormality and determinant constraints, by the three eigenvectors $v_i \in \mathbb{R}^{3N_S}$ corresponding to the smallest eigenvalues of $L$. Consequently, the nearest rotation matrices under Frobenius norm can be obtained by a projection of the $3 \times 3$ submatrices of $V = [v_1, v_2, v_3] \in \mathbb{R}^{3N_S \times 3}$ onto the orthonormal matrices and enforcing the determinant $\det(\hat{R}^*_i) = 1$ to avoid the reflections.

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

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

(11)

can be written as [8]

$$t^* = L^+ b$$

(12)

where $t^* = [t^T_1, \ldots, t^T_{N_S}]^T \in \mathbb{R}^{3N_S}$ and $b = [b^T_1, \ldots, b^T_{N_S}]^T \in \mathbb{R}^{3N_S}$ with

$$b_i := - \sum_{j \in N(i)} c_{ij} \hat{R}_{ij}^T \hat{t}_{ij}.$$  

(13)

where $N(i)$ denotes all the neighboring vertices of $S_i$ in graph $G$.

### 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 $C$ and their associated features $F$

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

(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 $C$, we use a GPU implementation of the voxel grid downsampling [4] with the voxel size $v := 2.5$ 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 stochastic 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 $[0^\circ, 360^\circ]$ 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 $\psi_{\text{fin}}(\cdot)$) is based on the PointNet-like architecture [10] where each of the fully connected layers ($P$ in Fig. 2) operates on individual correspondences. The local context is then aggregated

\footnote{For $\psi_{\text{fin}}(\cdot)$ the input dimension is increased from 6 to 8 (weights and residuals added).}
normalization, and ii) an order-aware block. For each point cloud pair, putative correspondences are fed into three consecutive ResNet blocks followed by a differentiable pooling layer, which maps the $N_c$ putative correspondences to $M_c$ clusters $X_{k+1}$ at the level $k+1$. These serve as input to the three order-aware blocks. Their output $X_{k+1}$ is fed along with $X_k$ 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 indicating whether the putative correspondence is an outlier or an inlier. Additionally, the 128-dim features (denoted as $X^{conf}$) before the last perceptron layer $P$ are used as input to the confidence estimation block.

The recovered features are then used as input to the remaining three ResNet blocks. The output of the registration block are the scores indicating whether the putative correspondence is an outlier or an inlier. Additionally, the 128-dim features (denoted as $X^{conf}$) before the last perceptron layer $P$ are used as input to the confidence estimation block.

using the instance normalization layers [12] defined as

$$y_i = \frac{x_i - \mu_i}{\sigma_i}$$  \hspace{1cm} (15)

where $x_i$ is the output of the layer $l$ and $\mu_i$ and $\sigma_i$ 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 $S_{pool} \in \mathbb{R}^{N_c \times M_c}$ and unpooling $S_{unpool} \in \mathbb{R}^{N_c \times M_c}$ operators as

$$X_{k+1} = S_{pool}X_k \quad \text{and} \quad X_k = S_{unpool}X_{k+1}$$  \hspace{1cm} (16)

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

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 $\{S_i\}_{i=1}^{N_S}$
Output: globally optimized poses $\{M^*_i\}_{i=1}^{N_S}$

# Compute the pairwise transformations
for each pair of scans $S_i, S_j \subset S, i \neq j$ do
  # find the putative correspondences using $\phi(\cdot)$
  $- X_{ij} = \text{cat}(\{S_i, \phi(S_i, S_j)\}) \in \mathbb{R}^{N_S \times 6}$
  # compute the weights $w_{ij} \in \mathbb{R}^{N_S}$ using $\psi_{\text{init}}(\cdot)$
  $- w_{ij} = \psi_{\text{init}}(X_{ij}) \in \mathbb{R}^{N_S}$
  - calculate $R_{ij}, t_{ij}$ using SVD according to (4)
  # Iterative network for transformation synchronization
  $X_{ij}^{(0)} \leftarrow X_{ij}, w_{ij}^{(0)} \leftarrow w_{ij}, r_{ij}^{(0)} \leftarrow r_{ij}$
  for $k = 1, 2, \ldots, \max\text{iters}$ do
    # Build the graph and perform the synchronization
    if $k = 1$ then
      - $c^{(k)} = \text{local}\{c^{(k)}\}$
    else
      - $c^{(k)} = \psi_{\text{sync}}(\text{local}\{c^{(k)}\}); \text{global}\{c^{(k-1)}\}$
      - $R^{(k)}, t^{(k)} = \text{Transf-Sync}(R^{(k)}, t^{(k)}, c^{(k)})$
    # update step
    for each pair of scans $S_i, S_j \subset S, i \neq j$ do
      - $X_{ij}^{(k+1)} = \text{cat}(\{S_i, M_{ij}^{(k)} \odot \phi(S_i, S_j)\})$
      - $w_{ij}^{(k+1)} = w_{ij}^{(k)}$
      - $r_{ij}^{(k+1)} = ||S_i - M_{ij}^{(k)} \odot \phi(S_i, S_j)||_2$

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 I2 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 (1st iter.), and pairwise registration results of our method from the fourth iteration Ours (4th 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 similarity and repetitive structures (Fig. 9 top and bottom, respectively).
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.

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

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


