Deep Global Registration

Christopher Choy*
Stanford University

Wei Dong*
Carnegie Mellon University

Vladlen Koltun
Intel Labs

Abstract

We present Deep Global Registration, a differentiable framework for pairwise registration of real-world 3D scans. Deep global registration is based on three modules: a 6-dimensional convolutional network for correspondence confidence prediction, a differentiable Weighted Procrustes algorithm for closed-form pose estimation, and a robust gradient-based SE(3) optimizer for pose refinement. Experiments demonstrate that our approach outperforms state-of-the-art methods, both learning-based and classical, on real-world data.

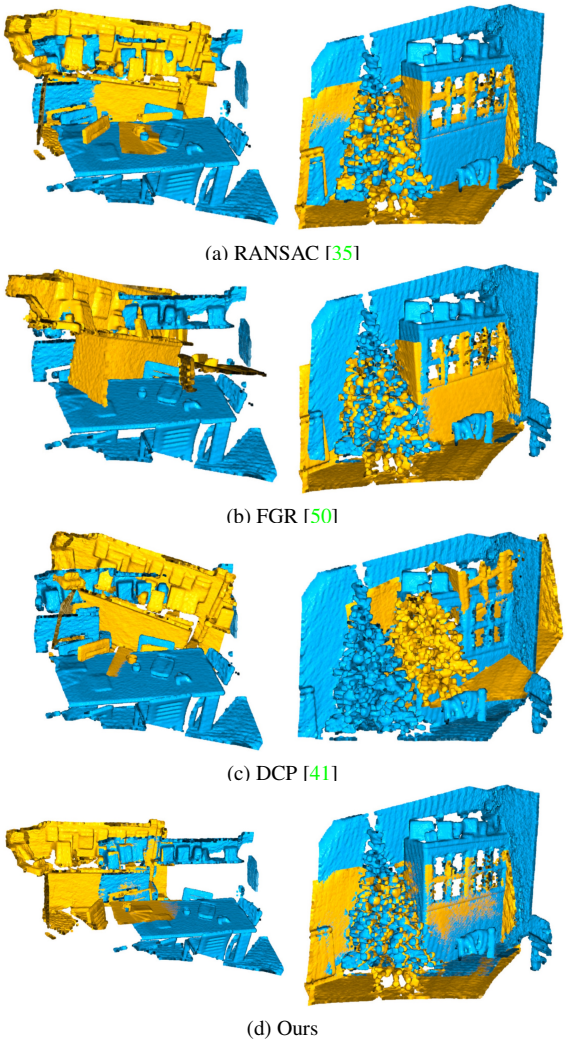
1. Introduction

A variety of applications, including 3D reconstruction, tracking, pose estimation, and object detection, invoke 3D registration as part of their operation [35, 5, 33]. To maximize the accuracy and speed of 3D registration, researchers have developed geometric feature descriptors [22, 11, 41, 9], pose optimization algorithms [38, 45, 24, 50], and end-to-end feature learning and registration pipelines [41, 2].

In particular, recent end-to-end registration networks have proven to be effective in relation to classical pipelines. However, these end-to-end approaches have some drawbacks that limit their accuracy and applicability. For example, PointNetLK [2] uses globally pooled features to encode the entire geometry of a point cloud, which decreases spatial acuity and registration accuracy. Deep closest point [41] makes strong assumptions on the distribution of points and correspondences, which do not hold for partially overlapping 3D scans.

In this work, we propose three modules for robust and accurate registration that resolve these drawbacks: a 6-dimensional convolutional network for correspondence confidence estimation, a differentiable Weighted Procrustes method for scalable registration, and a robust SE(3) optimizer for fine-tuning the final alignment.

The first component is a 6-dimensional convolutional network that analyzes the geometry of 3D correspondences and estimates their accuracy. Our approach is inspired by a number of learning-based methods for estimating the validity of correspondences in 2D [46, 34] and 3D [31]. These methods stack coordinates of correspondence pairs, forming a vector $[\mathbf{x}; \mathbf{y}] \in \mathbb{R}^{2\times D}$ for each correspondence



(a) RANSAC [35]

(b) FGR [50]

(c) DCP [41]

(d) Ours

Figure 1: Pairwise registration results on the 3DMatch dataset [47]. Our method successfully aligns a challenging 3D pair (*left*), while RANSAC [35], FGR [50], and DCP [41] fail. On an easier pair (*right*), our method achieves finer alignment.

*indicates equal contribution

$\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$. Prior methods treat these $2 \times D$-dimensional vectors as a set, and apply global set processing models for analysis. Such models largely disregard local geometric structure. Yet the correspondences are embedded in a metric space ($\mathbb{R}^{2\times D}$) that induces distances and neighborhood relationships. In particular, 3D correspondences form a geometric structure in 6-dimensional space [8] and we use a high-dimensional convolutional network to analyze the 6D structure formed by correspondences and estimate the likelihood that a given correspondence is correct (*i.e.*, an inlier).

The second component we develop is a differentiable Weighted Procrustes solver. The Procrustes method [15] provides a closed-form solution for rigid registration in $\mathrm{SE}(3)$. A differentiable version of the Procrustes method by Wang *et al.* [41] has been used for end-to-end registration. However, the differentiable Procrustes method passes gradients through coordinates, which requires $O(N^2)$ time and memory for N keypoints, limiting the number of keypoints that can be processed by the network. We use the inlier probabilities predicted by our first module (the 6D convolutional network) to guide the Procrustes method, thus forming a differentiable Weighted Procrustes method. This method passes gradients through the weights associated with correspondences rather than correspondence coordinates. The computational complexity of the Weighted Procrustes method is linear in the number of correspondences, allowing the registration pipeline to use dense correspondence sets rather than sparse keypoints. This substantially increases registration accuracy.

Our third component is a robust optimization module that fine-tunes the alignment produced by the Weighted Procrustes solver. This optimization module minimizes a differentiable loss via gradient descent on the continuous $\mathrm{SE}(3)$ representation space [52]. The optimization is fast since it does not require neighbor search in the inner loop [48].

Experimentally, we validate the presented modules on a real-world pairwise registration benchmark [47] and large-scale scene reconstruction datasets [18, 6, 32]. We show that our modules are robust, accurate, and fast in comparison to both classical global registration algorithms [50, 35, 45] and recent end-to-end approaches [31, 41, 2]. All training and experiment scripts are available at https://github.com/chrischoy/DeepGlobalRegistration.

2. Related Work

We divide the related work into three categories following the stages of standard registration pipelines that deal with real-world 3D scans: feature-based correspondence matching, outlier filtering, and pose optimization.

Feature-based correspondence matching. The first step in many 3D registration pipelines is feature extraction. Local and global geometric structure in 3D is analyzed to produce high-dimensional feature descriptors, which can then be used to establish correspondences.

Traditional hand-crafted features commonly summarize pairwise or higher-order relationships in histograms [20, 37, 40, 36, 35]. Recent work has shifted to learning features via deep networks [47, 22]. A number of recent methods are based on global pooling models [12, 11, 49], while others use convolutional networks [14, 9].

Our work is agnostic to the feature extraction mechanism. Our modules primarily address subsequent stages of the registration pipeline and are compatible with a wide variety of feature descriptors.

Outlier filtering. Correspondences produced by matching features are commonly heavily contaminated by outliers. These outliers need to be filtered out for robust alignment. A widely used family of techniques for robust model fitting is based on RANdom SAmple Consensus (RANSAC) [38, 1, 35, 29, 19], which iteratively samples small sets of correspondences in the hope of sampling a subset that is free from outliers. Other algorithms are based on branch-and-bound [45], semi-definite programming [28, 25], and maximal clique selection [44]. These methods are accurate, but commonly require longer iterative sampling or more expensive computation as the signal-to-noise ratio decreases. One exception is TEASER [44], which remains effective even with high outlier rates. Other methods use robust loss functions to reject outliers during optimization [50, 4].

Our work uses a convolutional network to identify inliers and outliers. The network needs only one feed-forward pass at test time and does not require iterative optimization.

Pose optimization. Pose optimization is the final stage that minimizes an alignment objective on filtered correspondences. Iterative Closest Points (ICP) [3] and Fast Global Registration (FGR) [50] use second-order optimization to optimize poses. Makadia *et al.* [26] propose an iterative procedure to minimize correlation scores. Maken *et al.* [27] propose to accelerate this process by stochastic gradient descent.

Recent end-to-end frameworks combine feature learning and pose optimization. Aoki *et al.* [2] combine PointNet global features with an iterative pose optimization method [24]. Wang *et al.* [41, 42] train graph neural network features by backpropagating through pose optimization.

We further advance this line of work. In particular, our Weighted Procrustes method reduces the complexity of optimization from quadratic to linear and enables the use of dense correspondences for highly accurate registration of real-world scans.

3. Deep Global Registration

3D reconstruction systems typically take a sequence of partial 3D scans as input and recover a complete 3D model of the scene. These partial scans are scene fragments, as shown in Fig. 1. In order to reconstruct the scene, reconstruction systems often begin by aligning pairs of fragments [6]. This stage is known as pairwise registration. The accuracy and robustness of pairwise registration are critical and often determine the accuracy of the final reconstruction.

Our pairwise registration pipeline begins by extracting pointwise features. These are matched to form a set of putative correspondences. We then use a high-dimensional convolutional network (ConvNet) to estimate the veracity of each correspondence. Lastly, we use a Weighted Procrustes method to align 3D scans given correspondences with associated likelihood weights, and refine the result by optimizing a robust objective.

The following notation will be used throughout the paper. We consider two point clouds, $X = [\mathbf{x}_1, ..., \mathbf{x}_{N_x}] \in \mathbb{R}^{3\times N_x}$ and $Y = [\mathbf{y}_1, ..., \mathbf{y}_{N_y}] \in \mathbb{R}^{3\times N_y}$, with N_x and N_y points respectively, where $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^3$. A correspondence between $\mathbf{x}_i$ and $\mathbf{y}_j$ is denoted as $\mathbf{x}_i \leftrightarrow \mathbf{y}_j$ or (i, j).

3.1. Feature Extraction

To prepare for registration, we extract pointwise features that summarize geometric context in the form of vectors in metric feature space. Our pipeline is compatible with many feature descriptors. We use Fully Convolutional Geometric Features (FCGF) [9], which have recently been shown to be both discriminative and fast. FCGF are also compact, with dimensionality as low as 16 to 32, which supports rapid neighbor search in feature space.

3.2. Correspondence Confidence Prediction

Given the features $\mathcal{F}_x = \{\mathbf{f}_{\mathbf{x}_1}, ..., \mathbf{f}_{\mathbf{x}_{N_x}}\}$ and $\mathcal{F}_y = \{\mathbf{f}_{\mathbf{y}_1}, ..., \mathbf{f}_{\mathbf{y}_{N_x}}\}$ of two 3D scans, we use the nearest neighbor in the feature space to generate a set of putative correspondences or matches $\mathcal{M} = \{(i, \text{argmin}_j \|\mathbf{f}_{\mathbf{x}_i} - \mathbf{f}_{\mathbf{y}_j}\|) | i \in [1, ..., N_x]\}$. This procedure is deterministic and can be hand-crafted to filter out noisy correspondences with ratio or reciprocity tests [50]. However, we propose to learn this heuristic filtering process through a convolutional network that learns to analyze the underlying geometric structure of the correspondence set.

We first provide a 1-dimensional analogy to explain the geometry of correspondences. Let A be a set of 1-dimensional points $A = \{0, 1, 2, 3, 4\}$ and B be another such set $B = \{10, 11, 12, 13, 14\}$. Here B is a translation of A: $B = \{a_i + 10 | a_i \in A\}$. If an algorithm returns a set of possible correspondences $\{(0, 10), (1, 11), (2, 12), (3, 13), (4, 14), (0, 14), (4, 10)\}$, then the set of correct correspondences (inliers) will form a line (first 5 pairs), whereas incorrect correspondences (outliers) will form random noise outside the line (last 2 pairs). If we extend this to 3D scans and pointclouds, we can also represent a 3D correspondence $\mathbf{x}_i \leftrightarrow \mathbf{y}_j$ as a point in 6-dimensional space $[\mathbf{x}_i^T, \mathbf{y}_j^T]^T \in \mathbb{R}^6$. The inlier correspondences will be distributed on a lower-dimensional surface in this 6D space, determined by the geometry of the 3D input. We denote $\mathcal{P} = \{(i, j) | \ \|T^*(\mathbf{x}_i) - \mathbf{y}_j\| < \tau, (i, j) \in \mathcal{M}\}$ as a set of inliers or a set of correspondences (i, j) that align accurately up to the threshold τ under the ground truth transformation T^*. Meanwhile, the outliers $\mathcal{N} = \mathcal{P}^C \cap \mathcal{M}$ will be scattered outside the surface $\mathcal{P}$. To identify the inliers, we use a convolutional network. Such networks have been proven effective in related dense prediction tasks, such as 3D point cloud segmentation [16, 7]. The convolutional network in our setting is in 6-dimensional space [8]. The network predicts a likelihood for each correspondence, which is a point in 6D space $[\mathbf{x}_i^T, \mathbf{y}_j^T]^T$. The prediction is interpreted as the likelihood that the correspondence is true: an inlier.

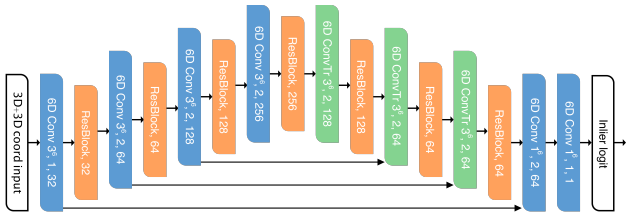


Figure 2: 6-dimensional convolutional network architecture for inlier likelihood prediction (Sec. 3.2). The network has a U-net structure with residual blocks between strided convolutions. Best viewed on the screen.

Note that the convolution operator is translation invariant, thus our 6D ConvNet will generate the same output regardless of the absolute position of inputs in 3D. We use a similar network architecture to Choy et al. [9] to create a 6D convolutional network with skip connections within the spatial resolution across the network. The architecture of the 6D ConvNet is shown in Fig. 2. During training, we use the binary cross-entropy loss between the likelihood prediction that a correspondence (i, j) is an inlier, $p_{(i,j)} \in [0, 1]$, and the ground-truth correspondences $\mathcal{P}$ to optimize the network parameters:

$$L_{\text{bce}}(\mathcal{M}, T^*) = \frac{1}{|\mathcal{M}|}\left(\sum_{(i,j)\in\mathcal{P}} \log p_{(i,j)} + \sum_{(i,j)\in\mathcal{N}} \log p^C_{(i,j)}\right), \tag{1}$$

where $p^C = 1 - p$ and $|\mathcal{M}|$ is the cardinality of the set of putative correspondences.

3.3. Weighted Procrustes for SE(3)

The inlier likelihood estimated by the 6D ConvNet provides a weight for each correspondence. The original

Procrustes method [15] minimizes the mean squared error between corresponding points $\frac{1}{N}\sum_{(i,j)\in\mathcal{M}}\|\mathbf{x}_i-\mathbf{y}_j\|^2$ and thus gives equal weight to all correspondences. In contrast, we minimize a weighted mean squared error $\sum_{(i,j)\in\mathcal{M}} w_{(i,j)}\|\mathbf{x}_i-\mathbf{y}_j\|^2$. This change allows us to pass gradients through the weights, rather than through the position [41], and enables the optimization to scale to dense correspondence sets.

Formally, Weighted Procrustes analysis minimizes:

$$e^2 = e^2(R,\mathbf{t};\mathbf{w},X,Y) \tag{2}$$

$$= \sum_{(i,j)\in\mathcal{M}} \tilde{w}_{(i,j)}(\mathbf{y}_j-(R\mathbf{x}_i+\mathbf{t}))^2 \tag{3}$$

$$= \mathrm{Tr}\left((Y-RX-\mathbf{t}\mathbf{1}^T)W(Y-RX-\mathbf{t}\mathbf{1}^T)^T\right), \tag{4}$$

where $\mathbf{1}=(1,...,1)^T$, $X=[\mathbf{x}_1,...,\mathbf{x}_{|\mathcal{M}|}]$, and $Y=[\mathbf{y}_{J_1},...,\mathbf{y}_{J_{|\mathcal{M}|}}]$. J is a list of indices that defines the correspondences $\mathbf{x}_i \leftrightarrow \mathbf{y}_{J_i}$. $\mathbf{w}=[w_1,\cdots,w_{|\mathcal{M}|}]$ is the weight vector and $\tilde{\mathbf{w}}=[\tilde{w}_1,\cdots,\tilde{w}_{|\mathcal{M}|}] \triangleq \frac{\phi(\mathbf{w})}{||\phi(\mathbf{w})||_1}$ denotes the normalized weight after a nonlinear transformation ϕ that applies heuristic prefiltering. $W=\mathrm{diag}(\tilde{\mathbf{w}})$ forms the diagonal weight matrix.

Theorem 1 *: The R and $\boldsymbol{t}$ that minimize the squared error $e^2(R,\boldsymbol{t})=\sum_{(i,j)} w_{(i,j)}(\boldsymbol{y}_j-R\boldsymbol{x}_i-\boldsymbol{t})^2$ are $\hat{\boldsymbol{t}}=(Y-RX)W\mathbf{1}$ and $\hat{R}=USV^T$ where $U\Sigma V^T=SVD(\Sigma_{xy})$, $\Sigma_{xy}=YKWKX^T$, $K=I-\sqrt{\tilde{\mathbf{w}}}\sqrt{\tilde{\mathbf{w}}}^T$, and $S=diag\,(1,\cdots,1,det(U)det(V))$.*

Sketch of proof. First, we differentiate e^2 w.r.t. $\mathbf{t}$ and equate the partial derivative to 0. This gives us $\hat{\mathbf{t}}=(Y-RX)W\mathbf{1}$. Next, we substitute $X=KX+X\sqrt{\tilde{\mathbf{w}}}\sqrt{\tilde{\mathbf{w}}}^T$ on Eq. 4 and do the same for Y. Then, we substitute $\mathbf{t}=\hat{\mathbf{t}}$ into the squares. This yields $e^2=\mathrm{Tr}\left((Y-RX)KWK^T(Y-RX)^T\right)$ and expanding the term results in two squares plus $-2\mathrm{Tr}(YKWK^TX^TR^T)$. We maximize the last negative term whose maximum is the sum of all singular values, which leads to $\hat{R}$. The full derivation is in the supplement. □

We can easily extend the above theorem to incorporate a scaling factor $c\in\mathbb{R}^+$, or anisotropic scaling for tasks such as scan-to-CAD registration, but in this paper we assume that partial scans of the same scene have the same scale.

The Weighted Procrustes method generates rotation $\hat{R}$ and translation $\hat{\mathbf{t}}$ as outputs that depend on the weight vector $\mathbf{w}$. In our current implementation, $\hat{R}$ and $\hat{\mathbf{t}}$ are directly sent to the robust registration module in Section 4 as an initial pose. However, we briefly demonstrate that they can also be embedded in an end-to-end registration pipeline, since Weighted Procrustes is differentiable. From a top-level loss function L of $\hat{R}$ and $\hat{\mathbf{t}}$, we can pass the gradient through the closed-form solver, and update parameters in downstream modules:

$$\frac{\partial}{\partial\mathbf{w}}L(\hat{R},\hat{\mathbf{t}}) = \frac{\partial L(\hat{R},\hat{\mathbf{t}})}{\partial\hat{R}}\frac{\partial\hat{R}}{\partial\hat{\mathbf{w}}} + \frac{\partial L(\hat{R},\hat{\mathbf{t}})}{\partial\hat{\mathbf{t}}}\frac{\partial\hat{\mathbf{t}}(\hat{R},\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}, \tag{5}$$

where $L(\hat{R},\hat{\mathbf{t}})$ can be defined as the combination of differentiable rotation error (RE) and translation error (TE) between predictions $\hat{R},\hat{\mathbf{t}}$ and ground-truth $R^*,\mathbf{t}^*$:

$$L_{\text{rot}}(\hat{R}) = \arccos\frac{\mathrm{Tr}(\hat{R}^TR^*)-1}{2}, \tag{6}$$

$$L_{\text{trans}}(\hat{\mathbf{t}}) = ||\hat{\mathbf{t}}-\mathbf{t}^*||_2^2, \tag{7}$$

or the Forbenius norm of relative transformation matrices defined in [2, 41]. The final loss is the weighted sum of L_{rot}, L_{trans}, and L_{bce}.

4. Robust Registration

In this section, we propose a fine-tuning module that minimizes a robust loss function of choice to improve the registration accuracy. We use a gradient-based method to refine poses, where a continuous representation [52] for rotations is adopted to remove discontinuities and construct a smooth optimization space. This module initializes the pose from the prediction of the Weighted Procrustes method. During iterative optimization, unlike Maken *et al.* [27], who find the nearest neighbor per point at each gradient step, we rely on the correspondence likelihoods from the 6D ConvNet, which is estimated only once per initialization.

In addition, our framework naturally offers a failure detection mechanism. In practice, Weighted Procrustes may generate numerically unstable solutions when the number of valid correspondences is insufficient due to small overlaps or noisy correspondences between input scans. By computing the ratio of the sum of the filtered weights to the total number of correspondences, *i.e.* $\sum_i\phi(w_i)/|\mathcal{M}|$, we can easily approximate the fraction of valid correspondences and predict whether an alignment may be unstable. When this fraction is low, we resort to a more time-consuming but accurate registration algorithm such as RANSAC [38, 1, 35] or a branch-and-bound method [45] to find a numerically stable solution. In other words, we can detect when our system might fail before it returns a result and fall back to a more accurate but time-consuming algorithm, unlike previous end-to-end methods that use globally pooled latent features [2] or a singly stochastic matrix [41] – such latent representations are more difficult to interpret.

4.1. SE(3) Representation and Initialization

We use the 6D representation of 3D rotation proposed by Zhou *et al.* [52], rather than Euler angles or quaternions. The new representation uses 6 parameters $\mathbf{a}_1,\mathbf{a}_2\in\mathbb{R}^3$ and

can be transformed into a 3×3 orthogonal matrix by

$$f\left(\begin{bmatrix} | & | \\ \mathbf{a}_1 & \mathbf{a}_2 \\ | & | \end{bmatrix}\right) = \begin{bmatrix} | & | & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \\ | & | & | \end{bmatrix}, \tag{8}$$

where $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \in \mathbb{R}^3$ are $\mathbf{b}_1 = N(\mathbf{a}_1)$, $\mathbf{b}_2 = N(\mathbf{a}_2 - (\mathbf{b}_1 \cdot \mathbf{a}_2)\mathbf{b}_1)$, and $\mathbf{b}_3 = \mathbf{b}_1 \times \mathbf{b}_2$, and $N(\cdot)$ denotes L2 normalization. Thus, the final representation that we use is $\mathbf{a}_1, \mathbf{a}_2, \mathbf{t}$ which are equivalent to $R, \mathbf{t}$ using Eq. 8.

To initialize $\mathbf{a}_1, \mathbf{a}_2$, we simply use the first two columns of the rotation matrix R, i.e., $\mathbf{b}_1$, $\mathbf{b}_2$. For convenience, we define f^{-1} as $f^{-1}(f(R)) = R$ though this inverse function is not unique as there are infinitely many choices of $\mathbf{a}_1, \mathbf{a}_2$ that map to the same R.

4.2. Energy Minimization

We use a robust loss function to fine-tune the registration between predicted inlier correspondences. The general form of the energy function is

$$E(R, \mathbf{t}) = \sum_{i=1}^{n} \phi(w_{(i,J_i)}) L(\mathbf{y}_{J_i}, R\mathbf{x}_i + \mathbf{t}), \tag{9}$$

where $\tilde{w}_i$ and J_i are defined as in Eq. 3 and $\phi(\cdot)$ is a pre-filtering function. In the experiments, we use $\phi(w) = I[w > \tau]w$, which clips weights below τ elementwise as neural network outputs bounded logit scores. $L(\mathbf{x}, \mathbf{y})$ is a pointwise loss function between $\mathbf{x}$ and $\mathbf{y}$; we use the Huber loss in our implementation. The energy function is parameterized by R and $\mathbf{t}$ which in turn are represented as $\mathbf{a}_1, \mathbf{a}_2, \mathbf{t}$. We can apply first-order optimization algorithms such as SGD, Adam, etc. to minimize the energy function, but higher-order optimizers are also applicable since the number of parameters is small. The complete algorithm is described in Alg. 1.

Algorithm 1: Deep Global Registration

Input: $X \in \mathbb{R}^{n \times 3}, Y \in \mathbb{R}^{m \times 3}$
Output: $R \in \mathbb{R}^{3 \times 3}, \mathbf{t} \in \mathbb{R}^{3 \times 1}$

$\mathcal{F}_x \leftarrow$ Feature(X) // § 3.1
$\mathcal{F}_y \leftarrow$ Feature(Y)
$J_{x \to y} \leftarrow$ NearestNeighbor$(\mathcal{F}_x, \mathcal{F}_y)$ // § 3.2
$\mathcal{M} \leftarrow \{(i, J_{x \to y, i}) \mid i \in [1, ..., n]\}$
$\mathbf{w} \leftarrow$ InlierProbability$(\mathcal{M})$
if $\mathbb{E}_i \phi(w_i) < \tau_s$ **then**
  return SafeGuardRegistration(X, Y) // §4
else
  $\hat{R}, \hat{\mathbf{t}} \leftarrow \text{argmin}_{R,\mathbf{t}}\, e^2(R, \mathbf{t}; \mathbf{w}, X, Y)$ // § 3.3
  $\mathbf{a} \leftarrow f^{-1}(\hat{R}), \mathbf{t} \leftarrow \hat{\mathbf{t}}$ // § 4.1
  while *not converging* **do**
    $\ell \leftarrow \sum_{(i,j) \in \mathcal{M}} \phi(w_{(i,j)}) L(Y_j, f(\mathbf{a})X_i + \mathbf{t})$
    $\mathbf{a} \leftarrow$ Update$(\mathbf{a}, \frac{\partial}{\partial \mathbf{a}} \ell(\mathbf{a}, \mathbf{t}))$
    $\mathbf{t} \leftarrow$ Update$(\mathbf{t}, \frac{\partial}{\partial \mathbf{t}} \ell(\mathbf{a}, \mathbf{t}))$
  return $f(\mathbf{a}), \mathbf{t}$

5. Experiments

We analyze the proposed model in two registration scenarios: pairwise registration where we estimate an SE(3) transformation between two 3D scans or fragments, and multi-way registration which generates a final reconstruction and camera poses for all fragments that are globally consistent. Here, pairwise registration serves as a critical module in multi-way registration.

For pairwise registration, we use the 3DMatch benchmark [47] which consists of 3D point cloud pairs from various real-world scenes with ground truth transformations estimated from RGB-D reconstruction pipelines [17, 10]. We follow the train/test split and the standard procedure to generate pairs with at least 30% overlap for training and testing [12, 11, 9]. For multi-way registration, we use the simulated Augmented ICL-NUIM dataset [6, 18] for quantitative trajectory results, and Indoor LiDAR RGB-D dataset [32] and Stanford RGB-D dataset [6] for qualitative registration visualizations. Note in this experiment we use networks trained on the 3DMatch training set and do not fine-tune on the other datasets. This illustrates the generalization abilities of our models. Lastly, we use KITTI LIDAR scans [13] for outdoor pairwise registration. As the official registration splits do not have labels for pairwise registration, we follow Choy *et al.* [9] to create pairwise registration train/val/test splits.

For all indoor experiments, we use 5cm voxel downsampling [35, 51], which randomly subsamples a single point within each 5cm voxel to generate point clouds with uniform density. For safeguard registration, we use RANSAC and the safeguard threshold $\tau_s = 0.05$, which translates to 5% of the correspondences should be valid. We train learning-based state-of-the-art models and our network on the training split of the 3DMatch benchmark. During training, we augment data by applying random rotations varying from -180 to 180 degrees around a random axis. Ground-truth pointwise correspondences are found using nearest neighbor search in 3D space. We train the 6-dimensional ConvNet on a single Titan XP with batch size 4. SGD is used with an initial learning rate 10^{-1} and an exponential learning rate decay factor 0.99.

5.1. Pairwise Registration

In this section, we report the registration results on the test set of the 3DMatch benchmark [47], which contains 8 different scenes as depicted in Fig. 3. We measure translation error (*TE*) defined in Eq. 6, rotation error (*RE*) defined

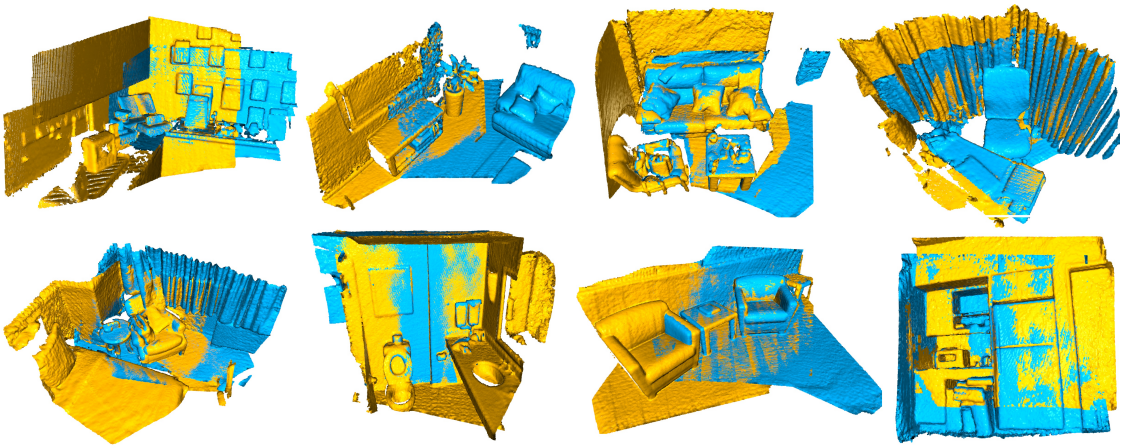


Figure 3: Global registration results of our method on all 8 different test scenes in 3DMatch [47]. Best viewed in color.

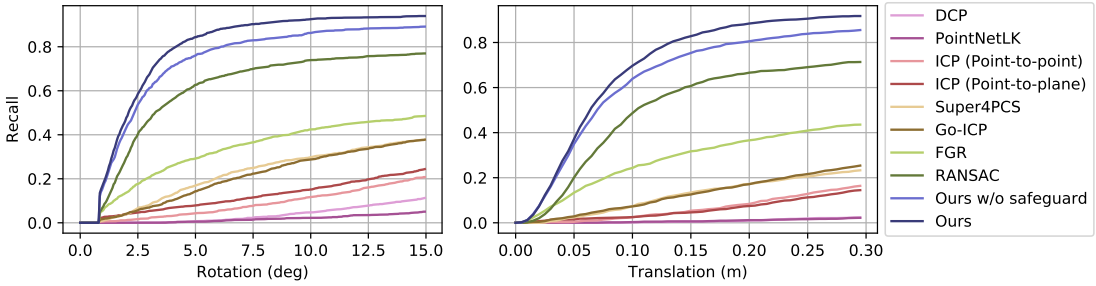


Figure 4: Overall pairwise registration recall (y-axis) on the 3DMatch benchmark with varying rotation (left image) and translation (right image) error thresholds (x-axis). Our approach outperforms baseline methods for all thresholds while being $6.5\times$ faster than the most accurate baseline.

in Eq. 7, and *recall*. Recall is the ratio of successful pairwise registrations and we define a registration to be successful if its rotation error and translation error are smaller than predefined thresholds. Average TE and RE are computed only on these successfully registered pairs since failed registrations return poses that can be drastically different from the ground truth, making the error metrics unreliable.

We compare our methods with various classical methods [50, 35, 45] and state-of-the-art learning based methods [41, 42, 2, 31]. All the experiments are evaluated on an Intel i7-7700 CPU and a GTX 1080Ti graphics card except for Go-ICP [45] tested on an Intel i7-5820K CPU. In Table 1, we measure recall with the TE threshold 30cm which is typical for indoor scene relocalization [30], and RE threshold 15 degrees which is practical for partially overlapping scans from our experiments. In Fig. 4, we plot the sensitivity of recall on both thresholds by changing one threshold and setting the other to infinity. Fig. 5 includes detailed statistics on separate test scenes. Our system outperforms all the baselines on recall by a large margin and achieves the lowest translation and rotation error consistently on most scenes.

Classical methods. To compare with classical methods, we evaluate point-to-point ICP, Point-to-plane ICP, RANSAC [35], and FGR [50], all implemented in Open3D [51]. In addition, we test the open-source Python bindings of Go-ICP [45] and Super4PCS [29]. For RANSAC and FGR, we extract FPFH from voxel-downsampled point clouds. The results are shown in Table 1.

ICP variants mostly fail as the dataset contains challenging 3D scan sequences with small overlap and large camera viewpoint change. Super4PCS, a sampling-based algorithm, performs similarly to Go-ICP, an ICP variant with branch-and-bound search.

Feature-based methods, FGR and RANSAC, perform better. When aligning 5cm-voxel-downsampled point clouds, RANSAC achieves recall as high as 70%, while FGR reaches 40%. Table 1 also shows that increasing the number of RANSAC iterations by a factor of 2 only im-

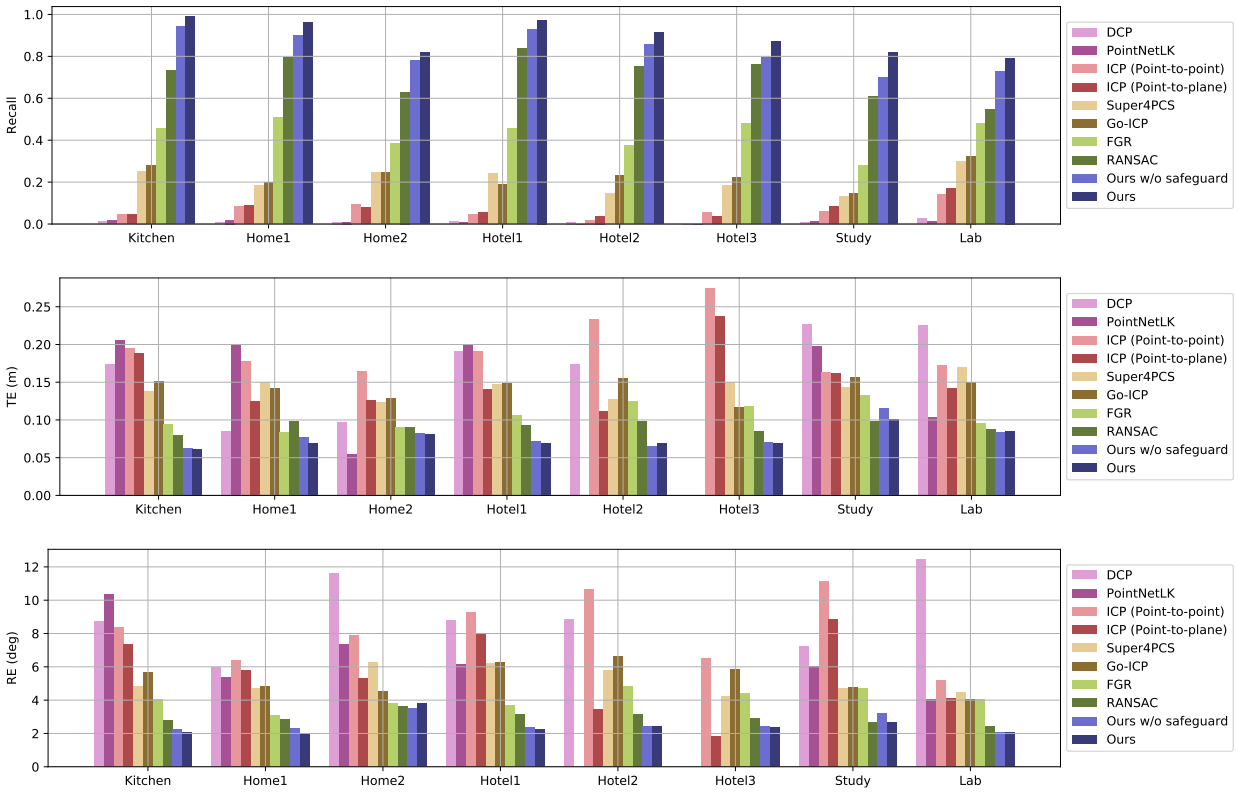


Figure 5: Analysis of 3DMatch registration results per scene. *Row 1*: recall rate (higher is better). *Row 2-3*: TE and RE measured on successfully registered pairs (lower is better). Our method is consistently better on all scenes, which were not seen during training. Note: a missing bar corresponds to zero successful alignments in a scene.

Table 1: *Row 1-6*: registration results of our method and classical global registration methods on point clouds voxelized with 5cm voxel size. Our method outperforms RANSAC and FGR while being as fast as FGR. *Row 7-10*: results of ICP variants. *Row 11 - 12*: results of learning-based methods. The learning-based methods generally fail on real-world scans. Time includes feature extraction.

	Recall	TE (cm)	RE (deg)	Time (s)
Ours w/o safeguard	85.2%	7.73	2.58	0.70
Ours	**91.3%**	**7.34**	**2.43**	1.21
FGR [50]	42.7%	10.6	4.08	**0.31**
RANSAC-2M [35]	66.1%	8.85	3.00	1.39
RANSAC-4M	70.7%	9.16	2.95	2.32
RANSAC-8M	74.9%	8.96	2.92	4.55
Go-ICP [45]	22.9%	14.7	5.38	771.0
Super4PCS [29]	21.6%	14.1	5.25	4.55
ICP (P2Point) [51]	6.04%	18.1	8.25	0.25
ICP (P2Plane) [51]	6.59%	15.2	6.61	0.27
DCP [41]	3.22%	21.4	8.42	0.07
PointNetLK [2]	1.61%	21.3	8.04	0.12

proves performance marginally. Note that our method is about twice as fast as RANSAC with 2M iterations while achieving higher recall and registration accuracy.

Learning-based methods. We use 3DRegNet [31], Deep Closest Point (DCP) [41], PRNet [42], and PointNetLK [2] as our baselines. We train all the baselines on 3DMatch with the same setup and data augmentation as ours for all experiments.

For 3DRegNet, we follow the setup outlined in [31], except that we do not manually filter outliers with ground truth, and train and test with the standard realistic setup. We find that the registration loss of 3DRegNet does not converge during training and the rotation and translation errors are consistently above 30 degrees and 1m during test.

We train Deep Closest Point (DCP) with 1024 randomly sampled points for each point cloud for 150 epochs [41]. We initialize the network with the pretrained weights provided by the authors. Although the training loss converges, DCP fails to achieve reasonable performance for point clouds with partial overlap. DCP uses a singly stochastic matrix to find correspondences, but this formulation assumes that all points in point cloud X have at least one corresponding point in the convex hull of point cloud Y.

Table 2: ATE (cm) error on the Augmented ICL-NUIM dataset with simulated depth noise. For InfiniTAM, the loop closure module is disabled since it fails in all scenes. For BAD-SLAM, the loop closure module only succeeds in 'Living room 2'.

	ElasticFusion [43]	InfiniTAM [21]	BAD-SLAM [39]	Multi-way + FGR [50]	Multi-way + RANSAC [51]	Multi-way + Ours
Living room 1	66.61	46.07	fail	78.97	110.9	**21.06**
Living room 2	24.33	73.64	40.41	24.91	**19.33**	21.88
Office 1	**13.04**	113.8	18.53	14.96	14.42	15.76
Office 2	35.02	105.2	26.34	21.05	17.31	**11.56**
Avg. Rank	3	5	5	3.5	2.5	**2**

This assumption fails when some points in X have no corresponding points in Y, as is the case for partially overlapping fragments. We also tried to train PRNet [42] on our setup, but failed to get reasonable results due to random crashes and high-variance training losses.

Lastly, we fine-tune PointNetLK [2] on 3DMatch for 400 epochs, starting from the pretrained weights provided by the authors. PointNetLK uses a single feature that is globally pooled for each point cloud and regresses the relative pose between objects, and we suspect that a globally pooled feature fails to capture complex scenes such as 3DMatch.

In conclusion, while working well on object-centric synthetic datasets, current end-to-end registration approaches fail on real-world data. Unlike synthetic data, real 3D point cloud pairs contain multiple objects, partial scans, self-occlusion, substantial noise, and may have only a small degree of overlap between scans.

5.2. Multi-way Registration

Multi-way registration for RGB-D scans proceeds via multiple stages. First, the pipeline estimates the camera pose via off-the-shelf odometry and integrates multiple 3D scans to reduce noise and generate accurate 3D fragments of a scene. Next, a pairwise registration algorithm roughly aligns all fragments, followed by multi-way registration [6] which optimizes fragment poses with robust pose graph optimization [23].

We use a popular open-source implementation of this registration pipeline [51] and replace the pairwise registration stage in the pipeline with our proposed modules. Note that we use the networks trained on the 3DMatch training set and test on the multi-way registration datasets [18, 32, 6]; this demonstrates cross-dataset generalization.

We test the modified pipeline on the Augmented ICL-NUIM dataset [6, 18] for quantitative trajectory results, and Indoor LiDAR RGB-D dataset [32] and Stanford RGB-D dataset [6] for qualitative registration visualizations. We measure the absolute trajectory error (ATE) on the Augmented ICL-NUIM dataset with simulated depth noise. As shown in Table 2, compared to state-of-the-art online SLAM [43, 21, 39] and offline reconstruction methods [50], our approach yields consistently low error across scenes.

For qualitative results, we compare pairwise fragment registration on these scenes against FGR and RANSAC in Fig. 6. Full scene reconstruction results are shown in the supplement.

Table 3: Registration on the KITTI test split [13, 9]. We use thresholds of 0.6m and 5 degrees. 'Ours + ICP' refers to our method followed by ICP for fine-grained pose adjustment. The runtime includes feature extraction.

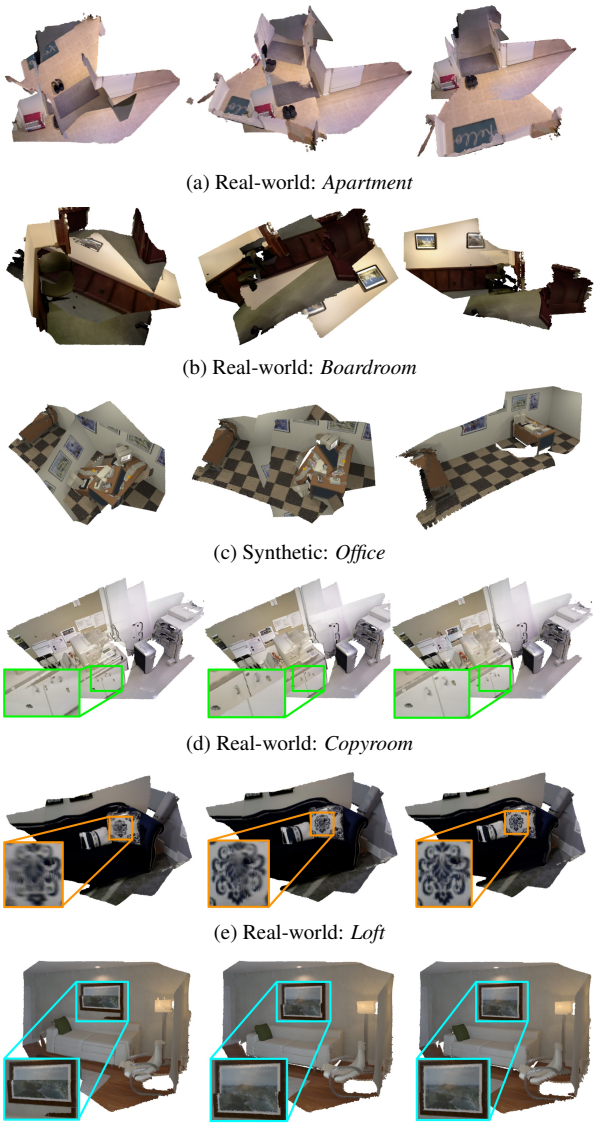
	Recall	TE (cm)	RE (deg)	Time (s)
FGR [50]	0.2%	40.7	1.02	1.42
RANSAC [35]	34.2%	25.9	1.39	1.37
FCGF [9]	**98.2%**	10.2	0.33	6.38
Ours	96.9%	21.7	0.34	**2.29**
Ours + ICP	98.0	**3.46**	**0.14**	2.51

5.3. Outdoor LIDAR Registration

We use outdoor LIDAR scans from the KITTI dataset [13] for registration, following [9]. The registration split of Choy *et al*. [9] uses GPS-IMU to create pairs that are at least 10m apart and generated ground-truth transformation using GPS followed by ICP to fix errors in GPU readings. We use FCGF features [9] trained on the training set of the registration split to find the correspondences and trained the 6D ConvNet for inlier confidence prediction similar to how we trained the system for indoor registration. We use voxel size 30cm for downsampling point clouds for all experiments. Registration results are reported in Tab. 3 and visualized in Fig. 7.

6. Conclusion

We presented Deep Global Registration, a learning-based framework that robustly and accurately aligns real-world 3D scans. To achieve this, we used a 6D convolutional network for inlier detection, a differentiable Weighted Procrustes algorithm for scalable registration, and a gradient-based optimizer for pose refinement. Experiments show that our approach outperforms both classical and learning-based registration methods, and can serve as a ready-to-use plugin to replace alternative registration methods in off-the-shelf scene reconstruction pipelines.



(a) Real-world: *Apartment*

(b) Real-world: *Boardroom*

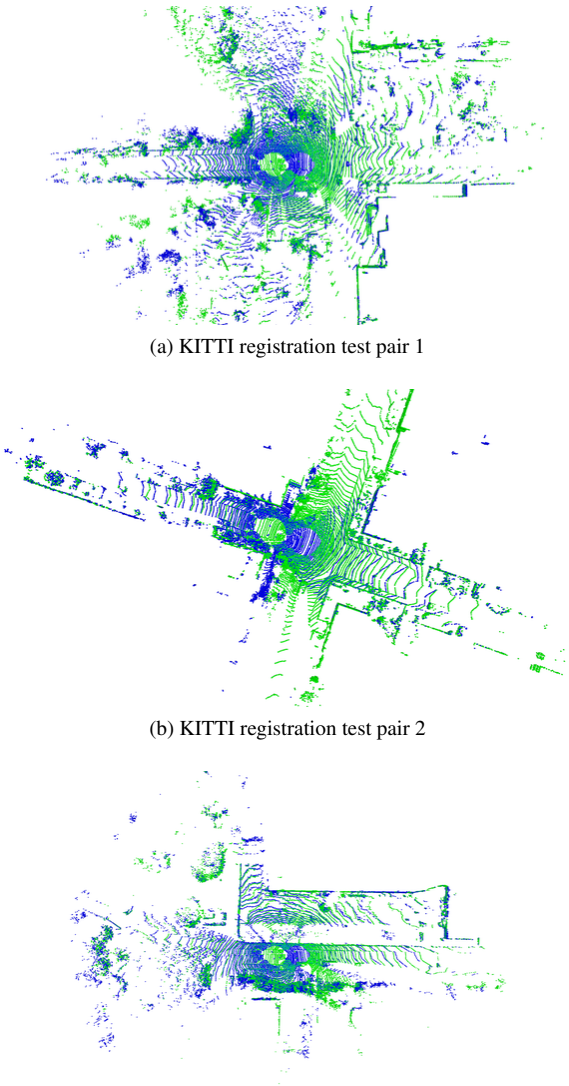
(c) Synthetic: *Office*

(d) Real-world: *Copyroom*

(e) Real-world: *Loft*

(f) Synthetic: *Livingroom*

Figure 6: Fragment registrations on [6, 32]. From left to right: FGR [50], RANSAC [35], Ours. *Row 1-3*: our method succeeds on scenes with small overlaps or ambiguous geometry structures while other methods fail. *Row 4-6*: by combining Weighted Procrustes and gradient-based refinement, our method outputs more accurate registrations in one pass, leading to better aligned details.



(a) KITTI registration test pair 1

(b) KITTI registration test pair 2

(c) KITTI registration test pair 3

Figure 7: KITTI registration results. Pairs of scans are at least 10m apart.

References

[1] Dror Aiger, Niloy J Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust pairwise surface registration. *ACM Transactions on Graphics*, 2008. 2, 4

[2] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. PointNetLK: Robust & efficient point cloud registration using PointNet. In *CVPR*, 2019. 1, 2, 4, 6, 7, 8

[3] Paul J Bes, Neil D McKay, et al. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992. 2

[4] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse iterative closest point. In *Eurographics*, 2013. 2

[5] Qin Cai, David Gallup, Cha Zhang, and Zhengyou Zhang. 3D deformable face tracking with a commodity depth camera. In *ECCV*, 2010. 1

[6] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015. 2, 3, 5, 8, 9

[7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *CVPR*, 2019. 3

[8] Christopher Choy, Junha Lee, René Ranftl, Jaesik Park, and Vladlen Koltun. High-dimensional convolutional networks

for geometric pattern recognition. In *CVPR*, 2020. 2, 3

[9] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *ICCV*, 2019. 1, 2, 3, 5, 8

[10] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics*, 2017. 5

[11] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors. In *ECCV*, 2018. 1, 2, 5

[12] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNet: Global context aware local features for robust 3D point matching. In *CVPR*, 2018. 2, 5

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 5, 8

[14] Zan Gojcic, Caifa Zhou, Jan Dirk Wegner, and Andreas Wieser. The perfect match: 3D point cloud matching with smoothed densities. In *CVPR*, 2019. 2

[15] John C Gower. Generalized procrustes analysis. *Psychometrika*, 1975. 2, 4

[16] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 3

[17] Maciej Halber and Thomas Funkhouser. Fine-to-coarse global registration of RGB-D scans. In *CVPR*, 2017. 5

[18] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *ICRA*, 2014. 2, 5, 8

[19] Dirk Holz, Alexandru E Ichim, Federico Tombari, Radu B Rusu, and Sven Behnke. Registration with the point cloud library: A modular framework for aligning in 3-D. *RA-L*, 2015. 2

[20] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *Pattern Analysis and Machine Intelligence*, 1999. 2

[21] Olaf Kähler, V. A. Prisacariu, and David W. Murray. Real-time large-scale dense 3D reconstruction with loop closure. In *ECCV*, 2016. 8

[22] Marc Khoury, Qian-Yi Zhou, and Vladlen Koltun. Learning compact geometric features. In *ICCV*, 2017. 1, 2

[23] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *ICRA*, 2011. 8

[24] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *DARPA Image Understanding Workshop*, 1981. 1, 2

[25] João Maciel and Joao Paulo Costeira. A global solution to sparse correspondence problems. *Pattern Analysis and Machine Intelligence*, 2003. 2

[26] Ameesh Makadia, Alexander Patterson, and Kostas Daniilidis. Fully automatic registration of 3D point clouds. In *CVPR*, 2006. 2

[27] Fahira Afzal Maken, Fabio Ramos, and Lionel Ott. Speeding up iterative closest point using stochastic gradient descent. In *ICRA*, 2019. 2, 4

[28] Haggai Maron, Nadav Dym, Itay Kezurer, Shahar Kovalsky, and Yaron Lipman. Point registration via efficient convex relaxation. *ACM Transactions on Graphics*, 2016. 2

[29] Nicolas Mellado, Dror Aiger, and Niloy J Mitra. Super 4PCS fast global pointcloud registration via smart indexing. *Computer Graphics Forum*, 2014. 2, 6, 7

[30] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 2015. 6

[31] G. Dias Pais, Pedro Miraldo, Srikumar Ramalingam, Jacinto C. Nascimento, Venu Madhav Govindu, and Rama Chellappa. 3DRegNet: A deep neural network for 3D point registration. *arXiv*, 2019. 1, 2, 6, 7

[32] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *ICCV*, 2017. 2, 5, 8, 9

[33] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *CVPR*, 2014. 1

[34] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *ECCV*, 2018. 1

[35] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *ICRA*, 2009. 1, 2, 4, 5, 6, 7, 8, 9

[36] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *IROS*, 2008. 2

[37] S. Salti, F. Tombari, and L. di Stefano. SHOT: Unique signatures of histograms for surface and texture description. *CVIU*, 2014. 2

[38] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 2007. 1, 2, 4

[39] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *CVPR*, 2019. 8

[40] F. Tombari, S. Salti, and L. Di Stefano. Unique shape context for 3D data description. In *ACM Workshop on 3D Object Retrieval*, 2010. 2

[41] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *ICCV*, 2019. 1, 2, 4, 6, 7

[42] Yue Wang and Justin M. Solomon. PRNet: Self-supervised learning for partial-to-partial registration. In *NeurIPS*, 2019. 2, 6, 7, 8

[43] Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. ElasticFusion: Dense SLAM without a pose graph. In *Robotics: Science and Systems*, 2015. 8

[44] Heng Yang and Luca Carlone. A polynomial-time solution for robust registration with extreme outlier rates. In *Robotics: Science and Systems*, 2019. 2

[45] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *Pattern Analysis and Machine Intelligence*, 2015. 1, 2, 4, 6, 7

[46] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 1

[47] 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. 1, 2, 5, 6

[48] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 1994. 2

[49] Chen Zhao, Zhiguo Cao, Chi Li, Xin Li, and Jiaqi Yang. NM-Net: Mining reliable neighbors for robust feature correspondences. In *CVPR*, 2019. 2

[50] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *ECCV*, 2016. 1, 2, 3, 6, 7, 8, 9

[51] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv*, 2018. 5, 6, 7, 8

[52] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019. 2, 4