(Just) A Spoonful of Refinements Helps the Registration Error Go Down

Sérgio Agostinho[1] Aljoša Ošep[2] Alessio Del Bue[3] Laura Leal-Taixé[2]
[1] Instituto Superior Técnico, Portugal [2]Technical University of Munich, Germany
[3]Fondazione Istituto Italiano di Tecnologia, Italy
[1]sergio.agostinho@tecnico.ulisboa.pt [2]{aljosa.osep, leal.taixe}@tum.de
[3]alessio.delbue@iit.it

Abstract

We tackle data-driven 3D point cloud registration. Given point correspondences, the standard Kabsch algorithm provides an optimal rotation estimate. This allows to train registration models in an end-to-end manner by differentiating the SVD operation. However, given the initial rotation estimate supplied by Kabsch, we show we can improve point correspondence learning during model training by extending the original optimization problem. In particular, we linearize the governing constraints of the rotation matrix and solve the resulting linear system of equations. We then iteratively produce new solutions by updating the initial estimate. Our experiments show that, by plugging our differentiable layer to existing learning-based registration methods, we improve the correspondence matching quality. This yields up to a 7% decrease in rotation error for correspondence-based data-driven registration methods.

1. Introduction

Finding a geometrical transformation that aligns two point sets is at the core of several down-stream tasks such as range data fusion [20], ego- or object pose tracking [15, 17, 46], 3D shape completion [16] and camera re-localization [1]. This challenging problem has a long-standing research history [14, 36, 2, 35, 26], as the correspondence between point clouds is usually not known, or it may not even explicitly exist. Other challenges include the fact that 3D sensors often observe object or scene surfaces only partially, that the density of scans varies with respect to the distance from the sensor, and that point clouds are usually corrupted by severe noise and outliers.

Existing optimization-based methods typically solve point cloud registration by finding a transformation that minimizes the distance between two point sets, according to some criterion, *e.g.*, Chamfer distance [12]. Their performance critically depends on the quality of the 3D point correspondences between the point sets [21, 33, 32, 22, 45, 43, 9]. To cope with the limitations of prior heuristic approaches, recent methods [45, 43, 9, 8, 7, 15, 18] leverage the representational power of deep neural networks to learn an estimate of the transformation that aligns point clouds in an end-to-end manner.

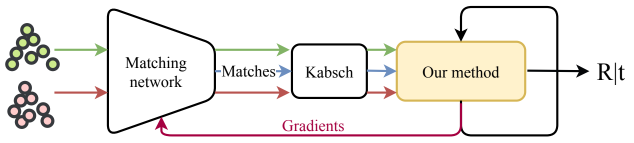

Figure 1. We propose a novel, differentiable rotation estimator. Given a trainable, correspondence-based registration method and initial rotation estimate (by, *e.g.*, Kabsch algorithm), our iterative rotation estimator performs a series of pose refinements to produce gradients that guide the matching network towards improving point correspondences.

To ensure that the estimated matrix is a valid rigid transformation matrix, state-of-the-art learning-based methods [38, 44, 5] employ the Kabsch algorithm [14] and thus need to differentiate through the SVD operation [28]. Kabsch produces globally optimal estimates for a given set of correspondences. However, if the point matches are not perfect, we can design other pose oriented geometric incentives to guide the correspondence network to learn better matches.

To this end, we formulate a local approximation of the correspondence distance minimization problem and show that this helps the network to produce better correspondences. Our method takes the estimate produced by Kabsch as input and it performs a number of recurrent iterative steps that over time push the matching network to improve correspondence matching quality (see Figure 1). As rotation matrices are governed by non-linear constraints, we propose a linearization of these constraints around the computed initial estimate and solve the optimization problem using the method of Lagrange multipliers. The result is a linear system of equations, where the new rotation estimate can be extracted in closed-form. This estimate is recurrently refined over a fixed number of iterations, akin to the classic approach from Drummond and Cipolla [11]. In comparison to Kabsch, the use of linearized constraints makes our estimator increasingly sensitive to particular geometric configurations of point clouds, that result in estimates that might diverge from the original Kabsch estimates. We discuss these configurations in more detail in the supplementary material. However, Kabsch also benefits from a network that is encouraged to prevent these particular configurations.

We show experimentally that our approach is an add-on module that can be used in combination with different learning-based methods for point cloud registration, such as Deep Closest Point [38] and RPM-Net [44]. We improve on the results of two state-of-the-art methods, in a registration task on the ModelNet40 [39] dataset. Moreover, we show that when using our layer, there is no need to impose an additional loss that aids the matching network (as used in, *e.g.*, RPM-Net [44]), as the correspondences improve during the training implicitly. In summary, our contributions are the following: i) We propose a novel, differentiable rotation estimation layer that promotes the trainable matching network to improve correspondence matching quality; ii) We show that this differentiable rotation estimator can be obtained by linearizing the governing constraints for rotation matrices around the initial solution in an iterative fashion, resulting in linear constrained optimization problem with closed-form solution; iii) We augment two state-of-the-art deep global registration methods with our layer and improve upon them in the task of point cloud registration on the ModelNet40 benchmark. Our module can improve learning-based registration methods, at the minimal cost of adding a parameter-free layer during training.[1]

2. Related Work

Point cloud registration is an extensively studied topic, with a considerable literature spanning decades [14, 36, 2, 35, 26]. In this section, we present some of the most relevant optimization-based methods and recent data-driven approaches.

Optimization-based methods. Consolidated solutions for point clouds registration are based on Iterative Closest Point (ICP) [4, 2] and its variants [31, 34, 3]. These methods alternate between the correspondence search and optimal pose estimation given such matches in the point sets. Standard ICP approaches use a nearest neighbor strategy in coordinate space to establish correspondences. This approach is sensitive to the initial transformation estimate and is therefore considered to be a local method. With exact correspondences, estimating the optimal pose can be formulated as an Orthogonal Procrustes problem, that can be minimized using the Kabsch algorithm [14]. For this reason, the community invested efforts in developing robust point descriptors for finding the best correspondences. Most of them rely on hand-crafted descriptors, *e.g.*, SPIN [21], SHOT [33] and FPFH [32]. Instead, recent efforts privilege a data-driven approach by leveraging point cloud encoders [30, 24] to learn point descriptors [22, 9, 7].

Beyond improving feature detection and matching, global registration methods often adopt a robust cost function to improve robustness to outliers [47]. The work of [40] explicitly focuses on identifying outliers in the point matches, while [42] finds a globally optimal solution through branch-and-bound systematic search in the SE(3) solution space. Different from previous approaches, [23] uses sophisticated sampling and graph matching mechanisms to bypass having to establish putative correspondences.

Data-driven methods. Recent advances in the area of point cloud representation learning [29] paved the way towards data-driven methods for point cloud alignment [15, 13, 38, 5, 18, 10]. While it is possible to use a learned feature detector [22, 8, 45, 43] in combination with global registration methods [47], the ultimate goal is to optimize the point representation with respect to the final task in an end-to-end manner.

Several methods directly learn to regress the transformation between two point clouds [13, 15, 27, 18]. PoinNetLK [13] requires an initial transformation estimate and proceeds iteratively by minimizing the distances between the point cloud embeddings. AlignNet [15] computes relative transformation in a single shot by first estimating the canonical pose, followed by the estimation of residual transformation. PointGMM [18] leverages hierarchical Gaussian Mixture Models to learn a multi-scale representation of the point cloud that disentangles orientation and shape in the embedding space. The relative transformation can then be computed by estimating a canonical pose of each point cloud.

Other methods explicitly establish correspondences. 3DRegNet [27] leverages a correspondence classification mechanism inspired by Kim *et al.* [25] and regresses the rigid transformation by optimizing directly in the SE(3) manifold. Other methods use Kabsch [14] to ensure that the estimated transformation is a valid euclidean transformation, composed by a proper rotation matrix. Deep Closest Point (DCP) [38] employs a pointer network module [37] to establish soft correspondences between two point clouds based on the learned embeddings. Deep Global Registration [5] additionally employs a network module [6] for correspondence confidence weighting, used in combination with a weighted variant of Kabsch. RPM-Net [44] couples local and global spatial coordinates together with hand-

[1]Our implementation and trained models can be found at https://github.com/SergioRAgostinho/just-a-spoonful

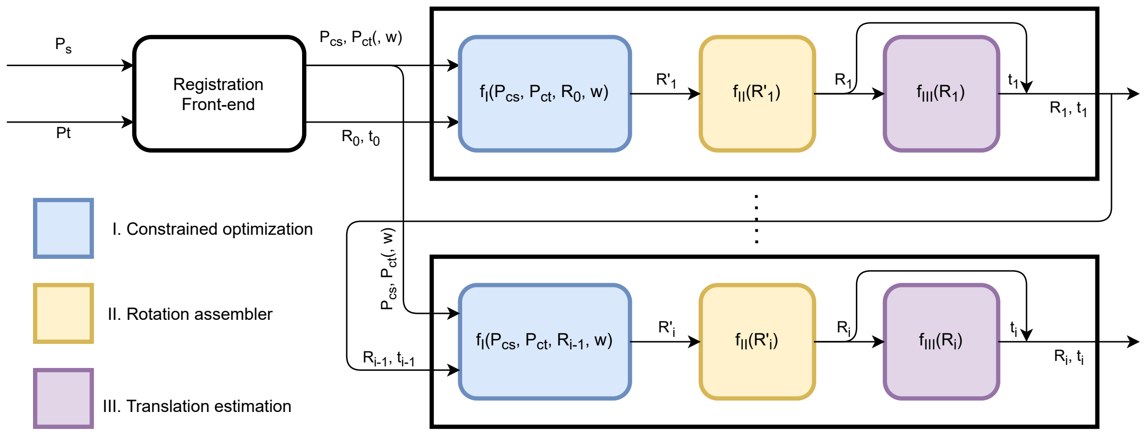

Figure 2. An overview of our proposed method: The variables $\mathtt{P}_s$, $\mathtt{P}_t$ represent the source and target point clouds, and $\mathbf{w}$ is a set of optional weights pondering each correspondence. The trainable *matching network* produces 3D point correspondences $\mathtt{P}_{cs}$ and $\mathtt{P}_{ct}$ from both point clouds, an initial pose estimate $(\mathtt{R}_0, \mathbf{t}_0)$ and an optional, represented in parenthesis $(\cdot)$, vector of weights $\mathbf{w} \in \mathbb{R}_+^N$ ranking each correspondence. At refinement iteration i, we first produce an approximate rotation estimate $\mathtt{R}'_i$ which minimizes the point-to-point distance between correspondences *(constrained optimization module)*. Then, our *rotation assembler* generates a proper rotation matrix from the approximate rotation estimate $\mathtt{R}'_i$. Finally, we estimate the optimal translation $\mathbf{t}_i$ between correspondences using *translation estimator*, given the rotation estimate from the *rotation assembler*.

crafted point-pair features [32], as input to a PPFNet feature encoder [9]. We show experimentally that our method is a worthy add-on to be used in combination with end-to-end trainable correspondence-based methods, such as Deep Closest Point [38] and RPM-Net [44], to improve the registration performance.

3. Method

In this section, we detail our differentiable iterative refinement method that can be added as a complementary step to any correspondence-based registration deep learning method, as can be seen in Figure 2. Inputs to our method are an initial rotation estimate, *e.g*., supplied by Kabsch, a set of point correspondences estimated by any trainable matching network and, optionally, a set of weights ranking the quality of these correspondences. Note, these correspondences are not necessarily correct, in fact, we will show we improve them during the model training thanks to our method. We then perform the following steps iteratively (Figure 2):

Constrained optimization (🟦): We produce an approximate rotation estimate by linearizing governing constraints for rotation matrices around the previous estimate. This module minimizes the weighted point-to-point distance between correspondences, however, the resulting matrix is not necessarily a valid rotation matrix (see subsection 3.2).

Rotation assembler (🟨): We convert the matrix of the previous step into a valid rotation matrix by applying Gram-Schmidt orthogonalization to the first two columns of the input and a cross product to generate the final column.

Translation estimation (🟪): Given an input rotation we can compute the optimal translation vector in a closed-form (see subsection 3.1).

These operations define our novel layer, which refines the rotation estimates iteratively. However, since the pose returned from Kabsch is already optimal, our main contribution is not in improving the final pose directly but instead conditioning the matching part of the network towards learning better correspondences. Empirically we show that our novel differentiable rotation estimator aids correspondence matching registration even though we only impose supervision on the pose.

3.1. Preliminaries

Given a set of correspondences between the source and target point clouds $\mathtt{P}_s, \mathtt{P}_t \in \mathbb{R}^{N\times 3}$ (see Figure 2) our aim is to find a rigid transformation $(\mathtt{R}, \mathbf{t})$ that minimizes the following error:

$$\underset{\mathtt{R},\mathbf{t}}{\arg\min} \quad \sum_{i=1}^N w_i \|\mathbf{p_t}_i - \mathtt{R}\mathbf{p_s}_i - \mathbf{t}\|^2 \tag{1a}$$

$$\text{s.t.} \quad \mathtt{R} \in SO(3), \tag{1b}$$

where $\mathbf{w} \in \mathbb{R}_+^N$ represents the (optionally supplied) set of weights and $\mathbf{p_s}_i$ and $\mathbf{p_t}_i$ are individual points of the source and target point clouds. Given a rotation matrix $\mathtt{R}$, we can simply extract the optimal translation vector $\mathbf{t}$ in a closed-form as:

$$\mathbf{t}^* = \frac{\sum_{i=1}^N w_i(\mathbf{p_t}_i - \mathtt{R}\mathbf{p_s}_i)}{\sum_{i=1}^N w_i} = \bar{\mathbf{p_t}} - \mathtt{R}\bar{\mathbf{p_s}}, \tag{2}$$

where $\bar{\mathbf{p}}_\mathbf{t}$ and $\bar{\mathbf{p}}_\mathbf{s}$ represent the weighted means of the points for each point cloud. We further define $\tilde{\mathbf{p}}_{\mathbf{t}i}$ and $\tilde{\mathbf{p}}_{\mathbf{s}i}$ as the mean-subtracted versions of $\mathbf{p}_{\mathbf{t}i}$ and $\mathbf{p}_{\mathbf{s}i}$, such that $\tilde{\mathbf{p}}_\mathbf{s} = \mathbf{p}_\mathbf{s} - \bar{\mathbf{p}}_\mathbf{s}$ and $\tilde{\mathbf{p}}_\mathbf{t} = \mathbf{p}_\mathbf{t} - \bar{\mathbf{p}}_\mathbf{t}$. We can then factor out the translation component and Eq. (1) can be formulated entirely with the respect to the rotation. Back-substituting Eq. (2) yields the following simplification:

$$\begin{aligned} \underset{\mathtt{R}}{\arg\min} \quad & \textstyle\sum_{i=1}^{N} w_i \|\tilde{\mathbf{p}}_{\mathbf{t}i} - \mathtt{R}\tilde{\mathbf{p}}_{\mathbf{s}i}\|^2 && \text{(3a)} \\ \text{s. t.} \quad & \mathtt{R} \in SO(3). && \text{(3b)} \end{aligned}$$

The Kabsch algorithm [14] provides a closed-form, globally optimal solution (given correspondences) to this problem via SVD as follows:

$$\mathtt{H} = \sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{t}i} \tilde{\mathbf{p}}_{\mathbf{s}i}^\top \tag{4}$$

$$\mathtt{U}, \mathtt{S}, \mathtt{V} = \operatorname{svd}(\mathtt{H}) \tag{5}$$

$$\mathtt{R} = \mathtt{U}\operatorname{diag}([1, 1, \det(\mathtt{U}\mathtt{V}^\top)])\mathtt{V}^\top. \tag{6}$$

The operator diag( ) produces diagonal square matrices, in which the input vector represents the diagonal. Kabsch can only provide the *correct* rotation estimate for pose estimation if correspondences are also correct. Our formulation of the optimization problem (Eq. 3a) as an iterative procedure helps the network to produce better correspondences, as we will show in the experimental section.

3.2. Just a spoonful of refinements

We first present the governing constraints of the rotation matrix and then discuss our proposed relaxation to their local linear approximation. The membership of $\mathtt{R}$ in $SO(3)$ can be expressed as:

$$\begin{aligned} \mathtt{R}^\top\mathtt{R} &= \mathtt{I}_3 && \text{(7a)} \\ \det \mathtt{R} &= 1, && \text{(7b)} \end{aligned}$$

where $\mathtt{I}_3$ is the 3×3 identity matrix. These are quadratic and cubic equality constraints, respectively. Eq. (7a) supplies six constraints and Eq. (7b) an additional one. However, as shown in the supplementary material, given that all expressions are linearized around a point that represents a rotation matrix, the determinant constraint is not longer linearly independent w.r.t. the orthogonality ones and is therefore redundant. The next stage of the formulation requires that all constraints are linearly independent, so we choose to drop the determinant constrained as it allows to proceed with the formulation taking solely into account the orthogonaly related expressions.

Linearization of constraints (▬). Denoting our prior rotation estimate with $\mathtt{R}_{t-1}$, we start by linearizing the linearly independent components of Eq. (7a) around the initialization $\mathtt{R}_{t-1}$, by only taking into consideration the upper triangle section of the constraints matrix. We define matrix $c(\mathtt{R}) = \mathtt{R}^\top\mathtt{R} - \mathtt{I}_3$ such that $c(\mathtt{R}) : \mathbb{R}^{3\times3} \to \mathbb{R}^{3\times3}$ and refer to $c_{ij}(\mathtt{R})$ as the element in the i-th row and j-th column, defined as $c_{ij}(\mathtt{R}) = \mathbf{e}_i^\top(\mathtt{R}^\top\mathtt{R} - \mathtt{I}_3)\mathbf{e}_j$. The variables $\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^3$ are Euclidean bases, vectors of zeros with a single element equal to one at the i-th and j-th elements, respectively. Using Taylor expansion around the initial estimate $\mathtt{R}_{t-1}$ and retaining only terms up to the first order, leads to the following linearized constraints:

$$\begin{aligned} c_{ij}^{(1)}(\mathtt{R}, \mathtt{R}_{t-1}) &= c_{ij}(\mathtt{R}_{t-1}) + \operatorname{tr}\left(\mathtt{E}_{ij}^{\mathbb{S}}\mathtt{R}_{t-1}^\top(\mathtt{R} - \mathtt{R}_{t-1})\right) \\ \text{for} \quad & i = 1, \dots, 3;\ j = i, \dots, 3, \end{aligned} \tag{8}$$

where $c_{ij}^{(1)}$ is the first-order Taylor approximation, of the i-th row and j-th column of the orthogonality constraints $c(\mathtt{R})$. The matrix $\mathtt{E}_{ij}^{\mathbb{S}} = \mathbf{e}_i\mathbf{e}_j^\top + \mathbf{e}_j\mathbf{e}_i^\top = \mathtt{E}_{ij} + \mathtt{E}_{ji} \in \mathbb{S}^3$, with $\mathbb{S}^3$ representing the space of real symmetric matrices of size 3×3. Full derivations for this approximation are provided in the supplementary material.

Langrangian formulation (▬). After the relaxation and linearization of our constraints, we now have an optimization problem with a quadratic cost function and linear constraints. Then, we can enforce the (linearized) equality constraints in Eq. 8 using the method of Lagrange multipliers. Thus, we obtain a closed-form solution that can be formulated as a linear system of equations. We write the new Lagrangian of (3a) as:

$$\mathcal{L}(\mathtt{R}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} \frac{w_i}{2}\|\tilde{\mathbf{p}}_{\mathbf{t}i} - \mathtt{R}\tilde{\mathbf{p}}_{\mathbf{s}i}\|^2 + \sum_{k=1}^{6} \lambda_k c_k^{(1)}(\mathtt{R}, \mathtt{R}_{t-1}), \tag{9}$$

where indices ij previously used to specify the row and column of the constraints $c_{ij}^{(1)}(\mathtt{R}, \mathtt{R}_{t-1})$, are now replaced by the single index k, iterating over the upper triangular part of the matrix. The variables λ_k represent the Lagrange multipliers of the constraints. We form a linear system for which the solution returns our newly refined estimate, by computing the gradient of Eq. (9) with respect to both $\mathtt{R}$ and $\boldsymbol{\lambda}$, and setting it to 0. The optimal $\mathtt{R}$ and $\boldsymbol{\lambda}$ are determined by solving the linear system:

$$\begin{bmatrix} \mathtt{A} & \mathtt{B} \\ \mathtt{B}^\top & 0 \end{bmatrix} \begin{bmatrix} \operatorname{vec}(\mathtt{R}) \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \operatorname{vec}\left(\sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{t}i}\tilde{\mathbf{p}}_{\mathbf{s}i}^\top\right) \\ \mathbf{d} \end{bmatrix}, \tag{10}$$

where the operator vec represents a column-wise vectorization of its input matrix and the matrices $\mathtt{A}$ and $\mathtt{B}$ are defined as in the following. The matrix $\mathtt{A}$ is given by $\mathtt{A} = \begin{bmatrix} \mathbf{a}_1, & \dots, & \mathbf{a}_9 \end{bmatrix}^\top \in \mathbb{R}^{9\times9}$, with each vector $\mathbf{a}_r \in \mathbb{R}^9$, $r = 1, \dots, 9$ such that

$$\mathbf{a}_r = \operatorname{vec}\left(\mathtt{E}_{mn}\left(\sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{s}i}\tilde{\mathbf{p}}_{\mathbf{s}i}^\top\right)\right). \tag{11}$$

Matrix $\mathtt{E}_{mn} \in \mathbb{R}^{3\times3}$ has all elements equal to 0, except the one in row m and column n, which is 1. Matrix $\mathtt{A}$ is constructed from vec $\left(\frac{\partial \mathcal{L}}{\partial \mathtt{R}} = 0\right)$ and retaining all terms that depend on $\mathtt{R}$. Matrix $\mathtt{B} = \begin{bmatrix} \mathbf{b}_1, & \ldots, & \mathbf{b}_6 \end{bmatrix} \in \mathbb{R}^{9\times 6}$ is composed by the following columns $\mathbf{b}_k$ with $k = 1, \ldots, 6$, such that:

$$\mathbf{b}_k = \text{vec}(\mathtt{R}_{t-1}\mathtt{E}_k^{\mathbb{S}}), \tag{12}$$

where the vector $\mathbf{d} \in \mathbb{R}^6$, with each element given by

$$d_k = \text{tr}(\mathtt{E}_k^{\mathbb{S}}) - c_k(\mathtt{R}_{t-1}). \tag{13}$$

In the supplementary material we provide a detailed derivation on how to compose the linear system of equations, specifically matrices $\mathtt{A}$, $\mathtt{B}$ and vector $\mathbf{d}$. After solving the linear system of equations, we obtain a newly refined rotation estimate.

Producing a rotation matrix from a candidate refinement (🟨). Due to the linearization of the original rotation constraints, there is no guarantee that our solution from the optimization module (Figure 2, 🟦) is a valid rotation matrix. One solution to this problem would be to project the matrix to the closest orthogonal matrix using SVD by minimizing the Frobenius norm distance to the input. This projection step might be non-differentiable because the gradient is not defined if the input matrix is already a valid rotation. This problem is discussed in Ionescu *et al*. [19], where they show that the gradient of an SVD is not defined in the case of the input matrix having equal singular values, as a rotation matrix does. Moreover, the closer the singular values are to each other, the more numerically ill-conditioned the gradient becomes. Since we intentionally target having matrices very close to true rotations *i.e*., having all singular values equal to 1, using SVD is not an option. To generate a new rotation representation from our estimate, we instead adopt the strategy proposed by Zhou *et al*. [48] that has close ties to Gram-Schmidt orthogonalization. Assume that $\mathtt{R}'$ is a 3×3 input matrix formed by the columns:

$$\mathtt{R}' = \begin{bmatrix} \mathbf{r}'_1 & \mathbf{r}'_2 & \mathbf{r}'_3 \end{bmatrix}. \tag{14}$$

The output matrix is going to be composed by the following three columns:

$$\mathbf{r}_1 = \frac{\mathbf{r}'_1}{\|\mathbf{r}'_1\|}, \tag{15}$$

$$\mathbf{r}_2 = \frac{(\mathtt{I} - \mathbf{r}_1\mathbf{r}_1^\top)\mathbf{r}'_2}{\|(\mathtt{I} - \mathbf{r}_1\mathbf{r}_1^\top)\mathbf{r}'_2\|}, \tag{16}$$

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2. \tag{17}$$

These operations are performed in the differentiable orthogonalization step (Figure 2, 🟨). Contrary to SVD projection, with this method we are not operating close to a gradient singularity. Equations (15) and (16) have singularities if the denominator is 0. However, as shown in the supplementary material, we are not operating close to this region, leading to a stable training procedure.

3.3. Augmenting the Loss

Using our refinement module, we produce $t = 1, \ldots, N_r$ additional pose estimates. We apply the loss term to every pose estimate, both initialization and refinements. This is applicable to every term that depends on the predicted pose. As an example, the original loss function used to train Deep Closest Point [38] is formulated as

$$\text{Loss} = \|\mathtt{R}^\top\mathtt{R}_{gt} - \mathtt{I}_3\|^2 + \|\mathbf{t} - t_{gt}\|^2 + \lambda\|\theta\|^2. \tag{18}$$

In this expression, θ represents the networks parameters and λ is a hyper-parameter balancing weight-decay during training. With the new pose estimates the loss now becomes:

$$\begin{aligned} \text{Loss} &= \frac{1}{N_r+1}\sum_{i=1}^{N_r+1}\|\mathtt{R}_i^\top\mathtt{R}_{gt} - \mathtt{I}_3\|^2, \\ &+ \frac{1}{N_r+1}\sum_{i=1}^{N_r+1}\|\mathbf{t}_i - t_{gt}\|^2 + \lambda\|\theta\|^2. \end{aligned} \tag{19}$$

During training, our refiner is required to output all possible poses, initializing each new refinement of the pose from the previous iteration. *We stress that our refinement strategy is only used during training*, initializing each new refinement of the pose from the previous iteration. At test time, we output the pose produced by Kabsch, which is optimal given correct correspondences.

3.4. Understanding the Differences

The estimator we propose solves a very similar problem to Kabsch, minimizing the same correspondence loss, but under a different set of constraints: a linear approximation of the original second-order equality constraints. A network trained with and without our additional layers will learn a different set of parameters and will have different registration performance. The differences in resulting parameters indicate that the gradients used to optimize the network during training are different. The loss in Eq. (19) ponders an average pose error between all poses produced: if the poses are all equal, the gradient w.r.t. to one pose or w.r.t. to all poses is the same. In the majority of situations, given a rotation estimate from Kabsch, our estimator will replicate this estimate. However, the linearized constraints make our estimator increasingly sensitive to certain geometric configurations of point clouds. Under these configurations, the estimator will produce a pose estimate that will diverge from Kabsch at each iteration. We stress that in our case, divergence comes paired with the positive effect of facilitating the network to avoid said configurations. The geometric relationship, e.g. distance and angle, between Kabsch's (constrained) solution and the unconstrained one, is one example of aspects that govern the occurrence of the divergent behavior. We expand on these ideas and provide additional insightful examples in the supplementary material.

4. Experimental Evaluation

In this section, we show the merit of our method by improving the accuracy of existing matching-based deep global registration methods, Deep Closest Point [38] and RPM-Net [44]. We show that our method helps by improving the quality of correspondence matching and, as a result, improves the pose estimates. We compare the performance of our full pipeline to several baselines. We conduct all experiments employing 5 refinements of our method, a choice supported by our ablation studies, reported in the supplemental material. Finally, we show the improvements produced by our method are more than a byproduct of a particular initialization, by conducting multiple training sessions with different weight initializations and showing the average pose error is consistent with our prior benchmarks.

Datasets. We conduct our experiments using ModelNet40 [39] and 3DMatch [45] based on the experimental setting of [38, 44]. ModelNet40 consists of CAD models containing several symmetrical objects which create pose ambiguities. 3DMatch is a dataset composed of RGB-D scene fragments. We create rigid transformations by randomly sampling rotations as three Euler angles from the interval $[0°, 45°]$ and translations in the range of $[-0.5, 0.5]$ along each axis.

Metrics. To compare our method to prior work, we follow their experimental setting and use the same evaluation metrics, including rotation and translation errors, Chamfer distance, and mean point distance. We compute the rotation error as:

$$\Delta \mathtt{R} = \mathtt{R}^\top \mathtt{R}_{gt} \tag{20}$$

$$\angle \Delta \mathtt{R}_{\text{iso}} = \arccos\left(\frac{\operatorname{tr}(\Delta \mathtt{R}) - 1}{2}\right) \tag{21}$$

$$(\angle_z, \angle_y, \angle_x)_{\text{ani}} = f_{\text{Euler}_{z\to y\to x}}(\Delta \mathtt{R}), \tag{22}$$

in both an isotropic (21) and an anisotropic (22) forms. The matrices $\mathtt{R}$ and $\mathtt{R}_{gt}$ stand for the rotation estimate and ground-truth, respectively. The operator $\operatorname{tr}(\cdot)$ stands for the trace of a matrix, $f_{\text{Euler}_{z\to y\to x}}$ is a function which decomposes the rotation matrix in intrinsic Euler angles following the axes sequence $z \to y \to x$. The translation error is computed as the $p - norm$:

$$\Delta \mathbf{t} = \|\mathbf{t} - \mathbf{t}_{gt}\|_{p=\{1,2\}}, \tag{23}$$

with $\mathbf{t}$ and $\mathbf{t}_{gt}$ standing for the translation estimate and ground-truth. The Chamfer distance between two point sets $\mathtt{P}_s$ and $\mathtt{P}_t$ is given by:

$$\mathrm{CD}(\mathtt{P}_s, \mathtt{P}_t) = \frac{1}{|\mathtt{P}_s|}\sum_{\mathbf{p}_{s_i}\in \mathtt{P}_s} \min_{\mathbf{p}_{t_j}\in \mathtt{P}_t} \|\mathbf{p}_{s_i} - \mathbf{p}_{t_j}\|_2^2 \tag{24}$$

$$+ \frac{1}{|\mathtt{P}_t|}\sum_{\mathbf{p}_{t_j}\in \mathtt{P}_t} \min_{\mathbf{p}_{s_i}\in \mathtt{P}_s} \|\mathbf{p}_{s_i} - \mathbf{p}_{t_j}\|_2^2. \tag{25}$$

Lastly, the mean point distance is computed as:

$$\Delta \mathbf{p} = \frac{1}{N}\sum_{i=1}^{N} \|(\mathtt{R} - \mathtt{R}_{gt})\mathbf{p}_{s_i} + \mathbf{t} - \mathbf{t}_{gt}\|_2. \tag{26}$$

For this metric, we pick the source point cloud to compute the distance without loss of generality.

4.1. Deep Closest Point with ModelNet40 Data

We augment Deep Closest Point (DCP) [38] with our refinement stage and evaluate the performance of the final network following their evaluation protocol. We use 9,843 meshes for training and 2,468 for testing. From each mesh, we uniformly sample 1024 points based on the face area and scale them to be within a unit sphere. DCP's reports the anisotropic rotation error from Eq. (22) and translation error, computing the Mean Squared Error (MSE), the Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). We present and discuss the simpler scenario of alignment for identical point clouds in the supplementary material.

Alignment under Gaussian noise. In Table 1 we report point cloud registration results obtained on instances of CAD models that were held-out during the model training. Furthermore, we add a Gaussian noise $\mathcal{N}(0, 0.01^2)$, clamped at $[-0.05, 0.05]$ during test time using the model trained on noise-free data. To ensure that there are no true correspondences, we applied the noise independently to the source point cloud.[2] Our refinement strategy significantly improves the rotation estimator error while incurring in negligible worsening of the translation error. This experiment confirms that we improve the generalization to held-out instances even when 1-to-1 correspondences do not exist. We improve the rotation error by 9% at the marginal cost of translation accuracy of 0.06%, when normalized by the maximum magnitude of the rotation and translations sampled. This is an experiment in which DCP performs worse because the perfect correspondence assumption is broken during testing. The interesting aspect is that despite never having access to cases during training in which there were no perfect correspondences, applying our layer assists the network to generalize better to this case.

Alignment for unseen categories. In Table 2, we report results for unseen categories. In prior experiments, the network was trained with all categories, with separate instances being part of the training and testing datasets. In this experiment, all instances from the first 20 categories are used for training and all instances of the last 20 categories are used for testing, putting the generalization capabilities to the test. Our method achieves the lowest errors in all metrics reported. In particular, we improve the rotation error

[2]Note that this is different to the experiment conducted in [38], where the noise was not added independently.

Model	RMSE($\mathbb{R}$)°	MAE($\mathbb{R}$)°	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)
DCP-v2	12.974750	5.830703	**0.003941**	**0.002502**
DCP-v2 + ours	**8.938098**	**4.373459**	0.004221	0.002777

Table 1. Deep Closest Point on ModelNet40: Test on objects with Gaussian noise only added to the source point cloud. The noise is only added at test time, using the models trained under noiseless conditions. *There are no perfect correspondences between the source and target point clouds.*

Model	RMSE($\mathbb{R}$)°	MAE($\mathbb{R}$)°	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)
ICP	29.876431	23.626110	0.293266	0.251916
Go-ICP [41]	13.865736	2.914169	0.022154	0.006219
FGR [47]	9.848997	1.445460	0.013503	0.002231
PointNetLK [13]	17.502113	5.280545	0.028007	0.007203
DCP-v2	3.150191	2.007210	0.005039	0.003703
DCP-v2 + ours	**2.051713**	**1.431898**	**0.004543**	**0.003333**

Table 2. Deep Closest Point on ModelNet40: Test on unseen categories.

by 2% and the translation error by 0.1%, when normalized by the maximum magnitude of the rotation and translations sampled. This is an experiment in which despite the unseen categories, there are still perfect correspondences in place, which is a more favorable scenario for DCP and that is why our improvements are less substantial than when Gaussian noise was added.

4.2. RPM-Net with ModelNet40 Data

Similar to DCP, we augment RPM-Net [44] with our proposed layer and evaluate the performance of the combined network. We use implementation and pre-trained models provided by the authors. As in subsection 4.1, we use the first 20 categories of ModelNet40 for training and validation and the last 20 categories solely for testing. The training, validation, and testing splits are composed of 5,112, 1,202, and 1,266 models, respectively. As in [44], we also report mean rotation error as in Eq. (21) and mean translation error as in Eq. (23) with $p = 2$, marked as isotropic errors, and Chamfer distance (25). We report the experiment of point cloud alignment under Gaussian noise in the supplementary material and advance directly to the more challenging scenario of partial point cloud alignment.

Alignment with partial point clouds. In this experiment, we evaluate partial point cloud registration. We randomly sample random half-space of the point cloud and retain only 70% of the original point cloud, ensuring that there is a partial overlap in extent between point clouds. Simultaneously, we independently subsample 717 points of each half-space and add Gaussian noise $\mathcal{N}(0, 0.01^2)$ clamped at $[-0.05, 0.05]$. RPM-Net uses two loss terms: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{reg}} + \lambda\mathcal{L}_{\text{inliers}}$. While the $\mathcal{L}_{\text{reg}}$ term penalizes the pose error

Method	Anisotropic err.		Isotropic err.		$\tilde{CD}$
	(Rot.)	(Trans.)	(Rot.)	(Trans.)	
ICP	13.719	0.132	27.250	0.280	0.0153
RPM	9.771	0.092	19.551	0.212	0.0081
FGR	19.266	0.090	30.839	0.192	0.0119
PointNetLK	15.931	0.142	29.725	0.297	0.0235
DCP-v2	6.380	0.083	12.607	0.169	0.0113
RPM-Net	0.893	0.0087	1.712	0.018	**0.00085**
RPM-Net + Ours	**0.826**	0.0081	1.575	0.017	**0.00085**
RPM-Net†	0.993	0.0087	1.861	0.018	0.00099
RPM-Net + Ours†	0.872	**0.0074**	**1.554**	**0.015**	0.00088

Table 3. RPM-Net on ModelNet40: Performance on partially visible data with noise. The Chamfer distance using groundtruth transformations is 0.00055. †Models trained without the inlier term in the loss function.

directly, the second $\lambda\mathcal{L}_{\text{inliers}}$ term acts directly on the correspondences. This term incentivizes the network to rank more matches as inliers. Since our layer also targets correspondence quality, we perform a test where we discard the inlier loss term. As can be seen in Table 3, we improve the rotation error by 0.137°and the translation error by 1e-3. Contrary to the baseline, when our layer is included, it benefits from not having the inlier term acting directly on the correspondences. Without it, we manage to further lower rotation and translation errors.

4.3. Deep Closest Point with 3DMatch Data

The 3DMatch dataset is composed of 7317, 643 and 1861 point cloud pairs for training, validation and testing, respectively. Each fragment is downsampled to a point cloud of 1024 points through voxel grid average filtering *i.e.*, all points inside a voxel are averaged to produce the resulting point for that voxel. Each point cloud pair is rescaled as to ensure that all points lie inside a unit ball norm, and each pair is guaranteed to have a minimum surface overlap of 30%. We present our results in Table 5 and these show an improvement of 2.2% and 1.7% when normalized by the by the maximum magnitude of the rotation and translations sampled, showing that out discoveries generalize to real data.

4.4. Meaningful Improvements Beyond a Convenient Initialization

The inclusion of our layer produces improvements that occasionally can be marginal. To ensure that these are not attributed to a particular initialization we conduct an experiment where we train DCP 26 times with and without our layer, evaluating the mean and standard deviation of pose error over all runs. We conduct the experiment following

Model \(mean / std)	MSE($\mathtt{R}$)°	RMSE($\mathtt{R}$)°	MAE($\mathtt{R}$)°	MSE($\mathbf{t}$)	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)
DCP-v2	20.969088 / 1.439e+01	4.264873 / 1.667e+00	2.670795 / 9.765e-01	**0.000047 / 5.962e-05**	**0.006250 / 2.900e-03**	**0.004555 / 2.153e-03**
DCP-v2 + ours 5 all	**12.647069 / 1.046e+01**	**3.311812 / 1.296e+00**	**2.130985 / 7.879e-01**	0.000052 / 6.235e-05	0.006532 / 3.039e-03	0.004757 / 2.291e-03

Table 4. Initialization test with Deep Closest Point on ModelNet40. Test with unseen point clouds where all the instances of the last 20 categories are held-out during training and only used for evaluation. The network is trained for 250 epochs.

Model	RMSE($\mathtt{R}$)°	MAE($\mathtt{R}$)°	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)
DCP-v2	11.211800	8.554642	0.137432	0.105473
DCP-v2 + ours	**10.232300**	**7.777227**	**0.128846**	**0.099061**

Table 5. Deep Closest Point on 3DMatch with a 30% minimum of surface overlap.

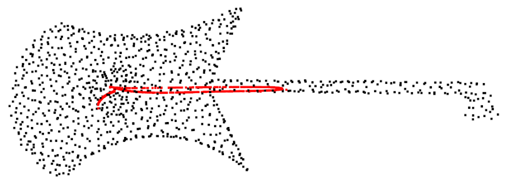

Figure 3. A qualitative example of the how correspondences are generated by Deep Closest Point. In the image we see point clouds of two different colors: black and red. We cherry-pick an example with the lowest pose estimation error, $\angle \Delta \mathtt{R}_{\text{iso}} = 0.2712°$, $\Delta \mathbf{t} = 0.0002$. **Black**: Point cloud generated by applying the ground-truth transformation to the source point cloud i.e., each point is given by $\mathbf{p}_{t_i} = \mathtt{R}_{\text{gt}} \mathbf{p}_{s_i} + \mathbf{t}_{\text{gt}}$. **Red**: The correspondences produced by the network to perform the registration task, where each point represents $\mathbf{p}'_{t_i}$.

the unseen categories protocol just as in previous sections. The results are presented in Table 4. Our method decreases the rotation error by 22% while incurring in a residual increase in translation error.

4.5. Measuring Correspondence Improvement

In this section, we provide some intuition on why evaluating correspondence quality simply based on the Euclidean distance is misleading and only the error in the resulting rotation from Kabsch, accurately captures correspondence quality improvement. Due to space limitations, we further expand some of the results and ideas in the supplementary material.

Correspondence Error is Misleading. For each point $\mathbf{p}_{s_i}$ in the source point cloud, both DCP and RPM-Net regress the coordinates of its corresponding point, expressing it as $\mathbf{p}'_{t_i} = \sum_{j=1}^{N} \alpha_{ij} \mathbf{p}_{t_j}$, where α_{ij} is the probability of point $\mathbf{p}_{s_i}$ matching $\mathbf{p}_{t_i}$. The mean subtracted version of these pairs of correspondences $\tilde{\mathbf{p}}_{s_i}$ and $\tilde{\mathbf{p}}'_{t_i}$ are used to compute $\mathtt{H}$ in Eq. (4). To produce a correct rotation estimate, it is not necessary that $\|\tilde{\mathbf{p}}'_{t_i} - \mathtt{R}\tilde{\mathbf{p}}_{s_i}\|_{\forall i} = 0$. Kabsch is a method that is invariant to scale. Multiplying the source and target point clouds by arbitrary non-negative scalars will produce the same rotation matrix. In an effort to evaluate the quality of correspondences based on the average point distance, we revisit the Gaussian noise experiment from subsection 4.1, where noise is added independently to one of the point clouds at test time. The results show that: the correspondence error is fairly high despite the low pose error; for a marginal improvement of 1% in relative correspondence error, we obtain a 7% improvement in rotation error, almost one order of magnitude above. This cements the idea that correspondence error does fully capture the quality of the correspondences established when these are used with Kabsch.

In Defense of Pose Error as Evaluation Metric. Figure 3 shows a cherry picked example where the pose error is the smallest over the entire test set. Contrary to intuition, the regressed target points (red) hardly resemble the ground truth target points (black) and yet the network is still able to estimate an almost perfect pose. This confirms that a high positional error between regressed and ground truth target points does not imply a bad pose estimate. However, both red and black point clouds have roughly similar principal directions and centroids and we conjecture that this is a sufficient condition for Kabsch to produce accurate estimates. Measuring a difference in orientation in principal directions and a difference in position between centroids is similar to computing a pose error. Therefore, we argue that pose error is a better metric to measure correspondence quality than correspondence positional error.

5. Conclusion

In this paper, we presented a differentiable rotation estimation approach that can be used in combination with the Kabsch algorithm. We show that it improves correspondence matching quality resulting in improved registration. We expect our method to benefit future methods tackling learning-based end-to-end correspondence-based registration method.

Acknowledgements. The authors would like to thank Franziska Gerken, Mark Weber, and Qunjie Zhou for their insightful suggestions. This work was partly funded by FCT through grant PD/BD/114432/2016 and project UIDB/50009/2020. It was also supported by project 24534 - INFANTE, funded by the COMPETE 2020 and Lisboa 2020 programs, under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund.

References

[1] Ioan Andrei Barsan, Shenlong Wang, Andrei Pokrovsky, and Raquel Urtasun. Learning to localize using a lidar intensity map. In *CoRL*, pages 605–616, 2018. 1

[2] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992. 1, 2

[3] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse iterative closest point. *Computer Graphics Forum (Symposium on Geometry Processing)*, 32(5):1–11, 2013. 2

[4] Yang Chen and Gérard G Medioni. Object modeling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, 1992. 2

[5] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 1, 2

[6] Christopher Choy, Junha Lee, René Ranftl, Jaesik Park, and Vladlen Koltun. High-dimensional convolutional networks for geometric pattern recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11227–11236, 2020. 2

[7] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully Convolutional Geometric Features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1, 2

[8] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Eur. Conf. Comput. Vis.*, 2018. 1, 2

[9] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2, 3

[10] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 3d local features for direct pairwise registration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[11] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):932–946, 2002. 2

[12] Andrew W Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and vision computing*, 21(13-14):1145–1153, 2003. 1

[13] Hunter Goforth, Yasuhiro Aoki, Arun Srivatsan Rangaprasad, and Simon Lucey. Pointnetlk: Robust and efficient point cloud registration using pointnet. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 2, 7

[14] John C Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975. 1, 2, 4

[15] Johannes Groß, Aljoša Ošep, and Bastian Leibe. Alignnet-3d: Fast point cloud registration of partially observed objects. In *IEEE Int. Conf. on 3D Vision*, 2019. 1, 2

[16] Jiayuan Gu, Wei-Chiu Ma, Sivabalan Manivasagam, Wenyuan Zeng, Zihao Wang, Yuwen Xiong, Hao Su, and Raquel Urtasun. Weakly-supervised 3d shape completion in the wild. In *Eur. Conf. Comput. Vis.*, 2020. 1

[17] David Held, Jesse Levinson, Sebastian Thrun, and Silvio Savarese. Combining 3d shape, color, and motion for robust anytime tracking. In *Robotics: Science and Systems*, 2014. 1

[18] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Pointgmm: A neural gmm network for point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2

[19] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 5

[20] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011. 1

[21] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999. 1, 2

[22] Marc Khoury, Qian-Yi Zhou, and Vladlen Koltun. Learning compact geometric features. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2

[23] Huu M Le, Thanh-Toan Do, Tuan Hoang, and Ngai-Man Cheung. SDRSAC: Semidefinite-Based Randomized Approach for Robust Point Cloud Registration Without Correspondences. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2019. 2

[24] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution On X-Transformed Points. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 820–830. Curran Associates, Inc., 2018. 2

[25] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2

[26] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010. 1, 2

[27] G Dias Pais, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C Nascimento, Rama Chellappa, and Pedro Miraldo. 3dregnet: A deep neural network for 3d point registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7193–7203, 2020. 2

[28] Théodore Papadopoulo and Manolis IA Lourakis. Estimating the jacobian of the singular value decomposition: Theory and applications. In *European Conference on Computer Vision*, pages 554–570. Springer, 2000. 1

[29] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 2

[30] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: deep learning on point sets for 3d classification

and segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 2

[31] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *International Conference on 3-D Digital Imaging and Modeling*, 2001. 2

[32] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009. 1, 2, 3

[33] Samuele Salti, Federico Tombari, and Luigi Di Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014. 1, 2

[34] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: Science and Systems*, 2009. 2

[35] Yanghai Tsin and Takeo Kanade. A correlation-based approach to robust point set registration. In *Eur. Conf. Comput. Vis.*, 2004. 1, 2

[36] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Computer Architecture Letters*, 13(04):376–380, 1991. 1, 2

[37] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700, 2015. 2

[38] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *Int. Conf. Comput. Vis.*, 2019. 1, 2, 3, 5, 6

[39] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D shapenets: A deep representation for volumetric shapes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015. 2, 6

[40] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. *arXiv preprint arXiv:2001.07715*, 2020. 2

[41] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp: A globally optimal solution to 3D icp point-set registration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(11):2241–2254, 2015. 7

[42] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Int. Conf. Comput. Vis.*, 2019. 2

[43] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 630–646, Cham, 2018. Springer International Publishing. 1, 2

[44] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 1, 2, 3, 6, 7

[45] Andy Zeng, Shuran Song, Matthias Niessner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 2, 6

[46] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy. Robust multi-modality multi-object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2365–2374, 2019. 1

[47] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *Eur. Conf. Comput. Vis.*, 2016. 2, 7

[48] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 5