Supplementary Material for the Paper: (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

1. Introduction

We address and expand certain aspects of the paper that were not relevant for its general understanding, but that give additional insight into our work and validate certain claims we made. We provide proofs and detailed derivations whenever suitable, for some less obvious steps presented in the paper. The documents is organized into 7 main sections: i) In section 2 we provide an in-depth derivation of our local approximate estimator; ii) In section 3 we evaluate the redundancy of our constraints and show that once linearized around a rotation, the determinant constraint can be expressed as a linear combination of the orthogonality constraints. Furthermore, we also demonstrate the Linear Independence Constraint Qualification (LICQ) of the orthogonality constraints; iii) In section 4 we demonstrate how the input matrix supplied to the *rotation assembler* will never land on one of its singularities; iv) In section 5 we provide insights into what makes the linear rotation estimator different from Kabsch, identify how these differences manifest themselves, as well as discuss how the problem geometry influences their occurrence; v) In section 6 we include some additional experiments and ablations with Deep Closest Point and RPM-Net; vi) In section 7 we expand on our claim that the most accurate way of establishing correspondence quality in Kabsch is by looking at pose error.

2. Detailed Derivation of the Linearized Local Approximation Estimator

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

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

where $\mathbf{w} \in \mathbb{R}_+^N$ represents a (n optionally supplied) set of weights and $\mathbf{p_s}_i, \mathbf{p_t}_i \in \mathbb{R}^3$ are individual points of the source and target point clouds. Given an optimal rotation matrix $\mathtt{R}$, the problem reduces itself to a linear regression, where we can simply extract the optimal translation vector $\mathbf{t}$ in a closed-form as:

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

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 $\mathbf{p_t}_i$ and $\mathbf{p_s}_i$, such that $\tilde{\mathbf{p}}_{\mathbf{s}i} = \mathbf{p_s}_i - \bar{\mathbf{p}}_\mathbf{s}$ and $\tilde{\mathbf{p}}_{\mathbf{t}i} = \mathbf{p_t}_i - \bar{\mathbf{p}}_\mathbf{t}$. 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 membership of $\mathtt{R}$ in $SO(3)$ can be expressed as:

$$\begin{aligned} \mathtt{R}^\top\mathtt{R} &= \mathtt{I}_3 && \text{(4a)} \\ \det \mathtt{R} &= 1, && \text{(4b)} \end{aligned}$$

where $\mathtt{I}_3$ is a 3×3 identity matrix. These are quadratic and cubic equality constraints, respectively. Eq. (4a) supplies six constraints and Eq. (4b) an additional one. However, when evaluating the first-order terms of the its Taylor expansion around a rotation matrix, only the orthogonality constraints respect the LICQ. We can optionally drop either one of the orthogonality constraints or the determinant constraint. We drop the latter to simplify the formulation. We expand this further in section 3.

Linearization of Constraints. Denoting our prior rotation estimate as $\mathtt{R}_{t-1}$, we linearize Eq. (4a) around it, only taking into consideration the upper triangle section of the constraints' matrix. We define matrix $c(\mathtt{R}) = \mathtt{R}^\top\mathtt{R} - \mathtt{I}$ such that $c(\mathtt{R}) : \mathbb{R}^{3\times 3} \rightarrow \mathbb{R}^{3\times 3}$ and refer to $c_{ij}(\mathtt{R})$ as the element in the i-th row and j-th column, defined as $c_{ij}(\mathtt{R}) = \mathbf{e}_i^\top(\mathtt{R}^\top\mathtt{R} - \mathtt{I})\mathbf{e}_j$. 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. To linearize it, let us formulate a first-order Taylor expansion for each element in $c(\mathtt{R})$, around an initial estimate $\mathtt{R}_{t-1}$:

$$c_{ij}^{(1)}(\mathtt{R}, \mathtt{R}_{t-1}) = c_{ij}(\mathtt{R}_{t-1}) + \mathrm{tr}\left(\frac{\partial c_{ij}(\mathtt{R}_{t-1})}{\partial \mathtt{R}}^\top (\mathtt{R} - \mathtt{R}_{t-1})\right), \tag{5}$$

where the superscript $^{(1)}$ represents the first-order Taylor approximation. The derivative $\frac{\partial c_{ij}(\mathtt{R})}{\partial \mathtt{R}}$ is computed as follows:

$$\begin{aligned} \frac{\partial c_{ij}(\mathtt{R})}{\partial \mathtt{R}} &= \frac{\partial}{\partial \mathtt{R}} \mathbf{e}_i^\top(\mathtt{R}^\top\mathtt{R} - \mathtt{I})\mathbf{e}_j & \text{(6a)} \\ &= \frac{\partial}{\partial \mathtt{R}} \mathbf{e}_i^\top \mathtt{R}^\top \mathtt{R}\mathbf{e}_j & \text{(6b)} \\ &= \mathtt{R}\mathbf{e}_j\mathbf{e}_i^\top + \mathtt{R}\mathbf{e}_i\mathbf{e}_j^\top. & \text{(6c)} \end{aligned}$$

Substituting it back yields,

$$\begin{aligned} c_{ij}^{(1)}(\mathtt{R}, \mathtt{R}_{t-1}) &= c_{ij}(\mathtt{R}_{t-1}) + \mathrm{tr}\left(\mathtt{E}_{ij}^{\mathbb{S}}\mathtt{R}_{t-1}^\top(\mathtt{R} - \mathtt{R}_{t-1})\right) \quad (7) \\ \text{for} \quad & i = 1, \dots, 3;\ j = i, \dots, 3, \end{aligned}$$

where $\mathtt{E}_{ij}^{\mathbb{S}} = \mathbf{e}_i\mathbf{e}_j^\top + \mathbf{e}_j\mathbf{e}_i^\top = \mathtt{E}_{ij} + \mathtt{E}_{ji} \in \mathbb{S}^3$ and $\mathtt{E}_{ij} = \mathbf{e}_i\mathbf{e}_j^\top$.

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. (7) 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 Lagrangian of Eq. (3a) as:

$$\mathcal{L}(\mathtt{R}, \mathtt{R}_{t-1}) = \sum_{i=1}^{N} \frac{w_i}{2} \|\tilde{\mathbf{p}_\mathbf{t}}_i - \mathtt{R}\tilde{\mathbf{p}_\mathbf{s}}_i\|^2 + \sum_{k=1}^{6} \lambda_k c_k^{(1)}(\mathtt{R}, \mathtt{R}_{t-1}), \tag{8}$$

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:

$$c^{(1)}(\mathtt{R}, \mathtt{R}_{t-1}) = \underbrace{\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ & c_{22} & c_{23} \\ & & c_{33} \end{bmatrix}}_{ij} = \underbrace{\begin{bmatrix} c_1 & c_2 & c_4 \\ & c_3 & c_5 \\ & & c_6 \end{bmatrix}}_{k}. \tag{9}$$

The variables λ_k represent the Lagrange multipliers of the constraints. The minimum to the original constrained problem is guaranteed to be a stationary point of the Lagrangian. Thus, we compute the gradient of the Lagrangian and set it to 0 to look for possible optimal candidates. We start by fully expanding all terms in Eq. (8):

$$\begin{aligned} \mathcal{L}(\mathtt{R}, \mathtt{R}_{t-1}) = &\sum_{i=1}^{N} \frac{w_i}{2} \mathrm{tr}(\tilde{\mathbf{p}_\mathbf{t}}_i^\top \tilde{\mathbf{p}_\mathbf{t}}_i - 2\tilde{\mathbf{p}_\mathbf{t}}_i^\top \mathtt{R}\tilde{\mathbf{p}_\mathbf{s}}_i + \tilde{\mathbf{p}_\mathbf{s}}_i^\top \mathtt{R}^\top\mathtt{R}\tilde{\mathbf{p}_\mathbf{s}}_i) \\ &+ \sum_{k=1}^{6} \lambda_k \left(c_k(\mathtt{R}_{t-1}) + \mathrm{tr}\left(\mathtt{E}_k^{\mathbb{S}}\mathtt{R}_{t-1}^\top(\mathtt{R} - \mathtt{R}_{t-1})\right)\right), \end{aligned} \tag{10}$$

followed by computing its partial derivatives:

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \mathtt{R}} &= \sum_{i=1}^{N} w_i(\mathtt{R}\tilde{\mathbf{p}_\mathbf{s}}_i\tilde{\mathbf{p}_\mathbf{s}}_i^\top - \tilde{\mathbf{p}_\mathbf{t}}_i\tilde{\mathbf{p}_\mathbf{s}}_i^\top) \\ &+ \sum_{k=1}^{6} \lambda_k \mathtt{R}_{t-1}\mathtt{E}_k^{\mathbb{S}} & \text{(11a)} \\ \frac{\partial \mathcal{L}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \lambda_k} &= c_k(\mathtt{R}_{t-1}) + \mathrm{tr}\left(\mathtt{E}_k^{\mathbb{S}}\mathtt{R}_{t-1}^\top(\mathtt{R} - \mathtt{R}_{t-1})\right). & \text{(11b)} \end{aligned}$$

We find the stationary points by finding the values of $\mathtt{R}$ and λ_k that verify $\nabla\mathcal{L} = 0$. These equations are linear w.r.t. to the variables $\mathtt{R}$ and λ_k, meaning we can estimate them by solving a linear system of equations. The system needs to be rewritten so that a linear solver can estimate estimate the unknowns $\begin{bmatrix}\mathrm{vec}(\mathtt{R})^\top & \boldsymbol{\lambda}^\top\end{bmatrix}^\top$. The operator vec, represents a column-wise vectorization operator and $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_6]^\top$. Let us also define $\mathbf{r} = \mathrm{vec}(\mathtt{R})$ for brevity. The optimal $\mathtt{R}$ and $\boldsymbol{\lambda}$ are determined by solving a linear system of the form:

$$\begin{bmatrix} \mathtt{A} & \mathtt{B} \\ \mathtt{B}^\top & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_\mathtt{R} \\ \mathbf{d}_{\boldsymbol{\lambda}} \end{bmatrix}. \tag{12}$$

We derive $\mathtt{A}$, $\mathtt{B}$, $\mathbf{d}_\mathtt{R}$ and $\mathbf{d}_{\boldsymbol{\lambda}}$ in the following paragraphs.

Partial Derivative w.r.t. $\mathtt{R}$. We need to rearrange the terms in Eq. (11) in order to now express them w.r.t. $\mathbf{r}$. Starting with Eq. (11a), $\frac{\partial \mathcal{L}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \mathtt{R}} : \mathbb{R}^{3\times3} \to \mathbb{R}^{3\times3}$, we need to access individual elements of this matrix in order to vectorize $\mathtt{R}$. Applying a similar step to Eq. (6), we formulate

$$\left.\frac{\partial \mathcal{L}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \mathtt{R}}\right._{mn} = \mathbf{e}_m^\top \frac{\partial \mathcal{L}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \mathtt{R}} \mathbf{e}_n, \tag{13}$$

for $m = 1, \dots, 3$ and $n = 1, \dots, 3$. We introduce the following identity $\mathrm{tr}(\mathtt{A}^\top\mathtt{B}) = \mathrm{vec}(\mathtt{A})^\top \mathrm{vec}(\mathtt{B})$. Focusing only in the terms that depend on $\mathtt{R}$, we have:

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathtt{R}}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \mathtt{R}} &= \mathbf{e}_m^\top \mathtt{R} \sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{s}i} \tilde{\mathbf{p}}_{\mathbf{s}i}^\top \mathbf{e}_n && (14)\\
&= \operatorname{tr}\left(\mathbf{e}_m^\top \mathtt{R} \sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{s}i} \tilde{\mathbf{p}}_{\mathbf{s}i}^\top \mathbf{e}_n\right) && (15)\\
&= \operatorname{tr}\left(\sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{s}i} \tilde{\mathbf{p}}_{\mathbf{s}i}^\top \mathbf{e}_n \mathbf{e}_m^\top \mathtt{R}\right) && (16)\\
&= \operatorname{tr}\left(\left(\mathbf{e}_m \mathbf{e}_n^\top \sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{s}i} \tilde{\mathbf{p}}_{\mathbf{s}i}^\top\right)^\top \mathtt{R}\right) && (17)\\
&= \operatorname{tr}\left(\left(\mathtt{E}_{mn} \sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{s}i} \tilde{\mathbf{p}}_{\mathbf{s}i}^\top\right)^\top \mathtt{R}\right) && (18)\\
&= \mathbf{a}_{mn}^\top \mathbf{r}, && (19)
\end{aligned}$$

with $\mathbf{a}_{mn} = \operatorname{vec}\left(\mathtt{E}_{mn} \sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{s}i} \tilde{\mathbf{p}}_{\mathbf{s}i}^\top\right)$. This generates an equation for each element of $\frac{\partial \mathcal{L}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \mathtt{R}}$. However, we still need to stack these equations as rows of a matrix, in a suitable form for a linear solver. We do so by vectorizing $\frac{\partial \mathcal{L}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \mathtt{R}}$ following a column-wise order. This composes matrix $\mathtt{A}$ as presented in the paper:

$$\mathtt{A} = \begin{bmatrix}\mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} & \mathbf{a}_{21} & \dots & \mathbf{a}_{33}\end{bmatrix}^\top. \qquad (20)$$

In the main paper, we express the different rows of $\mathtt{A}$ according to a single index r. The mapping between indices mn and r is given by,

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}_{mn} = \underbrace{\begin{bmatrix} a_1 & a_4 & a_7 \\ a_2 & a_5 & a_8 \\ a_3 & a_6 & a_9 \end{bmatrix}}_{r}. \qquad (21)$$

Next we look at $\frac{\partial \mathcal{L}_{\lambda_k}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \mathtt{R}}$ *i.e.*, all terms from the partial derivative that depend on λ_k:

$$\frac{\partial \mathcal{L}_{\lambda_k}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \mathtt{R}} = \lambda_k \mathtt{R}_{t-1} \mathtt{E}_k^{\mathbb{S}}. \qquad (22)$$

The terms λ_k are already scalar so we only need to vectorize the expression in order stack all equation like we did for Eq. (20), resulting in the following definitions

$$\begin{aligned}
\mathbf{b}_k &= \operatorname{vec}(\mathtt{R}_{t-1} \mathtt{E}_k^{\mathbb{S}}) && (23)\\
\mathtt{B} &= \begin{bmatrix}\mathbf{b}_1 & \dots & \mathbf{b}_6\end{bmatrix}. && (24)
\end{aligned}$$

All that remains are the constant terms

$$\frac{\partial \mathcal{L}_{\text{constant}}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \mathtt{R}} = \sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{t}i} \tilde{\mathbf{p}}_{\mathbf{s}i}^\top, \qquad (25)$$

that similar to Eqs. (20) (23), are also vectorized:

$$\mathbf{d}_{\mathtt{R}} = \operatorname{vec}\left(\sum_{i=1}^{N} w_i \tilde{\mathbf{p}}_{\mathbf{t}i} \tilde{\mathbf{p}}_{\mathbf{s}i}^\top\right). \qquad (26)$$

This concludes the upper part of the linear system of equations.

Partial Derivative w.r.t. λ_k.
Every partial derivate w.r.t. λ_k, with $k = 1, \dots, 6$, contributes with an equation to the linear system. Addressing first, the terms that depend on $\mathtt{R}$, we have

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathtt{R}}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \lambda_k} &= \operatorname{tr}\left(\mathtt{E}_k^{\mathbb{S}} \mathtt{R}_{t-1}^\top \mathtt{R}\right) && (27)\\
&= \operatorname{tr}\left((\mathtt{R}_{t-1} \mathtt{E}_k^{\mathbb{S}})^\top \mathtt{R}\right) && (28)\\
&= \operatorname{vec}\left(\mathtt{R}_{t-1} \mathtt{E}_k^{\mathbb{S}}\right)^\top \mathbf{r} && (29)\\
&= \mathbf{b}_k^\top \mathbf{r}, && (30)
\end{aligned}$$

now expressed in terms of $\mathbf{r}$. Performing the same operation for every value of k and stacking $\mathbf{b}_k^\top$ as rows will produce the matrix $\mathtt{B}^\top$.

The only terms remaining are

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\text{constant}}(\mathtt{R}, \mathtt{R}_{t-1})}{\partial \lambda_k} &= \operatorname{tr}\left(\mathtt{E}_k^{\mathbb{S}} \mathtt{R}_{t-1}^\top \mathtt{R}_{t-1}\right) - c_k(\mathtt{R}_{t-1}) && (31)\\
&= \operatorname{tr}\left(\mathtt{E}_k^{\mathbb{S}}\right) - c_k(\mathtt{R}_{t-1}) && (32)\\
&= d_{\lambda_k}. && (33)
\end{aligned}$$

Stacking d_{λ_k} for every possible value of k produces the vector $\mathbf{d}_{\boldsymbol{\lambda}} = [d_{\lambda_1}, \dots, d_{\lambda_6}]^\top$.

3. Linear Independence Constraint Qualification

Our optimization problem is governed by rotation constraints as expressed in Eqs. (4). In order to be able to employ the method of Lagrange Multipliers, our problem needs to verify the LICQ. In our particular case, ensuring the LICQ is what guarantees that the linear system of equations in Eq. (12) can be inverted and we can retrieve a unique solution. Contrary to what was done in the previous section, to study the LICQ, it is convenient to express constraints w.r.t. to the column vectors of $\mathtt{R}$. To do so, let us define vectors $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$, such that

$$\mathtt{R} = \begin{bmatrix} | & | & | \\ \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \\ | & | & | \end{bmatrix}, \qquad (34)$$

where $\mathbf{r}_i \in \mathbb{R}^3$, for $i = \{1, 2, 3\}$. With these new variables, we now represent Eqs. (4) as:

$$\begin{aligned}
&\mathbf{r}_i^\top \mathbf{r}_i = 1 \quad \text{for } i = \{1, 2, 3\} && (35a)\\
&\mathbf{r}_i^\top \mathbf{r}_j = 0 \quad \text{for } (i, j) = \{(1, 2), (2, 3), (3, 1)\} && (35b)\\
&\mathbf{r}_1^\top \lfloor \mathbf{r}_2 \rfloor_\times \mathbf{r}_3 = 1 \quad \text{(determinant).} && (35c)
\end{aligned}$$

The operator $\lfloor\cdot\rfloor_\times$ takes a vector in $\mathbb{R}^3$ and produces the skew symmetric matrix

$$\lfloor\mathbf{v}\rfloor_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}. \tag{36}$$

This matrix can be used to represent a vector cross-product in matrix form, e.g. $\mathbf{v}_1 \times \mathbf{v}_2 = \lfloor\mathbf{v}_1\rfloor_\times \mathbf{v}_2$. The determinant constraint in Eq. (35c) is expressed as the scalar triple product of the column vectors. A scalar triple product is a product of the form $\mathbf{v}_1 \cdot (\mathbf{v}_2 \times \mathbf{v}_3)$. The product does not change with a circular shift of the vectors, meaning the determinant of $\mathtt{R}$ can also be represented as $\mathbf{r}_2^\top \lfloor\mathbf{r}_3\rfloor_\times \mathbf{r}_1$ or $\mathbf{r}_3^\top \lfloor\mathbf{r}_1\rfloor_\times \mathbf{r}_2$.

We write the first-order Taylor expansion for each constraint, now formulated w.r.t. column vectors, assuming the form:

$$c^{(1)}(\mathbf{r}_i, \mathbf{r}_{i_{t-1}}) = c(\mathbf{r}_{i_{t-1}}) + \frac{\partial c(\mathbf{r}_{i_{t-1}})}{\partial \mathbf{r}_i}^\top (\mathbf{r}_i - \mathbf{r}_{i_{t-1}}). \tag{37}$$

The method of Lagrange multipliers requires that equality constraints are expressed as functions that are equal to 0. At the same time, because the linearization point $\mathtt{R}_{t-1}$ will always be a rotation matrix, its columns will always ensure that $c(\mathbf{r}_{i_{t-1}}) = 0$.

Unit Norm. We look first into the (matrix) orthogonality constraints that place requirements in the norm of the column vectors. Following this requirement, we define

$$c_{n_i} = \mathbf{r}_i^\top \mathbf{r}_i - 1 \quad \text{for } i = \{1, 2, 3\}. \tag{38}$$

The corresponding derivative at the linearization point is given by

$$\frac{\partial c_{n_i}(\mathbf{r}_{i_{t-1}})}{\partial \mathbf{r}_i} = 2\mathbf{r}_{i_{t-1}}, \tag{39}$$

resulting in the following linearized constraint

$$\begin{aligned} c_{n_i}^{(1)} &= 2\mathbf{r}_{i_{t-1}}^\top (\mathbf{r}_i - \mathbf{r}_{i_{t-1}}) & (40) \\ &= 2(\mathbf{r}_{i_{t-1}}^\top \mathbf{r}_i - 1) \quad \text{for } i = \{1, 2, 3\}. & (41) \end{aligned}$$

For brevity, we shall omit the explicit dependency of c_{n_i} w.r.t. $\mathbf{r}_i$ and $\mathbf{r}_{i_{t-1}}$, as well as in all constraints mentioned henceforth. This constraint forces the vector $\mathbf{r}_i$ to belong to a tangent plane to the unit sphere, with the plane defined by the normal $\mathbf{r}_{i_{t-1}}$ as displayed in Figure 1.

Orthogonality. The column (and row) vectors of a rotation matrix are all perpendicular to each other. This constraint is expressed as

$$c_{o_{ij}} = \mathbf{r}_i^\top \mathbf{r}_j \quad \text{for } (i, j) = \{(1, 2), (2, 3), (3, 1)\}. \tag{42}$$

The corresponding derivatives at the linearization point are given by

$$\frac{\partial c_{o_{ij}}}{\partial \mathbf{r}_i} = \mathbf{r}_{j_{t-1}}, \quad \frac{\partial c_{o_{ij}}}{\partial \mathbf{r}_j} = \mathbf{r}_{i_{t-1}}, \tag{43}$$

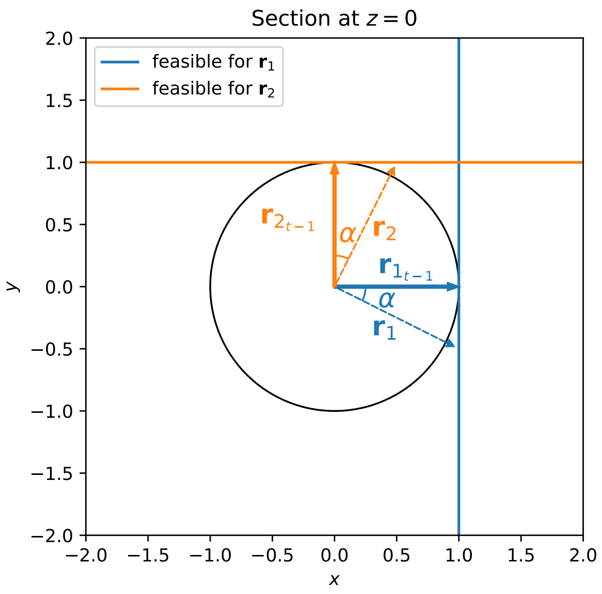

Figure 1. This figure depicts how rotation (matrix) orthogonality constraints manifest themselves in space once they are linearized. The figure shows a cross section at $z = 0$ and only the first two column vectors of the rotation matrices $\mathtt{R}$ and $\mathtt{R}_{t-1}$ are represented. The frame of reference is aligned with the bases of $\mathtt{R}_{t-1}$. The (blue) vertical and (orange) horizontal lines represent the 3D planes created from linearizing the vector's unit norm constraints. They show that the resultant column vectors of the matrix our linear estimator produces, will always have a norm larger or equal to one. Simultaneously, we can also see that the *projections* of $\mathbf{r}_1$ and $\mathbf{r}_2$ to the plane formed by vectors $\mathbf{r}_{1_{t-1}}$ and $\mathbf{r}_{2_{t-1}}$, in this image represented as $z = 0$, will also be orthogonal to each other. However, since the vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ are not restricted to $z = 0$, there is no guarantee that they are orthogonal.

resulting in the following linearized constraint

$$c_{o_{ij}}^{(1)} = \mathbf{r}_{j_{t-1}}^\top \mathbf{r}_i + \mathbf{r}_{i_{t-1}}^\top \mathbf{r}_j \quad \text{for } (i, j) = \{(1, 2), (2, 3), (3, 1)\}. \tag{44}$$

Please refer to Figure 1 for additional insights on how these constraints look in space.

Unit Determinant. The determinant constraint of a rotation matrix is given by

$$\begin{aligned} c_d &= \mathbf{r}_1^\top \lfloor\mathbf{r}_2\rfloor_\times \mathbf{r}_3 - 1 & (45) \\ &= \mathbf{r}_2^\top \lfloor\mathbf{r}_3\rfloor_\times \mathbf{r}_1 - 1 & (46) \\ &= \mathbf{r}_3^\top \lfloor\mathbf{r}_1\rfloor_\times \mathbf{r}_2 - 1. & (47) \end{aligned}$$

We will make use of all three variants to compute the different partial derivatives. The partial derivatives at the linearization point are

$$\frac{\partial c_d}{\partial \mathbf{r}_1} = \lfloor \mathbf{r}_{2_{t-1}} \rfloor_{\times} \mathbf{r}_{3_{t-1}} \tag{48}$$
$$= \mathbf{r}_{1_{t-1}} \tag{49}$$
$$\frac{\partial c_d}{\partial \mathbf{r}_2} = \lfloor \mathbf{r}_{3_{t-1}} \rfloor_{\times} \mathbf{r}_{1_{t-1}} \tag{50}$$
$$= \mathbf{r}_{2_{t-1}} \tag{51}$$
$$\frac{\partial c_d}{\partial \mathbf{r}_3} = \lfloor \mathbf{r}_{1_{t-1}} \rfloor_{\times} \mathbf{r}_{2_{t-1}} \tag{52}$$
$$= \mathbf{r}_{3_{t-1}}, \tag{53}$$

resulting in the following linearized constraint

$$c_d^{(1)} = \sum_{i=1}^{3} \mathbf{r}_{i_{t-1}}^{\top} (\mathbf{r}_i - \mathbf{r}_{i_{t-1}}) \tag{54}$$
$$= \sum_{i=1}^{3} \mathbf{r}_{i_{t-1}}^{\top} \mathbf{r}_i - 3. \tag{55}$$

The determinant constraint can be written as

$$c_d^{(1)} = \frac{1}{2} \sum_{i=1}^{3} c_{n_i}^{(1)}, \tag{56}$$

implying this constraint is linearly dependent and as such it is removed from the final formulation.

Linear Independence Constraint Qualification. In order for the method of Lagrange Multipliers to produce a single solution, our constraints need to verify the Linear Independence Constraint Qualification (LICQ). The LICQ requires all gradients of the constraints to be linearly independent at the optimum $\mathtt{R}^*$, assuming one exists. Since the determinant constraint could be formulated as a linear combinations of the orthogonality ones, it was dropped. If a constraint is linearly dependent, its gradients will also be. If we only consider the (matrix) orthogonality constraints and we stack all their gradients into a single matrix it yields

$$\mathtt{C} = \begin{bmatrix} 2\mathbf{r}_{1_{t-1}} & 0 & 0 & \mathbf{r}_{2_{t-1}} & \mathbf{r}_{3_{t-1}} & 0 \\ 0 & 2\mathbf{r}_{2_{t-1}} & 0 & \mathbf{r}_{1_{t-1}} & 0 & \mathbf{r}_{3_{t-1}} \\ 0 & 0 & 2\mathbf{r}_{3_{t-1}} & 0 & \mathbf{r}_{1_{t-1}} & \mathbf{r}_{2_{t-1}} \end{bmatrix}, \tag{57}$$

where $\mathtt{C} \in \mathbb{R}^{9\times 6}$. Each column $\mathbf{c}_i \in \mathbb{R}^9$ of $\mathtt{C}$, represents a gradient of one of the constraints. These vectors are constant because the constraints are linear. Therefore, if we ensure they are linearly independent, this also holds at the optimum $\mathtt{R}^*$. To guarantee linear independence, the columns of $\mathtt{C}$ need to satisfy the following property:

$$\sum_{i=1}^{6} \alpha_i c_i = 0 \rightarrow \forall 1 \le i \le 6 : \alpha_i = 0. \tag{58}$$

The entries in $\mathtt{C}$ consist of the column vectors of matrix $\mathtt{R}_{t-1} = \begin{bmatrix} \mathbf{r}_{1_{t-1}} & \mathbf{r}_{2_{t-1}} & \mathbf{r}_{1_{t-1}} \end{bmatrix} \in SO(3)$. This matrix is the solution at iteration $t-1$ and hence satisfies the properties in Eqs. (4a) and (4b). Eq. (4a) ensures $\mathtt{R}_{t-1}$ is orthogonal. This implies that its column (and row) vectors are linearly independent and hence we are given that for $\forall \beta_1, \beta_2, \beta_3 \in \mathbb{R}$ with $\beta_1 \mathbf{r}_{1_{t-1}} + \beta_2 \mathbf{r}_{2_{t-1}} + \beta_3 \mathbf{r}_{3_{t-1}} = 0 \Rightarrow \beta_1 = \beta_2 = \beta_3 = 0$. Now assume that we are given such $\alpha_1, \dots, \alpha_6 \in \mathbb{R}$ with $\alpha_1 \mathbf{c}_1 + \dots + \alpha_6 \mathbf{c}_6 = 0$. We obtain the following three equations:

$$2\alpha_1 \mathbf{r}_{1_{t-1}} + \alpha_4 \mathbf{r}_{2_{t-1}} + \alpha_5 \mathbf{r}_{3_{t-1}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{59}$$
$$2\alpha_2 \mathbf{r}_{2_{t-1}} + \alpha_4 \mathbf{r}_{1_{t-1}} + \alpha_6 \mathbf{r}_{3_{t-1}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{60}$$
$$2\alpha_3 \mathbf{r}_{3_{t-1}} + \alpha_5 \mathbf{r}_{1_{t-1}} + \alpha_6 \mathbf{r}_{2_{t-1}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{61}$$

Since we know that the columns vectors $\mathbf{r}_{1_{t-1}}$ with $1 \le i \le 3$ are linear independent, we can deduce that

$$(59) \rightarrow \alpha_1 = \alpha_4 = \alpha_5 = 0 \tag{62}$$
$$(60) \rightarrow \alpha_2 = \alpha_4 = \alpha_6 = 0 \tag{63}$$
$$(61) \rightarrow \alpha_4 = \alpha_5 = \alpha_6 = 0, \tag{64}$$

and thus we obtain that $\forall 1 \le i \le 6 : \alpha_i = 0$.

4. Orthogonalization Singularities

In the paper, we make the claim that during the orthogonalization steps of our *rotation assembler* stage, the estimates produced by our linear estimator will not lie close to the singularities of this operation. We present once more the steps involved into turning the estimates back into rotation matrices. Assume that $\mathtt{R}'$ is a 3×3 input matrix to this stage, resulting from the output of the linear estimator and formed by the columns $\mathbf{r}'_1, \mathbf{r}'_2, \mathbf{r}'_3 \in \mathbb{R}^3$. This matrix is constructed in the following way:

$$\mathbf{r}_1 = \frac{\mathbf{r}'_1}{\|\mathbf{r}'_1\|}, \tag{65}$$
$$\mathbf{r}_2 = \frac{(\mathtt{I}_3 - \mathbf{r}_1 \mathbf{r}_1^{\top}) \mathbf{r}'_2}{\|(\mathtt{I}_3 - \mathbf{r}_1 \mathbf{r}_1^{\top}) \mathbf{r}'_2\|}, \tag{66}$$
$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2. \tag{67}$$

To avoid the singularities associated with the denominator terms of Eqs. (65) and (66), we need to ensure that both $\|\mathbf{r}'_1\| > 0$ and $\|(\mathtt{I}_3 - \mathbf{r}_1 \mathbf{r}_1^{\top}) \mathbf{r}'_2\| > 0$. This is accomplished by guaranteeing that the input column vectors have a positive norm, something that is directly observable in Figure 1, and by guaranteeing that $\mathbf{r}'_1$ and $\mathbf{r}'_2$ are not collinear.

Positive Vector Norm. To show that all column vectors have a positive norm, we recover the linearized norm constraints

$$\mathbf{r}_{i_{t-1}}^\top \mathbf{r}'_i = 1 \quad \text{for } i = \{1,2,3\}. \tag{68}$$

From here the we perform the following manipulation

$$\mathbf{r}_{i_{t-1}}^\top \mathbf{r}'_i = 1 \tag{69}$$

$$\Leftrightarrow \quad -2\mathbf{r}_{i_{t-1}}^\top \mathbf{r}'_i = -2 \tag{70}$$

$$\Leftrightarrow \quad \|\mathbf{r}_{i_{t-1}}\|^2 - 2\mathbf{r}_{i_{t-1}}^\top \mathbf{r}'_i + \|\mathbf{r}'_i\|^2 = \|\mathbf{r}_{i_{t-1}}\|^2 - 2 + \|\mathbf{r}'_i\|^2 \tag{71}$$

$$\Leftrightarrow \quad \|\mathbf{r}'_i - \mathbf{r}_{i_{t-1}}\|^2 = \|\mathbf{r}'_i\|^2 - 1 \tag{72}$$

$$\Leftrightarrow \quad 1 \leq \|\mathbf{r}'_i\|, \tag{73}$$

confirming the intuition from Figure 1.

Non-collinear Vectors. We show that $\mathbf{r}'_1$ and $\mathbf{r}'_2$ cannot be collinear by contradiction. If $\mathbf{r}'_1$ were $\mathbf{r}'_2$ indeed collinear, one would be able to write $\mathbf{r}'_2 = a\mathbf{r}'_1$, with a representing an arbitrary non-null scalar in $\mathbb{R}$. However, if we substitute this relation into the linearized (vector) orthogonality constraints, it yields

$$\mathbf{r}_{2_{t-1}}^\top \mathbf{r}'_1 + \mathbf{r}_{1_{t-1}}^\top \mathbf{r}'_2 = 0 \tag{74}$$

$$\Leftrightarrow \quad \frac{1}{a}\mathbf{r}_{2_{t-1}}^\top \mathbf{r}'_2 + a\mathbf{r}_{1_{t-1}}^\top \mathbf{r}'_1 = 0 \tag{75}$$

$$\Leftrightarrow \quad \frac{1}{a}\underbrace{\mathbf{r}_{2_{t-1}}^\top \mathbf{r}'_2}_{\overset{(68)}{=}1} + a\underbrace{\mathbf{r}_{1_{t-1}}^\top \mathbf{r}'_1}_{\overset{(68)}{=}1} = 0 \tag{76}$$

$$\Leftrightarrow \quad \frac{1}{a} + a = 0 \tag{77}$$

$$\Leftrightarrow \quad a^2 = -1. \tag{78}$$

There is no $a \in \mathbb{R}$ that satisfies the equation above, contradicting the original assumption that $\mathbf{r}'_1$ and $\mathbf{r}'_2$ are collinear.

5. Understanding the Differences Between Estimators

As mentioned in the main paper, 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, a byproduct of the different gradients that our extra layers produce. To monitor gradient differences, we revisit Deep Closest Point [4] (DCP). DCP uses a Transformer architecture [3] to compute point-wise features. To limit the influence of a backpropagation cascading effect, i.e. that a difference in gradient in the last layers causes considerably stronger differences in gradients in the early layers, we restrict our focus to the gradients of parameters of the very last layer of the transformer decoder, namely the mean and standard deviation of a Normalization layer.

Different Gradients. *If Kabsch and our Linear Estimator produce different rotation estimates, then including our layer will produce different gradients.* Without loss of generality, consider the loss function used to train DCP, specifically only the terms that explicitly penalize rotation error. This loss is of the form

$$\mathcal{L}_{\mathtt{R}} = \frac{1}{N_r + 1} \sum_{i=1}^{N_r+1} \|\mathtt{R}_i^\top \mathtt{R}_{gt} - \mathtt{I}_3\|^2, \tag{79}$$

where N_r is the number of refinements performed with the linear estimator. Consider the simple case where $N_r = 1$ and let us denote Kabsch by $f(\mathtt{P}'_t)$ and the Linear Estimator by $g(\mathtt{P}'_t, f(\mathtt{P}'_t))$, omitting variables in both functions that are not dependent on learnable parameters. The matrix $\mathtt{P}'_t$ represents the regressed (target) correspondences by DCP. When Kabsch is used standalone, the gradient will respect the following relationship:

$$\frac{\partial \mathcal{L}_{\mathtt{R}}}{\partial \mathtt{P}'_t} = \frac{\partial \mathcal{L}_{\mathtt{R}}}{\partial f}\frac{\partial f}{\partial \mathtt{P}'_t}. \tag{80}$$

Performing a single refinement of the linear estimator, will modify this gradient to

$$\frac{\partial \mathcal{L}_{\mathtt{R}}}{\partial \mathtt{P}'_t} = \frac{1}{2}\frac{\partial \mathcal{L}_{\mathtt{R}}}{\partial f}\frac{\partial f}{\partial \mathtt{P}'_t} + \frac{1}{2}\frac{\partial \mathcal{L}_{\mathtt{R}}}{\partial g}\left(\frac{\partial g}{\partial \mathtt{P}'_t} + \frac{\partial g}{\partial f}\frac{\partial f}{\partial \mathtt{P}'_t}\right). \tag{81}$$

In situations where the linear estimator produces estimates similar to Kabsch, we have that $g(\mathtt{P}'_t, f(\mathtt{P}'_t)) \approx f(\mathtt{P}'_t)$, yielding

$$\frac{\partial \mathcal{L}_{\mathtt{R}}}{\partial \mathtt{P}'_t} \approx \frac{1}{2}\frac{\partial \mathcal{L}_{\mathtt{R}}}{\partial f}\frac{\partial f}{\partial \mathtt{P}'_t} + \frac{1}{2}\frac{\partial \mathcal{L}_{\mathtt{R}}}{\partial g}\left(0 + \mathtt{I}\frac{\partial f}{\partial \mathtt{P}'_t}\right) \tag{82}$$

$$\approx \frac{\partial \mathcal{L}_{\mathtt{R}}}{\partial f}\frac{\partial f}{\partial \mathtt{P}'_t}. \tag{83}$$

Conversely, we can only have different gradients if the linear estimator produces a different estimate than Kabsch.

To confirm this, we conducted an experiment over a single training epoch, where we measured the relative gradient difference as a function of the amount of divergence error. This experiment uses the same training data as in DCP's unseen categories experiment. To measure gradient difference, we perform a forward and backwards pass without our layer, and store Kabsch's gradient. We then do a forward and backward pass with our layers added, storing also this gradient. We proceed with training using our gradient as the descent direction. To report a relative gradient difference, we compute an *element-wise* relative difference between both gradients, concretely

$$\Delta(\nabla\mathcal{L})_i = \frac{|\nabla\mathcal{L}_{\text{ours}_i} - \nabla\mathcal{L}_{\text{Kabsch}_i}|}{|\nabla\mathcal{L}_{\text{Kabsch}_i}|}. \tag{84}$$

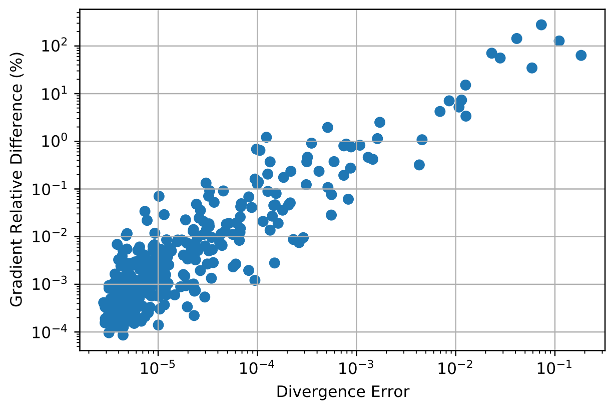

Figure 2. A representation of the gradient relative difference, between employing and not employing our extra layers, as a function of the divergence error between the rotation estimates of the Linear Estimator and Kabsch. This image shows that networks trained with our layers, experience difference gradients whenever the Linear Estimator produces different rotation estimates than Kabsch.

We report the mean relative gradient difference, only for the parameters in the last layer. We compute divergence as the chordal distance between Kabsch's and our estimates:

$$\mathcal{D} = \sum_{i=1}^{5} \|\mathtt{R}_{\text{ours}_i} - \mathtt{R}_{\text{Kabsch}}\|_F. \tag{85}$$

In Figure 2 we can see that as predicted, networks trained with our layers, experience difference gradients whenever the Linear Estimator produces different rotation estimates than Kabsch. In the majority of situations, given a rotation estimate from Kabsch, our estimator will replicate it. 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, something that is also beneficial for Kabsch.

Understanding Divergent Cases. We have established that divergence from Kabsch is what produces different training outcomes, but we have yet to understand under which conditions our method diverges. At the time of writing, we still do not hold a definitive answer that fully identifies the underlying cause, but we have found certain conditions that establish some empirical upper boundaries on whether divergence has a chance to occur. These mostly depend on the geometric relationship between the unconstrained solution to

$$\mathtt{R}_u = \arg\min_{\mathtt{R}} \sum_{i=1}^{N} w_i \|\tilde{\mathbf{p}_{\mathbf{t}}}_i - \mathtt{R}\tilde{\mathbf{p}_{\mathbf{s}}}_i\|^2 \tag{86}$$

and the solution produced by Kabsch, that we shall henceforth designate by $\mathtt{R}_K \in SO(3)$.

It is important to recognize that the optimization problem in Eq. (86) is solving three independent optimization problems. To illustrate that, we employ an alternative formulation of the problem in Eq. (86)

$$\mathtt{R}_u = \arg\min_{\mathtt{R}} \sum_{i=1}^{N} w_i \|\mathtt{R}^\top \tilde{\mathbf{p}_{\mathbf{t}}}_i - \tilde{\mathbf{p}_{\mathbf{s}}}_i\|^2 \tag{87}$$

$$= \underbrace{\left(\sum_{i=1}^{N} w_i \tilde{\mathbf{p}_{\mathbf{t}}}_i \tilde{\mathbf{p}_{\mathbf{t}}}_i^\top\right)^{-1}}_{\mathtt{G}} \underbrace{\left(\sum_{i=1}^{N} w_i \tilde{\mathbf{p}_{\mathbf{t}}}_i \tilde{\mathbf{p}_{\mathbf{s}}}_i^\top\right)}_{\mathtt{F}}, \tag{88}$$

with matrices $\mathtt{R}_u, \mathtt{F} \in \mathbb{R}^{3\times 3}$ and $\mathtt{G} \in \mathbb{S}^3$. Each column of matrix $\mathtt{R}$ will independently pick the best location in 3D that will minimize its correspondence error. Each column $\mathbf{r}_{u_i} \in \mathbb{R}^3$ of matrix $\mathtt{R}_u$ is given by

$$\mathbf{r}_{u_i} = \mathtt{G}^{-1}\mathbf{f}_i \tag{89}$$

where $\mathbf{f}_i \in \mathbb{R}^3$ is the i-th column of $\mathtt{F}$. So each column $\mathbf{r}_{u_i}$ has its own distinct loss landscape in $\mathbb{R}^3$ that can be expressed as

$$\mathcal{L}_{\mathbf{r}_{u_j}} = \sum_{i=1}^{N} w_i (\tilde{\mathbf{p}_{\mathbf{t}}}_i^\top \mathbf{r}_j - \tilde{\mathbf{p}_{\mathbf{s}}}_{ij})^2 \quad \text{with } 1 \le j \le 3. \tag{90}$$

Note the usage of index j to represent the column index, to avoid a clash with index i that denotes correspondences. The level set surfaces of these loss landscapes form quadrics in 3D space as exemplified in Figure 3.

Figure 3 shows an example where the linear estimator produces an estimate that does not diverge from Kabsch's estimate. In contrast, in Figure 4 we show a particular example where the linear estimator diverged considerably. This example captures some of the representative aspects that cause the linear estimator to diverge: the unconstrained solutions is considerably far away from Kabsch's estimate, usually a consequence of level set quadrics that are close to being degenerate due to the target point cloud not spanning full 3D space; some of the unconstrained solution column vectors are close to being orthogonal to their corresponding column vectors in Kabsch's estimate. We explore these two cases further.

Distance between the Unconstrained and Kabsch's Solutions. We conducted an experiment over a single training epoch, where we measured the maximum distance between Kabsch's and the unconstrained solutions, across all three column vectors. This experiment uses the same training data as in DCP's unseen categories experiment. We map the divergence error from Eq. (85) to the maximum distance between both solutions, computed as

$$d_{\text{unc, Kabsch}} = \max_i \|\mathbf{r}_{u_i} - \mathbf{r}_{K_i}\|, \tag{91}$$

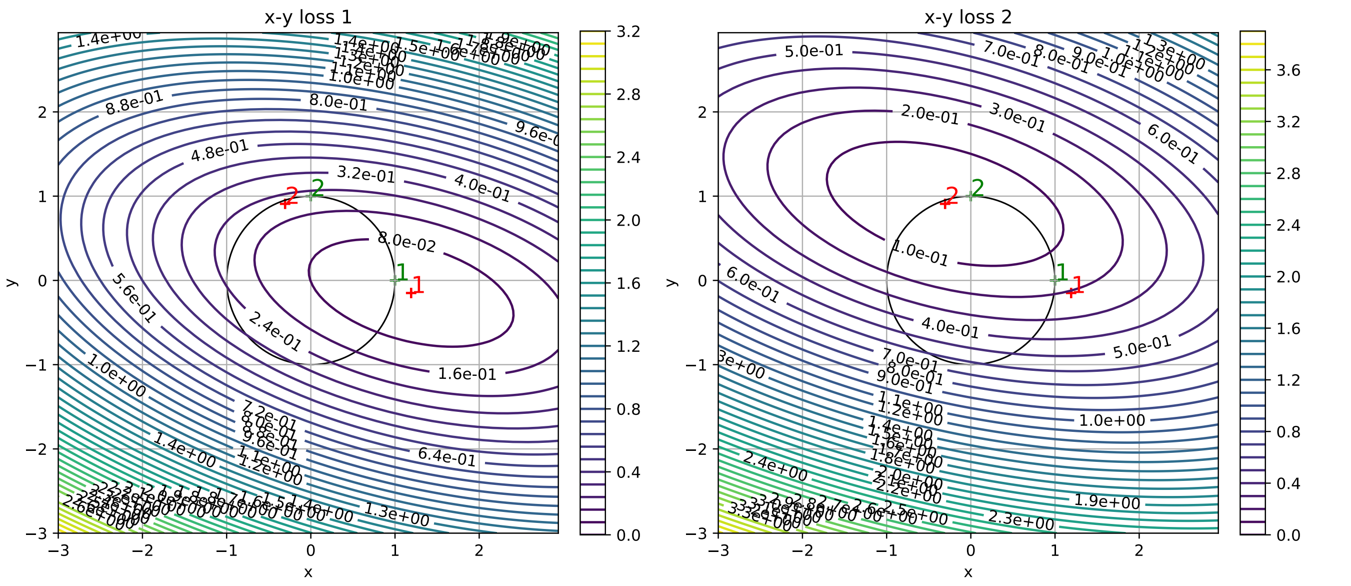

Figure 3. A cross-section of the loss landscape for the first two column vectors of $\mathtt{R}$. *left:* Loss landscape for the column vector $\mathbf{r}_1$. *right:* Loss landscape for the column vector $\mathbf{r}_2$. The figure's axes are aligned with the initialization $\mathtt{R}_{t-1}$ provided to the linear estimator. The numbers 1 and 2 indicate the location of each column vector, with red representing the solution of the unconstrained problem $\mathbf{r}_{u_i}$ and green the solution from Kabsch $\mathbf{r}_{K_i}$. The level set surfaces form quadrics in 3D space. This is a particular example in which the linear estimator does not diverge from the Kabsch's estimate.

and show it in Figure 5. We can see that this distance establishes a practical upper bound on the divergence error, that increases when the distance between both solutions also increases. In Figure 6, we validate our claim of the strong correlation between the unconstrained solution distance and how this usually manifests itself when the level set quadrics are close to being degenerate.

Angle between the Unconstrained and Kabsch's Solutions. We conducted an experiment over a single training epoch, where we measured the maximum angle between Kabsch's and the unconstrained solutions, across all three column vectors. This experiment also relies on the same training data as in DCP's unseen categories experiment. We map the divergence error from Eq. (85) to the maximum angle between both solutions, computed as

$$\angle_{\text{unc, Kabsch}} = \max_i \frac{180}{\pi} \arccos\left(\left| \frac{\mathbf{r}_{u_i}}{\|\mathbf{r}_{u_i}\|}^\top \frac{\mathbf{r}_{K_i}}{\|\mathbf{r}_{K_i}\|} \right| \right), \quad (92)$$

and show it in Figure 7. We can see the angle also establishes a practical upper bound on the divergence error, that increases when the angle between both solutions also increases.

6. Additional Experiments

6.1. Deep Closest Point with ModelNet40 Data

Alignment for identical point clouds. In Table 1, we report the point cloud registration results obtained on the instances of CAD models that were held-out during the model training. In this setting, we are aligning identical point clouds; therefore, perfect correspondences between them exist. As can be seen, with the addition of our refinements, we obtain a performance that is on-par with the original method. We marginally improve the rotation error by 0.07% at the marginal cost of translation accuracy of 0.19%, when normalized by the maximum magnitude of the rotation and translations sampled. When the problems is simple for the underlying network, our method does not provide any assistance, but it will not make the results worse. This allows to blindly use it as an add-on that can only improve performance. In the experiments presented in the main paper, we show that in more challenging scenarios, our layer provides more meaningful improvements.

6.2. RPM-Net with ModelNet40 Data

Alignment under Gaussian noise. We sample a different set of 1024 points (from the original 2048) for each point cloud and add Gaussian noise $\mathcal{N}(0, 0.01^2)$ independently to both, clamped at $[-0.05, 0.05]$. We show the results in Table 2. In this experiment, we do not provide any measur-

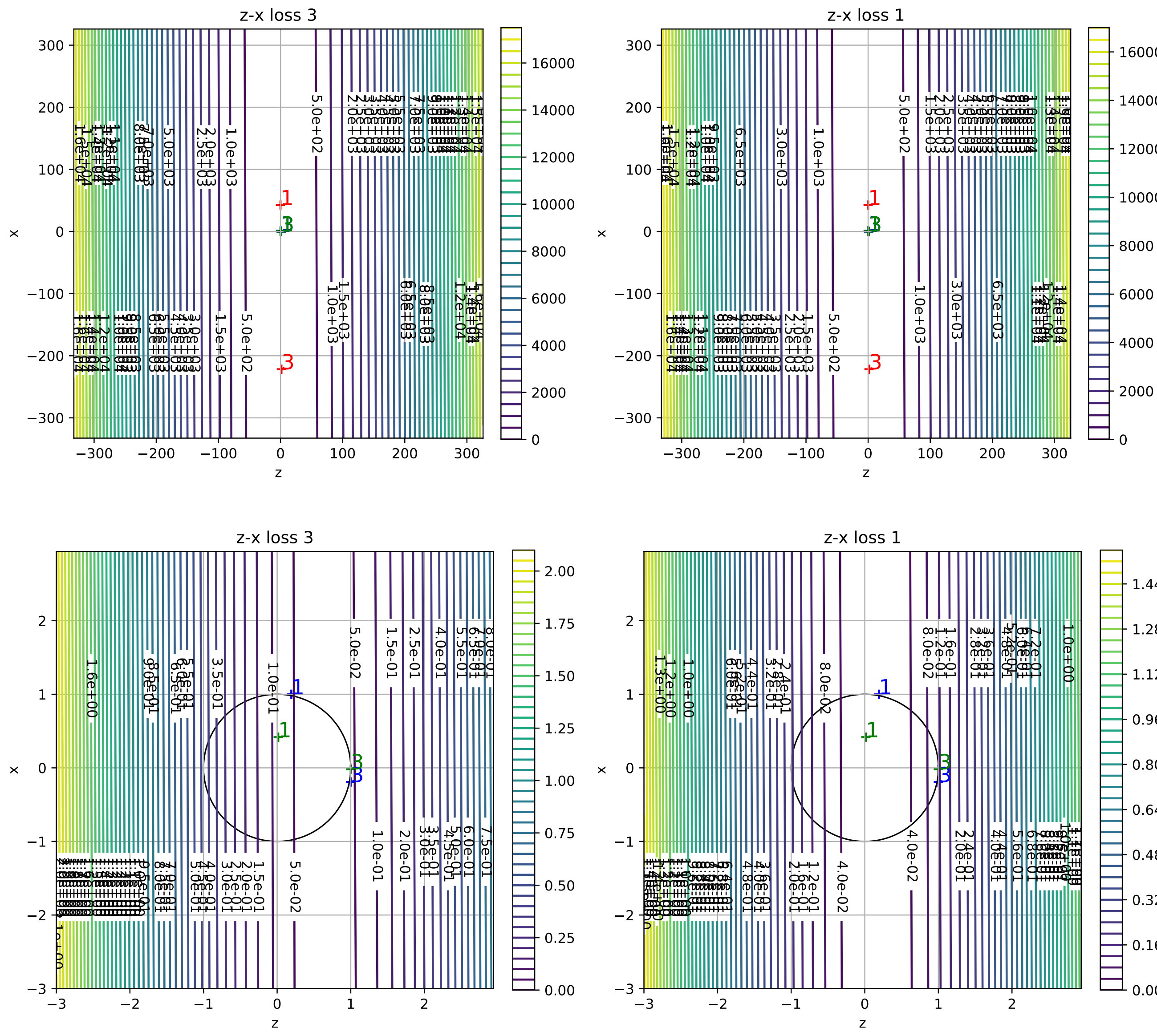

Figure 4. A cross-section of the loss landscape for the first and third column vectors of $\mathtt{R}$. *top-left:* Loss landscape for the $\mathbf{r}_3$. *top-right:* Loss landscape for the $\mathbf{r}_1$. *bottom-left:* Loss landscape for the $\mathbf{r}_3$. A zoomed-in view of top-left. *bottom-right:* Loss landscape for the $\mathbf{r}_1$. A zoomed-in view of top-right. The numbers 1 and 3 indicate the location of each column vector, with red representing the solution of the unconstrained problem $\mathbf{r}_{u_i}$, blue representing the solution of the linear estimator $\mathbf{r}_i$ and green the solution from Kabsch $\mathbf{r}_{K_i}$. The figure's axes are aligned with the initialization $\mathtt{R}_{t-1}$ provided to the linear estimator, and explain why the solution from Kabsch is no longer contained inside the unit circle. In this particular example, the linear estimator diverges from Kabsch's estimate. Contrary to Figure 3, note how the unconstrained solution is considerably distant from Kabsch's estimate and how the level set curves appear to be almost parallel, usually a sign that the correspondences in the target point cloud are close to being distributed along a linear subspace in 3D space, e.g. a plane or a line. In this image we can also confirm that the solutions from the linear estimator respect the constraints shown in Figure 1.

able improvement to the baseline method. Similar to DCP, this happens when the correspondence problem is too simple. However, the results encourage the idea that we do not incur a penalty in including our layer and that it can be blindly applied as an add-on. We also evaluated increasing the magnitude of Gaussian error at test time, but both approaches produce similar results.

6.3. Ablation Studies

Our method is governed by two critical design choices: how many refinement iterations should we conduct and whether to impose a loss in all poses output by our method

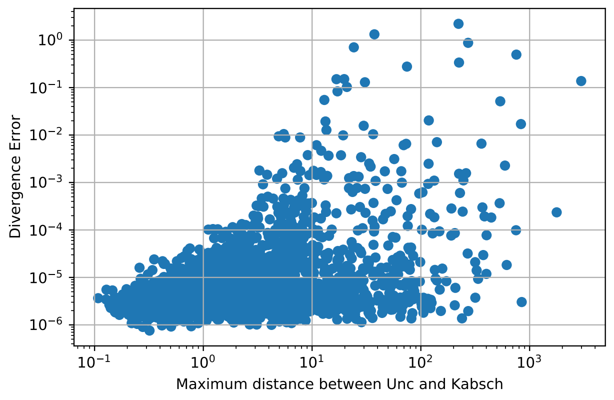

Figure 5. A representation of the divergence error, as a function of the maximum distance across all column vectors, between Kabsch's and the unconstrained solution. The distance establishes a practical upper bound on the divergence error, that increases when the distance between both solutions also increases.

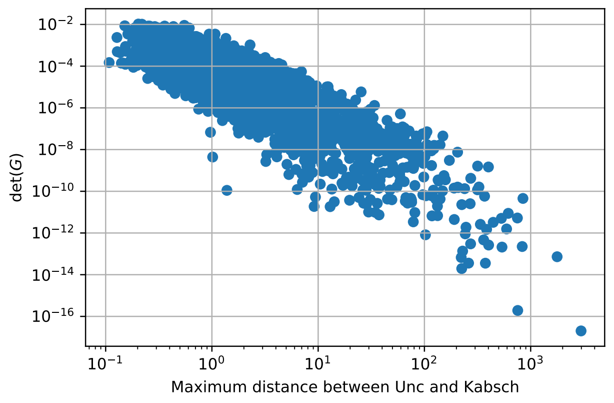

Figure 6. A representation of $\det(\mathtt{G})$, as a function of the maximum distance across all column vectors, between Kabsch's and the unconstrained solutions. The closer the points in the target point cloud are to span only a linear subspace of 3D, like a plane or a line, the closer the $\det(\mathtt{G})$ will be to the value 0. When that happens, our the level set quadrics approximate degeneracy and the unconstrained solution shifts considerably in space.

or only in the last one. In this section we show how both these decisions affect registration performance. We evaluate the choice of performing 1, 2, 5 and 10 refinement iterations. We carry-over the experimental setup from DCP with unseen categories and from RPM-Net on partially visible data with noise. We report RMSE for both rotation and translation for these variants. We report results for DCP in Table 3 and for RPM-Net in Table 4. In both cases, applying the pose loss to all poses produced by the network, in conjunction with performing 5 iterations of our method produces the best pose error.

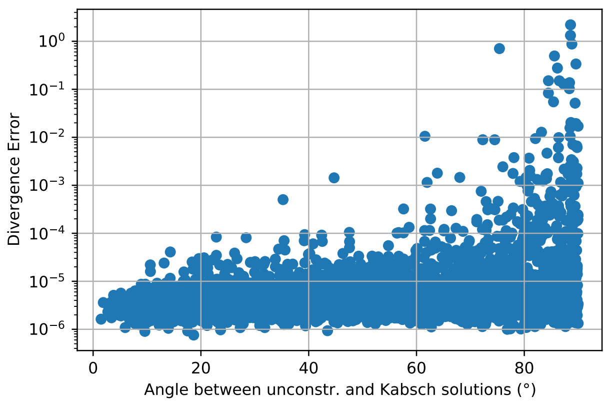

Figure 7. A representation of the divergence error, as a function of the maximum angle across all column vectors, between Kabsch's and the unconstrained solution. Similar to Figure 5, the angle also establishes a practical upper bound on the divergence error, that increases when the angle between both solutions also increases.

Model	RMSE($\mathtt{R}$)°	MAE($\mathtt{R}$)°	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)
ICP	29.914835	23.544817	0.290935	0.248755
Go-ICP [5]	11.852313	2.588463	0.025665	0.007092
FGR [6]	9.362772	1.999290	0.013939	0.002839
PointNetLK [1]	15.095374	4.225304	0.022065	0.005404
DCP-v2	1.093971	**0.751517**	**0.001717**	**0.001173**
DCP-v2 + ours	**1.063893**	0.760524	0.002677	0.001865

Table 1. Deep Closest Point on ModelNet40: Test on unseen point clouds with perfect correspondences. We marginally improve the rotation error by 0.07% at the marginal cost of translation accuracy of 0.19%, when normalized by the maximum magnitude of the rotation and translations sampled. The problem is too simple for our layer to provide meaningful improvements to the baseline, but it will not make the results worse, allowing it to blindly used as an add-on that can only improve match or improve performance.

7. Measuring Correspondence Improvement

In the main paper, we made the claim that evaluating correspondence quality simply based on the Euclidean distance can be misleading. This is because improvements in this metric do not necessarily translate in improvements in pose. In fact, it is possible to engineer a particular case that for a higher average Euclidean distance error, the network produces a better pose estimate. In light of this, we make the argument that only pose error can accurately represent a measure of correspondence quality improvement, for the task of point cloud registration. To establish an initial intuition behind our claims we refer to Figure 8, also present in the main paper. In here, we show a cherry picked example where the pose error is particularly small. Contrary to intuition, the regressed target points (red) hardly resemble the ground truth target points (black) and yet the net-

Method	Anisotropic err. (Rot.)	(Trans.)	Isotropic err. (Rot.)	(Trans.)	$\tilde{CD}$
ICP	3.414	0.0242	6.999	0.0514	0.00308
RPM	1.441	0.0094	2.994	0.0202	0.00083
FGR	1.724	0.0120	2.991	0.0252	0.00130
PointNetLK	1.528	0.0128	2.926	0.0262	0.00128
DCP-v2	4.528	0.0345	8.922	0.0707	0.00420
RPM-Net	0.343	**0.0030**	**0.664**	**0.0062**	**0.00063**
RPM-Net + Ours	**0.342**	**0.0030**	**0.664**	**0.0062**	**0.00063**

Table 2. RPM-Net on ModelNet40: Performance on data with Gaussian noise. The Chamfer distance using groundtruth transformations is 0.00055. Our network provides no improvement in this case because the problem is simple for the matching network, but it also does not hinder performance.

Iters. / Loss	$\Delta\mathbf{p}$	$\Delta\mathtt{R}_{\text{ani}}$	$\Delta\mathbf{t}$
1 / all	0.491894	5.472425	0.005496
2 / all	0.491909	5.573030	0.007119
5 / all	**0.491878**	**2.051718**	**0.004543**
10 / all	0.492074	5.558087	0.014462
1 / last	0.491913	3.791935	0.006351
2 / last	0.491890	2.485598	0.004585
5 / last	0.491963	4.859206	0.009861
10 / last	0.492326	6.622592	0.021234

Table 3. Ablations on the Deep Closest Point unseen categories experiment. We evaluate the influence of the number of refinement iterations used, as well as the effect of employing a loss term to all or just the last pose produced by the combination of Kabsch and our method. We present results for the mean point distance $\Delta\mathbf{p}$, and RMSE for rotation $\Delta\mathtt{R}_{\text{ani}}$ and translation $\Delta\mathbf{t}$ ($\mathcal{L}_2$ norm) errors.

Method	Anisotropic err. (Rot.)	(Trans.)	Isotropic err. (Rot.)	(Trans.)	$\tilde{CD}$
1 / all	0.8944	0.00877	1.704	0.0184	0.00089
2 / all	0.8851	0.00861	1.686	0.0183	0.00091
5 / all	**0.8318**	**0.00805**	**1.577**	**0.0169**	**0.00085**
10 / all	0.8473	0.00815	1.604	0.0172	**0.00085**
1 / last	0.8798	0.00833	1.661	0.0175	0.00087
2 / last	0.8792	0.00835	1.695	0.0176	**0.00085**
5 / last	0.8570	0.00843	1.634	0.0177	0.00087
10 / last	0.8876	0.00839	1.687	0.0177	0.00087

Table 4. Ablation RPM-Net on ModelNet40: Performance on partially visible data with noise. The Chamfer distance using groundtruth transformations is 0.00055.

work is still able to estimate an almost perfect pose. This confirms that a seemingly high positional error between regressed and ground truth target points does not imply a bad pose estimate. In fact, Figure 8 suggests that in order to retrieve an accurate pose estimate, it only matters that the centroids and principal directions of both point sets are relatively consistent.

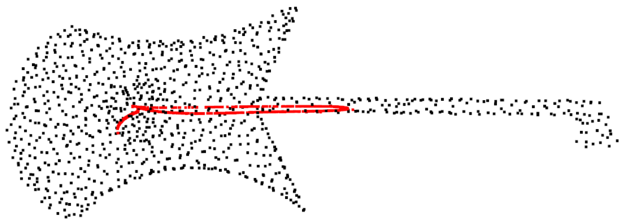

Figure 8. A qualitative example of the how correspondences are generated by Deep Closest Point. In the image we see point clouds of two different colors: black and red. We cherry-pick an example with the lowest pose estimation error, $\angle\Delta\mathtt{R}_{\text{iso}} = 0.2712°$, $\Delta\mathbf{t} = 0.0002$. **Black**: Point cloud generated by applying the groundtruth 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 $\mathbf{p}'_{t_i}$.

Recall that 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 as input to Kabsch. The Kabsch algorithm [2] provides a closed-form to the problem in Eq. (3). Given correspondences, Kabsch computes a globally optimal solution via SVD, as follows:

$$\mathtt{H} = \sum_{i=1}^{N} w_i \tilde{\mathbf{p}'_{\mathbf{t}}}_i \tilde{\mathbf{p}_{\mathbf{s}}}_i^\top \tag{93}$$

$$\mathtt{U}, \mathtt{S}, \mathtt{V} = \operatorname{svd}(\mathtt{H}) \tag{94}$$

$$\mathtt{R} = \mathtt{U}\operatorname{diag}([1, 1, \det(\mathtt{U}\mathtt{V}^\top)])\mathtt{V}^\top. \tag{95}$$

The operator diag( ) produces a diagonal square matrix, in which the input vector represents the diagonal. To produce a correct rotation estimate, it is not necessary that $\forall i : \|\tilde{\mathbf{p}}'_{t_i} - \mathtt{R}\tilde{\mathbf{p}}_{s_i}\| = 0$. Furthermore, 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, because these positive scalars will be "absorbed" by the diagonal matrix of singular values $\mathtt{S}$.

To further stress this idea, consider a problem composed of (already centered) point clouds $\tilde{\mathtt{P}}'_t, \tilde{\mathtt{P}}_s \in \mathbb{R}^{N\times 3}$, with each row $\tilde{\mathbf{p}}'_{t_i}, \tilde{\mathbf{p}}_{s_i} \in \mathbb{R}^3$ representing a corresponding point, for which we already have extracted the optimal rotation $\mathtt{R}$ using Kabsch. Let us define the mean squared correspondence error as

$$d_0 = \frac{1}{N}\sum_{i=1}^{N} \|\tilde{\mathbf{p}}'_{t_i} - \mathtt{R}\tilde{\mathbf{p}}_{s_i}\|^2. \tag{96}$$

From the previous paragraph, we know that if we multiply $\tilde{\mathtt{P}}'_t$ by $a \in \mathbb{R}_+$, the optimal rotation that minimizes the corre-

Metric	DCP	DCP + Ours
Abs. Rot. (°)	10.565127	**8.030258**
Rel. Rot. (%)	27.130154	**20.763237**
Abs. Trans.	**0.005020**	0.005533
Rel. Trans. (%)	**1.188571**	1.303502
Abs. Corr.	0.522025	**0.515820**
Rel. Corr. (%)	96.776123	**95.725166**

Table 5. Mean absolute and relative rotation (Rot.), translation (Trans.) and correspondence position (Corr.) errors, averaged over all data samples in ModelNet40's testing set. DCP represents the network trained with its original architecture using the pretrained model supplied the authors of the paper. DCP + Ours represents a network trained with our proposed layer after the Kabsch. We show that a 1% improvement in correspondence error results in a 7% improvement in rotation error.

spondence error remains unchanged. We are now interested in finding the mean squared correspondence error with this new scaled point cloud.

$$d_a = \frac{1}{N}\sum_{i=1}^{N} \|a\tilde{\mathbf{p}}'_{t_i} - \mathtt{R}\tilde{\mathbf{p}}_{s_i}\|^2 \tag{97}$$

$$= \frac{1}{N}\sum_{i=1}^{N} a^2\tilde{\mathbf{p}}_{t_i}'^{\top}\tilde{\mathbf{p}}'_{t_i} - 2a\tilde{\mathbf{p}}_{t_i}'^{\top}\mathtt{R}\tilde{\mathbf{p}}_{s_i} + \tilde{\mathbf{p}}_{s_i}^{\top}\mathtt{R}^{\top}\mathtt{R}\tilde{\mathbf{p}}_{s_i} \tag{98}$$

$$= d_0 + \frac{1}{N}\sum_{i=1}^{N}(a^2-1)\tilde{\mathbf{p}}_{t_i}'^{\top}\tilde{\mathbf{p}}'_{t_i} - 2(a-1)\tilde{\mathbf{p}}_{t_i}'^{\top}\mathtt{R}\tilde{\mathbf{p}}_{s_i} \tag{99}$$

$$= d_0 + \frac{a^2-1}{N}\sum_{i=1}^{N}\tilde{\mathbf{p}}_{t_i}'^{\top}\tilde{\mathbf{p}}'_{t_i} - \frac{2}{a+1}\tilde{\mathbf{p}}_{t_i}'^{\top}\mathtt{R}\tilde{\mathbf{p}}_{s_i} \tag{100}$$

$$= d_0 + \underbrace{\frac{a^2-1}{N}\sum_{i=1}^{N}\tilde{\mathbf{p}}_{t_i}'^{\top}\left(\tilde{\mathbf{p}}'_{t_i} - \frac{2}{a+1}\mathtt{R}\tilde{\mathbf{p}}_{s_i}\right)}_{\Delta d}. \tag{101}$$

From Eq. (101), one can see that as long as all points $\tilde{\mathbf{p}}_{s_i}$ are finite, we will always be able find a large enough a that ensures $\Delta d > 0$. In fact, we can make Δd arbitrarily large, effectively increasing the mean squared correspondence error by an arbitrary amount, without incurring in any additional pose error.

Despite the arguments presented, in the interest of completeness, we evaluate the quality of correspondences based on the average point distance, when DCP is trained with and without our layers. We revisit the DCP's Gaussian noise experiment, where noise is added independently to one of the point clouds at test time. We present results for mean correspondence position, isotropic rotation, and translation errors in Table 5. The relative errors are normalized w.r.t. ground-truth values. Despite the seemingly marginal improvement in correspondence error, this produces a significant improvement in the quality of pose estimated. We improve the rotation error by 7% just from a 1% improvement in correspondence error.

References

[1] 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. 10

[2] John C Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975. 11

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 6

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

[5] 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. 10

[6] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *Eur. Conf. Comput. Vis.*, 2016. 10