Minimal Solvers for 3D Scan Alignment with Pairs of Intersecting Lines

André Mateus[1], Srikumar Ramalingam[2], and Pedro Miraldo[1]
[1]Instituto Superior Técnico, Lisboa [2]Google

Abstract

We explore the possibility of using line intersection constraints for 3D scan registration. Typical 3D registration algorithms exploit point and plane correspondences, while line intersection constraints have not been used in the context of 3D scan registration before. Constraints from a match of pairs of intersecting lines in two 3D scans can be seen as two 3D line intersections, a plane correspondence, and a point correspondence. In this paper, we present minimal solvers that combine these different type of constraints: 1) three line intersections and one point match; 2) one line intersection and two point matches; 3) three line intersections and one plane match; 4) one line intersection and two plane matches; and 5) one line intersection, one point match, and one plane match. To use all the available solvers, we present a hybrid RANSAC loop. We propose a non-linear refinement technique using all the inliers obtained from the RANSAC. Vast experiments with simulated data and two real-data data-sets show that the use of these features and the combined solvers improve the accuracy. The code is available at `https://github.com/3DVisionISR/3DMinRegLineIntersect`.

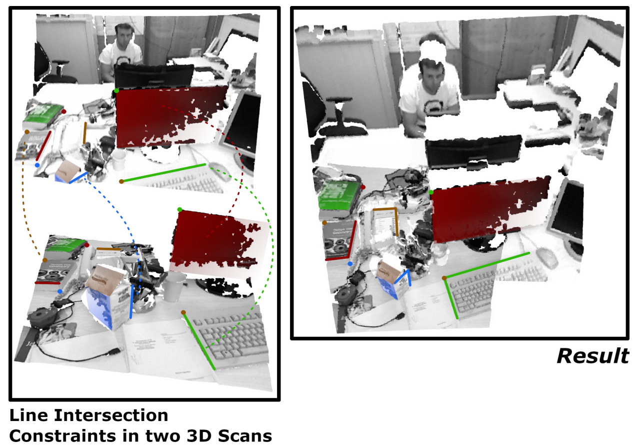


Figure 1: *Basic idea of this paper: We identify pairs of 3D lines that intersect from two scans, i.e., a line from the first scan (image on the top left) intersects with the line from the second (image on the bottom left) in the common reference frame (image on the right). In this paper, we jointly exploit this novel line intersection constraints along with traditional point and plane correspondence constraints to develop a family of minimal solvers for the task of 3D scan registration.*

1. Introduction

3D sensors such as RGB-D and 3D LiDAR devices are becoming less costly and more accurate. Since they provide more information about the scene than perspective cameras, they are becoming more useful in applications such as augmented reality [26, 3, 7, 15], navigation [29, 59, 40, 25], and SLAM [65, 67, 28, 69]. A problem that naturally arises is the 3D registration, which aims at finding the rigid transformation that aligns pairs of point clouds.

The 3D registration problem is usually solved by using iterative techniques, such as the well known ICP [8, 4]. Incremental improvements on ICP have been proposed for almost thirty years. The majority of them attempt to improve robustness to outliers, to reduce the chances of falling into local minima, and to reduce the computational complexity of the problem. A way of dealing with these issues if to use minimal solvers [46] and RANSAC [18]. These consist in finding a consensus for the registration that is obtained by randomly sampling and fitting minimum sets of data. Both techniques are prone to drift when applied to the visual odometry problem. To minimize the drift, researchers use methods like rotation/transformation averaging (see for example [21, 23, 5]) or loop closure (e.g. [24, 63, 73, 17, 10, 46]), that require more than two 3D scans. In this work, we focus on just a pair of scans.

Most existing methods exploit the use of points in the point cloud, either using all points from the 3D scan (e.g., the ICP [4]) or using 3D descriptors (such as the FPFH [61]). This paper aims at using other types of 3D features. We focus on environments containing 3D planes and straight lines, as shown in Fig. 1. Then, the questions we aim at exploring are the following. What kind of constraints can we obtain from pairs of intersecting lines in 3D scans? Does the use of pairs of line intersections improve

3D registration? In this paper, we are particularly interested in robust techniques, which require the development of minimal solvers, a RANSAC loop, and a non-linear refinement technique. Thus, we propose novel minimal solvers that combine pairs of point and plane correspondences with line intersections. The list of contributions are:

- The use of pairs of 3D intersecting lines for 3D scan alignment;
- Five minimal solvers for the cases of mixing line intersections with plane and point matches;
- A hybrid RANSAC scheme to account for all possible combination of minimal sets; and
- A non-minimal refinement solver that, using the inliers from the RANSAC loop, refines the 3D registration.

We test our methods with simulated data and two different available data-sets (SUN3D and TUM [74, 65]). With the former, we validate the solvers and show the merits of mixing different types of 3D features. Real data is used to compare our method with the baselines, revealing that the use of pairs of line intersections improves the results.

2. Related Work

We present some of the existing techniques in 3D registration and minimal solvers. Survey papers in [80, 68].

3D Registration: The standard method is the Iterative Closest Point (ICP) [4], that proposes a method that alternates between estimating the closest points and computing the registration. Many alternatives have been presented, e.g., [76, 60, 9, 19, 70, 43, 47, 36, 75]. Most of these alternatives aim at minimizing some of the ICP issues, namely improving the results in the presence of outliers and ensuring a global minimum. In the last decade, oher methods have been proposed. [50] presents a branch-and-bound method for 3D registration with guarantees of global optimality. KinectFusion, [26], focused on getting accurate and real-time registration, in complex and arbitrary indoor scenes and variable lighting conditions. [10] combines geometric registration of scene fragments with robust global optimizationThe Super4PCS method [41], that aims at getting a robust solution to the registration, is an extension of the 4PCS [1]. It has linear time complexity (*vs.* quadratic for the 4PCS) in the number of data points. The authors use sets of four points and co-planar constraints. Fast Global Registration method (FGR), [78], proposes a technique for outliers removal from a single objective function. The method gets the registration by partially overlapping 3D surfaces. Recently, [52] optimizes a joint photometric and geometric objective to estimate the 3D scan alignment.

Some works use/combine different types of 3D features. For example, [67, 57] presents a method that combines points and planes, [59, 20] proposes a method for point cloud registration with plane to plane matches, and [58] solves the problem with curves and surfaces.

Minimal solvers: Minimal solvers are one of the active research topics in computer vision. From their use in RANSAC frameworks, they have proven to be one of the more successfully strategies for robust estimation: odometry/relative camera pose, e.g. [48, 35, 64, 71, 45], and localization/camera pose, such as [30, 34, 33, 6]. The use of RANSAC has been proving its efficiency in 3D registration before. For example, 4PCS and Super4PCS [1, 41] use sets of four points within a RANSAC framework for robust estimation. A more basic and standard pipeline is to consider sets of three points [62, 46], which is the minimum set of point correspondences required to get the transformation. [46] derive minimal solvers for mini loop closures in 3D scan alignment. The authors show that combining correspondences from a cycle of 3D scans improve the overall accuracy of the 3D registration.

Deep learning methods: Many researchers have spent much effort in solving computer vision problems with Deep Neural Networks (DNNs) [31]. In the last few years, we have witnessed an increasing interest in DNNs for 3D registration. Some approaches consist of using DNNs to extract point features, which is then followed by a RANSAC loop to retrieve the pose. The network architectures include auto-encoders in [16, 12], and attention mechanisms in [27]. An extension of [12] to account for pose invariants is proposed in [13]. An approach for multiple view feature extraction is presented in [77]. A second type of approach consists of training an end-to-end network to estimate the pose. In [2] the PointNet [55] is coupled with the Lukas-Kanade algorithm in a single network. A three-part network consisting of a point cloud embedding model, an attention-based model, and a differential SVD layer is presented in [72]. In [51], the authors present a DNN to classify 3D input correspondences as inliers/outliers, while computing the 3D registration at the same time. An SVD layer is also used in [39], where instead of matching point features, they are generated from learned features. DeepMapping, [14], presents a multiple view registration problem as binary occupancy classification, by using two networks for both pose estimation and modeling the scene structure.

3. Registration with Pairs of Line Intersections

Consider 3D scans, obtained either from RGB-D or LiDAR sensors, taken in human-made scenarios. These usually contain planes that intersect each other, generating 3D straight lines. Now, contrarily to point matches, the use of 3D straight lines and planes benefits from the possibility of these being estimated from a set of 3D points. Noise can then be minimized by, for example, using least-squares or RANSAC for 3D line/plane fitting utilizing a set of 3D points. Besides, in contrast to line images, the use of 3D line intersections provides not only more information about the scene being observed but also more geometric constraints.

Now, consider pairs of 3D line intersections matches in a pair of 3D scans (see Fig. 1). In this paper, we are not assuming a one to one match of 3D straight lines in the pair of 3D scans; we only assume that the pair of intersecting lines are coplanar and intersect in the same 3D point in both scans. Each pair of lines generates two intersection constraints between the 3D scans. The first 3D line in the first scan must intersect the second line in the second scan, and *vice versa*. Besides, matches of line intersection pairs generate the following geometric features: 1) a 3D point that results from the intersection of the lines; and 2) a 3D plane defined by the two intersecting lines. These two geometric features generate two additional types of correspondences between the two 3D scans: points and plane matches. To summarize, in this paper, each pair of intersecting lines between two scans generates three different types of constraints: 1) two line intersections; 2) one point correspondence; and 3) one plane correspondence.

This paper aims at combining the different types of features for 3D registration. For the use of 3D point and plane features that result from the 3D line intersections, we can use state-of-the-art techniques [62, 46], and get a single solution to the relative transformation. We denote the use of three 3D point matches for registration as 3Q. Another problem in which we can use state-of-the-art techniques is the line intersection constraints in the two scans. Consider that we have a set of 3D straight lines in one 3D scan that has to intersect a set of 3D lines in the second scan. Finding the transformation that aligns both sets of lines, such that they intersect in the world, is equivalent to the relative pose estimation for general camera models [22, 44, 56, 53, 66]. [64] proposes a solution to the minimal case that requires sets of six 3D lines in both scans and can get up to 64 solutions. This method is denoted in the paper as 6L.

In addition to the use of 3Q and 6L, we propose five minimal solvers for the combination of line intersections, point matches, and plane correspondences (Sec. 5):

- 3L1P: 3 line intersections and 1 plane match;
- 1L2P: 1 line intersection and 2 plane matches;
- 3L1Q: 3 line intersections and 1 point match;
- 1L2Q: 1 line intersection and 2 point matches; and
- 1L1Q1P: 1 line intersection and 1 plane & 1 point matches.

Table 1 shows the list of minimal solvers that we use for 3D registration. To use all these solvers, we present a hybrid RANSAC scheme (Sec. 6). In Sec. 7, we propose a nonlinear refinement method that uses the inliers and the initial guess given by the RANSAC. In Sec. 8, experiments using synthetic and real data-sets show that the line intersection constraints in a RANSAC loop improve the results when compared to the baselines.

Solver	#Lines	#Points	#Planes	Total	#Sol	Paper
6L	6	0	0	6	64	[64]
3Q	0	3	0	3	1	[62, 46]
3L1P	**3**	**0**	**1**	**4**	**4**	**Ours**
1L2P	**1**	**0**	**2**	**3**	**1**	**Ours**
3L1Q	**3**	**1**	**0**	**4**	**8**	**Ours**
1L2Q	**1**	**2**	**0**	**3**	**2**	**Ours**
1L1Q1P	**1**	**1**	**1**	**3**	**2**	**Ours**

Table 1: *Available and proposed solvers for the 3D scan alignment using line intersection constraints.*

4. Notations

We use *Plücker* coordinates to represent 3D lines [54], i.e., $l \in \mathbb{R}^6 \dot{\sim} [\bar{l}\ \hat{l}]$, where $\bar{l}$ and $\hat{l}$ are the line's direction and moment, respectively. Planes are represented by a four-tuple $\pi \in \mathbb{R}^4 \doteq [\overline{\pi}\ \widetilde{\pi}]$, in which $\overline{\pi}$ and $\widetilde{\pi}$ are the plane's normal direction and distance to the origin, respectively. 3D points are denoted by $\mathbf{q} \in \mathbb{R}^3$. Subscripts denote the number of the features (e.g., l_i denote the i^{th} 3D line), and the apostrophes are used to identify the matches of features (e.g., the tuples (l_1, l_1') and $(\mathbf{q}_1, \mathbf{q}_1')$ represent the pairs of 3D lines and points, in the first and second frames).

We aim at estimating a rigid transformation $T \doteq (R, \mathbf{t})$ that aligns 3D scans. $R \in \mathcal{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$ denote the rotation and translation unknowns. The goal is to combine the use of 3D line intersections and 3D point & plane matches. Consider the tuple (l_i, m_i') representing two intersecting lines in 3D. The constraint that ensures they intersect in the world was derived in [53]:

$$m_i'^{\,T} \begin{bmatrix} -[\mathbf{t}]_{\text{x}} R & R \\ R & 0 \end{bmatrix} l_i = 0. \tag{1}$$

5. Minimal Solvers

We present new minimal solvers for 3D scan alignment using line intersections and plane & point matches.

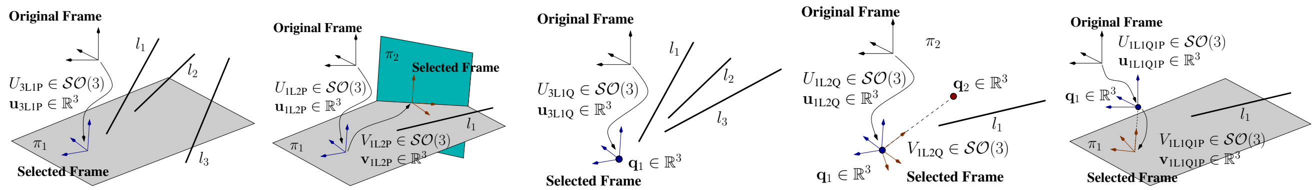
5.1. 3L1P: 3 line intersections and 1 plane match

Consider three pairs of intersecting lines $\{(l_1, m_1'), (l_2, m_2'), (l_3, m_3')\}$ and one pair of corresponding planes (π_1, π_1').

Selected Frames: We select appropriate coordinate systems to the 3D scans, verifying:

1. Planes π_1 and π_1' set as the $xy-$plane;

This is achieved by applying predefined transformations $U_{\text{3L1P}} \in \mathcal{SO}(3)$ and $\mathbf{u}_{\text{3L1P}} \in \mathbb{R}^3$ as defined in the supplementary materials, to the data in both scans. A graphical



(a) 3L1P (b) 1L2P (c) 3L1Q (d) 1L2Q (e) 1L1Q1P

Figure 2: *Representation of the selected frames to solve the mix line intersection and plane/point matches. (a) & (b) show the cases in which we have 3D plane correspondences; (c) & (d) depict the cases in which we have 3D point correspondences; and (e) depicts the case in which we have both 3D point and plane correspondences.*

representation of this selected coordinate system is shown in Fig. 2(a). With both frames verifying these specifications, the relative transformation between both scans is given by[1]

$$R = \frac{1}{1+s^2}\begin{bmatrix} 1-s^2 & -2s & 0 \\ 2s & 1-s^2 & 0 \\ 0 & 0 & 1+s^2 \end{bmatrix} \text{ and } \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix}, \tag{2}$$

decreasing the total degrees of freedom from six to three.

Solver: To compute the unknowns t_x, t_y, and s, we use the three line intersections. Replacing R and $\mathbf{t}$ in (1) by the ones in (2) and multiplying the result by $(1+s^2)$, we get three constraints of the form

$$\kappa_1^3[s, t_x, t_y] = \kappa_2^3[s, t_x, t_y] = \kappa_3^3[s, t_x, t_y] = 0, \tag{3}$$

where $\kappa_k^j[.]$[2] denotes the k^{th} polynomial with degree j. The monomials in these polynomial are linear in t_x and t_y; and quadratic in s. Taking the first and second algebraic constraints in (3), and solving them for t_x and t_y, we get

$$t_x = \frac{\kappa_4^4[s]}{\kappa_5^4[s]} \text{ and } t_y = \frac{\kappa_6^4[s]}{\kappa_7^4[s]}. \tag{4}$$

Substituting t_x and t_y in the third constraint from (3) by (4) and simplifying the equation, we get

$$\frac{\kappa_8^4[s]}{\kappa_9^2[s]} = 0 \implies \kappa_8^4[s] = 0. \tag{5}$$

To compute the transformation between both scans, we find the roots of the fourth degree polynomial equation $\kappa_8^4[s]$, which can be computed in closed-form by using the *Ferrari*'s formula. We get up to four solutions for s. Then, for each s, we get t_x and t_y by solving (4). The correct pose is computed by replacing t_x, t_y, and s in (2) and reversing the predefined transformations U_{3L1P} and $\mathbf{u}_{3L1P}$.

5.2. 1L2P: 1 line intersection and 2 plane matches

Consider one line intersection, (l_1, m_1'), and two 3D plane matches, $\{(\pi_1, \pi_1'), (\pi_2, \pi_2')\}$.

[1]We are using the Cayley's parameterization for $\mathcal{SO}(3)$ matrices.
[2]Due to space limitations, we omit the coefficients and monomials.

Selected Frames: As shown in Fig. 2(b), we transform the data in both 3D scans, such that:

1. Planes π_1 and π_1' are in the xy–plane;
2. The x–axis of both frames is along the intersection of the planes π_1 and π_2 in the first 3D scan. Similarly, the x–axis is along the intersection of the planes π_1' and π_2' in the second 3D scan.

For this purpose, we first apply the rotation $U_{1L2P} \in \mathcal{SO}(3)$ and translation $\mathbf{u}_{1L2P} \in \mathbb{R}^3$ to satisfy the first constraint, and then $V_{1L2P} \in \mathcal{SO}(3)$ and $\mathbf{v}_{1L2P} \in \mathbb{R}^3$ to satisfy the second constraint. U_{1L2P}, V_{1L2P} and $\mathbf{u}_{1L2P}, \mathbf{v}_{1L2P}$ are shown in the supplementary materials.

After applying these predefined transformations to the two 3D scans (i.e., considering π_1 and π_1'), the relative pose is determined up to a single translation parameter:

$$R = I \text{ and } \mathbf{t} = \begin{bmatrix} t_x & 0 & 0 \end{bmatrix}^T, \tag{6}$$

where I is the 3×3 identity matrix.

Solver: Since we only have one unknown t_x, we only need one intersecting line constraint, (1). By substituting R and $\mathbf{t}$ in (1) by (6) and solving for t_x, we get

$$t_x = \frac{\langle \bar{l}_1, \hat{m}_1' \rangle + \langle \hat{l}_1, \overline{m}_1' \rangle}{\bar{l}_{1,2}\overline{m}_{1,3}' - \bar{l}_{1,3}\overline{m}_{1,2}'}, \tag{7}$$

where the subscripts i in $l_{1,i}$ and $m_{1,i}$ denote the i^{th} element of the vector. Thus, we have a single solution to the relative transformation between both scans: we compute $\mathbf{t}$ as shown in (6), and revert to the original coordinate frames by using predefined transformations (U_{1L2P}, $\mathbf{u}_{1L2P}$, V_{1L2P}, and $\mathbf{v}_{1L2P}$).

5.3. 3L1Q: 3 line intersections and 1 point match

Consider three pairs of intersecting lines, (l_i, m_i') for $i = 1, 2, 3$, and one point match, $(\mathbf{q}_1, \mathbf{q}_1')$.

Selected frames: We apply predefined transformations to the both 3D scan frames, such that:

1. Points $\mathbf{q}_1$ and $\mathbf{q}_1'$ are the origin of the coordinate systems;

This is achieved by applying the transformation $U_{\text{3L1Q}} \in \mathcal{SO}(3)$ and $\mathbf{u}_{\text{3L1Q}} \in \mathbb{R}^3$ as defined in the supplementary materials. A graphical representation of the selected frame is presented in Fig. 2(c). Having translated both coordinate systems to $\mathbf{q}_1$ and $\mathbf{q}_1'$, respectively, the transformation between both frames is only given by a rotation $R \in \mathcal{SO}(3)$ (three rotational degrees of freedom) and $\mathbf{t} = \mathbf{0}$.

Solver: To obtain the rotation matrix, the three pairs of intersecting lines are used. Setting $\mathbf{t} = \mathbf{0}$ in (1), we obtain three linear independent equations of the form

$$\overline{m}_i'^{\,T} R\hat{l}_i + \hat{m}_i' R\bar{l}_i = 0, \text{ with } i = 1,2,3. \tag{8}$$

Vectorizing matrix R as $\text{vec}(R) \in \mathbb{R}^9$, we write

$$\underbrace{\left(\hat{l}_i^{\,T} \otimes \overline{m}_i'^{\,T} + \bar{l}_i^{\,T} \otimes \hat{m}_i'^{\,T}\right)}_{\mathbf{a}_i^T \in \mathbb{R}^9} \text{vec}(R) = 0 \tag{9}$$

with $i = 1,2,3$. Operator $\otimes$ denotes the *Kronecker* product.

Now, by stacking the three vectors $\mathbf{a}_i^T$ into a matrix $A \in \mathbb{R}^{3\times 9}$ and computing its Singular Value Decomposition, we get six vectors $\mathbf{b}_1, \dots, \mathbf{b}_6 \in \mathbb{R}^9$, which span the right null-space of A. Therefore, one can write

$$\begin{aligned} \text{vec}(R) &\sim \beta_1 \mathbf{b}_1 + ... + \beta_6 \mathbf{b}_6 \\ &\implies R \sim \beta_1 B_1 + \cdots + \beta_5 B_5 + \beta_6 B_6, \end{aligned} \tag{10}$$

where $\sim$ denotes an up to a scale equality, and $B_i \in \mathbb{R}^{3\times 3}$ are obtained by unstacking the vectors $\mathbf{b}_i$. From this equation, six degrees of freedom will remain, scalars β_i, for $i = 1, \dots, 6$. However, we note that R must belong to the space of rotation matrices, $R \in \mathcal{SO}(3)$, i.e. $R^T R = I$. To solve for β_i, we consider the following procedure. Without loss of generality, we set $\beta_6 = 1$ in (10). Then, we solve for $\beta_1, ..., \beta_5$ by inputting (10) in the ten orthogonality constraints of R:

$$\begin{aligned}
\|\mathbf{r}_1\|^2 - \|\mathbf{r}_2\|^2 = 0 \quad (11) &\qquad \|\mathbf{r}_1\|^2 - \|\mathbf{r}_3\|^2 = 0 \quad (12) \\
\|\mathbf{c}_1\|^2 - \|\mathbf{c}_2\|^2 = 0 \quad (13) &\qquad \|\mathbf{r}_1\|^2 - \|\mathbf{c}_3\|^2 = 0 \quad (14) \\
\mathbf{r}_1 \cdot \mathbf{r}_2 = 0 \quad (15) &\qquad \mathbf{r}_1 \cdot \mathbf{r}_3 = 0 \quad (16) \\
\mathbf{r}_2 \cdot \mathbf{r}_3 = 0 \quad (17) &\qquad \mathbf{c}_1 \cdot \mathbf{c}_2 = 0 \quad (18) \\
\mathbf{c}_1 \cdot \mathbf{c}_3 = 0 \quad (19) &\qquad \mathbf{c}_2 \cdot \mathbf{c}_3 = 0, \quad (20)
\end{aligned}$$

where $\mathbf{c}_j$, and $\mathbf{r}_j$ are the j-th column and row of matrix R respectively. To obtain the solutions for $\beta_1, ..., \beta_5$ we follow the steps in [71], that uses the automatic generator in [32]. We get up to eight possible solutions for R given by this solver. To compute the poses, we revert the predefined transformations, by applying the inverse of the transformations U_{3L1Q} and $\mathbf{u}_{\text{3L1Q}}$.

5.4. 1L2Q: 1 line intersection and 2 point matches

Consider one pair of intersecting lines $(l_1, m_1',)$ and 2 pairs of point correspondences $\{(\mathbf{q}_1, \mathbf{q}_1'), (\mathbf{q}_2, \mathbf{q}_2')\}$.

Selected Frames: We select appropriate reference frames before computing the 3D registration. For that purpose, we consider predefined transformations $U_{\text{1L2Q}}, V_{\text{1L2Q}} \in \mathcal{SO}(3)$ and $\mathbf{u}_{\text{1L2Q}} \in \mathbb{R}^3$, such that:

1. Points $\mathbf{q}_1$ and $\mathbf{q}_1'$ are the origin of the coordinate systems; and
2. The z–axis points towards points $\mathbf{q}_2$ and $\mathbf{q}_2'$, respectively.

$U_{\text{1L2Q}}, V_{\text{1L2Q}} \in \mathcal{SO}(3)$ and $\mathbf{u}_{\text{1L2Q}} \in \mathbb{R}^3$ are defined in the supplementary materials. A depiction of the result of these transformations is shown in Fig. 2(d). With these settings, we are left with a single rotational degree-of-freedom (i.e., $\mathbf{t} = \mathbf{0}$), which corresponds to the rotation about the z–axis. Then, R can be written as

$$R = \frac{1}{1+s^2}\begin{bmatrix} 1-s^2 & -2s & 0 \\ 2s & 1-s^2 & 0 \\ 0 & 0 & 1+s^2 \end{bmatrix}. \tag{21}$$

Solver: By setting $\mathbf{t} = \mathbf{0}$ and R as defined in (21) into the single constraint (1), and multiplying both sides by $(1+s^2)$, we obtain a second order polynomial in s:

$$\mu_2 s^2 + \mu_1 s + \mu_0 = 0, \tag{22}$$

where

$$\mu_0 = \langle \bar{l}_1, \hat{m}_1' \rangle + \langle \hat{l}_1, \overline{m}_1' \rangle \tag{23}$$

$$\mu_1 = 2\bar{l}_{1,1}\hat{m}_{1,2}' - 2\bar{l}_{1,2}\hat{m}_{1,1}' + 2\hat{l}_{1,1}\overline{m}_{1,2}' - 2\hat{l}_{1,2}\overline{m}_{1,1}' \tag{24}$$

$$\begin{aligned} \mu_2 = \bar{l}_{1,3}\hat{m}_{1,3}' - \hat{l}_{1,1}\overline{m}_{1,1}' - \bar{l}_{1,2}\hat{m}_{1,2}' - \\ \hat{l}_{1,2}\overline{m}_{1,2}' - \bar{l}_{1,1}\hat{m}_{1,1}' + \hat{l}_{1,3}\overline{m}_{1,3}'. \end{aligned} \tag{25}$$

This polynomial in (22) can be solved in closed-form with the quadratic formula, yielding two solutions for parameter s. Both values of s are then replaced in (21) to retrieve the relative rotation between the two frames. The final transformation is obtained by reverting the predefined transformations U_{1L2Q}, V_{1L2Q}, and $\mathbf{u}_{\text{1L2Q}}$.

5.5. 1L1Q1P: 1 line intersection and 1 point & plane matches

To end the minimal solvers, consider the scenario where one pair of intersection lines $(l_1, m_1',)$, one pair of plane correspondences (π_1, π_1'), and one pair of point matches $(\mathbf{q}_1, \mathbf{q}_1')$ are available.

Algorithm 1: *Hybric RANSAC loop for 3D scan alignment using points and plane correspondences and line intersections.*

Output: Transformation that aligns both Point Clouds
Data: Sets $\mathcal{L}$,Π, $\mathcal{Q}$, $\mathcal{S}$, n_g, m_g, o_g, $\delta_{\mathcal{L}}$, δ_{Π}, $\delta_{\mathcal{Q}}$, priors $P_p(g)$, and maximum number of iterations K

$\forall g \in \mathcal{G}$ Initialize $P(g) = 1$
while $\sum_g j_g < K$ **do**
  Select a solver s with probability $P(g)P_p(g)$
  Increment j_g
  Sample n_g line intersections from $\mathcal{L}$
  Sample m_g plane correspondences from Π
  Sample o_g point correspondences from $\mathcal{Q}$
  Compute T using solver g
  Compute number of inliers $I(T) = I_{\mathcal{L}}(T, \delta_{\mathcal{L}}) + I_{\Pi}(T, \delta_{\Pi}) + I_{\mathcal{Q}}(T, \delta_{\mathcal{Q}})$
  Compute $\epsilon_{\mathcal{L}}$, ϵ_{Π}, and $\epsilon_{\mathcal{Q}}$
  if $I(T) > I(T^*)$ **then**
    $T^* = T$
    for *each* $g \in \mathcal{G}$ **do**
      Update $P(g)$ with (26)
      Update $J(g)$ with (27)
  if $j_g > J(g)$ **then**
    return T^*

Selected Frames: Consider predefined transformations $U_{\text{1L1Q1P}} \in \mathcal{SO}(3)$ & $\mathbf{u}_{\text{1L1Q1P}} \in \mathbb{R}^3$ and $V_{\text{1L1Q1P}} \in \mathcal{SO}(3)$ & $\mathbf{v}_{\text{1L1Q1P}} \in \mathbb{R}^3$ such that:

1. The orthogonal projection of points $\mathbf{q}_1$ and $\mathbf{q}'_1$ to the planes π_1 and π'_1 (i.e., the projection through the normal direction of the planes) are the origin of the coordinate systems; and

2. The planes π_1 and π'_1 match the xy–plane.

$U_{\text{1L1Q1P}} \in \mathcal{SO}(3)$ & $\mathbf{u}_{\text{1L1Q1P}} \in \mathbb{R}^3$ and $V_{\text{1L1Q1P}} \in \mathcal{SO}(3)$ & $\mathbf{v}_{\text{1L1Q1P}} \in \mathbb{R}^3$ are defined in the supplementary materials, and the result is shown in Fig. 2(e). After applying the predefined transformations to both reference frames, the relative pose is given by a single rotation parameter, similar to (21) and $\mathbf{t} = \mathbf{0}$.

Solver: The single unknown parameter corresponds to the rotation around the z–axis, which aligns both frames. To compute this rotation the procedure of Sec. 5.4 was used. Once again the solver yields two solutions to the problem.

6. RANSAC for Pairs of Line Intersections

While conventional RANSAC loops use a single minimal solver (e.g., five 2D-2D points correspondences for monocular visual odometry in [49]), in this work, we have seven solvers using different types of features. This section presents a hybrid RANSAC scheme that is based on [6]. See Algorithm 1.

Consider all the solvers in Tab. 1, and let $\mathcal{L}$ be the set of all intersecting lines, Π be the set of all plane correspondences, and $\mathcal{Q}$ be the set of all point correspondences. Let $\epsilon_{\mathcal{L}}$, ϵ_{Π}, and $\epsilon_{\mathcal{Q}}$ be the inlier ratios, and $\delta_{\mathcal{L}}$, δ_{Π}, and $\delta_{\mathcal{Q}}$ be the inlier thresholds, where the subscripts denote the corresponding set. Finally, let $\mathcal{G}$ denote the set of minimal solvers g in Tab. 1, which require n_g samples from $\mathcal{L}$, m_g samples from Π, and o_g samples from $\mathcal{Q}$. For the distance metrics, we use geometric distance between points and intersecting lines, and the algebraic distance between plane π' and transformed π, with $\|\overline{\pi}\| = \|\overline{\pi}'\| = 1$.

At the beginning of each iteration, a solver is chosen based on the probability of each solver succeeding. In addition, one wants that the fewer times a solver was chosen the most likely it is to be selected. Given a solver g, and assuming a guess for the number of inliers[3], the probability of success for solver g is then given as

$$P(g) = \epsilon_{\mathcal{L}}^{n_g} \epsilon_{\Pi}^{m_g} \epsilon_{\mathcal{Q}}^{o_g} (1 - \epsilon_{\mathcal{L}}^{n_g} \epsilon_{\Pi}^{m_g} \epsilon_{\mathcal{Q}}^{o_g})^{j_g - 1}. \tag{26}$$

Besides taking the solver's probability of success, its performance in terms of runtime and numerical stability must be taken into account. A performance measure of each solver is introduced in its probability of being selected by defining a prior $P_p(g)$ for each one. These priors are empirical and selected based on the solvers' numerical stability, runtimes, and number of solutions. The final probability of selection is given by the product of $P(g)$ and $P_p(g)$.

The RANSAC loop stops when one of the solvers as been run for $J(g)$ iterations. $J(g)$ corresponds to the minimum number of iterations solver g should be run to guarantee a good solution with some probability P[4]:

$$J(g) = \frac{\log(1-P)}{\log(1 - \epsilon_{\mathcal{L}}^{n_g} \epsilon_{\Pi}^{m_g} \epsilon_{\mathcal{Q}}^{o_g})}. \tag{27}$$

Since the real inlier ratios are not known, $J(g)$ is updated each time a better model is found, i.e., having a higher inlier count. The inlier count is the sum of the inliers of each feature. Alternatively, a weighted sum based on the information each feature type provides can be considered [11].

7. Non-Linear Refinement

This section presents a non-linear refinement method used following the RANSAC loop described in Sec. 6. We start by defining a cost function which consists of the sum of the different distance metrics involved plus a regularization

[3] An estimate of these ratios is used, which consists in the inlier ratios for the best model given the previous estimates in a sequence.

[4] In this paper we set $P = 99\%$.

term for the rotation matrix:

$$\mathcal{F} = \sum_{k=1}^{\#\mathcal{L}} d_{\mathcal{L}}(l_k, m'_k) + \sum_{k=1}^{\#\mathcal{Q}} d_{\mathcal{Q}}(\mathbf{q}_k, \mathbf{q}'_k) + \sum_{k=1}^{\#\Pi} d_{\Pi}(\pi_k, \pi'_k) + \lambda ||RR^T - \mathbf{I}_3||^2, \quad (28)$$

where $d_{\mathcal{L}}(,)$, $d_{\mathcal{Q}}(,)$, and $d_{\Pi}(,)$ are the distance functions for the intersection of skew lines, point matches, and plane matches, respectively (see Sec. 6). Since, the cost function $\mathcal{F}$ is a sum of non-linear squares, we solve this problem using the Levenberg-Marquardt (LM) algorithm.

8. Experiments

The numerical stability of each solver in Tab. 1 is evaluated in Sec. 8.1. In Sec. 8.2, the solvers are injected in the RANSAC algorithm presented in Sec. 6, and are applied in real data-sets. Tests using simulated data are presented in the supplementary materials. We use the following methods for comparison: 1) ICP [60]; 2) Global Registration (GR) [10]; 3) FGR [78]; 4) Super4PCS[5] [41]; 5) RANSAC 6L; 6) RANSAC 3Q; 7) RANSAC Ours (Sec. 5); 8) RANSAC All (6L + 3Q + Ours); and 9) Ours + Refinement. The code for the solvers in Tab. 1, the RANSAC loop, and the non-linear refinement were implemented in C++. For the remaining methods, the available C++ implementations were used: ICP, GR, and FGR from the Open3D [79]; Super4PCS from the OpenGR [42].

For the error metrics, we consider the error in rotation and translation, $e_R(R)$ and $e_{\mathbf{t}}(\mathbf{t})$, as follows:

$$e_R(R) = \operatorname{acos}\left(\frac{\operatorname{trace}\left(R^{-1}R_{\mathrm{GT}}\right)-1}{2}\right) \quad (29)$$

$$e_{\mathbf{t}}(\mathbf{t}) = \|\mathbf{t} - \mathbf{t}_{\mathrm{GT}}\|, \quad (30)$$

where R_{GT} and $\mathbf{t}_{\mathrm{GT}}$ are the ground-truth rotation and translation respectively.

8.1. Numerical Validation of the Solvers

To evaluate the minimal solvers, each one was run 10^6 times with the corresponding minimum set, generated randomly. For each run, the data generation process consists of randomly creating a rigid transformation matrix, then sampling points from a 40 unit side cube. Points were used directly, lines are represented by two points, and planes are obtained from three points. Solvers' performance is measured by runtime, number of solutions, and rotation & translation errors. The first two metrics aim at evaluating how the solvers will perform in RANSAC. Since we aim at running solvers multiple times, the faster they yield a solution the faster RANSAC will run. Furthermore, the more solutions a solver outputs the slower RANSAC will be.

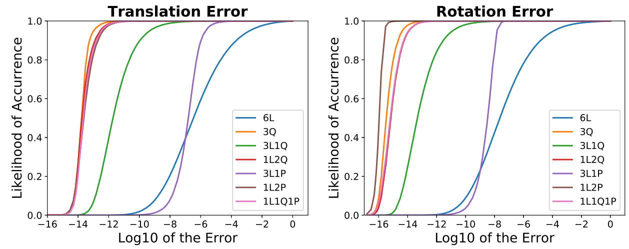


Figure 3: *Cumulative density plots of translation and rotation errors of the solvers presented in this paper, compared to the 6L and 3Q.*

	Runtime [μs]		#Solutions	
Solvers	**Mean**	**Median**	**Mean**	**Median**
6L	2757.763	2747.507	20.40	20
3Q	0.723	0.717	2	2
3L1Q	142.023	136.024	5.46	6
1L2Q	1.203	1.195	2	2
3L1P	5.073	5.138	3.10	4
1L2P	0.993	0.988	1	1
1L1Q1P	1.452	1.446	2	2

Table 2: *Mean and median runtime in microseconds, and number of solutions for each solver in Table 1 for* 10^6 *runs.*

The rotation and translation errors are presented in Fig. 3. All solvers converge to one, meaning that the solvers are estimating the solution correctly. However, we conclude that the solvers 6L and 3L1P have the worst performance, followed by the solver 3L1Q. The remaining solvers present similar performance. Tab. 2 presents the mean and median for the runtimes and number of solutions of each solver. As expected the solvers with higher runtimes are the ones that yield more solutions, particularly the solvers 6L and 3L1Q. The remaining solvers output four or fewer solutions and can be computed in closed-form.

8.2. Results

This subsection presents the results – proposed vs. baseline methods – in two real data-sets, SUN3D [74], and TUM [65]. Three sequences of each data-set were used (more sequences are shown in the supplementary materials).

To generate 3D lines, we extract line segment correspondences in pairs of RGB images, using the method in [37, 38]. Then, we iterate through each line and project every pixel to 3D, using the respective depth map. A 3D line is fitted to the points using least squares, and the distance of each line to the others is computed in both frames. If both distances are smaller than a threshold (we use 1cm), the lines are considered to intersect. Point and plane correspondences and line intersections are created from the previously fitted 3D lines as detailed in Sec. 3. This approach,

[5] Whenever Super4PCS fails to converge, ICP is run to keep the estimation of the path.

Rotation Error [Deg]

Method	SUN3D			TUM		
	MIT	BROWN	HOTEL	ROOM	DESK	XYZ
ICP [60]	0.95	3.35	4.11	8.22	1.17	1.74
GR [10]	1.20	1.44	1.92	2.55	1.36	2.01
Super4PCS [41]	2.89	4.55	7.78	7.36	1.86	2.70
FGR [78]	1.05	1.81	**1.26**	2.03	1.03	1.63
RANSAC 6L	0.86	1.32	1.45	1.84	1.01	1.78
RANSAC 3Q	1.07	1.44	1.86	2.24	1.42	2.05
RANSAC Ours	0.87	1.24	1.48	1.79	1.02	1.35
RANSAC All	**0.75**	**1.20**	1.43	1.80	**1.00**	**1.19**
Ours + Refinement	0.84	**1.20**	1.42	**1.76**	1.01	1.31

Translation Error [cm]

Method	SUN3D			TUM		
	MIT	BROWN	HOTEL	ROOM	DESK	XYZ
ICP [60]	4.14	15.45	10.02	13.79	3.19	4.96
GR [10]	4.79	6.73	6.15	5.51	3.78	3.77
Super4PCS [41]	5.06	12.07	7.23	10.09	6.49	7.88
FGR [78]	3.15	6.95	3.51	3.60	2.41	2.23
RANSAC 6L	3.17	6.17	**3.41**	3.83	2.36	2.50
RANSAC 3Q	3.57	6.94	4.21	4.59	3.16	3.14
RANSAC Ours	2.98	6.26	3.51	3.39	2.47	2.24
RANSAC All	**2.77**	**5.56**	3.57	3.69	2.34	**2.07**
Ours + Refinement	2.91	5.94	3.42	**3.32**	**2.21**	2.32

Table 3: *Median rotation and translation errors for real data-sets SUN3D [74] and TUM [65].*

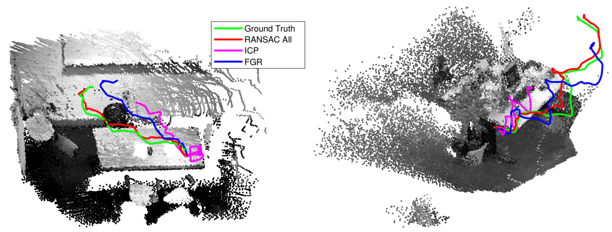


Figure 4: *Two examples of odometry results. On the left, a subset of the MIT sequence from Sun3D is presented, on the right we show a subset of the DESK sequence from TUM. In both figures, RANSAC All presents the smaller drift.*

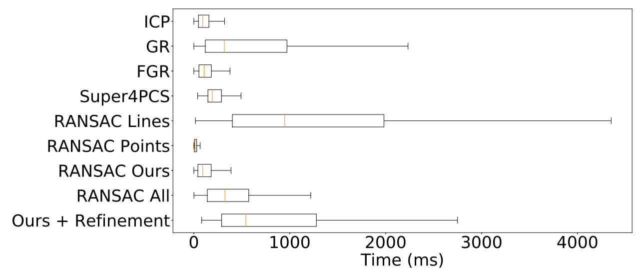


Figure 5: *Runtimes for the solvers in sequences in Tab 3.*

currently, takes 0.1 seconds for computing the line intersections, but it can be improved using neighborhood constraint on the potential line intersections.

All methods were run for pairs of point clouds, 10 frames apart. This distance was chosen, such that the displacement is significant enough for the identity matrix not to be a better solution than the baseline solvers. The RANSAC parameters were set to $K = 1000$, $\delta_{\mathcal{L}} = 0.5 \times 10^{-4}$, $\delta_{\Pi} = 0.01$, and $\delta_{\mathcal{Q}} = 0.001$. The LM algorithm was run for 20 iterations, and we set $\lambda = 1 \times 10^3$. The median rotation and translation errors are presented in Tab. 3. Two examples of odometry results using RANSAC ALL, FGR and ICP are shown in Fig. 4, where we can see that RANSAC All is the closest to the ground truth. Box plots of the runtimes of each method for both data-sets are presented in Fig. 5[6]. The runtimes[7] presented consider the total time of running the alignment methods, for example, the RANSAC runtime consists of the total time of executing Algorithm 1.

RANSAC 6L is the slowest method, which is expected given the complexity of the solver and its number of solutions (see Tab. 2). RANSAC 3Q method is the fastest overall. Nevertheless, it does not perform as well as the other RANSAC methods. In general, in terms of accuracy, the proposed solvers for the use of line intersection constraints are outperforming the baselines in the sequences from both data-sets. Exceptions are the rotation error in the SUN3D HOTEL sequence and, in some sequences the method RANSAC 3Q is being outperformed by FGR and ICP. Finally, the improvements made by the non-linear refinement (check Tab 3) are small, indicating that the solutions obtained by RANSAC Ours are close to the optimum of the cost function in (28).

[6] Pre-prossing runtimes are excluded.

[7] All methods were run in a Intel Core$^{\text{TM}}$ i7-5820K

9. Discussion

The main contribution is the use of line intersection constraints in 3D registration. This is addressed by means of robust techniques, namely RANSAC. We proposed five novel minimal solvers and developed a hybrid RANSAC, that can use all the solvers. Furthermore, a non-linear refinement is proposed to be applied after RANSAC. The results in real data-sets show that the use of line intersections improves the 3D registration accuracy. The improvements made by the non-linear refinement are small, showing that the solutions obtain by RANSAC are near-optimal. Future work consists of deriving new minimal solver, which take into account not only line intersections but also line correspondences, and the incorporation of motion averaging in multiple 3D scan alignments.

Acknowledgements

This work was supported by the Portuguese National Funding Agency for Science, Research and Technology (FCT) grant `PD/BD/135015/2017`, project `PTDC/EEI-SII/4698/2014`, and the LARSyS - FCT Plurianual funding 2020-2023.

References

[1] Dror Aiger, Niloy J. Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust surface registration. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2008)*, 27(3):1–10, 2008.

[2] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 7163–7172, 2019.

[3] Hrvoje Benko, Ricardo Jota, and Andrew Wilson. Miragetable: Freehand interaction on a projected augmented reality tabletop. In *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 199–208, 2012.

[4] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 14(2):239–256, 1992.

[5] Uttaran Bhattacharya and Venu Madhav Govindu. Efficient and robust registration on the 3d special euclidean group. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 5885–5894, 2019.

[6] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid camera pose estimation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 136–144, 2018.

[7] Jiawen Chen, Shahram Izadi, and Andrew Fitzgibbon. KinÊtre: Animating the world with the human body. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 435–444, 2012.

[8] Yang Chen and Gerard Medioni. Object modeling by registration of multiple range images. In *IEEE Int'l Conf. Robotics and Automation (ICRA)*, volume 3, pages 2724–2729, 1991.

[9] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The trimmed iterative closest point algorithm. In *Object recognition supported by user interaction for service robots*, volume 3, pages 545–548, 2002.

[10] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5556–5565, 2015.

[11] Brian Clipp, Christopher Zach, Jan-Michael Frahm, and Marc Pollefeys. A new minimal solution to the relative pose of a calibrated stereo camera with small field of view overlap. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 1725–1732, 2009.

[12] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *European Conf. Computer Vision (ECCV)*, pages 602–618, 2018.

[13] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 3d local features for direct pairwise registration. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 3244–3253, 2019.

[14] Li Ding and Chen Feng. Deepmapping: Unsupervised map estimation from multiple point clouds. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 8650–8659, 2019.

[15] Mingsong Dou, Henry Fuchs, and Jan-Michael Frahm. Scanning and tracking dynamic objects with commodity depth cameras. In *IEEE Int'l Symposium on Mixed and Augmented Reality (ISMAR)*, pages 99–106, 2013.

[16] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3d point cloud registration for localization using a deep neural network auto-encoder. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4631–4640, 2017.

[17] Felix Endres, Jurgen Hess, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. 3-d mapping with an rgb-d camera. *IEEE Trans. Robotics (T-RO)*, 30(1):177–187, 2014.

[18] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[19] Andrew W. Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing (IVC)*, 21(13):1145–1153, 2003.

[20] Wolfgang Forstner and Kourosh Khoshelham. Efficient and accurate registration of point clouds with plane to plane correspondences. In *IEEE Int'l Conf. Computer Vision Workshops (ICCVW)*, pages 2165–2173, 2017.

[21] Venu Madhav Govindu. Combining two-view constraints for motion estimation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2001.

[22] Michae D. Grossberg and Shree K. Nayar. A general imaging model and a method for finding its parameters. In *IEEE Int'l Conf. Computer Vision (ICCV)*, volume 2, pages 108–115, 2001.

[23] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *Int'l J. Computer Vision (IJCV)*, 103(3):267–305, 2013.

[24] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research (IJRR)*, 31(5):647–663, 2012.

[25] Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Int'l Symposium Robotics Research (ISRR)*, pages 235–252, 2017.

[26] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 559–568, 2011.

[27] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *European Conf. Computer Vision (ECCV)*, pages 607–623, 2018.

[28] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 2100–2106, 2013.

[29] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Robust odometry estimation for rgb-d cameras. In *IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 3748–3754, 2013.

[30] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2969–2976, 2011.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[32] Zuzana Kukelova, Martin Bujnak, , and Tomas Pajdla. Automatic generator of minimal problem solvers. In *European Conf. Computer Vision (ECCV)*, pages 302–315, 2008.

[33] Viktor Larsson, Zuzana Kukelova, and Yinqiang Zheng. Making minimal solvers for absolute pose estimation compact and robust. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2335–2343, 2017.

[34] Gim Hee Lee, Bo Li, Marc Pollefeys, and Friedrich Fraundorfer. Minimal solutions for the multi-camera pose estimation problem. *The International Journal of Robotics Research*, 34(7):837–848, 2015.

[35] Hongdong Li and Richard Hartley. Five-point motion estimation made easy. In *IEEE Int'l Conf. Pattern Recognition (ICPR)*, volume 1, pages 630–633, 2006.

[36] Hongdong Li and Richard Hartley. The 3d-3d registration problem revisited. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 1–8, 2007.

[37] Kai Li, Jian Yao, and Xiaohu Lu. Robust line matching based on ray-point-ray structure descriptor. In *Asian Conf. Computer Vision (ACCV)*, pages 554–569, 2014.

[38] Kai Li, Jian Yao, Xiaohu Lu, Li Li, and Zhichao Zhang. Hierarchical line matching based on line–junction–line structure descriptor and local homography estimation. *Neurocomputing*, 184:207–220, 2016.

[39] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. Deepvcp: An end-to-end deep neural network for point cloud registration. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 12–21, 2019.

[40] Yan Lu and Dezhen Song. Robust rgb-d odometry using point and line features. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 3934–3942, 2015.

[41] Nicolas Mellado, Dror Aiger, and Niloy J Mitra. Super 4pcs fast global pointcloud registration via smart indexing. *Computer Graphics Forum*, 33(5):205–215, 2014.

[42] Nicolas Mellado et al. Opengr: A c++ library for 3d global registration. https://storm-irit.github.io/OpenGR/, 2017.

[43] Javier Minguez, Luis Montesano, and Florent Lamiraux. Metric-based iterative closest point scan matching for sensor displacement estimation. *IEEE Trans. Robotics (T-RO)*, 22(5):1047–1054, 2006.

[44] Pedro Miraldo and Helder Araujo. Calibration of smooth camera models. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 35(9):2091–2103, 2013.

[45] Pedro Miraldo, Tiago Dias, and Srikumar Ramalingam. A minimal closed-form solution for multi-perspective pose estimation using points and lines. In *European Conf. Computer Vision (ECCV)*, pages 490–507, 2018.

[46] Pedro Miraldo, Surojit Saha, and Srikumar Ramalingam. Minimal solvers for mini-loop closures in 3d multi-scan alignment. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 9699–9708, 2019.

[47] Andriy Myronenko, Xubo Song, and Miguel A. Carreira-Perpian. Non-rigid point set registration: Coherent point drift. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1009–1016, 2007.

[48] David Nister. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 26(6):756–770, 2004.

[49] David Nister, Oleg Naroditsky, and James Bergen. Visual odometry. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2004.

[50] Carl Olsson, Fredrik Kahl, and Magnus Oskarsson. Branch-and-bound methods for euclidean registration problems. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 31(5):783–794, 2009.

[51] 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 *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020.

[52] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 143–152, 2017.

[53] Robert Pless. Using many cameras as one. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 587, 2003.

[54] Helmut Pottmann and Johannes Wallner. *Computational Line Geometry*. Springer-Verlag Berlin Heidelberg, 1 edition, 2001.

[55] 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. Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[56] Srikumar Ramalingam and Peter Sturm. A unifying model for camera calibration. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 39(7):1309–1319, 2017.

[57] Srikumar Ramalingam and Yuichi Taguchi. A theory of minimal 3d point to 3d plane registration and its generalization. *Int'l J. Computer Vision (IJCV)*, 102(1):73–90, 2013.

[58] Carolina Raposo and Joao Barreto. 3d registration of curves and surfaces using local differential information. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 9300–9308, 2018.

[59] Carolina Raposo, Miguel Loureno, Joao Barreto, and Miguel Loureno. Plane-based odometry using an rgb-d camera. In *British Machine Vision Conference (BMVC)*, 2013.

[60] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling (3DIM)*, volume 1, pages 145–152, 2001.

[61] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 3212–3217, 2009.

[62] Peter H. Schonemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[63] Frank Steinbrucker, Christian Kerl, Daniel Cremers, and Jurgen Sturm. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 3264–3271, 2013.

[64] Henrik Stewenius, Magnus Oskarsson, Kalle Astrom, and David Nister. Solutions to minimal generalized relative pose problems. In *Workshop on Omnidirectional Vision (OMNIVIS)*, 2005.

[65] Jurgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 573–580, 2012.

[66] Peter Sturm. Multi-view geometry for general camera models. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 206–212, 2005.

[67] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane slam for hand-held 3d sensors. In *IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 5182–5189, 2013.

[68] Gary K.L. Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C. Langbein, Yonghuai Liu, David Marshall, Ralph R. Martin, Xian-Fang Sun, and Paul L. Rosin. Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. *IEEE Trans. Visualization and Computer Graphics (T-VCG)*, 19(7):1199–1217, 2013.

[69] Raul Mur-Artal ; Juan D. Tardos. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. Robotics (T-RO)*, 33(5):1255–1262, 2017.

[70] Yanghai Tsin and Takeo Kanade. A correlation-based approach to robust point set registration. In *European Conf. Computer Vision (ECCV)*, pages 558–569, 2004.

[71] Jonathan Ventura, Clemens Arth, Gerhard Reitmayr, and Dieter Schmalstieg. A minimal solution to the generalized pose-and-scale problem. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 422–429, 2014.

[72] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 3522–3531, 2019.

[73] Thomas Whelan, Michael Kaess, John J. Leonard, and John McDonald. Deformation-based loop closure for large scale dense rgb-d slam. In *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 548–555, 2013.

[74] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 1625–1632, 2013.

[75] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-icp: Solving 3d registration efficiently and globally optimally. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 1457–1464, 2013.

[76] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int'l J. Computer Vision (IJCV)*, 13(2):119–152, 1994.

[77] Lei Zhou, Siyu Zhu, Zixin Luo, Tianwei Shen, Runze Zhang, Mingmin Zhen, Tian Fang, and Long Quan. Learning and matching multi-view descriptors for registration of point clouds. In *European Conf. Computer Vision (ECCV)*, pages 505–522, 2018.

[78] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conf. Computer Vision (ECCV)*, pages 766–782, 2016.

[79] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv:1801.09847*, 2018.

[80] Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing (IVC)*, 21(11):977–1000, 2003.