MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Fast and Accurate 3D Registration from Line Intersection Constraints

Mateus, Andre; Ranade, Siddhant; Ramalingam, Srikumar; Miraldo, Pedro

TR2023-007 March 01, 2023

Abstract

3D Registration is a fundamental part of several robotics and automation tasks. While classical methods predominantly exploit constraints from points or plane correspondences, we have a dif- ferent take using line intersections. In other words, we focus on exploiting geometric constraints arising from the intersection of two (different) 3D line segments in two scans. In particular, we derive nine minimal solvers from various geometric constraints arising from line intersections along with other constraints: plane correspondences, point correspondences, and line matches. We fol- low a two-step method for 3D registration: a coarse estimation with outlier rejection followed by refinement. In the first step, we use a hybrid RANSAC loop that utilizes all the minimal solvers. This RANSAC outputs a rough estimate for the 3D registration and the outlier/inlier classification for the 3D features. As for the refinement, we offer a non-linear technique using all the inliers obtained from the RANSAC and the coarse estimate. This method is of alternate minimization type, in which we alternate between estimating the rotation and the translation at each step. Thorough experiments with simulated data and two real-world datasets show that using these features and the combined solvers improves accuracy and is faster than the baselines.

International Journal of Computer Vision 2023

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Fast and Accurate 3D Registration from Line Intersection Constraints

André Mateus[1], Siddhant Ranade[2], Srikumar Ramalingam[3] and Pedro Miraldo[4*]

[1]Ericsson Research, Stockholm, Sweden.
[2]School of Computing, University of Utah, US.
[3]Google Research, New York, US.
[4]Mitsubishi Electric Research Labs (MERL), Cambridge, US.

*Corresponding author(s). E-mail(s): miraldo@merl.com;
Contributing authors: andre.mateus@ericsson.com; sidra@cs.utah.edu;
rsrikumar@google.com;

Abstract

3D Registration is a fundamental part of several robotics and automation tasks. While classical methods predominantly exploit constraints from points or plane correspondences, we have a different take using line intersections. In other words, we focus on exploiting geometric constraints arising from the intersection of two (different) 3D line segments in two scans. In particular, we derive nine minimal solvers from various geometric constraints arising from line intersections along with other constraints: plane correspondences, point correspondences, and line matches. We follow a two-step method for 3D registration: a coarse estimation with outlier rejection followed by refinement. In the first step, we use a hybrid RANSAC loop that utilizes all the minimal solvers. This RANSAC outputs a rough estimate for the 3D registration and the outlier/inlier classification for the 3D features. As for the refinement, we offer a non-linear technique using all the inliers obtained from the RANSAC and the coarse estimate. This method is of alternate minimization type, in which we alternate between estimating the rotation and the translation at each step. Thorough experiments with simulated data and two real-world datasets show that using these features and the combined solvers improves accuracy and is faster than the baselines.

Keywords: 3D registration, line intersections, alternative minimization

1 Introduction

3D devices such as RGB-D and 3D LiDAR are the primary sensors for applications such as augmented reality [1, 2, 3, 4], navigation [5, 6, 7, 8], and SLAM [9, 10, 11, 12, 13], especially because they provide 3D depth information about the environment. One can obtain this kind of data from different sources. For example from rigidly mounted multiple perspective sensors, a set of RGB cameras; from structured light [14]; from Time-of-Flight (ToF) cameras, such as [15]; or from rotating Light Detection And Ranging (LiDAR) sensors, [16], which is a specific type of ToF based sensor. These devices have become more accurate and less expensive over the last few years, mainly due to autonomous navigation needs and robotics.

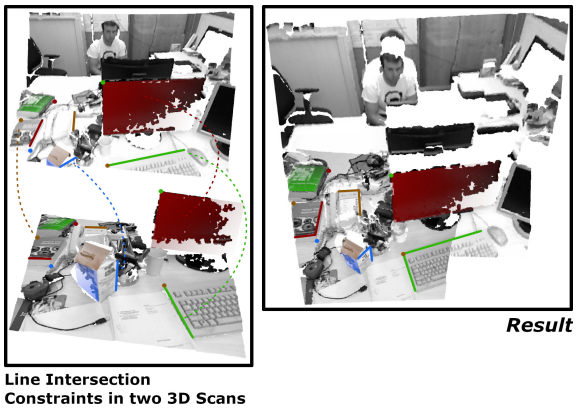

Figure 1: **Main idea:** We identify pairs of 3D lines that intersect from two scans, i.e., a line from the first scan (image on the top left) intersects with the line from the second (image on the bottom left) in the common reference frame (image on the right). In this paper, we jointly exploit these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

3D registration is the problem of finding the pose that aligns two 3D point cloud scans. If we consider a mobile device moving in space, acquiring 3D scans, this alignment is equivalent to the visual egomotion problem using a conventional monocular imaging device. A key method for solving the 3D registration problem is the well-known iterative closest point (ICP) [17, 18], an iterative technique of the alternative minimization type that alternates between estimating the 3D point correspondence set and finding the transformation that aligns the two scans. Although the main ideas remain, ICP was improved significantly in the last decades with most advances attempting to 1) improve robustness to outliers, and 2) reduce the chances of getting stuck in local minima, and reduce the problem's computational complexity. One way of dealing with these issues is to use RANSAC [19] and minimal solvers [20]. We find solutions to minimal random subsets of constraints instead of solving an optimization problem that directly computes the transformation. The output solution is obtained by choosing the solution that gathers the most consensus, i.e., the solution with the largest number of inliers. This paper focuses on improving points 1) and 2). Specifically, we tackle the robustness problem by removing outliers. Then, to reduce the chances of falling in local minima, we use a rough estimate of the transformation obtained from RANSAC as an initial estimate for a non-linear refinement.

Most existing methods for solving 3D registration exploit the use of points in the point cloud, either using all points from the 3D scan (e.g., ICP [18]) or explicitly using correspondences obtained from 3D descriptors (such as FPFH [22]). This work uses the latter approach with a different descriptor. First, we focus on environments containing 3D planes and straight lines, as shown in Fig. 1. In contrast to points correspondences, lines are one-dimensional features that are simpler and more accurately computed because one can get them from a set of points belonging to the line (both in 2D and 3D). Moreover, these lines are usually obtained from the intersection of plane features, making their description easier. More accurate 3D features should yield a more precise and robust registration. Contrary to 2D images, another significant advantage of using lines instead of points in 3D scans is related to the sparse nature of the data. Both RGB-D and 3D LiDAR data are sparse, making the point matches harder or even impossible to get. For example, if we consider two 3D LiDAR scans, for a 3D point obtained in the first one, we will not have a 3D point in the point cloud of the second scan.

1.1 Contributions

In this paper we study the following questions.

RQ1: *How to formulate and exploit the rich constraints from line intersections to improve 3D registration?*

RQ2: *Can we develop a robust and efficient algorithm to exploit the line intersection constraints?*

Our contributions are:

1. Nine minimal solvers for the cases of mixing line intersections with plane, point, and line matches (see Tab. 1);
2. A hybrid RANSAC scheme to account for all possible combinations of minimal sets;
3. A new and fast non-linear refinement solver that uses all the inliers from the RANSAC

Solver	# Intersecting Lines (L)	Matches			Total #Features	Pre-Transformation		#Sol
		#Points (Q)	#Planes (P)	#Lines (M)		Matches	#DoF (T, R)	
6L[21]	6	0	0	0	6	–	(3, 3)	64
3Q[20]	0	3	0	0	3	3Q	(0, 0)	1
3L1P	3	0	1	0	4	1P	(2, 1)	4
1L2P	1	0	2	0	3	2P	(1, 0)	1
3L1Q	3	1	0	0	4	1Q	(0, 3)	8
1L2Q	1	2	0	0	3	2Q	(0, 1)	2
1L1Q1P	1	1	1	0	3	1Q1P	(0, 1)	2
1M1Q	0	1	0	1	2	1M1Q	(0, 0)	1
2M	0	0	0	2	2	2M	(0, 0)	1
2L1M	2	0	0	1	3	1M	(1, 1)	4
1M1P	0	0	1	1	2	1M1P	(0, 0)	1

Table 1: **Minimal solvers:** Available and proposed solvers used/derived to compute the 3D alignment using line intersection constraints, including the resulting maximum number of solutions each method gives, i.e., problem complexity. Furthermore, the number of DoF (both rotation and translation) remaining to estimate after applying the pre-transformations are also presented. In general, this number of solutions corresponds to the degree of the polynomial equation obtained in this paper. See Sec. 5 for more detail.

loop and refines the 3D registration using an alternating minimization method that alternates between estimating the rotation and translation;

4. Our methods were tested with simulated data and two different available real world datasets, SUN3D and TUM RGB-D [23, 10] for validation and comparison with the baselines; and
5. We conclude that only using RANSAC and the line intersection constraints help improve the overall results, being faster than the baselines. When utilizing our non-linear refinement, the accuracy improves significantly with a negligible increment in computational time (concerning our RANSAC), still being substantially faster than current baselines.

1.2 Outline

We organize the paper as follows. The next section presents the related work. Section 3 shows the notations, and our approach is described in Sec. 4. We also include the basic geometric constraints used in the rest of the paper. The minimal solvers using line intersection constraints are presented in Sec. 5. The hybrid RANSAC loop derived for using all the minimal solvers is presented in Sec. 6. Section 7 derives our alternative minimization refinement method. The experimental results are shown in Sec. 8, and we conclude the paper in Sec. 9.

2 Related Work

This section is divided into five subsections. We start by discussing previous techniques in 3D registration, focused on classical/non-data-driven methods. Section 2.2 describes some previous approach on robust estimation using minimal solvers, and Sec. 2.3 focus on data-driven methods for 3D registration. Since, in this work, we derive an alternating minimization method for registration refinement, Sec. 2.4 offers some previous approaches in computer vision using these kinds of strategies. We end this section by describing the authors' prior methods to this work.

2.1 Traditional computer vision methods

The gold standard method for obtaining the 3D registration from two scans without correspondences is the Iterative Closest Point (ICP), presented in [18]. Most of the proposed alternatives over the years aimed at improving some of the ICP issues, namely improving the results in the presence of outliers and ensuring the convergence to a global minimum. Some examples are [24, 25, 26, 27, 28, 29, 30, 31, 32].

A more recent approach in [33] derives a branch-and-bound method with guarantees of global optimality. KinectFusion in [1] is a well-known method focused on getting accurate and real-time registration in complex and arbitrary indoor scenes and varying lighting conditions. In [34], the authors combine geometric registration of scene fragments with robust global optimization, based on online processes for the robust reconstruction of indoor scenes. Another well-known method is the Super4PCS method [35], which aims at getting a robust solution to the registration. The method uses random samples of four points with the assumption that they verify the co-planar constraints. Super4PCS is an extension of the 4PCS [36], aiming at fixing the large computational complexity involved in the prior method. While Super4PCS is of linear time complexity, 4PCS is quadratic for the number of data points. The current state-of-the-art non-data-driven method is the Fast Global Registration method (FGR), presented in [37]. The authors propose a robust optimization technique that uses a robust distance function for outlier removal. The goal is to avoid running RANSAC before the refinement procedure. The method gets the registration by partially overlapping 3D surfaces. A more recent method in [38] optimizes joint photometric and geometric properties to compute the 3D registration.

The previous paragraph shows some related work on using points for computing the 3D registration. (Although 4PCS and Super4PCS use planar constraints implicitly, explicitly they consider points.) However, a few previous works use/combine different 3D features. In [11, 39] the authors present a method that combines points and planes, and [6, 40] proposes methods for point cloud registration with a plane to plane match. [41] solves the problem with curves and surfaces.

Some other authors study other research questions in large-scale 3D registration problems. One of the critical problems for reducing drift with estimating localization over time is to use methods like rotation/transformation averaging (see, for example, [42, 43, 44]) or loop closure (e.g., [45, 46, 47, 48, 34, 20]), which require more than two 3D scans. However, these techniques are again iterative (latest ones rely on graph optimization [49]), and therefore a good initialization is mandatory to deliver an accurate solution. In this paper, we consider only the pairwise 3D registration problem.

2.2 Using minimal solvers

The standard technique for obtaining a robust estimate is RANSAC, in [19]. This technique runs a loop for a number of iterations. In theory, this number is selected to ensure that at least one sample set is error/outlier free with high probability. However, in practice, one needs to know the probability of a data point being an inlier to obtain this value, which is not known a priori in most applications. The algorithm proceeds as follows: each iteration computes a hypothesis based on a randomly generated sample of minimal data; the hypothesis is scored based on an inlier metric and; if it outscores the previous best hypothesis, the current estimation is updated. RANSAC's output is the best hypothesis calculated using all the data. Besides some improvements to the original RANSAC, which is transverse to all applications (many works have been published to improve RANSAC, see for example [50, 51, 52, 36, 53, 54, 55]), minimal solvers are the tools that describe each problem.

Over the last couple of decades, authors proposed many solvers for different problems. Among many minimal solvers, we highlight some relevant ones for odometry/relative camera pose, e.g., [56, 57, 21, 58, 59], and localization/camera pose, such as [60, 61, 62, 63]. Minimal solvers and RANSAC provide efficient and robust solutions to 3D registration. For example, 4PCS and Super4PCS in [36, 35], mentioned in Sec. 2.1, use sets of four points within a RANSAC framework for robust estimation. A more basic and standard pipeline for 3D registration is to consider sets of three points, for example, with the solver in [64, 20], which is the minimum set of point correspondences required to get the transformation. In [20], the authors derive minimal solvers for mini loop closures in 3D scan alignment. That paper shows that combining correspondences from a cycle of 3D scans improves the overall accuracy of the 3D registration.

2.3 Deep learning alternatives

We have witnessed a significant increase in published methods for 3D registration in the last few years. Unfortunately, it is impossible to list

all the works; we will list some key methods. However, most of these concentrate on data processing (feature detection and matching) rather than attacking the actual alignment problem, consisting of finding the transformation that aligns the data. The main reason is related to the well-defined algebraic constraints involved in finding the transformation parameters. There is no apparent advantage in using DNN strategies for solving a well-known and straightforward algebraic problem. Most authors rely on traditional methods for finding this transformation, as is reflected in the related work below.

In this work, we focus on estimating the transformation parameters rather than the feature extraction or matching.

3D registration:

In PointNetLK, [65], the authors derive an iterative optimization solution for 3D registration problem. First, they change the PointNet encoder to match the classical Lucas-Kanade algorithm for 2D images. Then, the source and template point clouds are injected into the two PointNets for feature extraction at each iteration. Using the discrepancy in the feature space obtained from the two point clouds and a modified inverse compositional Lucas-Kanade algorithm, the method computes the alignment (exponential map of a six-dimensional vector) for the current iteration. This transformation is then applied to the source point cloud in the next iteration, and the same process is repeated until a minimum threshold for the estimated transformation is found. This method was extended in [66] by using analytical Jacobians to circumvent the numerical instabilities of the original PointNetLK.

A different alternative is proposed in [67]. The authors offer a method that estimates the pose directly (against iterative approaches of [18, 65]). The authors derive a network for obtaining matches between a template and a source point cloud. The proposed architecture passes the input point clouds through a graph convolutional network, followed by a transformer, and Softmax in a row, treating inputs as unordered sets. After getting the matches, the authors run a traditional 3D registration method to get the pose.

The work in [68] aims at solving the classical 3D registration problem from [69]; i.e., calculate the alignment from a previously computed set of point correspondences. The goal is to get a fast, agnostic estimator to the feature extractor. The authors present a DNN to classify 3D input correspondences as inliers/outliers while computing the 3D registration simultaneously. A similar problem was addressed in [70]. However, instead of having six dimensional vectors as the stacking coordinates of correspondence pairs in a PointNet style, like [68], the authors use a high-dimensional convolutional network to analyze the structure and classify the given correspondences. The authors also derive a differentiable weighted Procrustes solver with linear computational complexity with respect to the number of correspondences and a robust optimization module that fine-tunes the computed alignment.

Feature extraction and correspondences:

Following the reasoning indicated above about solving the alignment with traditional techniques vs. DNN ones, most authors aim at explicitly extracting features and descriptors from point-clouds.

In [71], the authors start with creating super-points from the row data. Then, they create 2D depth maps by projecting the normalized points inside each super-point. Irrelevant super-points are filtered out, and the remaining ones go through a dimension reduction using an auto-encoder. Candidates for matching and coarse and fine estimations are computed using traditional 3D registration methods.

PPFnet, in [72], proposes to use 4D geometrical descriptors (PPF) representing the surface from a pair of oriented 3D points. The authors encode the local geometry from patches of points, in which the features are obtained by stacking PPF descriptors from the representative point in the patch and its neighbors. These geometric features pass independently through PointNet to compute local features, followed by a max-pooling to obtain the global ones. Finally, the global and local features are concatenated and pass through a multilayer perceptron that computes the final features. In [73] the authors extend PPFnet to work without supervision and pose invariants.

In [74], the authors start with sampling clusters from row data. Then propose a three-branch Siamese network structure, using anchor, positive,

and negative point clouds as inputs. The network consists of a detector network followed by a descriptor network. Finally, the output of positive & anchors and negative & anchors features are injected in a feature alignment proposed in [75] and trained with the triplet loss.

The Fully Convolutional Geometric Features (FCGF) method is presented in [76]. Instead of using geometric patches centered in points of interest like [72, 73], which is computationally expensive and implicitly decreases resolution, the authors use 3D convolutions on the entire point clouds by transforming convolutions to fully-convolutional counterparts. Another fully convolutional method is derived in [77]. The authors use Kernel Point Convolution to get local geometry information, using kernel points carrying convolution weights as the backbone network for dense feature description extraction.

In [78], the authors changed the Robust Point Matching of [79] by replacing the spatial distances with learned hybrid feature distances, and letting the network decide the softassign hyperparameter values. The "PREDATOR" method in [80] starts with voxelized point clouds that are injected in an encoder-decoder scheme with shared weights (one per 3D scan). An overlap-attention module with a series of self- and cross-attention blocks updates the features in the latent space before the decoder estimates the conditioned features and overlap scores to per-point feature descriptors, overlap scores, and matchability scores.

2.4 Alternative minimization methods in computer vision

Computer vision researchers have used Alternate Minimization (AM) techniques for decades in various problems. Concerning 3D registration, ICP in [18] is one of these. As mentioned in Sec. 2.1, it is the gold standard method for dense point-to-point registration, that alternates between finding closest correspondences and computing the rigid transformation. Another well-known method that uses an alternative minimization method to solve the 3D registration is the one in [37]. In this case, the authors assume the point correspondences are known and alternate between finding the line process over the correspondences (robust estimation) and finding the transformation that aligns both scans.

Furthermore, AM and projection algorithms [81, 82, 83] have been used for human pose and other geometrical problems [84, 85, 86, 87]. In this work, we model an AM method for 3D registration refinement, considering line intersection constraints. The method is iterative and alternates between estimating the rotation and translation parameters.

2.5 Prior work

This work follows some of the authors' groups prior works, namely [88, 89, 90]. In both [90, 89], we introduce the use of line intersections for 3D registration. In [89], we describe the 3D registration problem as an AM one, which alternates between finding local alignments (transformations that align a single feature) and finding the consensus transformation given all the transformed features. The work provides superior results concerning the 3D alignment but with high computational costs. The method is extremely slow and not applicable to real-time problems. In [90], we took a different approach. We define constraints from line intersections and then derive five new minimal solvers for the 3D registration problem. Then, we present a hybrid RANSAC loop for considering all the solvers and offer a simple non-linear refinement strategy. However, this refinement is computationally heavy while showing minor or no improvements in accuracy. This is due to the highly non-linear nature of the constraints. [90] is the closest to work to this paper. The main differences are four new minimal solvers considering line matches and a refinement technique that provides a more accurate 3D alignment. This new refinement technique is of the AM framework presented in [88]. This method relaxes the non-linearities involved in the cost function, by splitting the problem in two: rotation only, translation only optimization problems, fixing the translation and rotation respectively, and solving the problem by alternating between the two estimators. This makes the solver converge significantly faster and produce more accurate results.

3 Notations

Capital letter represent matrices (e.g., $A \in \mathbb{R}^{N\times K}$), small bold letter represent column vectors (e.g., $\mathbf{a} \in \mathbb{N}$), and small regular letters represent

scalars. We use *Plücker* coordinates to represent 3D lines, i.e., $\mathbf{l} \in \mathbb{R}^6 \overset{\text{def}}{\sim} [\bar{\mathbf{l}}\ \hat{\mathbf{l}}]$, where $\bar{\mathbf{l}}$ and $\hat{\mathbf{l}}$ are the line's direction and moment, respectively. (See [91] for more details.) Planes are represented by a four dimensional vector $\boldsymbol{\pi} \in \mathbb{R}^4 \overset{\text{def}}{=} [\overline{\boldsymbol{\pi}}\ \widetilde{\pi}]$, in which $\overline{\boldsymbol{\pi}}$ and $\widetilde{\pi}$ represent the plane's normal vector and distance to the origin, respectively. 3D points are represented by $\mathbf{q} \in \mathbb{R}^3$. Subscripts are used to denote the index of the feature (e.g., $\mathbf{l}_i$ denotes the i^{th} 3D line), and the apostrophes are used to identify the matches of features in pairs of scans (e.g., the tuples $(\mathbf{l}_1, \mathbf{l}'_1)$ and $(\mathbf{q}_1, \mathbf{q}'_1)$ represent the pairs of 3D lines and points in the first and second frames, respectively). The symbol $\otimes$ denotes the Kronecker product. This operation is used to simplify algebraic equations of the type $AXB = C$ by

$$\left(B^T \otimes A\right) \text{vec}(X) = \text{vec}(C) \tag{1}$$

where vec(.) represents the stacking of the columns of a matrix.

4 Problem Definition and Method Overview

We aim to estimate a rigid transformation $T \overset{\text{def}}{=} (R, \mathbf{t})$ that aligns two 3D point cloud 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 & line matches.

4.1 Our approach

We follow a standard two-step procedure for obtaining a robust estimation. First, we get a rough estimate for the transformation and remove outliers. Then, we use the inliers and first guess for the 3D registration to refine the solution. As mentioned in Sec. 1.1, we are interested in exploring different kinds of features for improving speed and accuracy.

Geometric constraints:

Before starting with the solvers, we define the possible feature matches that may arise from line intersection constraints. The first constraint is the intersecting constraint from one line in the first scan and a second line in the second scan. The constraint arises because these two lines must intersect in 3D.

The next set of constraints arises because these 3D lines are usually obtained from the intersection of planar structures. This means that, in general, we also have plane matches. The intersection point of the lines in 3D may also be extracted from both scans. Note that this is not necessarily a point belonging to the 3D scans, rather a common 3D point in both scans that results from the intersection of sets of two lines that are not necessarily matches of lines. Therefore, although this is not possible for every possible line intersection constraint, it makes sense to consider the case of point matches in the solvers. Another kind of constraint that can occur is the line matches. For example, assume we have a pair of intersecting lines, and one of them is a strong edge of a planar structure. Then, there is a chance that this edge is present in both scans.

A detailed geometric description of the considered constraints is shown in Sec. 4.2.

Coarse estimate:

For the first step of our solution, we propose to use a RANSAC-based estimator that finds a set of inliers for the kinds of features described in the previous paragraph and outputs a rough estimate for the transformation that aligns both scans. Our Hybrid RANSAC loop is described in Sec. 6. The main characteristic of our RANSAC is that it handles multiple solvers representing the constraints described in the previous paragraph.

As indicated above, the minimal solvers are used to model the problem into the RANSAC-based estimators. Thus, taking into account the problem tackled in this work, i.e., the constraints identified in the previous paragraph, in Sec. 5 we propose a set of minimal solvers that exploit point, plane & line correspondences, and line intersections.

Registration refinement:

Finally, in the last step of our approach, the inlier set yielded by the RANSAC is used in a refinement method derived in Sec. 7. The proposed solution is of alternative minimization type, alternating between estimating the rotation (having the translation parameters fixed) and estimating the translation (having the rotation parameters fixed). The solvers are of the steepest descent type

in the respective manifolds, and we formulate the cost function of every feature mentioned in the previous paragraphs.

4.2 Geometric constraints

This subsection describes the constraints and the respective distances that may arise from line intersection features. We use these metrics for inlier counting in our RANSAC framework and for the formalization of our refinement technique.

Line intersection:

Consider the tuple $(\mathbf{l}, \mathbf{m}')$ representing two intersecting lines in 3D, in two different scans. The constraint that ensures they intersect in the world was derived in [92], and is given as

$$\mathbf{m}'^{T} \begin{bmatrix} -[\mathbf{t}]_{\times} R & R \\ R & \mathbf{0} \end{bmatrix} \mathbf{l} = 0. \tag{2}$$

Point match:

Consider now the tuple $(\mathbf{q}, \mathbf{q}')$ representing a point correspondence in two 3D scans. The geometrical constraints used for this feature is the Euclidean distance between two 3D points. It is given as

$$d_{\mathcal{Q}} = \|R\mathbf{q} + \mathbf{t} - \mathbf{q}'\|. \tag{3}$$

Plane correspondence:

The metric considered for planes correspondences in our prior work, [90], was an algebraic constraint; the Euclidean norm of the difference between the coordinates in each frame. In this work, we will exploit a geometric constraint. To achieve this, we sample three points in the plane of the first frame, compute the distance to the plane in the second frame for a given rigid transformation, and average those distances. Let us consider the tuple $(\boldsymbol{\pi}, \boldsymbol{\pi}')$ representing a point correspondence, the metric is given as

$$d_{\Pi} = \frac{1}{3} \sum_{j=1}^{3} \|(R\mathbf{q}_j + \mathbf{t})^T \overline{\boldsymbol{\pi}}' + \widetilde{\pi}'\|, \quad \text{where } \mathbf{q}_j \in \boldsymbol{\pi}. \tag{4}$$

Line correspondence:

Finally, we consider a constraint similar to the one used for planes for line correspondences. Two points are sampled from the line in the first frame; we use the endpoints of the line segment. The rigid transformation is applied to the sampled points, and the distance of the points to the line in the second frame is computed and averaged. Let the tuple $(\mathbf{l}, \mathbf{l}')$ represent a line correspondence in two scans. The geometric constraint is given as

$$d_{\mathcal{M}} = \frac{1}{2} \sum_{j=1}^{2} \|(\mathbf{q}'_{\mathbf{l}} - R\mathbf{q}_{j,\mathbf{l}} + \mathbf{t}) \times \vec{\mathbf{l}}'\|, \tag{5}$$

where $\mathbf{q}'_{\mathbf{l}} = \vec{\mathbf{l}}' \times \widehat{\mathbf{l}}'$ is a point on the line $\mathbf{l}'$. See [91] for more detail.

5 Minimal Solvers

This section presents new minimal solvers for 3D scan alignment using line intersections and plane, line & point correspondences. First, we describe the strategy for obtaining the solvers, and each following subsection derives a solver in Tab. 1.

5.1 Getting the solvers

The solvers are described by a single-degree polynomial equation, in which the degrees of the polynomials get us the maximum number of solutions to the problem. Instead of trying to describe all the involved constraints and put them in an automatic generator (such as [93]), we focus on removing unknowns from the problem by first looking at the geometric properties of the data and then trying to derive the polynomial coefficients analytically[1].

The strategy followed for all the solvers is presented next. First, we split the features into two types, line intersections constraints, and feature matches. Since the match constraints are stronger and easy to manipulate geometrically, we start with these and remove as many degrees of freedom from the problem as possible. Then, we plug the remaining degrees of freedom in the intersection constraints and derive the single unknown polynomial equation using algebraic manipulation

[1] There are pros and cons to both approaches. Both need to apply transformations to the data to get a single-unknown polynomial equation. However, we understand the problem better and its degeneracies by deriving the polynomial analytically.

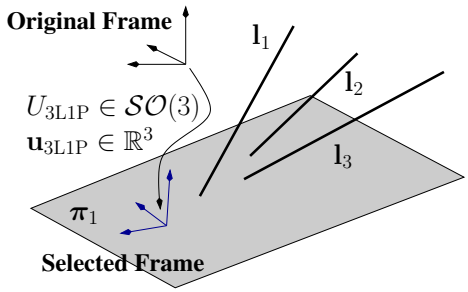

Figure 2: **3L1P:** Selected frame for the 1 planes match and 3 line intersection solver.

with the remaining degrees of freedom. From all the nine solvers, we could get a single polynomial equation for solving the problem in eight of them by deriving the coefficients analytically. The proposed and existing solvers are presented in Tab. 1.

5.2 3L1P: 3 line intersections and 1 plane match

Consider three pairs of intersecting lines $\{(\mathbf{l}_1, \mathbf{m}'_1), (\mathbf{l}_2, \mathbf{m}'_2), (\mathbf{l}_3, \mathbf{m}'_3)\}$ and one pair of corresponding planes $(\boldsymbol{\pi}_1, \boldsymbol{\pi}'_1)$. Following what we described in the previous subsection, we start by selecting appropriate frames for the problem, by considering the match features.

Selected frames:

We select coordinate systems to the two 3D scans, with planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}'_1$ set as the $xy-$plane. To achieve this, we start by defining the third column of U_{3L1P}:

$$\mathbf{u}_3 = \overline{\boldsymbol{\pi}}_1 / \|\overline{\boldsymbol{\pi}}_1\|. \tag{6}$$

Then, we set two possible guesses for $\mathbf{u}_1$ (this rotation can be defined up to a rotation degree of freedom):

$$\mathbf{u}_1^- = [0\ 1\ 0] \times \mathbf{u}_3 \quad \text{and} \quad \mathbf{u}_1^+ = [1\ 0\ 0] \times \mathbf{u}_3, \tag{7}$$

and set the first column of U_{3L1P} as $\mathbf{u}_1 = \mathbf{u}_1^* / \|\mathbf{u}_1^*\|$ where $\mathbf{u}_1^*$ is equal to the vector in (7) with the larger norm. Then, we define

$$U_{3L1P} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_3 \times \mathbf{u}_1 & \mathbf{u}_3 \end{bmatrix}, \quad \text{and} \tag{8}$$

$$\mathbf{u}_{3L1P} = \widetilde{\pi}_1 U_{3L1P} \overline{\boldsymbol{\pi}}_1. \tag{9}$$

A graphical representation of this selected frame is shown in Fig. 2. With both frames associated with the two scans verifying these specifications, the relative transformation between both scans is given by[2]

$$R = \frac{1}{1+s^2} \begin{bmatrix} 1-s^2 & -2s & 0 \\ 2s & 1-s^2 & 0 \\ 0 & 0 & 1+s^2 \end{bmatrix} \text{ and } \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix} \tag{10}$$

which means that we reduced the total degrees of freedom from six to three.

Solver:

Following what we describe in Sec. 5.1, to compute the unknowns t_x, t_y, and s we use the three line intersections. Replacing R and $\mathbf{t}$ in (2) by the ones in (10) and multiplying the result by $(1+s^2)$ we get three constraints of the form

$$\kappa_1^3[s, t_x, t_y] = \kappa_2^3[s, t_x, t_y] = \kappa_3^3[s, t_x, t_y] = 0, \tag{11}$$

where $\kappa_k^j[.]$[3] denotes the k^{th} polynomial with degree j. The monomials in these polynomials are linear in t_x and t_y; and quadratic in s. Taking the first and second algebraic constraints in (11), and solving them for t_x and t_y, we get

$$t_x = \frac{\kappa_4^4[s]}{\kappa_5^4[s]} \quad \text{and} \quad t_y = \frac{\kappa_6^4[s]}{\kappa_7^4[s]}. \tag{12}$$

Now, substituting t_x and t_y in the third constraint of (11) by (12) and simplifying the equation, we get

$$\frac{\kappa_8^4[s]}{\kappa_9^2[s]} = 0 \implies \kappa_8^4[s] = 0. \tag{13}$$

To compute the transformation between both scans, we find the roots of the four degree polynomial equation $\kappa_8^4[s]$, which can be computed in closed-form, e.g., by using the *Ferrari*'s formula; we get up to four solutions for s. Then, for each s, we get solutions for t_x and t_y by solving (12). The correct transformation is obtained by replacing t_x, t_y, and s in (10) and reversing the predefined transformations U_{3L1P} and $\mathbf{u}_{3L1P}$.

[2] We are using Cayley's parameterization for $\mathcal{SO}(3)$ matrices because they allow a more compact representation and are more suited for deriving the polynomials.

[3] Due to space limitations, we omit the coefficients and monomials.

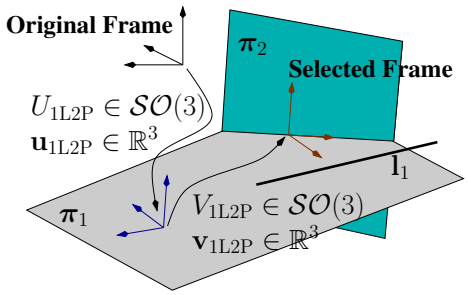

Figure 3: **1L2P:** Representation of the selected coordinate system for the 2 planes matches and one line intersection case.

5.3 1L2P: 1 line intersection and 2 plane matches

Consider one line intersection, $(\mathbf{l}_1, \mathbf{m}'_1)$, and two 3D plane matches, $\{(\boldsymbol{\pi}_1, \boldsymbol{\pi}'_1), (\boldsymbol{\pi}_2, \boldsymbol{\pi}'_2)\}$. Again, we follow the steps of Sec. 5.1.

Selected Frames:

As shown in Fig. 3, we transform the data in both 3D scans, to satisfy:

1. Planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}'_1$ are in the xy–plane;
2. The x–axis of both frames is along the intersection of the planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ in the first 3D scan. Similarly, the x–axis is along the intersection of the planes $\boldsymbol{\pi}'_1$ and $\boldsymbol{\pi}'_2$ in the second 3D scan.

The transformation arising from item 1, $U_{1L2P} \in \mathcal{SO}(3)$ and translation $\mathbf{u}_{1L2P} \in \mathbb{R}^3$, are obtained in a similar fashion to U_{3L1P} and $\mathbf{u}_{3L1P}$, as shown in Sec. 5.2. For $V_{1L2P} \in \mathcal{SO}(3)$ and $\mathbf{v}_{1L2P} \in \mathbb{R}^3$ associated with item 2, we first find the 3D line $\mathbf{r}$ (in *Plücker* coordinates) that represents the intersection of the two planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ (see more details in [91]):

$$\mathbf{r} = \left[\overline{\boldsymbol{\pi}}_1 \times \overline{\boldsymbol{\pi}}_2 \;\; \widetilde{\pi}_1 \overline{\boldsymbol{\pi}}_2 - \widetilde{\pi}_2 \overline{\boldsymbol{\pi}}_1\right]. \tag{14}$$

Then, we define

$$V_{1L2P} = \begin{bmatrix} r_1/\sqrt{r_1^2 + r_2^2} & -r_2/\sqrt{r_1^2 + r_2^2} & 0 \\ r_2/\sqrt{r_1^2 + r_2^2} & r_1/\sqrt{r_1^2 + r_2^2} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{15}$$

where r_1 and r_2 are the 1st and 2nd elements in vector $\mathbf{r}$. Now, for the translation vector $\mathbf{v}_{1L2P}$, we first compute the closest 3D point $\mathbf{x} \in \mathbb{R}^3$ of line $\mathbf{r}$ to the origin as

$$\mathbf{x} = \widehat{\mathbf{r}} \times \overline{\mathbf{r}}, \tag{16}$$

and $\mathbf{v}_{1L2P}$ is given by

$$\mathbf{v}_{1L2P} = V_{1L2P}\mathbf{x}. \tag{17}$$

After applying these predefined transformations to the two 3D scans, the relative transformation is determined up to a single translation parameter. More specifically, the relative transformation can be expressed as

$$R = I \;\; \text{and} \;\; \mathbf{t} = \begin{bmatrix} t_x & 0 & 0 \end{bmatrix}^T. \tag{18}$$

where I is the 3×3 identity matrix.

Solver:

Since we only have one unknown t_x, we only need one intersecting line constraint. By substituting R and $\mathbf{t}$ in (2) by (18) and solving for t_x, we get

$$t_x = \frac{\overline{\mathbf{l}}_1^T \widehat{\mathbf{m}}'_1 + \widehat{\mathbf{l}}_1^T \overline{\mathbf{m}}'_1}{\overline{l}_{1,2}\overline{m}'_{1,3} - \overline{l}_{1,3}\overline{m}'_{1,2}}, \tag{19}$$

where the subscript i in $l_{1,i}$ denotes 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 (18), and revert to the original coordinate frames by using predefined transformations (U_{1L2P}, $\mathbf{u}_{1L2P}$, V_{1L2P}, and $\mathbf{v}_{1L2P}$).

5.4 3L1Q: 3 line intersections and 1 point match

Consider three pairs of intersecting lines $\{(\mathbf{l}_1, \mathbf{m}'_1), (\mathbf{l}_2, \mathbf{m}'_2), (\mathbf{l}_3, \mathbf{m}'_3)\}$, and one point match $(\mathbf{q}_1, \mathbf{q}'_1)$.

Selected frames:

We apply predefined transformations to the both 3D scan frames, such that points $\mathbf{q}_1$ and $\mathbf{q}'_1$ are the origin of the coordinate systems. In this case the transformation consists only of a translation, since we are moving the origin of the 3D data to the point $\mathbf{q}_1$. Thus, the $U_{3L1Q} \in \mathcal{SO}(3)$ and $\mathbf{u}_{3L1Q} \in \mathbb{R}^3$ in Fig. 4, are represented as

$$U_{3L1Q} = I \;\; \text{and} \;\; \mathbf{u}_{3L1Q} = -\mathbf{q}_1. \tag{20}$$

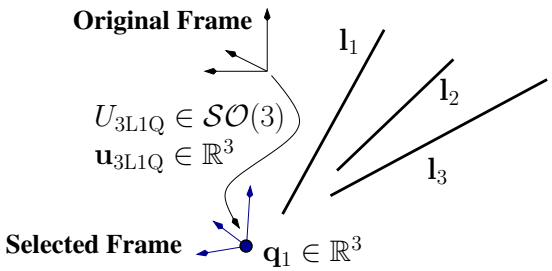

Figure 4: **3L1Q:** Coordinate system for the 1 point correspondence and 3 line intersection case.

Having translated both coordinate systems to $\mathbf{q}_1$ and $\mathbf{q}'_1$, respectively, the transformation between both frames is given only by a rotation $R \in \mathcal{SO}(3)$ (three rotational degrees of freedom), i.e., $\mathbf{t} = \mathbf{0}$.

For parametrizing the R, we again use Cayley's parameterization (in this case for 3D rotation) as follows:

$$R = \frac{1}{1+s_1^2+s_2^2+s_3^2} R^\dagger, \tag{21}$$

where

$$R^\dagger = \begin{bmatrix} 1+s_1^2-s_2^2-s_3^2 & 2s_1s_2-2s_3 & 2s_2+2s_1s_3 \\ 2s_1s_2+2s_3 & 1-s_1^2+s_2^2-s_3^2 & 2s_2s_3-2s_1 \\ 2s_1s_3+2s_2 & 2s_1+2s_2s_3 & 1-s_1^2-s_2^2+s_3^2 \end{bmatrix}. \tag{22}$$

Solver:

To obtain the rotation matrix, the three pairs of intersecting lines are used. Setting $\mathbf{t} = \mathbf{0}$ in (2), we obtain three linear independent equations of the form[4]

$$\overline{\mathbf{m}}_i'^T R^\dagger \widehat{\mathbf{l}}_i + \widehat{\mathbf{m}}_i'^T R^\dagger \bar{\mathbf{l}}_i = 0, \tag{23}$$

with $i = 1, 2, 3$. Now, by substituting $R^\dagger$ in (23) by (22), we get

$$\begin{aligned} a_{i,9}s_1^2 + a_{i,8}s_1s_2 + a_{i,7}s_1s_3 + a_{i,6}s_1 + \\ a_{i,5}s_2^2 + a_{i,4}s_2s_3 + a_{i,3}s_2 + \\ a_{i,2}s_3^2 + a_{i,1}s_3 + a_{i,0} = 0, \end{aligned} \tag{24}$$

where

$$a_{i,0} = \bar{\mathbf{l}}_i^T \widehat{\mathbf{m}}_i^T + \widehat{\mathbf{l}}_i^T \overline{\mathbf{m}}_i^T \tag{25}$$

$$a_{i,1} = 2\bar{l}_{i,1}\widehat{m}'_{i,2} - 2\bar{l}_{i,2}\widehat{m}'_{i,1} + 2\widehat{l}_{i,1}\overline{m}'_{i,2} - 2\widehat{l}_{i,2}\overline{m}'_{i,1} \tag{26}$$

$$a_{i,2} = -\bar{l}_{i,1}\widehat{m}'_{i,1} - \bar{l}_{i,2}\widehat{m}'_{i,2} + \bar{l}_{i,3}\widehat{m}'_{i,3} - \widehat{l}_{i,1}\overline{m}'_{i,1} - \widehat{l}_{i,2}\overline{m}'_{i,2} + \widehat{l}_{i,3}\overline{m}'_{i,3} \tag{27}$$

$$a_{i,3} = -2\bar{l}_{i,1}\widehat{m}'_{i,3} + 2\bar{l}_{i,3}\widehat{m}'_{i,1} - 2\widehat{l}_{i,1}\overline{m}'_{i,3} + 2\widehat{l}_{i,3}\overline{m}'_{i,1} \tag{28}$$

$$a_{i,4} = 2\bar{l}_{i,2}\widehat{m}'_{i,3} + 2\bar{l}_{i,3}\widehat{m}'_{i,2} + 2\widehat{l}_{i,2}\overline{m}'_{i,3} + 2\widehat{l}_{i,3}\overline{m}'_{i,2} \tag{29}$$

$$a_{i,5} = -\bar{l}_{i,1}\widehat{m}'_{i,1} + \bar{l}_{i,2}\widehat{m}'_{i,2} - \bar{l}_{i,3}\widehat{m}'_{i,3} - \widehat{l}_{i,1}\overline{m}'_{i,1} + \widehat{l}_{i,2}\overline{m}'_{i,2} - \widehat{l}_{i,3}\overline{m}'_{i,3} \tag{30}$$

$$a_{i,6} = 2\bar{l}_{i,2}\widehat{m}'_{i,3} - 2\bar{l}_{i,3}\widehat{m}'_{i,2} + 2\widehat{l}_{i,2}\overline{m}'_{i,3} - 2\widehat{l}_{i,3}\overline{m}'_{i,2} \tag{31}$$

$$a_{i,7} = 2\bar{l}_{i,1}\widehat{m}'_{i,3} + 2\bar{l}_{i,3}\widehat{m}'_{i,1} + 2\widehat{l}_{i,1}\overline{m}'_{i,3} + 2\widehat{l}_{i,3}\overline{m}'_{i,1} \tag{32}$$

$$a_{i,8} = 2\bar{l}_{i,1}\widehat{m}'_{i,2} + 2\bar{l}_{i,2}\widehat{m}'_{i,1} + 2\widehat{l}_{i,1}\overline{m}'_{i,2} + 2\widehat{l}_{i,2}\overline{m}'_{i,1} \tag{33}$$

$$a_{i,9} = \bar{l}_{i,1}\widehat{m}'_{i,1} - \bar{l}_{i,2}\widehat{m}'_{i,2} - \bar{l}_{i,3}\widehat{m}'_{i,3} + \widehat{l}_{i,1}\overline{m}'_{i,1} - \widehat{l}_{i,2}\overline{m}'_{i,2} - \widehat{l}_{i,3}\overline{m}'_{i,3}. \tag{34}$$

Now, if we have three line intersection constraints $\{(\mathbf{l}_1, \mathbf{m}'_1), (\mathbf{l}_2, \mathbf{m}'_2), (\mathbf{l}_3, \mathbf{m}'_3)\}$, we get three algebraic constraints of type (24); three 2-degree polynomial equations with three unknowns. Geometrically, solving these three constraints for s_1, s_2, and s_3 corresponds to finding the intersection points of three 3D quadrics. It is also known that, in general, this problem has up to 8 solutions. In fact, the work in [94] proposes a set of derivations for getting an efficient solution to this problem. Following [94][5], we algebraically derive a single variable 8-degree polynomial equation as a function of s_1. After getting the solutions[6] (up to eight like the solution in the first draft), we compute s_2 and s_3 analytically by back-substituting s_1 in the derived equations.

5.5 1L2Q: 1 line intersection and 2 point matches

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

Selected Frames:

For obtaining a suitable frame, we consider predefined transformations $U_{1L2Q}, V_{1L2Q} \in \mathcal{SO}(3)$ and $\mathbf{u}_{1L2Q} \in \mathbb{R}^3$, 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.

From item 1, the first transformation consists in a pure translation to the point $\mathbf{q}_1$, which can be

[4] Notice we have simplified the constraint by pre-multiplying $1 + s_1^2 + s_2^2 + s_3^2$

[5] We do not show all these derivations for space purposes.

[6] We solved the polynomial using the `Eigen`'s library `PolynomialSolver` class.

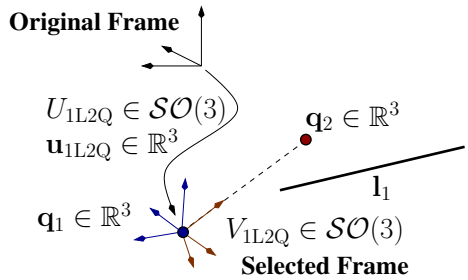

Figure 5: **1L2Q:** Representation of the selected coordinate system for the 2 point matches and 1 line intersection case.

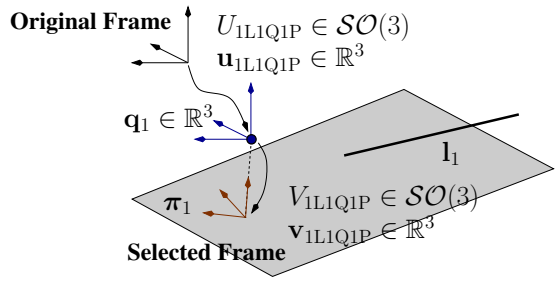

Figure 6: **1L1Q1P:** Selected frame for the 1 point and 1 plane matches, and 1 line intersection case.

written as

$$U_{1L2Q} = I, \quad \text{and} \quad \mathbf{u}_{1L2Q} = -\mathbf{q}_1, \tag{35}$$

similar to the predefined transformation in Sec. 5.4. The item 2 consists in a pure rotation V_{3L1Q}, which can be obtained in a similar fashion to the matrix U_{1L2P} in Sec. 5.3. Both cases can be seen as aligning the $z-$axis with either the plane normal in Sec. 5.3 or the direction from $\mathbf{q}_1$ to $\mathbf{q}_2$. A graphical representation of these transformations is presented in Fig. 5. With these settings, we are left with a single rotational degree-of-freedom ($\mathbf{t} = \mathbf{0}$), which corresponds to the rotation about the z–axis. Then, R can be written as

$$R = \frac{1}{1+s^2}\begin{bmatrix} 1-s^2 & -2s & 0 \\ 2s & 1-s^2 & 0 \\ 0 & 0 & 1+s^2 \end{bmatrix}. \tag{36}$$

Solver:

By setting $\mathbf{t} = \mathbf{0}$ and R as defined in (36) into the single constraint (2), and multiplying both sides by $(1+s^2)$, we obtain a second order polynomial in s as

$$\mu_2 s^2 + \mu_1 s + \mu_0 = 0, \tag{37}$$

where

$$\mu_0 = \bar{\mathbf{l}}_1^T \widehat{\mathbf{m}}'_1 + \widehat{\mathbf{l}}_1^T \overline{\mathbf{m}}'_1 \tag{38}$$

$$\mu_1 = 2\bar{l}_{1,1}\widehat{m}'_{1,2} - 2\bar{l}_{1,2}\widehat{m}'_{1,1} + 2\widehat{l}_{1,1}\overline{m}'_{1,2} - 2\widehat{l}_{1,2}\overline{m}'_{1,1} \tag{39}$$

$$\mu_2 = \bar{l}_{1,3}\widehat{m}'_{1,3} - \widehat{l}_{1,1}\overline{m}'_{1,1} - \bar{l}_{1,2}\widehat{m}'_{1,2} - \widehat{l}_{1,2}\overline{m}'_{1,2} - \bar{l}_{1,1}\widehat{m}'_{1,1} + \widehat{l}_{1,3}\overline{m}'_{1,3}. \tag{40}$$

The polynomial (37) can be solved in closed-form with the quadratic formula, yielding two solutions for parameter s. Both values of s are then replaced in (36) to retrieve the relative rotation between the two frames. The final transformation is obtained by reverting the predefined transformations U_{1L2Q}, V_{1L2Q}, and $\mathbf{u}_{1L2Q}$.

5.6 1L1Q1P: 1 line intersection and 1 point & plane matches

Consider the scenario where one pair of intersection lines $(\mathbf{l}_1, \mathbf{m}'_1,)$, one pair of plane correspondences $(\boldsymbol{\pi}_1, \boldsymbol{\pi}'_1)$, and one pair of point correspondences $(\mathbf{q}_1, \mathbf{q}'_1)$ are available.

Selected Frames:

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

1. The orthogonal projection of points $\mathbf{q}_1$ and $\mathbf{q}'_1$ to the planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}'_1$ (i.e., the projection through the normal direction of the planes) are the origin of the coordinate systems; and
2. The planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}'_1$ match the xy–plane.

The transformation associated to item 1 consists in a pure translation to the projection of point $\mathbf{q}_1$ to $\boldsymbol{\pi}_1$. This projection can be obtained by computing the signed distance $d_{\mathbf{q}_1\boldsymbol{\pi}_1}$ of the point to the plane, and then subtracting the plane normal vector scaled by the distance to the point. The signed distance can be obtained by inputting the point in the plane equation as

$$d_{\mathbf{q}_1\boldsymbol{\pi}_1} = \overline{\boldsymbol{\pi}}^T \mathbf{q} + \widetilde{\pi}. \tag{41}$$

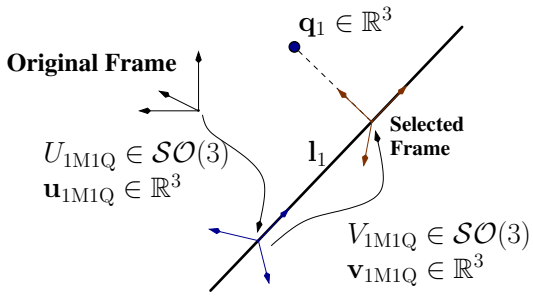

Figure 7: **1M1Q:** Graphical representation of the selected coordinate system for the case of 1 point match and 1 line correspondence.

The point $\mathbf{q}_{\boldsymbol{\pi}}$ in the plane corresponding to the projection of $\mathbf{q}$ to $\boldsymbol{\pi}$ is obtain by

$$\mathbf{q}_{\boldsymbol{\pi}} = \mathbf{q} - d_{\mathbf{q}_1 \boldsymbol{\pi}_1} \overline{\boldsymbol{\pi}}. \tag{42}$$

The first transformation is thus given by

$$U_{1L1Q1P} = I, \quad \text{and} \quad \mathbf{u}_{1L1Q1P} = -\mathbf{q}_{\pi}, \tag{43}$$

The second step, corresponding to item 2, consists in a pure rotation V_{1L1Q1P}, which can be obtained in a similar fashion to the matrix U_{1L2P} in Sec. 5.3. The result is shown in Fig. 6.

After applying the predefined transformations to both reference frames, the relative pose is given by a single rotation parameter, similar to (36) and $\mathbf{t} = \mathbf{0}$.

Solver:

The missing parameter from the previous paragraph corresponds to a rotation around the z–axis, which align both frames. To compute this rotation the procedure of Sec. 5.5 was used. Once again the solver yields two solutions.

5.7 1M1Q: 1 line correspondence and 1 point match

In this scenario, consider one pair of correspondent lines $(\mathbf{l}_1, \mathbf{l}'_1,)$, and one pair of point matches $(\mathbf{q}_1, \mathbf{q}'_1)$. Given those features, the pose can be retrieved directly from the selected frame as follows.

Selected Frames:

Consider predefined transformations $U_{1M1Q} \in \mathcal{SO}(3)$ & $\mathbf{u}_{1M1Q} \in \mathbb{R}^3$ and $V_{1M1Q} \in \mathcal{SO}(3)$ & $\mathbf{v}_{1M1Q} \in \mathbb{R}^3$ such that:

1. The origin of the reference frame is a point in the line $\mathbf{l}_1$, with the $z-$axis parallel to the line direction; and
2. The origin of the reference frame is moved to the projection of $\mathbf{q}_1$ onto $\mathbf{l}_1$, with the $x-$axis pointing towards the point.

A graphical representation of these transformations is presented in Fig. 7. The transformation associated to item 1 is obtained as follows. The rotation matrix U_{1M1Q} can be obtained in a similar fashion to U_{3L1P}, from Sec. 5.2, but defining the third column to be the normalized direction of $\mathbf{l}_1$. The translation vector $\mathbf{u}_{1M1Q}$ is defined as

$$\mathbf{u}_{1M1Q} = \mathbf{x} - \lambda_1 \bar{\mathbf{l}}_1, \tag{44}$$

with $\mathbf{x}$ being the closest point to the origin in the line $\mathbf{l}_1$ as defined in (16), and $\lambda_1 = \mathbf{x}^T \bar{\mathbf{l}}_1$. Concerning item 2, the translation vector $\mathbf{v}_{1M1Q}$ is given as

$$\mathbf{v}_{1M1Q} = -\lambda_2 \bar{\mathbf{l}}_1, \tag{45}$$

where $\lambda_2 = \mathbf{q}_1^T \bar{\mathbf{l}}_1$. The rotation V_{1M1Q} is a rotation around the $z-$axis which aligns the $x-$axis with the line $\mathbf{l}_1$ joining $\mathbf{q}_1$ with its projection on the line. This is defined as

$$V_{1M1Q} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{46}$$

where θ is the the angle between the direction of $\mathbf{l}_1$ and the $x-$axis. Notice that, in this case there are no remaining degrees-of-freedom to estimate. Thus, no line intersections are used.

Solver:

To compute the poses, we just have to revert the predefined transformations, by taking the inverse of those in the second scan and applying them to the ones in the first scan. We get a single solution to the problem.

5.8 2M: 2 line correspondence

Consider two pairs of lines matches $(\mathbf{l}_1, \mathbf{l}'_1,)$, and $(\mathbf{l}_2, \mathbf{l}'_2,)$. Given those features, the transformation can be obtained directly from the selected frame as follows.

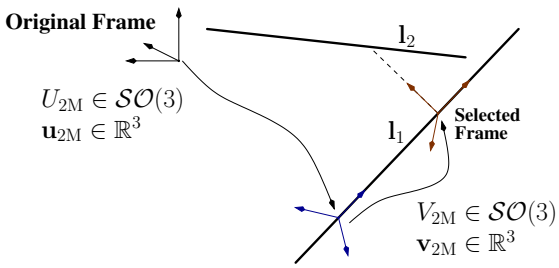

Figure 8: **2M:** Depiction of the selected frames for the case of 2 line matches.

Selected Frames:

Consider predefined transformations $U_{2M} \in \mathcal{SO}(3)$ & $\mathbf{u}_{2M} \in \mathbb{R}^3$ and $V_{2M} \in \mathcal{SO}(3)$ & $\mathbf{v}_{2M} \in \mathbb{R}^3$ such that:

1. The origin of the reference frame is a point in the line $\mathbf{l}_1$, with $z-$axis is parallel to the line direction; and
2. The origin of the reference frame is moved to the projection of closest point in $\mathbf{l}_1$ to $\mathbf{l}_2$, with the $x-$axis pointing towards that point.

All transformations involved in items 1 and 2 are obtained in a similar fashion to Sec. 5.7. The difference lies in instead of projecting a point correspondence to $\mathbf{l}_1$, we need to find the closest points in each line to the other. Let us consider $\mathbf{q}_1$ and $\mathbf{q}_2$ to be points in lines $\mathbf{l}_1$ and $\mathbf{l}_2$, respectively, $\mathbf{n} = \bar{\mathbf{l}}_1 \times \bar{\mathbf{l}}_2$, $\mathbf{n}_1 = \bar{\mathbf{l}}_1 \times \mathbf{n}$, and $\mathbf{n}_2 = \bar{\mathbf{l}}_2 \times \mathbf{n}$. Then, the nearest points $\hat{\mathbf{q}}_1$ and $\hat{\mathbf{q}}_2$ in line $\mathbf{l}_1$ closer to line $\mathbf{l}_2$, and vice-versa, are given as

$$\hat{\mathbf{q}}_1 = \mathbf{q}_1 + \frac{(\mathbf{q}_2 - \mathbf{q}_1)^T \mathbf{n}_2}{\bar{\mathbf{l}}_1^T \mathbf{n}_2} \mathbf{l}_1 \tag{47}$$

$$\hat{\mathbf{q}}_2 = \mathbf{q}_2 + \frac{(\mathbf{q}_1 - \mathbf{q}_2)^T \mathbf{n}_1}{\bar{\mathbf{l}}_2^T \mathbf{n}_1} \mathbf{l}_2. \tag{48}$$

A graphical representation of these transformations is presented in Fig. 8. Similar to the previous solver, there are no remaining degrees-of-freedom left to estimate.

Solver:

The solver for this case follows the one in Sec. 5.7.

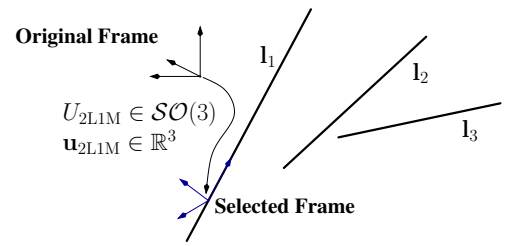

Figure 9: **2L1M:** Coordinate systems for the 1 line match and 2 line intersections.

5.9 2L1M: 2 line intersections and 1 line correspondence

Now, consider one pair of correspondent lines $(\mathbf{l}_1, \mathbf{l}'_1,)$, and two pairs of intersecting lines $(\mathbf{l}_2, \mathbf{m}'_2,)$ and $(\mathbf{l}_3, \mathbf{m}'_3,)$. Given those features the pose can be retrieved as follows.

Selected Frames:

Consider predefined transformations $U_{2L1M} \in \mathcal{SO}(3)$ & $\mathbf{u}_{2L1M} \in \mathbb{R}^3$ such that the origin of the reference frame is a point in the line $\mathbf{l}_1$, with $z-$axis parallel to the line's direction. These transformations are obtained in a similar fashion to Sec. 5.7. Given this setup, there are two degrees-of-freedom to compute, a rotation and a translation on the $z-$axis:

$$R = \frac{1}{1+s^2} \begin{bmatrix} 1-s^2 & -2s & 0 \\ 2s & 1-s^2 & 0 \\ 0 & 0 & 1+s^2 \end{bmatrix} \text{ and } \mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ t_z \end{bmatrix}. \tag{49}$$

A graphical representation of these transformations is presented in Fig. 9.

Solver:

The two remaining unknowns s and t_z can be computed by using the two line intersection constraints. Replacing (49) in (2), we get the following two constraints:

$$k^3_{10}[s, t_z] = k^3_{11}[s, t_z] = 0. \tag{50}$$

Solving the first for t_z (which is linear in t_z), we get

$$t_z = \frac{k^2_{12}[s]}{k^2_{13}[s]}. \tag{51}$$

Then, substituting (51) in the second constraint in (50), and simplifying the equation yields

$$\frac{k_{14}^4[s]}{k_{15}^2[s]} = 0 \implies k_{14}^4[s] = 0. \tag{52}$$

To compute s, we find the root of the four degree polynomial $k_{14}^4[s]$, which can be obtained in closed-form, yielding up to four solutions. These solutions are injected in (51) to obtain the solutions of the translation parameter. The final transformation is obtained by reversing the pre-defined transformations.

5.10 1M1P: 1 line correspondence and 1 plane match

Now, consider one pair of line matches $(\mathbf{l}_1, \mathbf{l}'_1,)$, and one pair of plane matches $(\boldsymbol{\pi}_1, \boldsymbol{\pi}'_1)$.

Selected Frames:

Let us consider the predefined transformations $U_{1\text{M1P}} \in \mathcal{SO}(3)$ & $\mathbf{u}_{1\text{M1P}} \in \mathbb{R}^3$ and $V_{1\text{M1P}} \in \mathcal{SO}(3)$ such that:

1. The origin of the reference frame is the intersection point between the line $\mathbf{l}_1$ and the plane $\boldsymbol{\pi}_1$, with the latter being the $xy-$plane; and
2. The $x-$axis pointing towards the projection of $\bar{\mathbf{l}}_1$ onto plane $\boldsymbol{\pi}_1$.

The rotation matrix $U_{1\text{M1P}}$ in item 1 can be obtained in a similar fashion to $U_{3\text{L1P}}$, in Sec. 5.2. The translation vector $\mathbf{u}_{1\text{M1P}}$ is defined as

$$\mathbf{u}_{1\text{M1P}} = -\frac{\overline{\boldsymbol{\pi}}_1 \times \widehat{\mathbf{l}}_1 - \widetilde{\pi}_1 \bar{\mathbf{l}}_1}{\overline{\boldsymbol{\pi}}_1^T \bar{\mathbf{l}}_1}. \tag{53}$$

The transformation in item 2 consists of a rotation around the $z-$axis, which aligns the $x-$axis with the projection of $\bar{\mathbf{l}}_1$ onto plane $\boldsymbol{\pi}_1$. This projection is defined as

$$\overline{\mathbf{v}} = \bar{\mathbf{l}}_1 - (\bar{\mathbf{l}}_1^T \overline{\boldsymbol{\pi}}_1)\overline{\boldsymbol{\pi}}_1. \tag{54}$$

Then, the rotation $V_{1\text{M1P}}$ is given as

$$V_{1\text{M1P}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{55}$$

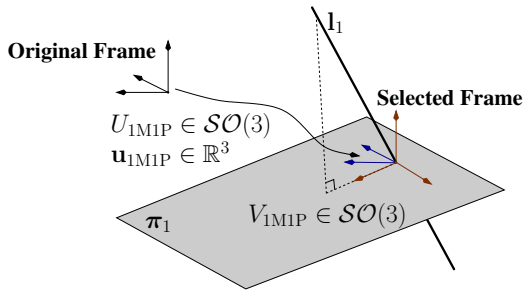

Figure 10: **1M1P:** Selected frame for the 1 line match and 1 plane correspondence case.

where θ is the the angle between $\overline{\mathbf{v}}$ and the $x-$axis. A graphical representation of these transformations is presented in Fig. 10. In this case, as for the solvers exploiting only correspondences, it is possible to obtain all degrees-of-freedom directly from the predefined transformations.

Solver:

To compute the poses, we only have to revert the predefined transformations, by taking the inverse of those in the second scan and applying them to the ones in the first scan. We have a single solution to this problem.

6 Hybrid RANSAC

Typical RANSAC loops rely on a single solver to obtain the model. For instance, we would use each solver in Tab. 1 in a separate RANSAC loop and assess which solver provided the best pose estimation. However, in this work, we argue that the solvers should not be made to compete but to complete each other. Given this insight and the recent results on Hybrid Pose Estimation presented in [63] and in 3D Registration in [90], we propose a RANSAC loop, which in each iteration selects the solver to use, see Alg. 1.

The goal is to select a solver given the inlier ratios of the different features, the minimal set, i.e., the number of features the minimal solver uses, the number of iterations the solver was selected, and a prior. In a noisy data set, it is not possible to know *a priori* the actual inlier ratios, so we use the inlier information from previous runs of the algorithm in a sequence. The minimal sets of each solver are presented in Tab. 1. The priors are defined based on each solver's stability,

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

Input: Transformation that aligns both Point Clouds
Data: Sets $\mathcal{L}$,Π, $\mathcal{Q}$, $\mathcal{M}$, $\mathcal{S}$, l_g, p_g, q_g, m_g, $\delta_{\mathcal{L}}$, δ_{Π}, $\delta_{\mathcal{Q}}$, $\delta_{\mathcal{M}}$, priors $p_p(g)$, and maximum # iterations ν

$\forall g \in \mathcal{G}$ Initialize $p(g) = 1$
while $\sum_g j_g < \nu$ **do**
  Sample a solver g with probability $p(g)p_p(g)$
  Increment j_g
  Sample l_g line intersections from $\mathcal{L}$
  Sample p_g plane correspondences from Π
  Sample q_g point correspondences from $\mathcal{Q}$
  Sample m_g line correspondences from $\mathcal{M}$
  Compute T using solver g
  Compute number of inliers $\iota(T) = \iota_{\mathcal{L}}(T, \delta_{\mathcal{L}}) + \iota_{\Pi}(T, \delta_{\Pi}) + \iota_{\mathcal{Q}}(T, \delta_{\mathcal{Q}}) + \iota_{\mathcal{M}}(T, \delta_{\mathcal{M}})$
  Compute $\epsilon_{\mathcal{L}}$, ϵ_{Π}, $\epsilon_{\mathcal{Q}}$, and $\epsilon_{\mathcal{M}}$
  if $\iota(T) > \iota(T^*)$ **then**
    $T^* = T$
    for each $g \in \mathcal{G}$ **do**
      Update $p(g)$ with (56)
      Update $\gamma(g)$ with (57)
    end for
  end if
  if $j_g > \gamma(g)$ **then**
    return T^*
  end if
end while

runtime, and the number of solutions obtained in Fig. 11 and Tab. 2. If a solver has a high prior or a small minimal set, it will be chosen more often. To prevent this and allow for exploration of other feature combinations, we penalize the probability of selecting a solver the more it has been selected. We define the probability of a solver g being selected as

$$p(g) = \left(\epsilon_{\mathcal{L}}^{l_g}\epsilon_{\Pi}^{p_g}\epsilon_{\mathcal{Q}}^{q_g}\epsilon_{\mathcal{M}}^{m_g}(1-\epsilon_{\mathcal{L}}^{l_g}\epsilon_{\Pi}^{p_g}\epsilon_{\mathcal{Q}}^{q_g}\epsilon_{\mathcal{M}}^{m_g})^{j_g-1}\right) p_p(g), \tag{56}$$

where $\epsilon_{\mathcal{L}}$, ϵ_{Π}, $\epsilon_{\mathcal{Q}}$, and $\epsilon_{\mathcal{M}}$ are the current inlier ratios of line intersections, and point, plane and line correspondences, respectively; l_g, p_g, q_g, and m_g are the minimal sets of each feature (presented in the same order as the inlier ratios); j_g is the number of RANSAC iterations where solver g was selected; and $p_p(g)$ is the solver prior.

The Hybrid RANSAC in each iteration selects a solver g with a sampling probability $p(g)$. Then, the minimal sets of each feature are chosen randomly. Finally, solver g is run. For each solution, the number of inliers of each feature is computed. For point correspondences, the metric used was the geometric distance in (3). A point correspondences is consider to be an inlier if $d_{\mathcal{Q}_i} < \delta_{\mathcal{Q}}$, with $\delta_{\mathcal{Q}}$ being the point inlier threshold. For planes, we used the geometric constraint in (4). If $d_{\Pi_i} < \delta_{\Pi}$, the correspondence is considered to be an inlier, with δ_{Π} being the plane inlier threshold. The metric for line correspondences is the geometric constraint in (5), and $\delta_{\mathcal{M}}$ is the line correspondence inlier threshold. Finally, for line intersections, the metric used was (2). The total score (number of inliers) of each solution is given by a weighted sum of the number of inliers of each feature. Each feature's weight is related to the amount of information it provides. Recall that a point and plane correspondences allow to fix three degrees of freedom each, a line correspondence restricts four, and a line intersection allows to solve for a single degree of freedom. If the model's score is higher than the current best, the model and the inlier ratios are updated.

The stopping condition is defined as the number of iterations we need to perform to obtain a good solution with probability p. This condition is defined for each solver since it depends on the minimal set and the inlier ratios. The maximum number iterations of each solver g is given as

$$\gamma(g) = \max\left(\frac{\log(1-p)}{\log(1-\epsilon_{\mathcal{L}}^{l_g}\epsilon_{\Pi}^{p_g}\epsilon_{\mathcal{Q}}^{q_g}\epsilon_{\mathcal{M}}^{m_g})}, \nu\right). \tag{57}$$

The probability p was set to 0.99 in the experimental results. Notice that, in the presence of a high percentage of outliers, the inlier ratios are low and thus can lead the maximum number of iterations per solver, i.e., $\gamma(g)$ to be high. To prevent the RANSAC loop from running for a long time, we introduce an upper bound to the maximum iterations of RANSAC denoted by ν. Similar to (56), the true inlier ratios are unknown. Thus the ratios used correspond to the current best model. When a new best model is found, the maximum number of iterations is also updated using (57). The Hybrid RANSAC is stopped as soon as a solver hits the maximum iterations to prevent higher runtimes.

Algorithm 2 AM algorithm for 3D registration refinement

Input: Initial guesses for R and $\mathbf{t}$
Data: Feature correspondences/intersections

$\mathbf{t}_0 \leftarrow$ `initial guess`; ▷ Sets an initial guess for the translation
$R_0 \leftarrow$ `initial guess`; ▷ Sets an initial guess for the rotation
$\delta \leftarrow 1$ ▷ Defines an initial value for the error
$k \leftarrow 1$ ▷ Variable identifying the iterations
$\tau \leftarrow$ `tol` ▷ Sets the limit tolerance
$k_{\max} \leftarrow$ `max_iter` ▷ Maximum # iterations
while $\delta > \tau$ and $k < k_{\max}$ **do**
  $R_k \leftarrow \operatorname{argmin}_{X \in \mathcal{SO}(3)} f(X, \mathbf{t}_{k-1})$ ▷ New rotation
  $\mathbf{t}_k \leftarrow \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^3} f(R_k, \mathbf{x})$ ▷ New translation
  $\delta = \| f(R_k, \mathbf{t}_k) - f(R_{k-1}, \mathbf{t}_{k-1}) \|$ ▷ Updates error
  $k = k + 1$ ▷ Adds one iteration
end while
$R = R_k$ and $\mathbf{t} = \mathbf{t}_k$ ▷ Sets the output estimation

Algorithm 3 Solver for the rotation matrix. Although $g(\mathbf{R})$ depends on the translation, this variable remains constant during the algorithm, so it is omitted. $\nabla g(\mathbf{R})$ stands for the calculated Euclidean gradient of the objective function.

Input: Initial guesses for R and $\mathbf{t}$
Data: Feature correspondences/intersections

$X_0 \in \mathcal{SO}(3) \leftarrow$ `initial guess`;
$\mu_1 \leftarrow 1$ ▷ Initial deviation angle in the $\mathcal{SO}(3)$
$\delta \leftarrow 1$ ▷ Sets an initial value for the error
$\tau \leftarrow$ `tol` ▷ Sets a limit for the tolerance
$k \leftarrow 0$ ▷ Initiates the number of iterations
while $\delta > \tau$ **do** ▷ Optimization cycle
  $Z_k \leftarrow \nabla g(X_k) X_k^T - X_k \nabla g(X_k)^T$ ▷ Gradient
  $z_k \leftarrow 0.5 \operatorname{trace}(Z_k Z_k^T)$ ▷ Rate of rotation step
  $P_k \leftarrow I + \sin(\mu_k) Z_k^T + (1 - \cos(\mu_k)) \left(Z_k^T\right)^2$ ▷ Step
  $Q_k \leftarrow P_k P_k$ ▷ Initial hypothesis
  while $g(X_k) - g(Q_k X_k) \geq \mu_k z_k$ **do** ▷ Updates
    $P_k \leftarrow Q_k$ ▷ Updates iterative step
    $Q_k \leftarrow P_k P_k$ ▷ Computes new hypothesis
    $\mu_k \leftarrow 2\mu_k$ ▷ Updates step
  end while
  while $g(X_k) - g(Q_k X_k) < 0.5 \mu_k z_k$ **do** ▷ Updates step
    $P_k \leftarrow I + \sin(\mu_k) Z_k^T + (1 - \cos(\mu_k)) \left(Z_k^T\right)^2$ ▷ Step
    $\mu_k \leftarrow 0.5 \mu_k$ ▷ Updates rotation angle
  end while
  $X_{k+1} \leftarrow P_k X_k$ ▷ Computes new estimate
  $\delta \leftarrow \| X_{k+1} - X_k \|_{\text{frob}}$ ▷ Sets new error
  $k \leftarrow k + 1$ ▷ Updates iterative counter
end while
$R \leftarrow X_k$ ▷ Returns best estimate

7 Alternating Minimization for Registration Refinement

Although it sometimes obtains a good enough estimate, RANSAC's main goal is not to estimate an accurate solution. The main goal of a RANSAC is to reject the feature outliers by finding the model which gathers the most consensus, i.e., with higher number of inliers, and get a good first guess for the pose estimation. Since the model is obtained with a small fraction of all inliers (minimal set), the model may not be the optimal one. In this section, we propose a refinement method for improving the solution obtained by the Hybrid RANSAC of Sec. 6. The refinement consists of minimizing a cost function applied to only the inliers found by the Hybrid RANSAC. The next subsections present an overview of the optimization framework, the cost functions and their respective gradients.

Algorithm 4 Solver for the translation vector. $h(\mathbf{t})$ and $\nabla h(\mathbf{t})$ represent the objective function and calculated gradient to the translation's elements

Input: Initial guesses for R and $\mathbf{t}$
Data: Feature correspondences/intersections

$\mathbf{x}_0 \in \mathbb{R}^3 \leftarrow$ `initial guess`
$\delta \leftarrow 1$ ▷ Initial value for the error
$\alpha \leftarrow$ `step` ▷ Chooses a step
$\tau \leftarrow$ `tol` ▷ Sets a limit for the tolerance
$k \leftarrow 0$ ▷ Initiates the number of iterations
while $\delta > \tau$ **do** ▷ Optimization cycle
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha \nabla h(\mathbf{x}_k)$ ▷ Updates the guess
  $\alpha = \frac{(\mathbf{x}_{k+1} - \mathbf{x}_k)^T (\nabla h(\mathbf{x}_{k+1}) - \nabla h(\mathbf{x}_k))}{\| \nabla h(\mathbf{x}_{k+1}) - \nabla h(\mathbf{x}_k) \|^2}$ ▷ Updates α
  if $h(\mathbf{x}_{k+1}) > h(\mathbf{x}_k)$ **then** ▷ Checks if value increased
    break ▷ If it is, stop the cycle
  end if
  $\delta \leftarrow \| h(\mathbf{x}_{k+1}) - h(\mathbf{x}_k) \|_{\text{frob}}$ ▷ Updates the error
  $k \leftarrow k + 1$ ▷ Updates the iterative counter
end while
$\mathbf{t} \leftarrow \mathbf{x}_k$; ▷ Returns the best estimate

7.1 Proposed refinement

The refinement scheme we exploit uses the POSEAMM framework proposed in the group's previous work, [88]. This framework allows solving pose estimation problems by alternating between two solvers, one for rotation and one for translation. The detailed algorithm is shown in Alg. 2. The main idea is to fix the translation at each iteration and solve for the rotation first, then fix the newly estimated rotation and solve for the translation. Formally, given an objective function $f(R, \mathbf{t})$, with the goal of finding R and $\mathbf{t}$ that minimizes $f(R, \mathbf{t})$, the k^{th} iteration will find R_k and $\mathbf{t}_k$ by solving the two following problems in a row

$$R_k = \underset{R \in \mathcal{SO}(3)}{\operatorname{argmin}} \; f(R, \mathbf{t}_{k-1}), \tag{58}$$

$$\mathbf{t}_k = \underset{\mathbf{t} \in \mathbb{R}^3}{\operatorname{argmin}} \; f(R_k, \mathbf{t}). \tag{59}$$

This process is repeated until the decrease in the cost function is smaller than a given tolerance or until a maximum number of iterations is reached. To have a general framework for pose problems, the only thing missing is to derive solvers for the problems in (58) and (59). In fact, as noted in [88], all pose estimators have some properties that we can use here. Namely, we always find a function that depends only on $g(R) = f(R, \mathbf{t}_{k-1})$, which makes (58) an optimization problem in the rotation matrix manifold, and $h(\mathbf{t}) = f(R_k, \mathbf{t})$, turning (59) a simple optimization problem in a $\mathbb{R}^3$ space. In [88], we derive algorithms of the steepest descent type for both problems in Alg. 2, namely Algs. 3 and 4. The POSEAMM framework is quite general and proved useful in many pose estimation problems. To exploit this framework in our line intersection constraints, we only need to define a cost function and its gradients with respect to the rotation and translation, i.e. $g(R)$, $h(\mathbf{t})$, $\nabla g(R)$, and $\nabla h(\mathbf{t})$. Notice that $\nabla g(R)$, and $\nabla h(\mathbf{t})$ are a matrix and a vector, respectively. Below, we define these functions and respective gradients for each considered type of feature.

7.2 Cost functions

We start our derivations by getting the cost functions for each feature type, point, line, plane match, and line intersection. To simplify the calculations, i.e., avoid going through all the data at each iteration, we fix all the coefficients that only depend on the data and model the problem with matrix polynomial equations with rotation and translation parameters as unknowns. This turned out to be a simple process, with high computational efficiency gains given the nature of the Algs. 3 and 4.

The full cost function in the alternating minimization is

$$f(R, \mathbf{t}) = \eta_1 f_{\mathcal{Q}}(R, \mathbf{t}) + \eta_2 f_{\Pi}(R, \mathbf{t}) + \eta_3 f_{\mathcal{M}}(R, \mathbf{t}) + \eta_4 f_{\mathcal{L}}(R, \mathbf{t}), \quad (60)$$

where η_i with $i = 1, 2, 3, 4$ are the weights of each cost function associated with each feature type. Subscripts $\mathcal{Q}$, Π, $\mathcal{M}$, and $\mathcal{L}$ are used to distinguish the objective functions for point, plane, and line matches, and line intersections respectively.

Defining objective functions $g(R)$ and $h(\mathbf{t})$ from $f(R, \mathbf{t})$ by fixing $\mathbf{t}$ and R respectively, the gradients with respect to the rotation matrix and the translation vector are

$$\nabla g(R) = \eta_1 \nabla g_{\mathcal{Q}}(R) + \eta_2 \nabla g_{\Pi}(R) + \eta_3 g_{\mathcal{M}}(R) + \eta_4 g_{\mathcal{L}}(R), \quad (61)$$

and

$$\nabla h(\mathbf{t}) = \eta_1 \nabla h_{\mathcal{Q}}(\mathbf{t}) + \eta_2 \nabla h_{\Pi}(\mathbf{t}) + \eta_3 h_{\mathcal{M}}(\mathbf{t}) + \eta_4 h_{\mathcal{L}}(\mathbf{t}). \quad (62)$$

Next, we present the derivation of each cost functions and their respective gradients with respect to the rotation matrix and the translation vector.

Point matches:

The cost function used for point correspondences is the square of (3), summed for all inlier point correspondences, which is obtained as

$$f_{\mathcal{Q}}(R, \mathbf{t}) = \sum_{i=1}^{n_{\mathcal{Q}}} \|R\mathbf{q}_i + \mathbf{t} - \mathbf{q}'_i\|^2, \quad (63)$$

where $n_{\mathcal{Q}}$ is the number of elements in $\mathcal{Q}$. For simplicity, we use $\mathcal{Q}$ here also to denote the set of inlier point correspondences. By expanding (63) and writing it in matrix form, we have

$$f_{\mathcal{Q}}(R, \mathbf{t}) = \mathbf{r}^T M_{\mathcal{Q},1} \mathbf{r} + n_{\mathcal{Q}} \mathbf{t}^T \mathbf{t} + 2\mathbf{t}^T M_{\mathcal{Q},2} \mathbf{r} - 2\mathbf{m}_{\mathcal{Q},3}^T \mathbf{r} - 2\mathbf{m}_{\mathcal{Q},4}^T \mathbf{t} + m_{\mathcal{Q},5}, \quad (64)$$

where

$$M_{\mathcal{Q},1} = \sum_{i=1}^{n_{\mathcal{Q}}} \left(\mathbf{q}_i^T \otimes I_3\right)^T \left(\mathbf{q}_i^T \otimes I_3\right) \quad (65)$$

$$M_{\mathcal{Q},2} = \sum_{i=1}^{Q} \mathbf{q}_i^T \otimes I_3 \quad (66)$$

$$\mathbf{m}_{\mathcal{Q},3} = \sum_{i=1}^{n_{\mathcal{Q}}} \mathbf{q}_i \otimes \mathbf{q}'_i \quad (67)$$

$$\mathbf{m}_{\mathcal{Q},4} = \sum_{i=1}^{n_{\mathcal{Q}}} \mathbf{q}'_i \quad (68)$$

$$m_{\mathcal{Q},5} = \sum_{i=1}^{n_{\mathcal{Q}}} {\mathbf{q}'_i}^T \mathbf{q}'_i. \quad (69)$$

and $\mathbf{r} = \text{vec(R)}$. Objective functions $g_{\mathcal{Q}}(R)$ and $h_{\mathcal{Q}}(\mathbf{t})$ are derived directly from $f_{\mathcal{Q}}(R, \mathbf{t})$.

Now computing the gradients of (64) with respect to R and $\mathbf{t}$, using matrix computations, we obtain

$$\nabla g_{\mathcal{Q}}(R) = 2M_{\mathcal{Q},1}\mathbf{r} + 2M_{\mathcal{Q},2}^T\mathbf{t} - 2\mathbf{m}_{\mathcal{Q},3}, \quad (70)$$

and

$$\nabla h_{\mathcal{Q}}(\mathbf{t}) = 2\mathbf{t} + 2M_{\mathcal{Q},2}\mathbf{r} - 2\mathbf{m}_{\mathcal{Q},4}, \quad (71)$$

respectively.

Plane correspondences:

For plane correspondences the cost function used was the square of (4), summed for all inlier planes set (here denoted as Π), it can be written as

$$f_{\Pi}(R,\mathbf{t}) = \sum_{i=1}^{n_{\Pi}} \left[\frac{1}{3} \sum_{j=1}^{3} \|(R\mathbf{q}_j + \mathbf{t})^T \overline{\boldsymbol{\pi}}_i' + \widetilde{\pi}_i'\|^2 \right], \quad (72)$$

where n_{Π} is the number of elements of Π. Again expanding and writing this equation matrix form, we obtain

$$f_{\Pi}(R,\mathbf{t}) = \mathbf{r}^T M_{\Pi,1}\mathbf{r} + \mathbf{t}^T M_{\Pi,2}\mathbf{t} + \frac{2}{3}\mathbf{t}^T M_{\Pi,3}\mathbf{r} + \frac{2}{3}\mathbf{m}_{\Pi,4}^T\mathbf{r} + 2\mathbf{m}_{\Pi,5}^T\mathbf{t} + m_{\Pi,6}, \quad (73)$$

where

$$M_{\Pi,1} = \sum_{i=1}^{n_{\Pi}} (\mathbf{q}_1^T \otimes \overline{\boldsymbol{\pi}}_i'^T)^T (\mathbf{q}_1^T \otimes \overline{\boldsymbol{\pi}}_i'^T) + (\mathbf{q}_2^T \otimes \overline{\boldsymbol{\pi}}_i'^T)^T (\mathbf{q}_2^T \otimes \overline{\boldsymbol{\pi}}_i'^T) + (\mathbf{q}_3^T \otimes \overline{\boldsymbol{\pi}}_i'^T)^T (\mathbf{q}_3^T \otimes \overline{\boldsymbol{\pi}}_i'^T) \quad (74)$$

$$M_{\Pi,2} = \sum_{i=1}^{n_{\Pi}} \overline{\boldsymbol{\pi}}_i' \overline{\boldsymbol{\pi}}_i'^T \quad (75)$$

$$M_{\Pi,3} = \sum_{i=1}^{n_{\Pi}} \left(\sum_{j=1}^{3} \mathbf{q}_j \right)^T \otimes (\overline{\boldsymbol{\pi}}_i' \overline{\boldsymbol{\pi}}_i'^T) \quad (76)$$

$$\mathbf{m}_{\Pi,4} = \sum_{i=1}^{n_{\Pi}} \left(\sum_{j=1}^{3} \mathbf{q}_j \right) \otimes (\widetilde{\pi}_i' \overline{\boldsymbol{\pi}}_i') \quad (77)$$

$$\mathbf{m}_{\Pi,5} = \sum_{i=1}^{n_{\Pi}} \widetilde{\pi}_i' \overline{\boldsymbol{\pi}}_i' \quad (78)$$

$$m_{\Pi,6} = \sum_{i=1}^{n_{\Pi}} \widetilde{\pi}_i' \widetilde{\pi}_i'. \quad (79)$$

Now, taking the matrix derivations, the gradients $\nabla g_{\Pi}(R)$ and $\nabla g_{\Pi}(\mathbf{t})$ are given by

$$\nabla g_{\Pi}(R) = 2M_{\Pi,1}\mathbf{r} + \frac{2}{3}M_{\Pi,3}\mathbf{t} + \frac{2}{3}\mathbf{m}_{\Pi,4}, \quad (80)$$

and

$$\nabla h_{\Pi}(\mathbf{t}) = 2M_{\Pi,2}\mathbf{t} + \frac{2}{3}M_{\Pi,3}\mathbf{r} + 2\mathbf{m}_{\Pi,5}. \quad (81)$$

Line matches:

The cost function of line correspondences is defined as the mean of the distance of two points of the line in the first frame to the corresponding line in the second (target) frame. See (5). The function is given as

$$f_{\mathcal{M}}(R,\mathbf{t}) = \frac{1}{2}\sum_{i=0}^{n_{\mathcal{M}}} \|(\mathbf{q}_{\mathbf{l}_i}' - R\mathbf{q}_{1,\mathbf{l}_i} + \mathbf{t}) \times \bar{\mathbf{l}}_i'\|^2 + \|(\mathbf{q}_{\mathbf{l}_i}' - R\mathbf{q}_{2,\mathbf{l}_i} + \mathbf{t}) \times \bar{\mathbf{l}}_i'\|^2, \quad (82)$$

where $\mathbf{q}_{\mathbf{l}_i}'$ is a point in the corresponding line in the target frame, which can be obtained from the Plücker coordinates with (16), $\|\bar{\mathbf{l}}_i\| = 1$, and $n_{\mathcal{M}}$ is the number of elements in $\mathcal{M}$. Expanding this equation and writing it in matrix form yields

$$f_{\mathcal{M}}(R,\mathbf{t}) = \mathbf{r}^T M_{\mathcal{M},1}\mathbf{r} + \mathbf{t}^T M_{\mathcal{M},2}\mathbf{t} + \mathbf{t}^T M_{\mathcal{M},3}\mathbf{r} + \mathbf{m}_{\mathcal{M},4}^T\mathbf{r} + 2\mathbf{m}_{\mathcal{M},5}^T\mathbf{t} + m_{\mathcal{M},6}, \quad (83)$$

where

$$M_{\mathcal{M},1} = \sum_{i=0}^{n_{\mathcal{M}}} (\mathbf{q}_{1,\mathbf{l}_i}^T \otimes [\bar{\mathbf{l}}_i]_{\mathrm{x}})^T (\mathbf{q}_{1,\mathbf{l}_i}^T \otimes [\bar{\mathbf{l}}_i]_{\mathrm{x}}) + (\mathbf{q}_{2,\mathbf{l}_i}^T \otimes [\bar{\mathbf{l}}_i]_{\mathrm{x}})^T (\mathbf{q}_{2,\mathbf{l}_i}^T \otimes [\bar{\mathbf{l}}_i]_{\mathrm{x}}) \quad (84)$$

$$M_{\mathcal{M},2} = \sum_{i=0}^{n_{\mathcal{M}}} [\bar{\mathbf{l}}_i]_{\mathrm{x}} [\bar{\mathbf{l}}_i]_{\mathrm{x}} \quad (85)$$

$$M_{\mathcal{M},3} = \sum_{i=0}^{n_{\mathcal{M}}} (\mathbf{q}_{1,\mathbf{l}_i} + \mathbf{q}_{2,\mathbf{l}_i})^T \otimes [\bar{\mathbf{l}}_i]_{\mathrm{x}} [\bar{\mathbf{l}}_i]_{\mathrm{x}} \quad (86)$$

$$\mathbf{m}_{\mathcal{M},4} = \sum_{i=0}^{n_{\mathcal{M}}} (\mathbf{q}_{1,\mathbf{l}_i} + \mathbf{q}_{2,\mathbf{l}_i}) \otimes \left(\mathbf{q}_{\mathbf{l}_i}'^T [\bar{\mathbf{l}}_i]_{\mathrm{x}} [\bar{\mathbf{l}}_i]_{\mathrm{x}} \right) \quad (87)$$

$$\mathbf{m}_{\mathcal{M},5} = \sum_{i=0}^{n_{\mathcal{M}}} [\bar{\mathbf{l}}_i]_{\mathrm{x}} [\bar{\mathbf{l}}_i]_{\mathrm{x}} \mathbf{q}_{\mathbf{l}_i}' \quad (88)$$

$$m_{\mathcal{M},6} = \sum_{i=0}^{n_{\mathcal{M}}} \mathbf{q}_{\mathbf{l}_i}^{\prime T} [\bar{\mathbf{l}}_i]_{\times} [\bar{\mathbf{l}}_i]_{\times} \mathbf{q}_{\mathbf{l}_i}^{\prime}. \quad (89)$$

Again, by considering matrix calculus, we get the gradients by derivation of (83) as

$$\nabla g_{\mathcal{M}}(R) = 2M_{\mathcal{M},1}\mathbf{r} + M_{\mathcal{M},3}\mathbf{t} + \mathbf{m}_{\mathcal{M},4}, \quad (90)$$

and

$$\nabla h_{\mathcal{M}}(\mathbf{t}) = M_{\mathcal{M},3}\mathbf{r} + 2M_{\mathcal{M},2}\mathbf{t} + 2\mathbf{m}_{\mathcal{M},5}. \quad (91)$$

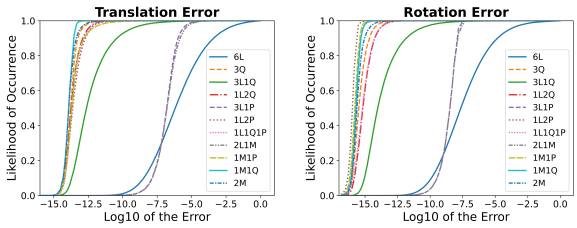

Figure 11: Solvers' Numerical Analysis: Cumulative density plots of translation and rotation errors of the solvers presented in this paper, compared to the 6L and 3Q.

Line intersections:

The geometric distance between two intersecting lines cannot be computed trivially like the distances in the previous subsections. Then, in this case, we use an algebraic distance to make sure our solver is fast. The cost function for line intersections used is the one presented in [88] for the Generalized Relative Pose Problem, with derivations in the paper's supplementary materials.

8 Experiments

This section presents the results of our method. Section 8.1 gives the numerical stability of each solver in Tab. 1, namely the solvers' runtime, number of solutions, and numerical stability. The entire pipeline, i.e., the Hybrid RANSAC using different combinations of the minimal solvers, followed by the our refinement technique is evaluated in two real-world datasets, in Sec. 8.2. For both experiments we consider the following error metrics.

Error metrics:

Rotation and translation errors, $e_R(R)$ and $e_{\mathbf{t}}(\mathbf{t})$, respectively, are set with the following metrics

$$e_R(R) = \text{acos}\left(\frac{\text{trace}\left(R^{-1}R_{\text{GT}}\right)-1}{2}\right) \quad (92)$$

$$e_{\mathbf{t}}(\mathbf{t}) = \|\mathbf{t} - \mathbf{t}_{\text{GT}}\|, \quad (93)$$

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

8.1 Minimal solvers stability

Being one of the paper's main contributions, we start the experiments with the stability analysis of minimal solvers. The solvers in Tab. 1 were implemented in C++. Each was evaluated 10^6 times on synthetic data without noise, and the runtine, rotation error and translation error are computed. For solvers with multiple solutions only the correct one was evaluated. In these conditions, the solver stability corresponds to the ability of the solver to yield the correct solution, given different sets of noiseless data. Furthermore, noiseless data allows to evaluate the solvers' runtime and the number of real solutions. Both the solver runtime and its number of solutions affect the total runtime of the Hybrid RANSAC in Sec. 6, since each of them must be evaluated multiple times for computing the inliers, which is a time-consuming step, especially for large amounts of data.

Data generation:

First, a rigid transformation (rotation and translation) is selected randomly. The rotation is obtained by sampling three Euler angles and converting them into matrix notation. The translation was obtained by sampling each axis from -10 to 10 units. The features are created from points sampled from a 40 unit side cube. Point correspondences are obtained directly. Lines and planes are obtained from two and three points, respectively.

Numerical analysis:

The rotation and translation errors of all solvers in Tab. 1 are presented in Fig. 11. The plots show the cumulative density of the errors, i.e., the area under the curve gives us the probability of the solver yielding a solution with an error smaller than some value. For example, the probability of solver $2M$ yielding an error under 10^{-2} is approximately one. As far as stability is concerned, the

Solver	Runtime [μs]		#Solutions	
	Mean	**Median**	**Mean**	**Median**
6L[21]	2757.8	2747.5	20.4	20
3Q[20]	0.72	0.71	2	2
3L1Q	26.83	24.56	5.46	6
1L2Q	1.20	1.20	2	2
3L1P	5.07	5.14	3.10	4
1L2P	0.99	0.99	1	1
1L1Q1P	1.45	1.45	2	2
2L1M	2.24	2.27	3.53	4
1M1P	1.54	1.53	1	1
1M1Q	1.29	1.29	1	1
2M	1.64	1.63	1	1

Table 2: **Solvers' Runtime:** Mean and median runtime in microseconds, and number of real valued solutions for each solver in Tab. 1 for 10^6 runs.

faster the curve reaches one, the more stable the solver is. For both errors, the same behavior is observed. The least stable solver is 6L, which is expected since it requires solving high degree polynomial equations[7]. It is followed by the solvers that use Ferrari's formula (3L1P and 2L1M). The solver 3L1Q is the fourth-best in terms of stability. Even though it has up to eight solutions – while 3L1P and 2L1M have four – it is more stable due to the efficient solver used to solve the eight-degree polynomial. The remaining solvers present similar performance.

Runtimes:

Table 2 shows the mean and median for the runtimes and number of real valued solutions of each solver. As expected, the solvers with higher runtimes are the ones that yield more solutions, particularly the solvers 6L and 3L1Q, because they are associated with the higher degree polynomial equations that need to be solved. The remaining solvers output four or fewer solutions and can be computed in closed-form.

[7]The solver was derived using the Gröbner basis, which requires performing Gauss-Jordan elimination on big matrices as shown in [21]. This process can be numerically unstable, especially if the matrices are ill-conditioned [95].

8.2 Results on real data

We use two RGB-D datasets, SUN3D [23], and TUM RGB-D [10]. For the methods tests, we consider the following baselines and variations of our two-step approach.

Methods:

We use the following baselines for comparison:

- ICP: Iterative Closest Point; [25];
- GR: Global Registration [34];
- FGR: Fast Global Registration [37];
- TEASER++: Fast and Certifiable Point Cloud Registration [96]; and
- DGR: Deep Global Registration [70].

For the method proposed in this work, we considered several alternatives. Namely, we consider RANSAC with the following solvers:

- RANSAC 6L: six intersecting lines;
- RANSAC 3Q: three corresponding points;
- RANSAC Corrs: feature matches only;
- RANSAC Ours: solvers in Sec. 5; and
- RANSAC All: all solvers in Tab. 1.

All the above-listed methods are tested with and without the refinement technique proposed in Sec. 7. We include Ref after the methods listed above to tell that the refinement followed it. For example, the method RANSAC All Ref denotes the RANSAC All followed by the refinement proposed in Sec. 7. Notice that, only the constraints associated with point and line matches were used in the refinement step.

We implemented the RANSAC loop in C++. Regarding the AM refinement, we use the code in [88], also implemented in C++. For the baselines, the available C++ implementations were used: ICP, GR, and FGR from Open3D [97]. For TEASER++, the C++ implementation released by the authors[8] was used. For DGR, the python code[9] and weights[10] provided by the authors were used. We tested the Super4PCS from OpenGR in [98], but the results were significantly worse than GR and FGR, as shown in [90]. We decided to omit these from the tables below.

[8]https://github.com/MIT-SPARK/TEASER-plusplus
[9]https://github.com/chrischoy/DeepGlobalRegistration
[10]We use the version trained on 3DMatch

Method	SUN Sequences [23]								TUM Sequences [10]							
	#1	#2	#3	#4	#5	#6	#7	Avg	#1	#2	#3	#4	#5	#6	#7	Avg
Rotation Errors [deg]: Baselines																
ICP[25]	0.95	3.35	4.11	2.73	4.41	2.10	1.20	2.69	8.22	1.17	1.74	5.71	3.40	0.75	7.24	4.03
GR[34]	1.20	1.44	1.92	1.54	1.14	1.14	1.45	1.40	2.55	1.36	2.01	3.35	3.01	1.14	2.92	2.33
FGR[37]	1.05	1.81	1.26	1.74	1.64	1.16	1.32	1.43	2.03	1.03	1.63	2.84	2.77	0.70	4.31	2.19
TEASER++[96]	1.60	2.26	2.07	2.84	2.21	2.13	2.05	2.17	2.22	1.14	1.64	2.23	3.40	0.88	5.47	2.43
DGR[70]	—	—	—	—	—	–	–	—	1.86	0.88	1.43	2.18	2.91	0.66	**1.78**	1.67
Rotation Errors [deg]: Ours without Refinement																
RANSAC 6L	0.79	1.17	1.46	1.48	1.30	1.18	0.86	1.18	1.78	1.09	1.54	2.09	2.83	0.84	1.97	1.73
RANSAC 3Q	1.10	1.76	1.97	1.68	1.85	1.63	1.24	1.60	2.34	1.41	1.90	2.89	4.36	1.10	3.47	2.50
RANSAC Corrs	1.13	1.81	2.05	1.65	1.75	1.64	1.15	1.60	2.50	1.46	1.84	3.38	3.25	1.06	3.96	2.49
RANSAC Ours	0.92	1.36	1.64	1.33	1.29	1.34	1.08	1.28	1.83	1.14	1.63	2.38	2.59	0.84	2.22	1.80
RANSAC All	0.92	1.34	1.61	1.31	1.53	1.20	0.94	1.26	1.84	1.10	1.46	2.29	2.90	0.91	2.03	1.79
Rotation Errors [deg]: Ours with Refinement																
RANSAC 6L Ref	**0.54**	**1.00**	**1.01**	1.21	**1.07**	**1.03**	**0.81**	**0.95**	**1.40**	**0.62**	1.15	**1.70**	**2.09**	**0.54**	**1.62**	**1.30**
RANSAC 3Q Ref	0.63	1.53	1.23	1.37	1.61	1.33	0.96	1.24	1.59	0.65	1.36	2.28	3.25	0.64	3.61	1.91
RANSAC Corrs Ref	0.64	1.61	1.28	1.29	1.49	1.37	0.95	1.23	1.77	0.68	1.39	2.25	3.11	0.67	3.02	1.84
RANSAC Ours Ref	**0.57**	**1.01**	1.11	**1.02**	**1.02**	**1.06**	**0.77**	**0.94**	1.46	**0.63**	**1.03**	1.85	2.24	**0.61**	2.00	1.40
RANSAC All Ref	0.59	1.08	**1.09**	**1.06**	1.24	1.10	0.82	1.00	**1.39**	0.66	**0.96**	**1.80**	**1.95**	0.64	1.89	**1.33**
Translation Errors [cm]: Baselines																
ICP[25]	4.14	15.45	10.0	19.6	10.7	11.5	7.63	11.29	13.79	3.19	4.96	11.52	8.00	**1.46**	18.63	8.79
GR[34]	4.79	6.73	6.15	7.87	6.45	5.93	5.92	6.26	5.51	3.78	3.77	5.18	6.84	3.39	13.33	5.97
FGR[37]	3.15	6.95	3.51	9.75	6.87	4.78	4.51	5.65	3.60	2.41	2.23	4.34	5.19	**1.43**	14.62	4.83
TEASER++[96]	4.63	9.41	4.70	17.70	9.02	7.07	7.01	8.51	3.28	2.99	1.84	3.15	6.22	1.95	18.97	5.49
DGR[70]	—	—	—	—	—	—	—	—	**3.15**	2.05	2.12	3.69	6.02	1.48	10.19	4.10
Translation Errors [cm]: Ours without Refinement																
RANSAC 6L	3.13	**5.17**	3.42	7.16	6.49	4.87	**3.38**	4.80	3.47	2.53	2.39	3.25	6.87	3.07	8.88	4.35
RANSAC 3Q	3.62	7.52	4.83	7.69	8.14	6.35	5.20	6.19	4.71	3.37	3.21	4.99	8.41	3.61	12.16	5.78
RANSAC Corrs	3.66	7.57	4.70	7.60	8.31	6.20	5.04	5.24	4.61	3.42	3.31	4.64	7.59	3.59	14.51	5.95
RANSAC Ours	3.38	6.27	4.28	6.46	7.58	6.34	4.49	5.54	3.67	2.66	2.13	3.19	6.41	2.58	9.24	4.27
RANSAC All	3.24	5.81	3.96	6.33	8.55	5.53	4.03	5.35	3.85	2.65	2.07	**3.00**	6.74	2.78	8.47	4.22
Translation Errors [cm]: Our with Refinement																
RANSAC 6L Ref	**2.16**	**5.39**	**2.69**	6.27	**5.49**	**4.31**	**3.20**	**4.24**	**3.30**	**1.42**	2.14	3.25	5.57	1.87	**8.11**	**3.67**
RANSAC 3Q Ref	2.46	7.41	3.13	6.92	7.58	5.28	4.09	5.27	3.62	**1.49**	2.44	3.56	7.33	2.04	11.91	4.63
RANSAC Corrs Ref	2.40	7.30	3.25	6.71	7.17	5.73	4.10	5.24	**3.15**	1.67	2.92	4.07	7.58	2.17	14.26	5.12
RANSAC Ours Ref	**2.23**	5.93	3.03	**5.89**	**6.16**	4.63	3.70	4.51	3.31	1.52	**1.70**	3.04	**5.34**	2.03	9.25	3.74
RANSAC All Ref	2.31	5.62	**2.88**	**5.90**	6.84	**4.47**	3.41	**4.49**	**3.15**	**1.42**	**1.64**	**2.60**	**4.55**	2.02	**7.70**	**3.30**

Table 3: **Results:** Median rotation and translation errors of our method against the baselines, with and without refinement. Two datasets are used, the SUN3D and TUM RGB-D. Sequences #1 to #7 denote the `mit_32`, `brown_cogsci`, `hotel_nips`, `brown_cs_3`, `harvard_c5`, `mit_76_studyroom`, and `mit_lab_h` sequences in the SUN3D data-set, respectively. TUM sequences #1 to #7 denote the `fr1_room`, `fr2_desk`, `fr1_xyz`, `fr1_desk`, `fr3_cabinet`, `fr3_sitting`, and `fr2_pioneer`, respectively. **Purple** and **teal** colors indicate the best and second best results in both the translation and rotation errors. Sequences in the training set of DGR were not tested and are indicated by —.

Setup:

To obtain the features required by our solvers in Tab. 1, we exploit the RGB image to get line correspondences. The method used is based on Line-Junction-Line Structure Descriptor and Local Homography Estimation presented in [99, 100]. For each pair of correspondences, we compute their respective 3D counterparts in the respective camera frame. Then, we check for line intersections in each frame. Let us consider two lines in the first frame, if they intersect, i.e., their distance is smaller than a threshold (7[mm]), and their correspondences in the second frame also

intersect, we define a correspondence of intersecting lines. Notice that, our method does not require line intersection points to be present in either of the point-clouds. Nevertheless, we search for corresponding line intersections, since from those we can compute all the other features. After obtaining the line intersection correspondences, we can obtain point matches by computing the intersection point of the lines in both frames. Since the lines do not intersect exactly, we use the middle point of the closest points of each line to the other. Finally, plane correspondences are obtained by fitting a plane to the 3D points in each line.

Both datasets were acquired with a 30 frame-per-second RGB-D camera, with small displacements between consecutive frames, which, as observed in practice, leads to the identity matrix providing smaller errors than the outputs of each method. Therefore, comparing the different approaches using such a small baseline will not assess the benefits of using one over the other. To prevent configurations in which the identity matrix provides a better solution than the estimated ones, we assume pairs with 10 frames apart. The Hybrid RANSAC parameters for both datasets were $\delta_{\mathcal{L}} = 9.6 \times 10^{-5}$, $\delta_{\Pi} = 4.5 \times 10^{-4}$, $\delta_{\mathcal{Q}} = 1.3 \times 10^{-3}$, $\delta_{\mathcal{M}} = 4.2 \times 10^{-4}$. These parameters were obtained by running the tool presented in [101] on an additional sequence, which was not included in Tab. 3. The PoseAMM parameters used were $\alpha = 0.05$, $\tau = 1 \times 10^{-9}$, $\eta_1 = 0.43$, $\eta_2 = 0$, $\eta_3 = 0.57$, and $\eta_4 = 0$ (see Alg. 4 and (60)).

Results:

For the results, we compare the different versions of our method with the baselines using 14 complete sequences of the two real-world datasets, SUN3D and TUM RGB-D. We compute the median of the pairwise rotation and translation errors for all the method and sequences. Table 3 shows these results.

Another relevant evaluation criterion for assessing the merits of 3D registration algorithms is the computational time. Then, we take pairs of scans in three sequences of each dataset used. In addition to the accuracy, we compute the computational time obtained for estimating the transformations that align the scans. Results are shown in Fig. 12. Notice that, these computational times correspond to running the method used in Tab. 3, i.e., no change in the hyperparameters was done.

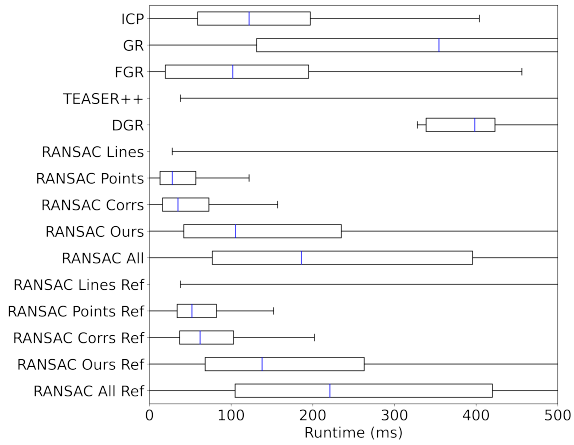

Figure 12: Run time of each method in Tab. 3 in milliseconds. The runtimes considered were from the `fr1_room`, `fr2_desk` and `fr1_xyz` sequences of the TUM RGB-D data-set, and the `mit_32`, `brown_cogsci` and `mit_76_studyroom` from the SUN3D data-set. Given the high runtime of the RANSAC 6L, RANSAC 6L Ref and TEASER++, the median is not displayed in the figure. The median runtime of those methods were 1653.5, 1677.5 and 4481.5 [ms], respectively.

To conclude these results, we show some validation plots. We take two sequences of each dataset and run the algorithms for 30 scans. All three baselines were used. Concerning our two-step method, given limitations in space and from Tab. 3 and Fig. 12, we only utilize alternatives RANSAC Ours and the RANSAC Ours Ref (i.e., with and without refinement). Figure 13 shows the final 3D registration results compared to the ground truth. Specifically, we show a single point cloud obtained by transforming all the 30 scans to the frame of the first one using the estimated transformations. Since 3D scans are starting to be the primary perception sensor for agent localization, we also want to evaluate the obtained trajectory of our method against the ground truth and the baselines ones. Therefore, in Fig. 14 we compare the obtained 3D trajectories of each of the methods for sequences of 60 scans. The point clouds shown are obtained from the ground truth pairwise transformations.

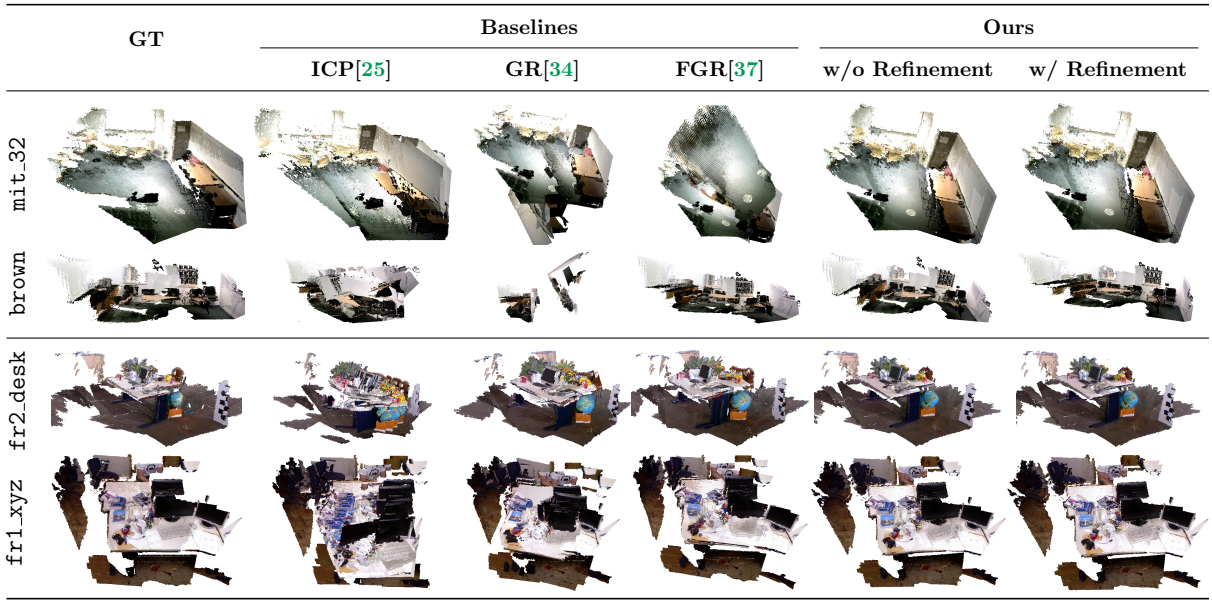

Figure 13: Results of the registration of 30 point clouds. We use `mit_32`, `brown_cogsci` sequences of the SUN3D dataset and `fr2_desk`, `fr1_xyz` sequences of the TUM data-set [10]. Recall that each point cloud is sampled sample at 3 frames per second, i.e., each pair is 10 frames apart. The results consist of applying the methods directly without transformation averaging.

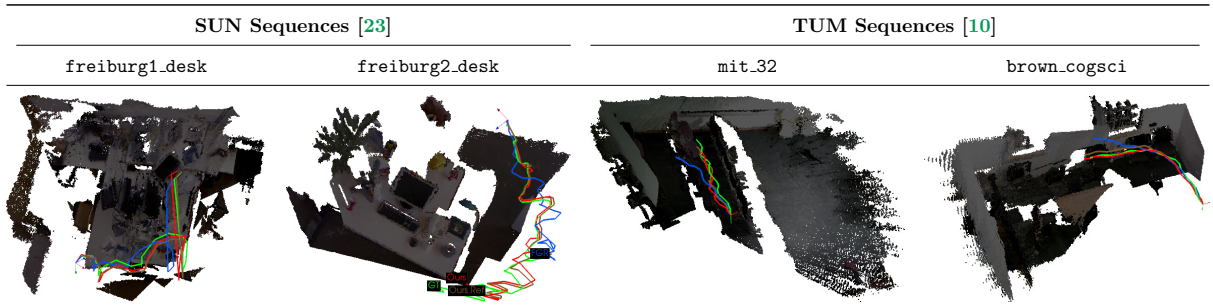

Figure 14: Lines describing the estimated trajectories for the camera motion from RANSAC Ours, RASNAC Ours Ref, FGR. The ground-truth for comparison is shown green. We are plotting a sequence of up to 60 camera positions estimates; the 3D registration of the point cloud was obtained with the ground-truth localization. Sometimes, when the translation is small, FGR fails in estimating the pose. This happens a couple of times in each scan. We replaced these failing estimates with the ground-truth ones to have a continuous trajectory. Notice again that we are only showing results by integrating pairwise estimates.

Discussion:

We end these experiments by discussing the results. We start with accuracy shown in Tab. 3 and with the motivation for the use of line intersection constraints. Comparing the RANSAC 6L approach to the baselines, we see that the proposed method has lower rotation and translation errors. Although it is close, there are only two sequences in which FGR outperforms RANSAC 6L. The average performance in the TUM RGB-D dataset is also better than DGR's. Notice

that our method is motivated/requires a predominantly manufactured environment that may not always be the case. On the other hand, if we look at the run times in Fig. 12, we can conclude that this RANSAC 6L approach is computational deficient, limiting its applications to real-time scenarios. With this, we can answer one of the paper's research questions. Line intersection constraints help get a more accurate 3D registration in a human-made environment. See research question number 1.

Although the 6L solver is able to get good results, the fact that it is slow limits the applications and motivates the second researcher questions; the other predominant constraints in 3D line intersections and how can they be useful in 3D registration. See research question number 2. From Tab. 3, we see that if we only take the solvers derived in this paper that combine line intersections with plane, point, and line matches, the accuracy is still very competitive to RANSAC 6L. In some sequences, RANSAC Ours outperforms RANSAC 6L. RANSAC Ours is better than the baselines in six out of the 14 sequences and better than RANSAC 6L in six sequences. The main gain is, however, computational efficiency. Figure 12 shows that RANSAC Ours reduces the computation time significantly compared to RANSAC 6L, it is slightly slower than FGR, but significantly faster than DGR. It can therefore meet real-time requirements of applications such as SLAM and SfM[11]. Other combinations of solvers were considered in the tests, showing additional advantages of using different solvers in RANSAC concerning accuracy vs. computational time.

Our two-step method is a coarse and fine estimation approach. While the coarse estimate already provides good results, from Tab. 3 we see that our AM refinement approach, in general, is able to improve the accuracy significantly. For example, RANSAC Ours Ref is outperformed by the baselines in only two of the 14 sequences. More critical than that[12], when our two-step approach using line intersections is better than the baselines, the refinement can improve the results significantly with respect to the coarse approach obtained from RANSAC. For example, when compared to the baselines, we have improvements of around 48% in rotation and 31% in translation for the TUM sequence #7; 42% in rotation and 24% in translation in SUN sequence #1. When looking to Fig. 12, we see that the proposed refinement technique has a minor impact on the computation time. The conclusions outlined for our strategy without refinement are kept for our approach with refinement. For example, RANSAC Ours Ref is only slightly slower than FGR, and about three times faster than DGR.

With respect to visualization, from Fig. 13 we can see that FGR produces good 3D registration results. However, in all the cases shown, our strategy performs even better in the details, both with and without refinement. Note, for example, the teddy bear head in `fr2_desk` of our solution compared to FGR. Figure 14 shows that the trajectories obtained from our methods obtained the best trajectories and, therefore, the best 3D registration.

9 Conclusion

This paper aims to assess the use of 3D line intersection constraints in 3D registration. This is addressed utilizing a two-step technique. First, we run a robust estimator, namely RANSAC. For that purpose, we proposed nine novel minimal solvers with a set of line intersection constraints and developed a hybrid RANSAC that can use all the solvers for that purpose. The second step of our method runs a non-linear optimization solver that takes all the inliers from step one and refines the solution. For this, we derive an alternative minimization method that alternates between finding the rotation and then the translation at each iteration step. Compared to our previous work, this paper introduces four new minimal solvers (out of nine total) that exploit line matches. Furthermore, this work replaced the non-linear refinement, which consisted of using the Levenberg-Marquardt, by the alternate minimization method described above.

Results with real data show that line intersections improve the 3D registration accuracy. Even before running the refinement, in general, our coarse estimate from RANSAC already performs on par with the baselines. On the other hand, the improvements made by the non-linear

[11] We note that comparisons in this paper are focusing only on the alignment, not in the feature extraction and matching

[12] Although we never lose much when compared to the baselines, we know that not every sequence have the perfect conditions for the line intersection constraints.

refinement are significant, proving that the use of line intersection constraints improves the accuracy of the registration in human-made scenarios significantly.

For the future, to achieve a complete SLAM/SfM pipeline, we envision the incorporation of motion averaging and loop closure techniques that will minimize the accumulated drift by using multiple 3D scans.

Acknowledgements

A. Mateus was partially supported by the Portuguese National Funding Agency for Science, Research and Technology (FCT) grant `PD/BD/135015/2017` and project `LARSyS - FCT Plurianual funding 2020-2023`.

References

[1] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *ACM Symposium on User Interface Software and Technology (UIST)*, 2011, pp. 559–568.

[2] H. Benko, R. Jota, and A. Wilson, "Miragetable: Freehand interaction on a projected augmented reality tabletop," in *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2012, pp. 199–208.

[3] J. Chen, S. Izadi, and A. Fitzgibbon, "Kinetre: Animating the world with the human body," in *ACM Symposium on User Interface Software and Technology (UIST)*, 2012, pp. 435–444.

[4] M. Dou, H. Fuchs, and J.-M. Frahm, "Scanning and tracking dynamic objects with commodity depth cameras," in *IEEE Int'l Symposium on Mixed and Augmented Reality (ISMAR)*, 2013, pp. 99–106.

[5] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2013, pp. 3748–3754.

[6] C. Raposo, M. Lourenco, J. Barreto, and M. Antunes, "Plane-based odometry using an rgb-d camera," in *British Machine Vision Conference (BMVC)*, 2013.

[7] Y. Lu and D. Song, "Robust rgb-d odometry using point and line features," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2015, pp. 3934–3942.

[8] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," in *Int'l Symposium Robotics Research (ISRR)*, 2017, pp. 235–252.

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012.

[10] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.

[11] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane slam for hand-held 3d sensors," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2013, pp. 5182–5189.

[12] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 2013, pp. 2100–2106.

[13] R. Mur-Artal and J. D. Tardos, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Trans. Robotics (T-RO)*, vol. 33, no. 5, pp. 1255–1262, 2017.

[14] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns," U.S. Patent 200 802 405O2A1, 10 2, 2008.

[15] C. Bamji, "Method and system for lossless dealiasing in time-of-flight (tof) systems," U.S. Patent 7 791 715B1, 9 7, 2010.

[16] C. L. Niclass, A. Shpunt, G. A. Agranov, M. C. Waldon, M. A. Rezk, and T. Oggier, "Light detection and ranging sensor," U.S. Patent 10 324 171B2, 6 18, 2019.

[17] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, vol. 3, 1991, pp. 2724–2729.

[18] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 14, no. 2, pp. 239–256, 1992.

[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[20] P. Miraldo, S. Saha, and S. Ramalingam, "Minimal solvers for mini-loop closures in 3d multi-scan alignment," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019.

[21] H. Stewenius, M. Oskarsson, K. Astrom, and D. Nister, "Solutions to minimal generalized relative pose problems," in *Workshop on Omnidirectional Vision (OMNIVIS)*, 2005.

[22] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2009, pp. 3212–3217.

[23] J. Xiao, A. Owens, and A. Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2013, pp. 1625–1632.

[24] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *Int'l J. Computer Vision (IJCV)*, vol. 13, no. 2, pp. 119–152, Oct 1994.

[25] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm." in *3-D Digital Imaging and Modeling (3DIM)*, vol. 1, 2001, pp. 145–152.

[26] 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*, vol. 3, 2002, pp. 545–548.

[27] A. W. Fitzgibbon, "Robust registration of 2d and 3d point sets," *Image and Vision Computing (IVC)*, vol. 21, no. 13, pp. 1145–1153, 2003.

[28] Y. Tsin and T. Kanade, "A correlation-based approach to robust point set registration," in *European Conf. Computer Vision (ECCV)*, 2004, pp. 558–569.

[29] J. Minguez, L. Montesano, and F. Lamiraux, "Metric-based iterative closest point scan matching for sensor displacement estimation," *IEEE Trans. Robotics (T-RO)*, vol. 22, no. 5, pp. 1047–1054, 2006.

[30] A. Myronenko, X. Song, and M. A. Carreira-Perpian, "Non-rigid point set registration: Coherent point drift," in *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[31] H. Li and R. Hartley, "The 3d-3d registration problem revisited," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2007, pp. 1–8.

[32] J. Yang, H. Li, and Y. Jia, "Go-icp: Solving 3d registration efficiently and globally optimally," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2013, pp. 1457–1464.

[33] C. Olsson, F. Kahl, and M. Oskarsson, "Branch-and-bound methods for euclidean registration problems," *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 31, no. 5, pp. 783–794, 2009.

[34] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5556–5565.

[35] N. Mellado, D. Aiger, and N. J. Mitra, "Super 4pcs fast global pointcloud registration via smart indexing," *Computer Graphics Forum*, vol. 33, no. 5, pp. 205–215, 2014.

[36] D. Aiger, N. J. Mitra, and D. Cohen-Or, "4-points congruent sets for robust surface registration," *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2008)*, vol. 27, no. 3, pp. #85, 1–10, 2008.

[37] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European Conf. Computer Vision (ECCV)*, 2016, pp. 766–782.

[38] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2017, pp. 143–152.

[39] S. Ramalingam and Y. Taguchi, "A theory of minimal 3d point to 3d plane registration and its generalization," *Int'l J. Computer Vision (IJCV)*, vol. 102, no. 1, pp. 73–90, 2013.

[40] W. Forstner and K. Khoshelham, "Efficient and accurate registration of point clouds with plane to plane correspondences," in *IEEE Int'l Conf. Computer Vision Workshops (ICCVW)*, 2017, pp. 2165–2173.

[41] C. Raposo and J. Barreto, "3d registration of curves and surfaces using local differential information," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9300–9308.

[42] V. M. Govindu, "Combining two-view constraints for motion estimation," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2001.

[43] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *Int'l J. Computer Vision (IJCV)*, vol. 103, no. 3, pp. 267–305, 2013.

[44] U. Bhattacharya and V. M. Govindu, "Efficient and robust registration on the 3d special euclidean group," in *IEEE Int'l Conf. Computer Vision (ICCV)*, vol. 2, 2019.

[45] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor

environments," *The International Journal of Robotics Research (IJRR)*, vol. 31, no. 5, pp. 647–663, 2012.

[46] F. Steinbrucker, C. Kerl, D. Cremers, and J. Sturm, "Large-scale multi-resolution surface reconstruction from rgb-d sequences," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2013, pp. 3264–3271.

[47] T. Whelan, M. Kaess, J. J. Leonard, and J. McDonald, "Deformation-based loop closure for large scale dense rgb-d slam," in *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 2013, pp. 548–555.

[48] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Trans. Robotics (T-RO)*, vol. 30, no. 1, pp. 177–187, 2014.

[49] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^2o: A general framework for graph optimization," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2011, pp. 3607–3613.

[50] P. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding (CVIU)*, vol. 78, no. 1, pp. 138–156, 2000.

[51] O. Chum, J. Matas, and J. Kittler, "Locally optimized ransac," in *Pattern Recognition*, 2003, pp. 236–243.

[52] O. Chum and J. Matas, "Matching with prosac - progressive sample consensus," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2005, pp. 220–226.

[53] D. Barath and J. Matas, "Graph-cut ransac," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6733–6741.

[54] D. Barath, J. Matas, and J. Noskova, "Magsac: Marginalizing sample consensus," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 189–10 197.

[55] F. Kluger, E. Brachmann, H. Ackermann, C. Rother, M. Y. Yang, and B. Rosenhahn, "Consac: Robust multi-model fitting by conditional sample consensus," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4633–4642.

[56] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 26, no. 6, pp. 756–770, 2004.

[57] H. Li and R. Hartley, "Five-point motion estimation made easy," in *IEEE Int'l Conf. Pattern Recognition (ICPR)*, vol. 1, 2006, pp. 630–633.

[58] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg, "A minimal solution to the generalized pose-and-scale problem," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 422–429.

[59] P. Miraldo, T. Dias, and S. Ramalingam, "A minimal closed-form solution for multi-perspective pose estimation using points and lines," in *European Conf. Computer Vision (ECCV)*, 2018, pp. 490–507.

[60] L. Kneip, D. Scaramuzza, and R. 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)*, 2011, pp. 2969–2976.

[61] G. H. Lee, B. Li, M. Pollefeys, and F. Fraundorfer, "Minimal solutions for the multi-camera pose estimation problem," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 837–848, 2015.

[62] V. Larsson, Z. Kukelova, and Y. Zheng, "Making minimal solvers for absolute pose estimation compact and robust," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2335–2343.

[63] F. Camposeco, A. Cohen, M. Pollefeys, and T. Sattler, "Hybrid camera pose estimation," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 136–144.

[64] P. H. Schonemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[65] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust & efficient point cloud registration using pointnet," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7163–7172.

[66] X. Li, J. K. Pontes, and S. Lucey, "Pointnetlk revisited," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 12 758–12 767.

[67] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2019.

[68] G. D. Pais, P. Miraldo, S. Ramalingam, V. M. Govindu, J. C. Nascimento, and R. Chellappa, "3DRegNet: A deep neural network for 3d point

registration," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020.

[69] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 13, no. 4, pp. 376–380, 1991.

[70] C. Choy, W. Dong, and V. Koltun, "Deep global registration," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2511–2520.

[71] G. Elbaz, T. Avraham, and A. Fischer, "3d point cloud registration for localization using a deep neural network auto-encoder," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4631–4640.

[72] H. Deng, T. Birdal, and S. Ilic, "Ppfnet: Global context aware local features for robust 3d point matching," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 195–205.

[73] ——, "Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors," in *European Conf. Computer Vision (ECCV)*, 2018, pp. 602–618.

[74] Z. Jian Yew and G. Hee Lee, "3dfeat-net: Weakly supervised local 3d features for point cloud registration," in *European Conf. Computer Vision (ECCV)*, 2018, pp. 607–623.

[75] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3128–3137.

[76] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2019, pp. 8957–8965.

[77] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan, and C.-L. Tai, "D3feat: Joint learning of dense detection and description of 3d local features," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6358–6366.

[78] Z. J. Yew and G. H. Lee, "Rpm-net: Robust point matching using learned features," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 821–11 830.

[79] S. Gold, C.-P. Lu, A. Rangarajan, S. Pappu, and E. Mjolsness, "New algorithms for 2d and 3d point matching: Pose estimation and correspondence," in *Advances in Neural Information Processing Systems (NIPS)*, 1994, pp. 957–964.

[80] S. Huang, Z. Gojcic, M. Usvyatsov, A. Wieser, and K. Schindler, "Predator: Registration of 3d point clouds with low overlap," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 4265–4274.

[81] I. Csiszár and G. Tusnády, "Information geometry and alternating minimization procedures," *Statistics and Decisions, Supplement Issue*, vol. 1, pp. 205–237, 1984.

[82] H. H. Bauschke and J. M. Borwein, "On projection algorithms for solving convex feasibility problems," *SIAM Review*, vol. 38, no. 3, pp. 367–426, 1996.

[83] U. Niesen, D. Shah, and G. W. Wornell, "Adaptive alternating minimization algorithms," *IEEE Trans. Information Theory*, vol. 55, no. 3, pp. 1423–1429, 2009.

[84] X. Zhou, S. Leonardos, X. Hu, and K. Daniilidis, "3d shape reconstruction from 2d landmarks: A convex formulation," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015.

[85] X. Zhou, M. Zhu, and K. Daniilidis, "Multi-image matching via fast alternating minimization," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2015.

[86] J. Yan, J. Wang, H. Zha, X. Yang, and S. M. Chu, "Multi-view point registration via alternating optimization," in *AAAI Conference on Artificial Intelligence*, 2015.

[87] T. Schops, T. Sattler, and M. Pollefeys, "Bad slam: Bundle adjusted direct rgb-d slam," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019.

[88] J. Campos, J. R. Cardoso, and P. Miraldo, "Poseamm: A unified framework for solving pose problems using an alternating minimization method," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2019, pp. 3493–3499.

[89] S. Ranade, X. Yu, S. Kakkar, P. Miraldo, and S. Ramalingam, "Mapping of sparse 3d data using alternating projection," in *Asian Conf. Computer Vision (ACCV)*, 2020, pp. 295–313.

[90] A. Mateus, S. Ramalingam, and P. Miraldo, "Minimal solvers for 3d scan alignment with pairs of intersecting lines," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[91] H. Pottmann and J. Wallner, *Computational Line Geometry*, 1st ed. Springer-Verlag Berlin Heidelberg, 2001.

[92] R. Pless, "Using many cameras as one," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 2, June 2003, p. 587.

[93] Z. Kukelova, M. Bujnak, , and T. Pajdla, "Automatic generator of minimal problem solvers," in *European Conf. Computer Vision (ECCV)*, 2008, pp. 302–315.

[94] Z. Kukelova, J. Heller, and A. Fitzgibbon, "Efficient intersection of three quadrics and applications in computer vision," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1799–1808.

[95] S. Bhayani, Z. Kukelova, and J. Heikkilä, "Computing stable resultant-based minimal solvers by hiding a variable," in *IEEE Int'l Conf. Pattern Recognition (ICPR)*, 2021, pp. 6104–6111.

[96] H. Yang, J. Shi, and L. Carlone, "Teaser: Fast and certifiable point cloud registration," *IEEE Trans. Robotics (T-RO)*, vol. 37, no. 2, pp. 314–333, 2020.

[97] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing," *arXiv:1801.09847*, 2018.

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

[99] K. Li, J. Yao, and X. Lu, "Robust line matching based on ray-point-ray structure descriptor," in *Asian Conf. Computer Vision (ACCV)*, 2014, pp. 554–569.

[100] K. Li, J. Yao, X. Lu, L. Li, and Z. Zhang, "Hierarchical line matching based on line–junction–line structure descriptor and local homography estimation," *Neurocomputing*, vol. 184, pp. 207–220, 2016.

[101] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.