# Compact World Anchors: Registration Using Parametric Primitives as Scene Description

Fernando Reyes-Aviles, *Student Member, IEEE,* Philipp Fleck, *Student Member, IEEE,*
Dieter Schmalstieg, *Fellow, IEEE,* and Clemens Arth, *Member, IEEE*

**Abstract**—We present a registration method relying on geometric constraints extracted from parametric primitives contained in 3D parametric models. Our method solves the registration in closed-form from three line-to-line, line-to-plane or plane-to-plane correspondences. The approach either works with semantically segmented RGB-D scans of the scene or with the output of plane detection in common frameworks like ARKit and ARCore. Based on the primitives detected in the scene, we build a list of descriptors using the normals and centroids of all the found primitives, and match them against the pre-computed list of descriptors from the model in order to find the scene-to-model primitive correspondences. Finally, we use our closed-form solver to estimate the 6DOF transformation from three lines and one point, which we obtain from the parametric representations of the model and scene parametric primitives. Quantitative and qualitative experiments on synthetic and real-world data sets demonstrate the performance and robustness of our method. We show that it can be used to create *compact world anchors* for indoor localization in AR applications on mobile devices leveraging commercial SLAM capabilities.

**Index Terms**—Camera localization, correspondence problem, 3D registration, closed-form method, augmented reality

## 1 Introduction

Registration of 3D models is a fundamental problem in computer vision across a wide range of applications, such as localization in augmented reality (AR) or mesh completion in 3D reconstruction. Given a set of correspondences from live scene data (*e.g.*, a 3D point cloud) to a prior reference (*e.g.*, a 3D parametric model), registration consists in finding the optimal transformation $\mathbf{T} = [\ \mathbf{R} \mid \mathbf{t}\ ] \in \mathbf{SE}(3)$ with six degrees of freedom that minimizes the rotational and translational error between scene and model.

Solving the registration problem in a robust fashion is crucial in AR applications. The term *world anchor* (or, sometimes, *spatial anchor*) denotes a model curated for the purpose of re-localizing (*i.e.*, registering) the camera with respect to a given scene. World anchors are usually created to find correspondences between keypoints and consist of heterogeneous data collections, such as RGB keyframes, IMU data, feature descriptors, 3D point clouds, *etc*. The data types making up an anchor depend on the platform's sensing and computational capabilities. For example, traditional monocular SLAM [25, 26] uses keyframes and small blurry images. In contrast, the "Azure spatial anchors" stored by the Microsoft HoloLens use a proprietary combination of imagery from multiple cameras with partial 3D reconstructions. In general, the implementations of world anchors are very diverse. Not only are they lacking interchangeability across devices and platforms, but they also have a substantial storage footprint, ranging from a few megabytes to tens of megabytes in size.

Registration of an anchor usually relies on the large amount of previous research on point-to-point, point-to-line, and point-to-plane registration methods [17, 19, 36, 50]. For instance, classical algorithms for solving point-to-point registration problems were proposed by Arun *et al*. [2], Umeyama [44] and Horn [21], the latter being an essential part of the widely used iterative closest point (ICP) method [4]. Iterative methods such as ICP perform very well with a good initialization and dense data models, but they require partially overlapping 3D surfaces and are often slow to converge. In contrast, closed-form algorithms [8, 45, 46] leverage the collinearity or coplanarity of points [32, 38] or normals [10] to find a solution in a fixed number of steps.

In this paper, we propose to raise the level of abstraction of world anchors using *parametric models*, which represents a fresh approach to the localization problem within known scenes (see Figure 1). Our contributions are two-fold:

**Compact world anchors.** First, we introduce a scene representation designed to solve the correspondence problem purely from geometric information, using matches between sparse parametric primitives. The memory footprint of our representation is just a few kilobytes; hence, we call it *compact world anchor*. Our anchors contain pre-computed point pair feature (PPF) descriptors created from parametric primitives (planes, cylinders, spheres) identified in the reference data. As the anchors only consist of geometry-based descriptors, they are exchangeable across platforms and devices, and illumination-independent. We compute such descriptors using the normals and centroids extracted from parametric representations of the primitives (*e.g.*, plane equations). To find the scene-to-model correspondences, we search possible assignments between the PPF set stored in the anchor and the PPF set estimated online from primitives detected in the scene. Primitive detection in the live scene either relies on semantic segmentation [42] or on the plane detection in AR Foundation[1]. Since the plane detection tends to struggle in defining precise boundaries, we cannot rely on finding distinct keypoints, such as a corner of a table. This situation precludes the use of point-to-*any* correspondences [7, 8, 35, 45]. Instead, we solve the registration entirely without keypoints.

- *F. Reyes-Aviles is with VRVis Competence Center in Vienna, Austria. E-mail: fernando.reyes-aviles@icg.tugraz.at*
- *P. Fleck, D. Schmalstieg and C. Arth are with Graz University of Technology, Austria.*

[1] ARFoundation: https://unity.com/unity/features/arfoundation, a Unity wrapper around ARKit and ARCore.

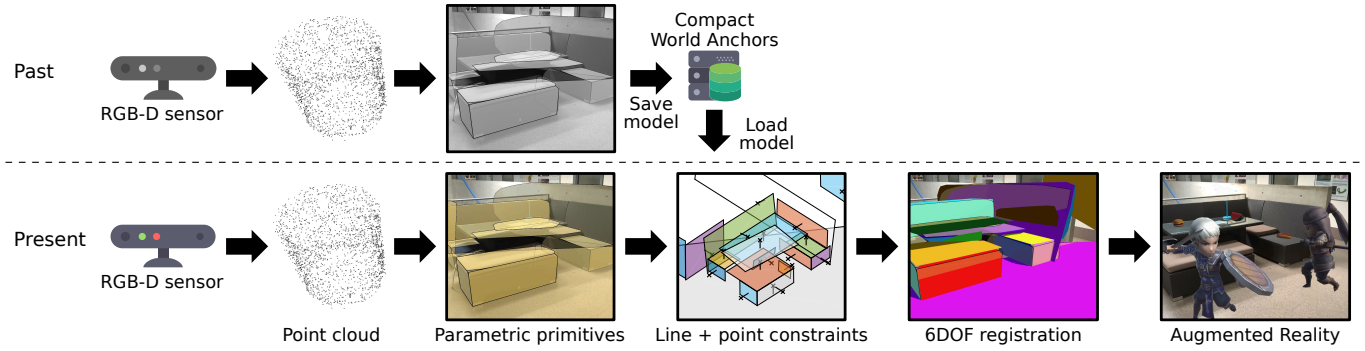



Fig. 1: Registration of a *parametric* model and virtual objects to *semantic* scene data by leveraging correspondences from parametric primitives. We use SLAM to scan a scene with an RGB-D camera and obtain parametric primitives (semantic data), such as planes, from which line and point constraints are harvested. This kind of scene description, which we call a compact world anchor, can serve to compute a 6DOF scene registration with a map from SLAM obtained at a later point. The main advantage of our scene description is its compact format ($<$ 100 KB), which requires little storage and computation power.

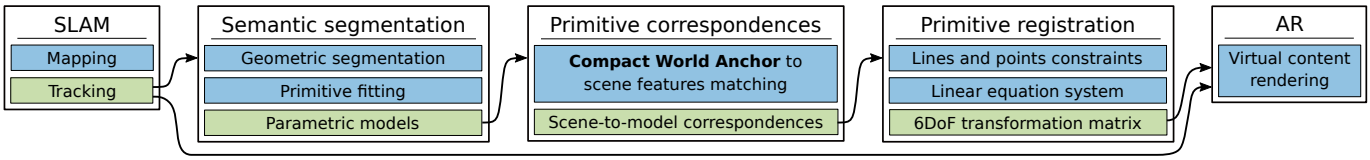



Fig. 2: We use a SLAM front-end to scan a known scene and get a real-time updated pose in the camera coordinate system. Note that we only use SLAM to establish a consistent track of poses, while we completely ignore the 3D mapping. If the SLAM system does not have any *plane detection* capability (like AR Foundation), we execute a semantic segmentation on the scene raw point clouds in order to find primitives in the scene. Next, we estimate the scene-to-model correspondences between our compact world anchor and geometric features in the scene. We use such correspondences to obtain line and point constraints from model and scene parametric primitives. We use these constraints to build a linear equation estimate that solves the model-to-scene registration. Finally, we use the pose given by the SLAM system and the given transformation matrix to render virtual objects in the observed environment.

**Closed-form registration from sparse correspondences.** Second, we introduce a novel solver for determining the registration of an anchor to a live scene. Because both model and live scene data are parametric and sparse, we cannot use iterative methods [37] to solve the correspondences and registration simultaneously. Instead, we introduce a closed-form method leveraging line-to-line, line-to-plane and plane-to-plane correspondences for registration. We choose three corresponding primitives to extract geometric constraints and stack them in a linear equation system to solve the model-to-scene pose.

In our experiments, we show that our method can localize the camera accurately in room-size environments where a few dozens of parametric primitives have been observed. With its lightweight footprint, our localization method can work with any SLAM + semantic segmentation framework like AR Foundation, in particular on mobile devices, to replace the native heavy-weight scene descriptions. We demonstrate the utility of our approach with examples in guidance, training, and indoor localization.

## 2 Related Work

Our work is mainly concerned with registration, pose estimation, camera localization, and the related topic of semantic SLAM. The registration problem has a long history of research, with a large number of solutions relying on keypoint correspondences. Previous work can be divided into methods which aim to solve correspondences and pose estimation simultaneously [11, 28, 49] and methods that determine the registration given such correspondences [21, 31, 44, 46]. Among the latter, least squares approaches are most widely studied [2, 5, 14, 51].

### 2.1 Least-squares methods for pose estimation

Least-squares solvers, while conceptually simple, require careful numerical conditioning to yield a stable solution. Some variants use a Cayley-Gibbs-Rodriguez (CGR) parameterization of rotation matrices [18, 27, 43] to reduce the number of unknowns, but this increases the complexity of the methods, often compromising their numerical stability. Consequently, a large set of correspondences may be needed, increasing the computational complexity [20].

As rich enough data sources for solving the correspondence problem under these circumstances, most methods rely on keypoint descriptors [1, 8, 32, 50]. In contrast, our method trades keypoints for parametric primitives, which are sufficient even if the model is sparse.

### 2.2 Minimal solvers

Linear and algebraic solutions are less sensitive in terms of numerical stability, but at the cost of having to solve an equation system with more unknown parameters. Some methods use intermediate transformations [8, 32, 38] or pre-rotations [10, 46] to relax the original problem and find the rigid-transformation matrix in multiple steps. Unlike these methods, we leverage the properties of 3D geometric primitives to build a linear equation system in 12 unknowns which solves the registration in a single step.

Ramalingam and Taguchi [38] developed a family of minimal solvers relying on point-to-plane correspondences. They solve the registration problem by building a specific equation system for each minimal combination of point-to-plane correspondences. In contrast, our method always uses the same equation system

independent of the observed correspondences (*e.g.*, three planes, two planes and one line, two lines and one plane, *etc*.). Also, our approach does not suffer from degeneracies or singularities when all the planes are orthogonal.

Mateus *et al*. [32] proposed a minimal registration method similar to ours. They use point-to-point and plane-to-plane correspondences to register sets of overlapping point clouds, whereas we use line-to-line, line-to-plane and plane-to-plane correspondences to register model to scene.

Camposeco *et al*. [8] developed a minimal solver based on one point-to-point and two point-to-ray correspondences (*i.e.*, one 3D-3D match and two 2D-3D matches). They obtain the 3D-3D match by triangulating a point from two or more views. The 2D-3D matches are used to infer the location of two other points. Once the 3D location of the three points is recovered, they use the algorithm proposed by Umeyama [44] to estimate the camera pose. Instead of using point-to-ray matches, we use semantic information in the form of parametric primitives to estimate the camera pose.

The registration of lines to planes was also studied by Chen [10]. The author uses lines, the normal vector of planes and a given point on the line to solve the problem. Chen's work also includes a thorough study of the existence of a solution and gives five theorems stating the necessary and sufficient conditions under which the problem can be solved. We refer the reader to his paper for a detailed explanation of such conditions.

### 2.3 Model-based localization

Model-based localization within known environments plays an important role in augmented reality. Previous methods focus either on single-shot localization [24, 41] or robust relocalization to allow instant recovery from tracking failure [16, 47].

Shotton *et al*. [41] developed the SCoRe Forest method to estimate the pose of a camera relative to a known 3D scene from a single RGB-D frame. They infer the camera pose from 3D scene points to 2D image pixels correspondences. Glocker *et al*. [16] propose a randomized ferns encoding for instant recovery from tracking failure. Their mesh-to-volume registration does not need to find explicit scene-to-model correspondences; however, they need a good initialization to use their approach for AR applications. The user must provide such an initial solution by manually aligning the model to the scene.

Rather than using raw point clouds from SLAM, our method uses input data which is on a higher level of abstraction. The only input data to our proposed method is our compact world anchor and the parametric representations of the primitives detected in the scene. By doing so, we decouple the localization problem within known environments from the underlying *device tracking* and *surface detection* technologies. Also, our method does not need any user interaction; we support automatic camera localization within known scenes.

### 2.4 Semantic SLAM

Volumetric integration for SLAM was pioneered by KinectFusion [33], while InfiniTAMv3 [23] showed how volumetric integration can be done rapidly even for large scenes. Even though many refinements exist, these approaches still represent the state of the art for non-semantic SLAM from depth images.

One of the first algorithms proposed to incorporate semantic information was SLAM++ [40]. It incorporates semantic information on a per-object level. Its objects are meshes contributing error terms to the overall graph-based optimization problem. Later approaches, such as QuadricSLAM [34] and others [9, 29, 30, 48] are similar in that they all use specific complex object instances rather than generic shapes.

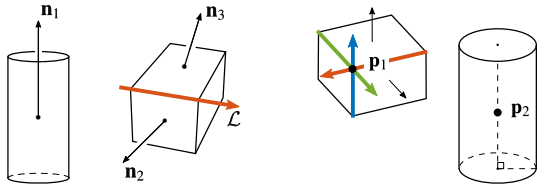



Fig. 3: Sources of lines and points used to register 3D parametric primitives. (a) We can obtain lines from the normal vector of a primitive (*e.g.*, $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3$) or the intersection of two planes (*e.g.*, $\mathcal{L}$). (b) A point may be obtained from the intersection of three planes (*e.g.*, $\mathbf{p}_1$), the midpoint of the axis of a cylinder (*e.g.*, $\mathbf{p}_2$), *etc*.

While SLAM++ and its successors are indeed closely related to our approach, there are significant differences. First, we are not aiming at reconstructing (mapping) a scene, but we are merely interested in camera pose estimation, akin to localization in SLAM or model-based localization. Second, our algorithm leverages semantic information in the form of parametric primitives derived from semantic segmentation [42] rather than from object-specific (or shape-specific) detectors [3, 15, 22]. This choice of simple but distinctive primitives rather than complex meshes is important, as the cost of creating, storing, and maintaining a scene representation is significantly reduced.

## 3 Registration Method

At the core of our work lies (i) a novel closed-form method for the six degrees of freedom (6DOF) registration based on three correspondences between model and scene, and (ii) the compact world anchor we use to find such correspondences. In the remainder of this section, we assume that the correspondences are already known. Our compact world anchors and the method we use for computing the correspondences is explained in Section 4.

We use the parametric representations from both, scene and model primitives, to obtain line-to-line, line-to-plane or plane-to-plane correspondences, where lines and planes are treated as infinite. We must extract three lines and one point from three scene-to-model correspondences to solve the registration problem. For instance, a line can be obtained from the axis of a cylinder or the intersection of two non-parallel planes (see Figure 3). Similarly, a point can be obtained from the intersection of three planes, of two coplanar lines, or the geometric center of a cylinder.

Our closed-form method is robust and determines a unique solution for $\mathbf{T} = [\,\mathbf{R} \mid \mathbf{t}\,] \in \mathbf{SE}(3)$ from a linear system of equations with 12 unknowns, *i.e.*, we use nine unknowns for rotation to avoid problems caused by input data with numerically poorly conditioning. Consequently, it is not guaranteed that the solution we obtain is a valid rotation matrix $\mathbf{R} \in \mathbf{SO}(3)$, and we must re-orthogonalize the estimated rotation matrix after solving.

The resulting registration is unique up to scale, because three planes (or lines) that intersect in a single point do not reveal the scale. Resolving scale needs more information, which we obtain from the radius, height, width or location of another 3D geometric primitive.

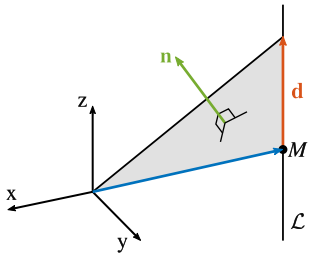



Fig. 4: Geometric interpretation of the Plücker line $\mathcal{L}$. The vector $\mathbf{d}$ gives the direction of the line. $M$ is a point on $\mathcal{L}$, and $\mathbf{n}$ is the normal of the plane spawned by $\mathcal{L}$ and the origin.

In the following sections, we explain how to find lines from plane intersections (Section 3.1) and points from line intersections (Section 3.2). Then, we describe how to set up the linear equation system (Section 3.3) and how to disambiguate the solutions (Section 3.4).

### 3.1 Intersection of planes

A line in 3-space can be represented with Plücker line coordinates as a pair of 3-vectors $\mathbf{d}$ and $\mathbf{n}$, where $\mathbf{d}^\top\mathbf{n}=0$, *i.e.*, $\mathbf{d}$ is orthogonal to $\mathbf{n}$. As shown in Figure 4, the Plücker line coordinates of the line $\mathcal{L}=[\mathbf{d},\mathbf{n}]^\top$ reveal the direction $\mathbf{d}$ of the line and the normal $\mathbf{n}$ of the plane spawned by the line $\mathcal{L}$ and the origin of the coordinate system. A Plücker line $\mathcal{L}$ remains unchanged when undergoing a rotation about $\mathbf{d}$ or a translation along $\mathbf{d}$.

Plücker coordinates provide a convenient representation of the intersection of two planes. We can find the dual line $\mathcal{L}^*$, a $4\times4$ skew-symmetric matrix, formed by the intersection of the planes $P=[P_1,P_2,P_3,P_4]^\top$ and $Q=[Q_1,Q_2,Q_3,Q_4]^\top$ as

$$\mathcal{L}^* = PQ^\top - QP^\top, \tag{1}$$

with $P$ and $Q$ being the 4-vector coefficients of the parametric representation of the planes

$$P_1\mathrm{x}+P_2\mathrm{y}+P_3\mathrm{z}+P_4=0 \;\text{ and }\; Q_1\mathrm{x}+Q_2\mathrm{y}+Q_3\mathrm{z}+Q_4=0. \tag{2}$$

As we mentioned before, the boundaries of planes detected with front-ends like AR Foundation might not be precise. Nevertheless, their parametric equations are always well defined. By using Plücker coordinates, we can obtain stable and precise lines (from intersections of planes) without the need to use points on the lines to represent them.

### 3.2 Intersection of lines

Let $\mathbf{a}$ and $\mathbf{b}$ be direction vectors, and let $\mathbf{p}_a$ and $\mathbf{p}_b$ be points on two coplanar lines (see Figure 5a). With $\mathbf{c}=\mathbf{p}_b-\mathbf{p}_a$, the intersection point $\mathcal{I}$ of the two lines is given by

$$\mathcal{I} = \mathbf{p}_a + s\cdot\frac{\|\mathbf{b}\times\mathbf{c}\|}{\|\mathbf{b}\times\mathbf{a}\|}\,\mathbf{a}, \text{ where } s=\begin{cases}+1, & \text{if } (\mathbf{b}\times\mathbf{c})^\top(\mathbf{b}\times\mathbf{a})>0,\\ -1, & \text{otherwise.}\end{cases} \tag{3}$$

### 3.3 Linear equation system

Given three Plücker lines $\mathcal{L}_i$ in model coordinates and their corresponding three Plücker lines $\mathcal{L}'_i$ in camera coordinates, such that

$$\mathcal{L}'_i = \begin{bmatrix}\mathbf{R} & \mathbf{0}\\ -[\mathbf{t}]_\times\mathbf{R} & \mathbf{R}\end{bmatrix}\mathcal{L}_i, \tag{4}$$

we can construct a linear equation system $\mathbf{Ax}=\mathbf{b}$ to solve the unknown 6DOF transformation formed by the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ that relates $\mathcal{L}_i$ to $\mathcal{L}'_i$. The rotation $\mathbf{R}$ only affects the direction vector of $\mathcal{L}_i=[\mathbf{d},\mathbf{n}]^\top$:

$$\mathbf{d}' = \mathbf{R}\,\mathbf{d} = \begin{bmatrix}\mathrm{R}_{11} & \mathrm{R}_{12} & \mathrm{R}_{13}\\ \mathrm{R}_{21} & \mathrm{R}_{22} & \mathrm{R}_{23}\\ \mathrm{R}_{31} & \mathrm{R}_{32} & \mathrm{R}_{33}\end{bmatrix}\begin{bmatrix}d_1\\ d_2\\ d_3\end{bmatrix}. \tag{5}$$

Therefore, given the three lines $\mathcal{L}_i$, where

$$\begin{aligned}\mathcal{L}_1 &= \begin{bmatrix}a_1,\ a_2,\ a_3,\ \widetilde{a}_1,\ \widetilde{a}_2,\ \widetilde{a}_3\end{bmatrix}^\top,\\ \mathcal{L}_2 &= \begin{bmatrix}b_1,\ b_2,\ b_3,\ \widetilde{b}_1,\ \widetilde{b}_2,\ \widetilde{b}_3\end{bmatrix}^\top,\\ \mathcal{L}_3 &= \begin{bmatrix}c_1,\ c_2,\ c_3,\ \widetilde{c}_1,\ \widetilde{c}_2,\ \widetilde{c}_3\end{bmatrix}^\top,\end{aligned} \tag{6}$$

we construct the following linear equation system $\mathbf{Ax}=\mathbf{b}$ using a point $\mathbf{p}=[p_x,p_y,p_z]^\top$, which can be obtained from intersecting three planes:

$$\mathbf{A} = \begin{bmatrix}
a_1 & a_2 & a_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & a_1 & a_2 & a_3 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & a_1 & a_2 & a_3 & 0 & 0 & 0\\
b_1 & b_2 & b_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & b_1 & b_2 & b_3 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & b_1 & b_2 & b_3 & 0 & 0 & 0\\
c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & c_1 & c_2 & c_3 & 0 & 0 & 0\\
p_x & p_y & p_z & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & p_x & p_y & p_z & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & p_x & p_y & p_z & 0 & 0 & 1
\end{bmatrix},$$

$$\mathbf{x} = \begin{bmatrix}\mathrm{R}_{11}\ \mathrm{R}_{12}\ \mathrm{R}_{13}\ \mathrm{R}_{21}\ \mathrm{R}_{22}\ \mathrm{R}_{23}\ \mathrm{R}_{31}\ \mathrm{R}_{32}\ \mathrm{R}_{33}\ \mathrm{t}_x\ \mathrm{t}_y\ \mathrm{t}_z\end{bmatrix}^\top,$$

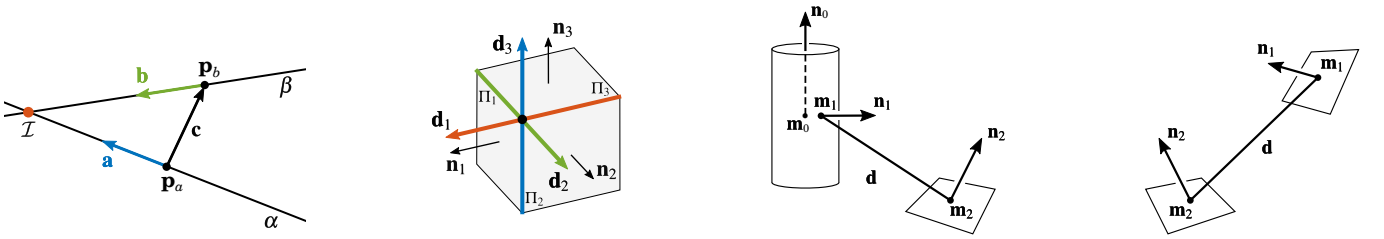
$$\mathbf{b} = \begin{bmatrix}a'_1\ a'_2\ a'_3\ b'_1\ b'_2\ b'_3\ c'_1\ c'_2\ c'_3\ p'_x\ p'_y\ p'_z\end{bmatrix}^\top. \tag{7}$$

The matrix $\mathbf{A}$ has a unique solution as long as the three lines are not in a degenerate configuration [10]. If, for example, all lines are coplanar, or two of them are parallel, the matrix $\mathbf{A}$ will become singular, and there will be no solution.

### 3.4 Direction of lines

The calculation of a line as the intersection of two planes yields two solutions, since the sign of the line direction is undetermined. Consequently, with three lines as input, we obtain $2^3=8$ solutions, which are identical up to a permutation of the axes. A common approach for closed-form registration resolves this ambiguity by verifying which of the permutations is correct with the help of additional points [18, 38]. However, we enforce a consistent orientation between the corresponding lines from scene and model to avoid an explicit verification step after solving the equation system. This approach is preferred if additional points may not be available or not reliable.

To ensure consistent line orientation, we prescribe that all direction vectors $\mathbf{d}_i$ should point in the same direction as the surface normals. For example, given the planes $\Pi_1$ and $\Pi_2$, and their intersection line $\mathbf{d}_3$ (see Figure 5b), we let $\mathbf{d}_3$ point in the same direction as the surface normal of the plane $\Pi_3$. If the line forming the intersection of the planes points away from the normal, we invert the line's direction.




(a) (b) (c) (d)

Fig. 5: (a) Intersection point $\mathcal{I}$ of lines $\alpha$ and $\beta$. (b) Convention established for pointing plane intersection lines in the right direction before building the equation system. A point pair feature $\mathbf{F}$ can be obtained from (c) a point $\mathbf{m}_1$ that lies on the surface of a cylinder and a point $\mathbf{m}_2$ on a plane, or (d) a point $\mathbf{m}_1$ that lies on a plane and a point $\mathbf{m}_2$ on another plane.

### 3.5 Orthonormality constraints

Our registration method is overdetermined, as we solve a 6DOF problem with an equation system in 12 unknowns. Our method is minimal only in the number of corresponding parametric primitives, such as planes, from which we obtain the lines and point constraints. Since we do not rely on point-to-point correspondences for reasons of robustness, we do not use a minimal least-squares solver [2, 8, 21, 44].

We explored the option of using the orthonormality constraints [17] or CGR parameterization [43] to reduce the number of unknowns in the equation system. However, as explained by Wientapper *et al.* [46], solvers based on the Cayley parameterization will not succeed whenever the correct solution for the rotation has an angle of $\pi$.

We could use orthonormality constraints to reduce the number of unknowns from 12 down to nine. This approach could reduce the necessary number of lines from three to two. However, we would also need intermediate transformations [38] to ensure the $9 \times 9$ matrix $\mathbf{A}$ will not become *singular* or *rank-deficient*. For example, if the point happens to coincide with the origin of the coordinate system, and both lines lie on the *xy*-plane, $\mathbf{A}$ has rank 6.

Therefore, we use three lines and one point to ensure we get a unique solution in a single step, regardless of the orientation and position of the lines and the point. In practice, we usually find that more than three primitives are detected in the scene within a few seconds of scanning. Therefore, the advantage of a minimal polynomial system with only six unknowns is negligible.

## 4 Scene Description and Correspondences

So far, we have assumed the scene-to-model correspondences are already known. In this section, we describe how to efficiently create a scene description and the approach to find correspondences. Correspondences are established by analyzing all possible assignments between the primitives detected in the scene and the primitives in the reference model. Usually, localization methods use gradient-based keypoint detectors (*e.g.*, SIFT features) to estimate the camera pose in known scenes. In contrast, we use geometry-based descriptors to create compact world anchors that need very little storage space and computational power.

### 4.1 Compact world anchors

In order to find a proper geometric characteristic, we revisit the work of Drost *et al.* [12] describing a 4-vector descriptor called *point pair feature*. PPF is a method to detect 3D objects in point clouds using descriptors built from oriented point clouds (*i.e.*, each point has an associated normal) to establish correspondences from each scene point to a model point and, ultimately, align the model to the scene.

A PPF describes the relative position and orientation of two oriented points $\mathbf{m}_1$ and $\mathbf{m}_2$, with normals $\mathbf{n}_1$ and $\mathbf{n}_2$, within the same relative coordinate system. Our 3-vector descriptor $\mathbf{F}$ is constructed from $\mathbf{m}_1$ and $\mathbf{m}_2$ as

$$\mathbf{F}(\mathbf{m}_1, \mathbf{m}_2) = (\angle(\mathbf{n}_1, \mathbf{d}),\ \angle(\mathbf{n}_2, \mathbf{d}),\ \angle(\mathbf{n}_1, \mathbf{n}_2)), \tag{8}$$

where $\mathbf{d} = \mathbf{m}_2 - \mathbf{m}_1$, and $\angle$ denotes the angle between two vectors. Drost *et al.* use a fourth dimension characterizing the Euclidean distance from $\mathbf{m}_1$ to $\mathbf{m}_2$ (see Figure 5c). However, our experiments showed that its exclusion from the descriptor presents no significant decrease in robustness (in line with the results obtained by Rusu *et al.* [39]).

In contrast to Drost *et al.*, we build our PPF set from surface points and normals of the primitives. For example, given the cylinder and the plane in Figure 5c, we use the normal $\mathbf{n}_1$ of the point $\mathbf{m}_1$ that lies on the surface of the cylinder, and the normal $\mathbf{n}_2$ of the point $\mathbf{m}_2$ on the plane, to construct our 3-vector descriptor $\mathbf{F}(\mathbf{m}_1, \mathbf{m}_2)$. The point $\mathbf{m}_1$ is the closest point on the cylinder to the point $\mathbf{m}_2$. It is coplanar with the center of the cylinder $\mathbf{m}_0$, and its normal $\mathbf{n}_1$ is orthogonal to the normal of the cylinder $\mathbf{n}_0$. The point $\mathbf{m}_2$ is the centroid of the plane, *i.e.*, the arithmetic mean of all the four corners of the plane. For the example in Figure 5d, we use the point $\mathbf{m}_1$, which is the centroid of the plane on the right of the figure, and the point $\mathbf{m}_2$, which is the centroid of the plane on the left of the figure.

Our compact anchor consists of a PPF collection computed offline from pairs of parametric primitives. Note that not every combination of two primitives in a model yields a meaningful descriptor $\mathbf{F}$. For example, any combination of two planes from a cube will produce the same descriptor. In the end, we would obtain ${}_6P_2 = 30$ identical descriptors. Parallel planes also produce meaningless descriptors, because the three angles $\angle(\mathbf{n}_1, \mathbf{d})$, $\angle(\mathbf{n}_2, \mathbf{d})$, and $\angle(\mathbf{n}_1, \mathbf{n}_2)$ of Equation 8 would be equal in that case. Therefore, we exclude descriptors $\mathbf{F}$ from parallel planes, planes with equal areas, adjacent planes, as well as primitives with the same height or radius (*e.g.*, two identical cylinders).

### 4.2 Finding correspondences

For every descriptor $\mathbf{F}_i^{\mathrm{S}}(\mathbf{s}_a^i, \mathbf{s}_b^i)$, which we estimate online from the primitives detected in the scene, we run a brute-force search of a corresponding model descriptor $\mathbf{F}_j^{\mathrm{A}}(\mathbf{m}_a^j, \mathbf{m}_b^j)$ in our anchor. First, as depicted in Figure 6, we measure the similarity $\varepsilon_i$ of every scene descriptor against each model descriptor as

$$\varepsilon_i = \min \left\{ \left\| \mathbf{F}_j^{\mathrm{A}} - \mathbf{F}_i^{\mathrm{S}} \right\|_{j=1}^{m} \right\}, \quad i = 1, \ldots, n \leq {}_{\mathcal{S}}P_2, \tag{9}$$

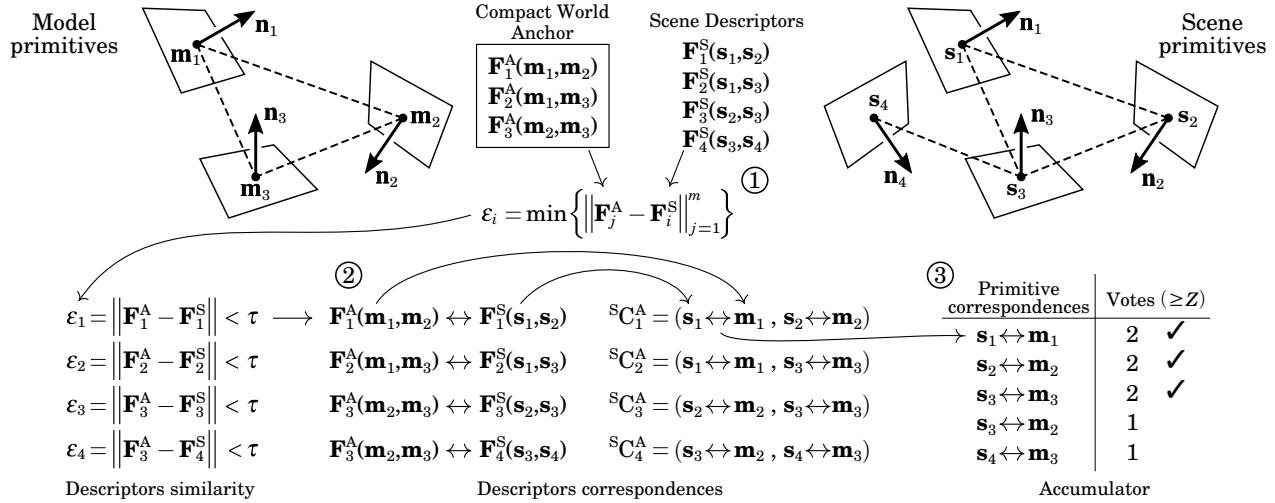



Fig. 6: Steps for robust scene-to-model correspondences estimation using our compact scene representation. (1) We measure the similarity $\varepsilon_i$ between the scene descriptors and the descriptors in our compact world anchor. (2) All the $\varepsilon_i < \tau$ give us scene-to-model *descriptor correspondences*, which, in turn, cast *primitive correspondences* votes that we collect in an accumulator. (3) The correspondences $\mathbf{s} \leftrightarrow \mathbf{m}$ with at least $Z$ votes give us the scene-to-model correspondences. When a model primitive is mapped to more than one scene primitive (or vice versa), we take the correspondence with the maximum number of votes. In this example, $\mathbf{m}_2$ is mapped to $\mathbf{s}_2$ and $\mathbf{s}_3$, since $\mathbf{s}_2 \leftrightarrow \mathbf{m}_2$ received more votes than $\mathbf{s}_3 \leftrightarrow \mathbf{m}_2$; we take the former as the right correspondence.

where $\mathcal{S}$ is the number of primitives in the scene, $n$ is the size of the PPF set computed from the primitives in the scene, and $m$ is the size of the PPF set comprising the compact world anchor.

Secondly, we take all the $\varepsilon_i$ below a threshold $\tau$ to create scene-to-model *descriptor correspondences*

$$ {}^{\mathrm{S}}\mathbf{C}_k^{\mathrm{A}} = \left( \mathbf{s}_a^i \leftrightarrow \mathbf{m}_a^j \ , \ \mathbf{s}_b^i \leftrightarrow \mathbf{m}_b^j \right) \ \ \forall \ \ \varepsilon_i < \tau, \quad 0 < k \leq n, \qquad (10) $$

where $\mathbf{s}_a^i \leftrightarrow \mathbf{m}_a^j$ and $\mathbf{s}_b^i \leftrightarrow \mathbf{m}_b^j$ represent a scene-to-model *primitive correspondence*. The threshold $\tau = 0.1$ was empirically determined from the results obtained in Section 6.2. We kept the same value for the experiments in Sections 6.4, 6.5 and 6.6, since we always obtained good results.

In scenes with symmetries or clutter, the $\varepsilon_i < \tau$ test may produce false-positive *descriptor correspondences* ${}^{\mathrm{S}}\mathbf{C}_k^{\mathrm{A}}$. Consequently, at the end of our *primitive correspondence* matching, we use a voting scheme for robust matching. The votes are collected in an accumulator $\mathcal{A}$ of size $\ell \leq 2p$, with $p$ being the number of *descriptor correspondences* ${}^{\mathrm{S}}\mathbf{C}_k^{\mathrm{A}}$. A primitive may be associated with multiple descriptors $\mathbf{F}$, because every descriptor $\mathbf{F}$ is computed from two primitives. Therefore, every scene-to-model *descriptor correspondence* ${}^{\mathrm{S}}\mathbf{C}_k^{\mathrm{A}}$ casts a vote for both of its two associated scene-to-model *primitive correspondences* $\mathbf{s}_a^i \leftrightarrow \mathbf{m}_a^j$ and $\mathbf{s}_b^i \leftrightarrow \mathbf{m}_b^j$.

If a correspondence $\mathbf{s}^i \leftrightarrow \mathbf{m}^j$ receives at least $Z$ number of votes, we attest that we have found a scene-to-model correspondence. The metric $Z$, given by

$$ Z = \mu - \sigma \cdot \zeta, \quad \mu = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{A}_i, \quad \sigma = \sqrt{\frac{\sum_{i=1}^{\ell} (\mathcal{A}_i - \mu)^2}{\ell - 1}}, \qquad (11) $$

indicates the number of standard deviations $\sigma$ by which the $\mathcal{A}_i$ differ from the mean value $\mu$ of the accumulator $\mathcal{A}$. The variable $\zeta = 1.5$ is a threshold we use to control the lower bound of the metric $Z$. The value of $\zeta$ was chosen based on the results described in Sections 6.2, 6.4, 6.5 and 6.6.

When a model primitive is mapped to two (or more) different scene primitives (or vice versa) by multiple $\mathbf{s}^i \leftrightarrow \mathbf{m}^j$ correspondences that received at least $Z$ number of votes, we take the correspondence $\mathbf{s}^i \leftrightarrow \mathbf{m}^j$ with the maximum number of votes.

The maximum theoretical number of $\varepsilon_i$ is ${}_{\mathcal{S}}P_2$ (see Equation 9 and the first step in Figure 6), where $\mathcal{S}$ is the number of primitives detected in the scene. Therefore, the maximum theoretical number of votes to be considered is $2_{\mathcal{S}}P_2$, because every $\varepsilon_i$ produces a descriptor correspondence ${}^{\mathrm{S}}\mathbf{C}_k^{\mathrm{A}}$ (see Equation 10), and each correspondence casts two votes (see step 2 in Figure 6). However, not all $\varepsilon_i < \tau$. Moreover, the size of the PPF set extracted from the primitives in the scene is usually smaller than ${}_{\mathcal{S}}P_2$. Therefore, the maximum number of votes to be considered is typically much smaller than $2_{\mathcal{S}}P_2$, resulting in faster computation.

## 5 Parametric Primitives and Registration

Assuming a PPF set has been prepared offline and stored as an anchor, online registration still requires to detect parametric primitives at runtime in order to finally calculate the overall model-to-scene transform.

### 5.1 Detection of parametric primitives

We need to extract three lines and one point from corresponding primitives in scene and model. For detecting such primitives in the live scene, we use two different technologies, the semantic segmentation of Stanescu *et al*. [42] and AR Foundation.

On the one hand, Stanescu *et al*. [42] fit primitives (planes, cylinders, spheres, *etc*.) to raw point clouds by estimating their parameters within a RANSAC [13] framework, and improve the fitted models, as the sensor moves through the scene. This semantic segmentation method needs point clouds which are registered in a global coordinate system, *i.e.*, the segmentation works on top of an existing SLAM system like KinectFusion [33] or InfiniTAMv3 [23].

On the other hand, AR Foundation is a multi-platform commercial framework for AR development within the Unity game engine, of which we use two main features, namely *device tracking* and *plane detection*. This framework uses visual natural features combined with the device's IMU measurements to estimate the pose of the device. In order to find geometric planes, it looks for

clusters of keypoints that appear to lie on common horizontal or vertical surfaces[2].

Both technologies produce a semantic model of the scene consisting of a set of primitives and, to create our compact world anchor, we use only the parametric representations of the primitives in the model. Therefore, the only input data to our proposed method consists of the compact anchor and the parametric representations of the primitives detected in the scene, from which we estimate geometric correspondences. Since we use SLAM only to establish a consistent track of poses, we can replace the SLAM system with any device tracking approach.

### 5.2 Robust registration estimation

Given a set of correspondences, we need to select the best subset of correspondences in order to get the best solution. We wrap our closed-form method in a RANSAC [13] loop to account for all possible combinations of minimal sets of primitives. Our solver uses only three parametric primitives to solve the model-to-scene registration problem. When more than three scene primitives are available, we compute the registration matrix $\mathbf{T} = [\,\mathbf{R} \mid \mathbf{t}\,] \in \mathbf{SE}(3)$ for all possible subsets of three primitives. Then, we evaluate the estimated transformations by using them to transform the model primitives and computing the mean rotational and translational error between the transformed model primitives and their corresponding scene primitives. We measure the rotational error by computing the normal deviation error, and, the translational error by computing the primitive-to-primitive Euclidean distance. Finally, the transformation $\mathbf{T}$ which minimizes the mean registration error between model and scene data is selected as the solution. Inside our RANSAC scheme, we also reject subsets of primitives which produce degenerate configurations, such as parallel planes or lines.

## 6 Experimental Results

We first analyzed the performance of our method on synthetic data and then used real-world data sets for validation. In order to show the practical value, we also include several test scenarios and describe the results in the following[3].

### 6.1 Simulations

For the simulated experiments, we created 100 random scenes, each one of them consisting of a cube, a box and two cylinders, with the cube being the smallest object with a side length of five units. All four objects were spawned inside a square area with a side length of 50 units. The planes are represented by four corner points and a parametric equation. The cylinders are represented by two points, the centers of the top and bottom base, and a radius.

A test set was created in three steps. First, we placed the objects at random non-overlapping positions, then we transformed all the parametric primitives of the scene using random rotations and translations, and finally we added Gaussian noise to the points defining the primitives.

We tested $10^4$ trials (100 random transformations for each of the 100 random scenes) with different levels of noise (see Figure 7). We compare our method to the minimal solvers of Arun *et al.* [2], Horn [21], and Ramalingam and Taguchi [38]. The latter provide

[2]Note that newer generations of devices featuring LIDAR or time-of-flight sensors include real depth sensing to facilitate primitive detection.

[3]We refer the reader to the supplementary material video for a more compelling demonstration.

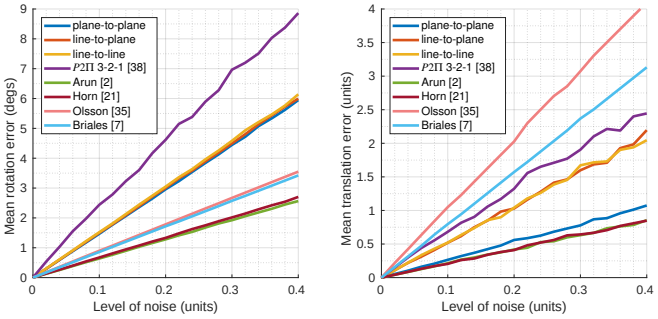
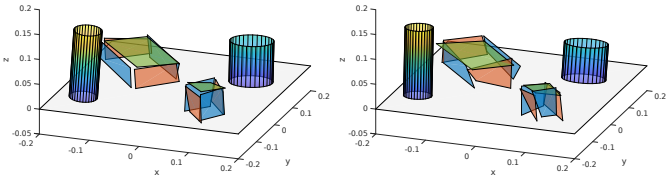



Fig. 7: Rotation and translation error for $10^4$ tests as a function of the level of noise on the parametric primitives of the scenes. The proposed algorithm was tested using only line-to-line, line-to-plane, and plane-to-plane correspondences at a time. The rotation error was similar for all three types of correspondences, while plane-to-plane correspondences produced the best results in terms of translational error.



(a) (b)

Fig. 8: Example of the effect of noise on the parametric primitives for a given scene with a level of noise of (a) 0.4 and (b) 0.6 units.

ten different point-to-plane solvers. In particular, we compare our method to their $Points(3,2,1) \leftrightarrow Planes(3)$ solver ($P2\Pi$ 3-2-1), since they present it as the main configuration (it does not achieve the lowest registration error, but it is the fastest one).

The rotational and translational errors, $e_R(\mathbf{R})$ and $e_t(\mathbf{t})$, were measured as

$$e_R(\mathbf{R}) = \text{acos}\left(\frac{1}{2}(\text{Tr}(\mathbf{R}^\top \mathbf{R}_{\text{GT}}) - 1)\right), \quad e_t(\mathbf{t}) = \|\mathbf{t} - \mathbf{t}_{\text{GT}}\|. \quad (12)$$

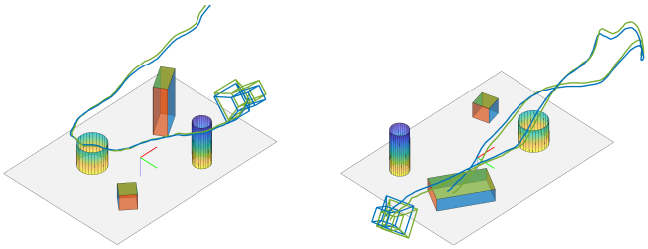
The errors in the estimated 6DOF transformation matrix were measured using only line-to-line, line-to-plane, and plane-to-plane correspondences at a time. Figure 7 shows the mean errors in the estimated rotation and translation for the $10^4$ trials at different noise levels. The minimum $e_R(\mathbf{R})$ and $e_t(\mathbf{t})$ errors are in line with related work [38].

The $P2\Pi$ 3-2-1 solver [38] registers six points to three planes assuming the following correspondences between points and planes

$$\Pi_1 \Leftarrow \{P_1, P_2, P_3\}, \quad \Pi_2 \Leftarrow \{P_4, P_5\}, \quad \Pi_3 \Leftarrow \{P_6\}. \quad (13)$$

Therefore, we also used six points on three planes for testing the least-square methods [2, 21], while avoiding any spatial distribution that might make them fail.

We also compare our method to the non-minimal solvers of Olsson and Eriksson [35] and Briales and Gonzalez-Jimenez [7]. They leverage point-to-point, point-to-line and point-to-plane correspondences to solve the registration. The latter use the so-called *effective number of correspondences*, $\hat{m} = 3m_{point} + 2m_L + 1m_\Pi$, to define the minimum number of matches for which a solution *may* exist. The minimum value is seven. Similarly to their tests on synthetic data, we used a problem size $\hat{m} = 10$ for testing their method. We used two points ($m_{point}$), one line ($m_L$) and two planes ($m_\Pi$). We used the same problem size ($\hat{m} = 10$) to test the solver of Olsson and Eriksson [35].



(a) (b)

Fig. 9: Results of our approach on two out of the ten real test data sets that we created. (a) ds3, (b) ds5. Each figure shows the parametric model, the ground truth initial position (green camera), the registered initial position (blue camera), the ground truth trajectory (green line), and the estimated trajectory (blue line).

The maximum level of noise we used to test our method on synthetic data was 0.4 units (see Figure 8). Given the size of our scenes is 50 units, a standard deviation ($\sigma$) of 0.4 is less than 1% of the size of our scenes. Although the noise levels we use to test the proposed algorithm seem low, a $\sigma$ of 0.4 units also represents 8% of the size of the cube, the smallest object in the scenes. The distortion that the scenes, respectively the parametric primitives, suffer with a $\sigma$ above 0.4 is significant (compare Figure 8a), since applying the distortion to each point separately amplifies the effect.

The results in Figure 7 show that our algorithm is outperformed by the least-squares methods of Arun *et al*. [2] and Horn [21], which rely on point-to-point correspondences. However, the main difference between our proposed method and the least-squares methods [2, 21] is the nature of the input data they work with. We focus on solving the model-to-scene registration problem for parametric models consisting of high-level primitives, such as planes made up of only a few vertices, and environments with little to no surface texture. We do not require dense reference data like meshes, as obtained from RGB-D reconstructions.

The results in Figure 7 also show that our algorithm achieves lower translation error than the methods of Olsson and Eriksson [35] and Briales and Gonzalez-Jimenez [7]. We used two points, one line and two planes to test these methods, nevertheless, we achieve better accuracy with fewer correspondences (only three).

### 6.2 Paper models

For experimenting with real scenes, we physically created ten data sets of cardboard models placed on a flat surface. Each data set consists of a cube, a box and two cylinders, for which the digital reference model was created on Matlab and exported as plain text.

We recorded a sequence for each scene with the Occipital Structure Core[4] RGB-D sensor and used InfiniTAMv3 to establish a pose track across frames. We obtained the parametric primitives by using the semantic segmentation of Stanescu *et al*. [42]. We further used the ArUco fiducial markers[5] to create a corresponding track treated as ground truth. In Figure 9, the camera tracks for two of the sequences are depicted. The blue track is reconstructed by registering the reference model to the segmented primitives, while the green track is the one estimated using the fiducial markers. Table 1 shows the minimum and mean errors $e_R(\mathbf{R})$ and $e_t(\mathbf{t})$ per data set. Overall, the errors match our observations from the simulations.

[4]Structure IO: https://structure.io/structure-core
[5]ArUco: https://docs.opencv.org/master/d5/dae/tutorial_aruco_detection.html
[6]Code on Github [7]: https://github.com/jbriales/CVPR17

| | Min | | MAE | |
|---|---|---|---|---|
| **Data set** | $e_R(\mathbf{R})$ | $e_t(\mathbf{t})$ | $e_R(\mathbf{R})$ | $e_t(\mathbf{t})$ |
| ds1 | 2.8440° | 0.3666 cm | 3.2184° | 0.6325 cm |
| ds2 | 0.9149° | 0.2878 cm | 1.4043° | 0.7021 cm |
| ds3 | 1.4388° | 0.0556 cm | 1.6645° | 0.3757 cm |
| ds4 | 2.7624° | 0.2045 cm | 3.1306° | 0.4631 cm |
| ds5 | 1.6083° | 0.1533 cm | 1.9092° | 0.3977 cm |
| ds6 | 0.9616° | 0.5443 cm | 1.2019° | 0.6915 cm |
| ds7 | 0.6672° | 0.6243 cm | 1.1250° | 0.8107 cm |
| ds8 | 0.4837° | 0.4385 cm | 0.8936° | 0.5876 cm |
| ds9 | 1.1876° | 0.7681 cm | 1.3608° | 0.9824 cm |
| ds10 | 1.3495° | 0.5998 cm | 1.4841° | 0.8017 cm |

TABLE 1: Minimum and mean absolute errors $e_R(\mathbf{R})$ and $e_t(\mathbf{t})$ of the ten real paper model data sets.

| | Runtime ($\mu s$) | |
|---|---|---|
| **Correspondences** | **Mean** | **Median** |
| plane-to-plane | 6.74 | 6.7 |
| line-to-plane | 6.91 | 6.9 |
| line-to-line | 6.64 | 6.6 |
| *P*2Π 3-2-1 [38] | 10.54 | 10.4 |
| point-to-point [2] | 3.29 | 3.3 |
| point-to-point [21] | 1.94 | 1.9 |

| | Runtime ($ms$) | |
|---|---|---|
| **Correspondences** | **Mean** | **Median** |
| point-to-*any* [35]* | 169.79 | 168.14 |
| point-to-*any* [7]* | 239.19 | 239.08 |

| | Segmentation ($ms$) | |
|---|---|---|
| **Iteration** | **Mean** | **Median** |
| init | 95.79 | 93.95 |
| update | 5.08 | 4.75 |

* We obtained the point-to-*any* [7, 35] runtimes using the implementation made available by the authors[6].

TABLE 2: Registration and segmentation runtime performance on synthetic and real-world data sets, respectively. Left and top right: Mean and median runtime for $10^4$ runs on simulation data in microseconds (left), respectively milliseconds for convex optimization based algorithms (top right). Bottom right: Mean and median point cloud segmentation runtime in milliseconds for the initial run and the subsequent frame-by-frame updates for the ten data sets based on paper models in Section 6.2.

### 6.3 Runtime

Our algorithm was implemented in C/C++, compiled with GNU gcc 8.3.0 and tested on a desktop computer with an Intel Core i7-7700HQ CPU at 2.80GHz and 16 GB RAM.

The left of Table 2 shows the runtime of all the simulation tests for our pose estimator and the methods of Ramalingam and Taguchi [38], Arun *et al*. [2] and Horn [21]. As expected, the overall runtime was similar for plane-to-plane, line-to-plane and line-to-line correspondences, because we always solve the same linear equation system built from three lines and one point.

Like the results in Figure 7, the results in Table 2 show that our algorithm is outperformed by the least-squares methods of Arun *et al*. [2] and Horn [21]. However, the runtimes in Table 2 for our approach not only consider the execution time for solving the equation system in Equation 7, but also the runtime to extract the three lines and one point. The input data for the least-squares methods [2, 21] are six points on three planes, while our method takes as input three parametric primitives, from which we extract three lines and one point.

At the bottom right of Table 2, the runtime of the iterative semantic segmentation algorithm [42] for the tests in Section 6.2 is given. The segmentation of the first backprojected point cloud takes most time, while incremental updates are incorporated quickly.

*Convex optimization algorithms*: We obtained the results in Figure 7 and the top right of Table 2 for the methods of Olsson and Eriksson [35], and Briales and Gonzalez-Jimenez [7], using the implementation made available by the latter, featuring a highly

optimized C++ convex optimization toolbox[7] underneath a Matlab user interface. Note that both methods are iterative. The average number of iterations to solve the registration for the $10^4$ simulation tests was 26 and 16, respectively.

The core of the work of Olsson and Eriksson [35], and Briales and Gonzalez-Jimenez [7], is a *constant size* semidefinite program (SDP). Their quadratic formulation allows them to solve the registration problem in almost constant runtime per iteration, irrespective of the number of correspondences. These methods are therefore highly suitable for problems with large numbers of correspondences. Note that the dominating order of operations to perform *per iteration* to solve an SDP is $max\{np^3, n^2p^2, n^3\}$, with $n$ and $p$ being the problem dimensions [6]. Therefore, the runtime of their solutions is at the order of $10^3$ times higher (per iteration) compared to all other methods tested. In this respect, our findings are consistent with the results originally discussed by Briales and Gonzalez-Jimenez [7]. They both note that the convex optimization problem is harder to solve as the number of correspondences approaches the minimal case and might give suboptimal results, therefore suggesting to use their methods only in non-minimal cases.

*Mobile implementation*: Implementing semantic segmentation on mobile devices from scratch would be relatively expensive. However, AR Foundation relies on scene segmentation (*i.e.*, plane detection) already optimized for the target hardware, with negligible extra cost. As a result, the overall runtime is in the order of a few *ms* in practice. This can be seen in the accompanying video material.

### 6.4 Model-based localization

In order to provide a first AR use case, we recorded two sequences. They show a typical training scenario and, in both examples, the parametric model consists of a set of planes and cylinders (green wireframe in the left of Figure 10). Similar to our paper models, the reference models for these scenes were created manually. The first scenario shows an office-like environment where the user must follow instructions. For example, such a setup may be used as part of a fire safety drill, where employees learn procedures to follow in case of an emergency. The second scenario was recorded in a typical industrial environment. It showcases a repair person referring to an interactive digital manual to learn basic maintenance tasks on a factory floor.

### 6.5 Indoor localization

So far, we have only presented experiments in relatively small scenes. In order to demonstrate the full power of our approach, we apply it to a room-scale environment, a kitchen, and explore the feasibility of using the method for indoor localization.

First, we reconstructed a 12 $m^3$ room using InfiniTAMv3 [23]. Second, we ran the semantic segmentation of Stanescu *et al*. [42] on the raw point cloud (see the reconstruction of the scene in Figure 11a) using the recorded pose track to create a semantic model of the kitchen. In Figure 11b, the parametric model created using the segmented primitives from the given reconstruction is shown. The parametric model contains 45 parametric primitives, and the compact world anchor contains 396 PPF items, which represent the scene in approximately 19 KB of uncompressed plain text. For comparison, we also created an Azure world anchor of the same scene with a Microsoft HoloLens. The size of the obtained Azure anchor was 4.6 MB of binary data.

[7]CVX Toolbox: http://cvxr.com

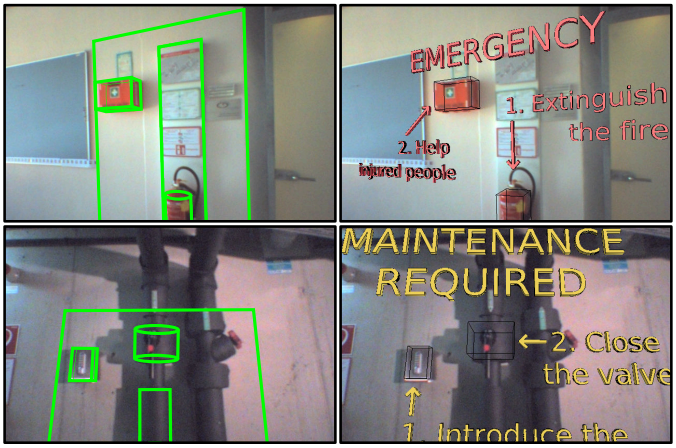



Fig. 10: Use case of our method for AR in real world scenarios. Left: Model reprojection into the image using the registered pose. Right: Instructions to complete a task presented in AR.

We recorded two new sequences starting from the west side and the east side of the scene, respectively. These sequences were further chopped into smaller parts with random starting points to demonstrate the re-initialization capabilities of our approach. Four examples are shown in Figure 14.

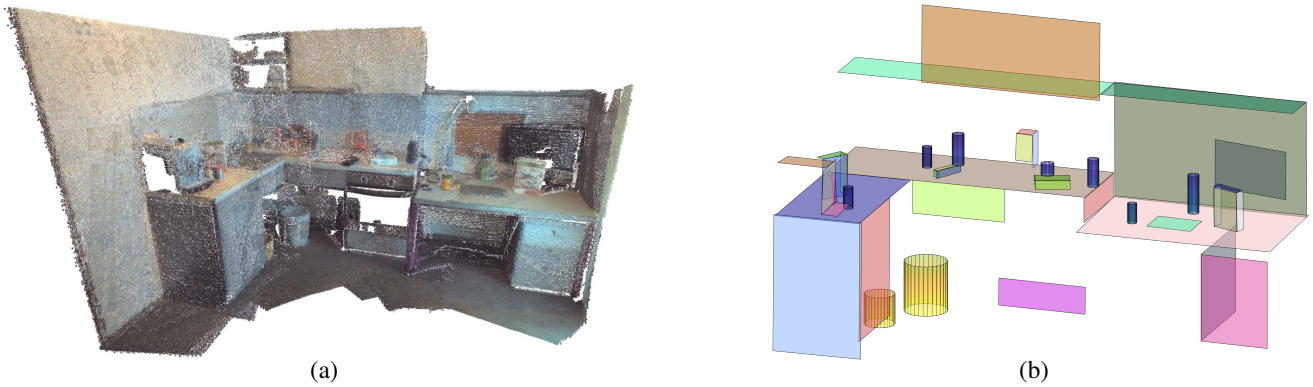
Since we need three correspondences (observed and matched primitives), most of the time, we need to take a few frames before getting a registration result (*i.e.*, our method does not allow instant localization from a single frame). The number of depth frames we need to segment before having a localization result depends on the size of the scene. For example, for the ten paper models data sets (see Section 6.2), we required an average of five frames in order to get an initial registration result. In the case of the room in Figure 11, we needed approximately 12 frames.

In Figure 12 our method is compared to Horn's method [21] as a representative for the algorithms tested in Sections 6.1 and 6.3. There is no strong difference in registration accuracy, however, there is a significant difference in *how many primitives* are required in practice to obtain a successful registration result. For the method of Horn [21] we need to infer four 3D points, which requires seven primitives in this scenario to avoid degenerate cases. In contrast, our algorithm requires only three primitives. More examples and comparisons are provided in the supplementary material.

### 6.6 Localization based on commercial SLAM

To demonstrate the interoperability of our method with a commercial SLAM solution, we created a mobile AR application based on AR Foundation. We created an iOS application for the iPad Pro which demonstrates how our method can be used to replace native spatial anchors for re-localization with our lightweight alternative.

The mobile app lets the user scan a scene and save a model for later localization inside the same scene. During the model creation, the user can place virtual objects on the detected planes and save them (see the hamburger on the fridge and the avatars on the desk in the top right of Figure 13). Later, the user can scan the scene again, and, after a successful localization of the device in the new scan of the scene, these virtual objects are re-spawned in the same place where the user put them during the model creation process (see the avatars in the bottom right of Figure 13).



(a) (b)

Fig. 11: Reconstruction of (a) a 12 $m^3$ room and (b) its corresponding parametric model created by segmenting the 3D scan.

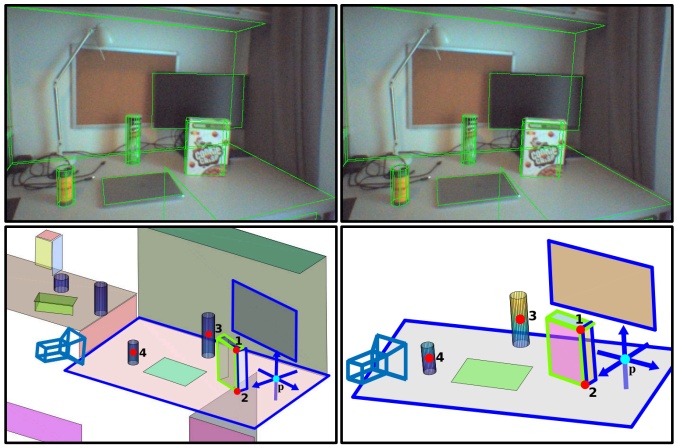


Fig. 12: Comparison of the registration accuracy of the method of Horn [21] (top left) to our method (top right). The results in the top row do not show any major visual discrepancies. The bottom row shows the parametric model (left) and the primitives detected in the scene (right). To enable the use of Horn's algorithm, we need at least four points (*i.e.*, the four red dots 1 to 4). To estimate the points 1 and 2, we use the planes with blue contour and also the planes with green contour (five planes in total), while points 3 and 4 are the geometric centers of the cylinders. In contrast, the blue arrows and the cyan dot (**p**) are the three lines and the point our algorithm requires. To obtain those, we compute the intersection of the planes with blue contour only. In summary, for such a scenario we need seven parametric primitives to estimate enough input data for Horn's algorithm, while we only need three parametric primitives for our algorithm.

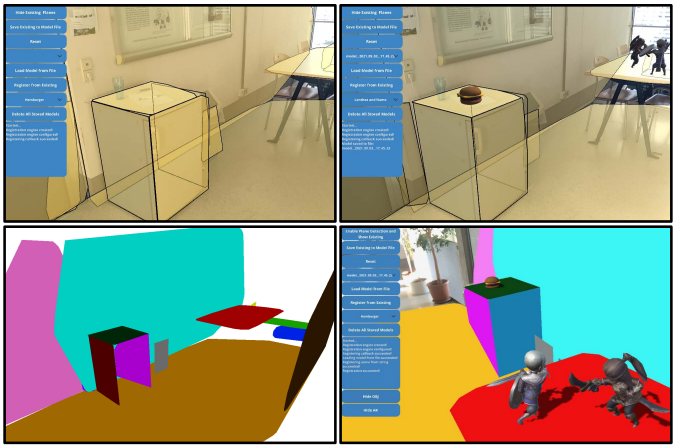


Fig. 13: Example of modeling and localization based on commercial SLAM. Top left: Scene scanning for model creation. Top right: Storing of parametric model and virtual objects. Bottom left: Screenshot (taken on a desktop PC) of the parametric model created and stored on the mobile device. Bottom right: Registered parametric model, and virtual objects, in a new scan of the scene.

The app saves three files in the model creation process: a JSON file containing our compact world anchor for localization, an OBJ file containing the 3D parametric model of the scene (see bottom left of Figure 13), and a JSON file with the position, orientation and scale of the virtual objects that were spawned in the scene during the modeling process.

We created parametric models of different scenes to test the re-localization capabilities of our compact world anchor and registration method using AR Foundation. During the model creation, we placed virtual objects on different planes in the observed scene to visually assess the quality of the registration. Four examples are shown in Figure 15. The fourth column shows the detected planes during the localization process and the registered parametric model. For example, the cabinet in the first row of Figure 15 and the mini fridge in the second row show that the result of the registration between the detected planes (yellow planes) and the parametric model (colorful planes) is visually very good.

## 7 Discussion

Based on our experimental results, we would like to revisit our current approach and discuss some aspects in detail.

*Tracking*: The proposed registration method is agnostic of the tracking system used to feed camera poses information. We used InfiniTAMv3 or AR Foundation, but any SLAM system can be used. As our work is computationally lightweight, it is well suited for running on all mobile platforms with built-in SLAM capability, such as ARKit or ARCore devices.

*Mapping and segmentation*: For the desktop experiments, we used InfiniTAMv3 for tracking, but the segmentation method of Stanescu *et al.* [42] was based on the raw depth data, simply backprojected into a common reference frame using the pose tracking. This decision (owing to the fact that performing a full integration of the two systems is not an easy task) relinquishes the superior reconstruction quality of InfiniTAMv3. The results presented in this paper should be interpreted with the fact in mind that the registration performs well *despite* a rather poor ad-hoc reconstruction stage.

*Correspondences*: With respect to the creation of correspondences for registration, the current main limitation in using our 3-vector PPF descriptor to find the correspondences is that the given geometric primitives must be finite. Although we designed the registration algorithm to work with infinite representations of

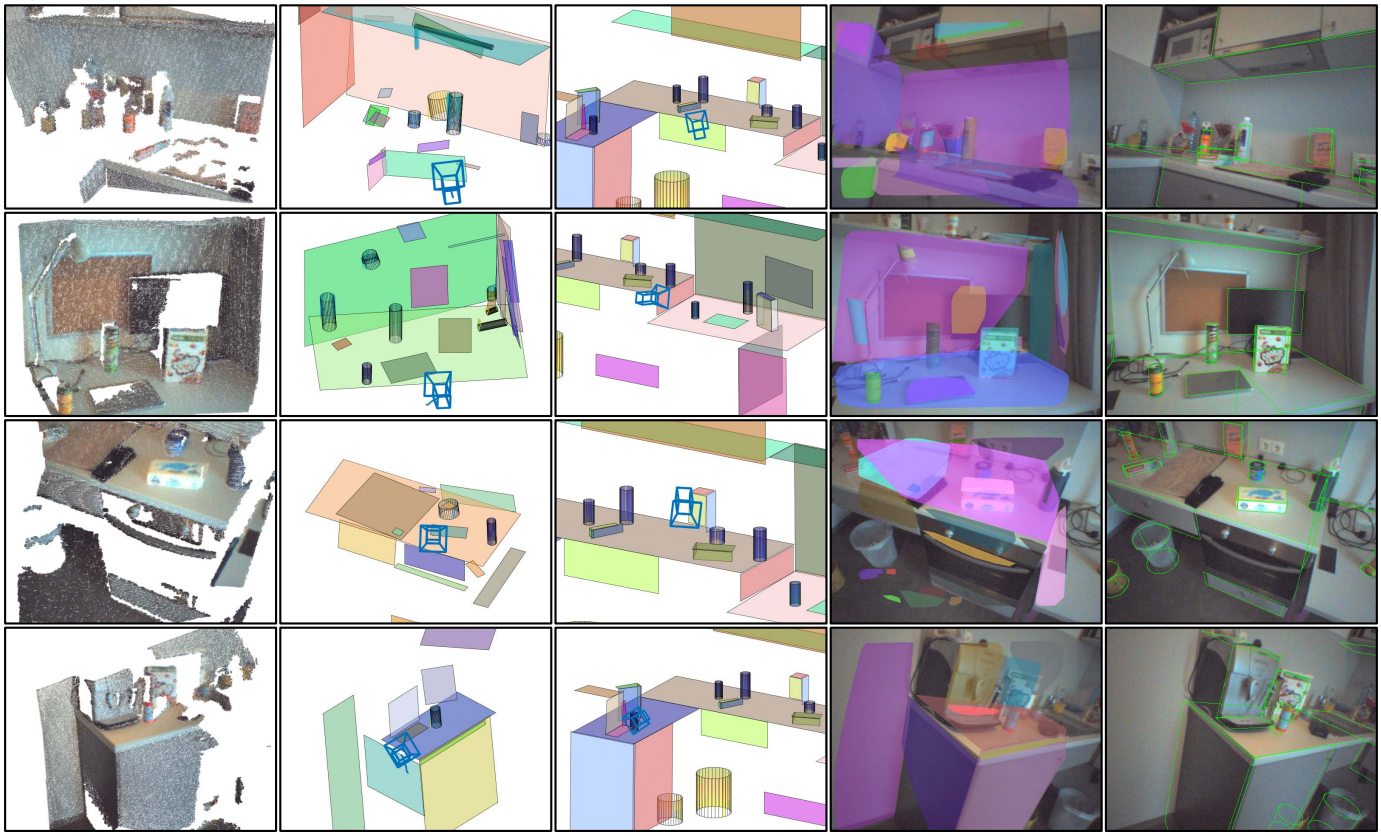


Fig. 14: Indoor localization examples for random initialization points using InfiniTAMv3 and semantic segmentation. First column: Backprojected point cloud of the first $n$ frames recorded before obtaining an initial registration. Second column: Segmented primitives from the scan in the first column, and camera pose. Third column: Estimated camera pose in the model coordinate system. Fourth column: Registered segmented primitives. Fifth column: Registered parametric model (green wireframe).

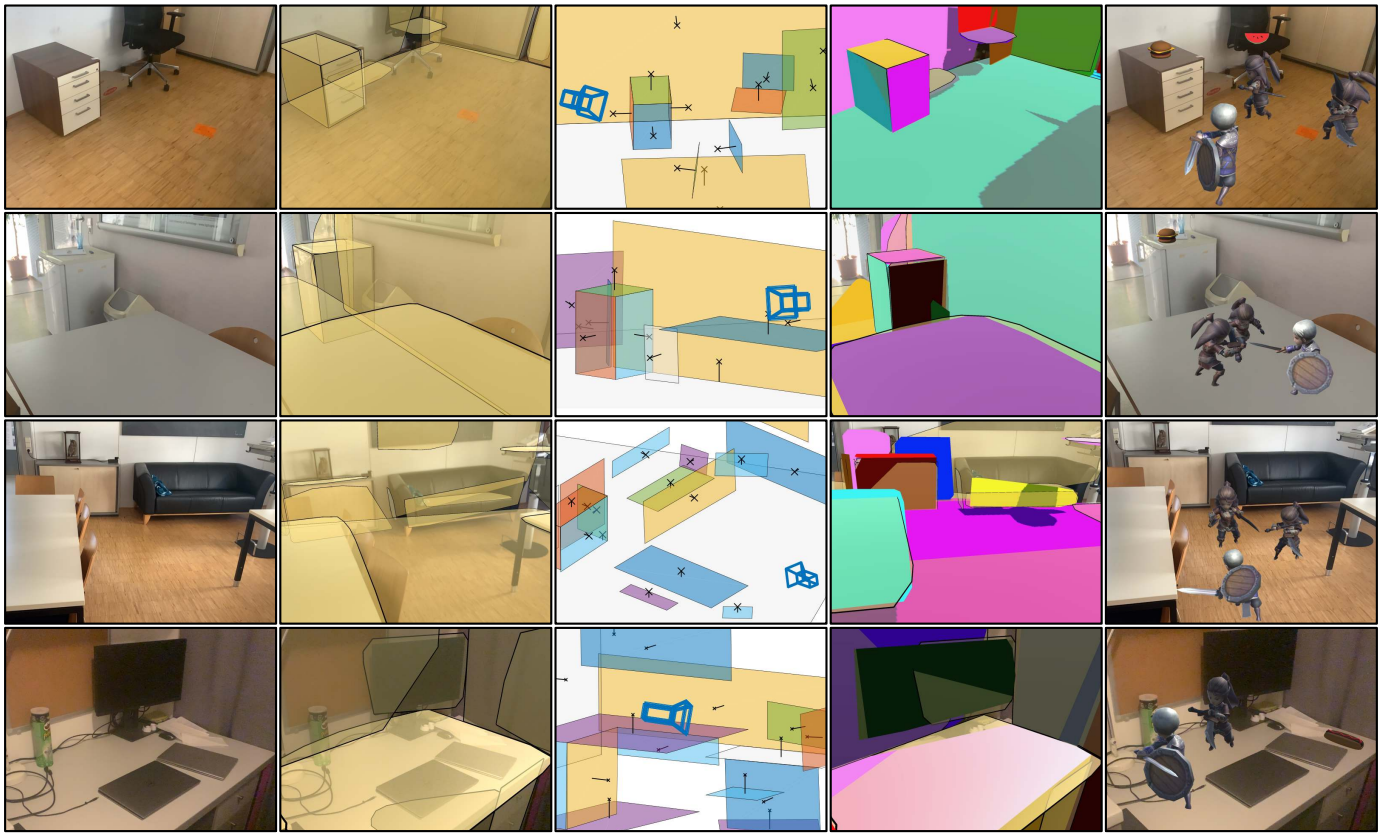


Fig. 15: Indoor localization examples for AR applications on mobile devices leveraging AR Foundation capabilities. First column: RGB image of the scene. Second column: Detected planes. Third column: Estimated camera pose in the model coordinate system. Fourth column: Registered parametric model planes to planes detected in the scene. Fifth column: Registered virtual objects.

lines and planes, we rely only on parametric information to find the correspondences. Consequently, infinite primitives would make it impossible to automatically detect the scene-to-model correspondences. However, we do not believe this is a major limitation in practice.

*Registration*: Although the experimental results show the proposed method has a good performance, we are not claiming this algorithm seeks the best numerical stability, the best convergence time, or outperforms previous solutions as a whole. Rather, we want to emphasize that we propose an *alternative* solution to the registration problem, which works well, but is much more frugal in its needs to store and process scene data.

*Sources of error*: The experimental results show low registration errors in simulations, but higher errors when working with real data sets. We explain this discrepancy with the inaccuracies confounding our real-world tests, where calibration inaccuracy, drift of the SLAM front-end, and noise in the segmentation accumulate. A detailed analysis of the magnitude of these various categories of error is left for future work.

*Limitations of commercial SLAM*: While AR Foundation has great device tracking and plane detection, it lacks the ability of detecting anything else than planes. In industrial environments, where cylindrical objects like pipes may be dominant, this could be a limitation. Moreover, device tracking quality with AR Foundation on mobile consumer hardware can be poor in scenarios where SLAM from dense depth maps (like InfiniTAM) works without problems. However, it is reasonable to expect that future generations of mobile hardware will overcome these issues.

## 8 Conclusion

We have presented a novel registration method for AR and indoor localization, which relies only on geometric information to register anchors consisting of geometry-based descriptors to parametric primitives in the scene. The experimental results show the effectiveness and robustness of our method in small and medium sized scenes.

While not intended to replace ICP or SoftAssign algorithms for registration, we believe our method can be a serious alternative to established methods for world anchors, as now used by commercial SLAM systems (*e.g.*, HoloLens), at a fraction of the storage cost and potentially better scalability.

Future work will concentrate on exploring the use of compact world anchors for scene detection in addition to scene registration. Because of the huge database sizes and high computational requirements for search, the detection of the user's current whereabouts from SLAM maps is currently deferred to cloud services such as the Microsoft Azure cloud. With our approach, scene detection can be carried out in a lightweight manner, but it is currently unknown which trade-off (*e.g.*, precision vs recall) is implied. We find this investigation to be a compelling topic for future research.

## Acknowledgments

The authors wish to thank Marco Stranner and Ana Stanescu for their support. This work was enabled by the Competence Center VRVis. VRVis is funded by BMVIT, BMWFW, Styria, SFG and Vienna Business Agency under the scope of COMET - Competence Centers for Excellent Technologies (854174) managed by FFG.

## References

[1] H. Abdellali, R. Frohlich, and Z. Kato. Robust absolute and relative pose estimation of a central camera system from 2D-3D line correspondences. In *IEEE Int. Conf. on Comp. Vision (ICCV)*, 2019.

[2] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 1987.

[3] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Nießner. Scan2CAD: Learning cad model alignment in RGB-D scans. In *IEEE Conf. on Comp. Vision and Patt. Rec. (CVPR)*, 2019.

[4] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1992.

[5] U. Bhattacharya and V. M. Govindu. Efficient and robust registration on the 3D special euclidean group. In *IEEE Int. Conf. on Comp. Vision (ICCV)*, 2019.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] J. Briales and J. Gonzalez-Jimenez. Convex global 3D registration with lagrangian duality. *IEEE Conf. on Comp. Vision and Patt. Rec. (CVPR)*, 2017.

[8] F. Camposeco, T. Sattler, and M. Pollefeys. Minimal solvers for generalized pose and scale estimation from two rays and one point. In *Europ. Conf. on Comp. Vision (ECCV)*, 2016.

[9] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An information-rich 3D model repository, 2015.

[10] H. H. Chen. Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 1991.

[11] P. David, D. Dementhon, R. Duraiswami, and H. Samet. SoftPOSIT: Simultaneous pose and correspondence determination. *Int. Journal of Comp. Vision (IJCV)*, 2004.

[12] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3D object recognition. *IEEE Conf. on Comp. Vision and Patt. Rec. (CVPR)*, 2010.

[13] 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*, 1981.

[14] W. Forstner and K. Khoshelham. Efficient and accurate registration of point clouds with plane to plane correspondences. In *IEEE Int. Conf. on Comp. Vision (ICCV)*, 2017.

[15] V. Gaudillière, G. Simon, and M.-O. Berger. Camera relocalization with ellipsoidal abstraction of objects. In *IEEE Int. Symp. on Mixed and Augm. Reality (ISMAR)*, 2019.

[16] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi. Real-time RGB-D camera relocalization. In *IEEE Int. Symp. on Mixed and Augm. Reality (ISMAR)*, 2013.

[17] R. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *Int. Journal of Comp. Vision*, 1994.

[18] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (DLS) method for PnP. In *2011 Int. Conf. on Comp. Vision*, 2011.

[19] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke. Registration with the point cloud library: A modular framework for aligning in 3-D. *IEEE Robotics & Automation Magazine*, 2015.

[20] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Comp. Vision, Graphics, and Image Processing*, 1989.

[21] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 1987.

[22] J. Hou, A. Dai, and M. Nießner. RevealNet: Seeing behind objects in RGB-D scans. In *IEEE Conf. on Comp. Vision and Patt. Rec. (CVPR)*, 2020.

[23] O. Kähler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Trans. on Visualization and Computer Graphics (TVCG)*, 2015.

[24] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *IEEE Int. Conf. on Comp. Vision (ICCV)*, 2015.

[25] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *IEEE Int. Symp. on Mixed and Augm. Reality*, 2007.

[26] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *IEEE Int. Symp. on Mixed and Augm. Reality*, 2009.

[27] L. Kneip, H. Li, and Y. Seo. UPnP: An optimal O(n) solution to the absolute pose problem with universal applicability. In *Europ. Conf. on Comp. Vision (ECCV)*, 2014.

[28] H. Li and R. Hartley. The 3D-3D registration problem revisited. *IEEE Int. Conf. on Comp. Vision (ICCV)*, 2007.

[29] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. GlobFit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph.*, 2011.

[30] K. Lianos, J. L. Schönberger, M. Pollefeys, and T. Sattler. VSO: Visual semantic odometry. In *Europ. Conf. on Comp. Vision*, 2018.

[31] M. I. A. Lourakis and G. Terzakis. Efficient absolute orientation revisited. *IEEE Int. Conf. on Intelligent Robots and Systems*, 2018.

[32] A. Mateus, S. Ramalingam, and P. Miraldo. Minimal solvers for 3D scan alignment with pairs of intersecting lines. *IEEE Conf. on Comp. Vision and Patt. Rec. (CVPR)*, 2020.

[33] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE Int. Symp. on Mixed and Augm. Reality (ISMAR)*, 2011.

[34] L. Nicholson, M. Milford, and N. Sunderhauf. QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 2018.

[35] C. Olsson and A. Eriksson. Solving quadratically constrained geometrical problems using lagrangian duality. *IEEE Int. Conf. on Patt. Rec. (ICPR)*, 2008.

[36] C. Olsson, F. Kahl, and M. Oskarsson. The registration problem revisited: Optimal solutions from points, lines and planes. In *IEEE Conf. on Comp. Vision and Patt. Rec. (CVPR)*, 2006.

[37] K. Pathak, A. Birk, N. Vaškevičius, and J. Poppinga. Fast registration based on noisy planes with unknown correspondences for 3-D mapping. *IEEE Trans. on Robotics*, 2010.

[38] S. Ramalingam and Y. Taguchi. A theory of minimal 3D point to 3D plane registration and its generalization. *Int. J. Comput. Vis.*, 2012.

[39] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009.

[40] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *IEEE Conf. on Comp. Vision and Patt. Rec. (CVPR)*, 2013.

[41] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *IEEE Conf. on Comp. Vision and Patt. Rec. (CVPR)*, 2013.

[42] A. Stanescu, P. Fleck, C. Arth, and D. Schmalstieg. Semantic segmentation of geometric primitives in dense 3D point clouds. In *IEEE Int. Symp. on Mixed and Augm. Reality*, 2018.

[43] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. gDLS: A scalable solution to the generalized pose and scale problem. In *Europ. Conf. on Comp. Vision (ECCV)*, 2014.

[44] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 1991.

[45] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. A minimal solution to the generalized pose-and-scale problem. In *IEEE Conf. on Comp. Vision and Patt. Rec. (CVPR)*, 2014.

[46] F. Wientapper, M. Schmitt, M. Fraissinet-Tachet, and A. Kuijper. A universal, closed-form approach for absolute pose problems. *Comp. Vision and Image Understanding*, 2018.

[47] H. Wuest, T. Engekle, F. Wientapper, F. Schmitt, and J. Keil. From cad to 3d tracking - enhancing & scaling model-based tracking for industrial appliances. In *IEEE Int. Symp. on Mixed and Augm. Reality (ISMAR)*, 2016.

[48] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song. Semantic slam based on object detection and improved octomap. *IEEE Access*, 2018.

[49] X. Zhang, Z. Zhang, Q. Yu, and J. Ou. Robust camera pose estimation from unknown or known line correspondences. *Applied optics*, 2012.

[50] L. Zhou, S. Wang, and M. Kaess. A fast and accurate solution for pose estimation from 3D correspondences. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020.

[51] Q.-Y. Zhou, J. Park, and V. Koltun. Fast global registration. In *Europ. Conf. on Comp. Vision (ECCV)*, 2016.


**Fernando Reyes-Aviles** is currently a Ph.D. student in Computer Science at Graz University of Technology, Austria. He graduated in B.Sc. Electronics Engineering (2015) from the Universidad Autonoma Metropolitana, Mexico. He received the M.Sc. in Computer Science Degree (2017) from the Universidad Autonoma Metropolitana, Mexico. His main interests are related to augmented reality and computer vision.

**Philipp Fleck** is currently working as a PhD Student at ICG at Graz University of Technology. He received his Masters degree from TU Graz in Computer Science with focus on Vision and Graphics. Philipp is working on AR tracking systems on mobiles under the supervision of Prof. Dieter Schmalstieg and Dr. Clemens Arth. Philipp worked at Magna Steyr in Graz as a Developer for several years. His research interests lie in SLAM, Tracking, AR, 3D Reconstruction and Image Recognition.

**Dieter Schmalstieg** is full professor and head of the Institute of Computer Graphics and Vision at Graz University of Technology, Austria. His current research interests are augmented reality, virtual reality, computer graphics, visualization and human-computer interaction. He received Dipl.-Ing. (1993), Dr. techn. (1997) and Habilitation (2001) from Vienna University of Technology. He received the START career award presented by the Austrian Science Fund (2002), the IEEE Virtual Reality technical achievement award (2012) and the IEEE ISMAR Career Impact Award (2020).

**Clemens Arth** is founder and CEO of AR4 GmbH, and he is technology consultant. He is also affiliated with Graz University of Technology. His main research interests are Computer Vision algorithms, Augmented Reality technology and Machine Learning. The current focus of his work is on accurate global localization of mobile devices for AR applications. He received Dipl.-Ing. (2004) and Dr. techn. (2008) degrees from Graz University of Technology. He is author and co-author of numerous peer-reviewed publications at ISMAR, VISAPP, CVPR, ACCV, ICPR and VR.