Unified Representation and Registration of Heterogeneous Sets of Geometric Primitives

Federico Nardi*, Bartolomeo Della Corte*, and Giorgio Grisetti,

***Abstract*—Registering models is an essential building block of many robotic applications. In case of 3D data, the models to be aligned usually consist of point clouds. In this work we propose a formalism to represent in a uniform manner scenes consisting of high-level geometric primitives, including lines and planes. Additionally, we derive both an iterative and a direct method to determine the transformation between heterogeneous scenes (solver). We analyzed the convergence behavior of this solver on synthetic data. Furthermore, we conducted comparative experiments on a full registration pipeline that operates on raw data, implemented on top of our solver. To this extent we used public benchmark datasets and we compared against state-of-the-art approaches. Finally, we provide an implementation of our solver together with scripts to ease the reproduction of the results presented in this work.**

***Index Terms*—SLAM, Localization.**

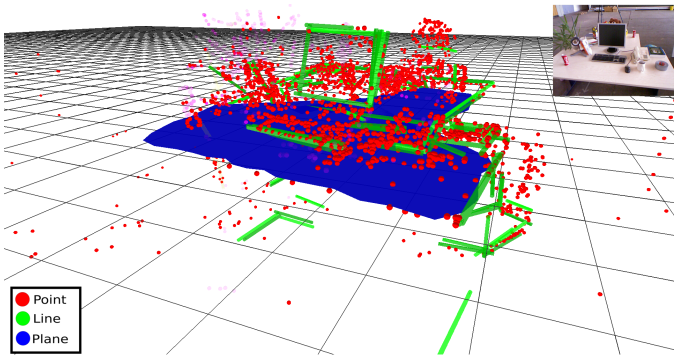


Fig. 1: Example reconstruction obtained by our approach from RGB-D raw data of TUM dataset (*fr2/desk*). This image shows how our approach is able to describe a complete indoor scene with a limited number of primitives.

I. Introduction

The majority of simultaneous localization and mapping (SLAM) algorithms consider simple (or low-level) geometric primitives in their formulation, *i.e.* points, and go under the name of *landmark-based* methods [1]. The spread of these methods is mainly due to the existence of well-known techniques for detecting [2]–[4], matching [5] and registering [6]–[9] such primitives. Despite their efficiency, these techniques have known drawbacks: detection may fail in textureless scenes, matching is hindered by the features low descriptiveness and registration may be unable to recover large rotations. In all the above mentioned situations, camera pose tracking is likely to yield poor results.

Man made environments are rich in structure, embedding higher-level features usually not exploited in standard techniques. Planar items such as walls, floors and tables cover a great portion of the environment. Similarly, the edges of these planar structures often form lines. In this paper we propose a unified representation for different types of primitives such as points, lines and planes. In this way, it is possible to devise a registration algorithm for aligning hybrid scenes in a single formulation. Moreover, we can define correspondences among items in the scenes that belong to different classes: besides enforcing that two planes or two points in the scenes are the same, we can also express constraints such as "a point lies on a line", or "a line lies on a plane".

Exploiting structure to describe a scene has clear advantages on the storage: describing a room as a set of walls and a floor requires far less memory than storing the corresponding point cloud. In turn, registration algorithms might gain in efficiency and display larger alignment capability coming from the higher descriptiveness of the primitives.

The major contributions of this work are summarized as follows:

- we propose a unified representation for a 3D scene consisting of the following geometric primitives: points, lines and planes. We refer to them as "matchables".
- we develop an algorithm to compute the most likely transformation between two scenes given a set of correspondences between matchables.

The derivations proposed in this paper capture in a uniform manner several Iterative Closest Point (ICP) variants [6], [8], [9]. Our system benefits from the structure when present, while it degrades to regular ICP in absence of structure. Furthermore, it provides compact models that can be used to reproduce the geometry of the scene. These results are confirmed by comparative experiments we conducted on publicly available benchmark datasets. Fig. 1 shows an example of a scene reconstructed with our method from real data. An open source implementation of our approach, along with the supplementary material, is available online[1].

Manuscript received: September 10, 2018 ; Revised: October 30, 2018 ; Accepted: December 24, 2018.

This paper was recommended for publication by Editor Cyrill Stachniss upon evaluation of the Associate Editor and Reviewers' comments.

* These two authors contributed equally to the work.

Federico, Bartolomeo and Giorgio are with Sapienza University of Rome, Department of Computer, Control, and Management Engineering Antonio Ruberti, Rome, Italy.{lastname}@diag.uniroma1.it.

Digital Object Identifier (DOI): see top of this page.

[1]Repository: http://srrg.gitlab.io/sa-sha.html

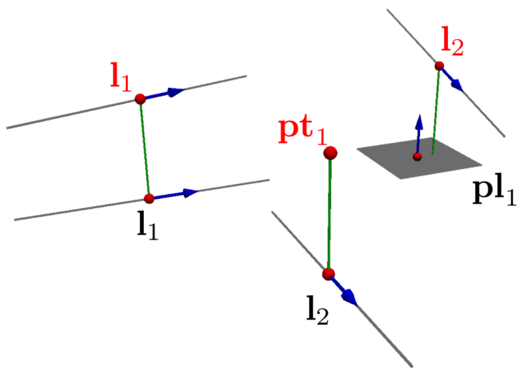


Fig. 2: A typical scenario addressable with the proposed representation: we want to register a moving scene (red) onto a fixed scene (black). The scenes are composed by points (**pt**), lines (**l**) and planes (**pl**). Green lines indicate the constraints between two geometric entities. The proposed representation allows to model both homogeneous (*e.g.* line-line) and heterogeneous (*e.g.*, point-plane) constraints.

II. Related Work

Point cloud registration is a core component of many robotic applications and several approaches to address it have been proposed. Among them, the most common and well known is the Iterative Closest Point (ICP) algorithm [6]. ICP works by iteratively searching for correspondences between points and minimizing the cumulative sum of distance between these pairs. One of the main drawback of the vanilla ICP approach is that, being point-based, it discards valuable information about the local surface shape.

Leveraging on this observation, Segal *et al.* [8] proposed Generalized ICP (GICP). This approach relies on the assumption that most range data is sampled from locally planar surfaces [7] and proposed a probabilistic variant of ICP by explicitly modeling the sensor noise. Despite their versatility, low level primitives lack of descriptiveness and do not support well robotic tasks. To overcome this limitation, Salas-Moreno *et al.* [10] proposed an object-based SLAM paradigm that offers both descriptive power and representation compression. The main drawback of this method is that it requires a prior knowledge about the models (chairs, tables, etc.) of objects that can be found in the scene. Olsson *et al.* [11] described a method to obtain a globally optimal solution from point-to-point, point-to-line, and point-to-plane correspondences using branch-and-bound. In contrast to this method our approach supports arbitrary constraints between primitives.

The straightforward benefit of using high-level features has been extensively exploited to improve the robustness of landmark-based registration approaches. Early works aim at improving the accuracy of camera tracking with edges [12], [13] and lines [14], [15]. These works propose different types of primitive representation: a point and a direction [12], [13]; Plücker coordinates [14] and endpoints [15].

Since the advent of RGB-D cameras, that provide per-pixel depth information, the interest on such high-level features increased. Choi *et al.* [16] made use of different types of edges, while line features combined with points are investigated in the works of Lu and Song [17] and Pumarola *et al.* [18]. Additionally, organized (and colored) point clouds returned by these sensors are particularly suited for 3D segmentation and structure detection [19], [20] and raised interest in registration methods that consider planes as higher-level features [21]–[24].

The methods presented so far use a different representation for each type of primitive, resulting in a distinct registration strategy for each of them. In fact, most of these methods have been designed as an extension of standard point-based ICP rather than being a holistic redesign that includes the original problem.

Our work is similar to Castellanos *et al.* [25], where the authors propose the symmetries and perturbation map (SPmap). In this model, a geometric feature is represented by combining a *perturbation vector* that expresses the uncertainty about its location and a *binding matrix* that accounts for its shape. Despite providing a powerful tool to represent any type of geometric entity, in [25] the authors limit themselves to line segments, while we explicitly implemented different primitives within our representation.

The unified representation presented in this paper has the common advantages of high-level feature approaches, such as compactness and robustness and it encapsulates different types of geometric primitives in a single type of data structure, namely a *matchable* (see Sec. III-A).

III. Generalized Scene Registration

Registration consists in finding the transform that better aligns two scenes through non-linear optimization of an objective function. As a consequence, the choice of this function has a strong impact on the final solution. In general, smooth objective functions are preferred, since they lead to an easier convergence.

In this work, we propose a formulation of the registration problem that includes not only point-point correspondences but any of the nine possible pairings between points, lines and planes (see Fig 2). This allows to exploit the higher descriptiveness of geometric primitives such as lines and planes for widening the objective function convergence basin and, consequently, having a more robust registration.

Of course, a straightforward approach is to define an *ad-hoc* error function for each type of correspondence (e.g., point-point, point-line, line-plane and so on). Then, for each of them, a different optimization has to be formulated. Our approach is to define a single parametrization for different geometric primitives. Therefore, once a primitive is represented as a unified entity, all the other computation modules are agnostic of the primitive's type.

In the remainder of this section, we first introduce our representation of matchables. Subsequently, we define how to apply an isometry to a scene. With this basic operation defined, we show how to define an error function between two matchables that are connected by a constraint. Finally, we propose both an iterative and a direct implementation of a solver capable of finding the optimal transformation between two scenes, given a set of known correspondences.

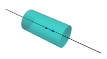
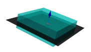
type	$\mathbf{p_m}$	$\mathbf{R_m}$	$\mathbf{\Omega_m}$	Shape of $\mathbf{\Omega_m}$
point	$\mathbf{p}$	$\mathbf{R}$	$\mathtt{diag}(1,1,1)$	
line	$\mathbf{p_l}$	$\mathbf{R_l}$	$\mathtt{diag}(0,1,1)$	
plane	$\mathbf{p}_\pi$	$\mathbf{R}_\pi$	$\mathtt{diag}(1,0,0)$	

TABLE I: Matchables table. The shape of $\mathbf{\Omega_m}$ discriminates the type of primitive represented by the matchable. The confidence ellipsoid obtained from $\mathbf{\Omega_m}$ is a sphere if the matchable is a point. If the primitive is a line or a plane the confidence ellipsoid degenerates respectively to a cylinder or to two parallel planes.

A. Representation

Our representation stems from the fact that points, lines and planes can be defined in terms of *degenerate quadrics*. In general, a quadric can be described by the following equation:

$$(\mathbf{x}-\mathbf{p})^\top \mathbf{A}(\mathbf{x}-\mathbf{p}) = 0. \tag{1}$$

Here, $\mathbf{p} \in \Re^3$ is the origin of the quadric and $\mathbf{A}$ a symmetric matrix. A point $\mathbf{x} \in \Re^3$ lies on the quadric if it satisfies Eq. (1). $\mathbf{A}$ can be factorized as $\mathbf{A} = \mathbf{R\Omega R}^\top$, where the columns of $\mathbf{R}$ are the axes of the quadric, and $\mathbf{\Omega}$ is a diagonal matrix containing the eigenvalues of $\mathbf{A}$. The shape of the quadric is entirely determined by its eigenvalues. In the remainder, we refer to the first column of $\mathbf{R}$ as the *direction* $\mathbf{d}$ of a primitive.

Points can be represented as spheres with a null radius, thus all eigenvalues should be equal and positive: $\mathbf{\Omega} = \mathrm{diag}(1,1,1)$. Lines can be represented as cylinders having null radius, thus, assuming the cylinder axis is parallel to the direction $\mathbf{d}$, the corresponding omega is $\mathbf{\Omega} = \mathrm{diag}(0,1,1)$. Finally, planes can be represented through quadric as two matching planes. In this case, being the plane normal parallel to $\mathbf{d}$, $\mathbf{\Omega} = \mathrm{diag}(1,0,0)$.

The reader might notice that the $\mathbf{\Omega}$ used to represent the above primitives can be scaled arbitrarily by any positive number. Setting the non-null eigenvalues to 1 has however the effect of turning the value of the left hand side $\|\mathbf{x}-\mathbf{p}\|^2_{\mathbf{A}}$ of Eq. (1) to the squared euclidean distance between $\mathbf{x}$ and the closest point on the quadric. Tab. I summarizes the parameters used for each type of primitive. In our system we call such quadrics *matchables*. For a matchable $\mathbf{m}$ we store: the origin $\mathbf{p_m}$, the rotation matrix $\mathbf{R_m}$ and the eigenvalues matrix $\mathbf{\Omega_m}$:

$$\mathbf{m} : \Big\langle \mathbf{p_m}, \quad \mathbf{R_m}, \quad \mathbf{\Omega_m} \Big\rangle. \tag{2}$$

Notice that the proposed representation is not unique as, for example, in 3D space a plane has 3 degrees of freedom and a line has 4. For lines and planes the matchable origin $\mathbf{p_m}$ is computed from their support (line segments and plane patches) and can be chosen arbitrarily. We provide implementation details for this operation in Sec. IV-A.

		$\mathbf{m}'$		
		point	**line**	**plane**
$\mathbf{m}$	**point**	$\mathbf{\Omega}_\mathrm{p} = \mathbf{I}$ $\mathbf{\Omega}_\mathrm{d} = \mathbf{0}$ $\Omega_\mathrm{o} = 0$	$\mathbf{\Omega}_\mathrm{p} = \mathbf{\Omega_{m'}}$ $\mathbf{\Omega}_\mathrm{d} = \mathbf{0}$ $\Omega_\mathrm{o} = 0$	$\mathbf{\Omega}_\mathrm{p} = \mathbf{\Omega_{m'}}$ $\mathbf{\Omega}_\mathrm{d} = \mathbf{0}$ $\Omega_\mathrm{o} = 0$
	line	$\mathbf{\Omega}_\mathrm{p} = \mathbf{\Omega_m}$ $\mathbf{\Omega}_\mathrm{d} = \mathbf{0}$ $\Omega_\mathrm{o} = 0$	$\mathbf{\Omega}_\mathrm{p} = \mathbf{\Omega_{m'}}$ $\mathbf{\Omega}_\mathrm{d} = \mathbf{I}$ $\Omega_\mathrm{o} = 0$	$\mathbf{\Omega}_\mathrm{p} = \mathbf{\Omega_{m'}}$ $\mathbf{\Omega}_\mathrm{d} = \mathbf{0}$ $\Omega_\mathrm{o} = 1$
	plane	$\mathbf{\Omega}_\mathrm{p} = \mathbf{\Omega_m}$ $\mathbf{\Omega}_\mathrm{d} = \mathbf{0}$ $\Omega_\mathrm{o} = 0$	$\mathbf{\Omega}_\mathrm{p} = \mathbf{\Omega_m}$ $\mathbf{\Omega}_\mathrm{d} = \mathbf{0}$ $\Omega_\mathrm{o} = 1$	$\mathbf{\Omega}_\mathrm{p} = \mathbf{\Omega_{m'}}$ $\mathbf{\Omega}_\mathrm{d} = \mathbf{I}$ $\Omega_\mathrm{o} = 0$

TABLE II: Information Matrix $\mathbf{\Omega}(\mathbf{m},\mathbf{m}')$ for each possible pair of matchables.

B. Transformation

Let $\mathbf{X} = [\mathbf{R_x}|\mathbf{t_x}] \in SE(3)$ be a transformation. The operation of applying a transformation to a matchable $\mathbf{m}$ results in a new matchable $\mathbf{m}' = \mathbf{X}\cdot\mathbf{m}$ with the following parameters:

$$\mathbf{m}' = \begin{pmatrix} \mathbf{R_x p_m} + \mathbf{t_x} \\ \mathbf{R_x R_m} \\ \mathbf{\Omega_m} \end{pmatrix}. \tag{3}$$

Here we parametrized $\mathbf{X}$ as a rotation matrix $\mathbf{R_x} \in SO(3)$ and a translation vector $\mathbf{t_x} \in \Re^3$. Intuitively, applying an isometry to a matchable results in a transformation of the Euclidean component $\mathbf{p_m}$ and in a rotation of the matrix $\mathbf{R_m}$, leaving unchanged $\mathbf{\Omega_m}$.

C. Distance

In this section we present a metric to evaluate a distance $e(\mathbf{m},\mathbf{m}')$ between a pair of matchables $\mathbf{m}$ and $\mathbf{m}'$. If $\mathbf{m}$ is a point and $\mathbf{m}'$ is any primitive Eq. (1) allows us to compute the squared distance between a point and the quadric in $\mathbf{m}'$ as:

$$e(\mathbf{m},\mathbf{m}') = (\mathbf{p_m} - \mathbf{p_{m'}})^\top \mathbf{A_{m'}}(\mathbf{p_m} - \mathbf{p_{m'}}) \tag{4}$$

To compute the distance between two planes (or, equivalently, two lines) we need to consider in the difference their directions $\mathbf{d_m}$ and $\mathbf{d_{m'}}$. Adding to Eq. (4) a quadratic term $\|\mathbf{d_m}-\mathbf{d_{m'}}\|^2$ that captures the difference in directions results in a metric that is zero when the two planes or lines are the same:

$$e(\mathbf{m},\mathbf{m}') = \|\mathbf{p_m}-\mathbf{p_{m'}}\|^2_{\mathbf{A_{m'}}} + \|\mathbf{d_m}-\mathbf{d_{m'}}\|^2 \tag{5}$$

While, for the distance between a line and a plane, we consider that a line lies on a plane if:

- the point of the line lies on the plane;
- their direction vectors are orthogonal.

The first requirement is satisfied when Eq. (4) is 0. To capture the orthogonality constraint expressed by the second requirement we add to the metric an additional semi-positive term that is zero when the two directions are orthogonal:

$$e(\mathbf{m},\mathbf{m}') = \|\mathbf{p_m}-\mathbf{p_{m'}}\|^2_{\mathbf{A_{m'}}} + \left\|\mathbf{d}_\mathbf{m}^\top \mathbf{d_{m'}}\right\|^2 = 0 \tag{6}$$

Having considered how to measure the distance between all possible combinations of matchables, we want to have a unique formulation for all of them. This will be useful in the remainder of this paper, as we will derive the registration procedure only once for all possible types of constraint. To this end, we rewrite the difference between two matchables as the following 7D vector:

$$\mathbf{e}(\mathbf{m},\mathbf{m}') = \begin{pmatrix} \mathbf{e}_\mathrm{p} \\ \mathbf{e}_\mathrm{d} \\ e_\mathrm{o} \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{\mathbf{m}'}^\top (\mathbf{p}_\mathbf{m} - \mathbf{p}_{\mathbf{m}'}) \\ \mathbf{d}_\mathbf{m} - \mathbf{d}_{\mathbf{m}'} \\ \mathbf{d}_\mathbf{m}^\top \mathbf{d}_{\mathbf{m}'} \end{pmatrix} \tag{7}$$

A distance $e(\mathbf{m},\mathbf{m}')$ between matchables is a non-negative scalar computed from the difference vector. If the distance is zero, the constraint between the two matchables is satisfied. To compute the distance, we employ adapted Ω-norm (*i.e.*, $\|\mathbf{v}\|_{\mathbf{\Omega}} = \mathbf{v}^\top \mathbf{\Omega} \mathbf{v}$) to the difference vector:

$$\begin{aligned} e(\mathbf{m},\mathbf{m}') &= \|\mathbf{e}(\mathbf{m},\mathbf{m}')\|^2_{\mathbf{\Omega}(\mathbf{m},\mathbf{m}')} \\ &= \mathbf{e}(\mathbf{m},\mathbf{m}')^\top \mathbf{\Omega}(\mathbf{m},\mathbf{m}') \mathbf{e}(\mathbf{m},\mathbf{m}'). \end{aligned} \tag{8}$$

The information matrix $\mathbf{\Omega}(\mathbf{m},\mathbf{m}') \in \Re^{7\times 7}$ activates the appropriate components of the difference vector during the minimization, based on the type of constraint, according to Tab. II. In particular, we enforce the following block diagonal structure for $\mathbf{\Omega}(\mathbf{m},\mathbf{m}')$:

$$\mathbf{\Omega}(\mathbf{m},\mathbf{m}') = \begin{pmatrix} \mathbf{\Omega}_\mathrm{p} & 0 & 0 \\ 0 & \mathbf{\Omega}_\mathrm{d} & 0 \\ 0 & 0 & \Omega_\mathrm{o} \end{pmatrix} \tag{9}$$

With this formulation, the generic distance between two matchables is computed as

$$\begin{aligned} e(\mathbf{m},\mathbf{m}') &= \|\mathbf{e}(\mathbf{m},\mathbf{m}')\|^2_{\mathbf{\Omega}(\mathbf{m},\mathbf{m}')} \\ &= \|\mathbf{e}_\mathrm{p}\|^2_{\mathbf{\Omega}_\mathrm{p}} + \|\mathbf{e}_\mathrm{d}\|^2_{\mathbf{\Omega}_\mathrm{d}} + \|e_\mathrm{o}\|^2_{\Omega_\mathrm{o}} \end{aligned} \tag{10}$$

Consequently, the lower $e(\mathbf{m},\mathbf{m}')$ is, the "closer" the two primitives are.

It follows from its definition that, in general, this distance does not satisfy the symmetry property. In Sec. III of the supplementary material we report experimental results to analyze its symmetric version. Additionally, since the value of $e(\mathbf{m},\mathbf{m}')$ depends on the matchable origins in Sec. II of the supplementary material we evaluate the impact of this choice on the estimation accuracy.

D. Registration

Having defined how to transform the matchables, and a way to compute how "far" is a constraint from being satisfied, we seek for the optimal transformation $\mathbf{X}^*$ that better alignes two scenes:

$$\begin{aligned} \mathbf{X}^* &= \underset{\mathbf{X}}{\operatorname{argmin}} \sum_k e(\mathbf{X}\mathbf{m}_k, \mathbf{m}'_k) \\ &= \underset{\mathbf{X}}{\operatorname{argmin}} \sum_k \left\|\mathbf{e}(\mathbf{X}\mathbf{m}_k, \mathbf{m}'_k)^\top\right\|^2_{\mathbf{\Omega}(\mathbf{m}_k,\mathbf{m}'_k)} \end{aligned} \tag{11}$$

for each pair of matchables $\langle \mathbf{m}_k, \mathbf{m}'_k \rangle_{1:K}$ between the "moving" and the "fixed" scene.

A standard way to minimize the above equation is through Gauss-Newton iterative optimization. In this paper, we propose two different schemes for the minimization problem: a non-linear iterative solver and a direct solver. The non-linear solution is obtained by iteratively performing the optimization on a local parameterization of the perturbations. The direct solution does not require an initial guess of the transform, and finds the minimum in just one iteration. It is obtained by relaxing the constraint on the rotation matrix of the transform, and then recovering the orthonormality of the solution through Singular Value Decomposition (SVD).

In the remainder we shortly overview the Gauss-Newton algorithm for state spaces that live in $SE(3)$. Subsequently, we specialize our solution to derive an iterative and a direct solver.

1) Gauss-Newton Algorithm: We define our state space as a transformation matrix $\mathbf{X}$. We seek for the best $\mathbf{X}^*$ that satisfies all the matchable constraints. Let $\mathbf{\Delta x} \in \Re^n$ be a perturbation vector for a transform, and let $\boxplus$ be an operator that applies the perturbation to a transform as:

$$\mathbf{X}' = \mathbf{X} \boxplus \mathbf{\Delta x}, \tag{12}$$

where $\mathbf{X}'$ is the transformation obtained from $\mathbf{X}$ applying the perturbation $\mathbf{\Delta x}$. We report in Alg. 1 the standard Gauss-Newton minimization procedure for Eq. (11). The error func-

Algorithm 1: Gauss-Newton Algorithm.

```
   Data: current configuration $\breve{\mathbf{X}}$.
   Result: best configuration $\breve{\mathbf{X}}^*$.
1  $\mathbf{H} \leftarrow 0, \mathbf{b} \leftarrow 0$
2  foreach $\mathbf{e}_k$ do
3      $\mathbf{e}_k \leftarrow \mathbf{e}(\breve{\mathbf{X}}\mathbf{m}_k, \mathbf{m}'_k)$   /* evaluate error */
4      $\mathbf{J}_k \leftarrow \begin{cases} \left.\frac{\partial \mathbf{e}((\breve{\mathbf{X}} \boxplus \mathbf{\Delta x})\mathbf{m}_k, \mathbf{m}'_k)}{\partial \mathbf{\Delta x}}\right|_{\mathbf{\Delta x}=\mathbf{0}} & \text{if iterative} \\ \left.\frac{\partial \mathbf{e}(\mathbf{X}\mathbf{m}_k, \mathbf{m}'_k)}{\partial \mathbf{X}}\right|_{\mathbf{X}=\breve{\mathbf{X}}} & \text{if direct} \end{cases}$
5      $\mathbf{H} += \mathbf{J}_k^\top \mathbf{\Omega}_k \mathbf{J}_k$
6      $\mathbf{b} += \mathbf{J}_k^\top \mathbf{\Omega}_k \mathbf{e}_k$
7  end
8  $\mathbf{\Delta x} \leftarrow -\mathbf{H}^{-1}\mathbf{b}$
9  $\breve{\mathbf{X}} \leftarrow \breve{\mathbf{X}} \boxplus \mathbf{\Delta x}$
```

tion $\mathbf{e}(\mathbf{X}\mathbf{m}_k, \mathbf{m}'_k)$ is defined as the vector difference between $\mathbf{m}_k$, which is a primitive in the moving scene to which we apply the current transformation and the fixed item $\mathbf{m}'_k$. The $\mathbf{\Omega}_k = \mathbf{\Omega}(\mathbf{m}_k, \mathbf{m}'_k)$ computed to evaluate the scalar error term is defined according to Tab. II.

In the next, we specify for both cases, *iterative* and *direct*, the specific representation of the state space $\mathbf{X}$, the perturbation vector $\mathbf{\Delta x}$ and the $\boxplus$ operator to perform the update.

2) Iterative Solver: We represent the perturbation vector $\mathbf{\Delta x} \in \Re^6$ as the following vector:

$$\mathbf{\Delta x} := (\underbrace{\Delta x \; \Delta y \; \Delta z}_{\mathbf{\Delta t}} \underbrace{\Delta\alpha_x \; \Delta\alpha_y \; \Delta\alpha_z}_{\mathbf{\Delta}\alpha})^\top. \tag{13}$$

Here $\mathbf{\Delta t}$ is the translational part, while $\mathbf{\Delta}\alpha$ embeds the rotations around the x, y and z axes. To apply this perturbation to $\mathbf{X}$ we first convert $\mathbf{\Delta x}$ into a rotation matrix and a translation

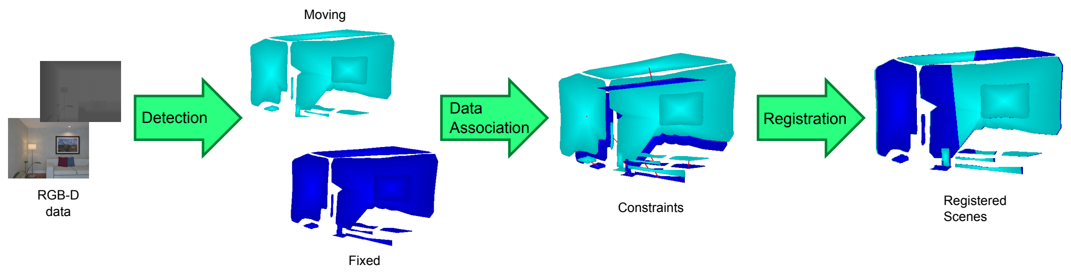


Fig. 3: Front-end. When a new pair of images is available, we extract a set of matchables from raw data using the strategy outlined in Sec. IV-A, here we show only planes for more clarity. Subsequently, we find corresponding matchables (here, linked by a red line) between the newly generated scene and the "fixed" scene with the methodology in Sec. IV-B. Finally, a minimization is conducted to find the transform that best aligns the two scenes according to Sec. III-D.

vector through the v2t function, and then we multiply the homogeneous transforms as follows:

$$\begin{aligned} \mathbf{\Delta X} &= \mathrm{v2t}(\mathbf{\Delta x}) \quad (14)\\ &= \begin{pmatrix} \mathbf{R_x}(\Delta\alpha_x)\mathbf{R_y}(\Delta\alpha_y)\mathbf{R_z}(\Delta\alpha_z) & \mathbf{\Delta t} \\ \mathbf{0} & 1 \end{pmatrix} \\ \mathbf{X} \boxplus \mathbf{\Delta x} &:= \mathbf{\Delta X} \cdot \mathbf{X} \quad (15) \end{aligned}$$

We report the derivation of the Jacobian matrix $\mathbf{J}_k$ in Sec. I-A of the supplementary material.

3) Direct Solver: For the direct solution of the Gauss-Newton algorithm, we consider a perturbation $\mathbf{\Delta x} \in \Re^{12}$ composed as follows:

$$\mathbf{\Delta x} = \left(\mathbf{\Delta t}^\top \; \mathbf{\Delta r}_1^\top \; \mathbf{\Delta r}_2^\top \; \mathbf{\Delta r}_3^\top \right)^\top \quad (16)$$

where $\mathbf{\Delta r}_i$ stands for the i^{th} column of a rotation matrix $\mathbf{\Delta R}$:

$$\mathbf{\Delta R} = (\mathbf{\Delta r}_1 \; \mathbf{\Delta r}_2 \; \mathbf{\Delta r}_3)\,. \quad (17)$$

We define the $\boxplus$ operator as:

$$\mathbf{X} \boxplus \mathbf{\Delta x} := \begin{pmatrix} \mathbf{R} + \mathbf{\Delta R} & \mathbf{t} + \mathbf{\Delta t} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (18)$$

Notice that during the solution we do not enforce the orthonormality of the rotation matrices involved. To lessen this problem, after applying the perturbation, we recondition the rotation matrix of $\mathbf{X}$ through SVD decomposition as follows:

$$\mathbf{USV} = \mathbf{R} \quad (19)$$
$$\mathbf{R}' = \mathbf{UV}. \quad (20)$$

The above procedure provides us with the "closest" rotation matrix to the original $\mathbf{R}$. If $\mathbf{S}$ is close to the identity, the SVD approximation is good, otherwise the direct solver provides a suboptimal solution. This typically occurs when several outliers are present in the constraints. The derivation of the Jacobian matrix $\mathbf{J}_k$ is reported in Sec. I-B of the supplementary material.

The direct version of the solver can be used to retrieve a candidate transformation when used within a RANSAC registration schema that does not require an initial guess. An alternative is to consider a minimal set of correspondences and use polynomial solvers based on Gröbner basis [26] or EPnP-style distance constraints formulation [27].

IV. Front-End

To validate the representation and the solver proposed in Sec. III on real data we developed a front-end based on RGB-D data. The processing pipeline, as shown in Fig. 3, is divided in three steps: detection, data association and registration.

A. Detecting Matchables from RGB-D data

Detection consists in extracting patterns from sensor output that correspond to distinguishable elements of the environment. Arguably, the most common technique is to extract corners from intensity images to obtain interest points. At the same time, most man-made objects are made of flat surfaces, this results in edges in the image that form straight lines. In a textureless scene, where a point feature detector is likely to fail, there are still good chances to detect lines.

Image based feature detectors do not account for the geometry of the scene and rely solely on the intensity channel. For this reason, their main limitation is a performance drop in detection under varying or severe lighting conditions. On the other hand, plane detection from range data depends on the 3D structure of the scene and is immune to light changes. However, plane detection depends on the estimation of geometric features (*i.e.*, surface normals and curvature) which may be severely limited by noisy and non-uniform data returned by range sensors.

Since our representation comprehends all these geometric entities, we can combine different detection techniques to compensate for their singular shortcomings. In the remainder we present the details on the detection of these primitives in our system.

1) Points and Lines: Points and lines are extracted in image coordinates using the intensity channel. In particular we use the FAST detector [28] to find salient pixels in the image, and we add to the scene the corresponding 3D point retrieved from the depth channel. Similarly, we use the Line Segment Detector [29] to find lines in the intensity image. For each line we consider the 3D points at the extrema to back-project it and we set the origin $\mathbf{p_m}$ as their centroid. To lessen the effect of spurious detections, we reject all lines that are too short or do not have sufficient support from the depth data. As an additional cue for points and lines matching, we label each point with a BRIEF descriptor [30] computed at the keypoint

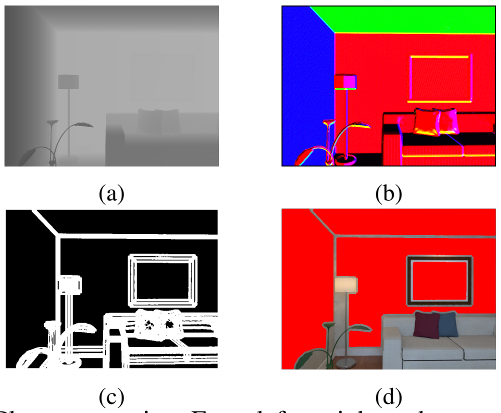


Fig. 4: Planes extraction. From left to right and top to bottom: (a) input depth image, (b) estimated surface normals, (c) connected regions (in black), (d) detected planes (in red).

location. Similarly we add to each line matchable a descriptor computed according to [31].

2) Planes: We first generate an organized 3D point cloud from the depth image and compute the surface normals for each point according to [32]. Subsequently, we mark each point characterized by a significant local variation of depth or normal. After this procedure, each pixel is classified as belonging to a continuous region or not and we recover the connected regions by visiting the image according to an 8-neighbor connectivity (see Fig. 4). For each of these regions that has a minimum number of points, we fit a plane. If the residual error of the fitted plane is below a threshold, we add a new plane to the scene. We repeat this procedure until no planes can be generated. The plane origin $\mathbf{p_m}$ is set as the centroid of the detected planar patch.

B. Data Association

The set of matchables detected in the current RGB-D frame constitutes the moving scene $\mathcal{S} = \{\mathbf{m}_{1:N}\}$. While, the set of matchables previously detected and registered in a global reference frame forms the fixed scene $\mathcal{S}' = \{\mathbf{m}'_{1:N'}\}$. The goal of the *data association* module is to find corresponding elements between the two sets. Then, each correspondence will result in a constraint for the solver.

Given the formalism presented in Sec. III-A, we are able to find two types of constraints: homogeneous and heterogeneous. A homogeneous constraint arises when we re-observe the same matchable, *e.g.*, a point in the moving scene *matches* a point in the fixed scene. While, a heterogeneous constraint derives from a correspondence between primitives of different types, such as a line in the moving scene *lies* on a plane in the fixed scene. We perform the constraints search in two steps, *i.e.* we first look for matchables of the same type and then we perform a hybrid match search.

1) Homogeneous Constraints: We designed a data association schema based on a Nearest Neighbor Search. Given the fixed scene $\mathcal{S}'$, the moving scene $\mathcal{S}$, and the current estimate $\mathbf{X}$, for each matchable $\mathbf{m}_n$ we want to find the matchable $\mathbf{m}'^*$ such that the error in Eq. (10) is minimized:

$$\mathbf{m}'^* = \operatorname*{argmin}_{\mathbf{m}' \in \mathcal{S}} e(\mathbf{X}\mathbf{m}, \mathbf{m}'). \tag{21}$$

To this end, we perform an approximate search based only on geometric information. We organize the matchables of the fixed scene $\mathbf{S}'$ in three KD-trees, one for each primitive, using their parameters, see Eq. (2). Subsequently, we perform an additional appearance-based culling to remove *bad* associations. That is, for points and lines we reject those associations whose descriptors are not similar. While, matched planes are discarded if their distance is bigger than a certain threshold.

2) Heterogeneous Constraints: Once performed the homogeneous search, we use the unassociated matchables of the moving scene $\mathbf{m}_n$ to perform a brute force search with the matchables of different type in the fixed scene. We evaluate the hybrid matchables distance as in Eq. (10), and we discard the constraints with error higher than a specified threshold t_{cross}.

V. Experimental Evaluation

In this section we present a set of experiments aiming at evaluating the performance of both the iterative and the direct version of the solver proposed in Section III. To this extent, we used a synthetically generated scene with known correspondences under different levels of noise and we analyzed the convergence behavior of the solver under different constraints. Section V-A reports the outcome of this set of experiments. Subsequently, in Section V-B we report the results of a comparative evaluation that involves a full registration pipeline built on top of our solver and other state-of-the-art approaches.

A. Synthetic Data

We designed a first experiment with synthetic data to validate our framework presented in Sec. III. We isolate the effects of the front-end and analyze the behavior of each type of constraint on both iterative and direct solver. To this end, we randomly spawn matchables in the scene according to its size w_s. The matchable origin is computed by sampling the x, y and z components from a uniform distribution in the interval $(-1, 1)$ and scaling them by w_s. For lines and planes the direction is again sampled from a 3D uniform distribution and subsequently normalized. This procedure allows to obtain the *fixed* scene. We set an arbitrary transform $\mathbf{X}$ and we apply its inverse to each generated matchable to obtain the *moving* scene. This allows to keep the correspondence between matchables in the two scenes (known data association).

In Fig. 5 we report the error evolution of the iterative solver for 10 iterations, while for the direct solver we present the result after one iteration. To estimate the effect of noise in the measurements we repeated the previous experiment by adding varying levels of noise to the matchables in the moving scene. The corresponding curves are marked as *N: no-noise*, *L: low-noise* $[\sigma_t^2 \sim 0.02, \sigma_r^2 \sim 0.05]$ and *H: high-noise* $[\sigma_t^2 \sim 0.5, \sigma_r^2 \sim 0.5]$. For the same type of constraint and level of noise both solvers operate on the same data and start from the same initial guess $[t_x = 0.3, t_y = -0.8, t_z = 0.6, q_x = 0.5, q_y = -0.5, q_z = 0.5]$. As evident from Fig. 5 both iterative and direct variants of our solver converge close to the optimum. The iterative solver is less sensitive to the

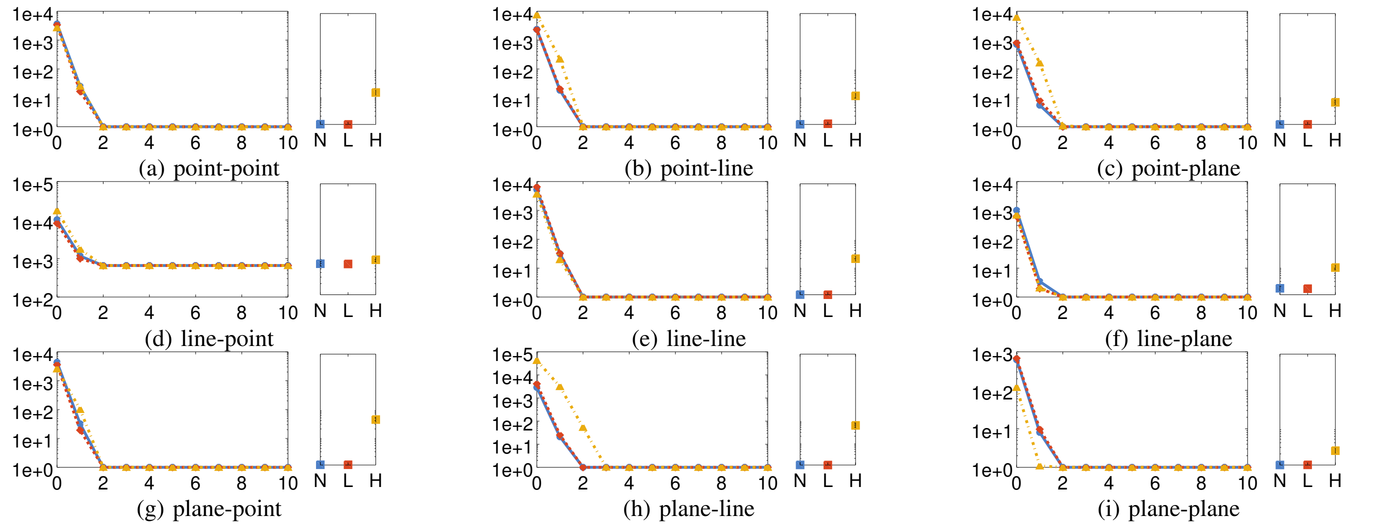


Fig. 5: Synthetic Experiments - Error Evolution is reported for all possible combination of *moving-fixed* matchables. In *solid blue* is shown the evolution of the error without noise, while in *dashed red* is reported the low-noise case and in *point-dashed yellow* the high-noise case. For each constraint, the left plot reports the Iterative Solver evolution, while the right plot shows the error after one Direct Solver iteration.

measurement noise, at a cost of multiple iterations. Conversely the direct solver should be preferred in low noise situations since it finds the optimum in just one step.

B. Simulated and Real Data

We compared the position tracking performances of our approach and three other state-of-the-art methods, namely: Normal ICP (NICP) [9], Dense Visual Odometry (DVO) [33] and Multi-Cue Photometric Registration (MPR) [32]. We used the author's open source implementations. The comparison has been conducted on the following publicly available datasets with ground truth:

- ICL-NUIM dataset [34], simulated RGB-D measurements in indoor scenarios.
- TUM benchmark suite [35], acquired with a Kinect v1 sensor in office-like environments.

For all the presented experiments we provide the achieved accuracy in terms of relative pose error (RPE), computed with the evaluation script provided by the TUM benchmark suite. For both datasets, we set $t_{cross} = 0.01$ as threshold to discard heterogeneous associations, as described in Sec. IV-B2. In Tab. III we report the results of each approach in terms of relative translational and rotational pose error per second on the ICL-NUIM sequences. The presented results show that our approach achieves accuracy comparable with state-of-the-art methods. To highlight the effects of using high-level primitives we present two variants of our approach, denoted as *SA-PT* and *SA-SHA*. *SA-PT* considers only points, while *SA-SHA* uses all primitives generated by the front end. *SA-PT* and *SA-SHA* does not present significant differences except in the first two sequences, where the camera is repeatetly pointed towards textureless walls, and salient features are poorly detected, causing an observable weakness in the *SA-PT* performances. On the other hand, the *SA-SHA* version takes advantage of the high-level geometric primitives. To provide the reader with an estimate of the memory occupancy of our method, we also report the number of matchables for our approach and the point cloud size of [9] in the final reconstructed scenes. We repeated the same procedure on eight sequences of the TUM benchmark suite. The data have been acquired with a Kinect v1 in office-like environments. Despite the simplistic front-end procedure we used in this work to extract matchables, our approach performs well on real data, as reported in Tab. IV. *SA-SHA* outperforms *SA-PT* in all the sequences, except when no structure is present (fr3/nostr-text-near-loop), where the two versions perform similarly. This encourages to further improve the front-end to perform a more robust registration.

VI. Conclusions

In this paper, we presented a unified and compact representation for scenes consisting of heterogeneous geometric primitives. Our representation allows to derive both iterative and direct registration methods. We conducted synthetic experiments to characterize the behavior of our solver. We furthermore validated our approach on real and simulated benchmark datasets. The experimental evaluation supports the claims that our approach yields compact models and that it is capable of exploiting the structure of the scene when present, while providing an accuracy comparable to other state-of-the-art methods. The focus of this work is on the representation of heterogeneous scenes and on an algorithm to register these scenes. By using a relatively simplistic registration front-end in conjunction with our solver, however we obtained performances close to those of other state-of-the-art system. The simplicity of the solver was compensated by the more descriptive representation and the use of multiple types of constraints between primitives.

TABLE III: ICL-NUIM - Relative Pose Error.

Approach	lr/tr0		lr/tr1		lr/tr2		lr/tr3		tr0		tr1		tr2		tr3	
	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]
MPR	0.0182	0.7055	0.0216	0.6307	0.0193	0.6366	0.0218	0.6351	0.0243	0.6464	0.0272	0.7173	0.0309	0.7469	0.0223	0.5733
DVO	**0.0063**	**0.5347**	0.0019	0.4773	**0.0046**	0.4983	**0.0055**	0.3607	**0.0064**	**0.4967**	0.0045	0.4834	0.0053	**0.5068**	**0.0037**	0.3567
NICP	**0.0063**	0.5403	**0.0014**	**0.4751**	0.0047	0.4948	0.0065	0.3907	0.0068	0.4988	**0.0043**	**0.4783**	**0.0052**	0.5078	0.0039	**0.3533**
{*cloud size*}	{2041773}		{1357503}		{1865157}		{2997922}		{2140708}		{1654631}		{1895441}		{2300953}	
SA-PT	0.0121	0.6466	0.0134	0.5802	0.0071	0.5667	0.0096	0.6553	0.0104	0.5254	0.0069	0.5223	0.0094	0.5472	0.0056	0.3685
SA-SHA	0.0077	0.5499	0.0042	0.4857	0.0053	**0.3843**	0.0059	**0.3512**	0.0094	0.5159	0.0057	0.4918	0.0071	0.5146	0.0052	0.3639
{*pt / ln / pl*}	{956 / 82 / 14}		{608 / 75 / 16}		{1180 / 69 / 13}		{981 / 65 / 10}		{1149 / 85 / 13}		{999 / 105 / 17}		{890 / 68 / 15}		{1257 / 145 / 16}	

TABLE IV: TUM - Relative Pose Error.

Approach	fr1/desk		fr1/desk2		fr2/desk		fr2/person		fr3/long-household		fr3/nostr-text-near-loop		fr3/str-text-far		fr3/str-text-near	
	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]
MPR	0.0617	3.3369	0.0921	5.1575	0.0364	1.6511	0.0478	1.4512	**0.0262**	1.2773	0.2505	7.6629	0.0413	1.2862	0.0451	2.1518
DVO	**0.0487**	2.8261	**0.0752**	**4.4137**	0.0377	1.5691	0.0332	0.9791	0.0441	1.4096	0.0249	1.0935	0.0943	2.4563	0.0364	1.9501
NICP	0.1123	8.6928	0.1989	9.9035	0.1032	2.8895	0.3479	13.8372	0.1412	4.8721	0.3179	8.9081	0.1002	1.4571	0.0736	2.3584
{*cloud size*}	{15488164}		{18325912}		{17430749}		{20887437}		{24204397}		{893384}		{3133948}		{4480124}	
SA-PT	0.0919	5.0505	0.1565	6.1042	0.0236	0.9921	0.0245	0.8820	0.0354	1.4398	**0.0211**	**0.9506**	0.0240	0.6646	0.0376	1.6553
SA-SHA	0.0888	**2.6464**	0.0974	5.9066	**0.0206**	**0.8661**	**0.0224**	**0.7693**	0.0274	**1.1352**	0.0227	1.0219	**0.0228**	**0.6546**	**0.0282**	**1.2725**
{*pt / ln / pl*}	{282 / 29 / 2}		{295 / 33 / 1}		{7934 / 59 / 2}		{10146 / 45 / 4}		{3849 / 93 / 10}		{2058 / 154 / 1}		{713 / 107 / 5}		{1223 / 26 / 2}	

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age," *IEEE Trans. on Robotics (TRO)*, vol. 32, pp. 1309–1332, 2016. [Online]. Available: http://arxiv.org/pdf/1606.05830v4

[2] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Intl. Journal of Computer Vision (IJCV)*, vol. 60, no. 2, 2004.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*. IEEE, 2011, pp. 2564–2571.

[4] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*. Springer, 2016, pp. 467–483.

[5] S. Madeo and M. Bober, "Fast, compact, and discriminative: Evaluation of binary descriptors for mobile applications," *IEEE Transactions on Multimedia*, vol. 19, no. 2, pp. 221–235, 2017.

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

[7] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*. IEEE, 1991, pp. 2724–2729.

[8] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Proc. of Robotics: Science and Systems (RSS)*, vol. 2, no. 4, 2009.

[9] J. Serafin and G. Grisetti, "Using extended measurements and scene merging for efficient and robust point cloud registration," *Journal on Robotics and Autonomous Systems (RAS)*, vol. 92, no. C, June 2017. [Online]. Available: https://doi.org/10.1016/j.robot.2017.03.008

[10] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2013, pp. 1352–1359.

[11] C. Olsson, F. Kahl, and M. Oskarsson, "The registration problem revisited: Optimal solutions from points, lines and planes," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1. IEEE, 2006, pp. 1206–1213.

[12] G. Klein and D. Murray, "Improving the agility of keyframe-based slam," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2008.

[13] E. Eade and T. Drummond, "Edge landmarks in monocular slam," *Journal on Image and Vision Computing (IVC)*, vol. 27, no. 5, 2009.

[14] T. Lemaire and S. Lacroix, "Monocular-vision based slam using line segments," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, April 2007, pp. 2791–2796.

[15] P. Smith, I. D. Reid, and A. J. Davison, "Real-time monocular slam with straight lines," *Proc. of British Machine Vision Conference (BMVC)*, 2006.

[16] C. Choi, A. J. Trevor, and H. I. Christensen, "Rgb-d edge detection and edge-based registration," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 1568–1575.

[17] Y. Lu and D. Song, "Robust rgb-d odometry using point and line features," in *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2015, pp. 3934–3942.

[18] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "PL-SLAM: Real-Time Monocular Visual SLAM with Points and Lines," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[19] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*. IEEE, 2014, pp. 6218–6225.

[20] A. J. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen, "Efficient organized point cloud segmentation with connected components," *Semantic Perception Mapping and Exploration*, 2013.

[21] M. Kaess, "Simultaneous Localization and Mapping with Infinite Planes," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[22] A. J. Trevor, J. G. Rogers, and H. I. Christensen, "Planar surface slam with 3d and 2d sensors," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*. IEEE, 2012, pp. 3041–3048.

[23] Y. Taguchi, Y. D. Jian, S. Ramalingam, and C. Feng, "Point-plane slam for hand-held 3d sensors," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, May 2013, pp. 5182–5189.

[24] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "CPA SLAM: Consistent Plane-Model Alignment for Direct RGB-D SLAM," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[25] J. A. Castellanos, J. Montiel, J. Neira, and J. D. Tardós, "The spmap: A probabilistic framework for simultaneous localization and map building," *IEEE Trans. on Robotics and Automation*, 1999.

[26] M. Bujnak, Z. Kukelova, and T. Pajdla, "Making minimal solvers fast," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2012, pp. 1506–1513.

[27] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *Intl. Journal of Computer Vision (IJCV)*, vol. 81, no. 2, p. 155, Jul 2008. [Online]. Available: https://doi.org/10.1007/s11263-008-0152-6

[28] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2006.

[29] R. G. von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 2010.

[30] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2010, pp. 778–792.

[31] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency," *Visual Communication and Image Representation*, pp. 794–805, 2013.

[32] B. Della Corte, I. Bogoslavskyi, C. Stachniss, and G. Grisetti, "A general framework for flexible multi-cue photometric point cloud registration," *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[33] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*. IEEE, 2013, pp. 3748–3754.

[34] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, May 2014.

[35] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.