On the Representation of Planes for Efficient Graph-based SLAM with High-level Features

Submitted: 18th July 2016; accepted: 16th August 2016

Jan Wietrzykowski

DOI: 10.14313/JAMRIS_3-2016/18

Abstract:
Despite the fact, that dense SLAM systems are extensively developed and are getting popular, feature-based ones still have many advantages over them. One of the most important matters in sparse systems are features. The performance and robustness of a system depends strictly on the quality of constraints imposed by feature observations and reliable matching between measurements and features. To improve those two aspects, higher-level features can be used, and planes are a natural choice. We tackle the problem of plugging planes into the g^2o optimization framework with two distinct plane representations: one based on a properly stated SE(3) parametrization and one based on a minimal parametrization analogous to quaternions. Proposed solutions were implemented as extensions to the g^2o framework and experiments that verify them were conducted using simulation. We provide a comparison of performance under various conditions that emphasized differences.

Keywords: *SLAM, features, plane parametrization, graph-based optimization*

1. Introduction

The simultaneous localization and mapping (SLAM) problem has to be solved whenever a mobile robot explores unknown environment. It can be a scenario of exploring a disaster site or a previously unvisited building. A variety of potential applications fosters development of new SLAM solutions and improvement of the existing ones. Particularly interesting is the domain of 3D SLAM systems based on affordable depth sensors, such as Kinect, because of their high availability, low price and ability to provide rich information about the environment [14]. Despite the recent growth of the number of dense SLAM systems, feature-based solutions still outperform them with respect to the precision of camera motion estimation and real-time performance [12]. The main component in feature-based systems are sparse features. Features have to provide enough information to determine the sensor/robot position relatively to an existing map. They have to be distinctive enough to prevent wrong associations between the new observations and the map. As the state estimation techniques, either filtration-based, like EKF [19], or optimization-based [11] are not suited for handling incorrect feature associations, the features in 3D SLAM have to be chosen carefully to fulfill those requirements.

Until recent, most systems were based on photometric point features, such as SURF [1] or ORB [15] or their geometric counterparts extracted from point clouds [14]. They are easy to compute and manage, but constrains produced by them are often inaccurate and they can be easily mismatched, which is a major issue. It is caused by the fact that point features are computed from a small local patch of the photometric or depth image, where the pixel values depend on many factors, such as lighting, camera exposition parameters or the depth range. A solution to this problem is to use higher level geometric features, whose positions relative to the sensor can be precisely determined from more global data. It is expected that features that describe spatially extended structures of the scene will be more distinctive and repeatable when re-observed by the sensor. A natural extension to point features are edge or plane features. The planes are particularly interesting, as they commonly exist in man-made environments, such as building interiors, and can be easily detected and isolated using a Kinect-like depth sensor. Walls, ceiling and floor are examples of large planar segments that can be used in localization and mapping. Due to the relatively small number of detected planes in a typical environment, they can be also easily matched between consecutive frames in the data stream.

Beside the issues related to the front-end part of a modern, optimization-based SLAM system [2], that deals with processing of the measurements and determination of measurement-to-object associations, attention has to be paid to the back-end. The back-end handles an optimization process that finds the positions of robot and features that minimizes certain criterion, given measurements and measurement-to-object associations. Among many such systems, particularly interesting are the factor-graph-based libraries, because of their flexibility and intuitive problem formulation. Thus, in section 5 we propose an extension to the popular factor graph g^2o back-end system [11] in the form of a new constraint edge and a corresponding feature vertex. The extension enables a fast and accurate optimization of pose-to-plane constraints by means of a minimal parametrization of the planes. We also compare the new approach to a simplified solution based on an overparametrized representation using the standard g^2o edges and vertices. This simplified solution is presented in section 4. We show at first that using the standard vertices and constraints available in g^2o is inefficient for planar features, and then we compare the two solutions proposed in the paper by applying the Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) metrics [17], widely used in the SLAM research community. As this is a preliminary work on SLAM based on high level features, we focus on the optimization backend and conduct simulations using a simplified system that lacks a real front-end for the extraction and matching of planar features. Similarly to the approach introduced in [3] we replace the front-end by a simulation that allows us to control the uncertainty of measurements by adding Gaussian noise and to control the number and location of features (i.e. planes in our case) in the environment.

2. Related Work

Since the introduction of the Kinect, that started an era of cheap depth sensors, many 3D SLAM solutions using depth and visual information emerged.

Pixel intensities and depth measurements are directly used in dense SLAM systems, like the one by Kerl *et al.* [10]. A motion between two consecutive frames is estimated by minimizing a difference between a predicted and an actual measurement in both, photometric and depth, domains. A large scale system, using a stereo camera instead of a depth sensor, is presented by Engel *et al*. [5]. A transformation between camera pose at two different frames can be also computed using iterative closest point, as in [13], where Kinect sensor is used to map and track dense surfaces. Another approach is to extract point features and use a sparse representation of measurements, as in the work by Belter *et al.* [2].

One of the earliest attempts to use planes as features was by Weingarten and Siegwart [19]. They adopted SPmodel [4] to represent planes and harnessed Extended Kalman Filter (EKF) to update the SPmap containing robot pose and feature locations. The SPmodel (symmetries and perturbation model) uses a probabilistic representation of the imprecision in the location of features, and the theory of symmetries to represent the partiality of the uncertainty due to the parametrization of the feature. Unfortunately, the plane representation is overparametrized and EKF-SLAM cannot exploit the sparsity of feature observations, in contrast to our solution.

Salas-Moreno *et al.* [16] proposed a method to densely map an environment with usage of bounded planes and surfels. Planar regions are refined and extended during camera's movement and can serve as a display for an augmented reality content.

A solution based on both, point and plane features was presented by Taguchi *et al.* [18]. They use a general form equation to parametrize planes, which is a non-minimal representation, and a sparse linear solver in the Gauss-Newton iterative optimization algorithm. Error calculation between the estimated and the measured plane is accomplished by means of random sampling of the measurement points. A sparse solver is also employed in the work by Kaess [8], where a minimal representation of planes based on quaternions was introduced. Optimization is done by the iSAM algorithm [9].

A popular tool for graph-based optimization is the g^2o framework, that outperforms many other systems, including iSAM [11]. It is widely used in point-feature-based SLAM systems, such as those by Mur-Artal *et al.* [12] or Belter *et al.* [3]. Thus, the g^2o library was chosen as the framework we want to test for handling optimization of pose-to-plane constraints, and then extend by a new minimal representation of the planar features.

3. Problem Formulation Using Graph

The part of the SLAM problem related to the back-end operation is to find the camera and feature positions that best fit the collected observations. To solve this problem efficiently, a proper representation of the constraints is needed. One of the possibilities is to model the system as a factorized probabilistic equation:

$$p(\mathbf{x}|\mathbf{z}) = \frac{1}{Z}\prod_{a\in F}\Psi_a(\mathbf{x}_a, \mathbf{z}_a), \tag{1}$$

where $\mathbf{x}$ are random variables, $\mathbf{z}$ are measurement variables, that are observed, Z is a normalization constant, F is a set of factors, $\Psi_a(\mathbf{x}_a, \mathbf{z}_a)$ is a value of a factor a, $\mathbf{x}_a$ is a subset of random variables that the factor a depends on, and $\mathbf{z}_a$ is a subset of measurement variables that the factor a depends on. Factor $\Psi_a(\mathbf{x}_a, \mathbf{z}_a)$ is a function, usually based on the Gaussian distribution, that measures how likely the state of variables $\mathbf{x}_a$ explain measurements $\mathbf{z}_a$. Throughout this paper, we use the following form of factors:

$$\Psi_a(\mathbf{x}_a, \mathbf{z}_a) = \exp\left\{-\frac{1}{2}\mathbf{e}_a(\mathbf{x}_a, \mathbf{z}_a)^T\Omega_a\mathbf{e}_a(\mathbf{x}_a, \mathbf{z}_a)\right\}, \tag{2}$$

where $\mathbf{e}_a(\mathbf{x}_a, \mathbf{z}_a)$ is an error function and an information matrix is denoted by Ω_a.

The error function returns a vector of differences between measurement prediction $\mathbf{h}_a(\mathbf{x}_a)$ based on the state of variables $\mathbf{x}_a$ and an actual measurement $\mathbf{z}_a$:

$$\mathbf{e}_a(\mathbf{x}_a, \mathbf{z}_a) = \mathbf{h}_a(\mathbf{x}_a) \ominus \mathbf{z}_a, \tag{3}$$

where $\ominus$ is an operator that is a generalization of the subtraction operation, for example taking into account rotation ambiguities, defined depending on the representation. Dimensionality of the error vector depends on the type of measurement and its representation. In the case of minimal representation of planes it is 3, and in the case of SE(3) it is 6.

The problem can be presented using a probabilistic graphical model, as shown in Fig. 1. The robot and feature positions are encoded by subsets of variables organized in a proper representation, e.g. a translation vector and 3 imaginary components of an unit quaternion for SE(3). If the position i has SE(3) representation, then the set $\mathbf{x}_{vi}$ contains 6 variables. There is no difference between variables representing feature and robot positions, besides from their meaning. Factor is dependent on all variables that represent the robot and feature positions connected to it.

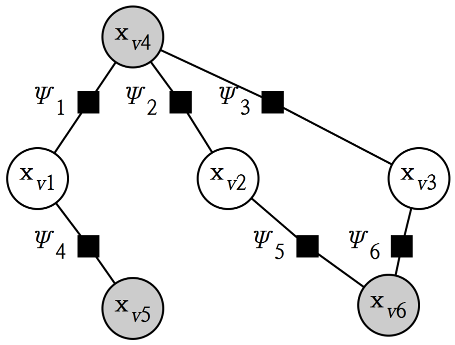

Fig. 1. Probabilistic graphical model representing the SLAM optimization problem. Variables encoding robot positions are marked as white circles, variables encoding feature positions are marked as gray circles, factors are marked as black squares and subset of variables that represent position i is denoted by $\mathbf{x}_{vi}$

This form emphasizes the sparsity of dependencies between the robot positions and feature positions. Properly exploiting the sparsity enables efficient optimization and is in the core of back-end systems development.

The optimization can be formulated as a process of finding values of variables $\mathbf{x}$ that maximizes the probability (1), denoted by $\mathbf{x}^*$. By taking the logarithm of probability it can be written as:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \frac{1}{Z} \prod_{a \in F} \Psi_a(\mathbf{x}_a, \mathbf{z}_a) \quad (4)$$

$$= \arg\min_{\mathbf{x}} \sum_{a \in F} \mathbf{e}_a(\mathbf{x}_a, \mathbf{z}_a)^T \Omega_a \mathbf{e}_a(\mathbf{x}_a, \mathbf{z}_a) \quad (5)$$

$$= \arg\min_{\mathbf{x}} \sum_{a \in F} l_a(\mathbf{x}_a, \mathbf{z}_a). \quad (6)$$

Although, in general, the problem is not convex, an iterative method is used to find the optimal values of $\mathbf{x}$. The assumption is made, that initial guess is good enough not to cause a divergence of the algorithm. At every iteration step, the functions $l_a(\mathbf{x}_a, \mathbf{z}_a)$ are linearized in the currently estimated state of variables $\mathbf{x}$ and an optimal step is calculated, denoted by $\Delta\mathbf{x}^*$. The linearization is expressed by (we omit dependence on $\mathbf{z}_a$ to simplify notation as values of $\mathbf{z}_a$ are constant):

$$l_a(\mathbf{x}_a + \Delta\mathbf{x}_a) \quad (7)$$

$$= \mathbf{e}_a(\mathbf{x}_a + \Delta\mathbf{x}_a)^T \Omega_a \mathbf{e}_a(\mathbf{x}_a + \Delta\mathbf{x}_a) \quad (8)$$

$$\simeq \left[\mathbf{e}_a(\mathbf{x}_a) + \mathbf{J}_a(\mathbf{x}_a)\Delta\mathbf{x}_a\right]^T \Omega_a \left[\mathbf{e}_a(\mathbf{x}_a) + \mathbf{J}_a(\mathbf{x}_a)\Delta\mathbf{x}_a\right] \quad (9)$$

$$= \mathbf{e}_a(\mathbf{x}_a)^T \Omega_a \mathbf{e}_a(\mathbf{x}_a) + 2\mathbf{e}_a(\mathbf{x}_a)^T \Omega_a \mathbf{J}_a(\mathbf{x}_a)\Delta\mathbf{x}_a + \Delta\mathbf{x}_a^T \mathbf{J}_a(\mathbf{x}_a)^T \Omega_a \mathbf{J}_a(\mathbf{x}_a)\Delta\mathbf{x}_a \quad (10)$$

$$= c_a(\mathbf{x}_a) + \mathbf{b}_a(\mathbf{x}_a)\Delta\mathbf{x}_a + \Delta\mathbf{x}_a^T \mathbf{H}_a(\mathbf{x}_a)\Delta\mathbf{x}_a, \quad (11)$$

where $\mathbf{J}_a$ is a Jacobian matrix of the error function with respect to variables $\mathbf{x}_a$ in the current point. If we expand all vectors and matrices in equation (11) to include all $\mathbf{x}$ variables, the iteration step can be written as:

$$\Delta\mathbf{x}^* = \arg\min_{\Delta\mathbf{x}} \sum_{a \in F} l_a(\mathbf{x}_a, \mathbf{z}_a) \quad (12)$$

$$\simeq \arg\min_{\Delta\mathbf{x}} \sum_{a \in F} c_a + \mathbf{b}_a\Delta\mathbf{x}_a + \Delta\mathbf{x}_a^T \mathbf{H}_a \Delta\mathbf{x}_a \quad (13)$$

$$= \arg\min_{\Delta\mathbf{x}} c + \mathbf{b}\Delta\mathbf{x} + \Delta\mathbf{x}^T \mathbf{H}\Delta\mathbf{x}, \quad (14)$$

where $c = \sum_{a \in F} c_a$, $\mathbf{b} = \sum_{a \in F} \mathbf{b}_a$, and $\mathbf{H} = \sum_{a \in F} \mathbf{H}_a$. The $\Delta\mathbf{x}^*$ value is calculated using equation:

$$\mathbf{H}\Delta\mathbf{x}^* = -\mathbf{b}. \quad (15)$$

After finding $\Delta\mathbf{x}^*$, current estimate is updated according to formula:

$$\mathbf{x}^{i+1} = \mathbf{x}^i \oplus \Delta\mathbf{x}^{i*}, \quad (16)$$

where $\oplus$ is a generalization of addition operator, defined depending on the representation. Note that increments are computed by considering derivatives of the error function with respect to variables, therefore units, in which those variables are expressed, are irrelevant.

Iterations are performed until an optimal solution is found. Usually, algorithms like Gauss-Newton or Lavenberg-Marquardt are used in combination with sparse linear optimizers to solve equation (15). The sparsity is encoded in the $\mathbf{H}$ matrix, since only some values are non-zero (value at position (i, j) can be non-zero only if variables x_i and x_j are related by a factor).

The g^{2}o framework organizes probabilistic graphical models in a form of vertices and edges. Vertices are representing subsets of variables denoting robot or feature poses and have to determine a proper representation of those variables. Therefore, vertices also implement $\oplus$ operator suitable for chosen representation. Edges are analogues of factors and, as such, they bind vertices with measurements. Generally, edges can connect multiple vertices, but in our system they always connect two. The operation that has to be implemented in an edge is error calculation, therefore they implement $\ominus$ operation. Optionally, edges can also implement analytical calculation of the Jacobian matrices, which are by default computed numerically [11].

4. SE(3) Plane Representation

This section presents a simplified solution based on a SE(3) representation of planes. It was necessary to introduce some assumptions and simplifications to plug planes into overparametrized representation.

Usually, using depth sensors, a plane measurement is represented as a normal vector $\mathbf{n}$ in the sensor frame of reference and distance d to the sensor (hereinafter called camera for convenience). Some assumptions have to be made to convert this representation to the SE(3) one, since the number of such overparametrized representations is infinite. Hence, we assumed that a plane coordinate system has the following properties:
- The origin is located in the plane point nearest to a camera.

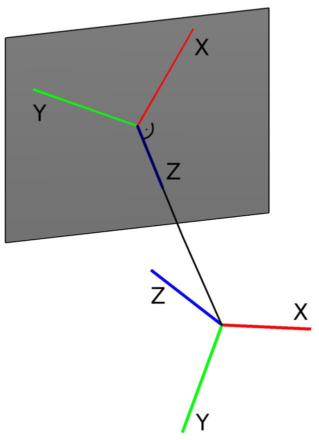

Fig. 2. A schematic view of assumptions about coordinate system of a plane

- The z axis is perpendicular to a plane.
- The x axis direction is determined by a cross product of the normal vector and the $[1,0,0]^T$ vector (if they are parallel, with $[0,1,0]$ vector). This assumption is only important to assure that the x axis will be parallel to the z axis.
- The y axis direction is determined by a cross product of the unit vectors in the z axis and the x axis directions.

Obviously, there is an ambiguity in the representation, and the global coordinates of a plane depend on the camera position (they won't be the same for different camera poses). It is caused by the fact that a plane is an object with 3 degrees of freedom (DOF), whereas the SE(3) representation has 6 DOF. Therefore, the frame of reference of an infinite plane can move freely along the x and y axes of this plane, and can rotate around it's z axis, which gives extra 3 DOF. The different placement of the origin cannot be avoided in a real-world scenario, but, as it will be shown, the difference has no effect on results thanks to a proper information matrix formulation. A schematic illustration of a plane coordinate system is presented in Fig. 2.

The standard g^{2}o SE(3) vertex represents the position in the form of a translation-quaternion (TQ) vector:

$$\mathbf{v} = \begin{bmatrix} t_x \\ t_y \\ t_z \\ q_x \\ q_y \\ q_z \end{bmatrix}, \qquad (17)$$

where t_x, t_y, t_z are Euclidean coordinates and q_x, q_y, q_z are imaginary components of an unit quaternion with the real component $q_w \geqslant 0$. In the SE(3) edge values $\mathbf{z}_a$ represent measurement of a transformation from the camera frame of reference to the plane frame and is also represented by a TQ vector. As $\mathbf{x}$ and $\mathbf{z}$ are just sets of numbers, we use $\mathbf{v}(\cdot)$ operator to indicate that they should be treated as a TQ vector. The symbols $\mathbf{x}_{ac}$ and $\mathbf{x}_{ap}$ denote subsets of $\mathbf{x}_a$ variables representing global camera and global robot position, respectively. Transformations between frames of reference are depicted in Fig 3.

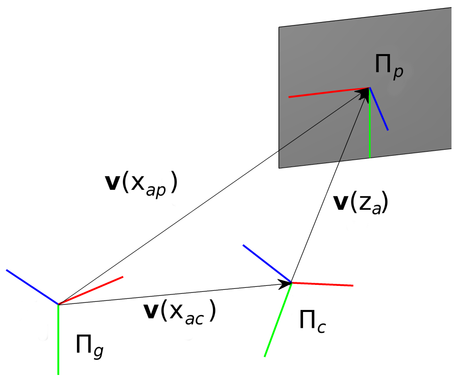

Fig. 3. Transformations between frames of reference. Π_g is a global frame of reference, Π_c is a camera frame of reference and Π_p is plane frame of reference

An error, introduced in the equation (3), is defined as follows:

$$\mathbf{e}_a(\mathbf{x}_a, \mathbf{z}_a) = \mathbf{v}(\mathbf{z}_a)^{-1}\left[\mathbf{v}(\mathbf{x}_{ac})^{-1}\mathbf{v}(\mathbf{x}_{ap})\right], \qquad (18)$$

where multiplication is a concatenation of transformations (not a matrix multiplication), $\mathbf{v}^{-1}$ is an inversion of the transformation $\mathbf{v}$, and $\mathbf{v}(\mathbf{x}_{ac})^{-1}\mathbf{v}(\mathbf{x}_{ap})$ can be interpreted as measurement prediction.

The error is defined in the plane's frame of reference, therefore an information matrix $\mathbf{\Omega}_a$ has to be defined in the same frame. In an overparametrized representation, such as the one considered here, the information matrix is particularly important. It should define large (theoretically infinite) uncertainty in the dimensions, in which the representation is ambiguous. If the i-th dimension is a surplus, then the element of $\mathbf{\Omega}_a$ at location (i,i) should be equal to zero. Unfortunately, the matrix constructed in such way would be rank deficient and impossible to invert. The inability to invert would discard a large number of optimization algorithms. Therefore, we decided to circumvent this limitation by inserting a very small number instead of 0. Another problem was how to specify an information matrix for rotation represented by a quaternion. To overcome the problem, we constructed a covariance matrix for the extended representation (including q_w value) in the form:

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_t & \mathbf{0}_{3x4} \\ \mathbf{0}_{4x3} & \mathbf{C}_{rq} \end{bmatrix}, \qquad (19)$$

where $\mathbf{C}_t$ was defined as follows:

$$\mathbf{C}_t = \mathrm{diag}(c, c, 1) \qquad (20)$$

and $\mathbf{C}_{rq}$ as follows:

$$\mathbf{C}_{rq} = \mathbf{J}_{qr}\mathbf{C}_{rr}\mathbf{J}_{qr}^T. \qquad (21)$$

Here $\mathbf{C}_{rr}$ is the covariance matrix for a rotation expressed by a 3×3 rotation matrix, defined as (for a row-major order of matrix elements):

$$\mathbf{C}_{rr} = \mathrm{diag}(c, c, 1, c, c, 1, 1, 1, 1) \qquad (22)$$

and $\mathbf{J}_{qr}$ is Jacobian matrix of the conversion from a rotation matrix to a quaternion at the identity point (derived from the equation converting rotation matrix representation to a quaternion representation):

$$\mathbf{J}_{qr} = \begin{bmatrix} 0 & 0 & 0 & 0.125 \\ 0 & 0 & 0.250 & 0 \\ 0 & -0.250 & 0 & 0 \\ 0 & 0 & -0.250 & 0 \\ 0 & 0 & 0 & 0.125 \\ 0.250 & 0 & 0 & 0 \\ 0 & 0.250 & 0 & 0 \\ -0.250 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.125 \end{bmatrix}^T . \quad (23)$$

In the above equations c is some large value that indicates that in this dimension a variance is infinite. We used $c = 1000$ which was a good compromise between accuracy and numerical stability. The final information matrix is the 6×6 upper-left corner part of the inverse of the $\mathbf{C}$ matrix:

$$\Omega_a = [\mathbf{C}^{-1}]_{6\times 6}. \quad (24)$$

The $\oplus$ operator is realized by multiplying a transformation represented by the current state of variables by a transformation expressed by the computed increment $\Delta\mathbf{x}^{i*}$:

$$\mathbf{v}(\hat{\mathbf{x}}_{av}^{i+1}) = \mathbf{v}(\hat{\mathbf{x}}_{av}^{i})\mathbf{v}(\Delta\mathbf{x}_{av}^{i*}). \quad (25)$$

The implementation was intended to be simple and demand small amount of work, so it was realized using standard g^{2}o edges and vertices. The additional advantage of this approach is that the g^{2}o framework implements analytical calculation of Jacobian matrices for standard classes. It was necessary to compute information matrix and implement a measurement simulation. The simulation was accomplished by adding Gaussian noise to the normal vector $\mathbf{n}$ components and a distance value d, and then calculating $\mathbf{v}(\mathbf{z}_a)$ using the previously defined assumptions about the coordinate system of a plane.

5. Minimal Plane Representation

To represent a plane only 3 values are required, since it is an object with 3 DOF. In this section we present a solution based on a representation that uses only 3 values and therefore is minimal. Nevertheless, using a normal vector and a distance requires 4 values: 3 components of $\mathbf{n} = [n_x, n_y, n_z]^T$ and d. The same problem occurs with a general plane equation:

$$p_1 x + p_2 y + p_3 z + p_4 = 0. \quad (26)$$

The solution is to normalize the general plane equation, so $\|\mathbf{p}\| = \|[p_1, p_2, p_3, p_4]^T\| = 1$ and restrict $p_4 \geqslant 0$. After doing so, only the first 3 components of the vector $\mathbf{p}$ are relevant, since the last one can be retrieved using formula:

$$p_4 = \sqrt{p_1^2 + p_2^2 + p_3^2}. \quad (27)$$

It is an analogue to an unit quaternion and all operations on quaternions can be transfered to this representation [8]. Therefore, it is a minimal representation without singularities, what makes it suitable for optimization purpose.

With the minimal representation, we used exponential and logarithm map to calculate the error and update current positions. An exponential map is a map from the Lie algebra to a Lie group and a logarithm map is a map in the reverse direction. The error calculation in an edge connecting a camera position and a plane position is performed using the logarithm map of quaternions:

$$\mathbf{e}_a(\mathbf{x}_a, \mathbf{z}_a) = \mathbf{q}\left[\mathbf{T}(\mathbf{x}_{ac})^T \mathbf{p}(\mathbf{x}_{ap})\right] \ominus \mathbf{q}(\mathbf{z}_a) \quad (28)$$

$$= \log\left\{\mathbf{q}\left[\mathbf{T}(\mathbf{x}_{ac})^T \mathbf{p}(\mathbf{x}_{ap})\right]^{-1} \mathbf{q}(\mathbf{z}_a)\right\}, \quad (29)$$

where $\mathbf{q}(\mathbf{x}_v)$ denotes a quaternion formed from variables $\mathbf{x}_v$, $\mathbf{T}(\mathbf{x}_v)$ is a homogeneous transformation matrix constructed from variables $\mathbf{x}_v$, and $\mathbf{p}(\mathbf{x}_v)$ is a plane equation based on variables $\mathbf{x}_v$. Note that transformation of a general plane equation from Π_g frame of reference to Π_c frame is expressed differently than the same transformation for a point. It is done by the equation ($\mathbf{T}_{i,j}$ denotes a homogeneous transformation matrix for a transformation from Π_i to Π_j):

$$\mathbf{p}_c = \mathbf{T}_{c,g}^{-T}\mathbf{p}_g \quad (30)$$

$$= \mathbf{T}_{g,c}^{T}\mathbf{p}_g. \quad (31)$$

The logarithm map is a 3 dimensional vector given by the equation:

$$\log(\mathbf{q}) = 2\frac{\cos^{-1}(q_w)}{\|\mathbf{q}_v\|}\mathbf{q}_v, \quad (32)$$

where $\mathbf{q}_v$ is an imaginary part of the quaternion $\mathbf{q}$.

Updates of variables are done using exponential maps for both, camera positions and plane positions. The update for planes is done using the following formula:

$$\mathbf{q}(\mathbf{x}_{ap}^{i+1}) = \exp\left[\omega(\Delta\mathbf{x}_{ap}^{i*})\right]\mathbf{q}(\mathbf{x}_{ap}^{i}), \quad (33)$$

where exponential map for a plane is defined as:

$$\exp(\omega) = \begin{bmatrix} \frac{1}{2}\sin(\frac{1}{2}\|\omega\|)\omega \\ \cos(\frac{1}{2}\|\omega\|) \end{bmatrix}. \quad (34)$$

The update for camera positions is done in a similar way, but instead of quaternions, operations are performed on TQ vectors:

$$\mathbf{v}(\mathbf{x}_{ac}^{i+1}) = \exp\left[\mathbf{d}(\Delta\mathbf{x}_{ac}^{i*})\right]\mathbf{v}(\mathbf{x}_{ac}^{i}). \quad (35)$$

The increment used in the above equation comprises a translational and a rotational part, same as the TQ vector:

$$\mathbf{d} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (36)$$

Denoting the result of the exponential mapping by:

$$\exp(\mathbf{d}) = \begin{bmatrix} \mathbf{t}_d \\ \mathbf{q}_d \end{bmatrix}, \tag{37}$$

the calculation can be done using equations:

$$\mathbf{t}_d = \mathbf{V}\upsilon \tag{38}$$

and

$$\mathbf{q}_d = \mathbf{q}(\mathbf{R}). \tag{39}$$

Note that, in the notation above, a quaternion is constructed from a rotation matrix, not from a 4-dimensional vector. The matrices $\mathbf{V}$ and $\mathbf{R}$ are given by equations:

$$\mathbf{V} = \mathbf{I} + \frac{1-\cos(\|\omega\|)}{\|\omega\|^2}[\omega]_\times + \frac{\|\omega\| - \sin(\|\omega\|)}{\|\omega\|^3}[\omega]_\times^2 \tag{40}$$

and

$$\mathbf{R} = \mathbf{I} + \frac{\sin(\|\omega\|)}{\|\omega\|}[\omega]_\times + \frac{1-\cos(\|\omega\|)}{\|\omega\|^2}[\omega]_\times^2, \tag{41}$$

where $[\omega]_\times$ is a skew-symmetric matrix:

$$[\omega]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{42}$$

We implemented an extension to the standard set of g^{2}o edges and vertices by adding a vertex representing a plane position and an edge connecting camera position and plane position. As the camera position vertex we used a SE(3) vertex that uses exponential map, included in the framework. During experiments, measurements were simulated by adding Gaussian noise to the components of the normal vector $\mathbf{n}$ and to the distance d value. The quaternion representation was obtained by constructing a general plane equation, and then properly normalizing a vector of parameters. The vector had the following form:

$$p = \begin{bmatrix} n_x \\ n_y \\ n_z \\ -d \end{bmatrix}. \tag{43}$$

In the current version, the Jacobian matrix was computed numerically.

6. Experiments and Results

To experimentally evaluate the proposed models of planar features and the constraints related to them, we simulated motion of a camera in an empty room. As demonstrated in [3], such a simple experiment clearly reveals how the behavior of the optimization back-end depends on the parametrization of the uncertainty model of the features. The simulated front-end does not introduce any errors due to wrong feature associations or multiplicated features, thus the results are isolated from the qualitative errors that are unavoidable in a real front-end. In order to make the simulation maximally realistic as to the dynamics of the sensor motion we used an example trajectory from the *ICL-NUIM Office Room Dataset* [7] and inserted a virtual floor and two walls that were always observed by the sensor. An overview of the simulated environment is shown in Fig. 4. Normal vectors $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3$ and distance values d_1, d_2, d_3 were the common source of information for both parametrizations. All representations were obtained from those values with added Gaussian noise of the standard deviation equal to 0.01.

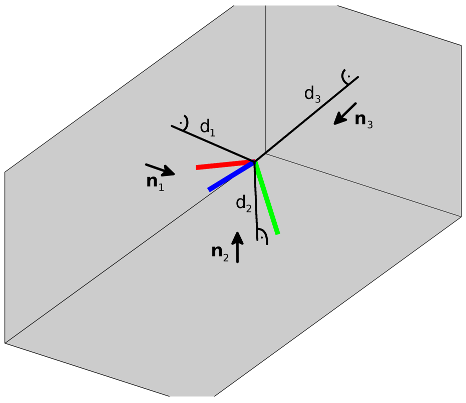

Fig. 4. An overview of the simulated environment. Normal vectors $\mathbf{n}_1$, $\mathbf{n}_2$, $\mathbf{n}_3$ and distance values d_1, d_2, d_3 were noised measurement values

Optimization was done in batch mode. First, all vertices, along with their initial position estimations, and edges were added to the graph and than the optimization process was triggered. In both cases we used Gauss-Newton algorithm with the Preconditioned Conjugate Gradient (PCG) linear solver. The number of iterations was limited to 100, although in all tests the algorithm converged earlier. We tested when a change of the error value l between iterations will drop below $10^{-6}l$. For the SE(3) representation it was after the 26-th iteration and took 0.343 s. In the case of the minimal representation, it happened after the 9-th iteration and took 0.257 s. Both results enable real-time operation, but the optimization with the minimal representation converges faster and needs fewer iterations.

An initial guess to camera positions was obtained by simulating dead reckoning (e.g. visual odometry, as used in [2]). We calculated differences between consecutive poses in the ground truth trajectory, added noise to every difference and then build an odometry trajectory by stacking noised difference transformations. If $\tilde{T}_{i,i+1}$ is a noised transformation from ground truth trajectory pose i to pose $i+1$, then the odometry pose $i+1$ is expressed by:

$$O_{i+1} = O_i \tilde{T}_{i,i+1}. \tag{44}$$

First, we investigated a behavior of the system when the information matrix $\mathbf{\Omega}_a$ for SE(3) representation was set to identity to highlight that using this representation for planes is not obvious. Setting the information matrix to identity is a common practice, but should be done carefully, in particular when the measurements or the state variables span over non-Euclidean manifold spaces [6]. Effects of neglecting

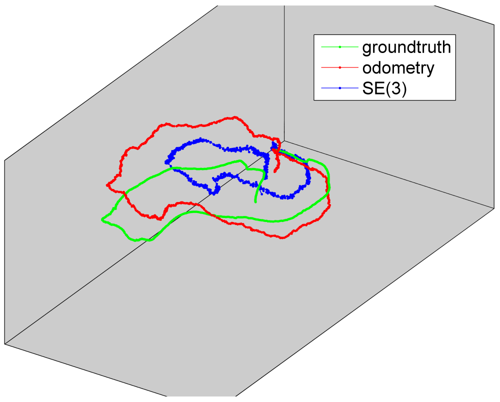

Fig. 5. Visualization of the SE(3) optimization results with the information matrices set to identity

Tab. 1. Results of the SE(3) and minimal representation optimizations with perpendicular planes

measure		SE(3)	minimal
RPE translational [m]	rmse	0.034	0.031
	mean	0.031	0.028
	median	0.029	0.026
	std	0.014	0.013
	max	0.111	0.107
RPE rotational [°]	rmse	1.001	1.139
	mean	0.923	1.049
	median	0.016	0.018
	std	0.389	0.446
	max	2.859	2.960
ATE [m]	rmse	0.021	0.017
	mean	0.019	0.016
	median	0.019	0.015
	std	0.008	0.007
	max	0.062	0.042

the partiality of uncertainty in planar features can be seen in Fig. 5. As expected, when the identity matrices are used, the optimized trajectory is a degenerated version of the ground truth one, because the least-squares minimization could not be constrained to the proper manifold.

The next experiment compared the SE(3) and minimal representations with properly set information matrices for the situation when the plane features were perpendicular each to the other. This "natural" configuration of walls in a room provides also the best constraints to the simple system under study, as there are similar constraints along each axis of the global coordinate system. Quantitative results are gathered in Tab. 1, while the estimated trajectories are visualized in Fig. 6. We apply the ATE and RPE metrics. ATE compares the distance between the estimated and ground truth trajectories, whereas RPE corresponds to the drift of the trajectory [17]. From the trajectories it is clearly visible that both solutions reconstructed the camera motion with small errors. The numeric results are slightly better for the minimal representation, but the differences are rather irrelevant.

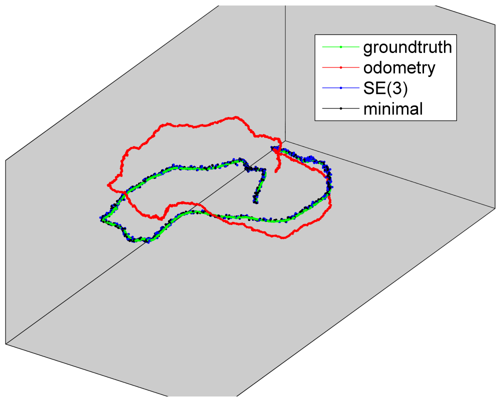

Fig. 6. Results of the SE(3) and minimal representation optimizations with perpendicular planes

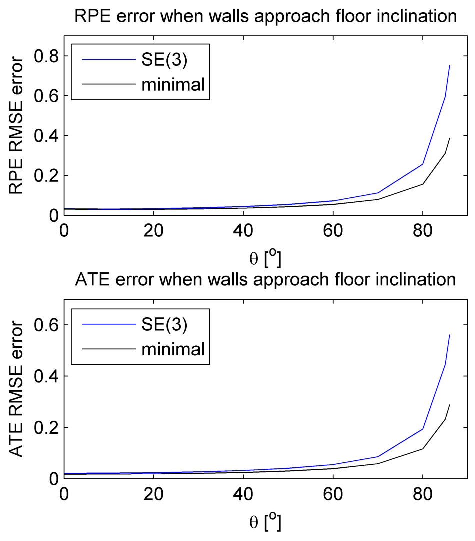

Fig. 7. Results of the SE(3) and minimal representation optimizations when walls approach floor inclination. The θ is an angle by which walls were tilted

Differences emerged with more challenging setups, in which walls were not perpendicular to the floor and each to the other. A dependency between the angle by which the walls were tilted and the errors in trajectory estimation is visible in Fig. 7. When the angle is small and the measurements of the positions of walls impose strong constraints, the error values are similar to the ones obtained in the previous experiment, but when the walls approach the floor inclination and are close to being parallel to the ground plane, the error for estimation with the SE(3) representation grows faster. Note that the ATE metrics plot behaves exactly as the relative positional error plot, which is caused by the fact, that there are no significant loop closures in the small simulation environment, hence the trajectory does not change much after the final optimization. The results with walls tilted by 80° are visualized in Fig. 8.

7. Conclusions

We proposed two solutions to the problem of representing plane-based features in the g^{2}o framework. First solution was based on a standard set of ver-

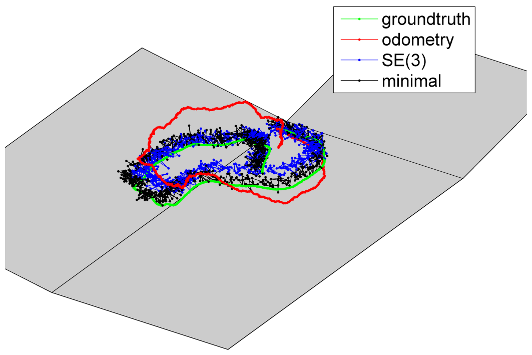

Fig. 8. Visualization of the SE(3) and minimal representation optimizations results with walls tilted by 80°

tices and edges from the framework and represented planes using SE(3) parametrization with carefully prepared information matrix. Second solution used minimal representation of planes and required an implementation of the vertex and the edge that extended the g^2o functionality. The implementation is an important contribution as it can be easily used in further development, as well as in other applications. Experiments verified that both approaches give reasonable results and can operate in real-time. When overparametrized representation is used, it is important to carefully construct information matrix. The matrix instructs the optimization algorithm which dimensions are relevant and what are relations between coordinate uncertainties. Despite the fact that presented approaches are theoretically equivalent, when conditions are harsh, the more specific solution performs better as comes to accuracy and convergence time. The work gives an insight how surplus dimensions affect the optimization process. The difference could be more significant if Jacobian matrices were computed analytically in both cases. Although experiments in a synthetic environment, without a real front-end, give no possibility to compare the performance of our approach with other systems, the presented solution provides a good start point for development of a complete SLAM system based on higher-level features.

Future work will focus on adding a front-end functionality to the system. We want to develop an algorithm for detecting and isolating planes, matching planes between consecutive frames and recognizing visited places. Considering other types of features in a single framework to build a robust and versatile system is also planned.

AUTHOR

Jan Wietrzykowski[*] – Poznań University of Technology, Institute of Control and Information Engineering, ul. Piotrowo 3A, 60-965 Poznań, Poland, e-mail: jan.wietrzykowski@cie.put.poznan.pl.

[*]Corresponding author

REFERENCES

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)", *Comput. Vis. Image Underst.*, vol. 110, no. 3, 2008, 346–359.

[2] D. Belter, M. Nowicki, and P. Skrzypczyński. "Accurate Map-Based RGB-D SLAM for Mobile Robots". In: L. P. Reis, A. P. Moreira, P. U. Lima, L. Montano, and V. Muñoz Martinez, eds., *Robot 2015: Second Iberian Robotics Conference*, volume 418 of *Advances in Intelligent and Soft Computing (AISC)*, 533–545. Springer International Publishing, 2016.

[3] D. Belter, M. Nowicki, and P. Skrzypczyński, "Improving accuracy of feature-based RGB-D SLAM by modeling spatial uncertainty of point features". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, 1279–1284.

[4] J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardos, "The SPmap: a probabilistic framework for simultaneous localization and map building", *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, 1999, 948–952.

[5] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras". In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 2015, 1935–1942.

[6] G. Grisetti, R. Kümmerle, and K. Ni, "Robust optimization of factor graphs by using condensed measurements". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, 581–588.

[7] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for rgb-d visual odometry, 3d reconstruction and slam". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, 1524–1531.

[8] M. Kaess, "Simultaneous Localization and Mapping with Infinite Planes". In: *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Seattle, WA, 2015, 4605 – 4611.

[9] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental Smoothing and Mapping", *IEEE Transactions on Robotics*, vol. 24, no. 6, 2008, 1365–1378.

[10] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, 2100–2106.

[11] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, 3607–3613.

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocu-

lar SLAM System", *IEEE Transactions on Robotics*, vol. 31, no. 5, 2015, 1147–1163.

[13] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinect-Fusion: Real-time dense surface mapping and tracking". In: *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, 2011, 127–136.

[14] M. Nowicki and P. Skrzypczyński, "Experimental Verification of a Walking Robot Self-Localization System with the Kinect Sensor", *Journal of Automation, Mobile Robotics and Intelligent Systems*, vol. 7, no. 4, 2013, 42–52.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*, 2011, 2564–2571.

[16] R. F. Salas-Moreno, B. Glocken, P. H. J. Kelly, and A. J. Davison, "Dense planar SLAM". In: *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, 2014, 157–164.

[17] 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 International Conference on Intelligent Robot Systems (IROS)*, 2012.

[18] Y. Taguchi, Y. D. Jian, S. Ramalingam, and C. Feng, "Point-plane SLAM for hand-held 3D sensors". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, 5182–5189.

[19] J. Weingarten and R. Siegwart, "3D SLAM using planar segments". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, 3062–3067.