VSLAM pose initialization via Lie groups and Lie algebras optimization

German Ros*†, Julio Guerrero‡, Angel D. Sappa*, Daniel Ponsa*† and Antonio M. Lopez*†
gros@cvc.uab.es, juguerre@um.es, asappa@cvc.uab.es, daniel@cvc.uab.es, antonio@cvc.uab.es
*Computer Vision Center, Campus UAB, 08193, Bellaterra, Spain
†Computer Science Dpt., Univ. Autònoma de Barcelona, Campus UAB, Bellaterra, Spain
‡Dept. Matemàtica Aplicada, Facultad de Informàtica, Universidad de Murcia, Spain

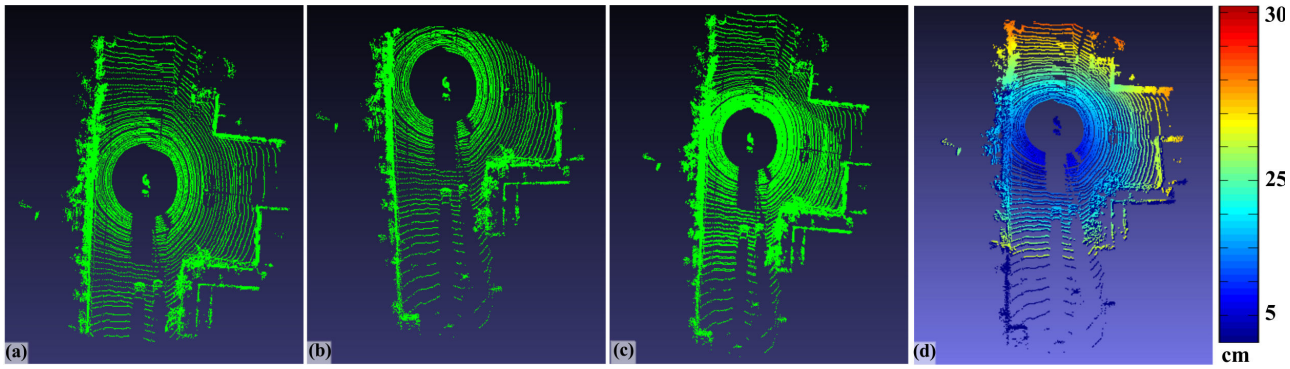

Fig. 1: Example of rigid pose estimation in 3D for a synthetic experiment based on real data [1]. The size of the scene is 217 m^2. The experiment assumes a 20% of outliers and 200 correspondences. (a) scene at t_0; (b) scene at t_1; (c) alignment obtained by the proposed approach; (d) representation of the error w.r.t the ground truth (**cm**).

***Abstract*— We present a novel technique for estimating initial 3D poses in the context of localization and Visual SLAM problems. The presented approach can deal with noise, outliers and a large amount of input data and still performs in real time in a standard CPU. Our method produces solutions with an accuracy comparable to those produced by RANSAC but can be much faster when the percentage of outliers is high or for large amounts of input data. On the current work we propose to formulate the pose estimation as an optimization problem on Lie groups, considering their manifold structure as well as their associated Lie algebras. This allows us to perform a fast and simple optimization at the same time that conserve all the constraints imposed by the Lie group $SE(3)$. Additionally, we present several key design concepts related with the cost function and its Jacobian; aspects that are critical for the good performance of the algorithm.**

I. INTRODUCTION

The pose estimation problem is present in many applications within the fields of computer vision and robotics. This problem has received much attention during the last decades [2], [3], [4]; being specially relevant the solutions proposed for dealing with high levels of noise and outliers [5]. Robust pose estimation plays a crucial role in Visual Simultaneous Localization and Mapping (VSLAM) [6] and Visual Odometry (VO) [7] problems, where relative poses $\mathcal{T}_r \in SE(3)$ need to be accurately estimated in real time. To do so, the most accepted practice consists of calculating an initial estimation of each pose as soon as the sensor captures a new frame. Then all these poses —so far considered as an initial guess, are jointly optimized to ensure consistency between them. These global (or partially global) optimizers require initial guesses to be as accurate as possible but do not have to be optimal. However, the time used to perform such initializations is critical and needs to be kept low. For these reasons, we present a novel technique to perform fast and accurate pose estimation applied to the initialization of VSLAM poses, a task that is carried out during the whole process of localization.

In this paper we focus on the particular case of estimating rigid transformations in 3D, $\mathcal{T}_r \in SE(3)$, between two different instants of the same scene (see Fig. 1 for an example). This is done by assuming that 3D information is available, for instance coming from a stereo rig. Such case commonly arises in VSLAM and VO problems, and therefore, both of them can benefit from our contribution. It is important to remark that our proposal is also valid for other types of models and input data, but for simplicity we only treat $SE(3)$ in this paper.

Although pose estimation is addressed by a plethora of methods in the literature, in practice the only approaches that can successfully deal with high levels of noise and outliers are the RANdom SAmple Consensus (RANSAC) and its variations [5]. Most classical approaches based on least-squares optimization and SVD decompositions, produce poor results in the presence of outliers and other natural artefacts. Furthermore, the process of estimating $\mathcal{T}_r$ must be fast as well as robust. Unfortunately, RANSAC-like approaches can be computationally prohibitive if conditions are not favourable, as it happens when increasing the percentage of outliers or the amount of input information. In these cases, the method still might be able to produce a satisfactory solution, but its execution time will be severely affected,

preventing real time performance.

The reasons presented above motivate us to propose a new approach for pose estimation that can deal with noise, outliers and a large amount of input data while keeping real-time performance in a standard CPU. Our method produces solutions with an accuracy comparable to those produced by RANSAC but it is much faster. To achieve this we have addressed the estimation of the parameters as an optimization problem on Lie groups, considering their manifold structure as well as their associated Lie algebras. The optimization is carried out on the linear space of the Lie algebra but we move back to the manifold in order to perform the evaluation of the solution candidates. Although similar approaches have been tested before, we will show that our formulation differs from those proposed in the literature.

Our contributions in this paper are twofold. First we present a novel technique for pose initialization that makes use of the formalism of Lie group spaces to achieve accurate and fast solutions. We experimentally show that our method can deal with noise, outliers and large amounts of input data while performing in real-time. Our comparisons demonstrate that in many cases the proposed approach can outperform state-of-the-art techniques, such as RANSAC. Secondly, we formalize key design concepts, which have a great impact on the good performance of this algorithm, as for instance: the algebraical reduction of the cost function, data normalization and an efficient way of computing the Jacobian of a function defined on the Lie algebra.

The remainder of this paper is structured as follows. Section II introduces the key mathematical concepts used in our proposal. In section III we formalize the problem of rigid pose estimation in 3D and give an overview of the representative approaches found in the literature. Section IV describes the presented approach and highlights its most important steps. The validation of these concepts is showed in section V, where our technique is tested and compared to other approaches. Finally we conclude this paper in section VI, giving an advance of our future work.

II. Preliminars

This section provides a brief introduction to Lie groups, Lie algebras and some basic properties that will be used along this paper. For further information about these concepts we refer the reader to [8], [9] and [10]. We will start defining a Lie group as a set G that satisfies all the conditions of a group and is also a manifold (i.e., it is smooth and has a differential structure). In this paper we only consider manifolds of finite dimension d, embedded in $\mathbb{R}^d$ (with $k \geq d$). We can say that the manifold $\mathcal{M}$ looks like $\mathbb{R}^d$ locally [11]. Then, given an open set $\mathcal{U}$ on the manifold, such that $\mathcal{U} \subset \mathcal{M} \subset \mathbb{R}^k$ and another set $\Omega \subset \mathbb{R}^d$, there is a homeomorphism (i.e., a continuous bijective map) that transforms from one to the other, and therefore, they are topologically equivalent.

It is important to know that Lie groups come with associated Lie algebras. A Lie algebra A of a Lie group G is a vector space that represents the tangent space $T_{\mathbf{p}}G$ of the group at a given point $\mathbf{p}$. At the identity, this tangent space provides a linearization of the Lie group, maintaining most of its properties [8].

The link between a Lie group and its Lie algebra is called the exponential map $\mathbf{exp}$. This map and its inverse $\mathbf{exp}^{-1} = \mathbf{log}$ allow us to move between the vector space generated by a Lie algebra and the group. It is important to remark that, for Lie groups like $SO(3)$ or $SE(3)$, the exponential map is surjective but not injective. In other words, all the elements of these groups can be "reached" from the algebra by the exponential map, but there are infinite elements in the algebra that will be mapped to the same group element (non-unique mapping) [8]. In the same way, **log** is a map just defined for some regions and under certain circumstances.

The $\mathbf{exp}$ map can be generally defined for all matrix groups $G \subset GL(n, \mathbb{R})$, where $GL(n, \mathbb{R})$ is the group of $n \times n$ real invertible matrices. In this way, $\mathbf{exp}(A)$ is defined as:

$$\mathbf{exp}(A) = I_n + \sum_{k \geq 1} \frac{A^k}{k!} = \sum_{k \geq 0} \frac{A^k}{k!}, \tag{1}$$

where I_n is the $n \times n$ identity matrix. It is proven that this series is absolutely convergent for any matrix [8]. However, there are explicit ways of calculating $\mathbf{exp}$ for some specific groups. This is the case of $SE(3)$, the group of rigid transformations in $\mathbb{R}^3$. Here, the exponential map can be computed as:

$$\mathbf{exp}(S) = I_4 + S + \frac{(1 - cos(\theta))\, S^2}{\theta^2} + \frac{(\theta - sin(\theta))\, S^3}{\theta^3}. \tag{2}$$

This expression is an adaptation of the well known Rodrigues' formula [12]. Here, $S \in \mathfrak{se}(3)$, which is the Lie algebra corresponding to the tangent space of $SE(3)$ and is parametrized as $S = \begin{bmatrix} \hat{\omega} & v \\ \vec{0} & 0 \end{bmatrix}$. Where $\omega = (\omega_x, \omega_y, \omega_z)$ is a 3-vector representing a 3D rotation and $\hat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$ is its representation in $\mathfrak{so}(3)$, as a 3×3 real skew-symmetric matrix; $v \in \mathbb{R}^3$ is a column vector that represents the translation and $\theta = \|\omega\|_{\ell_2}$, i.e., the amount of rotation in the direction defined by $\frac{\omega}{\theta}$. It is also important to note that $\mathfrak{se}(3)$ is isomorphic with $\mathbb{R}^6$, which means that there exists a map that transforms between both spaces. We will take advantage of this property by representing elements of $\mathfrak{se}(3)$ as the 6-vector (ω, v).

Our approach strongly depends on (2), and the isomorphism between $\mathfrak{se}(3)$ and $\mathbb{R}^6$. These properties allow us to enforce all the constraints of $SE(3)$ while keeping the optimization procedure simple on the vector space $\mathfrak{se}(3) \approx \mathbb{R}^6$.

III. Problem Definition and Related Work

As presented in the introduction, the interest of this paper is the estimation of rigid transformations between views as a framework for parameter initialization in VSLAM and VO applications. For this reason we are focused on the space

of rigid transformations in 3D, $SE(3)$ —intimately related to the previous applications. Nevertheless, our technique can also be applied to other transformation groups as long as they maintain the Lie group structure (e.g., $SO(3)$, $Sim(3)$, $Aff(3)$ and others.). In fact, it is also possible to use our technique on more general transformations if they have a connected-manifold structure and are endowed with an exponential map.

We address the pose estimation problem by assuming that there is a set of correspondences $\mathcal{X} = \{(\mathbf{X}_i, \mathbf{X'}_i)\}$ coming from two different instants of time of the same scene. From now on $\mathbf{X}_i$ should be understood as the i-th 3D point represented by a homogeneous vector, and $\mathbf{X}$ will be a $N \times 4$ matrix containing N points in homogeneous coordinates as row vectors. How to calculate these correspondences is out of the scope of this paper; we will suppose that they were obtained from a stereo rig. Several ways of performing this process can be consulted in [13] and [7]. Hence, given a set of correspondences, the problem of registering two rigid scenes in 3D is classically formulated as finding the transformation $\mathcal{T}_r$ that best fit the data. This is expressed as a least-squares problem:

$$\mathcal{T}_r{}^* = (R,T)^* = \underset{\mathcal{T}_r \in SE(3)}{\operatorname{argmin}} \sum_{i=1}^{N} \|h(\mathcal{T}_r, \mathbf{X}_i) - \mathbf{X}'_i\|_{\ell_2} \quad (3)$$

where N is the number of correspondences and h defines a valid way of transforming the points.

The correct estimation of $\mathcal{T}_r$ requires to consider all the constraints imposed by the group $SE(3)$. However, traditional methods, such as the proposed in [2], [3] and [14] proceed in a different way. First, they remove the mean from $\mathbf{X}$ and $\mathbf{X}'$ and reduce the problem to estimate a rotation matrix R. Accordingly, the computation of T is subordinated to the estimation of R, what will lead to incorrect solutions in the presence of noise and errors in the data association. Moreover, the computation of R is first approximated by a transformation $R_0 \notin SO(3)$ obtained by least squares. Then, these approaches try to find the closest element to R_0 in $SO(3)$. This is clearly inappropriate in presence of noise since it will produce inaccurate estimations for T.

Other techniques enforce the fulfilment of all the constraints by making use of Lagrange multipliers [15], something that leads to a harder optimization problem and requires extra computation. An additional drawback comes from the over-parametrization of R as an orthogonal matrix $\mathbb{R}^{3\times3}$. This produces a negative impact on the optimization procedure, having to be replaced by a minimal parametrization.

Fortunately, all these drawbacks can be solved by considering the Lie group structure of $SE(3)$ and its associated Lie algebra $\mathfrak{se}(3)$. The relationship between both spaces allows us to perform the optimization on the vector space $\mathfrak{se}(3)$ and move back to the group when necessary (please, refer to section II for more information).

This aspect has been previously exploited in different works. In [4] and [16] authors apply these concepts on the field of computer graphics. They carry out an optimization process for the estimation of rotations, although they do not consider computational time as a critical aspect. In a different way, [17] shows the application of these ideas to the estimation of the essential matrix $\mathcal{E}$. Although $\mathcal{E}$ does not have structure of Lie group, the same principles apply due to its global structure of Riemannian manifold with an **exp** map.

More recently, novel approaches have focused on the reformulation of this problem as an averaging problem on a manifold. This idea consists of using an initial set of noisy transformations $\{\mathcal{T}_r^1, \mathcal{T}_r^2, \ldots, \mathcal{T}_r^m\}$ and average them all in order to (hopefully) produce a better transformation $\mathcal{T}_r^*$ that is not affected by noise. There, Lie groups and Lie algebras are utilized to guarantee that all the constraints are met. This concept is is tested in [12] for estimating transformations in $SO(3)$ and $SE(3)$. After this work, [18] proposes a more robust approach based on the use of the ℓ_1-norm, which improves the tolerance of these techniques to noise. However, averaging techniques present a clear drawback since they require multiple initial poses to be already available.

Our proposal should be included in the category of optimization on manifolds, such as [4], [16] and [17]. However, our approach differs from these others in some important aspects. Firstly, here we address the problem of estimating transformations on $SE(3)$, while they deal with $SO(3)$ or $\mathcal{E}$. More important, they define update steps along geodesics, curves of minimal length on the manifold. Accordingly, they try to compute gradient directions and step length by using the properties of geodesics. Such a philosophy, although correct, requires to adapt the internal structure of current optimizers to accommodate these changes. In contrast to this, our approach does not require to modify any optimizer, as it is based on a simpler concept. We use the exponential map to transfer solution candidates from $\mathfrak{se}(3)$ to $SE(3)$ and then perform their evaluation on the group. After this, the update step is performed on the algebra by following the direction of the gradient, which is calculated as a function on the group in terms of the algebra parameters. Such a difference allows us to solve the problem by applying a standard optimizer, while keeping our solutions on the manifold.

IV. Optimization on Lie algebras and Lie groups

In order to overcome the aforementioned limitations we address the pose estimation problem by optimizing a cost function that evaluates candidates in the constrained space of $SE(3)$, but "moves" across the vector space of $\mathfrak{se}(3)$. To convert this idea into a practical solution, capable of producing reliable results and running in a short lapse of time, requires a careful design of all its steps. In the following, we describe the structure of our approach step by step, starting by the design of the cost function itself. A summary of the technique is shown in Algorithm. IV.1, at the end of this section.

A. Cost function

We have transformed the optimization problem defined in (3) into an equivalent problem on $\mathfrak{se}(3) \leftrightarrow SE(3)$ spaces.

This is formalized in (4), below:

$$(\omega, v)^* = \underset{(\omega,v)\in\mathfrak{se}(3)}{\operatorname{argmin}} \; \mathcal{C}(\omega, v), \tag{4}$$

where $\mathcal{C}$ is a cost function with domain on the Lie algebra: $\mathcal{C} : \mathfrak{se}(3) \rightarrow \mathbb{R}$. The solution candidates have to minimize the distance between points sets $\mathbf{X}$ and $\mathbf{X}'$, which implies to convert the candidate to its group representation (R, T) as an homogeneous matrix. This is done by introducing the $\mathbf{exp}$ map in the cost function, giving rise to:

$$\mathcal{C}(\omega, v) = \sum_{i=1}^{N} d(\mathbf{exp}(\omega, v)\mathbf{X}_i, \mathbf{X'}_i). \tag{5}$$

Here, $d(\cdot, \cdot)$ is a suitable distance function between two homogeneous points $\mathbf{X'}_i$ and $\mathbf{X''}_i = \mathbf{exp}(\omega, v)\mathbf{X}_i$.

The selection of the function $d(\cdot, \cdot)$ drastically affects the accuracy of solutions and the performance of the entire method. $d(\cdot, \cdot)$ has to minimize the impact that noisy data introduce into the system. This is specially important when the 3D points are obtained from a triangulation process, where errors will increase quadratically with respect to the distance of points to the camera centre [19]. Furthermore, the process of associating points along different views introduces wrong correspondences (outliers), that deviate the function from the actual solution.

In the context of optimization this task is usually carried out by the so called robustified functions. These are functions that try to reduce the influence of those correspondences that produce too large residuals $\epsilon_i = d\left((R, T)\mathbf{X}_i, \mathbf{X'}_i\right)$. The Huber function [20] and the ℓ_1-norm [18] are some instances of this paradigm. In contrast, functions based on the ℓ_2-norm square each of the residuals, what can amplify the influence of the noise.

Accordingly, it would be desirable to make use of a robustified function for our problem. However, it is important to remark that these functions have to perform an individual treatment of each residual in order to weight it appropriately. Therefore, the computational cost of evaluating a candidate solution will be proportional to the amount of input data. This behaviour compromises efficiency in favour of robustness and prevents of using large amounts of data.

This reason led us to choose $d(\cdot, \cdot) = \|\cdot\|^2_{\ell_2}$ as it can offer a good trade-off between robustness and speed. Although this function is not as robust to the presence of noise as the Huber function, it behaves well in practical situations if the data is appropriately treated. But the main reason is that this cost function leads itself to suitable algebraical reductions, which makes possible to evaluate candidate solutions independently of the amount of input data.

B. Data normalization

Before performing the algebraical reduction, it is appropriate to normalize the input data. An appropriate normalization is mandatory in optimization processes [19] [21]. This does not only prevent from converging to deliberately wrong solutions, but also reduces the number of iterations needed by the optimizer. In our case, we use the normalization procedure proposed in [21]; it consists of centering point distributions around zero and make their variance $\sigma^2 = \sqrt{3}$. This is done for both sets of points $\mathbf{X}$ and $\mathbf{X}'$ independently by computing the following transformation:

$$\mathcal{W} = \begin{bmatrix} \frac{N\sqrt{3}}{\sum_{i=1}^{N} \|\mathbf{X_i}\|_{\ell_2}} I_3 & \frac{-\sqrt{3}}{\sum_{i=1}^{N} \|\mathbf{X_i}\|_{\ell_2}} \sum_{i=1}^{N} \mathbf{X_i} \\ \vec{0} & 1 \end{bmatrix}. \tag{6}$$

Two matrices $\mathcal{W}$ and $\mathcal{W}'$ are calculated from $\mathbf{X}$ and $\mathbf{X}'$ respectively. Then, the correspondences are normalized to produce $\tilde{\mathbf{X}}_i = \mathcal{W}\ \mathbf{X}_i$ and $\tilde{\mathbf{X}}'_i = \mathcal{W}'\ \mathbf{X}'_i$ (here $\tilde{\cdot}$ stands for normalized points). In our experiments, the normalization of the data made the algorithm convergence more than ten times faster. As a final consideration, once the optimizer finds a solution $(\tilde{R}, \tilde{T})^*$ in normalized coordinates, the actual solution is recovered as:

$$(R, T)^* = \mathcal{W}'^{-1}(\tilde{R}, \tilde{T})^* \mathcal{W} \tag{7}$$

C. Reduced Measurement Matrix

After the normalization step, we can take advantage of the properties of $\mathcal{C}$ to derive an algebraically equivalent expression that reduces the evaluation of all the correspondences to a single matrix: the Reduced Measurement Matrix, $\boldsymbol{M}$. This concept was introduced by Hartley in [21], although it had been applied in previous works without an explicit mention [2] [3]. Here we have derived a modified version of $\boldsymbol{M}$ that fits in our problem. We base its construction on Lemma 1, below.

Lemma 1. Given a matrix $\mathbf{V}$ that contains N d-vectors $v^{(i)}$ as rows; $\mathbf{V} = \begin{bmatrix} v_1^{(1)} & \cdots & v_d^{(1)} \\ \vdots & \ddots & \vdots \\ v_1^{(N)} & \cdots & v_d^{(N)} \end{bmatrix}$, then it follows that:

$$\sum_{i=1}^{N} \|v^{(i)}\|^2_{\ell_2} = \|\mathbf{V}\|^2_F = \mathbf{tr}(\mathbf{V}^T\mathbf{V}). \tag{8}$$

Here **tr** stands for the matrix trace and the lemma can be easily proved by expanding both sides of the equality.

Accordingly, we can transform our cost function $\mathcal{C}$ into a more compact expression as the derived in (9).

$$\mathcal{C}(\omega, v) = \sum_{i=1}^{N} \|\mathbf{exp}(\omega, v)\tilde{\mathbf{X}}_i - \tilde{\mathbf{X}}'_i\|^2_{\ell_2} = \sum_{i=1}^{N} \|\mathcal{T}_r\tilde{\mathbf{X}}_i - \tilde{\mathbf{X}}'_i\|^2_{\ell_2}$$

$$= \sum_{i=1}^{N} \left\| \left[\mathcal{T}_r \;\middle|\; -I_4\right] \begin{bmatrix} \tilde{\mathbf{X}}_i \\ \tilde{\mathbf{X}}'_i \end{bmatrix} \right\|^2_{\ell_2} \overset{\textbf{(Lemma 1)}}{=} \tag{9}$$

$$\mathbf{tr}\left(\left[\mathcal{T}_r \;\middle|\; -I_4\right] \begin{bmatrix} \tilde{\mathbf{X}} \\ \tilde{\mathbf{X}}' \end{bmatrix} \left[\tilde{\mathbf{X}}^T \;\middle|\; \tilde{\mathbf{X}}'^T\right] \begin{bmatrix} \mathcal{T}_r^T \\ -I_4 \end{bmatrix} \right) =$$

$$\mathbf{tr}\left(\left[\mathcal{T}_r \;\middle|\; -I_4\right] \boldsymbol{M} \left[\mathcal{T}_r \;\middle|\; -I_4\right]^T \right) = \mathcal{C}_M(\omega, v).$$

In this way, we only have to compute $\boldsymbol{M}$ once, before starting the optimization procedure. Then, the optimization will be carried out on $\mathcal{C}_\mathbf{M}$. This variation drastically reduces the time required to perform each iteration, and therefore, the total execution time of the approach. Hence, a larger amount of correspondences can be managed while still

running in real-time. Such a capability is not achievable by RANSAC and its variations, as each of those hypotheses must be checked against the entire set (or a subset) of correspondences to estimate its number of supporters.

D. Optimization stage

It is important to remark some relevant properties before describing the optimization procedure. As stated before, the optimization of $\mathcal{C}_{\mathbf{M}}$ is carried out in $\mathfrak{se}(3)$, a vector space isomorphic with $\mathbb{R}^6$. This function is convex with respect to the parameters of the Lie group (its Hessian matrix is semi-definite positive). However, it is non-convex with respect to $\mathfrak{se}(3)$, what is derived from the fact that any element in $SE(3)$ is represented by infinite elements from $\mathfrak{se}(3)$ [8][9].

In practice this is not a problem since all the elements of $SE(3)$ can be unambiguously reached when considering a bounded region of its tangent space $T_I SE(3)$ (centered on the identity). As all the elements of $SE(3)$ are uniquely mapped to elements of that region, there will be a unique minimum reachable from there. Additionally, any initial point within that region will converge to the optimum. Thus, there is no need to constrain the optimization procedure.

Considering the previous facts, the optimization process is carried out by a first-order method based on a standard non-linear conjugate gradient (NLCG) [22]. The main reason for this is to avoid the explicit calculation of the Hessian matrix performed by second-order techniques. NLCG with the normalized data is enough to reach the optimum very quickly. In fact, in our experiments the optimizer took on average twelve iterations.

E. Calculation of the Jacobian matrix

One of the key elements of our proposal is the fast computation of the Jacobian matrix for $\mathfrak{se}(3)$. Some approaches, as for instance [4] and [17], apply gradient descent methods by computing directional derivatives on the geodesics defined by the $\mathbf{exp}$ map. In contrast to this, we opted for calculating the Jacobian matrix on $\mathbb{R}^6$, as it forms an isomorphism with $\mathfrak{se}(3)$. This variation allows to use standard optimization frameworks, but can lead to an inefficient expression if not done properly.

This is the case of the analytical expression of the Jacobian for the proposed cost function $\mathcal{C}_{\mathbf{M}}$; it is very complex, since we need to account for all the terms introduced by $\mathbf{exp}$. This complexity makes the use of the analytical Jacobian prohibitive in a real-time framework. On the other hand, numerical approximations are more efficient alternatives and, in practice, they can lead to the same accuracy.

According to that, we calculate a fast approximation of the Jacobian matrix based on the partial derivatives of the Taylor series of $\mathbf{exp}$. The idea works as follows. Given the cost function $\mathcal{C}_{\mathbf{M}}$ we have that:

$$\mathcal{C}_{\mathbf{M}}(\omega, v) = \mathbf{tr}\left(\left[\mathcal{T}_r \mid -I_4\right] \boldsymbol{M} \left[\mathcal{T}_r \mid -I_4\right]^T\right) = \mathbf{tr}\left(\left[\mathbf{exp}(\omega, v) \mid -I_4\right] \boldsymbol{M} \left[\mathbf{exp}(\omega, v) \mid -I_4\right]^T\right). \tag{10}$$

From where it follows that:

$$\begin{aligned}
\frac{\partial \mathcal{C}_{\mathbf{M}}}{\partial(\omega, v)_{m,n}} &= \frac{\partial \mathbf{tr}\left(\left[\mathbf{exp}(\omega, v) \mid -I_4\right] \boldsymbol{M} \left[\mathbf{exp}(\omega, v) \mid -I_4\right]^T\right)}{\partial(\omega, v)_{m,n}} \\
&= \frac{\partial \left[\mathbf{exp}(\omega, v) \mid -I_4\right]_{i,j}}{\partial(\omega, v)_{m,n}} \boldsymbol{M}_{j,k} \left[\mathbf{exp}(\omega, v) \mid -I_4\right]^T_{k,i} \\
&+ \left[\mathbf{exp}(\omega, v) \mid -I_4\right]_{i,j} \boldsymbol{M}_{j,k} \frac{\partial \left[\mathbf{exp}(\omega, v) \mid -I_4\right]^T_{i,j}}{\partial(\omega, v)_{m,n}} \\
&= \left[\tfrac{\partial \mathbf{exp}(\omega, v)}{\partial(\omega, v)_{m,n}} \mid 0\right]_{i,j} \boldsymbol{M}_{j,k} \left[\mathbf{exp}(\omega, v) \mid -I_4\right]^T_{k,i} \\
&+ \left[\mathbf{exp}(\omega, v) \mid -I_4\right]_{i,j} \boldsymbol{M}_{j,k} \left[\tfrac{\partial \mathbf{exp}(\omega, v)}{\partial(\omega, v)_{m,n}} \mid 0\right]^T_{k,i} \\
&= \left[\mathbf{exp}(\omega, v) \mid -I_4\right]_{i,j} \left(\boldsymbol{M}^T_{j,k} + \boldsymbol{M}_{j,k}\right) \left[\tfrac{\partial \mathbf{exp}(\omega, v)}{\partial(\omega, v)_{m,n}} \mid 0\right]^T_{k,i} \\
&= \mathbf{tr}\left(\left[\tfrac{\partial \mathbf{exp}(\omega, v)}{\partial(\omega, v)_{m,n}} \mid 0\right] \left(\boldsymbol{M} + \boldsymbol{M}^T\right) \left[\mathbf{exp}(\omega, v) \mid -I_4\right]^T\right).
\end{aligned} \tag{11}$$

Note that the derivative has been calculated by using abstract indices and due to this the trace is represented as a contraction of the form $A_{i,j} \cdots B_{k,i}$. When the final expression is obtained the trace appears back. Furthermore, this expression shows that the Jacobian of $\mathcal{C}_{\mathbf{M}}$ can be easily calculated from the partial derivatives of $\mathbf{exp}$ and a matrix that results from $\left(\boldsymbol{M} + \boldsymbol{M}^T\right) \left[\mathbf{exp}(\omega, v) \mid -I_4\right]^T$. The partial derivatives of $\mathbf{exp}$ have been approximated from the Taylor's series for $\mathbf{exp}$, such that:

$$\frac{\partial \mathbf{exp}(S)}{\partial(\omega, v)} = \frac{\partial \left(\sum_{k \geq 0} \frac{S^k}{k!}\right)}{\partial(\omega, v)} = \sum_{k \geq 1} \frac{1}{k!} \left(\frac{\partial S}{\partial(\omega, v)} S^{k-1} + S \frac{\partial S^{k-1}}{\partial(\omega, v)}\right), \tag{12}$$

where S is an element of $\mathfrak{se}(3)$ in its matrix form (see section II). One of the advantages of this expression is that the k-th term of the series can be quickly computed from the $(k-1)$-th term, giving an efficient way of computing the Jacobian of $\mathbf{exp}$. Furthermore, the convergence of this approximation is very fast. When twenty terms are used, the difference between this approximation and the actual Jacobian remains below 10^{-11}.

F. Refinement with the supporters set

This final step is borrowed from RANSAC [5] and consists on refining a given solution $(\tilde{R}, \tilde{T})^*$ from its set of supporters $\mathcal{X}_S = \{(\mathbf{X_i}, \mathbf{X'_i})\} \subset \mathcal{X}$. In our approach this is done by defining a threshold $\tau < 0.3$ and then selecting all the correspondences that meet $\|(\tilde{R}, \tilde{T})^* \tilde{\mathbf{X}}_i - \tilde{\mathbf{X}}'_i\|_{\ell_2} < \tau$. Next, a refined model $(\tilde{R}, \tilde{T})^*_S$ is computed from $\mathcal{X}_S$ by using a fast least-squares method based on singular value decomposition, as showed in [3].

The step here presented is optional, although it produces an important improvement of the solution. From a practical point of view, our recommendation is to apply it when the

amount of input data is not too large (around 500 points); otherwise this would affect the global speed of the method.

Algorithm IV.1: $\mathfrak{se}(3) \leftrightarrow SE(3)$ OPTIMIZATION()

1. Compute $\mathcal{W}$ and $\mathcal{W}'$ from $\mathbf{X}$ and $\mathbf{X}'$, as in (6)
2. Generate $\forall i\ \tilde{\mathbf{X}}_i \leftarrow \mathcal{W}\mathbf{X}_i$, $\tilde{\mathbf{X}}'_i \leftarrow \mathcal{W}'\mathbf{X}'_i$
3. Form $\boldsymbol{M}$ from $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{X}}'$ by applying (9)
4. Assign a random value to $x \in \mathbb{R}^6$, such that $\|x\| < \pi$
5. Apply gradient descent to optimize the cost function in (10), given the Jacobian matrix of (11) and (12), until the algorithm converges to the solution x^*
6. Use $\mathbf{exp}$ to convert x^* into $(\tilde{R}, \tilde{T})^*$
7. Estimate $(\tilde{R}, \tilde{T})^*_S$ from the inliers of $(\tilde{R}, \tilde{T})^*$ as in sec. IV.F
8. Denormalize $(\tilde{R}, \tilde{T})^*_S$ as in (7) to produce $(R, T)^*_S$

V. EXPERIMENTAL RESULTS

This section presents a comparison of our proposal and two well known representatives of the estimation methods: RANSAC and Horn's algorithm. RANSAC is still one of the most popular techniques to perform robust estimation, even after the large amount of variations found in the literature. On the other hand, we selected Horn's algorithm to represent classical least-squares techniques and to show that our approach produces more accurate results in comparison to such methods. All these techniques are coded and tested in MATLAB. The RANSAC implementation used in tests is part of the RANSAC Toolbox for MATLAB [23]. This is something to bare in mind, as the *C* counterpart of these methods will run much faster.

The experiments were carried out on synthetic cases, but using realistic configurations and conditions. The factors analysed here are accuracy, robustness against noise and outliers and finally computational performance. All the tests were carried out several times in order to produce average results. Our experimentation setup is common for all the experiments and works as follows. Each experiment is carried out by first creating two clouds of points $\mathbf{X}$ and $\mathbf{X}'$ with N correspondences perfectly matched. We distinguish here between two cases; the first case, in which the clouds are generated from random synthetic points within a given 3D volume (around 100 metres large); and the second case, which is based on real 3D scenarios obtained from the urban dataset in [1]. Fig. 1 is an example of the second class.

Then, regardless the origin of the data, we add a certain amount of noise Δ to each point independently. Δ simulates the actual noise caused by sensors during the acquisition process and by the 3D triangulation method. This is done by defining the parameters of a virtual stereo rig, which in our experiments consisted of: focal length of both cameras, $f = 350\ px$; baseline of the device, $B = 0.35\ m$; calibration error, $c_e = 0.06\ px$. From these parameters we defined the noise of a point as being $\Delta\mathbf{X_i} = [\frac{\mathbf{X_i}(z)c_e}{f}, \frac{\mathbf{X_i}(z)c_e}{f}, \frac{\mathbf{X_i}(z)^2 c_e}{fB}]$. Additionally, we further affect the correspondences set $\mathcal{X} = \{(\mathbf{X}_i, \mathbf{X'}_i)\}$ by substituting a percentage P_o of the associations with incorrect matchings. This failure matchings are controlled to avoid producing associations between points that are too far away in the scene. In our experiments this maximum distance is 10 metres.

Following this configuration, we tested the robustness of the methods against an increasing percentage of outliers in $\mathcal{X}$. This was done by generating different sets of $N = 160$ correspondences (all of them affected by an amount of noise Δ) and containing an increasing percentage of outliers $P_o \in [0, 50]\%$. Then, the error of each estimation (R, T) is measured against the ground truth $(R, T)^*_G$ as the Euclidean distance between matrices $\mathbf{E} = \|(R, T) - (R, T)^*_G\|_{\ell_2}$. The results of this first experiment are showed by Fig. 2**(a)**.

These results show that the average error of RANSAC stays below 0.09 units, even when the 50% of the data contains outliers. In contrast, Horn's algorithm solutions have errors that go far beyond 10 units, which in practice means an unusable model. Our technique produces estimations with average error of 0.4 units, which is still comparable to those produced by RANSAC. Note that, although Horn's method and our approach are both based on a ℓ_2-norm, the optimization of this norm assuming an underlying manifold structure produces much better results.

Fig. 2**(b)** shows the results for the same experiment, but this time from the viewpoint of the execution time. Notice that the vertical axis is shown in logarithmic scale. This is done to permit a correct visualization of results for the three methods. As shown, our method clearly outperforms RANSAC in this aspect. The average time of our method is 35 ms and remains stable even when the percentage of outliers increases. In contrast, RANSAC timings get drastically affected by such increase, even exceeding 200 seconds of computation. There are two explanation for this phenomenon. Firstly, due to the noise, RANSAC cannot estimate a proper bound of the inliers set, what makes it to compute far more hypotheses than the established by theoretical limits. The second reason has to do with the bad performance that large loops present in MATLAB. In any case, even with a faster version of RANSAC, our method tests less hypotheses and can evaluate each one much faster than RANSAC. In turn, Horn's algorithm produce the best times due to its non-iterative nature (less than 5 ms).

Our second experiment tests the influence that the amount of data has in the estimation process. For this case we fixed $P_o = 25\%$, value that we consider in concordance with the values found in real scenes. As presented in Fig. 3**(a)**, the inclusion of more data has a positive influence on the accuracy of the results, even when the data is proportionally affected by noise and outliers. This phenomenon is due to the law of large numbers and was previously pointed out in [24]. In this case, when 1000 correspondences are used, the error of our method decreases from 0.4 units to 0.2 units, while RANSAC moves from 0.09 units to 0.06 units. Horn's algorithm also improves, but its error is still too large.

On the other hand, Fig. 3**(b)** shows execution times asso-

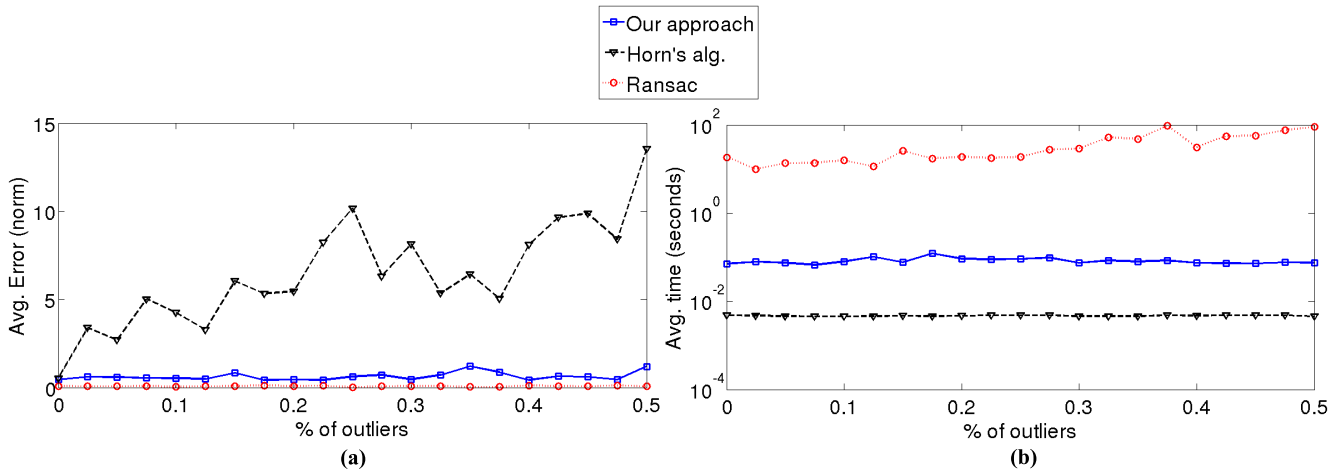

Fig. 2: Evaluation of the impact of different percentages of outliers in the methods performance ($N = 160$). **(a)** average error $\|(R,T) - (R,T)^*_G\|_{\ell_2}$; **(b)** computational time results given in a logarithm scale.

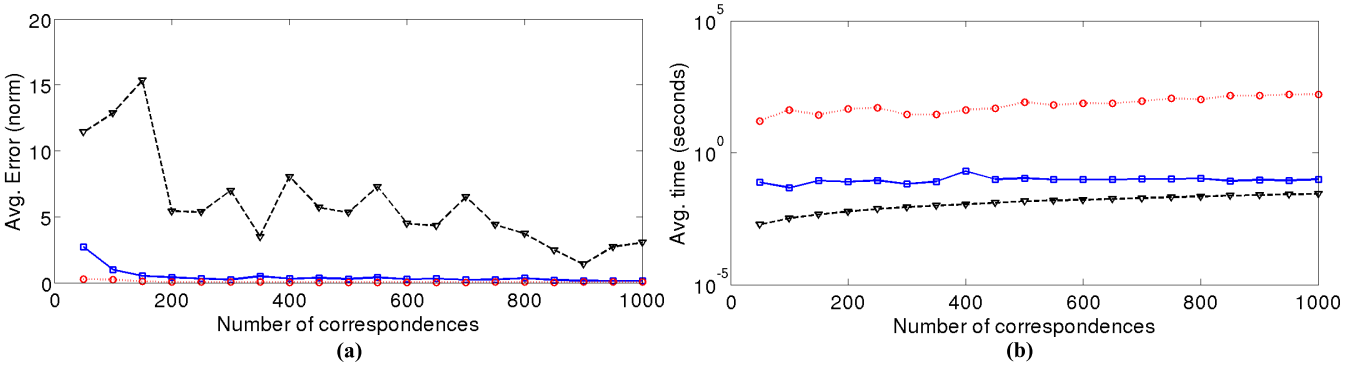

Fig. 3: Evaluation of the impact of the amount of input data in the methods performance ($P_o = 25\%$). **(a)** average error $\|(R,T) - (R,T)^*_G\|_{\ell_2}$; **(b)** computational time results given in a logarithm scale.

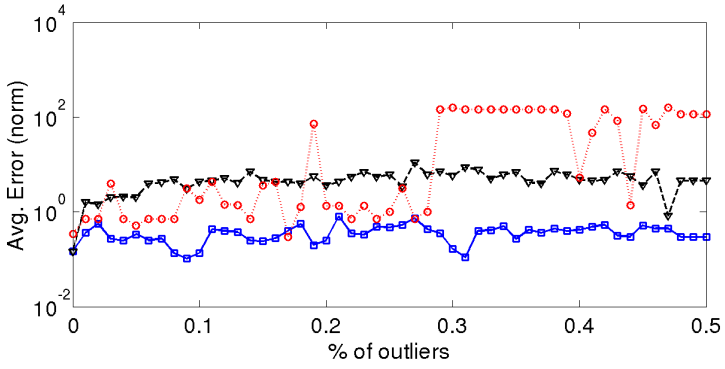

Fig. 4: Evaluation of the methods accuracy when restricted to perform in less than 20 ms ($N = 200$).

ciated with this experiment. Here we see that the proposed technique suffers much less than RANSAC when the input data is increased. We obtain a running time of 60 ms for the extreme case of 1000 correspondences. This is achieved by the use of the Reduced Measurement Matrix, that makes the optimization procedure independent of the data size. However, there are three steps of our method that are directly affected by the rise of the input data: the normalization process, the formation of $\boldsymbol{M}$ and the final refinement with the supporters set. The two former steps are computationally very efficient and can work well with large amounts of data. However, the refinement should be deactivated for such situations (when $N > 800$), in order to keep a low computational time. As shown in this plot, the computational time of our technique tends to the one of Horn's algorithm, although our accuracy is much better. Furthermore, RANSAC times increase linearly with the amount of data, reaching 160 seconds when tested with 1000 correspondences; i.e, more

than 2000 times slower than our approach.

Following with previous results, we decided to evaluate the accuracy and the robustness of these methods by testing the best solution they can produce in a given time. To do this, we fixed a maximum time of 20 ms and set the number of correspondences $N = 200$. The results of this experiment are shown in Fig. 4. Under these circumstances our approach produces better results than RANSAC and Horn's algorithm in the majority of the cases. Moreover, if the maximum time is further reduced or the number of correspondences is further increased, RANSAC performance decays.

These results, although drawn from synthetic experiments, are very encouraging. They show that proposal can deal with noise and outliers fairly well. Our MATLAB implementation runs in 20 ms for standard cases, and deal with a thousand correspondences in just 60 ms. We hope that a future C implementation will be able to run in a couple of milliseconds for standard conditions. Furthermore, the accuracy of our results are similar to those of RANSAC, even outperforming it in some situations. However, this critical aspect still has to be evaluated on real data. We believe that these results might be further improved by managing a larger amount of data. As shown, significant increments of the input data are well managed by our method, and may be used as a new alternative to achieve better estimations.

VI. Conclusion and Future Work

We have presented a novel technique for performing pose estimation between two views in the case of rigid transformations in 3D. We based our approach on the formalism of Lie-groups and Lie-algebras in order to deal with the non-linear constraints of $SE(3)$ during the optimization process. This was done by following the principle "optimize on the linear space of the Lie-algebra and then move to the manifold space of the Lie-group". As showed, a pipeline based on these ideas can produce accurate results without sacrifying real-time performance. A fast implementation of the Jacobian matrix as well as the utilization of the RMM to compress the input data play a critical role in achieving this performance.

Experimental evaluation on synthetic cases showed that the accuracy of our approach is comparable to that of RANSAC (and much better than that of the Horn method), while our execution time is much lower. In cases where the percentage of outliers increases or the amount of input data is too large, our method can still produce good results in real-time. When both approaches are constrained to produce a solution in less than 20 ms, our method produced the most accurate results.

We consider interesting to keep investigating the capability of this approach for managing large amounts of data. This might open a door for increasing the accuracy of the results by providing more input data —which is nowadays available due to high-resolution cameras and dense stereo techniques. We will also evaluate the behaviour of this approach on real imagery and its application to a final VSLAM system.

ACKNOWLEDGMENT

This work has been supported by the Universitat Autònoma de Barcelona, the FundaciónN Séneca 08814PI08 and the Spanish government, by the projects FIS201129813C0201; TRA201129454C0301 (eCo-DRIVERS); TIN201125606 (SiMeVé) and TIN201129494C0302 (FireWATCHER)

References

[1] K. Lai and D. Fox, "3D laser scan classification using web data and domain adaptation," in *Proc. of Robotics: Science and Systems*, Seattle, USA, Jun. 2009.

[2] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America*, 1987.

[3] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, 1987.

[4] S. Krishnan, P. Y. Lee, J. B. Moore, and S. Venkatasubramanian, "Global registration of multiple 3D point sets via optimization-on-a-manifold," in *Proc. Eurographics symposium on Geometry processing*. Eurographics Association, 2005.

[5] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, 1981.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Trans. Pattern Anal. Mach. Intell.*, Jun. 2007.

[7] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme," in *Proc. IEEE Intell. Veh. Symp.*, June 2010.

[8] J. Gallier, "Notes on differential geometry and Lie groups," University of Pennsylvania, Department of Computer and Information Science, Tech. Rep., Feb. 2012.

[9] K. Kanatani, *Group-Theoretical Methods in Image Understanding*. Springer-Verlag, 1990.

[10] J. M. Lee, *Introduction to Topological Manifolds*. Springer, 2011.

[11] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ: Princeton University Press, 2008.

[12] V. Govindu, "Lie-algebraic averaging for globally consistent motion estimation," in *Proc. IEEE Comput. Vision and Pattern Recog.*, Jun. 2004.

[13] A. Comport, E. Malis, and P. Rives, "Accurate quadrifocal tracking for robust 3D visual odometry," in *Proc. IEEE Int. Conf. on Robot. and Automat.*, Apr. 2007.

[14] K. Kanatani, "Analysis of 3-D rotation fitting," *IEEE Trans. Pattern Anal. Mach. Intell.*, May 1994.

[15] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. on Pattern Analysis and Machine Intell.*, 1991.

[16] S. Krishnan, , P. Y. Lee, J. B. Moore, and S. Venkatasubramanian, "Optimisation-on-a-manifold for global registration of multiple 3D point sets," *Int. J. Intell. Syst. Technol. Appl.*, Jun. 2007.

[17] Y. Ma, J. Košecká, and S. Sastry, "Optimization criteria and geometric algorithms for motion and structure estimation," *Int. J. Comput. Vision*, Sep. 2001.

[18] R. Hartley, K. Aftab, and J. Trumpf, "L1 rotation averaging using the weiszfeld algorithm," in *Proc. IEEE Conf. Comput. Vision and Pattern Recog.* Washington, DC, USA: IEEE Computer Society, 2011.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2003.

[20] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust Statistics: The Approach Based on Influence Functions (Wiley Series in Probability and Statistics)*. New York: Wiley-Interscience, 2005.

[21] R. I. Hartley, "Minimizing algebraic error in geometric estimation problems," in *Proc. Int. Conf. Comput. Vision*, Washington, DC, USA, 1998.

[22] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Pittsburgh, PA, USA, Tech. Rep., 1994.

[23] M. Zuliani, "RANSAC toolbox for matlab," [web page] http://www.mathworks.com/matlabcentral/fileexchange/18555, Nov. 2008, [Last accessed on September 16th 2012].

[24] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Real-time monocular slam: Why filter?" in *Proc. IEEE Int. Conf. on Robot. and Automat.*, Anchorage, Alaska, US, 2010.