Eigen-Factors: Plane Estimation for Multi-Frame and Time-Continuous Point Cloud Alignment

Gonzalo Ferrer

***Abstract*— In this paper, we introduce the Eigen-Factor (EF) method, which estimates a planar surface from a set of point clouds (PCs), with the peculiarity that these points have been observed from different poses, i.e. the trajectory described by a sensor. We propose to use multiple Eigen-Factors (EFs) or different planes' estimations, that allow to solve the multi-frame alignment over a sequence of observed PCs. Moreover, the complexity of the EFs optimization is independent of the number of points, but depends on the number of planes and poses. To achieve this, a closed-form of the gradient is derived by differentiating over the minimum eigenvalue with respect to poses, hence the name Eigen-Factor. In addition, a time-continuous trajectory version of EFs is proposed. The EFs approach is evaluated on a simulated environment and compared with two variants of ICP, showing that it is possible to optimize over all point errors, improving both the accuracy and computational time. Code has been made publicly available.**

I. Introduction

Robots are capable of gathering information while moving which stands apart robotics from other related fields. In this paper, we present a new method, Eigen-Factors (EFs), for calculating the alignment over a sequence of observations or point clouds (PCs) which complexity is independent of the number of points. We will refer to our proposed method as Eigen-Factors (EFs), since multiple planes are required to calculate a well-posed alignment problem.

The problem of *alignment* is typically formulated between a pair of poses and most of the contributions are from the graphics community [1]–[7]. All these methods and variants are extensively used in multiple robotic applications for 3D alignment, but under slightly different conditions for what they were originally designed, e.g. sensing error, density or dispersion of points.

The advances on depth sensors have been spectacular; however, they still are not exempt from limitations. Current lidars present non-uniform beam patterns which increase dispersion on PCs. Point density on lidars decreases with distance resulting in regions with few sampled points. On the other hand, mass-produced RGBD cameras output dense with low accuracy information. These limitations can be alleviated by moving the sensor, leading to a richer sampling of the same surfaces from different points of view.

The challenge is how to process all these sensed data while maintaining computational budget affordable. Our approach is capable of aggregating all information while drastically reducing the complexity. To achieve that, EFs require a preprocessing step for semantic segmentation of planes, which are fairly common in indoors, urban and semi-urban environments. Not surprisingly, planes have been used in multiple works in robotics [8]–[12]. EFs do not require explicit parametrization of the planes, they only require points and sensor poses. In this regard, EFs behave as non-parametric landmarks of planes.

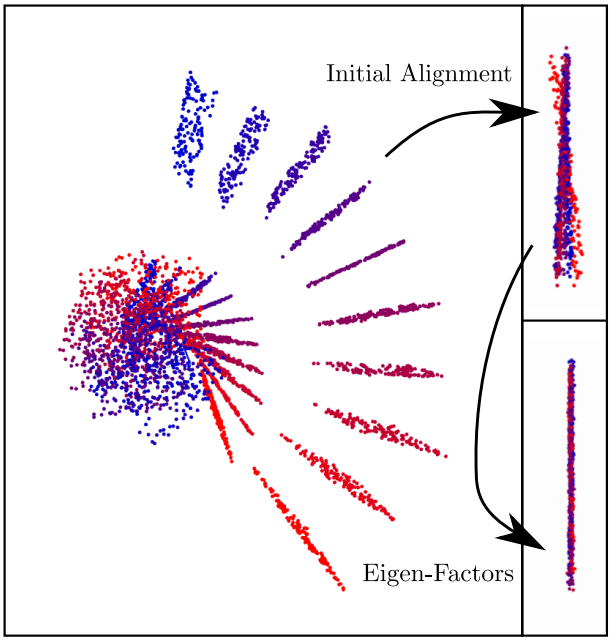

Fig. 1. *Left*: A sequence of point clouds before any alignment. Three planes are observed while moving under a strong rotation. *Right-Top*: The initial alignment of a single plane on a perpendicular view. *Right-Bottom*: The effect of EFs minimizing the plane error and optimizing the trajectory.

The main idea behind EFs is to re-formulate *plane estimation* to reduce the error in fitting planes and optimize over the corresponding eigenvalue w.r.t. sensor poses to improve the accuracy of the trajectory. Furthermore, EFs are able to store a compact representation of points belonging to the same plane and process them without any loss of information.

The contributions are listed below:

- EFs are a reformulation of plane estimation for multi-frame PC alignment. EFs' complexity is independent of the number of points.
- Closed-form derivation of the EFs' gradients using $SE(3)$ and Lie algebra.
- A time-continuous derivation of EFs using interpolation in the manifold.

The author is with Skolkovo Institute of Science and Technology. g.ferrer@skoltech.ru.

This research is based on the work supported by Samsung Research, Samsung Electronics.

In addition, our formulation is easily translated to the 2D domain, where instead of planes, Eigen-Factors estimate lines to optimize a 2D robot trajectory.

II. Prior work

The seminal work of Besl and McKay [1], Iterative Closest Point (ICP), is one of the most influential papers on PC alignment or registration. ICP finds the closest point in Euclidean distance over either pair of point clouds or other surface representations. This work sets the grounds for state of the art methods which are using similar approaches due to their simplicity and their performance.

PC alignment can also be calculated with direct methods such as the works of Horn [2] using quaternions or Arun et al. [3] and Umeyama [4] using the SVD factorization. Chen and Medioni [5] introduce a point-to-plane approach to ICP which improves over point-to-point, and the work of Zhang [6] on curves and surfaces.

Multiple variations for point association are surveyed by Rusinkiewicz and Levoy [7]. Most of the techniques are meant for *dense* point clouds, where the density and dispersion on the sampled points is similar to all regions of the point cloud, plus there are numerous points for surface estimation. These different association techniques are essential for a correct convergence of the algorithm. No algorithm can converge to a solution with poor data, hence EFs use all available data.

The robotics community is also interested in scan-matching showing large basins of convergence, as the work by Olson [13], although the same technique is not easily transferred into 3D.

Segal et al. [9] introduce the generalized-ICP (GICP) which effectively seeks for plane-to-plane relations between pairs of points. This is achieved by proposing *virtual* covariances to emulate planes. By doing this, the association distance can be increased, searching for further correspondences than ICP. Serafin and Grisetti [10] propose a variant of GICP which in addition takes into account the normal alignment (NICP). These ICP variants are a step forward into exploiting local geometry without explicitly assuming it. Nevertheless, these methods need to process all points.

Other 3D alignment approaches consider probabilistically point associations as an improved input for ICP, such as the works of Hahnel and Burgard [14] or Armesto et al. [15]. On the other end of the spectrum are 3D descriptors [16], [17] but they are not the default approach as is the case for images.

SLAM [18], [19] also proposes the use of planar features [8], [12], [20], [21] or a mixture of points and planes [11]. Most of these approaches suffer when the number of planes is insufficient, thus Visual-Intertial SLAM based on planes [22], [23].

III. Background

A. Plane estimation

In general, a plane can be determined by a normal vector η, and the plane distance to the origin d, where $\pi = [\eta^\top, d]^\top$. In total 4 variables and 1 constraint. A point $p = [x, y, z]^\top$ belongs to the plane π if:

$$\eta^\top p + d = 0 \qquad \text{or} \qquad \pi^\top \begin{bmatrix} p \\ 1 \end{bmatrix} = \pi^\top \tilde{p} = 0, \tag{1}$$

either for points expressed in Cartesian coordinates p or in homogeneous coordinates $\tilde{p}$.

We are mostly concerned with 3-dimensional planes, however, there are different methods to estimate them given a set of N points $P = \{p_n\}$ sampled from the same geometric plane π. The most popular plane estimation method, which we refer as the *centered method*, uses a centered set of points. The expectation of these points, according to (1), should be

$$\begin{aligned} E\{\eta^\top p + d\} &= 0 \\ E\{\eta^\top p\} &= -d. \end{aligned}$$

Therefore, we can write the optimization objective

$$\begin{aligned} &\min_\eta \left\{ \sum_{n=1}^{N} ||\eta^\top p_n + d||^2 \right\} \\ &\min_\eta \left\{ \sum_{n=1}^{N} ||\eta^\top p_n - E\{\eta^\top p\}||^2 \right\} \\ &\min_\eta \left\{ \sum_{n=1}^{N} ||\eta^\top (p_n - E\{p\})||^2 \right\} \\ &\min_\eta \left\{ \sum_{n=1}^{N} \eta^\top (p_n - E\{p\})(p_n - E\{p\})^\top \eta \right\} \\ &\min_\eta \left\{ \eta^\top C \eta \right\}, \end{aligned} \tag{2}$$

where the matrix $C \in \mathbb{R}^{3\times3}$ resembles an inner product of centered data or a covariance matrix. The well-known solution to (2) corresponds to the eigenvector associated with the minimum eigenvalue of C. The eigendecomposition is coincident with the Singular Value Decomposition (SVD) for symmetric and semi-positive definite matrices, and by construction C is. In the second step, the parameter d is calculated as $d = -\eta^\top E\{p\}$.

The disadvantage is that every time a new sample is added or modified, for instance as a result of a pose being updated (see below), C is modified as well and all calculations should be carried out again.

There exists an alternative method, the *homogeneous method*, that allows to calculate the plane parameters π without centering the data points:

$$\min_\pi ||\pi^\top \tilde{P}||^2 = \min_\pi \begin{bmatrix} \eta \\ d \end{bmatrix}^\top \underbrace{\tilde{P}\tilde{P}^\top}_{S} \begin{bmatrix} \eta \\ d \end{bmatrix}, \tag{3}$$

where $\tilde{P}$ is the $4 \times N$ matrix corresponding to N stacked homogeneous points. The solution of (3) is calculated again using the SVD decomposition of the 4×4 matrix S. If we seek a plane π as defined in (1), it is necessary to scale this solution such that the first three elements for η are unitary. The great advantage is that S allows for incremental updates.

There is a third approach to estimating a plane from a set of points, and it is based on the principle of orthogonality:

$$\eta \propto E\{(p_{n'} - p_n) \times (p_{n''} - p_n)\}. \tag{4}$$

The work of Klansing et al. [24] reports on the performance of these and other methods for plane estimations.

Lines in a 2-dimensional space are the equivalent for planes in 3D. The homogeneous equation of a line is expressed as

$$[m_x, m_y, d][x, y, 1]^\top = 0, \tag{5}$$

from where we could follow an analogous derivation of Eigen-Factors in 2D.

B. Rigid body transformations and its Lie Algebra

All possible matrix rotations in 3D (generalizable to any dimension) are included in the special orthogonal group $SO(3) = \{R \in \mathbb{R}^{3\times3} \,|\, RR^\top = I \wedge det(R) = 1\}$. Similarly, all possible rigid body transformation (RBT) matrices conform the special Euclidean group

$$SE(3) = \left\{ T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \,|\, R \in SO(3) \wedge t \in \mathbb{R}^3 \right\}, \tag{6}$$

which is the result of a rotation and a translation. The group operation is the matrix multiplication.

The solution to the alignment of a pair of point clouds is a rigid body transformation. The Lie algebra $\mathfrak{se}(3)$ associated with the group of RBT $SE(3)$ represents the group infinitesimal RBT around a given pose. There exist operators that relate both groups. In particular, the exponent operator $\exp : \mathfrak{se}(3) \to SE(3)$ and the logarithm $\ln : SE(3) \to \mathfrak{se}(3)$. The matrix of generators of the $\mathfrak{se}(3)$ group is

$$\xi^\wedge = \begin{bmatrix} w^\wedge & v \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -w_3 & w_2 & v_1 \\ w_3 & 0 & -w_1 & v_2 \\ -w_2 & w_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{7}$$

where there are 6 elements on the 4×4 matrix of generators. We will make use of this equation for deriving the gradients of the eigenvalues.

The vee $^\vee$ and hat $^\wedge$ operators simply encode (7) into a vector, which space is called the manifold and from the manifold back to the Lie group. One can map a RBT $T \in SE(3)$ to $\xi \in \mathbb{R}^6$ by $\xi = \ln(T)^\vee$ and vice-versa $T = \exp(\xi^\wedge)$. We will follow a more compact convention $\xi = \text{Ln}(T)$ and $T = \text{Exp}(\xi)$. In general, this mapping is surjective, but if $||w|| < \pi$, then we can consider it bijective.

One of the main advantages of using the Lie algebra associated with $SE(3)$ is that differentiation becomes enormously simplified.

For updating a pose ξ, we will follow the left-hand side update convention. Let the operator $\oplus$ be

$$\Delta\xi \oplus \xi = \text{Exp}(\Delta\xi)\text{Exp}(\xi). \tag{8}$$

This topic is vast and well documented. We just reviewed those concepts that are used on the sections below. For a more complete discussion on Lie algebra and its applications please check [25]–[28].

IV. Approach

A. Problem formulation

Given a set of observations, or point clouds $P_t = \{p_t^n\}$ during a time interval $t \in [1, H]$, the problem is to estimate the trajectory of 3D poses $\xi_t \in SE(3)$ from where these observations have been taken, such that the trajectory minimizes certain objective.

This statement is formulated as

$$\min_{\xi} J(\xi_1, \ldots, \xi_H), \tag{9}$$

where $\xi = [\xi_1^\top, \ldots, \xi_H^\top]^\top$ is the column vector containing all poses of the trajectory. In the following sections, we will define what this objective J is and how to minimize this expression.

B. Eigen-Factors

A single Eigen-Factor is the optimal estimation of a plane, given a set of points P_t observed from different poses ξ_t. From a single Eigen-Factor, or plane estimation, the alignment problem is ill-posed. On the other hand, multiple Eigen-Factors (EFs) result in a well-posed problem, if some conditions are met. In this section we will derive the solution for a single EF, bearing in mind that the alignment problem requires several of them, therefore we named our method Eigen-Factors (EFs).

From the *homogeneous method* in (3) one can derive an expression that considers the error from each point and the plane π. Now, each of these (homogeneous) points is observed from a different reference frame, such that

$$\min_{\pi} \sum_{n=1}^{N} ||\pi^\top T_t \tilde{p}_t^n||^2, \tag{10}$$

where T_t corresponds to the transformation associated with some pose ξ_t. This equation is equivalent to (3) with all points $\{\tilde{p}_t^n\} = \tilde{P}_t$ observed at time t, transformed by T_t. Thus, we can rewrite this summation as a matrix product of $\tilde{P}_t$ matrices

$$\min_{\pi} \pi^\top \underbrace{T_t \tilde{P}_t \tilde{P}_t^\top T_t^\top}_{Q_t} \pi, \tag{11}$$

where the matrix Q_t is a 4×4 matrix. It could also be written as $Q_t = T_t S_t T_t^\top$, using a similar notation than in (3). The solution to this problem, which is solved by the SVD, is a plane π which parameters are roughly speaking *rotated* from a local coordinate system to a different one through the transformation T_t. Note that Q_t requires only *once* the calculation of the squared term S_t obtained from all raw homogeneous points. Then, updating the matrix Q_t takes only two multiplications. There is no need to store all points, only the matrix S_t. Thus, the complexity no longer depends on the number of points N, but on the number of planes and poses. The number of points $N(t)$ may vary with t and there is no information loss.

For multiple poses, the overall error is accounted as

$$\min_{\pi} \sum_{t=1}^{H} \sum_{n=1}^{N(t)} ||\pi^\top T_t \tilde{p}_t^n||^2, \tag{12}$$

which can be grouped into

$$\min_{\pi} \sum_{t=1}^{H} \pi^\top T_t S_t T_t^\top \pi. \tag{13}$$

The plane vector π is independent of the time index t so it moves out of the summation and the 4×4 matrix $Q = \sum Q_t$ provides the solution to the plane estimation problem. More concretely, the accumulated matrix Q is

$$Q = \sum_{t=1}^{H} Q_t = S_1 + {}^1T_2 S_2 {}^1T_2^\top + \ldots + {}^1T_H S_H {}^1T_H^\top, \tag{14}$$

where we have chosen a fixed reference frame at the initial pose to be $T_1 = I$ and all the remaining transformations relate to frame 1.

The equivalent derivation obtained for the *centered method* for plane estimation (2), requires to recalculate the mean after each alteration/update of T_t, which makes the classical plane estimation a much less efficient option.

In addition, we could construct a weighted squared matrix $S = \tilde{P} W \tilde{P}^\top$, if there was a need to penalize points. In this work, we consider that all points are equal, in other words, they are the output of the same sensor, hence we set W to the scaled identity $\frac{1}{\sigma_z^2} I$, where σ_z is the standard deviation from sensing depth.

The error corresponding to the plane estimation is

$$\min_{\pi}(\pi^\top Q \pi) = \lambda_{min}(Q), \tag{15}$$

which equals the minimum eigenvalue of Q. It is important to note that Q depends on the trajectory ξ, so we can now define the cost function $J(\xi) = \pi^\top Q \pi$. However, we are interested in the following problem:

$$\{\xi_1, \xi_2, \ldots, \xi_H\} = arg\min J(\xi). \tag{16}$$

which does not include the plane parameter π on the estate variables to be estimated. The plane is just a mean to connect all poses in the trajectory that have observed the same geometric entity (sampled points). As an implicit result of evaluating the current trajectory ξ we obtain the plane parameters π, which are not required as state variables at all, thus EF is a non-parametric landmark. This is another advantage of the EFs method, the storage of PCs and plane parameters are not required.

Another important property of $\lambda_{min}(Q)$ is that it exactly accounts for the summation of squared errors of each point fitting to the plane $\lambda_{min}(Q) = \sum e_n^2$, where $e_n = \pi^\top \tilde{p}_n$. Moreover, it is worth mentioning that this error follows a Chi-squared distribution.

C. Gradient over a time-sequence

After defining the EF, it is desirable to calculate its gradient to optimize the trajectory. By definition, the eigen-decomposition is expressed as $Q\pi = \lambda\pi$. The vector of parameters π is a unit vector s.t. $||\pi||^2 = \pi^\top \pi = 1$.

Assuming small perturbations of the form $Q = Q_o + \mathrm{d}Q$, $\lambda = \lambda_o + \mathrm{d}\lambda$ and $\pi = \pi_o + \mathrm{d}\pi$, then the eigenvector definition is derived as

$$Q\,\mathrm{d}\pi + \mathrm{d}Q\,\pi = \lambda\,\mathrm{d}\pi + \mathrm{d}\lambda\,\pi \tag{17}$$

$$\text{s.t.} \quad \pi^\top \mathrm{d}\pi = 0. \tag{18}$$

Taking into account that the matrix Q is symmetric by construction, then

$$Q = Q^\top \quad \Longrightarrow \quad \pi^\top Q = \lambda \pi^\top. \tag{19}$$

One can pre-multiply the expression (17) by $\pi^\top$, substitute (19) and do some manipulations:

$$\begin{aligned}
\pi^\top Q\,\mathrm{d}\pi + \pi^\top \mathrm{d}Q\,\pi &= \pi^\top \lambda\,\mathrm{d}\pi + \pi^\top \mathrm{d}\lambda\,\pi \\
\lambda \underbrace{\pi^\top \mathrm{d}\pi}_{(18)} + \pi^\top \mathrm{d}Q\,\pi &= \lambda \underbrace{\pi^\top \mathrm{d}\pi}_{(18)} + \mathrm{d}\lambda \underbrace{\pi^\top \pi}_{1} \\
\pi^\top \mathrm{d}Q\,\pi &= \mathrm{d}\lambda.
\end{aligned} \tag{20}$$

This compact result expresses the small perturbations on our cost function by $\lambda = \lambda_o + \pi^\top \mathrm{d}Q\,\pi$. We can differentiate this small perturbation w.r.t the manifold $\xi_t \in \mathbb{R}^6$

$$\frac{\partial \lambda}{\partial \xi_t} = \frac{\partial}{\partial \xi_t}(\lambda_o + \pi^\top \mathrm{d}Q\,\pi) = \pi^\top \frac{\partial Q}{\partial \xi_t}\,\pi. \tag{21}$$

The expression $\frac{\partial Q}{\partial \xi_t}$ is the partial derivative of a 4-by-4 matrix which results in a tensor $4\times4\times6$. Using Lie algebra greatly simplifies the calculation of the derivatives, which are obtained on closed form.

The matrices Q_t are each of the components of Q, as defined in (14). In addition, it is symmetric by construction, and hence the column vectors spanning Q_t can be written

$$Q_t = [q_1, q_2, q_3, q_4] = \begin{bmatrix} q_1^\top \\ q_2^\top \\ q_3^\top \\ q_4^\top \end{bmatrix}_{4\times4}. \tag{22}$$

From (14) and being $\Delta\xi_t$ the infinitesimal update to the current transformation ${}^1T_t = \mathrm{Exp}(\xi_t)$

$$\begin{aligned}
\frac{\partial Q}{\partial \Delta\xi_t} &= \frac{\partial}{\partial \Delta\xi_t}\big(K + \mathrm{Exp}(\Delta\xi_t) {}^1T_t S_t {}^1T_t^\top \mathrm{Exp}(\Delta\xi_t)^\top\big) \\
&= \frac{\partial}{\partial \Delta\xi_t}\big(K + \mathrm{Exp}(\Delta\xi_t) Q_t \mathrm{Exp}(\Delta\xi_t)^\top\big) \\
&= \underbrace{\frac{\partial \mathrm{Exp}(\Delta\xi_t)}{\partial \Delta\xi_t} Q_t}_{A_t^L} + \underbrace{Q_t \frac{\partial \mathrm{Exp}(\Delta\xi_t)^\top}{\partial \Delta\xi_t}}_{A_t^R},
\end{aligned} \tag{23}$$

where K is a constant matrix independent of $\Delta\xi_t$ and we follow the left-hand side convention to expand and update transformations (8).

Now, we need to derivate the matrix of generators (7) for each of the components and multiply it to (23). For the sake

of simplicity we omit temporal indexes, which are referring to the matrix Q_t according to (22)

$$A_t^L = \left[\begin{array}{c|c|c|c|c|c} 0 & q_3^\top & -q_2^\top & q_4^\top & 0 & 0 \\ -q_3^\top & 0 & q_1^\top & 0 & q_4^\top & 0 \\ q_2^\top & -q_1^\top & 0 & 0 & 0 & q_4^\top \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]_{4\times4\times6} \tag{24}$$

where each of the "columns" $A_t^L(i)$ is a matrix 4×4 and the index $i = 1, \ldots, 6$ stands for each of the variables of a pose in the manifold.

Exploiting again the symmetry on the matrix Q_t, we rewrite the expression

$$A_t^R(i) = Q_t \frac{\partial \text{Exp}(\Delta\xi_t)^\top}{\partial \Delta\xi_t(i)} = \left(\frac{\partial \text{Exp}(\Delta\xi_t)}{\partial \Delta\xi_t(i)} Q_t^\top\right)^\top = A_t^{L\top}(i)$$

$$\frac{\partial Q}{\partial \Delta\xi_t(i)} = A_t^L(i) + A_t^{L\top}(i). \tag{25}$$

Finally, the overall gradient is defined as

$$\frac{\partial \lambda}{\partial \Delta\xi} = \left[\left(\pi^\top \frac{\partial Q}{\partial \Delta\xi_1}\pi\right)^\top, \ldots, \left(\pi^\top \frac{\partial Q}{\partial \Delta\xi_H}\pi\right)^\top\right]^\top, \tag{26}$$

which is a column vector composed of each of the gradients with respect to $\Delta\xi_t$.

D. Gradient-based optimization

Once a gradient $\frac{\partial\lambda}{\partial\Delta\xi}$ is obtained over the sequence of poses $\xi = [\xi_1, \xi_2, \ldots, \xi_H]$, we choose an optimization method in order to calculate the optimal trajectory, according to the minimization for plane estimation. For this task, we choose a simplification over the Nesterov Accelerated Gradient (NAG) [29] proposed by Bengio et al. [30]. Although the original NAG is meant for convex function and its main contribution is on achieving super linear convergence, for this problem we could not assume such strong conditions as convexity and Lipschisz continuity.

Bengio and collaborators describe the NAG method as a momentum gradient-based optimization. We consider the particular update for poses in the manifold

$$v_k = \beta_{k-1} v_{k-1} - \alpha_{k-1} \frac{\partial \lambda}{\partial \Delta\xi_{k-1}}, \tag{27}$$

$$\xi_k = \left(\beta_k \beta_{k-1} v_{k-1} - (1+\beta_k)\alpha_{k-1} \frac{\partial \lambda}{\partial \Delta\xi_{k-1}}\right) \oplus \xi_{k-1}, \tag{28}$$

where k expresses the iteration index on the optimization sequence. On an abuse of notation, the operator $\oplus$ updates all poses ξ, each of them as in (8). The velocity term v_k, is required to be calculated before (28).

E. Time Continuous trajectory as Interpolation on $SE(3)$

We define the continuous time trajectory as an interpolation directly in the manifold. In general, interpolation of RBT $T(\tau) : [0, 1] \to SE(3)$ of any pair of poses $T_o, T_f \in SE(3)$ is expressed as

$$T(\tau) = \text{Exp}\big(\tau \text{Ln}(T_f T_o^{-1})\big) T_o. \tag{29}$$

If we set the initial transformation as the identity $T_o = I$, then (29) becomes

$$T(\tau) = \text{Exp}\big(\tau \text{Ln}(T_f)\big) \tag{30}$$

which always holds.

We derive the corresponding gradient to the interpolation process with respect to the only pose to optimize, $\xi_f = \text{Ln}(T_f)$. Using the previous gradients from (26) and the chain rule to express the gradient between the pose at time t and the final pose ξ_f, then

$$\frac{\partial \lambda}{\partial \Delta\xi_f} = \sum_{t=1}^{H} \pi^\top \frac{\partial Q}{\partial \Delta\xi_t} \frac{\partial \Delta\xi_t}{\partial \Delta\xi_f} \pi = \sum_{t=1}^{H} \tau(t) \pi^\top \frac{\partial Q}{\partial \Delta\xi_t} \pi, \tag{31}$$

where $\tau(t)$ is the sequence of values from $\tau(1) = 0$ to $\tau(H) = 1$.

V. Evaluations

The evaluation environment generated a random trajectory and simulated the synthetic data corresponding to various planes. The points on planes are sampled at the XY plane and transformed according to a random transformation and the trajectory is generated by sampling in the manifold. The sampled values corresponding to poses, where each pose is $\xi = [w^\top, v^\top]^\top$, are obtained from a uniform distribution $w \sim U(-\pi, \pi)$ and $v \sim U(-4, 4)$. Fig. 2 shows an example of one of such trajectories, most of them describing sharp turns.

Although we showed a derivation for sequences of poses, the results showed that they simply over-fit to the observations giving raise to discontinuities on the trajectory. This result is expected, since we are not giving extra constraints in the form of other factors or any other regularizer. Still, EFs are able to calculate very smooth planes at the cost of displacing trajectories. For this reason, we have only evaluated the continuous-time interpolation version of EFs.

The second consideration is initialization. EFs are sensitive to initialization, so we provide a coarse initial alignment which is calculated between pairs of poses, mainly from the origin to subsequent poses. The result is depicted in Fig. 2-*Center*, however it is still a noisy initialization. Distance between transformations or poses is defined as the Euclidean distance of components in the manifold $d(T_i, T_j) = ||\text{Ln}(T_i T_j^{-1})||_2$.

First, we have selected the hyper-parameters for the optimization as $\alpha = \frac{0.2}{N_{all} H}$ and $\beta = 0.7$. The number of points accounted is N_{all}. The cost function considers all observed points, and therefore it is natural to normalize the gradient with respect to N_{all}. This was achieved by running some experiments and sweeping over the parameters. For $\beta = 0$ NAG behaves as a gradient decent without momentum. The number of planes ranges from 3 to 8, the number of points per observation $[400, 6400]$ and number of poses $[2, 40]$. Each point on the plane is sampled according to $d_e \sim \mathcal{N}(0, 0.01^2)$.

Once obtained the best possible set of parameters, we compared NAG with Gradient Descent (GD). In total, 20k simulations were taken. Fig. 3-*Top* shows the number of

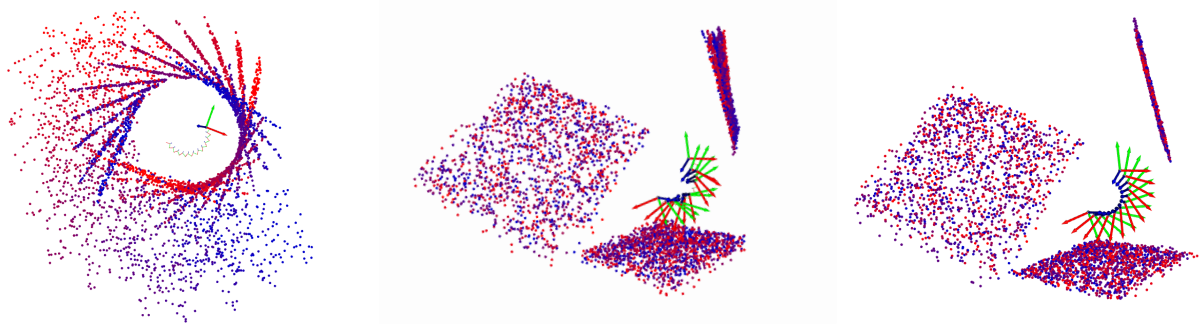

Fig. 2. Eigen-Factors qualitative results. *Left*: No alignment, all point clouds are rendered on their respective origin frames. Still it can be observed some persistent geometry as the sensor rotates. *Center*: Initialization of the trajectory. An orthogonal view to one of the planes shows the plane fitting error. *Right*: Result of EFs with interpolation. The trajectory is clearly described and the plane is better aligned, close to the sampling error.

iterations, w.r.t. number of poses and on the *Bottom* the full trajectory RMSE using the previous distance. The interesting result is that as we increase the number of observations, EFs keep improving.

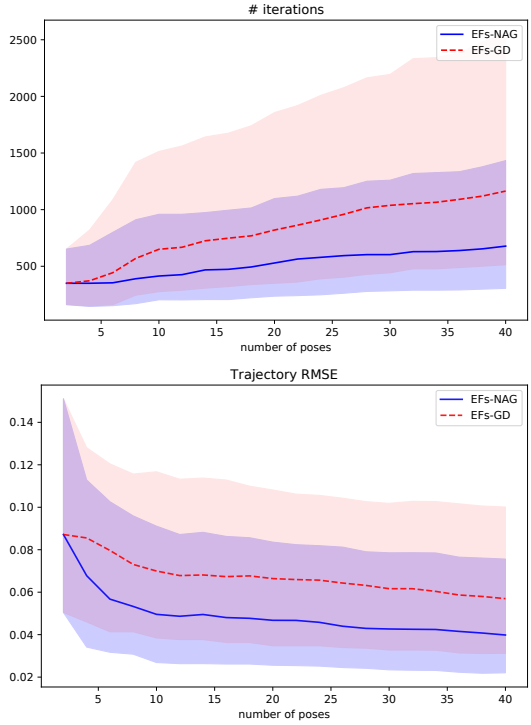

Fig. 3. EFs optimization. NAG gradient in blue is compared with Gradient Descent in lined red. Median results are drawn and the colored area corresponds to the 0.25 and 0.75 intervals.

On the next stage of evaluations we compare EFs with NAG versus a baseline of ICP-point-to-point and ICP-point-to-plane on a similar setting as described above. Both ICP algorithms are implemented on the Open3d library [31] and appear to be highly efficient. Both ICPs are aligning the initial PC and the final PC, discarding all other observed information. We have used the same initialization for the 3 methods. Fig. 4 depicts the main result that supports EFs. For very short trajectories (2-4 poses) the ICP-plane performs better than EFs. For larger trajectories, and thus more PCs evaluated, the error decreases for EFs.

The time of execution for EFs is shown in Fig. 5, and it supports the claim that EFs are more efficient than ICPs or any other method based on computing point error. ICPs use multiple cores while EFs is a single-threaded process, but still EFs outperform them.

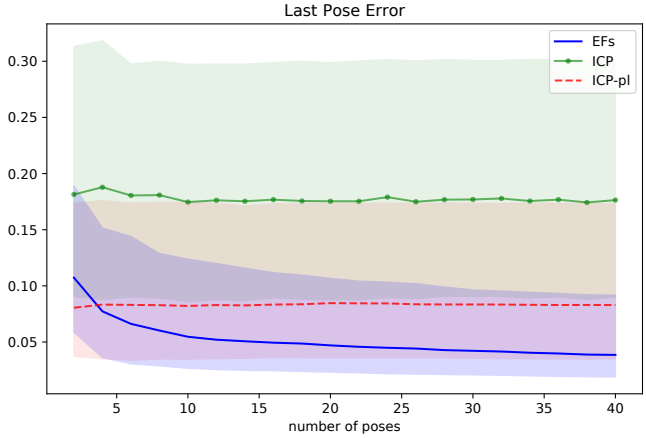

Fig. 4. Error of the last pose. Eigen-Factors improve its accuracy with more poses on the trajectory. ICP-plane (red) achieves high accuracy, but it is only considering a pair of poses, with its implicit error and is outperformed by EFs. ICP-point (green doted) performs worse given the initial conditions.

For a more precise description, please check the supplemental material showing short videos of our method[1]. This environment is coded on C++ and for visualizations and PC routines we used the Open3d library. Code is published at[2].

VI. Conclusions

We have presented Eigen-Factors, a new method for point cloud alignment over multiple frames and time-continuous trajectories. EFs optimize trajectories by estimating the fit of planar surfaces and calculates at each iteration the miss-alignment between different poses. Our formulation accounts for all points, while it does not require to keep them or even recalculate point errors, since these are kept on 4×4 S_t

[1] https://youtu.be/_1u_c43DFUE
[2] https://gitlab.com/gferrer/eigen-factors-iros2019

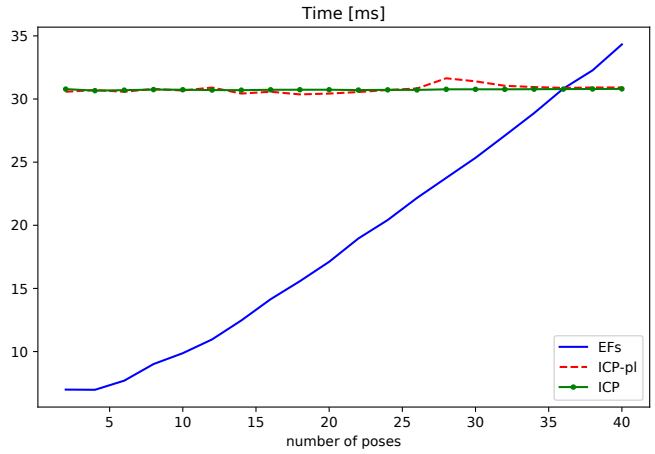

Fig. 5. Median execution time. EFs scale linearly with the number of poses, while ICPs stay constant. The size of the PC ranges from 400 to 6400.

matrices. We have also derived a closed-form of the gradient for the eigenvalue w.r.t the manifold, as a very effective way to manipulate rigid body transformations.

To the best of our knowledge, this is the first attempt to optimize trajectories by minimizing over eigenvalues on planes. We have showed on a simulated environment that EFs reduce plane and trajectory errors, on a continuous interpolated trajectory, although for multiple frames it overfits. More constraints are needed to address this issue.

In this work, we have used EFs on fixed time-window optimization for point cloud alignment, but indeed there is a direct link to other SLAM applications, which we intend to explore in the near future.

REFERENCES

[1] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[2] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[3] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Transactions on pattern analysis and machine intelligence*, no. 5, pp. 698–700, 1987.

[4] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[5] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *IEEE International Conference on Robotics and Automation*. IEEE, 1991, pp. 2724–2729.

[6] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International journal of computer vision*, vol. 13, no. 2, pp. 119–152, 1994.

[7] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. IEEE, 2001, pp. 145–152.

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

[9] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: science and systems*, vol. 2, no. 4, 2009.

[10] J. Serafin and G. Grisetti, "NICP: Dense normal based point cloud registration," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 742–749.

[11] 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*. IEEE, 2013, pp. 5182–5189.

[12] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4605–4611.

[13] E. B. Olson, "Real-time correlative scan matching," in *IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 4387–4393.

[14] D. Hähnel and W. Burgard, "Probabilistic matching for 3d scan registration," in *Proc. of the VDI-Conference Robotik*, 2002.

[15] L. Armesto, J. Minguez, and L. Montesano, "A generalization of the metric-based iterative closest point technique for 3D scan matching," in *International Conference on Robotics and Automation*. IEEE, 2010, pp. 1367–1372.

[16] J. Serafin, E. Olson, and G. Grisetti, "Fast and robust 3D feature extraction from sparse point clouds," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4105–4112.

[17] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3384–3391.

[18] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

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

[20] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct rgb-d slam," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1285–1291.

[21] S. Yang, Y. Song, M. Kaess, and S. Scherer, "Pop-up SLAM: Semantic monocular plane slam for low-texture environments," *arXiv preprint arXiv:1703.07334*, 2017.

[22] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "LIPS: Lidar-inertial 3d plane slam," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 123–130.

[23] M. Hsiao, E. Westman, and M. Kaess, "Dense planar-inertial slam with structural constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6521–6528.

[24] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3206–3211.

[25] B. Hall, *Lie groups, Lie algebras, and representations: an elementary introduction*. Springer, 2015, vol. 222.

[26] E. Eade, "Lie groups for 2d and 3d transformations," *URL http://ethaneade.com/lie.pdf, revised Dec*, 2013.

[27] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[28] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[29] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2)," in *Doklady AN USSR*, vol. 269, 1983, pp. 543–547.

[30] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8624–8628.

[31] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.