Curve-graph odometry: Orientation-free error parameterisations for loop closure problems *

Daniel Gutiérrez-Gómez† and J.J. Guerrero†

October 7, 2015

Abstract – **During incremental odometry estimation in robotics and vision applications, the accumulation of estimation error produces a drift in the trajectory. This drift becomes observable when returning to previously visited areas, where it is possible to correct it by applying loop closing techniques. Ultimately a loop closing process leads to an optimisation problem where new constraints between poses obtained from loop detection are applied to the initial incremental estimate of the trajectory. Typically this optimisation is jointly applied on the position and orientation of each pose of the robot using the state-of-the-art pose graph optimisation scheme on the manifold of the rigid body motions. In this paper we propose to address the loop closure problem using only the positions and thus removing the orientations from the optimisation vector. The novelty in our approach is that, instead of treating trajectory as a set of poses, we look at it as a curve in its pure mathematical meaning. We define an observation function which computes the estimate of one constraint in a local reference frame using only the robot positions. Our proposed method is compared against state-of-the-art pose graph optimisation algorithms in 2 and 3 dimensions. The benefit of eliminating orientations is twofold. First, the objective function in the optimization does not mix translation and rotation terms, which may have different scales. Second, computational performance can be improved due to the reduction in the state dimension of the nodes of the graph.**

1 Introduction

The probabilistic nature of Simultaneous Localisation and Mapping (SLAM) techniques, and the incremental estimation, lead to an unavoidable error build-up. The accumulated error gives raise to a drift in the trajectory, which becomes evident when the sensor platform revisits a previous location. It is expressly in these situations when the so called loop closing techniques can be applied to correct the drift.

The loop closure process can be divided into three steps: loop detection, computation of the loop closing constraint and trajectory correction with the new constraints. The detection and the constraint computation techniques are sensor dependent and are managed by the front-end.

The last step, trajectory correction, is the one which this work is mainly focused on. Traditionally, the new loop constraints are enforced by defining a non-linear least squares optimisation problem where the final trajectory is the one which minimises the combined cost of violating the initial odometry constraints from the SLAM estimate and the new loop closure constraints.

Following the state-of-the-art pose graph formulation [15], the loop closure optimisation problem is presented as a graph of nodes where each node represents one pose and the arcs represent the odometry and loop closure constraints. An odometric constraint encapsulates the incremental motion estimate between two consecutive poses $i-1$ and i in a reference frame attached to $i-1$, in such a way that an arbitrary spatial transformation applied to both poses should not modify the cost of violating this constraint. This applies similarly to a loop closure constraint between two poses, say, i and j, on the relative motion of pose j with respect to a reference frame attached to i or vice versa.

In a pose graph formulation, each pose includes a translation and a rotation and is represented as an element of the special Euclidean group, a manifold which describes the rigid body kinematics in 2 ($\mathbb{SE}(2)$) or 3 ($\mathbb{SE}(3)$) dimensions. In the context of optimisation, variables lying on manifolds different from the usual Euclidean space $\mathbb{R}^n$ are prone to violate the manifold constraints if no special care is taken. One typical approximation, thus subject to inaccuracies, is to impose these constraints after optimisation. However, for greater correctness and accuracy, smarter solutions impose the manifold topology directly during optimisation [2].

Another problematic specific to Euclidean Groups and pose-graph optimisation in SLAM is the inability to define an unambiguous metric [22], which arises from the

*This work was supported by spanish project DPI2012-31781, FEDER and FPU scholarship AP-2012-5507.

†Daniel Gutiérrez-Gómez and Josechu Guerrero are with the Departamento de Informática e Ingeniería de Sistemas (DIIS) y el Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain. {danielgg,josechu.guerrero}@unizar.es

different magnitudes in which rotation and translation are measured. Since error functions in pose-graph SLAM combine rotation and translation, the set up of the scaling between rotation and translation can have a strong influence in the final result of the optimisation. The information matrix for a given constraint solves this ambiguity by normalising both magnitudes. However it is not rare the case where the exact information matrix for some constraints of the graph is not available or easy to compute.

We propose to reformulate the loop closure optimisation problem using a state representation which removes the orientation of the poses. The novelty in our approach is that, instead of treating trajectory as a set of poses, we look at it as a curve in its pure mathematical meaning. A curve is a mathematical entity defined in an Euclidean Space $\mathbb{R}^n$ and has a set of local properties like speed, curvature and torsion, which can be defined point wise and are invariant to arbitrary rigid transformations. This leads to the idea that proper odometry and loop closure constraints can be computed using only the positions and that these constraints are related with the local properties of a discrete curve.

Resulting from our proposal three main advantages arise:

- We avoid mixing translation and rotation magnitudes in the same optimised vector, avoiding heuristic scalings in the norm of the residuals when the information matrix is not available.
- The number of degrees of freedom per pose is reduced from 6 to 3 in the 3D case and from 3 to 2 in the 2D case. This leads to a dimensionality reduction of the optimisation problem which can involve a potential increase in computational performance.
- The optimisation could be performed directly in a Euclidean space, with no need for defining an error function and special operators for non-Euclidean spaces.

Part of this work was presented as a conference paper in [11]. The novelties in this paper with respect our previous work are first the tackling of potential singularities in the cost functions of our method by applying a prior graph reduction to eliminate redundant and aligned poses, secondly the evaluation of our method in an extended set of datasets, specially in 3 dimensions, and finally a simple method to project almost planar 3D pose-graphs to 2D.

The paper is divided as follows. Sec. 2 is dedicated to comment the related work. In Sec.3 we describe the standard pose-graph optimisation problem. In 4 we explain our proposal for orientation-free graph optimisation, first in 2, and then in 3 dimensions. Finally in Sec. 5 we present the experiments where we evaluate our new parametrisation with state-of-the-art pose-graph approaches and in Sec. 6 we extract the conclusions.

2 Related Work

Several related works address efficiency and convergence issues of the optimisation algorithms for pose graphs. However the discussion on different representations of the nodes of the graph is less prevalent and, to the best of our knowledge, all of them assume that the optimisation must be performed both in the orientation and the position of the poses.

Concerning the optimisation algorithms the main objective is to make it robust to local minima and lowering the computational needs. Standard approaches to solve non-linear least squares problems are based on the Gauss-Newton method, which consists in iteratively linearising the energy function around the current solution and solving a linear system until convergence. However this involves a large computational cost and, unless a good initial estimate is provided, it is likely to stuck on a local minima. Approaches based on this method like iSAM [14] and g^{2}o [15] tend to exploit the structure of the graph to reduce the computational cost.

Another family of approaches introduce a relaxation of the problem, i.e., at each iteration they compute an update of only a subset of the nodes. Although these updates are approximate, the robustness to local minima sticking is generally increased. In this sense Duckett et al. proposed the use of Gauss-Seidel relaxation [7] and based on this, Frese et al. [8] introduced a multilevel relaxation (MLR). Olson et al. [21] propose a relaxation based approach using a Stochastic Gradient Descent algorithm (SGD) and an incremental pose parametrisation. Their results showed a dramatic reduction in computational cost compared with other existing approaches at that time. In [10], Grisetti et al. extend Olson's method including a novel tree parametrisation for the poses and extending to 3D poses. Grimes et al. [9] propose to apply a stochastic relaxation while solving the linearised system around current estimate. In [23], Peasley and Birchfield proposed first performing a SGD with relative pose parametrisation to get a quick initial solution close to the minimum and the switching to the Gauss-Seidel method to reach the minimum.

Martinez et al. [19] and Carlone et al. [3] proposed a linear approximation to compute a first suboptimal solution in 2D pose graphs without requiring an initial seed for the state of the nodes. This suboptimal solution could then be refined by non-linear optimisation approaches. Also for 2D graphs, Carlone and Censi [4] proposed recently a method for orientation estimation which is more robust to local minima than state-of-the-art approaches. In [6], Dubbelman et al. propose an efficient method which obtains a closed form solution for loop closure in trajectories where there is a single loop constraint. This work was further extended [5] enabling it to close multiple loops on the same trajectory.

Recently Anderson et al. [1] proposed a continuous-

time SLAM approach, which addresses the pose-graph optimisation by parametrising the motion of the platform by continuous wavelet functions based on B-splines instead of conventional parametrisation with discrete poses. This is beneficial when using high-rate sensors, multiple unsynchronized sensors, or scanning sensors, such as lidar and rolling-shutter cameras, during motion.

Concerning different descriptions of the states of the nodes, the proposal by Strasdat et al. [26] is specially convenient for pose graph optimisation from loop closure of visual odometry estimates obtained by some monocular front-end. By describing the poses and the constraints between them by similarity transforms instead of by rigid body motions, it allows scale drift correction in the optimisation back-end.

Another important issue in the scope of pose-graph optimisation which have been addressed in the literature is the robustness to false positives during the loop detection step. In this sense authors of [15] propose several robust cost functions in their implementation to decrease the effect of outliers. More sophisticated methods for robust pose-graph optimisation include the introduction of switchable constraints [27] which act as weights for the loop closure constraints, the RRR algorithm [17] based on consistency checks of topologically similar loop constraints, or the Max-Mixture Model [20] which accounts for the possibility of false loop closures using a high variance Gaussian distribution.

3 Standard Pose Graph Optimisation

Let $\mathbf{x} = (x_1 \cdots x_N)$ be the optimisation state vector which contains the configuration x_i of each node in the graph. Let $\hat{\mathbf{z}} = (\hat{z}_1 \cdots \hat{z}_M)$ be the vector containing all the constraints between the nodes in the graph. These constraints can not only represent edges joining pairs of nodes but also, more generally, cliques relating an undetermined number of nodes. Let also $f_k(\mathbf{x})$ be an observation function which computes the estimate of the constraint $\hat{z}_k$ given the current state $\mathbf{x}$ of the graph.

3.1 Node Parametrisation and Constraints in $\mathbb{SE}(n)$

When defining a pose-graph optimisation problem in the context of SLAM, nodes in the graph represent poses. One pose consists of a location in the space and an orientation and both can be jointly described in the manifold of rigid body motions in 2D ($\mathbb{SE}(2)$) or 3D ($\mathbb{SE}(3)$) by using transformation matrices. Then the configuration of a node x_i in the graph is parametrised as $x_i = {}_W\mathbf{T}^i$, i.e., the spatial transformation from the world frame W to the frame associated to pose i.

The observation function for a constraint between two poses is defined as:

$$f_{ij}(\mathbf{x}) \doteq f_k(\mathbf{x}) = \left({}_W\mathbf{T}^i\right)^{-1} {}_W\mathbf{T}^j. \tag{1}$$

The odometry constraints for the optimisation problem are computed by applying the observation function to the initial state $\hat{\mathbf{x}} = \{{}_W\hat{\mathbf{T}}^1,\ {}_W\hat{\mathbf{T}}^2,\ ...,\ {}_W\hat{\mathbf{T}}^N\}$, while the loop closure constraints are provided by a module in charge of both loop detection and computation of transforms $\boldsymbol{T_{LC}} = \{{}_{i_1}\hat{\mathbf{T}}^{j_1}_{LC},\ ...,\ {}_{i_L}\hat{\mathbf{T}}^{j_L}_{LC}\}$. That is:

$$\hat{z}_k = {}_i\hat{\mathbf{T}}^j = \begin{cases} \left({}_W\hat{\mathbf{T}}^i\right)^{-1} {}_W\hat{\mathbf{T}}^j & \text{if} \quad (i,j) \in \mathcal{S} \\ {}_i\hat{\mathbf{T}}^j_{LC} & \text{if} \quad (i,j) \in \mathcal{R} \end{cases}, \tag{2}$$

where $\mathcal{S} = \{(1,2), ..., (N-1, N)\}$ and $\mathcal{R} = \{(i_1, j_1), ..., (i_L, j_L)\}$ are respectively the sets of pose pairs sharing odometry and loop closure constraints.

3.2 Generalised Optimisation on Manifolds

The unambiguous description of elements lying on a manifold usually require more parameters than dimensions has the manifold. Such is the case of elements in $\mathbb{SE}(n)$, usually described by transformation matrices. For each additional parameter a constraint related with the manifold topology is established. To comply with these constraints during iterative optimisation, updates of the estimation must be computed in the tangent space of the manifold using a minimal parametrisation. For this purpose the operators $\boxminus$ and $\boxplus$ are used. Roughly speaking, the operator $\boxminus$ computes the difference between two transformations with a minimal parametrisation, while the operator $\boxplus$ applies a minimally parametrised perturbation to a rigid transformation (for a more detailed and rigorous explanation we refer the reader to [13]).

Then, the error e_k resulting from violating the graph constraint between poses i and j is:

$$e_k(\mathbf{x}) = f_k(\mathbf{x}) \boxminus \hat{z}_k. \tag{3}$$

The uncertainty for a constraint $\hat{z}_k$ is represented by the information matrix Ω_k . Assuming that all the constraints are independent, the cost function for pose graph optimisation is:

$$\begin{aligned} \mathbf{x}^* &= \arg\min_{\mathbf{x}} \sum_{(i,j)\in\mathcal{S}} e_k^\mathsf{T}(\mathbf{x}) \Omega_k e_k(\mathbf{x}) + \sum_{(i,j)\in\mathcal{R}} e_k^\mathsf{T}(\mathbf{x}) \Omega_k e_k(\mathbf{x}) \\ &= \arg\min_{\mathbf{x}} \mathbf{e}^\mathsf{T}(\mathbf{x}) \boldsymbol{\Omega} \mathbf{e}(\mathbf{x}), \end{aligned} \tag{4}$$

where $\mathbf{e} = (e_1 \cdots e_M)$ and $\mathbf{\Omega} = \text{diag}\,(\Omega_1, ..., \Omega_M)$ and $M = (N-1) + L$ is the total number of constraints.

Given the initial guess $\breve{\mathbf{x}} = \hat{\mathbf{x}}$, (4) can be solved iteratively until convergence by computing a first order Taylor expansion of the error at each iteration :

$$\mathbf{e}(\breve{\mathbf{x}} \boxplus \boldsymbol{\delta}) = \mathbf{e}(\breve{\mathbf{x}}) + \left.\frac{\partial\,(\mathbf{e}\,(\breve{\mathbf{x}} \boxplus \boldsymbol{\delta}))}{\partial \boldsymbol{\delta}}\right|_{\boldsymbol{\delta}=\mathbf{0}} \boldsymbol{\delta} = \breve{\mathbf{e}} + \mathbf{J}\boldsymbol{\delta}, \quad (5)$$

where $\boldsymbol{\delta} = (\delta_1 \cdots \delta_N)$ and abusing from notation $\breve{\mathbf{x}} \boxplus \boldsymbol{\delta} = \{\breve{x}_1 \boxplus \delta_1, ... \,, \breve{x}_N \boxplus \delta_N\}$. The Jacobian can be computed analytically (Chap. 2 in [25]).

Then, the resulting linear optimisation problem is solved to obtain the state incremental update $\boldsymbol{\delta}$ which is used to compute the state for next iteration:

$$\breve{\mathbf{x}} \leftarrow \breve{\mathbf{x}} \boxplus \boldsymbol{\delta}. \quad (6)$$

4 Our Approach: Graph Optimisation on $\mathbb{R}^n$

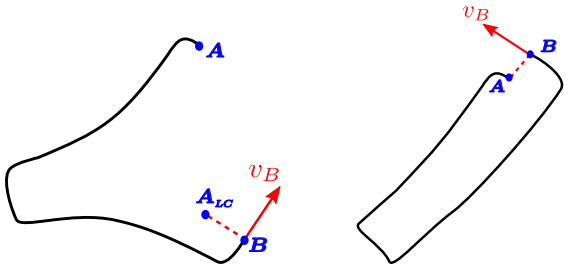
Instead of jointly estimating the position and orientation of the poses by carrying on an optimisation in the manifold of rigid body motions, we propose imposing the loop closure constraints by taking only the position part of the poses. The underlying idea behind this proposal is that a trajectory can be considered a discrete curve in the Euclidean space where new loop constraints between some points are imposed modifying as less as possible the local properties of the curve, which are encoded in the odometric constraints. This is intuitively shown in the example of Fig. 1. Given a trajectory where no information about the orientation is shown, one can perceive however how the trajectory has to be bended so that the point **A** is at the relative position w.r.t. **B** given by the loop closure constraint (dashed line) and the vector $\mathbf{v}_B$ tangent to the trajectory at **B**.

Moreover, the removal of the orientations out of the optimised variables seems reasonable, since the position and orientation of a body which freely moves in the space do not have to be coupled generally. In this sense, the inclusion of the orientation in the optimisation may respond primarily to the need of expressing the odometry and loop constraints in a local reference frame, such that that the error function is invariant to rigid body motions applied to the whole trajectory.

Thus the main challenge is to define appropriate constraints given only the set of positions composing the trajectory. These constraints must keep the error invariant to an arbitrary rigid motion applied to the curve.

4.1 Curves in 2D and 3D

Let us first introduce some notions about curves in 2 and 3 dimensions [24]. These notions though not applied in the implementation of our approach help to provide a mathematical insight of the implications of our approach, showing in fact that our proposed optimisation problem is equivalent to bending a curve so that it passes through new points, while trying to keep its original shape.



(a) (b)

Figure 1: (a) Curve with an open loop where the point **A** should be at the same position as $\mathbf{A_{LC}}$ and keep a relative orientation w.r.t. the vector tangent to the trajectory at **B**. (b) Intuition of how this loop should be closed.

Generally, a curve $\mathbf{r}$ can be defined as a mapping of a scalar t in a given interval $I = [a, b]$ onto the euclidean space $\mathbb{R}^n$, i.e., $\mathbf{r} : I \rightarrow \mathbb{R}^n$.

A n-dimensional curve is characterised by n properties defined locally at every point of the curve. For a 2D curve these properties are the metric derivative or speed $||\mathbf{r}'||$, and the curvature $\kappa(t)$ which is defined as:

$$\kappa(t) = \frac{\mathbf{r}'^{\mathsf{T}}(t)\mathbb{J}\mathbf{r}''(t)}{||\mathbf{r}'(t)||^3}, \quad (7)$$

with

$$\mathbb{J} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \quad (8)$$

For the 3D case we have to consider a new property of the curve: the torsion $\tau(t)$. The torsion of a curve is given by the variation of its osculating plane which can be defined as the plane which locally contains the curve in the vicinity of one of its points. Note that planar curves have no torsion since they are contained in the same plane at every point.

Then, for a 3D curve the curvature and torsion are defined by:

$$\kappa(t) = \frac{||\mathbf{r}'(t) \times \mathbf{r}''(t)||}{||\mathbf{r}'(t)||^3}, \quad (9)$$

$$\tau(t) = \frac{\mathbf{r}'''^{\mathsf{T}}(t)(\mathbf{r}'(t) \times \mathbf{r}''(t))}{||\mathbf{r}'(t) \times \mathbf{r}''(t)||^2}. \quad (10)$$

Note that these properties are invariant to rigid transformations applied to the curve both in the 2D and 3D cases.

4.2 Node Parametrisation and Constraints

Our approach parametrises each node in the graph as $x_i = \mathbf{r}_W^i = {}_W\mathbf{T}^i\,(1{:}3,4)$. Analogously to Sec. 3 we define an observation function for the constraints:

$$f_{ij}(\mathbf{x}) \doteq f_k(\mathbf{x}) = f(\mathcal{F}_{\mathbb{R}^n}(\mathbf{r}_W^i), \mathbf{r}_W^j), \tag{11}$$

where $\mathcal{F}_{\mathbb{R}^n}(\mathbf{r}_W^i)$ is a function which extracts from the graph, the minimum number of poses backwards from $\mathbf{r}_W^i$ to define a reference frame for a given number of spatial dimensions n. Details on how this reference frame is defined for different dimensions will be provided in next sections.

To apply our method we need the relative position measurements $\boldsymbol{\Delta}\hat{\mathbf{r}}_i^{ij}$ between all the pairs of frames (i,j) sharing one constraint. This is done in two steps. The first step consists in computing the absolute pose measurement of frame j as the result of concatenating the constraint between i and j with the absolute pose of frame i, and then taking the translation part of the pose. In the case of the odometric constraint this step is straightforward, since absolute positions correspond to the initial state of the graph $\hat{\mathbf{x}} = \{\hat{\mathbf{r}}_W^1, ..., \hat{\mathbf{r}}_W^N\}$ directly obtained from the odometry front-end. In the case of a loop closure constraint, the associated absolute pose ${}_W\hat{\mathbf{T}}_{LC}^{j_l}$ for a loop constraint between frames i_l and j_l is computed as:

$${}_W\hat{\mathbf{T}}_{LC}^{j_l} = {}_W\hat{\mathbf{T}}^{i_l}\,{}_{i_l}\hat{\mathbf{T}}_{LC}^{j_l} \tag{12}$$

where transformation ${}_W\hat{\mathbf{T}}^{i_l}$ is taken from the absolute poses given by odometry front-end, and ${}_{i_l}\hat{\mathbf{T}}_{LC}^{j_l}$ is computed by the loop detection front-end. The corresponding absolute position is extracted from the resulting transform, as $\hat{\mathbf{r}}_{W,LC}^{j_l} = {}_W\hat{\mathbf{T}}_{LC}^{j_l}\,(1{:}3,4)$.

Note that this step is the only one where we use the rotation part of the poses and that it is agnostic of whether the platform is oriented in the direction of movement or not. This means that our approach does not need to assume that the orientation of the platform is aligned with the platform speed vector.

The second step is the computation of the constraints for our optimisation problem applying the observation function to the measures of the absolute positions we have computed in the first step:

$$\hat{z}_k = \boldsymbol{\Delta}\hat{\mathbf{r}}_i^{ij} = \begin{cases} f\left(\mathcal{F}_{\mathbb{R}^n}(\hat{\mathbf{r}}_W^i), \hat{\mathbf{r}}_W^j\right) & \text{if} \quad (i,j) \in \mathcal{S} \\ f\left(\mathcal{F}_{\mathbb{R}^n}(\hat{\mathbf{r}}_W^i), \hat{\mathbf{r}}_{W,LC}^j\right) & \text{if} \quad (i,j) \in \mathcal{R} \end{cases}. \tag{13}$$

Then the optimisation is performed analogously to Sec. 3. Since we are optimising in $\mathbb{R}^n$, $\boxplus$ and $\boxminus$ operators are the conventional operators for addition and subtraction.

Since the number of properties of the curve is not the same in 2 and 3 dimensions, the definition of function $f(\cdot)$ and the structure of the optimisation problem slightly changes from one case to another. For sake of clearness we treat the two cases separately starting with the easiest 2D case and then stepping up to the 3D case. For each case we proceed as follows: first we compute an observation function $f(\cdot)$ and then show that it is related to the properties of the curve. In next sections we abuse of notation and merge the definition of both points in the curve, and positions, i.e., $\mathbf{r}_i \doteq \mathbf{r}_W^i$.

4.3 Loop closure in $\mathbb{R}^2$

In the 2D case we need two positions to define a reference frame, so we take $\mathcal{F}_{\mathbb{R}^2}(\mathbf{r}_W^i) = \{\mathbf{r}_W^{i-1},\ \mathbf{r}_W^i\}$ and define:

$$\Delta\mathbf{r}_0 = \mathbf{r}_W^i - \mathbf{r}_W^{i-1}, \tag{14}$$

$$\Delta\mathbf{r}_1 = \mathbf{r}_W^j - \mathbf{r}_W^i. \tag{15}$$

The observation function $f\left(\mathbf{r}_W^{i-1}, \mathbf{r}_W^i, \mathbf{r}_W^j\right)$ establishes ternary constraints and is computed as:

$$f\left(\mathbf{r}_W^{i-1}, \mathbf{r}_W^i, \mathbf{r}_W^j\right) = \begin{pmatrix} \mathbf{e}_\mathbf{x}^T \\ \mathbf{e}_\mathbf{y}^T \end{pmatrix} \Delta\mathbf{r}_1 = \frac{1}{||\Delta\mathbf{r}_0||} \begin{pmatrix} \Delta\mathbf{r}_0{}^\mathsf{T} \\ \Delta\mathbf{r}_0{}^\mathsf{T}\mathbb{J} \end{pmatrix} \Delta\mathbf{r}_1, \tag{16}$$

and expresses the odometry vector $\Delta\mathbf{r}_1$ in a reference frame whose unit vector $\mathbf{e}_\mathbf{x}$ is aligned with $\Delta\mathbf{r}_0$. Note that an arbitrary rotation applied both to $\Delta\mathbf{r}_0$and $\Delta\mathbf{r}_1$does not change the returned value of this function.

Now let us see how the observation function $f\left(\mathbf{r}_W^{i-1}, \mathbf{r}_W^i, \mathbf{r}_W^j\right)$, relates with the local properties of the curve in the case of odometric constraints. In our particular problem, the curve is discretised in a set of points, being the scalar which parametrises the curve the i^{th} position index. So we need to apply finite differences to compute the first and second order derivatives:

$$\mathbf{r}'(i) = \mathbf{r}_i - \mathbf{r}_{i-1} = \Delta\mathbf{r}_0, \tag{17}$$

$$\mathbf{r}''(i) = \mathbf{r}_j - 2\mathbf{r}_i + \mathbf{r}_{i-1} = \Delta\mathbf{r}_1 - \Delta\mathbf{r}_0. \tag{18}$$

Note that for the first order derivative we have taken the backwards difference convention. We hold this convention through the rest of the paper for $(2n+1)^{th}$ order derivatives.

Applying the definitions of the derivative we can compute the curvature κ_i at a curve point $\mathbf{r}_i$:

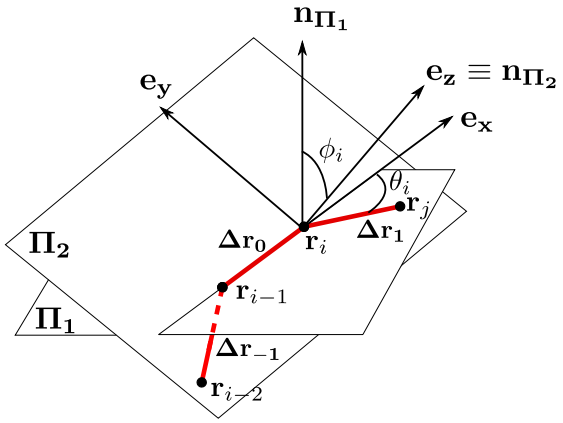


Figure 2: Detail of a discrete 3D curve and the variation of its osculating plane

$$\kappa_i = \frac{\Delta\mathbf{r_0}^{\mathsf{T}} \mathbb{J} \Delta\mathbf{r_1}}{||\Delta\mathbf{r_0}||^3}, \tag{19}$$

and putting it into (16) we get:

$$f\left(\mathbf{r}_W^{i-1}, \mathbf{r}_W^i, \mathbf{r}_W^j\right) = \begin{pmatrix} ||\Delta\mathbf{r_1}|| \sqrt{1 - \frac{||\Delta\mathbf{r_0}||^4}{||\Delta\mathbf{r_1}||^2} \kappa_i^2} \\ ||\Delta\mathbf{r_0}||^2 \kappa_i^2 \end{pmatrix}, \tag{20}$$

verifying that the odometry constraint encapsulates a preservation of the local properties of the curve.

Note that when treating the odometry as a curve, distances between adjacent points must not be zero. Otherwise it is not possible to compute the division by $||\Delta\mathbf{r}_0||$ and the optimisation is likely to fail. Special care must be taken by eliminating redundant points from the initial odometry estimate. It must be remarked that the two poses which a loop is closed at, do not have to obey this restriction since they are never used to compute $\Delta\mathbf{r}_0$. Intuitively speaking, a curve must be always "*in movement*" but it can intersect itself.

4.4 Loop closure in $\mathbb{R}^3$

While in a plane we only need one vector to define the reference frame, in the space we require a plane spanned by two vectors to define it unambiguously. Since we need an additional point to compute the second vector, we take $\mathcal{F}_{\mathbb{R}^3}(\mathbf{r}_W^i) = \{\mathbf{r}_W^{i-2}, \mathbf{r}_W^{i-1}, \mathbf{r}_W^i\}$ and define:

$$\Delta\mathbf{r_{-1}} = \mathbf{r}_W^{i-1} - \mathbf{r}_W^{i-2}. \tag{21}$$

Thus, the observation function $f\left(\mathbf{r}_W^{i-2}, \mathbf{r}_W^{i-1}, \mathbf{r}_W^i, \mathbf{r}_W^j\right)$ establishes quaternary constraints between positions. To define the observation function we proceed analogously to the 2D case and build a local coordinate frame such that $\mathbf{e_x}$ is aligned with $\Delta\mathbf{r_0}$ and $\mathbf{e_z}$ is the normal of the plane defined by $\Delta\mathbf{r_0}$ and $\Delta\mathbf{r_{-1}}$ (see Fig.2). Notating $[\Delta\mathbf{r_0}]_\times$ as the antisymmetric matrix formed with vector $\Delta\mathbf{r}_0$, the following observation function yields:

$$f\left(\mathbf{r}_W^i, \mathbf{r}_W^{i-1}, \mathbf{r}_W^{i-2}, \mathbf{r}_W^{i-3}\right) = \begin{pmatrix} \mathbf{e_x}^T \\ \mathbf{e_y}^T \\ \mathbf{e_z}^T \end{pmatrix} \Delta\mathbf{r_1} = \begin{pmatrix} \frac{\Delta\mathbf{r_0}^{\mathsf{T}}}{||\Delta\mathbf{r_0}||} \\ -\frac{\Delta\mathbf{r}_{-1}^{\mathsf{T}} [\Delta\mathbf{r_0}]_\times^2}{||[\Delta\mathbf{r_0}]_\times^2 \Delta\mathbf{r_{-1}}||} \\ -\frac{\Delta\mathbf{r}_{-1}^{\mathsf{T}} [\Delta\mathbf{r_0}]_\times}{||[\Delta\mathbf{r_0}]_\times \Delta\mathbf{r_{-1}}||} \end{pmatrix} \Delta\mathbf{r_1}. \tag{22}$$

However this observation function involves a division by zero when $\Delta\mathbf{r_{-1}}$ and $\Delta\mathbf{r_0}$ are aligned. Unlike the case where $||\Delta\mathbf{r}_0|| = 0$, this situation is likely to arise in a curve (concretely in straight parts) and and it stems from the fact that it is impossible to define a plane and consequently a reference frame from 3 aligned points. In order to avoid this risky situation the 3 graph must be preprocessed by joining the odometry segments whose relative angle is near zero. As in the 2-dimensional case note, that an arbitrary rotation applied to $\Delta\mathbf{r}_{-1}$, $\Delta\mathbf{r}_0$and $\Delta\mathbf{r}_1$does not change the returned value of the observation function.

As in the 2-dimensional case now we show that in the case of the odometric constraints the crafted observation function encapsulate restrictions on local properties of the curve (length, curvature κ and torsion τ).

Let us first discretise the third derivative,

$$\mathbf{r}'''(i) = \Delta\mathbf{r_{-1}} + \Delta\mathbf{r_1} - 2\Delta\mathbf{r_0}, \tag{23}$$

and take (17) and (18) for the first and second order derivatives to obtain the discretised expressions for curvature and torsion:

$$\kappa_i = \frac{||\Delta\mathbf{r_0} \times \Delta\mathbf{r_1}||}{||\Delta\mathbf{r_0}||^3}, \tag{24}$$

$$\tau_i = \frac{\Delta\mathbf{r}_{-1}^{\mathsf{T}} (\Delta\mathbf{r_0} \times \Delta\mathbf{r_1})}{||\Delta\mathbf{r_0} \times \Delta\mathbf{r_1}||^2}. \tag{25}$$

Recalling (24) and (25) and applying the definitions of the cross and dot product and some trigonometric properties we get

$$f\left(\mathbf{r}_W^{i-2}, \mathbf{r}_W^{i-1}, \mathbf{r}_W^i, \mathbf{r}_W^j\right) = \begin{pmatrix} ||\Delta\mathbf{r_1}|| \sqrt{1 - \kappa_i^2 \frac{||\Delta\mathbf{r_0}||^4}{||\Delta\mathbf{r_1}||^2}} \\ ||\Delta\mathbf{r_0}||^2 \kappa_i \sqrt{1 - \tau_i \frac{\kappa_i^2 ||\Delta\mathbf{r_0}||^8}{\kappa_{i-1}^2 ||\Delta\mathbf{r_{-1}}||^6}} \\ \tau_i \kappa_i^2 \frac{||\Delta\mathbf{r_0}||^5}{||\Delta\mathbf{r_{-1}}||} \end{pmatrix}, \tag{26}$$

proving that, as occurs in the 2D case, the crafted odometry function for the 3D case encapsulates a preservation of the local properties of the curve.

4.5 Graph Reduction

Latif and Neira [18] showed that graph sparsification by joining poses in segments can greatly reduce the computational cost of the initial pose-graph optimisation problem. In our approach graph reduction is proposed to avoid singularities in the observation functions. In the 2D case this is produced by redundant poses in the graph, while in the 3D case singularities are produced not only by redundant poses but also when poses are aligned, i.e., they form a segment. To avoid this, graphs in both 2D and 3D must be potentially reduced such that all distances between consecutive poses are greater than a threshold th_d, and in addition, for the 3D cases a further reduction must be performed such that the angles between consecutive odometric segments are greater than a threshold th_θ.

4.6 Projection of pose-graph from 3D onto 2D space

In some cases, as occurs for example with a visual system on a platform which moves on a planar surface, the odometry and the initial pose-graph are estimated in 3 dimensions though the motion of the platform occurs mostly on a plane. In pose graph optimisation, 2D problems are far more simple than 3D ones, and even some approaches are only able to work in 2-dimensional graphs. For this reason, if it is allowed by the nature of the motion, it can be convenient to convert a 3D graph into a simplified 2D version. In our approach the gain of doing this is two fold. First, we not only reduce the dimension of the problem, but also the number of nodes implied in the constraints. Second, it eliminates the need of dealing with the singularities in 3D graphs which arise from the alignment of points in the trajectory.To generate a 2D graph from a 3D one, we propose the procedure described in Algorithm 1. This essentially consists in aligning the $z-axis$ of the reference frame attached to each pose with the direction of plane which fits better the point cloud formed by all the trajectory point,s and then project the poses and the loop constraints onto the new $2D$ given by the computed plane.

Algorithm 1 Projection of 3D pose-graph onto 2D space

Require: Initial state $\hat{\mathbf{x}} = \{{}_W\hat{\mathbf{T}}^1,\ {}_W\hat{\mathbf{T}}^2,\ ...,\ {}_W\hat{\mathbf{T}}^N\}$ and loop closure constraints $\boldsymbol{T_{LC}} = \{{}_{i_1}\hat{\mathbf{T}}^{j_1}_{LC},\ ...,\ {}_{i_L}\hat{\mathbf{T}}^{j_L}_{LC}\}$ in $\mathbb{SE}(3)$

Ensure: Initial state $\hat{\mathbf{x}}_p = \{{}_W\hat{\mathbf{T}}^1_p,\ {}_W\hat{\mathbf{T}}^2_p,\ ...,\ {}_W\hat{\mathbf{T}}^N_p\}$ and loop closure constraints $\boldsymbol{T_{LC},_p} = \{{}_{i_1}\hat{\mathbf{T}}^{j_1}_{LC,p},\ ...,\ {}_{i_L}\hat{\mathbf{T}}^{j_L}_{LC,p}\}$ in $\mathbb{SE}(2)$

$\boldsymbol{\Delta}\mathbf{r}^i_W := \mathbf{r}^i_W - \text{mean}(\{\hat{\mathbf{r}}^1_W, ..., \hat{\mathbf{r}}^N_W\})$

$[\mathbf{U},\ \mathbf{S},\ \mathbf{V}] := \text{svd}\left(\sum_i \boldsymbol{\Delta}\mathbf{r}^i_W \left(\boldsymbol{\Delta}\mathbf{r}^i_W\right)^T\right)$; $\mathbf{v}_{\lambda_{min}} = \mathbf{V}(:,3)$;

$\mathbf{R}_{pre} := \mathbf{R} \in \mathbb{SO}(3) : \{\mathbf{v}_{pre} = \mathbf{R}\mathbf{v}_{\lambda_{min}}, v_{pre,z} = ||\mathbf{v}_{\lambda_{min}}||_\infty\}$

$\mathbf{R}_{al} := \mathbf{R} \in \mathbb{SO}(3) : \mathbf{e_z} = \mathbf{R}\mathbf{v}_{pre}$

$:= \exp\left(\frac{\text{asin}||\mathbf{v}_{pre}\times\mathbf{e_z}||}{||\mathbf{v}_{pre}\times\mathbf{e_z}||}\mathbf{v}_{pre}\times\mathbf{e_z}\right)$

for every $\mathbf{T}_{3D} \in \{\hat{\mathbf{x}},\ \boldsymbol{T_{LC}}\}$ **do**

  if $\mathbf{T}_{3D} \in \boldsymbol{T_{LC}}$ **then**

    $\mathbf{T}_{3D} \leftarrow {}_W\hat{\mathbf{T}}^{i_l}\mathbf{T}_{3D} \quad l = loop\_idx$

  end if

  $\mathbf{T}_{3D,al} = \begin{pmatrix}\mathbf{R}_{al} & \mathbf{0}\\ \mathbf{0}^T & 1\end{pmatrix}\begin{pmatrix}\mathbf{R}_{pre} & \mathbf{0}\\ \mathbf{0}^T & 1\end{pmatrix}\mathbf{T}_{3D}\begin{pmatrix}\mathbf{R}^T_{pre} & \mathbf{0}\\ \mathbf{0}^T & 1\end{pmatrix}\begin{pmatrix}\mathbf{R}^T_{al} & \mathbf{0}\\ \mathbf{0}^T & 1\end{pmatrix}$;

  $\theta = \text{atan2}\left(\mathbf{R}_{3D,al}(2,1),\ \mathbf{R}_{3D,al}(1,1)\right)$;

  $\mathbf{T}_{2D} = \begin{pmatrix}\cos\theta & -\sin\theta & r_{3Dal,x}\\ \sin\theta & \cos\theta & r_{3Dal,y}\\ 0 & 0 & 1\end{pmatrix}$;

  if $\mathbf{T}_{3D} \in \hat{\mathbf{x}}$ **then**

    $\hat{\mathbf{x}}_p \leftarrow \{\hat{\mathbf{x}}_p,\ \mathbf{T}_{2D}\}$

  else

    $\boldsymbol{T_{LC},_p} \leftarrow \{\boldsymbol{T_{LC},_p},\ \left({}_W\hat{\mathbf{T}}^{i_l}_p\right)^{-1}\mathbf{T}_{2D}\}$

  end if

end for

5 Experiments

In this section we provide experimental validation of our approach using publicly available data-sets and a comparison with state-of-the art pose graph optimisation methods. The implementation and comparison of our method has been performed within the g^{2}o framework [15]. The experiments were performed in an Intel Core i5-2500 at 3.30 GHz.

For the 2D case we have compared three different approaches: our optimisation on $\mathbb{R}^2$ with CSparse solver and the Levenberg-Marquardt algorithm, optimisation on $\mathbb{SE}(2)$ with the CSparse solver and the Levenberg-Marquardt algorithm and a g^{2}o implementation of the linear approximation method for 2D pose graphs of Carlone et al. [3]. The CSparse solver consists of an efficient implementation of a sparse Cholesky factorisation algorithm to solve linear systems and was selected among other available solvers in g^{2}o due to its accuracy and low computation times in the tested data-sets.

The experiments on synthetic 2D datasets (Fig. 3) show that in 2D our approach is comparable in accuracy to the other methods. In terms of convergence speed, as shown in Table 1, optimisation in $\mathbb{SE}(2)$ and our method (optimisation in $\mathbb{R}^2$) yield a similar performance. However, the linear approximation method outperforms both of them since it only takes few iterations to converge in most of the considered cases, but it is restricted to 2D graphs. Also, we can observe that for increasing levels of noise in the Manhattan3500 our proposal converges to qualitative better solutions than optimising on $\mathbb{SE}(2)$.

In real datasets (Fig. 4 and Table 2) our method with standard least squares suffers from low accuracy and

Table 1: Convergence speed comparison of different optimisation approaches in 2D datasets (time in seconds)

Dataset	LM $\mathbb{R}^2$	LM $\mathbb{SE}(2)$	Linear 2D
Manhattan 3500	0.0708 (4 iters)	0.0845 (5 iters)	0.0611 (2 iters)
Manhattan 3500a	0.5144 (20 iters)	0.253 (11 iters)	0.056 (3 iters)
Manhattan 3500b	1.689 (75 iters)	0.6078 (20 iters)	0.049 (3 iters)
Manhattan 3500c	3.1317 (100 iters)	0.566 (30 iters)	0.3656 (30 iters)
Manhattan 10000	1.8314 (10 iters)	1.5707 (10 iters)	0.3648 (2 iters)
City 10000	4.9098 (24 iters)	1.7951 (10 iters)	0.3543 (4 iters)

Table 2: Convergence speed comparison of different optimisation approaches in 2D datasets (time in seconds)

Dataset	LM $\mathbb{R}^2$	LM $\mathbb{R}^2$ + pseudoHuber	LM $\mathbb{SE}(2)$	Linear 2D
MIT Killian Court	0.6364 (282 iters)	0.1158 (50 iters)	0.0453 (21 iters)	0.00675 (3 iters)
Intel	1.1041 (300 iters)	0.6357 (200 iters)	1.455 (363 iters)	0.01099 (3 iters)

Table 3: Convergence speed comparison of different optimisation approaches in the 3D data-sets

Dataset	CSparse LM $\mathbb{R}^3$	CSparse LM $\mathbb{SE}(3)$
sphere2500	3.992 s (0.16 s/iter)	3.80634 s (0.30 s/iter)
torus10000	105.916 s (2.83 s/iter)	17.7078s (1.36 s/iter)
parking-garage	0.2652 s (0.039 s/iter)	0.4625 s (0.088 s/iter)

convergence speed. However, we have noted surprisingly that, though the input graphs do not contain outliers in any of the considered datasets, using a pseudo-Huber robust cost function available in g^{2}o significantly improves both the accuracy and the convergence speed when using our parametrisation. This beneficial effect has been also noted when optimising the MIT Killian Court dataset in $\mathbb{SE}(2)$, which when using a pseudo Huber cost function converges to the same solution as the linear 2D method instead of getting stuck in a local minima.

The 3D case has been tested with two synthetic datasets, sphere2500 and torus10000, from [14] and one real dataset, parking-garage, from [10]. We have used the CSparse solver in the two compared approaches: optimisation in $\mathbb{R}^3$ with Levenberg-Marquardt (ours) and optimisation on $\mathbb{SE}(3)$ with Levenberg-Marquardt. In the tested datasets (Fig. 5) we observe that both methods yield similar accuracy. Concerning the convergence speed, the difference of performance between both methods greatly varies between datasets (Table 3). In the torus10000 dataset $\mathbb{SE}(3)$ shows a better performance requiring less time per iteration, while in the sphere2500 and the parking-garage datasets our method requires less time per iteration.

Due to the elimination of the orientation from the nodes parametrisation, our approach reduces the dimensions of the optimisation problem by a factor of $2/3$ for 2D graphs, and by $1/2$ for 3D graphs. Since the cost of the Cholesky factorisation required to solve the linear system at each iteration has order $\mathcal{O}(n^3)$ in dense problems the computational cost per iteration is around 3.5 and 8 times lower for 2D and 3D respectively. However, in the case of sparse problems, which can be efficiently optimised with appropriate solvers, the performance is not only affected by dimension of the problem, but also by the sparsity and also the distribution of the non-zero elements in the Hessian. In this sense the substitution in our approach of binary constraints by ternary or quaternary constraints affect also the computational performance.

In Fig. 6 we show the sparsity patterns both of the Jacobians and the Hessians for the cases of the torus10000 graph and a sphere graph with an identical number of poses and constraints. In torus10000 it can be observed that loop constraints in the lower part of the jacobian show some aleatority in the distribution of the constraints, while in the case of the sphere graph the constraints show a better arrangement. Note that this last configuration is more realistic since in real world datasets it is a consequence of the spatio temporal consistency, i.e., if there exists a loop closure between poses i and j, it is quite likely that it holds also for poses $i+1$ and $j+1$. Being the Hessian of both graphs equally sparse, it has been verified that the graph of the sphere yielded a time per iteration one order of magnitude lower than for the torus10000 dataset, being the gain in speed of our parametrisation with respect to $\mathbb{SE}(3)$ similar to the obtained in the rest of the tested 3D datasets. This lead us to conclude that the negative effect on computational performance caused by constraints implying several nodes tends to be aggravated if the Hessian is poorly structured.

Since the dimension of the optimisation problems and the used cost functions are different, the quantitative comparison of χ^2 scores across different methods is not possible. Quantitative comparison is performed then only on those synthetic datasets whose Ground Truth is available (Table 4). The column for the $\mathbb{SE}(n)$ in the 2D datasets encapsulates both $LM + \mathbb{SE}(2)$ and $linear2d$ results since they produced the same final graph in the tabulated datasets. We have measured the translational $RMSE$ by taking the error in distance between the position of the corresponding nodes of the ground truth and the evaluated graphs. It can be noted that in the 2D case the accuracy of our proposal is similar and competitive with respect to the state of the

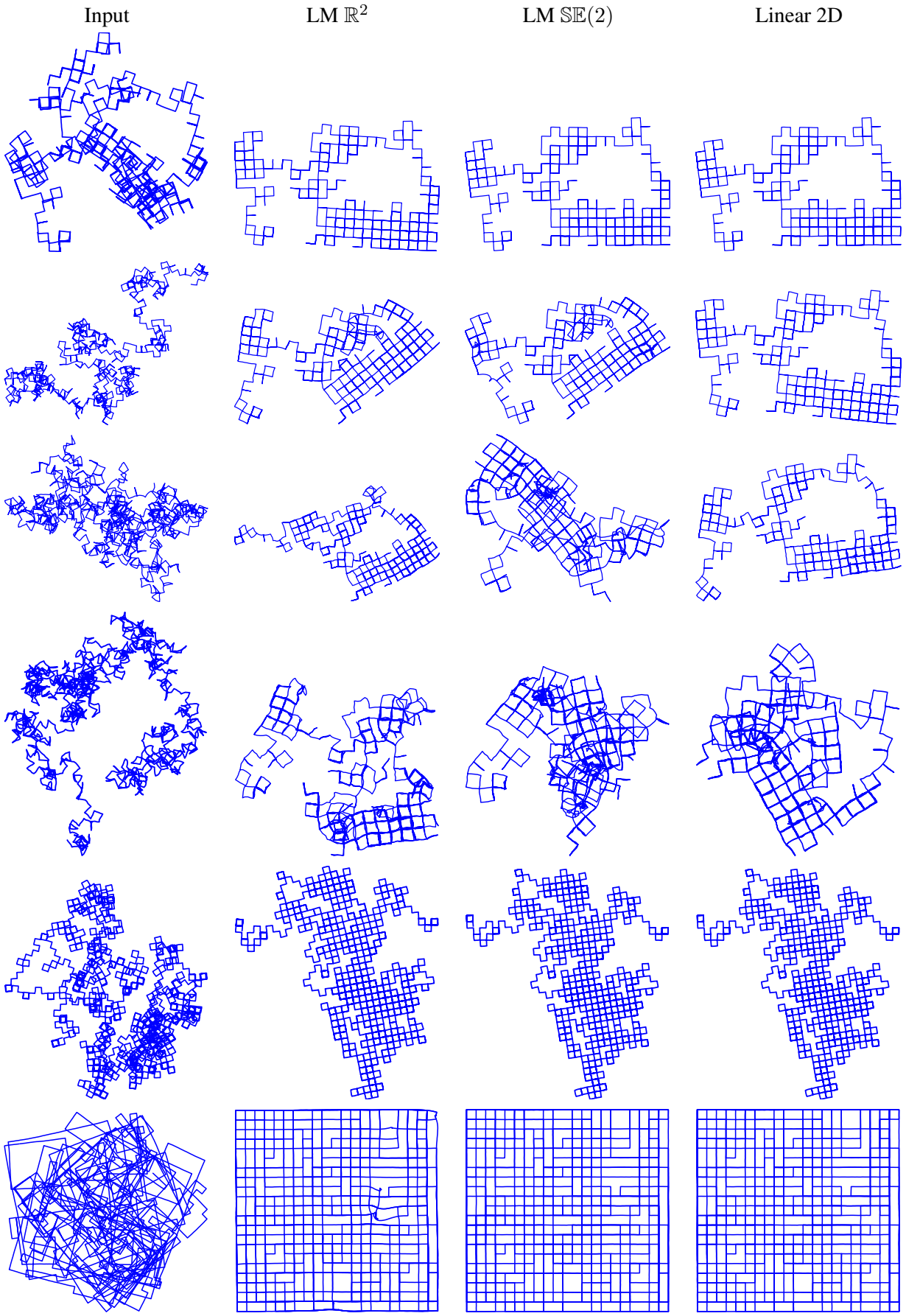


Figure 3: Comparison of different pose-graph optimisation methods on 2D synthetic datasets (from top to down) 4 versions of Manhattan3500 [21] dataset with increasing levels of noise [4], Manhattan10000 [10] and city10000 [14].

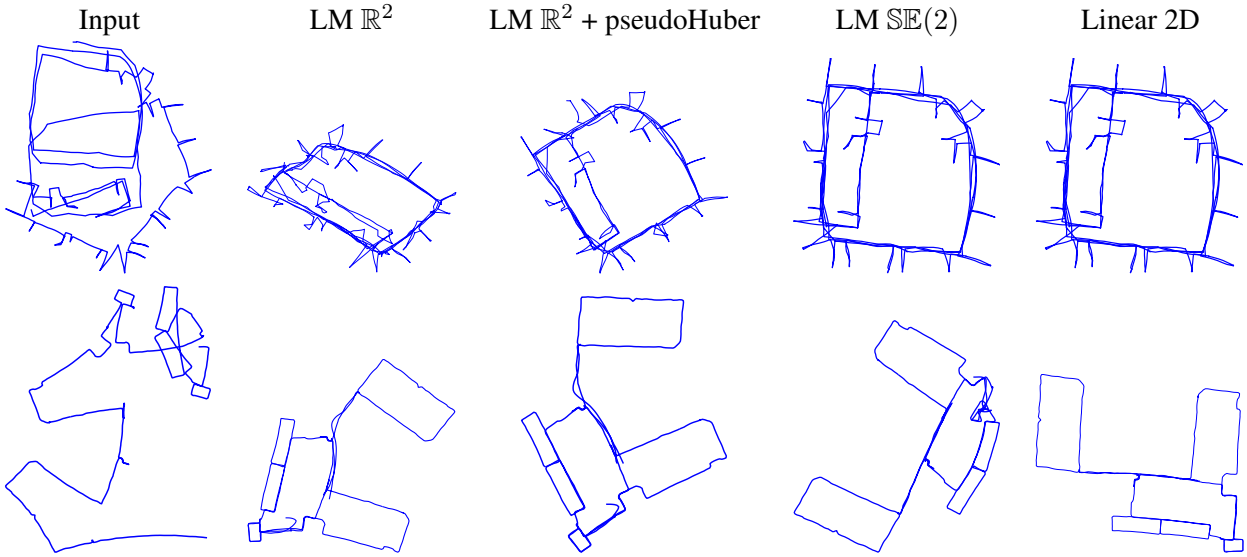


Figure 4: Comparison of different pose-graph optimisation methods on real 2D datasets (from top to down) Intel Research Lab. and MIT Killian Court, both from [16]

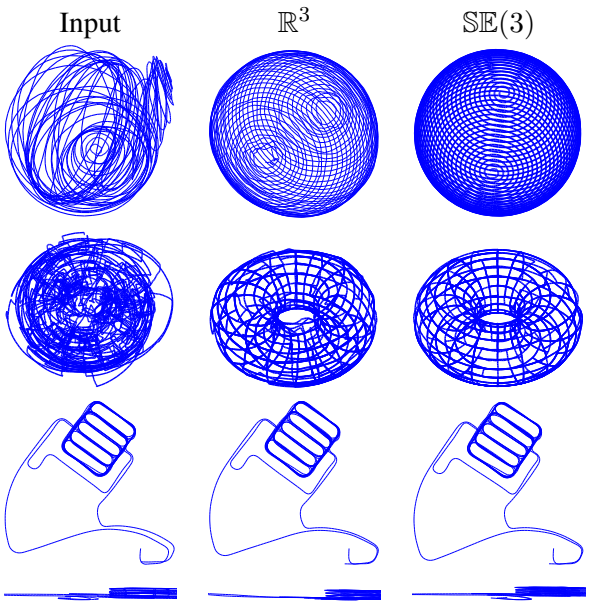
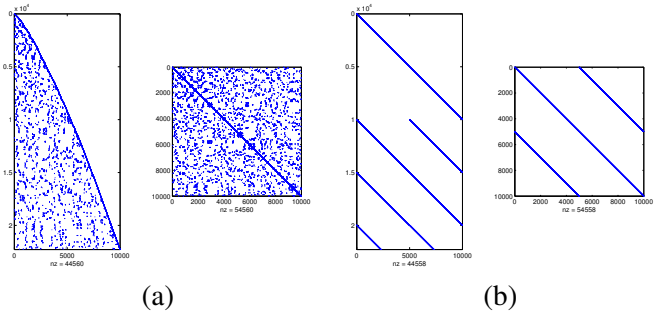


Figure 5: Comparison of different optimisation state vector parametrisation 3D datasets. From top to down, sphere2500, torus10000, parking-garage (plant view), parking-garage (side view).



(a) (b)

Figure 6: Jacobian and Hessian for (a) the torus10000 graph and for (b) a sphere graph with identical number of nodes and constraints. Note that the sparsity of both Hessians is approximately the same, but the more aleatory distribution of the non-zero elements over the Hessian matrix for the torus10000 datasets yields a higher computational cost of the optimisation.

Table 4: RMSE in translational units for synthetic datasets with available ground truth

Dataset	LM $\mathbb{R}^n$	LM $\mathbb{SE}(n)$
Manhattan 3500	0.4892	0.7982
City 10000	0.2875689	0.047672
sphere2500	6.7286	0.2568

art, while in the sphere2500 dataset the loss in accuracy of ours, though producing acceptable visual results Fig. 5, is numerically noticeable.

In the last experiment we have evaluated the effect of projecting a nearly planar 3D graph on a plane as described in Sec. 4.6 in a dataset acquired in our university campus while walking with a wearable omnidirectional camera camera attached to the head over a total distance of 886 m (Fig. 7). The initial odometry estimate is computed by a visual SLAM approach which includes including a trajectory scaling algorithm [12] to eliminate the scale ambiguity typical of monocular systems. Fig. 8 shows both the result after optimising the trajectory using our method both in its original 3D space and also in its projection over a 2D plane as explained in Sec. 4.6, taking advantage of the fact that the trajectory is quasi planar. It can be observed that the accuracy is improved by projecting the graph and optimising on $\mathbb{R}^2$. Also the computational cost is noticeably reduced from 1.292 s (10 iterations) in $\mathbb{R}^3$ to 0.2073 s (5 iterations) in $\mathbb{R}^2$.

Figure 7: Our helmet with catadioptric camera and example of images taking during the sequence.

6 Conclusions

In this paper we have presented an alternative formulation for the nodes and constraints in a graph for solving loop closure optimisation problems. Instead of building a pose graph in $\mathbb{SE}(n)$ including both translation and rotation we build a graph where nodes consists of points over the curve which represents the trajectory of the platform in the Euclidean space $\mathbb{R}^n$.

Our method has been evaluated in several public datasets yielding successful results, though the observation function used in our approach would seem initially prone to singularities. Compared to state of the art methods for pose graph optimisation yields similar accuracy and computational performance in the 2D datasets. In the 3D case however, our method suffers from a noticeable lack in accuracy, while the computational performance is generally improved but it seems to degrade severally in graphs with a large amount of constraints per nodes.

Although the experiments do not show a clear superiority of our graph parameterisation, we think that results are promising and it could benefit from an improvement in performance with an smarter choice or design of the cost function or the underlying optimisation algorithm. Also, it can find its utility in potential cases where the rotation of the platform is not known or considered or when the information matrix is not available for normalising the translation and rotation residuals.

7 Acknowledgements

This work was supported by Ministerio de Economía y Competitividad and European Union (project DPI2012-31781) and Ministerio de Educación, Cultura y Deporte (scholarship FPU12-05507)

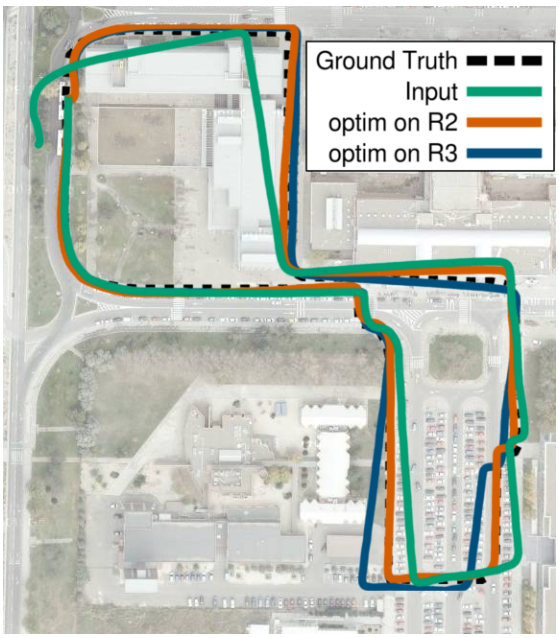


Figure 8: Evaluation of our approach in a data-set acquired with an omnidirectional camera, optimising on $\mathbb{R}^2$ after projecting the initial 3D graph on the dominant plane, and optimising on $\mathbb{R}^3$

References

[1] *A hierarchical wavelet decomposition for continuous-time SLAM*, 2014.

[2] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2007.

[3] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona. A linear approximation for graph-based simultaneous localization and mapping. In *Robotics: Science and Systems (RSS)*, 2011.

[4] L. Carlone and A. Censi. From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization. *IEEE Trans. on Robotics (T-RO)*, 30(2):475–492, 2014.

[5] G. Dubbelman and B. Browning. Closed-form online pose-chain slam. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 5190–5197, 2013.

[6] G. Dubbelman, I. Esteban, and K. Schutte. Efficient trajectory bending with applications to loop closure. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4836–4842, 2010.

[7] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots (AURO)*, 12(3):287–300, 2002.

[8] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Trans. on Robotics (T-RO)*, 21(2):196–207, 2005.

[9] M. K. Grimes, D. Anguelov, and Y. LeCun. Hybrid hessians for flexible optimization of pose graphs. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2997–3004, 2010.

[10] G. Grisetti, C. Stachniss, and W. Burgard. Nonlinear constraint network optimization for efficient map learning. *IEEE Trans. on Intelligent Transportation Systems (T-ITS)*, 10(3):428–439, 2009.

[11] D. Gutiérrez-Gómez and J. J. Guerrero. Curve-graph odometry: Removing the orientation in loop closure optimisation problems. In *Int. Conf. on Intelligent Autonomous Systems (IAS)*, 2014.

[12] D. Gutiérrez-Gómez, L. Puig, and J. J. Guerrero. Full scaled 3d visual odometry from a single wearable omnidirectional camera. In *IEEE/RSJ Int. Conf. on Intelligent Robot Systems (IROS)*, pages 4276–4281, 2012.

[13] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion (INFFUS)*, 14(1):57–77, 2013.

[14] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics (T-RO)*, 24(6):1365–1378, 2008.

[15] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A general framework for graph optimization. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 3607–3613, 2011.

[16] R. Kmmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[17] Y. Latif, C. Cadena, and J. Neira. Robust loop closing over time for pose-graph slam. *Int. J. of Robotics Research (IJRR)*, 2013.

[18] Y. Latif and J. Neira. Go straight, turn right: Pose graph reduction through trajectory segmentation using line segments. In *European Conference on Mobile Robots*, 2013.

[19] J. L. Martínez, J. Morales, A. Mandow, and A. García-Cerezo. Incremental closed–form solution to globally consistent 2d range scan mapping with two-step pose estimation. In *IEEE Int. Workshop on Advanced Motion Control (AMC)*, pages 252–257, 2010.

[20] E. Olson and P. Agarwal. Inference on networks of mixtures for robust robot mapping. *Int. J. of Robotics Research (IJRR)*, 32(7):826–840, 2013.

[21] E. Olson, J. J. Leonard, and S. J. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 2262–2269, 2006.

[22] F. C. Park. Distance metrics on the rigid-body motions with applications to mechanism design. *J. of Mechanical Design*, 117(1):48–54, 1995.

[23] B. Peasley and S. Birchfield. Fast and accurate poseslam by combining relative and global state spacess. In *Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[24] A. Pressley. *Elementary Differential Geometry*. Springer, 2001.

[25] H. Strasdat. *Local Accuracy and Global Consistency for Efficient Visual SLAM*. PhD thesis, Department of Computing, Imperial College London, 2012.

[26] H. Strasdat, J. M. M. Montiel, and A. Davison. Scale drift-aware large scale monocular slam. In *Robotics: Science and Systems (RSS)*, 2010.

[27] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph slam. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.