Curve-Graph odometry: Removing the orientation in loop closure optimisation problems

Daniel Gutiérrez-Gómez and J.J. Guerrero

Universidad de Zaragoza, Instituto de Investigación en Ingeniería de Aragón (I3A) and Departamento de Informática e Ingeniería de Sistemas (DIIS), 50018 Zaragoza, Spain

Abstract. In robot odometry and SLAM applications the real trajectory is estimated incrementally. This produces an accumulation of errors which gives raise to a drift in the trajectory. When revisiting a previous position this drift becomes observable and thus it can be corrected 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 main advantages of our method are the elimination of the need of mixing the orientation and position in the optimisation and the savings in computational cost due to the reduction of the dimension of the optimisation vector.

1 INTRODUCTION

Given the probabilistic nature of Simultaneous Localisation and Mapping (SLAM) techniques, and due to the incremental estimation, it is unavoidable an error build-up giving 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 applied to correct the drift.

The loop closure process can be divided into three steps: loop detection, computation of the loop closing constraint and the update of the trajectory with the new constraints. In visual SLAM, the first two steps, loop detection and loop constraint computation, can be jointly addressed from one of these three matching approaches [1]: image-to-image, image-to-map and map-to-map.

The last step 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 [2], 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 space transformation applied to both poses should not modify the cost of violating this constraint. This applies similarly to the loop closure constraint between two poses, say, α and β. In this case the loop closure must constraint the relative motion of pose α with respect to a reference frame attached to β or vice versa.

In a pose graph formulation, each pose includes a translation and a rotation and are usually represented as 2D or 3D transformations in the special Euclidean group which describes the rigid body kinematics in 2 ($\mathbb{SE}(2)$) or 3 ($\mathbb{SE}(3)$) dimensions. Although a space transformation can be described locally using a minimal parametrisation, globally the rotation part is subject to singularities and then its representation is usually over-parametrised.

Directly applying a standard optimisation on a Euclidean Space $\mathbb{R}^n$ introduces an error in the estimate resulting from the violation of the constraint induced by the over-parametrisation. For this reason a non-linear optimisation problem defined over the space of rigid transformations must be adapted such that the increments are computed using a minimal parametrisation.

In this paper, we propose to reformulate the loop closure optimisation problem using a state representation where the orientation of the poses is removed. 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 a 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:

- The optimisation could be performed in a Euclidean space, with no need to define an error function and special operators for non-Euclidean manifolds.
- We avoid mixing translation and rotation in the same optimised vector. Although these two magnitudes can be normalised by the information matrix, in the cases where this matrix is not available, an empirical factor must be used to normalise the translation.
- The number of degrees from 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 and since the cost of solving a linear system is cubic in the number of variables, it may involve important savings in computational cost.

2 RELATED WORK

We have noted that while there exist a lot of work towards increasing the efficiency and convergence speed of the optimisation algorithms for pose graphs, 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 [3] and g^{2}o [2] tend to exploit the structure of the graph to reduce the computational cost.

Another family of approaches introduce a relaxation on the problem, i.e., at each iteration they compute an update only in a subset of the nodes. Although these updates are only approximate, the robustness to local minima is generally increased. In this sense Duckett et al. proposed the use of Gauss-Seidel relaxation [4] and based on this, Frese et al. [5] introduced a multilevel relaxation (MLR). Olson et al. [6] 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 [7], Grisetti et al. extend Olson's method including a novel tree parametrisation for the poses and extending to 3D poses. Grimes et al. [8] propose to apply a stochastic relaxation while solving the linearised system around current estimate.

Martinez et al. [9] and Carlone et al. [10] 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 [11] proposed recently a method for orientation estimation which is more robust to local minima than state-of-the-art approaches.

3 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 $x_i = {}_W\mathbf{T}^i$.

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 contraints is provided by a module in charge of both loop detection and computation of transforms $\mathcal{T}_{\boldsymbol{LC}} = \{{}_{i_1}\hat{\mathbf{T}}^{j_1}_{LC},\ ...,\ {}_{i_L}\hat{\mathbf{T}}^{j_L}_{LC}\}$. 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 $\mathcal{S} = \{(1,2), ..., (N-1, N)\}$ is the set of pose pairs for the odometry constraints and $\mathcal{R} = \{(i_1, j_1), ..., (i_L, j_L)\}$ the set of pose pairs sharing a loop closure constraint.

3.2 Generalised optimisation on manifolds

The description of elements lying on a manifold usually require more parameters than dimensions has the manifold. Such is the case of elements in $\mathbb{SE}(n)$, described by transformation matrices. For each additional parameter a constraint is established. To avoid the violation of these constraints during iterative optimisation, updates of the estimation must be computed in the tangent space of the manifold using a minimal parameterisation. 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 [12]).

Using the $\boxminus$ operator, the error e_k resulting from violating the 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 $\boldsymbol{\Omega} = \text{diag}\,(\Omega_1, ..., \Omega_M)$ 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 by computing a first order Taylor expansion of the error at each iteration :

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

where $\boldsymbol{\delta} = (\delta_1 \cdots \delta_N)$ and abusing from notation $\breve{\mathbf{x}} \boxplus \boldsymbol{\delta} = \{\breve{x}_1 \boxplus \delta_1, ... \,, \breve{x}_N \boxplus \delta_N\}$.

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

$$\breve{\mathbf{x}} \leftarrow \breve{\mathbf{x}} \boxplus \boldsymbol{\delta}. \tag{6}$$

4 PROPOSED 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 non Euclidean Group 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 constraints between points are imposed modifying as less as possible the local properties of the curve. 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 responds 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, given only the set of positions composing the trajectory, to define an observation function for the constraints which keeps the error invariant to an arbitrary rigid motion applied to the poses implied in a constraint.

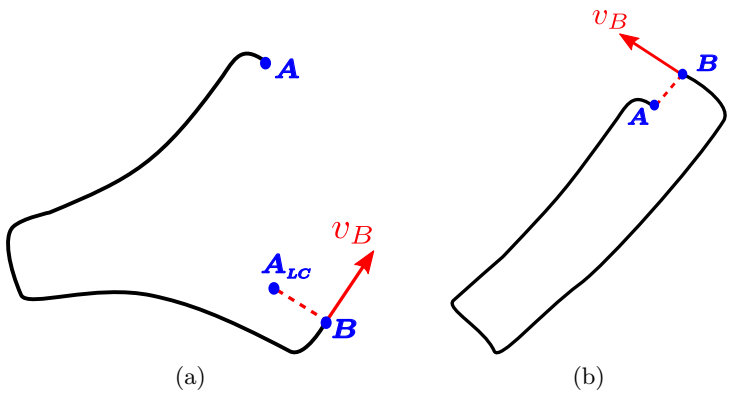

Fig. 1. (a) Curve with an open loop where the point **A** should be at the same position as $\mathbf{A_{LC}}$ and keep a relative orientation w.r.t. the vector tangent to the trajectory at **B**. (b) Intuition of how this loop should be closed.

4.1 Curves in 2D and 3D

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 $\mathbb{R}^n$.

$$\mathbf{r} : I \rightarrow \mathbb{R}^n. \tag{7}$$

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:

$$\kappa(t) = \frac{\mathbf{r}'^{\mathsf{T}}(t)\mathtt{I}_s\mathbf{r}''(t)}{\|\mathbf{r}'(t)\|^3}, \tag{8}$$

with

$$\mathtt{I}_s = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{9}$$

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 point of the curve. 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:

$$\kappa(t) = \frac{\|\mathbf{r}'(t) \times \mathbf{r}''(t)\|}{\|\mathbf{r}'(t)\|^3}, \tag{10}$$

$$\tau(t) = \frac{\mathbf{r}'''^{\mathsf{T}}(t)(\mathbf{r}'(t) \times \mathbf{r}''(t))}{\|\mathbf{r}'(t) \times \mathbf{r}''(t)\|^2}, \tag{11}$$

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

4.2 Vertex parameterisation and constraints

In our approach each node in the graph is parametrised as $x_i = \mathbf{r}^i_W = trans\left({}_W\mathbf{T}^i\right)$.

Analogously to previous section we define an observation function for the constraints:

$$f_{ij}(\mathbf{x}) \doteq f_k(\mathbf{x}) = f(\mathcal{F}_{\mathbb{R}^n}(\mathbf{r}^i_W), \mathbf{r}^j_W), \tag{12}$$

where $\mathcal{F}_{\mathbb{R}^n}(\mathbf{r}^i_W)$ is a function which extracts from the graph, the minimum number of poses backwards from $\mathbf{r}^i_W$ to define a reference frame for a given number of spatial dimensions n.

Given the observation function, the initial state of the graph $\hat{\mathbf{x}} = \{\hat{\mathbf{r}}^1_W,\ \hat{\mathbf{r}}^2_W,\ ...,\ \hat{\mathbf{r}}^N_W\}$, and the transforms between loop closure poses $\mathcal{T}_{\mathcal{LC}} = \{{}_{i_1}\hat{\mathbf{T}}^{j_1}_{LC},\ ...,\ {}_{i_L}\hat{\mathbf{T}}^{j_L}_{LC}\}$, the constraints for the optimisation problem only on the positions are computed as:

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

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

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 shake 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 a relative observation function $f_k(\mathbf{x})$ and then show that it is related to the properties of the curve. In next sections we will abuse of notation and merge the definition of points in the curve and positions ($\mathbf{r}_i \doteq \mathbf{r}^i_W$).

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

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

$$\Delta\mathbf{r}_0 = \mathbf{r}^i_W - \mathbf{r}^{i-1}_W, \qquad \Delta\mathbf{r}_1 = \mathbf{r}^j_W - \mathbf{r}^i_W. \tag{14}$$

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

$$f\left(\mathbf{r}^{i-1}_W, \mathbf{r}^i_W, \mathbf{r}^j_W\right) = \mathsf{R}\Delta\mathbf{r}_1 = \frac{1}{\|\Delta\mathbf{r}_0\|}\begin{pmatrix} \Delta\mathbf{r}_0{}^\mathsf{T} \\ \Delta\mathbf{r}_0{}^\mathsf{T}\mathtt{I}_s \end{pmatrix}\Delta\mathbf{r}_1, \tag{15}$$

and expresses the odometry vector $\Delta\mathbf{r}_1$ in a reference frame whose unit vector $\mathbf{e_x}$ is aligned with $\Delta\mathbf{r}_0$.

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 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:

$$\mathbf{r}'(i) = \mathbf{r}_i - \mathbf{r}_{i-1} = \Delta\mathbf{r}_0, \tag{16}$$

$$\mathbf{r}''(i) = \mathbf{r}_j - 2\mathbf{r}_i + \mathbf{r}_{i-1} = \Delta\mathbf{r}_1 - \Delta\mathbf{r}_0. \tag{17}$$

Note that for the first order derivative we have taken the backwards difference convention. We will 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 $\mathbf{r}_i$:

$$\kappa_i = \frac{\Delta\mathbf{r}_0{}^{\mathsf{T}}\mathtt{I}_s\Delta\mathbf{r}_1}{\|\Delta\mathbf{r}_0\|^3}, \tag{18}$$

and putting it into (15) we get:

$$f\left(\mathbf{r}_W^{i-1}, \mathbf{r}_W^i, \mathbf{r}_W^j\right) = \begin{pmatrix} \|\Delta\mathbf{r}_1\|\sqrt{1 - \dfrac{\|\Delta\mathbf{r}_0\|^4}{\|\Delta\mathbf{r}_1\|^2}\kappa_i^2} \\ \|\Delta\mathbf{r}_0\|^2\kappa_i^2 \end{pmatrix}. \tag{19}$$

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 the 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:

$$\Delta\mathbf{r}_{-\mathbf{1}} = \mathbf{r}_W^{i-2} - \mathbf{r}_W^{i-3}. \tag{20}$$

Thus the odometry function $f\left(\mathbf{r}_W^{i-2}, \mathbf{r}_W^{i-1}, \mathbf{r}_W^i, \mathbf{r}_W^j\right)$ will stablish quaternary constraints between positions. One initial attempt of defining the local odometry function is to 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

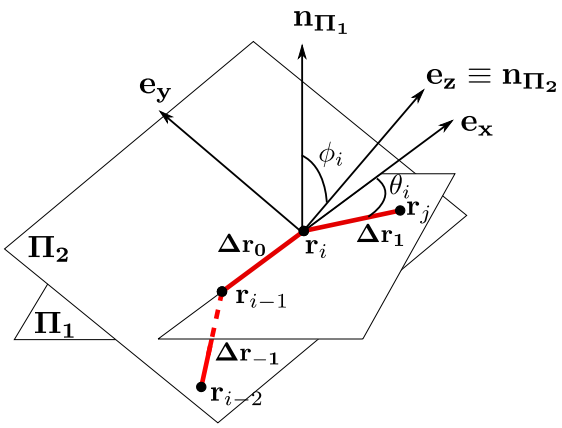

Fig. 2. Detail of a discrete 3D curve and the variation of its osculating plane

by $\Delta\mathbf{r_0}$ and $\Delta\mathbf{r_{-1}}$ (see Fig.2). Given that $\lfloor\Delta\mathbf{r_0}\rfloor$ is the antisymmetric matrix formed with vector $\Delta\mathbf{r}_0$, the following rotation matrix yields:

$$\mathsf{R} = \begin{pmatrix} \mathbf{e_x}^\mathsf{T} \\ \mathbf{e_y}^\mathsf{T} \\ \mathbf{e_z}^\mathsf{T} \end{pmatrix} = \begin{pmatrix} \frac{\Delta\mathbf{r_0}^\mathsf{T}}{\|\Delta\mathbf{r_0}\|} \\ -\frac{\Delta\mathbf{r_{-1}}^\mathsf{T}\lfloor\Delta\mathbf{r_0}\rfloor^2}{\|\lfloor\Delta\mathbf{r_0}\rfloor^2\Delta\mathbf{r_{-1}}\|} \\ -\frac{\Delta\mathbf{r_{-1}}^\mathsf{T}\lfloor\Delta\mathbf{r_0}\rfloor}{\|\lfloor\Delta\mathbf{r_0}\rfloor\Delta\mathbf{r_{-1}}\|} \end{pmatrix}, \tag{21}$$

which results in the following odometry function:

$$f_k\left(\mathbf{r}_W^i, \mathbf{r}_W^{i-1}, \mathbf{r}_W^{i-2}, \mathbf{r}_W^{i-3}\right) = \mathsf{R}\Delta\mathbf{r_1} = \begin{pmatrix} \frac{\Delta\mathbf{r_0}^\mathsf{T}\Delta\mathbf{r_1}}{\|\Delta\mathbf{r_0}\|} \\ -\frac{\Delta\mathbf{r_{-1}}^\mathsf{T}\lfloor\Delta\mathbf{r_0}\rfloor^2\Delta\mathbf{r_1}}{\|\lfloor\Delta\mathbf{r_0}\rfloor^2\Delta\mathbf{r_{-1}}\|} \\ -\frac{\Delta\mathbf{r_{-1}}^\mathsf{T}\lfloor\Delta\mathbf{r_0}\rfloor\Delta\mathbf{r_1}}{\|\lfloor\Delta\mathbf{r_0}\rfloor\Delta\mathbf{r_{-1}}\|} \end{pmatrix}. \tag{22}$$

However this odometry 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 thus we have to find another function free of them. We propose to use the following vector to express the curve constraints:

$$f\left(\mathbf{r}_W^{i-2}, \mathbf{r}_W^{i-1}, \mathbf{r}_W^{i}, \mathbf{r}_W^{j}\right) = \begin{pmatrix} \frac{\Delta\mathbf{r_0}^\mathsf{T}\Delta\mathbf{r_1}}{\|\Delta\mathbf{r_0}\|} \\ -\frac{\Delta\mathbf{r_{-1}}^\mathsf{T}\lfloor\Delta\mathbf{r_0}\rfloor^2\Delta\mathbf{r_1}}{\|\Delta\mathbf{r_0}\|^2\|\Delta\mathbf{r_{-1}}\|} \\ -\frac{\Delta\mathbf{r_{-1}}^\mathsf{T}\lfloor\Delta\mathbf{r_0}\rfloor\Delta\mathbf{r_1}}{\|\Delta\mathbf{r_0}\|\|\Delta\mathbf{r_{-1}}\|} \end{pmatrix}. \tag{23}$$

where the term $\|\lfloor \Delta\mathbf{r_0} \rfloor \Delta\mathbf{r_{-1}}\|$ has been substituted by $\|\Delta\mathbf{r_0}\|\|\Delta\mathbf{r_{-1}}\|$. Note that, although the singularity is removed, still remains an ambiguity when $\Delta\mathbf{r_{-1}}$ and $\Delta\mathbf{r_0}$ are aligned. In this case $\Delta\mathbf{r_1}$ can be rotated arbitrarily around $\Delta\mathbf{r_0}$ without varying the result of the observation function.

With this parametrisation, the constraints no longer correspond to the odometry vector expressed in a local coordinate frame. However, all the constraints are still expressed in the same length units and they encapsulate restrictions on local properties of the curve (length, curvature κ and torsion τ).

So, discretising the third derivative,

$$\mathbf{r}'''(i) = \Delta\mathbf{r_{-1}} + \Delta\mathbf{r_1} - 2\Delta\mathbf{r_0}, \tag{24}$$

and taking (16) and (17) for the first and second order derivatives we obtain the discretised expressions for curvature and torsion:

$$\kappa_i = \frac{\|\Delta\mathbf{r_0} \times \Delta\mathbf{r_1}\|}{\|\Delta\mathbf{r_0}\|^3}, \tag{25}$$

$$\tau_i = \frac{\Delta\mathbf{r}_{-\mathbf{1}}^{\mathsf{T}}(\Delta\mathbf{r_0} \times \Delta\mathbf{r_1})}{\|\Delta\mathbf{r_0} \times \Delta\mathbf{r_1}\|^2}. \tag{26}$$

Recalling (25) and (26) and applying the definitions of the cross and dot product and some trigonometric properties we get

$$f\left(\mathbf{r}_W^{i-2}, \mathbf{r}_W^{i-1}, \mathbf{r}_W^{i}, \mathbf{r}_W^{j}\right) = \begin{pmatrix} \|\Delta\mathbf{r_1}\|\sqrt{1 - \kappa_i^2 \frac{\|\Delta\mathbf{r_0}\|^4}{\|\Delta\mathbf{r_1}\|^2}} \\ \|\Delta\mathbf{r_0}\|\|\Delta\mathbf{r_{-1}}\|^2\kappa_i\kappa_{i-1}\sqrt{1 - \tau_i \frac{\kappa_i^2\|\Delta\mathbf{r_0}\|^8}{\kappa_{i-1}^2\|\Delta\mathbf{r_{-1}}\|^6}} \\ \tau_i\kappa_i^2 \frac{\|\Delta\mathbf{r_0}\|^5}{\|\Delta\mathbf{r_{-1}}\|} \end{pmatrix}, \tag{27}$$

and thus we verify that the new function, though not strictly an dometry function, still encapsulates a preservation of the local properties of the curve.

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 [2]. The experiments were performed in an Intel Core i5-2500 at 3.30 GHz.

For the 2D case we have compared three different approaches: optimisation on $\mathbb{SE}(2)$ with the CSparse solver and the Gauss Newton algorithm, a g^{2}o implementation of the linear approximation method for 2D pose graphs of Carlone et al. [10], and our optimisation on $\mathbb{R}^2$ with CSparse solver and the Gauss Newton algorithm. 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 (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, Gauss-Newton on $\mathbb{SE}(2)$ and $\mathbb{R}^2$ yield a similar performance. On the other hand, the linear approximation method seems to outperform them since it only needs two iterations to converge in all the considered cases, but it is restricted to 2D graphs.

The 3D case has been tested with the synthetic sphere data-set. We have used the CSparse solver in the three compared approaches: optimisation on $\mathbb{SE}(3)$ with Gauss-Newton, optimisation on $\mathbb{SE}(3)$ with Levenberg-Marquardt and optimisation in $\mathbb{R}^3$ with Levenberg-Marquadt (ours). The Gauss-Newton algorithm could not be used with our method since it produced a failure in the Cholesky factorisation. We think that this failure may be related with the ambiguity in the observation function commented in Sec. 4.4. Fig. 4 shows that while by optimising on $\mathbb{SE}(3)$, the correct solution is reached, our approach gets stuck in a local minima, which however, is still close to a correct spherical shape.

Table 2 shows that, as expected by the reduction of the number of optimised variables to the half, the time per iteration using our approach is significantly lower than in the optimisation on $\mathbb{SE}(3)$. Note that since the cost Cholesky factorisation algorithm has order $\mathcal{O}(n^3)$ one may expect a cost per iteration 8 times lower. However in practice, the cost of the sparse Cholesky factorisation is also affected by the density of the system matrix. Thus, the increase in density due to the use of quaternary, instead of binary constraints produces a cost overhead which yields a real cost reduction by a factor of 5.

This can also be appointed as the cause of the no gain in computation time in 2D. In this case the reduction of the dimensionality of the optimised vector yields a theoretical cost reduction by a factor of approximately 3.5, which has been verified in trials with a dense solver on small data-sets. Using the sparse solver, the sparsity of the system matrix plays also a role in the final cost of Cholesky decomposition, and then the ternary constraints of our approach produce a cost overhead which compensates the cost reduction from the reduced problem dimension.

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

Dataset	CSparse Gauss-Newton $\mathbb{R}^2$	CSparse Gauss-Newton $\mathbb{SE}(2)$	Linear 2D
Manhattan 3500	0.0465 (3 iters)	0.0453 (3 iters)	0.0446 (2 iters)
Manhattan 10000	1.315 (8 iters)	1.559 (8 iters)	0.3648 (2 iters)
Intel	0.0099 (2 iters)	0.0100 (2 iters)	0.0126 (2 iters)

Finally, we have also tested our method in a visual odometry estimate along a path of 886 m using a wearable omnidirectional camera and a visual SLAM estimation including a trajectory scaling algorithm [13]. The results shown in

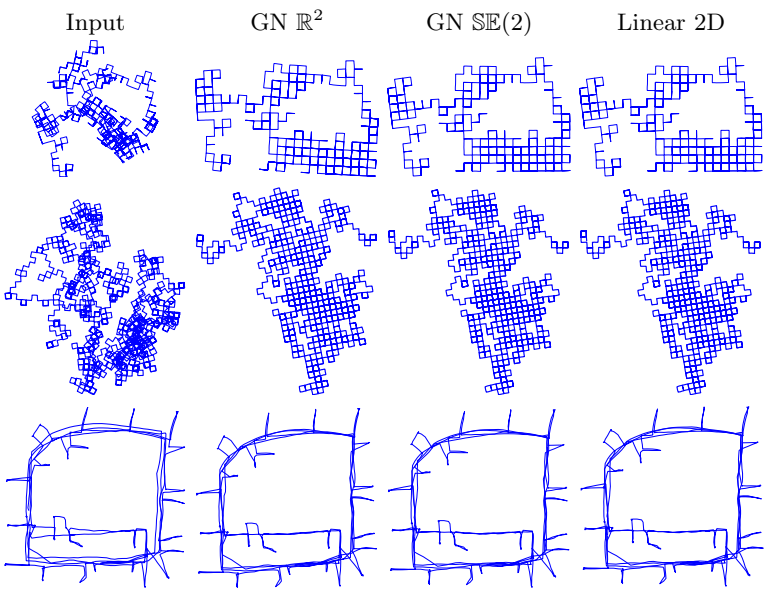

Fig. 3. Comparison of different pose-graph optimisation methods on the 2D datasets (from top to down) Manhattan3500, Manhattan10000 and Intel Research Lab.

Fig. 5 show the accuracy of our loop-closure method, performed over the 2D projection of the initial estimate.

6 CONCLUSION

In this paper we have presented a novel method to solve loop closure problems by pose graph optimisation. The method consists on optimising on the Euclidean Space $\mathbb{R}^n$ using only the position part of the poses, rather than in $\mathbb{SE}(n)$ with the complete poses. While the results for 2D graphs are similar to the obtained with state-of-the-art approaches, in 3D the reduction of the state vector leads to a great reduction of the computation time.

At the moment we have obtained in the 3D case a bit lower accuracy, which may be due to the ambiguous definition of the observation function in the 3D case to avoid singularities in straight parts. Although this situation is likely to occur in vehicles, with humanoid robots or in applications with human wearable sensors the oscillating nature of walking motion assures a non-straight local displacement in 3D even if the 2D displacement were aligned. In this case it is possible to define a unambiguous reference frame, which can improve the precision of the method.

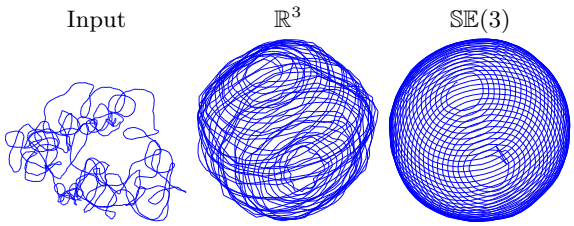

Fig. 4. Comparison of different optimisation state vector parametrisation the 3D synthetic sphere data-set.

Table 2. Convergence speed comparison of different optimisation approaches in the synthetic sphere data-set

CSparse Levenberg-Marquardt $\mathbb{R}^3$	CSparse Gauss-Newton $\mathbb{SE}(3)$	CSparse Levenberg-Marquardt $\mathbb{SE}(3)$
6 s (0.12 s/iter)	30s (0.62 s/iter)	30 s (0.62 s/iter)

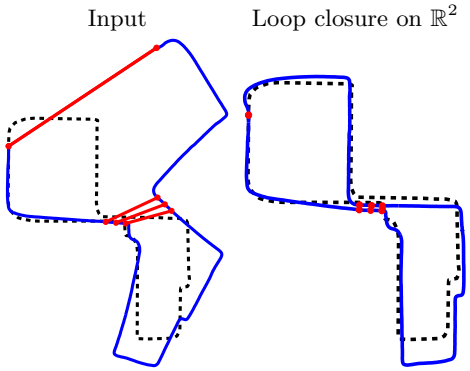

Fig. 5. Evaluation of our approach in a data-set acquired with an omnidirectional camera. Black dashed line represents the Ground Truth taken from Google Maps. The loop closure constraints are represented in red.

References

1. Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., Tardós, J.: A comparison of loop closing techniques in monocular slam. Robotics and Autonomous Systems (RAS) (2009)
2. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g^{2}o: A general framework for graph optimization. In: International Conference on Robotics and Automation (ICRA). (2011) 3607–3613
3. Kaess, M., Ranganathan, A., Dellaert, F.: isam: Incremental smoothing and mapping. IEEE Transactions on Robotics **24**(6) (2008) 1365–1378
4. Duckett, T., Marsland, S., Shapiro, J.: Fast, on-line learning of globally consistent maps. Auton. Robots **12**(3) (2002) 287–300

5. Frese, U., Larsson, P., Duckett, T.: A multilevel relaxation algorithm for simultaneous localization and mapping. IEEE Transactions on Robotics **21**(2) (2005) 196–207
6. Olson, E., Leonard, J.J., Teller, S.J.: Fast iterative alignment of pose graphs with poor initial estimates. In: International Conference on Robotics and Automation (ICRA). (2006) 2262–2269
7. Grisetti, G., Stachniss, C., Burgard, W.: Nonlinear constraint network optimization for efficient map learning. IEEE Transactions on Intelligent Transportation Systems **10**(3) (2009) 428–439
8. Grimes, M.K., Anguelov, D., LeCun, Y.: Hybrid hessians for flexible optimization of pose graphs. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2010) 2997–3004
9. Martínez, J.L., Morales, J., Mandow, A., GarcíaCerezo, A.: Incremental closedform solution to globally consistent 2d range scan mapping with twostep pose estimation. In: IEEE Int. Workshop on Advanced Motion Control. (2010) 252 257
10. Carlone, L., Aragues, R., Castellanos, J.A., Bona, B.: A linear approximation for graph-based simultaneous localization and mapping. In: Robotics: Science and Systems (RSS). (2011)
11. Carlone, L., Censi, A.: From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization. IEEE Transactions on Robotics **30**(2) (2014) 475–492
12. Hertzberg, C., Wagner, R., Frese, U., Schröder, L.: Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. Information Fusion **14**(1) (2013) 57–77
13. Gutierrez, D., Puig, L., Guerrero, J.J.: Full scaled 3d visual odometry from a single wearable omnidirectional camera. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robot Systems (IROS). (2012)