X CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA 2018
20 A 24 DE MAIO DE 2018 SALVADOR – BA – BRASIL

A POSE-GRAPH OPTIMIZATION TOOL FOR MATLAB

João Carlos Virgolino Soares, joaovirgolino@aluno.puc-rio.br[1]
Marco Antonio Meggiolaro, meggi@puc-rio.br[1]

[1]Pontifícia Universidade Católica do Rio de Janeiro - Department of Mechanical Engineering, Rua Marquês de São Vicente, 225, Gávea, CEP 22451-900, Rio de Janeiro, RJ

Abstract: *Several problems in mobile robotics need probabilistic formulations to handle the inherent uncertainty of motion and sensor measurements. Graph-SLAM is a probabilistic approach to the simultaneous localization and mapping problem that is based on maximum likelihood estimation and non-linear least squares optimization. It consists in generating a graph from the poses of the robot and from the constraints of measurements between poses, followed by the optimization of this graph to obtain a consistent trajectory. This paper presents the development and implementation of a pose-graph optimization tool for MATLAB. A pose quaternion representation is used and a manifold optimization approach is applied to handle attitude representation problems in the optimization. The system is evaluated using benchmark datasets available in the literature.*

Keywords: *Graph-SLAM, Non-linear Least Squares, Probabilistic Robotics, Maximum Likelihood Estimation*

1. INTRODUCTION

The simultaneous localization and mapping problem (SLAM) is one of the most important subjects in mobile robotics research due to the fact that is an essential step to a robot achieve full autonomy in an unknown environment. A robot needs an accurate model of the environment, as well as a precise estimate of its pose, in order to be fully autonomous.

A SLAM problem can be divided into two main parts: front-end and back-end. The front-end is heavily sensor-dependent because it uses sensor information to estimate the pose of the robot between states. However, pose estimation itself is inherently noisy and a probabilistic formulation is necessary to explicitly deal with the uncertainty using probability theory. This is done in the back-end step (Cadena *et al.*, 2016).

There are three main probabilistic formulations for SLAM: extended kalman filters, particle filters (Thrun *et al.*, 2005) and graph-based approaches. The first two are considered online SLAM. The graph-based is also called full SLAM because it gives an estimate of the complete trajectory of the robot, in contrast to the previous formulations that only estimate the most recent pose of the robot (Cadena *et al.*, 2016).

The Graph-SLAM formulation was introduced by Lu and Milios (1997) and consists in a graph-based representation of the states that is optimized using a non-linear least squares approach to create a maximum likelihood solution for the trajectory of the robot. In this formulation the nodes of the graph are the state variables and the edges are measurement relations affected by Gaussian noise.

There are several back-end implementations in the literature, such as TORO (Grisetti *et al.*, 2009) and g^2o (Kümmerle *et al.*, 2011), that were implemented in C++ and have both computational speed and precision. Wagner *et al.* (2011) implemented a MATLAB framework to make the technique accessible to non C++ developers, the Manifold ToolKit. Their framework uses a manifold optimization approach to deal with attitude representation problems in optimization algorithms. The implementation of Wagner et al. is based on the $\boxplus$-method, presented by Hertzberg (2008).

This implementation is based on the work of Wagner *et al.* (2011) for the 3D case. However, their framework is more general, extended to multi-sensor calibration problems, with an object-oriented implementation and a SO(3) exponential map. The present work is specifically designed to pose-graph optimization problems with a pose-quaternion representation.

Hence, this paper presents the development and implementation of a 2D and 3D pose-graph optimization tool for MATLAB. The system is evaluated using benchmark datasets available in the literature.

The work is organized as follows. Section 2 introduces the theoretical background of maximum likelihood estimation and Graph-SLAM. In section 3 is shown details of the 2D implementation. Section 4 details the 3D implementation. In section 5 is shown the evaluation of the system using 2D and 3D benchmark datasets. Section 6 presents the conclusions and also suggestions for improvements in future works.

2. METHODOLOGY

In the SLAM problem, the robot does not have prior knowledge of it own poses x, or access to a map m of the environment. Instead, only odometry information or other sensor measurements are available. In a graph-based approach, a graph is constructed using these raw sensor measurements. The nodes of the graph are the robot positions over time and the edges represent spatial constraints between two poses (Grisetti *et al.*, 2010). The graph optimization is done by estimating the state $\hat{x}$ that best explains the measurements, which is the state that maximizes the posterior belief $p(x|z)$ (Endres, 2015):

$$\hat{x} = \underset{x}{\operatorname{argmax}}\, p(x|z) \tag{1}$$

However, usually $p(x|z)$ is hard to obtain due to the non-linearity associated with the map between the measurements and the states. Using Bayes' Theorem Eq. (1) can be rewritten as:

$$\hat{x} = \underset{x}{\operatorname{argmax}}\, \frac{p(z|x)p(x)}{p(z)} \tag{2}$$

Since $p(z)$ is independent of x and there is no knowledge about the prior state, $p(x)$ and $p(z)$ can be dropped (Kümmerle *et al.*, 2011).

$$\hat{x} \propto \underset{x}{\operatorname{argmax}}\, p(z|x) \tag{3}$$

Eq. (3) represents the maximum likelihood solution. Assuming independent measurements, the problem becomes:

$$\hat{x} \propto \underset{x}{\operatorname{argmax}} \prod_{k=1}^{n} p(z_k|x) \tag{4}$$

With the assumption of locally Gaussian measurements, the likelihood of the measurements will also be Gaussian:

$$\hat{x} = \underset{x}{\operatorname{argmax}} \prod exp(-e_{ij}^T(x_i, x_j, z_{ij})\Omega_{ij} e_{ij}(x_i, x_j, z_{ij})) \tag{5}$$

where Ω is the inverse of the covariance matrix of the measurements, and e_{ij} represents the vector error function associated with the difference between prediction and observation. Taking the logarithm to transform the product into a sum:

$$\hat{x} = \underset{x}{\operatorname{argmin}} \sum_{ij} e_{ij}^T(x_i, x_j, z_{ij})\Omega_{ij} e_{ij}(x_i, x_j, z_{ij}) \tag{6}$$

Eq. (6) has the same structure of a non-linear least squares problem:

$$x^* = \underset{x}{\operatorname{argmin}}\, F(x) \tag{7}$$

where

$$F(x) = \sum e_{ij}^T \Omega e_{ij} \tag{8}$$

and can be solved using standard optimization methods such as Gauss-Newton or Levenberg-Marquardt. A solution to the non-linear least squares problem is to use the first order Taylor expansion around the initial guess $\breve{x}$ to approximate the error function, as stated in eq. (9).

$$e_{ij}(\breve{x} + \delta x) \approx e_{ij} + J_{ij}\delta_x \tag{9}$$

where J_{ij} is the Jacobian of the error function computed in $\breve{x}$. Thus, the cost function F of an observation between nodes i and j can be obtained rewritting a parcel of the sum in Eq. (6) with the local approximation of Eq. (9).

$$F_{ij}(\breve{x} + \delta x) \approx (e_{ij} + J_{ij}\delta_x)^T \Omega_{ij} (e_{ij} + J_{ij}\delta_x) \tag{10}$$

The global cost function can be found with the sum of all local approximations:

$$F(\breve{x} + \delta x) = \sum F_{ij}(\breve{x} + \delta x) \approx \sum (e_{ij}^T \Omega_{ij} e_{ij} + 2 e_{ij}^T \Omega_{ij} J_{ij} \delta_x + \delta_x^T J_{ij}^T \Omega_{ij} J_{ij} \delta_x) \tag{11}$$

Eq. (11) can be minimized solving the following linear system:

$$H \delta_x = -b \tag{12}$$

where

$$H = \sum J_{ij}^T \Omega_{ij} J_{ij} = J^T \Omega J \tag{13}$$

$$b = \sum J_{ij}^T \Omega_{ij} e_{ij} = J^T \Omega e \tag{14}$$

The solution for one iteration is then obtained by adding the increments to initial guess, as stated in Eq. (15).

$$x^* = \breve{x} + \delta_x \tag{15}$$

3. 2D IMPLEMENTATION

The poses of the robot are given by a translation vector $t_i = [x_i, y_i]$ and a rotation angle θ_i, that represents the orientation of the robot.

$$p = [t_i, \theta_i] \tag{16}$$

Each measurement between the nodes i and j is given by z_{ij}:

$$z_{ij} = [t_{ij}, \theta_{ij}] \tag{17}$$

The rotations of the robot are expressed with 2x2 rotation matrices, as shown in Eq. (18).

$$R_i = \begin{pmatrix} cos(\theta_i) & -sin(\theta_i) \\ sin(\theta_i) & cos(\theta_i) \end{pmatrix} \tag{18}$$

The error function is given by Eq. (19).

$$e_{ij} = z_{ij}^{-1} \begin{pmatrix} R_i^T (t_j - t_i) \\ \theta_j - \theta_i \end{pmatrix} \tag{19}$$

The Jacobian matrix is composed by the derivate of the error function in terms of each pose. However, as the error function of each measurement only depends on the values of two nodes, the Jacobian has the folowing structure:

$$J_{ij} = \begin{pmatrix} 0 \ldots 0 & \frac{\partial e_{ij}}{\partial x_i} & 0 \ldots 0 & \frac{\partial e_{ij}}{\partial x_j} & 0 \ldots 0 \end{pmatrix} \tag{20}$$

4. 3D IMPLEMENTATION

Oposed to 2D optimization, with a single normalized angle, the representation of orientation is very problematic in 3D. There are several parameterizations, for instance, euler angles, rotation matrices and unit quaternions. However, all three suffer from considerable drawbacks in problems such as pose-graph optimization. The use of Euler Angles is subject to singularities. When two of the three rotation axes are aligned, a DOF is lost, which is called the gimbal lock problem. To overcome this problem, a solution would be the use of an over-parametrized representation, such as rotation matrices or unit quaternions. The problem with the use of rotation matrices is the necessity to impose six non-linear constraints to ensure orthogonality and unit length of the columns. In other words, to ensure it remains in SO(3) (Grassia, 1998).

Quaternions are more suitable for optimization problems than rotation matrices due to the number of constraints that need to be maintained at every iteration. Quaternions were first described by W. R. Hamilton in 1843 (Jia, 2016), and can be seen as a generalization of complex numbers, with three different imaginary parts (Horn, 2001). In Eq.(21) is shown a general form of a quaternion.

$$q = q_x i + q_y j + q_z k + q_r \tag{21}$$

where

$$i^2 = j^2 = k^2 = ijk = -1 \tag{22}$$

and $q_x, q_y, q_z, q_r \in \mathbb{R}$. Thus, it is a sum of a scalar q_r and a vector part $q_v = [q_x, q_y, q_z]$.

The set of unit-lenght quaternions, also called unit quaternions, are a sub group of quaternions that are used to represent rotations. They satisfy the condition $\|q_u\| = 1$. The unit quaternion is given by Eq. (23). A full quaternion belongs to the $\mathbb{R}^4$ space. However, the unit quaternion belongs to a subspace of $\mathbb{R}^4$ called S^3, which represents the unit sphere in $\mathbb{R}^4$ (Gallier, 2012), (Ude,1999).

$$q_u = \frac{q}{\|q\|} = \frac{1}{\sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}} q \tag{23}$$

A unit quaternion just need to maintain its unit lenght throughout the optimization process. However, the addition of this constraint degrades the performance of the algorithm (Grassia, 1998). The problems about the quaternion parameterization occur because rotations have three DOF and the quaternion can change in four directions. Since estimation algorithms, in general, expect variables from euclidean vector spaces (Hertzberg, 2008), the goal is to use a representation with three parameters, such as euler angles, but without singularities. However, there is no SO(3) parameterization with only three parameters that has no singularities (Hertzberg, 2008).

To overcome these problems, the state is globally represented by a unit quaternion, but local perturbations around the current state have a minimal representation, ideally behaving as an euclidean space (Hertzberg, 2008). This parametrization is related to manifold theory and exponential maps.

A manifold is a mathematical space that can be locally approximated by an euclidean space, but it is not on a global scale (Lee, 2001). In other words, "every point of a manifold has a neighborhood that can be mapped bidirectionally to $\mathbb{R}^n$"(Hertzberg *et al.*, 2013).

The space of rotations and S^3, the unit quaternions, are manifolds and can be locally mapped to a euclidean space. Therefore, the parameterization problem can be dealt with using unit quaternions to represent the orientation of the state, and defining a operator $\boxplus$ that maps a local variation in the euclidean space to a variation on the manifold (Grisetti *et al.*, 2010).

The $\boxplus$-method, developed by Hertzberg (2008), defines the mapping functions between the manifold and the euclidean spaces, which are called the exponential and logarithmic maps. The operator $\boxplus$, stated in Eq. (24), represent the exponential map, which performs a rotation around axis δ with an angle $\|\delta\|$, accoding to Hertzberg *et al.* (2013).

$$p \boxplus \delta = p\; exp\Big(\frac{\delta}{2}\Big) \tag{24}$$

The operator $\boxminus$, stated in Eq. (25), represent the logarithmic map, which computes the rotation from p to q. The global difference in manifold space is mapped to a local perturbation in euclidean space (Grisetti *et al.*, 2010).

$$q \boxminus p = 2 \cdot log(p^{-1} q) \tag{25}$$

According to Ude (1999) and Hertzberg (2008), the exponential and logarithmic functions are given by the following equations, considering a quaternion with real part w and vector part u.

The exponential map function maps a vector v into a unit quaternion q:

$$exp: \; \mathbb{R}^3 \rightarrow \mathrm{S}^3 \tag{26}$$

$$exp(v) = q = \begin{cases} \Big[sin(\|v\|)\frac{v}{\|v\|}, cos(\|v\|)\Big] & \text{for} \quad \|v\| \neq 0 \\ [0,0,0,1] & \text{for} \quad v = 0 \end{cases} \tag{27}$$

The logarithmic map function maps a unit quaternion q into a vector v:

$$log : \mathrm{S}^3 \rightarrow \mathbb{R}^3 \tag{28}$$

$$log(q) = v = \begin{cases} 0 & \text{for} \quad u = 0 \\ \frac{atan(\|u\|/w)}{\|u\|}u & \text{for} \quad u \neq 0, w \neq 0 \\ \frac{\pi/2}{\|u\|}u & \text{for} \quad w = 0 \end{cases} \tag{29}$$

Thus, the pose of the robot is represented by a translation vector and a unit quaternion:

$$x_i = [x, y, z, q_x, q_y, q_z, q_r] \tag{30}$$

The error function can be approximated as:

$$e_{ij} = e_{ij}(\breve{x} \boxplus \delta_x) \backsimeq e_{ij} + J_{ij}\delta_x \tag{31}$$

where e_{ij} is the difference between the predicted and the actual measurement, as stated in Eq. (32)

$$e_{ij} = \hat{z}_{ij} \boxminus z_{ij} \tag{32}$$

The Jacobian is given by Eq. 33.

$$J_{ij} = \frac{\partial e_{ij}(\breve{x} \boxplus \delta_x)}{\partial \delta_x} \tag{33}$$

However, now the Jacobian matrix is computed numerically, according to Hertzberg (2008). A small perturbation is applied for each degree of freedom.

$$J_{ij} = \frac{e_{ij}(x \boxplus dv_j) - e_{ij}(x)}{d} \tag{34}$$

where e_{ij} is the error function, d is a small positive scalar and v_j is the unitary vector corresponding to the DOF.
Thus, the incremental addition to the initial guess is defined by the exponential map:

$$x^* = \breve{x} \boxplus \delta_x \tag{35}$$

The operator $\boxplus$ first converts the rotational part of δ_x to a full quaternion and then apply the transformation to $\breve{x}$ (Grisetti *et al.*, 2010), (Ude, 1998).

5. DATASET EVALUATION

The implementation was evaluated using several datasets available in the literature. There are two main formats to represent graph files: TORO and g^2o. They differ with respect to the ordering of the elements of the information matrix.

5.1 2D Evaluation

The Intel Dataset was chosen for the 2D evaluation, a benchmark dataset with real data acquired at the Intel Research Lab in Seattle, consisting of raw measurements from wheel odometry and laser range finder. Its graph contain 1228 poses and 1505 constraints. The graph is written in the TORO format. All nodes are represented by an ID, x, y and θ values, which correspond to the initial odometry poses.

All edge lines have the format: "IDfrom IDto x y θ I11 I12 I22 I33 I13 I23". The first two numbers "IDfrom IDto"correspond respectively to the ID of observing and observed nodes i and j. The x and y values compose the translation vector between nodes, and θ correspond to the rotation angle between nodes. The numbers "I11 I12 I22 I33 I13 I23"are the 6 top triangular elements of the 3x3 information matrix corresponding to each measurement. As the information matrix is symmetric, it becomes:

$$\Omega_{ij} = \begin{bmatrix} I11 & I12 & I13 \\ I12 & I22 & I23 \\ I13 & I23 & I33 \end{bmatrix} \tag{36}$$

Fig. 1 shows the initial pose-graph corrupted by drift in odometry estimation and measurement errors. The blue dots are the poses of the robot, and the red lines are the measurement constraints, derived from scan matching. In Fig. 2 the graph is successfully optimized and it is closest to the real trajectory of the robot. In Fig. 3 is shown a comparative image of the same dataset optimized by a method called MOLE2D, developed by Carlone and Censi (2012).

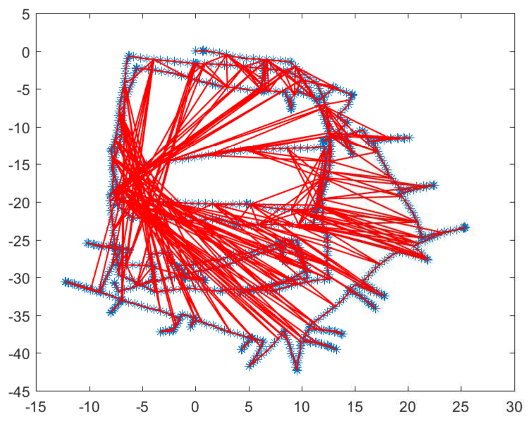

Figura 1: Intel - Initial corrupted pose-graph

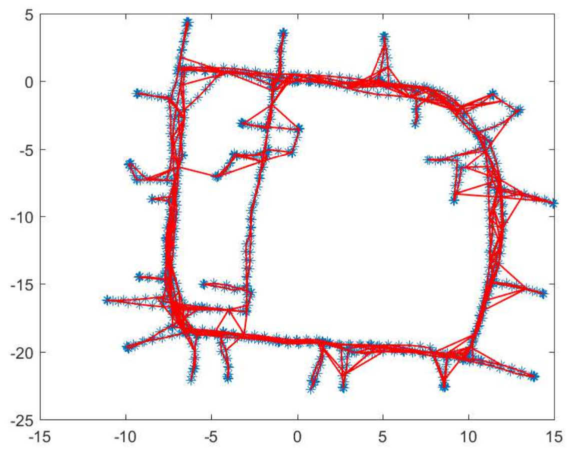

Figura 2: Intel Optimized pose-graph

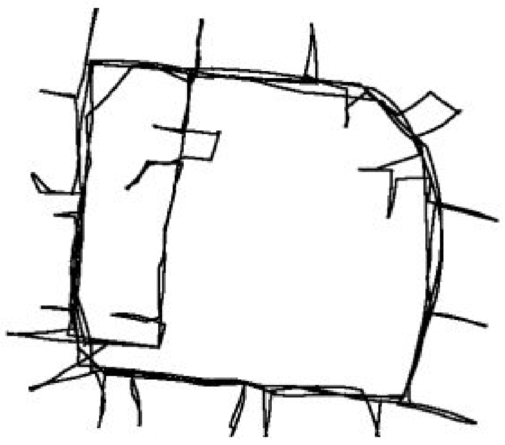

Figura 3: MOLE 2D Optimization

In Fig. 4 is shown the logarithmic global error per iteration. In only four iterations the system was able to optimize the entire graph, which shows the robustness of this implementation.

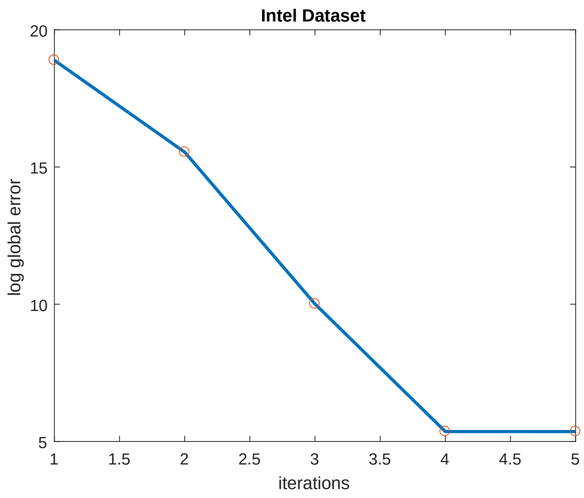

Figura 4: Intel dataset - Global error per iteration

5.2 3D Evaluation

For the 3D evaluation, the nodes are listed in the format "ID x y z q_x q_y q_z q_r", which corresponds to the number of the pose and its respective 3D position and orientation, represented by a unit quaternion. All edge lines have the format: "IDfrom IDto x y z q_x q_y q_z q_r I11, I12,...,I16, I22, ... I66". The first two numbers "IDfrom IDto"correspond to the ID of observing and observed nodes i and j. The x, y and z compose the translation vector between nodes, and q_x q_y q_z q_r compose the unit quaternion that represent the rotation between two nodes. The numbers I11 ... I66 are the 21 top triangular elements of the 6x6 information matrix, stated in Eq. (37). As the information matrix is symmetric, only 21 number are necessary to fill the entire matrix.

$$\Omega_{ij} = \begin{bmatrix} I11 & I12 & I13 & I14 & I15 & I16 \\ & I22 & I23 & I24 & I25 & I26 \\ & & I33 & I34 & I35 & I36 \\ & & & I44 & I45 & I46 \\ & \iddots & & & I55 & I56 \\ & & & & & I66 \end{bmatrix} \qquad (37)$$

The following 3D dataset represents the movement of a robot on a surface of a sphere. The graph has 2500 poses and 4949 contraints. Fig. 5 shows the initial graph configuration, a sphere corrupted by noise. Figure. 6 shows that the system is able to correctly optimize the graph, displaying the optimized sphere. Figure 7 shows the global error per iteration of the evaluation in logarithmic scale.

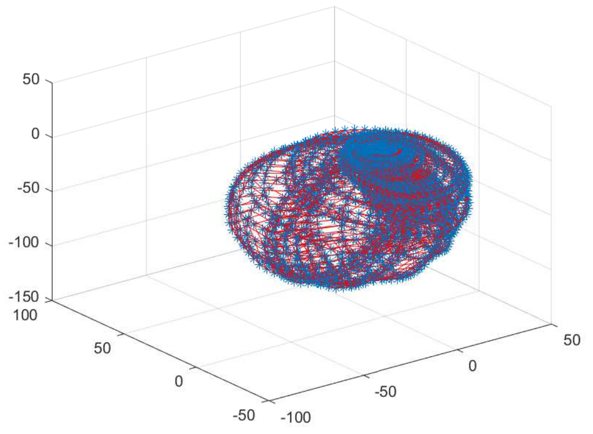
Figura 5: Initial pose-graph

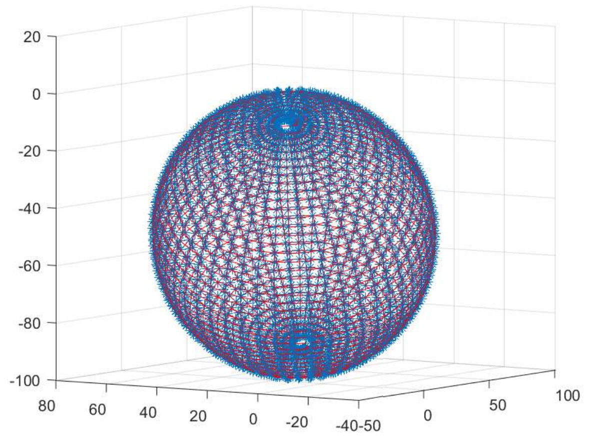
Figura 6: Optimized pose-graph

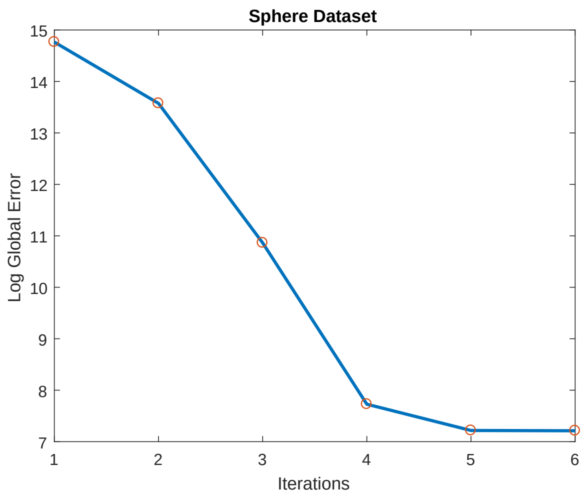

Figura 7: Sphere dataset - Global error per iteration

Another evaluation was made using real data aquired from an instrumented car at the Stanford parking garage. The parking garage dataset, provided by Carlone *et al.* (2015), has a graph with 1661 poses and 6275 constraints. Figure 8 shows the place were data was aquired. In Fig. 9 is shown the initial corrupted graph, and Fig. 10 shows the optimized trajcetory of the robot. Figure 11 shows the global error per iteration of the evaluation in logarithmic scale.

Figura 8: Stanford garage

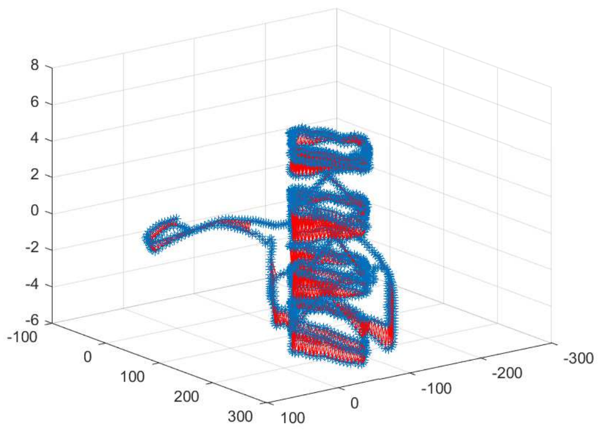
Figura 9: Initial pose-graph

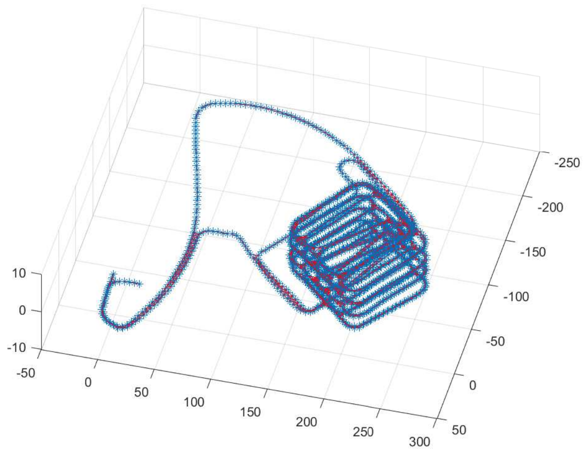
Figura 10: Optimized pose-graph

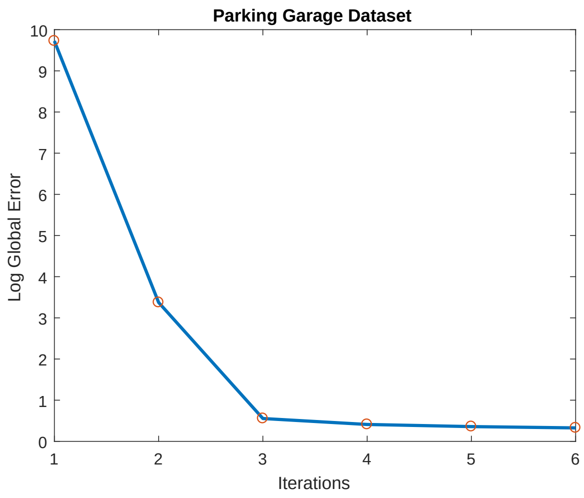

Figura 11: Garage dataset - Global error per iteration

6. CONCLUSION

In this paper an implementation of a pose-graph optimization tool for MATLAB, that can be used with 2D or 3D data, was presented. The technique is based on a maximum likelihood estimation and non-linear least squares optimization. A manifold optimization approach with a unit quaternion representation was used to handle attitute representation problems. The accuracy of the implementation was stated by benchmark dataset evaluation, with simulated and real world data, that assured the convergence of the initial corrupted trajectories to the desired result in few iterations. Future improvements include the use of other methods to solve the non-linear least squares problem, such as stochastic gradient descent.

7. ACKNOWLEDGEMENTS

The authors of this work would like to thank CAPES and CNPq for the financial support.

8. REFERENCES

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I. and Leonard, J., 2016. "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age". *IEEE Transactions on Robotics*, Vol. 32, pp. 1309–1332.

Carlone, L. and Censi, A., 2012. "From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization". *CoRR*, Vol. abs/1211.3063. URL `http://arxiv.org/abs/1211.3063`.

Carlone, L., Tron, R., Daniilidis, K. and Dellaert, F., 2015. "Initialization techniques for 3d slam: a survey on rotation estimation and its use in pose graph optimization". In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, pp. 4597–4604.

Endres, F., 2015. *Robot Perception for Indoor Navigation*. Ph.D. thesis, University of Freiburg.

Gallier, J., 2012. "Notes on differential geometry and lie groups".

Grassia, F.S., 1998. "Practical parameterization of rotations using the exponential map". *J. Graph. Tools*, Vol. 3, No. 3, pp. 29–48. ISSN 1086-7651. doi:10.1080/10867651.1998.10487493. URL `http://dx.doi.org/10.1080/10867651.1998.10487493`.

Grisetti, G., Kümmerle, R., Stachniss, C. and Burgard, W., 2010. "A tutorial on graph-based slam". *IEEE Intelligent Transportation Systems Magazine*, Vol. 2, pp. 31 – 43.

Grisetti, G., Stachniss, C. and Burgard, W., 2009. "Nonlinear constraint network optimization for efficient map learning". *Trans. Intell. Transport. Sys.*, Vol. 10, No. 3, pp. 428–439. ISSN 1524-9050.

Hertzberg, C., 2008. "A framework for sparse, non-linear least squares problems on manifolds".

Hertzberg, C., Wagner, R., Frese, U. and Schröder, L., 2013. "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds". *Information Fusion*, Vol. 14, No. 1, pp. 57–77.

Horn, B.K., 2001. "Some notes on unit quaternions and rotation".

Jia, Y.B., 2016. "Quaternions and rotation".

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K. and Burgard, W., 2011. "g2o: A general framework for graph optimization". In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Lee, J.M., 2001. "Introduction to smooth manifolds".

Lu, F. and Milios, E., 1997. "Globally consistent range scan alignment for environment mapping". *Autonomous Robots*, Vol. 4, pp. 333–349.

Thrun, S., Burgard, W. and Fox, D., 2005. *Probabilistic Robotics*. MIT Press.

Ude, A., 1998. "Nonlinear least squares optimisation of unit quaternion functions for pose estimation from corresponding features". In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*. IEEE, Vol. 1, pp. 425–427.

Ude, A., 1999. "Filtering in a unit quaternion space for model-based object tracking". *Robotics and Autonomous Systems*, Vol. 28, No. 2-3, pp. 163–172.

Wagner, R., Birbach, O. and Frese, U., 2011. "Rapid development of manifold-based graph optimization systems for multi-sensor calibration and slam". In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 3305–3312.

9. RESPONSIBILITY NOTICE