ICP-based pose-graph SLAM

Ellon Mendes, Pierrick Koch, Simon Lacroix

ICP-Based Pose-Graph SLAM

Ellon Mendes[1] and Pierrick Koch[1] and Simon Lacroix[1]

***Abstract*— Odometry-like localization solutions can be built upon Light Detection And Ranging (LIDAR) sensors, by sequentially registering the point clouds acquired along a robot trajectory. Yet such solutions inherently drift over time: we propose an approach that adds a graphical model layer on top of such LIDAR odometry layer, using the *Iterative Closest Points* (ICP) algorithm for registration. Reference frames called *keyframes* are defined along the robot trajectory, and ICP results are used to build a pose graph, that in turn is used to solve an optimization problem that provides updates for the keyframes upon loop closing, enabling the correction of the path of the robot and of the map of the environment. We present in details the configuration used to register data from the Velodyne High Definition LIDAR (HDL), and a strategy to build local maps upon which current frames are registered, either when discovering new areas or revisiting previously mapped areas. Experiments show that it is possible to build the graph using data from ICP and that the loop closings in the graph level reduces the overall drift of the system.**

I. Introduction

There is a wide corpus on the use of positioning techniques in robotics that exploit on-board sources of data as input to localization systems. If LIDAR sensors, like the Velodyne HDL sensors, have primarily been used for obstacle detection, they have rapidly been exploited for localization, using scan registration algorithms. These algorithms find the transformation that best aligns the points of one point cloud with respect to a reference point cloud. They have been used for several LIDAR odometry solutions, that compute the overall robot position by integrating elementary displacement estimates. Some solutions perform registration on the basis of matches established between features extracted from the scans [1], others directly exploit the scan points [2].

The main scan registration technique is the *Iterative Closest Points* (ICP) algorithm, originally introduced in [3]. ICP is a rather simple and modular algorithm, but numerous variants have been developed along the years, so that selecting the proper configuration and its parameters normally requires empirical tests and experience. The designer needs to consider the environment in which the robot evolves, and especially the sensor being used, the amount of data to process, how the data is organized, the measurement errors in the data, and so on.

When properly configured, the scan registration is precise. But as it is inherent to any odometry systems, the accumulation of errors made at every registration leads to a drift of the overall position estimate. In the absence of any prior map of the environment, resorting to a *Simultaneous Localization and Mapping* (SLAM) approach is the only way to reduce this drift. A recent trend is to use probabilistic graphical model theory to solve SLAM problems through optimization [4]. By exploring the sparsity of the SLAM problem, it is possible to considerably reduce the computation time needed to converge to a solution. Also, the graphical models used to model the optimization problem provide a powerful layer of abstraction that helps to better understand the problem and to design powerful solutions, like the iSAM2 algorithm [5].

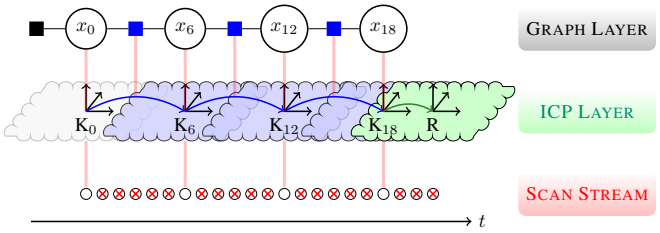


Fig. 1: Overview of the system, composed of two layers. Red vertical lines exhibit the links between the layers associations (see text for a full explanation).

The main objective of this work is to exploit scan registration algorithm and graphical model optimization to build a ICP-based localization system for autonomous mobile ground robots equipped with LIDAR sensors, for missions in semi-structured environments. Besides the integration of these two techniques, the configuration of the registration algorithm to work with data acquired from Velodyne HDL point clouds is precisely described.

This work has similarities with the work in [6] (a 3D extension of the work presented in [7] for 2D) that also uses ICP to localize the robot and achieves a global consistent map on loop closures using graph based optimization. The differences between this work and [6] will be detailed in the conclusion.

II. System Overview

This section provides an overview of the system, and the outline for of the remainder of the paper. The diagram shown in Fig. 1 describes an hypothetical state of the overall system after processing some scans. It exhibits the two main layers plus the scan stream, which is simply the history of acquired scans. The ICP layer is composed by *keyframes* K_i, and their associated scans (cloud shapes in the image), also called *keyframe scans*. To each keyframe K_i is associated a node x_i in the graph layer. Some of these keyframe scans are selected to compose *local maps* (blue clouds), that are used as reference input for the ICP process. The other keyframe scans (gray clouds) are stored, and may be later used to compose a local map if the robot revisits this part of the environment. The robot frame R is always expressed

[1]LAAS-CNRS, INSA, Université de Toulouse, CNRS, Toulouse, France.

with respect to the closest keyframe in the current local map. The system does not require a complete map of the environment, yet such a map can be computed on demand as the concatenation of all keyframe scans.

As the robot starts, a first keyframe K_0 is associated to the first acquired scan. The associated node x_0 is also created in the graph, and a factor representing the initial robot position with respect to the world origin is added to the graph (black square).

In nominal mode, every time a new scan (green cloud) is available at the current robot position R, the ICP finds the transformation that aligns it with the current local map (green arrow), correcting the robot position. If the overlap between the current scan and the current local map is lower than a threshold, a new keyframe is created – otherwise the scan is just discarded (crossed circles in the scan stream level). Section III depicts the ICP algorithm and how it is configured for the Velodyne HDL scans.

When a new keyframe is created, it is initialized with the corrected robot frame: a new frame variable is then added as a node to the graph, as well as a factor (blue squares) containing the transformation between the new and former keyframes (blue arrows). Finally the local map is rebuilt by incorporating the newest keyframe scan and removing the furthest keyframe scan. Section IV depicts the selection of keyframes and management of the local maps. The graph layer and the theory behind are presented in Section VI.

When the system detects a potential loop closing between two keyframes (not shown in Fig. 1 for simplicity), a local map is built around the oldest between these loop closing keyframes. Then an ICP call tries to align the scan of the other loop closing keyframes with this local map. If the ICP is successful, a new factor is added to the graph between the variables associated with these loop closing keyframes, closing a loop in the graph level. An optimization using the graph data is then triggered (also Section VI). The loop closing process ends with the repositioning of keyframes using the optimization result, and local maps are reconstructed using these new keyframe values. The robot frame R being expressed in the closest keyframe, it is also updated by the loop closing process. Section V presents the details of this loop closing procedure.

The following sections present the details all these processes, followed by Section VII with the presentation and analyses of the results obtained, and Section VIII that concludes the paper.

III. Point Cloud Registration

The first building block of the system is the well known *Iterative Closest Points* (ICP) registration algorithm. This section presents an overview of the ICP algorithm, followed by the details of the ICP solution designed for this work.

A. Overview of ICP

We present the ICP algorithm according to the in-depth review [8]. The ICP is responsible to find the transformation that best aligns a geometric shape called *reading*, to another shape called *reference*. This operation is known as *registration*. In our case, a shape $\mathcal{S}$ is a set of 3D points extracted from a LIDAR scan: as it is usual for ICP implementations, filters are applied to augment the points characteristics (e.g. add normal vectors), or to remove points that does not bring valuable information for the registration.

Let ${}^{A}\mathcal{P}$ the reading set, ${}^{B}\mathcal{Q}$ the reference set, and ${}^{A}\mathcal{P}'$,${}^{B}\mathcal{Q}'$ the filtered version of the respective sets. The registration algorithm estimates the transformation ${}^{A}\mathcal{T}_{B}$ that minimizes an error function $e(\mathcal{P}, \mathcal{Q})$:

$$ {}^{A}\mathcal{T}_{B} = \arg\min_{\mathcal{T}}(e(\mathcal{T}({}^{A}\mathcal{P}'), {}^{B}\mathcal{Q}')) \tag{1} $$

where $\mathcal{T}(\mathcal{S})$ is the transformation applied to a set $\mathcal{S}$.

The error function is in fact computed on pairs of points that have been associated between the input sets. The association is produced by a *matching function*, and is usually solved by associating to each point of the reading the closest point in the reference. For the sake of robustness, weights can be provided by an *outlier rejection function* to change the influence of the matches in the error function. Thus, if $\mathcal{M} = \text{match}(\mathcal{P}, \mathcal{Q}) = \{(p, q) : p \in \mathcal{P}, q \in \mathcal{Q}\}$ and $\mathcal{W} = \text{outlier}(\mathcal{M}) = \{w(p, q) : \forall (p, q) \in \mathcal{M}\}$, the error function is given by:

$$ e(\mathcal{P}, \mathcal{Q}) = \sum_{(p,q)\in\mathcal{M}} w(p, q)d(p, q) \tag{2} $$

where $d(p, q)$ is a *distance function* between two points.

In practice the associations from the matching function are not perfect, and the best estimate for ${}^{A}\mathcal{T}_{B}$ cannot be found perfectly by (1). Nevertheless, the idea behind ICP is that even with imperfect matches, minimizing the error can provide an estimation that, in turn, provides better matches, and so on. This leads to a iterative algorithm where each iteration provides a intermediary transformation ${}^{i+1}\mathcal{T}_{i}$ from:

$$ {}^{i+1}\mathcal{T}_{i} = \arg\min_{\mathcal{T}}(e(\mathcal{T}({}^{i}\mathcal{P}'), {}^{B}\mathcal{Q}')) \tag{3} $$

where ${}^{i}\mathcal{P}'$ is the filtered cloud that is iteratively transformed by the intermediary transformations. The final estimate is given by the iterative composition of all intermediary transformations and the initial transformation between the shapes, that is supplied to ICP at the beginning of the registration process. Note that the initial transformation is important, since initial transformations that does not provide enough good matches may cause ICP's minimization procedure to be trapped in a local minima.

B. Configuration of ICP

There is a wide spectrum of ICP solutions that can be implemented by different combinations of filters and match, outlier, and distance functions. The good choices for an ICP solution are strongly dependent on the environment, the computing resources, and the sensing capabilities of the robot: assessing them requires a significant amount of expertise. This section presents the details of the ICP solution used in this work for mobile robots with the Velodyne HDL 64 sensor in mostly urban areas, along with the rationale behind them.

1) Filters: The filters handle the transformation of input scans $\mathcal{S}$ into filtered scans $\mathcal{S}'$. The scans acquired from the Velodyne sensor contain a very large number of points, of which a large part is redundant data. To reduce the computational time of ICP, a first filter is applied to subsample the point cloud by keeping one of every N points in the scan. This systematic sub-sampling can be used because the scan data is somehow organized (points are stored in rows – if not, some randomness can be used to subsample the points). With Velodyne HDL 64 data, up to 80% to 90% of points can be removed without loosing the environment structure.

A second filter computes the normals for each point using *Principal Component Analysis* (PCA) with the K nearest-neighbors of each point. The good number of nearest neighbors depends on the resultant structure of the cloud after applying the first filter. Normals are important scan features for our ICP solution: they are used in the outlier rejection and distance function, as explained in paragraphs III-B.3 and III-B.4.

The last two filters adjust the direction of the computed normals: one computes the observation vector that links the sensor origin to each point in the scan, and the second one orients the normal vector of each point so that the angle between the observation vector and the normals are minimal.

2) Match Function: The matching function $\mathcal{M} = \text{match}(\mathcal{P}, \mathcal{Q})$ pairs each point of the reading shape $\mathcal{P}$ with the 3 nearest neighbor points in the reference cloud $\mathcal{Q}$. The decision to match the same point more than once adds robustness to the ICP, and 3 matches has shown to be a good compromise.

3) Outlier Rejection: The rejection function $\mathcal{W} = \text{outlier}(\mathcal{M})$ uses a *hard outlier rejection*, meaning that the weights $w(p, q)$ for a match are either zero or one. We consider that even if noisy, the initial transformation is close enough to the correct transformation. Thus matches whose euclidean distance are greater than a threshold are rejected (i.e. given zero weight). Matches whose point normals present angles greater than a threshold are also rejected. This makes ICP to consider only matches between points that belong to similar planes, and avoid matches made in cluttered areas (e.g. vegetation), that impede a good convergence.

4) Distance Function: The distance function $d(p, q)$ is the distance from a point p in the reading cloud to the plane defined by the normal of the matched point q in the reference cloud. This is known as *point-to-plane* distance. This distance function is best suited for the environments that partly exhibit planar structures, and also handles well multiple matches for the same reading point since the error is minimal when all the matched points belong to the same plane.

5) Convergence Tests: As an iterative algorithm, ICP needs some criteria to assess the minimization convergence. The main convergence test is the differential test, where we consider the ICP converged if the translational and rotational differences between transformation increments drops below given thresholds. This computed difference is normally smoothed through a few iterations for increased robustness. Also, we consider that the convergence failed if we reach a maximum number of allowed iterations or if the relative transformation between the initial and last computed transformation exceeds a maximum rotational and translational threshold. For the case where ICP is used in loop closing events, we also consider a failure if the final error is above a given threshold (this error is the sum of the distances $d(p, q)$ between matched points after convergence).

TABLE I: ICP Configuration for Velodyne HDL-64

ICP elements	Details
Filters	Keep one point in every 20 points Compute normals using 10 nearest neighbors Add observation vector pointing to sensor origin Orient normals using the observation vectors
Matching	Match to 3-nearest neighbors
Rejection	Remove pairs if distance is greater than 1m Remove pairs if normal angle is greater than 60°
Distance	Use distance from the point to plane
Convergence	Success if relative motions goes below 0.01m and 0.001rad. Failure if reaches 80 iterations Failure if transform goes beyond 0.8rad and 15m Failure if final error is above $70\,m$ (loop closing only)

The main elements of our ICP solution are summarized in Table I.

IV. Keyframes and Local Maps

An essential prerequisite for the registration is to have enough overlap of the observed environment in both reading and reference scans. Otherwise not enough good matches are found, compromising the quality of the resulting transformation.

An issue with Velodyne LIDARs (as with most LIDARs actually) is that the produced point clouds have a spatial resolution that drastically reduces with distance: the overlapping area between two scans acquired at different positions is sampled with very different spatial resolutions. As a result, the precision of the computed normal between the reading and reference scans is different, and matches can be established on points that do not correspond to the same scene element.

In our approach, reference clouds used by ICP are *local maps*, that are the concatenation of n selected keyframe scans. The points of these scans are expressed in the keyframe of the local map that is the closest to the reading scan frame. The reading frame is usually the robot frame, but it can also be another keyframe when attempting to close a loop.

The overlap between the new scans and the current local map decreases as the robot moves away from the local map. In order to always have enough overlap, the current overlap is estimated after every ICP call, and a new local map is composed if this estimate drops below a threshold. The overlap estimation used is simply the ratio between the number of matched points in the filtered reading scan after

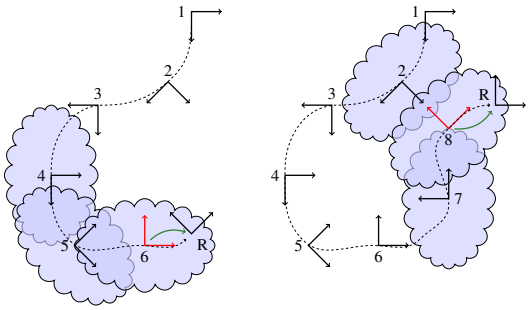


Fig. 2: Two possibilities to build local maps: exploration (left); and revisiting (right). The robot trajectory is represented by the dashed line, clouds represent keyframe scans. Keyframes are identified by numbers, and R represents the robot frame. Local maps are here composed of 3 scans, and are expressed in the frames shown in red. Green arrows represents the transformation from the local map to robot frame. For clarity reading scans are not shown.

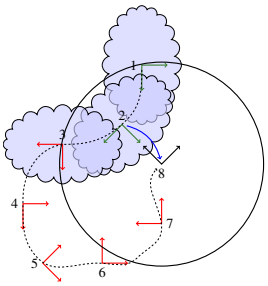


Fig. 3: Loop closing candidate selection and local map. The circle shows the maximum distance from keyframe 8 to allow the loop closing. Keyframes 3 and 7 are not possible candidates because they are inside the no loop window (shown in red). Possible candidates are keyframes 1 and 2, in green. The loop closing candidate is keyframe 2 since it is the closest between the possible candidates. A local map is build around it following the robot path when it first visited the region.

outlier rejection and the total number of points in the filtered reading scan.

To control the local map building we use the a concept of *state*: the robot may be *exploring* if it is moving towards a unknown part of the environment; or it may be *revisiting* if it is moving in a region that was already mapped.

In its nominal operation the robot is exploring. If the overlap drops below the threshold when exploring, the system attempts to find a set of keyframes *in sequence along the robot path* that would provide a good overlap. If a set is found this means the robot is going back through the path it was exploring, so the state changes to revisiting. If no set is found, a new keyframe is created and initialized to the current robot frame, with the last reading scan set as its associated scan. Then a loop closing may be performed using this new keyframe (more on this in Section V), and the state is changed to revisiting if the loop closing succeeds.

When the robot is in revisiting state, attempts to compose new local maps are also performed when the overlap drops below the threshold, but *any set of keyframes may be used to build the local map*, in opposition to the constraint in the exploring state. If no new set is found it means the robot started to explore the environment again, therefore a new keyframe is created as explained before and the state changes back to exploring. Fig. 2 shows the principle of local map building for both system states, with local map size $n = 3$.

Note that no new keyframes will be added when the robot moves in already mapped regions, this allows the number of keyframes to scale with the size of the environment being explored, instead of the length of the robot path, as done in FrameSLAM [9]. Note also that when the robot starts again to explore the environment from a region it was revisiting, new keyframes being created during this exploration will gradually replace the keyframes that are more distant to the robot in the local map of the last revisited region.

The keyframes are important in the system because it is through them that the information flows between the ICP localization level and the pose graph level. Every time a new keyframe is created a pose variable is added to the pose graph, and the last computed transformation along with the estimation of its covariance is added as a constraint (factor) in the pose graph. The keyframes are always positioned at the poses represented by their associated pose variables in the pose graph. Updates on these variables happens when loop closings are performed, as explained in the following section.

V. Loop Closing

Every time a new keyframe is created the system verifies if a loop can be closed. For this, the distances between the newest keyframe and the other keyframes are computed, and the candidate selection process illustrated in figure Fig. 3 is run. We assume that the robot is a ground robot and that the environment being explored does not contain tunnels or bridges, thus if the robot returns to a previously visited place along its path it should be at the same elevation as before. The keyframe distance for loop closing verification is then performed only in the 2D plane defined by the x and y axes of the first keyframe.

Note that we do not consider the m most recent keyframes as possible candidates for the loop closing. This is needed to avoid creating small loops in the graph, favorizing the creation of longer loops that provides a better correction of odometry drift. This set of m keyframes is called *no-loop window*, and it should be greater than the local map size.

The closest keyframe between the ones satisfying the previous conditions is defined as the *loop closing candidate*. A local map around the candidate keyframe is built, and a loop closing ICP is performed. Besides the normal ICP convergence tests described on Section III-B.5, the final value for the error function is also analyzed, and the loop closing is considered successful if it also lies below a threshold. This is important because sometimes the initial transform is not precise enough to ensure the overlap between the scans of the loop closing ICP. This additional test helps to avoid closing the loop with an incorrect transformation.

A successful loop closing ICP then adds a loop closing constraint to the factor graph between the variables associated to the newest keyframe and the candidate keyframe. The optimization using the graph data can be performed,

which updates the variables in a way that the overall robot drift along the loop is reduced. The optimization is reflected back to the ICP level by repositioning the keyframes with the poses in the associated graph values.

VI. Pose-Graph Building

The second main feature of the proposed system is the use of a pose graph to reduce the accumulated errors from ICP odometry when a loop closing is detected. A pose graph is a special case of a factor graph when all the variables being estimated are robot poses along the robot trajectory, and the factors are relative pose measurements between these poses.

Formally, a *factor graph* [10] is a bipartite graph $\mathcal{G} = (\mathcal{F}, \mathcal{X}, \mathcal{E})$ with factor nodes $f_i \in \mathcal{F}$ and variable nodes $x_j \in \mathcal{X}$. Edges $e_{ij} \in \mathcal{E}$ always connect factor nodes and variable nodes. The factor graph defines the factorization of a function $g(\mathcal{X})$ as:

$$g(\mathcal{X}) = \prod_i f_i(\mathcal{X}_i) \tag{4}$$

where $\mathcal{X}_i$ is the set of variables x_j adjacent to f_i (i.e variables connected to f_i through an edge e_{ij}). Thus each factor f_i is a function of the variables $\mathcal{X}_i$. The set $\mathcal{E}$ can be implicitly defined by $\mathcal{X}_i$ and is generally omitted. An example of factor graph can be seen in the Graph Layer of Fig. 1, with factors represented as small squares, and variables as circles.

Each factor f_i encodes a measurement function $h_i(\mathcal{X}_i)$ and a measurement z_i. When assuming Gaussian measurement models, f_i is defined as:

$$f_i(\mathcal{X}_i) \propto \exp\left(\frac{1}{2}\|h_i(\mathcal{X}_i) - z_i\|^2_\Sigma\right) \tag{5}$$

with $\|e\|^2_\Sigma = e^T\Sigma^{-1}e$ being the Mahalanobis distance with covariance matrix Σ. Then, finding the configuration for the variable nodes $\mathcal{X}^*$ that maximizes (4) comes to solve the nonlinear least-squares problem:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \frac{1}{2}\sum_i \|h_i(\mathcal{X}_i) - z_i\|^2_\Sigma \tag{6}$$

In our proposed approach, each variable in the pose graph represents a 3D pose and is associated with a keyframe of the ICP layer, and each factor is a constraint between two variables. Each factor encodes the following measurement function:

$$h(x_a, x_b) = \hat{z}_{ab} = (\ominus x_a) \oplus x_b \tag{7}$$

where $\ominus$ and $\oplus$ are respectively the inverse and composition operators, a and b the indexes of keyframes, and x_b the pose obtained by composing x_a with the transformation $\hat{z}_{ab}$. The transformation measurement z_{ab} to be used together with $h(x_a, x_b)$ in (6) is set to be an appropriate transformation obtained from the ICP layer: it is the one computed and used to re-localize the robot then the keyframe K_b was created during exploration; or it is the one obtained from loop closing ICP if the factor produces a loop in the graph.

Each factor should also be associated to a covariance matrix Σ to be used in $\|e\|^2_\Sigma$. Since the ICP only produces transformations as outputs, the covariance matrix Σ_{ab} for z_{ab} estimated using the technique proposed on [11].

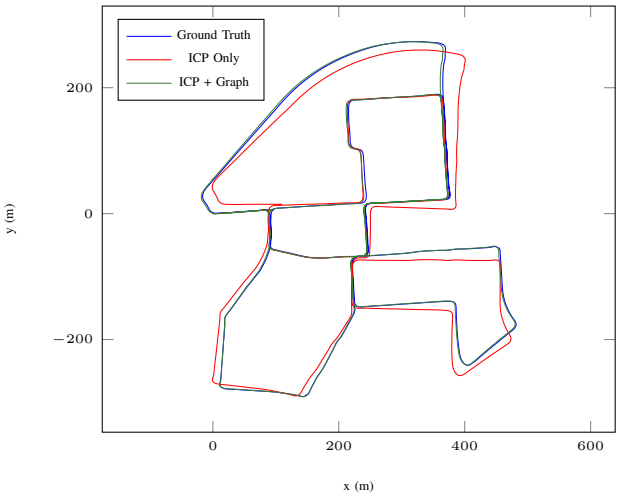


Fig. 4: Plot of final trajectories for the KITTI dataset. The visible deformations when using ICP only is caused mostly by accumulated angular errors, that makes the estimation drift along the z-axis. A much smaller drift occur when using the ICP together with the pose graph, being most visible in the regions far from the origin (right side of the plot).

TABLE II: Thresholds Used In The System

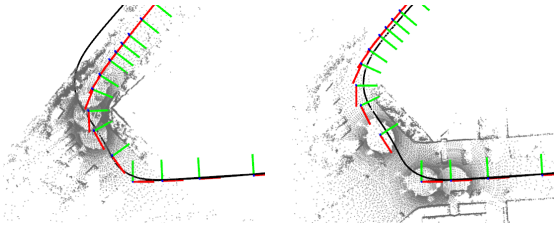
Dataset	KITTI
Local map size	3
No-loop window size	10
Overlap to create Kf	0.75
Max dist. to loop closing	15m
Max error for loop closing	5000

VII. Results

The proposed system was implemented using *libpointmatcher* [12] for the scan registration in the ICP layer, and *GTSAM* [13] for the optimizations in the graph layer. Experiments were performed using the KITTI dataset [14]. Noisy odometry measurements were drawn from the ground truth in order to provide a realistic initial transformation to ICP when the robot explores the environment. It was used a Gaussian noise of $\mathcal{N}(\mu, \sigma) = \mathcal{N}(0, 0.1\,\text{m})$ for each translation component and $\mathcal{N}(0, 1\,\text{deg})$ for each orientation component. Also, the scans are converted from the sensor frame to the robot frame after filtering but before the calls to ICP, in a way that the resulting transformations represent robot frame displacements, instead of sensor frame displacements. The 2D plots of the estimated trajectory can be seen in Fig. 4, and the thresholds used for each dataset are shown in Table II.

For the long run of KITTI dataset, initial orientation error is the main reason for the plot differences when the robot moves in regions far from the origin. This is mainly due to the fact that at the beginning we only have one keyframe (the initial one) that can be used to create a local map. Nevertheless, the robot was able to close a loop with the origin near the end of its trajectory.The state of the system before and after closing this loop can be seen in figure Fig. 5.

During the experiments, the covariances estimates used



(a) before (b) after

Fig. 5: Top-view before and after closing the loop at the origin of the KITTI dataset. The black line shows the ground truth, and the current local map is shown in gray. Compare the correction in the keyframe positions, and the local maps used before and after the loop closure.

to create factors in the graph were found to be remarkably small. Although the expected results were achieved, small covariances may lead to over-confident systems, or even to pose graphs whose optimization matrix are ill-posed and cannot be solved.

VIII. Discussion

This paper presented a system that uses transformations computed from ICP to feed a pose graph structure, that in turn is used on loop closings to build an optimization problem that provide updates of *keyframes* selected along the robot trajectory. These updates correct the map of the environment being built and reduce the accumulated errors from the ICP odometry. The paper also detailed the ICP configuration for Velodyne HDL-64 sensor.

The main difference between this work and [6] is the way the optimization problem is constructed: in [6] the optimization finds the robot poses that minimizes the positional error of points matched between two neighbor scans in the graph, resulting in very large linearized system to be solved at every iteration, and requiring a large amount of time to converge to the solution. In this work the transformations between keyframes outputed from ICP are used as measurements between keyframes in the graph, and the covariances are estimated as proposed by [11]. This leads to a system that is faster to solve, and may benefit from the new techniques based on factor graphs, like ISAM2.

One interesting feature of [6] is that edges may be added to or removed from the graph in between the optimization iterations, while in this work the graph is static. Other differences are: our use of an overlap criteria to create new keyframes, in contrast to a fixed distance criteria; and the restriction to create new keyframes only when exploring new parts of the environment, that was not addressed in [6].

The overlap threshold used to define new keyframes is a critical parameter of the system. This is due to the way the overlap is estimated and the influence that the filters have in the estimation: hard rejection based on normal angles helps to avoid unexpected drifts caused by vegetation, but also makes the overlap estimation go fast below the threshold. This results in keyframes that are close one to each other, and consequently in local maps that cover smaller areas of the environment. A more robust way to compute the overlap between shapes would benefit the proposed system.

Even if the proposed system improves the estimation of the robot trajectory through loop closings, it remains prone to local minima in which ICP can converge: the transformations computed in such cases are propagated to the graph level and cause discrepancies in the final map. Local minima occur more often when the environment is mostly symmetric, or if it has repetitive patterns. They can be avoided with good initial estimations for the registration, for example exploiting IMU-based initial estimations. IMU data can also be used with pre-integrated IMU factors [15] that can be added to the graph as constraints between keyframe variables instead of ICP- based ones. In this case, more variables would need to be added per keyframe (namely the velocity, and the accelerometer and gyrometer biases) and the graph would not be a pose graph anymore. Still the ICP transforms could be used to create small loop closings between keyframe poses along the path, helping to constrain these extra variables and allowing successful optimizations in the graph level.

Acknowledgments

The first author would like to thank the Brazilian council *CNPq* and the *Ciência Sem Fronteiras* program for the financial support.

References

[1] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems Conference*, July 2014.

[2] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart, "Tracking a depth camera: Parameter exploration for fast ICP," in *IEEE/RSJ IROS*, Sept 2011.

[3] P. J. Besl and H. D. McKay, "A method for registration of 3-D shapes," *IEEE PAMI*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[4] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A Tutorial on Graph-Based SLAM," *Intelligent Transportation Systems Magazine, IEEE*, vol. 2, no. 4, pp. 31–43, winter 2010.

[5] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *IJRR*, vol. 31, no. 2, pp. 216–235, 2012.

[6] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg, "Globally consistent 3d mapping with scan matching," *Robot. Auton. Syst.*, vol. 56, no. 2, pp. 130–142, Feb. 2008.

[7] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[8] F. Pomerleau, F. Colas, and R. Siegwart, "A Review of Point Cloud Registration Algorithms for Mobile Robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, 2013.

[9] K. Konolige and M. Agrawal, "FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping," *Trans. Rob.*, vol. 24, no. 5, pp. 1066–1077, Oct. 2008.

[10] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb 2001.

[11] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *IEEE ICRA*, April 2007, pp. 3167–3172.

[12] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP Variants on Real-World Data Sets," *Autonomous Robots*, vol. 34, no. 3, Feb. 2013.

[13] F. Dellaert, "Factor Graphs and GTSAM: A Hands-on Introduction," GT RIM, Tech. Rep. GT-RIM-CP&R-2012-002, Sept 2012.

[14] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," *IJRR*, 2013.

[15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015.