Efficient Long-term Mapping in Dynamic Environments

María T. Lázaro, Roberto Capobianco and Giorgio Grisetti

***Abstract*— As autonomous robots are increasingly being introduced in real-world environments operating for long periods of time, the difficulties of long-term mapping are attracting the attention of the robotics research community. This paper proposes a full SLAM system capable of handling the dynamics of the environment across a single or multiple mapping sessions.**

Using the pose graph SLAM paradigm, the system works on local maps in the form of 2D point cloud data which are updated over time to store the most up-to-date state of the environment. The core of our system is an efficient ICP-based alignment and merging procedure working on the clouds that copes with non-static entities of the environment. Furthermore, the system retains the graph complexity by removing out-dated nodes upon robust inter- and intra-session loop closure detections while graph coherency is preserved by using condensed measurements. Experiments conducted with real data from long-term SLAM datasets demonstrate the efficiency, accuracy and effectiveness of our system in the management of the mapping problem during long-term robot operation.

I. Introduction

In the last years, robots have been progressively introduced in public environments where space is shared with humans, such as shopping centers [1], airports [2], elder care homes [3], hospitals or museums. In order to navigate autonomously these environments, robots require a map of the building in which they have to operate. This map is usually obtained before the deployment of the robots by using Simultaneous Localization and Mapping (SLAM) techniques, for which reliable implementations are publicly available [4], [5], [6].

Typically, mapping public environments presents several challenges. First, despite the risk of introducing artifacts in the final map, data acquisition can rarely be executed in the absence of people. This constitutes a *highly dynamic* situation that makes the environment non-stationary, and must be handled by the SLAM system within the active mapping session. Second, even when a proper data acquisition is realized (e.g., with the building closed to the public), the obtained map might not be usable after few days, due to substantial changes in the environment (e.g., moved stands, new decorations for special events, etc). Nevertheless, after a reconfiguration, the map might be still valid for some time. Hence, we refer to this sort of environments – affected by *low dynamics* – as semi-static. The changes in semi-static environments might affect the topology of the environment and substantially influence its local appearance, thus invalidating portions of the previously acquired map that have to be estimated from scratch.

Although dealing with dynamic and semi-static environments in a life-long mapping process is one of the goals of mobile robotics, typical solutions are impractical since they require to acquire a completely new map of the whole environment whenever a semi-static change occurs. Conversely, it would be sufficient to update only the portion of the map that changed.

Most of the latest successful solutions to the SLAM problem are based on a pose graph representation whose nodes store discretized positions along the robot trajectory and edges encode relations between nodes. Using this representation, the SLAM problem simplifies to the resolution of an optimization problem whose complexity depends directly on the number of nodes present in the graph.

When considering this problem from a life-long perspective, a robot navigating continuously the environment will produce a limitless addition of nodes, making the optimization problem intractable. Directly related with the number of nodes is also the problem of loop closure detection. The unbounded addition of nodes will increase the dimension of the search for possible candidates and, with it, the possible inclusion of false positives. Furthermore, inconsistencies between the current state of the environment and what is present in the map due to the changing environment can lead to mislocalization or navigation errors.

Therefore, there are two main aspects to be considered in order to ensure the correct operation of a SLAM system in a long-term horizon: the maintenance of a constrained dimension of its inherent optimization problem and an efficient management of the changes that happen over time.

This paper presents a full SLAM system focused on the management of long-term situations capable of handling both *highly dynamic* situations within a mapping session and map updates within multiple sessions to cope with *low dynamics*. Our system estimates a map as a pose graph whose nodes contain local maps which allows to maintain a sparser graph representation. Local maps store the appearance information in the form of 2D point clouds while spatial relations between adjacent local maps are encoded as a graph edges. The use of point clouds offers higher definition and are less memory demanding than other data structures such as grids which can only represent the environment up to a certain resolution to maintain performance constraints. The core of our system is the use of an effective ICP-based point cloud alignment and merging procedure that allows to update the content of the local maps to preserve the most up-to-date information about the environment. Graph complexity

María T. Lázaro, Roberto Capobianco and Giorgio Grisetti are with Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti", Sapienza University of Rome, Italy. {mtlazaro, capobianco, grisetti}@diag.uniroma1.it

is retained bounded by removing nodes containing out-dated information, while the graph *coherency* is preserved by compressing prior knowledge through *condensed measurements* [7]. Map information obtained over different sessions is managed by the same graph structure obtaining a balance between simplicity and effectiveness in determining both intra- and inter-session loop closures.

We release our system as open source[1] which provides ROS compatibility and visualization tools.

II. Related Work

The problem of the dimensionality in SLAM is well known and a variety of techniques have been proposed over the years to overcome such issue. Solutions range from those studying the structure of the optimization problem [8], [9] to those based on graph reduction methods [10], [11].

In [12], a framework for multi-session map optimization is proposed, where several independently optimized pose-graphs acquired at different sessions are combined by using anchor nodes. While graph reduction methods are normally batch algorithms executed off-line (i.e., using as input an already constructed graph), their on-line application is not straightforward. The SLAM system proposed in [13] achieves long-term operation by using a reduced graph which retains bounded complexity over time proportional to the explored environment and not the total distance traveled. New nodes are added only if there are not previous nodes nearby, while new information from loop closures is incorporated as new edges between existing poses. However, the system does not update the sensor data contained in the nodes to consider the dynamic changes of the environment.

Regarding the works considering the non-static nature of the environment, the tendency is to maintain parallel maps, one for short-term periods characterized by high dynamics and another one for long-term periods capturing more stationary structures. Early works [14], [15] used a grid-based map representation to that end while Biber and Duckett [16] extended this perspective by maintaining not just two short- and long-term maps but by tracking and updating the state of the environment at multiple timescales. Also based on grid-based mapping [4], is the recent work by Krajnik et al. [17]. Their system builds a new map for each run, while consistency between sessions is achieved by using localization with respect the previous model as odometry measurements for the current session. Maps are integrated into a spatio-temporal model capturing the persistency and periodicity of the environment that allows to predict the future state of the environment with proven usefulness for navigation and path planning purposes.

In a more recent graph-based approach [18], Labbé and Michaud organize graph nodes in Short-term, Long-term and Working memories. Nodes are moved from one type of memory to another as they are required as in a cache mechanism. Also based on graph-based representation, Walcott-Bryant et al. [19] proposed a "Dynamic Pose Graph (DPG)" focused on managing low dynamics. Their work focuses on modeling the sensor data, where laser points are labeled as static, added or removed which allows to keep an active (current state) and dynamic maps, containing added or removed points. Changes are detected by projecting current and previous scans into grid maps and comparing their coverage. They also detect inactive nodes that are removed from the DPG to maintain a bounded graph complexity.

[1] https://gitlab.com/srrg-software/srrg_mapper2d

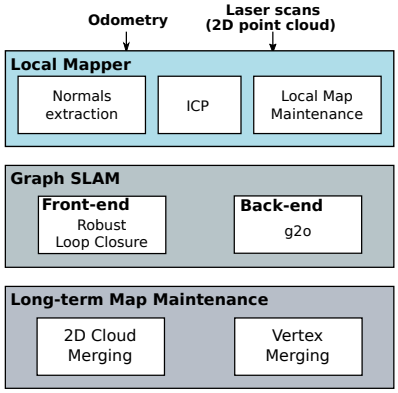


Fig. 1: System overview

III. System Overview

In this section, we describe our SLAM system together with the specific methods proposed to handle with long-term mapping in dynamic environments.

Figure 1 illustrates our system architecture. In essence, our system relies on the pose graph paradigm, but we enhance the *front-end* aspect of a standard graph SLAM system by allowing previous nodes to be modified and by dynamically re-estimating the graph topology to include the most up-to-date information while preventing the problem size to grow without limit. Nodes in the graph contain local maps in the form of 2D point clouds extended with structure and robot trajectory information, while edges in the graph encode spatial relations between the local maps.

The system is organized in three components as depicted in Fig. 1. The first component is a pose tracker and a local mapper that reconstructs the local area of the robot by registering and combining the acquired laser scans, seen as 2D point clouds. The registration process relies on a fast and accurate cloud alignment algorithm based on NICP [20] that exploits the structure of the environment by introducing surface normals in the minimization problem. Once two clouds are aligned, we perform a cloud merging procedure manage highly dynamic entities of the environment, such as traces left by people moving in the surroundings. This results in cleaner local maps with reduced spurious information.

The second component contains the main pipeline of our SLAM system, which addresses the graph construction (*front-end*) and on-line graph optimization (*back-end*) aspects of a standard graph SLAM system.

The third component executes the different mechanisms proposed for long-term map maintenance and update with the

most current information about the environment considering, therefore, dynamics between multiple mapping sessions.

A. Local Mapper

In this work, we use a variant of the NICP algorithm introduced by Serafin et al. [20] to perform 2D point cloud registration. To this end, we define a 2D point cloud $\mathcal{P} = \{\bar{\mathbf{p}}_{1:N}\}$ as a set of extended points $\bar{\mathbf{p}}$ composed of a point $\mathbf{p} = (p_x\, p_y)^T$ and its normal $\mathbf{n} = (n_x\, n_y)^T$. In computing such normal, we consider the neighborhood $\mathcal{D}_i$ of each point $\mathbf{p}_i$ and compute the covariance matrix of the Gaussian distribution $\mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$ of the points lying in $\mathcal{D}_i$, and set $\mathbf{n}_i$ to be the eigenvector corresponding to the smallest eigenvalue of the covariance matrix $\mathbf{\Sigma}_i$.

Given an extended point $\bar{\mathbf{p}}$ we can apply the composition operator $\oplus$ to transform the extended point by using a transformation matrix $\mathbf{T}$ composed of a rotation $\mathbf{R}$ and a translation vector $\mathbf{t}$ as

$$\bar{\mathbf{p}} = (\mathbf{p}, \mathbf{n})^T \qquad \mathbf{T} \oplus \bar{\mathbf{p}} = (\mathbf{R}\mathbf{p} + \mathbf{t}, \mathbf{R}\mathbf{n}) \tag{1}$$

Furthermore, we define a projection function $\mathbf{s} = \pi(\mathbf{p})$ that maps a Cartesian point $\mathbf{p}$ to range and bearing space $\mathbf{s} = (r, \alpha)$, i.e., the laser measurement space. Accordingly, we can define a back-projection function $\mathbf{p} = \pi^{-1}(\mathbf{s})$ that enables us to retrieve the two-dimensional Cartesian point $\mathbf{p}$ from $\mathbf{s}$. By using this function, we can represent a laser scan as a cloud of extended points obtained by back-projecting the set of measurements $\{\mathbf{s}_{1:N}\}$ to a set of points $\{\mathbf{p}_{1:N}\}$ in the Cartesian space and by computing, for each $\mathbf{p}_i$, the corresponding point normal $\mathbf{n}_i$.

1) Cloud registration: Given two extended point clouds $\mathcal{P}^{\mathbf{r}} = \{\bar{\mathbf{p}}^{\mathbf{r}}_{1:N_\mathbf{r}}\}$ and $\mathcal{P}^{\mathbf{c}} = \{\bar{\mathbf{p}}^{\mathbf{c}}_{1:N_\mathbf{c}}\}$, we use a set $\mathcal{C} = \{(i,j)_{1:M}\}$ of correspondences between extended points $\bar{\mathbf{p}}^{\mathbf{r}}_i$ and $\bar{\mathbf{p}}^{\mathbf{c}}_j$, to compute the transformation matrix $\mathbf{T}^*$ that minimizes the distance between corresponding points, expressed as the following least squares objective function:

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_{\mathcal{C}} (\mathbf{T} \oplus \bar{\mathbf{p}}^{\mathbf{c}}_i - \bar{\mathbf{p}}^{\mathbf{r}}_j)^T \bar{\mathbf{\Omega}}_{i,j} (\mathbf{T} \oplus \bar{\mathbf{p}}^{\mathbf{c}}_i - \bar{\mathbf{p}}^{\mathbf{r}}_j). \tag{2}$$

where, $\bar{\mathbf{\Omega}}_{i,j} = diag(\mathbf{\Omega}^p_{i,j}, \mathbf{\Omega}^n_{i,j})$ is an information matrix that takes into account the noise properties of the sensor and summarizes the contributions of the information $\mathbf{\Omega}^p_{i,j}$ and $\mathbf{\Omega}^n_{i,j}$, respectively coming from the point and normal correspondences. Finally, in order to obtain the transformation $\mathbf{T}^*$, we solve Eq. 2 iteratively by using a damped version of the Gauss-Newton algorithm. During optimization we use the robot odometry $\mathbf{T}_o$ both as initial guess and as prior. In this way we restrict the solution returned by the solver to lie in the confidence ellipsoid of the odometry estimate.

To find the set of correspondences $\mathcal{C}$, the relative transform $\mathbf{T}$ is used to observe $\mathcal{P}^{\mathbf{c}}$ from the origin of the reference cloud $\mathcal{P}^{\mathbf{r}}$. This step enables the use of the projection function on the two clouds to obtain measurements in laser space from the same point of view (Fig. 2. By discretizing the angular values of these measurements in K bins, the extended points $\bar{\mathbf{p}}^{\mathbf{r}}_k$ and $\bar{\mathbf{p}}^{\mathbf{c}}_k$ – whose components $\mathbf{p}^{\mathbf{r}}_k$ and $\mathbf{p}^{\mathbf{c}}_k$ are back-projected from the kth bin – are considered as candidate correspondences $(i,j)_k$. Such candidates are discarded if the Euclidean distance between the 2D points is greater than a threshold ϵ_d or if the angle between the normals is greater than a threshold ϵ_n, as shown in Eq. 3.

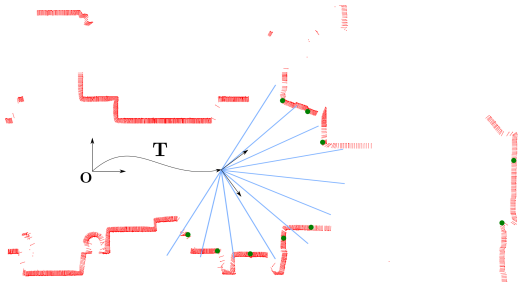


Fig. 2: Projection of cloud points into range and bearing measurements. By transforming the cloud points using $\mathbf{T}$ we can generate virtual laser measurements seen from that position. In the figure, cloud points with normals are depicted in red, blue lines represent the discretized bearing bins and green dots the selected points in the projection.

$$\mathbf{T} \oplus \mathbf{p}^{\mathbf{c}}_k - \mathbf{p}^{\mathbf{r}}_k > \epsilon_d \qquad (\mathbf{T} \oplus \mathbf{n}^{\mathbf{c}}_k) \cdot \mathbf{n}^{\mathbf{r}}_k < \epsilon_n. \tag{3}$$

The search for correspondences is repeated on each iteration of the optimization process using the current solution for $\mathbf{T}$.

2) Local Map Maintenance: The application of the described alignment procedure to subsequent sensor measurements allows to create locally consistent maps by accumulating the aligned data. However, its execution time grows with the number of points and, therefore, we need to limit the number of points as new scans are added to the cloud to preserve real time performance. At the same time, we want to reject out-dated information that would clutter the local map (e.g., moving people). In order to handle these issues we introduce a *cloud merging* process that limits the number of points in a cloud while denoising them and that handles dynamics by exploiting free space information.

During this phase, similarly to the correspondence finding process, we use the relative transform $\mathbf{T}$ between $\mathcal{P}^{\mathbf{r}}$ and $\mathcal{P}^{\mathbf{c}}$ to align the point of view of the two clouds and apply the projection function π. All the obtained measurements are then discretized and indexed according to their angular values, and for each equally indexed pair of measurements $\mathbf{s}^{\mathbf{r}}_i = \pi(\mathbf{p}^{\mathbf{r}}_i)$ and $\mathbf{s}^{\mathbf{c}}_i = \pi(\mathbf{p}^{\mathbf{c}}_i)$ the following operations – illustrated on Fig. 3 – are performed:

- if $r^c_i >> r^r_i$, the new beam crosses an existing element of the reference cloud, which is hence replaced with the corresponding transformed point in the current cloud;
- if $r^r_i >> r^c_i$, the new beam stops much ahead of the point in the reference cloud and it is added to the reference model, as it might be due to a new object entering in the scene;
- otherwise, $\bar{\mathbf{p}}^{\mathbf{c}}_i$ and $\bar{\mathbf{p}}^{\mathbf{r}}_i$ are merged into a new point $\bar{\mathbf{p}}$ as follows:

$$\bar{\mathbf{p}} = \frac{\bar{\mathbf{p}}^{\mathbf{c}}_i \sigma^c_i + \bar{\mathbf{p}}^{\mathbf{r}}_i \sigma^r_i}{\sigma^c_i + \sigma^r_i}, \qquad \sigma^c_i = \frac{1}{r^c_i}, \quad \sigma^r_i = \frac{1}{r^r_i} \tag{4}$$

The goal of this module is to produce local maps that will be used by graph SLAM as nodes. The origin within a

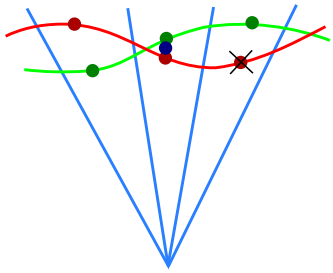


Fig. 3: Cloud points merging. Distances between points from the current cloud (green) and the reference cloud (red) that have been projected into the same angular bin are analyzed. On the left beam, a new point falls in front the old one ($r_i^r >> r_i^c$) and it is added into the cloud. On the right beam, a new point goes through the old one ($r_i^c >> r_i^r$), thus the new point replaces the old one. On the middle, points are close each other, then they are merged into one new point using Eq. 4.

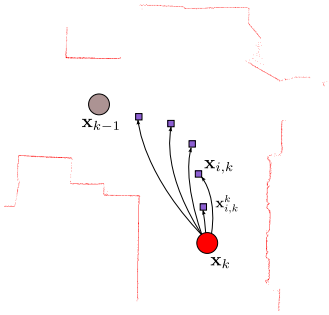


Fig. 4: 2D cloud with trajectory. Current node $\mathbf{x}_k$ and its associated cloud are depicted in red while purple squares are the waypoints representing the portion of the trajectory between nodes $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$. Relative transformations $\mathbf{x}_{i,k}^k$ are stored in the cloud.

local map is selected to be the current pose of the robot in the trajectory. Using local maps instead of raw scans to construct the graph results in much smaller graphs, at the cost of losing valuable information that is retained in the trajectory. This might ultimately lead to the inability to modify or "deform" a local map. As a trade-off we still preserve the information about the robot trajectory within the local maps. To this end, we store in the local maps a discretized set of trajectory's *waypoints* expressed with respect to the origin of the local map. These waypoints represent vantage points from where we can render scans to produce other data like, for example, occupancy grid maps. More in detail, we store the relative transformation $\mathbf{x}_{i,k}^k$ with respect to the current node position $\mathbf{x}_k$ for each waypoint $\mathbf{x}_{i,k}$, $i = \{1, ..., m\}$ belonging to the discretized portion of the trajectory between nodes $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$. This is illustrated on Fig. 4.

B. Graph SLAM

Our system uses as a basis a pose-graph for map representation. This graph is composed of nodes containing robot positions and edges encoding spatial relations between the nodes. More formally, let us denote as $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_n)^T$ the set of nodes representing the robot trajectory and $\mathbf{z}_{ij}$ as the measurement encoded in an edge connecting two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, affected by Gaussian noise with information matrix $\mathbf{\Omega}_{ij}$. Let also be $\mathbf{h}(\mathbf{x}_i, \mathbf{x}_j)$ the function that computes the relative relation between the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ given their current state. The estimation error $\mathbf{e}_{ij}$ given the measurement $\mathbf{z}_{ij}$ can be calculated as:

$$\mathbf{e}_{ij} = \mathbf{z}_{ij} - \mathbf{h}(\mathbf{x}_i, \mathbf{x}_j) \tag{5}$$

The solution of a graph-based SLAM approach minimizes the overall error to obtain a configuration of the nodes $\mathbf{x}^*$ that better explain the set of measurements:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i,j} \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij} \tag{6}$$

The above Eq. 6 represents the SLAM problem as a Least-Squares optimization problem that can be solved using iterative methods like Gauss-Newton or Levenberg-Marquardt. Our system uses the effective implementation of Levenberg-Marquardt algorithm offered by the $\mathbf{g^2o}$ [8] framework for graph optimization. Further insights on the resolution of the graph SLAM problem can be found on [21].

Each iteration of the above-mentioned methods implies solving a linear system whose complexity directly depends on the number of nodes in the graph. Our 2D cloud data structure allows us to efficiently address this issue by introducing new nodes into the graph only when a new local map is provided by the local mapper. This allows to sparsify the graph in the number of nodes and, therefore, reducing the dimension of the problem.

Our graph-SLAM pipeline continues as follows: each time a new node together with a 2D cloud is introduced into the graph we select a set of vertices lying within the Mahalanobis distance threshold of the current node. Then, we cluster these selected nodes into sets of connected nodes and choose the closest node of each set as a potential candidate in the search for loop closures. The same ICP-based algorithm explained in Section III-A is used to determine the alignment among the current cloud and those from the candidate nodes using the relative transformation between both nodes as initial guess. Each successful alignment can be translated into an edge which is introduced into a pool of candidate loop closure edges. Once a set of closure candidates have been obtained we use the voting scheme for robust map alignment presented in [22] to obtain the largest subset of inlier edges that minimizes the error introduced by the selected closures, which are finally added into the graph.

C. Long-Term Graph Maintenance

As introduced previously, the amount of data and the map structure to be stored and maintained in long-term mapping increases at each session. This makes necessary to implement maintenance procedures to retain the efficiency of the system and its consistency with the most up-to-date state of the environment.

In essence, our approach for graph maintenance consists in the fusion of local maps which have been determined to belong to the same area of the environment. Once the fusion has taken place, old nodes containing the out-dated local maps are removed. This causes a change on the graph

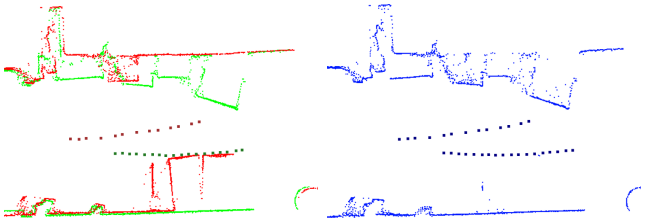


Fig. 5: Cloud merging. Left: Two clouds with trajectory information belonging to different sessions are being merged (red: oldest session, green: current session). Right: resulting cloud after merging. Red boxes on the bottom part have been removed since new observations confirm to "go through" them while new objects on the top are placed in front of old data. Trajectory information is also merged during this process.

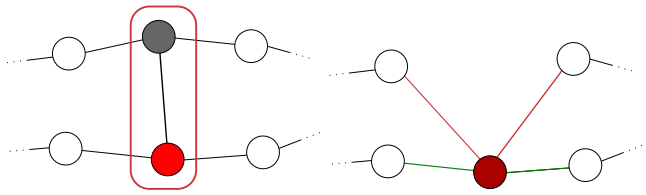


Fig. 6: Merging nodes in the graph. Left, the nodes involved in a closure are being merged, where the gray node is the oldest and the red one the newest. Right: when eliminating the oldest node, we create *condensed measurements* relating the merged node (used as a *gauge*) with the neighbors of the old one (in red) and the rest of nodes that were already connected (in green).

topology that must be handled to preserve the distribution over the poses before the node suppression.

Intuitively, we could state that two local maps share a portion of the environment if they form a loop closure. Therefore, our system attempts to merge local maps whenever a loop closure occurs. The goal is to obtain an updated and refined version of the part of the environment involved in the closure. Notice that loop closures relate nodes spatially close but distant in time, without distinguishing between different mapping sessions. Using reliable loop closures to trigger a local map merging procedure ensures the correct alignment of the resulting topology.

In summary, we must handle two issues: first, the 2D clouds must be merged and then, we must preserve a distribution over the poses at the optimum non-overconfident compared to the distribution over the poses before node suppression. At the same time, this distribution should include most of the information that is lost during the suppression process. We call this property *coherence* of the graph.

For the former, we use an extension of the cloud merging algorithm explained in Sec. III-A.2. The oldest and the newest clouds are identified given their timestamps. Then, using the oldest cloud as a reference, and taking into account the good initial alignment given by the closure, we perform a sequential refinement of the newest cloud onto the oldest. To this end, the waypoints of the newest cloud are used to generate virtual scans from such positions which are registered with respect to the oldest cloud to correct possible misalignments and finally merged to obtain a further refinement. The result of this procedure is illustrated in Fig. 5.

Once two local maps have been merged, the graph structure must also be updated to preserve its coherency. The exact procedure to apply when eliminating a node from a graph is to perform a full marginalization which creates a clique with all the neighbors of the suppressed node. Over time this would lessen the sparsity of the graph and negatively impact the performances of graph optimization, and therefore, it is not convenient for long-term mapping.

Instead, we use a robust approximation to marginalization based on *condensed measurements* [7] to maintain the graph coherency. The process is exemplified in Fig. 6. The oldest node belonging to the loop closure is removed from the graph while the remaining node is connected to the neighboring nodes by using a star-like topology. The remaining node is chosen as central node (referred as *the gauge* in [7]) which is connected to the neighbors by edges labeled with the condensed measurements which allow to retain the information that the gauge node has about the other nodes in the graph. We refer to [7] for details in the computation of these measurements.

IV. Experiments

In this section we present a quantitative and qualitative performance evaluation of our SLAM system. To this end, we use well-known multi-session and long-term SLAM datasets: the MIT Stata Center dataset[2] [23], which also includes ground-truth position estimates for accuracy analysis, and the Long-term indoor dataset[3] part of the LCAS-STRANDS long-term dataset collection [17].

A. High dynamics on long-term scenario

The first experiment is performed using the LCAS-STRANDS long-term dataset in which a robot was deployed in a care home for more than 100 days. This is a highly dynamic scenario for which we want to show the capability of our system to remove non-static obstacles during the local maps creation. Concretely, we use the data collected on Nov 21th, 2016, where the robot moved along 2km for 10 hours. We pre-processed the 10 hours-duration logs to skip laser scans when the robot was not moving in the environment. Our system takes around 4 minutes on a Intel(R) Core(TM) i7-6700 CPU@3.40GHz to process the resulting log. Figure 7 shows the map created for the care home facility. In this figure, the graph containing the local maps created using our local map maintenance mechanism for managing dynamic entities is confronted with respect to a version on which we project the original laser scan for each waypoint contained in the cloud. We can visually demonstrate the improvement in the map definition and removal of non-static obstacles.

B. Multi-session mapping with MIT Stata Center dataset

The second set of experiments are conducted over 5 different mapping sessions on the 2nd floor at MIT Stata Center, for which ground-truth about the robot trajectory is

[2]http://projects.csail.mit.edu/stata/index.php

[3]https://lcas.lincoln.ac.uk/owncloud/shared/datasets/index.html

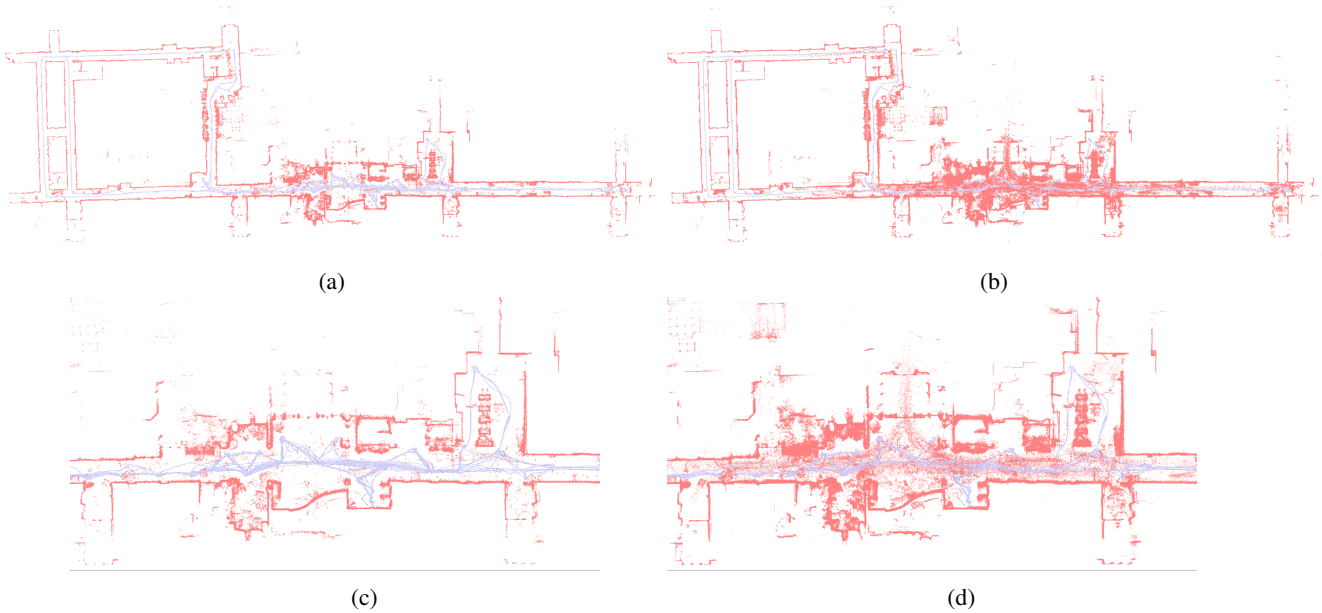


Fig. 7: Pruning highly dynamic obstacles with local map maintenance. a) Created graph and local maps in which removal of dynamic entities is considered. b) Original laser scans projected on the computed graph poses. c) and d) Zoom on areas with high transit of people.

TABLE I: MIT Stata Center dataset summary

Session #	Bagfile	Distance (m)	Time (min)
$s1$	2012-01-18-09-09-07	683	36
$s2$	2012-01-25-12-14-25	348	20
$s3$	2012-01-25-12-33-29	239	14
$s4$	2012-01-28-11-12-01	635	36
$s5$	2012-02-02-10-44-08	1003	52

TABLE II: Single session performance of our SLAM system

Session #	Nodes	Edges	Accuracy
$s1$	340	627	0.10634
$s2$	189	335	0.26230
$s3$	128	213	0.20490
$s4$	328	600	0.08326
$s5$	496	929	0.83637

provided. The concrete *bagfiles* – ROS-based logging file format – used in these experiments are summarized in Tab. I.

Over the different sessions, we measure the complexity of the generated graphs (i.e., number of nodes and edges), efficiency of our system in terms of graph optimization time and map accuracy based on the ground-truth. For accuracy evaluation we use the approach described in [24]: given the ground-truth poses associated to each laser scan we create a set of ground-truth edges relating nearby poses. The χ^2 error of these virtual edges can be computed when evaluated on the estimated poses. As a measure of accuracy we use the mean χ^2 error per edge.

Regarding the map construction, the same set of parameters are used along all the experiments reported in this section. The use of local maps allows to sparsify the nodes in the graph. Concretely, we add new nodes every 2.5m translation or 1rad rotation of the robot for these experiments.

Table II reports complexity and accuracy for the single-session runs of the reported datasets. During the execution, local map maintenance reported in Sec. III-A was applied to remove highly dynamic obstacles while the merging procedure reported in Sec. III-C for long-term graph maintenance was not enabled. This provides already a high quality result, as can be seen on Fig. 9 and on the numbers reported on the Table. We consider these results as a baseline for the ongoing comparison.

The second stage of this multi-session experiment consists of progressively accumulating data over sessions, i.e., for each mapping session we use as input the graph obtained in the preceding session. Initial localization of the robot with respect to the previous map is considered to be given by the user or assumed that the robot always starts a new mapping session from the same location (e.g., a docking station). Global localization of the robot is a well-studied problem by the SLAM community and it is not the focus of this paper.

We enable now the long-term map maintenance procedure that combines clouds and nodes in the graph when a loop closure takes place. Results obtained over the different sessions are reported on Tab. III and Fig. 10. We report accuracy of each mapping session (i.e., measuring error of the newly added nodes) given that we are using map information from previous sessions. Notice that with the graph maintenance procedure we already achieve a node reduction of the $\sim 50\%$ on session $s1$ while preserving accuracy (it actually improves slightly). Interestingly, the accuracy of the last session $s5$ – which was the last accurate on Tab. II – is greatly improved when using the previous map information. In general, we could confirm from these results that map accuracy is not traded-off with our long-term map management mechanisms.

Finally, we report on Fig. 8 the evolution of the graph optimization timings as new nodes are added in the graph. We observe that, even after accumulating different sessions,

TABLE III: Multiple session performance of our SLAM system

Session	Nodes	Edges	Accuracy
$s1$-m	164	319	0.09598
$s1 + s2$	216	462	0.33420
$s1 + s2 + s3$	258	556	0.19914
$s1 + s2 + s3 + s4$	295	753	0.08489
$s1 + s2 + s3 + s4 + s5$	430	1116	0.42881

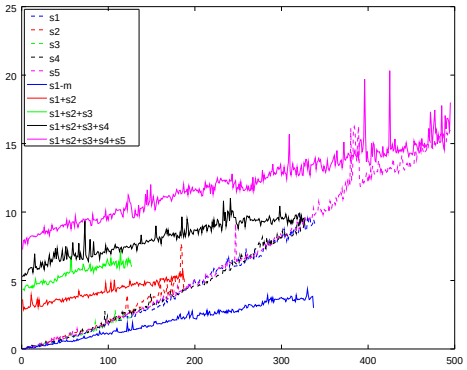


Fig. 8: Graph optimization times (ms) for each session execution. Times for the multi-session cases do not begin on zero since the mapping session starts from a previously created graph.

optimization times do not exceed those from single mapping sessions.

V. Conclusions

In this work we have presented our SLAM system whose design has been oriented towards the management of non-static entities of the environment and long-term mapping periods. Along the experiments we have shown the performance of our system which retains efficiency, accuracy and coherency between data structures over multiple mapping sessions.

Although long-term autonomy requires periodic mapping sessions to update robot knowledge about the environment, we think performing continuously SLAM on a fixed setting would be unnecessary. Future works will be oriented to the development of strategies based on active robot awareness of environmental changes that will allow to seamlessly switch from mapping to localization-only modalities if its current map suffices to obtain a good localization despite the dynamics.

Acknowledgments

The authors would like to thank Tom Krajnik and Jaime Pulido Fentanes for their help during the pre-processing of the STRANDS dataset used in this paper. We also thank the contributors of the MIT Stata Center dataset for their efforts on generating the referred multi-session SLAM dataset.

References

[1] L. Iocchi, M. Lázaro, L. Jeanpierre, A.-I. Mouaddib, E. Erdem, and H. Sahli, "COACHES: Cooperative autonomous robots in complex and human populated environments," in *Congress of the Italian Association for Artificial Intelligence*, 2015, pp. 465–477.

[2] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore *et al.*, "SPENCER: A socially aware service robot for passenger guidance and help in busy airports," in *Field and Service Robotics*, 2016, pp. 607–622.

[3] M. Hanheide, D. Hebesberger, and T. Krajnik, "The When, Where, and How: An Adaptive Robotic Info-Terminal for Care Home Residents – A long-term Study," in *ACM/IEEE Int. Conf. on Human-Robot Interaction (HRI)*, Vienna, 2017.

[4] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[5] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable SLAM system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, Nov. 2011.

[6] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar SLAM," in *IEEE Int. Conf. on Robotics and Automation*, 2016, pp. 1271–1278.

[7] G. Grisetti, R. Kmmerle, and K. Ni, "Robust optimization of factor graphs by using condensed measurements," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct 2012, pp. 581–588.

[8] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *IEEE Int. Conf. on Robotics and Automation*, Shangai, China, May 2011, pp. 3607–3613.

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

[10] Y. Latif and J. Neira, "Go straight, turn right: Pose graph reduction through trajectory segmentation using line segments," in *European Conference on Mobile Robots*, Barcelona, Spain, Sept. 2013.

[11] J. Vallvé, J. Sol, and J. Andrade-Cetto, "Graph SLAM sparsification with populated topologies using factor descent optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1322–1329, 2018.

[12] J. McDonald, M. Kaess, C. D. C. Lerma, J. L. Neira, and J. J. Leonard, "6-dof multi-session visual slam using anchor nodes," in *ECMR*, 2011.

[13] H. Johannsson, M. Kaess, M. Fallon, and J. J. Leonard, "Temporally scalable visual slam using a reduced pose graph," in *IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 54–61.

[14] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun, "Towards object mapping in non-stationary environments with mobile robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

[15] D. Wolf and G. S. Sukhatme, "Online simultaneous localization and mapping in dynamic environments," in *IEEE Int. Conf. on Robotics and Automation*, vol. 2, 2004, pp. 1301–1307.

[16] P. Biber and T. Duckett, "Experimental analysis of sample-based maps for long-term slam," *The Int. Journal of Robotics Research*, vol. 28, no. 1, pp. 20–33, 2009.

[17] T. Krajník, J. P. Fentanes, M. Hanheide, and T. Duckett, "Persistent localization and life-long mapping in changing environments using the frequency map enhancement," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016.

[18] M. Labbé and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sep 2014, pp. 2661–2666.

[19] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, "Dynamic pose graph slam: Long-term mapping in low dynamic environments," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct 2012, pp. 1871–1878.

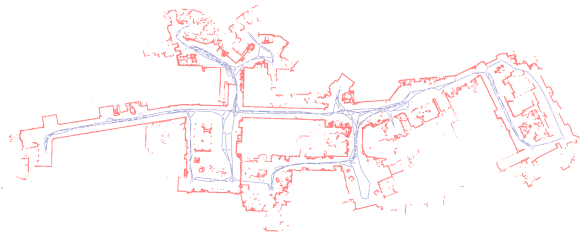
[20] J. Serafin and G. Grisetti, "Using extended measurements and scene merging for efficient and robust point cloud registration," *Robotics and Autonomous Systems*, vol. 92, pp. 91 – 106, 2017.

[21] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

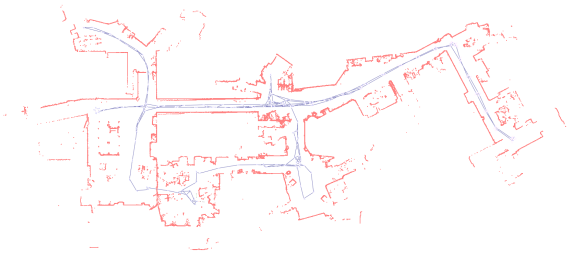
[22] M. Lázaro, L. Paz, P. Piniés, J. Castellanos, and G. Grisetti, "Multi-robot SLAM using condensed measurements," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Tokyo Big Sight, Japan, Nov. 2013.

[23] M. Fallon, H. Johannsson, M. Kaess, and J. J. Leonard, "The mit stata center dataset," *The Int. Journal of Robotics Research*, vol. 32, no. 14, pp. 1695–1699, Dec. 2013.

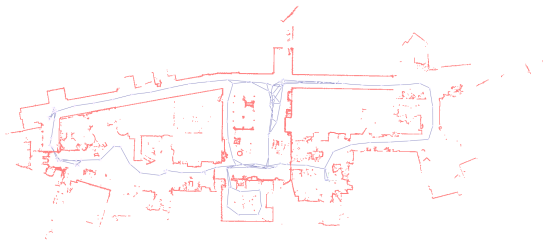
[24] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kmmerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. D. Tards, "A comparison of slam algorithms based on a graph of relations," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct 2009, pp. 2089–2095.



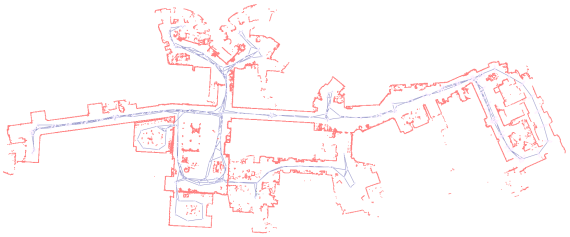
(a) Session 1



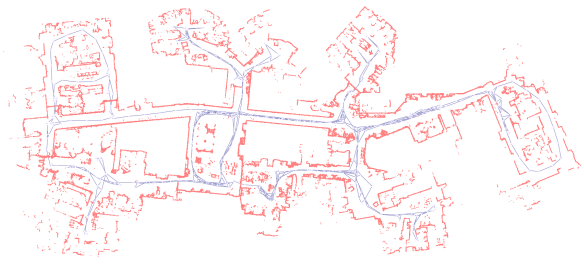
(b) Session 2



(c) Session 3

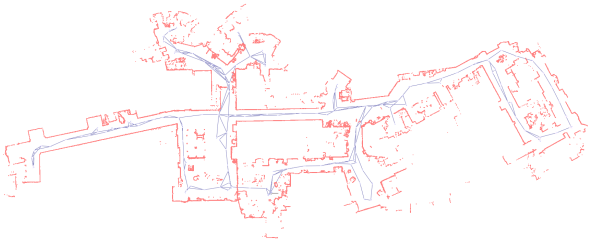


(d) Session 4

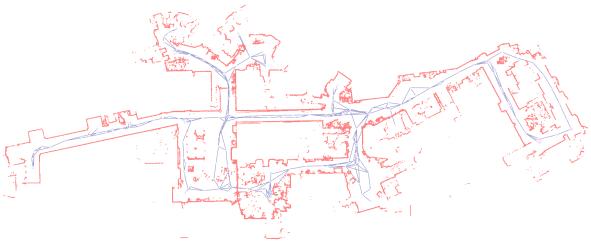


(e) Session 5

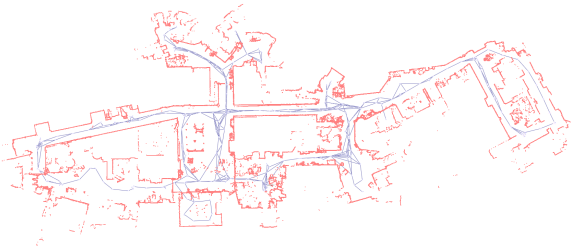
Fig. 9: Single session maps. Individual maps created by our system where high dynamics have been removed using the approach described in this paper (better seen on color).



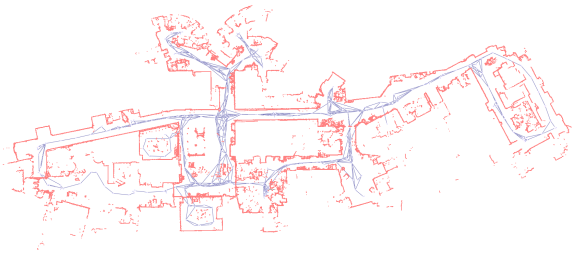
(a) Session 1 with merging



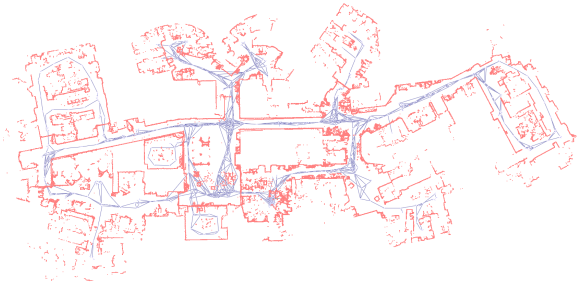
(b) Session 1 + session 2



(c) Session 1 + session 2 + session 3



(d) Session 1 + session 2 + session 3 + session 4



(e) All sessions

Fig. 10: Multiple session maps. Maps created after performing cumulative number of mapping sessions in which long-term graph maintenance has been applied (better seen on color).