3D Map Merging on Pose Graphs

Taigo Maria Bonanni[1], Bartolomeo Della Corte[1], and Giorgio Grisetti[1]

***Abstract*—In this paper, we propose an approach for merging 3D maps represented as pose graphs of point clouds. Our method can effectively deal with typical distortions affecting SLAM-generated maps. Traditional map merging techniques that use a single rigid body transformation to relate the reference frames of different maps. Instead, our approach achieves more accurate results by eliminating the inconsistencies resulting from distortions affecting the inputs, and can succeed in those situations where traditional approaches fail for substantial deformations. The core idea behind our solution is to localize the robot in a reference map by using the data from another map as observations. We validated our approach on publicly available datasets, and provide quantitative results that confirm its effectiveness on challenging instances of the merging problem.**

***Index Terms*—SLAM, Mapping, Localization**

I. INTRODUCTION

AFTER more than two decades of investigation, modern simultaneous localization and mapping (SLAM) approaches can produce on-line maps of large environments while operating under challenging conditions. The size of the environment to map is a critical factor influencing the success of a mapping session. The bigger the environment, the longer it takes to collect the data and build the map, and the higher the probability a failure will occur. To reduce the time required to map an area, one could rely on multi-robot SLAM techniques, where each robot employed will produce a partial map of the explored area. Regardless the number of robots used, even the more advanced systems are not immune to unexpected failures. SLAM engines might fail due to heterogeneous causes ranging from changes in the operating conditions to hardware malfunctions, that become more and more likely as the time of the mission increases. However, a partial map estimated before a failure occurred may be consistent and adequately represent a good portion of the area to be surveyed. Partial maps might also result from multiple data acquisitions in extremely large environments that cannot be mapped in a single session.

Map merging is the problem of constructing a single consistent map of an environment from a set of possibly noisy partial maps. Reusing already available incomplete maps, instead of entirely reprocessing the collected data, has the obvious computational advantage of divide-and-conquer strategies and exploits the spatial locality of the SLAM processes, while allowing to combine also data acquired at different times. Whereas the trend of modern multi-robot SLAM research is to reduce the amount of data exchanged among the robots [1] [2], trading off the quality of the reconstruction, map merging techniques focus on the off-line construction of highly detailed and consistent maps. The majority of map merging methods in the literature operate on 2D data, and address the problem by finding a set of transformations that maximizes the overlap of the input partial maps. This neglects unavoidable errors and deformations that affect the input maps. Under severe distortions, such as the ones obtained when a SLAM system operates for long time in open loop, a suitable transformation between partial maps cannot be found, even in presence of significant overlapping regions between them.

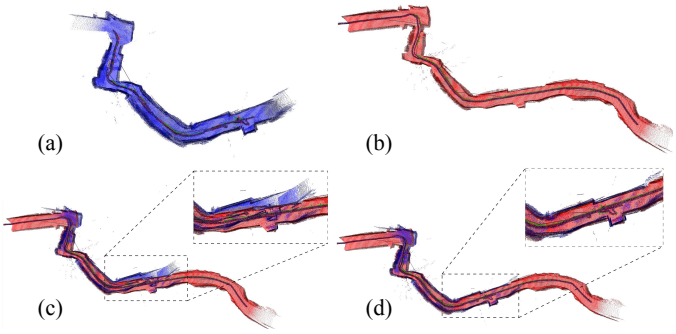


Fig. 1: Motivating example for our approach. **(a)** and **(b)** are two mergeable input maps. **(c)** shows the result of rigid merging procedure, where local inconsistencies are propagated in the reconstructed map. **(d)** is the output of our approach. The warping of the maps onto one another and the further optimization step produce a more consistent map.

In this paper, we present a map merging approach that operates in 3D and is specifically designed to cope with distortions in the input maps. Our solution models the maps as deformable graphs, and progressively applies local piece-wise deformations to maximize the overlap of similar regions, while removing the inconsistencies through local graph optimization. To the best of our knowledge, our approach is the first map merging method that works in 3D and can deal with distorted inputs. The core idea is to localize a robot in a reference map by using the data from the map to be merged. When operating on 3D point clouds, this might be computationally challenging since accurate localization requires the execution of time-consuming registration routines. To tackle this issue, our approach dynamically adapts the regions of the reference map where to perform the search, based on the state of the localization. To prevent false alignments, our method incorporates a consistency check that implicitly accounts for occlusions.

Manuscript received: September 10, 2016; Revised November 10, 2016; Accepted January 08, 2017.

This paper was recommended for publication by Editor Cyrill Stachniss upon evaluation of the Associate Editor and Reviewers' comments. *This work was supported by the European Commission under FP7-600890-ROVINA

[1]Bonanni, Della Corte and Grisetti are with Department of Computer, Control, and Management Engineering Antonio Ruberti, Sapienza University of Rome, Italy bonanni@dis.uniroma1.it

Digital Object Identifier xxxxxxxxxxxxxxxxxxx

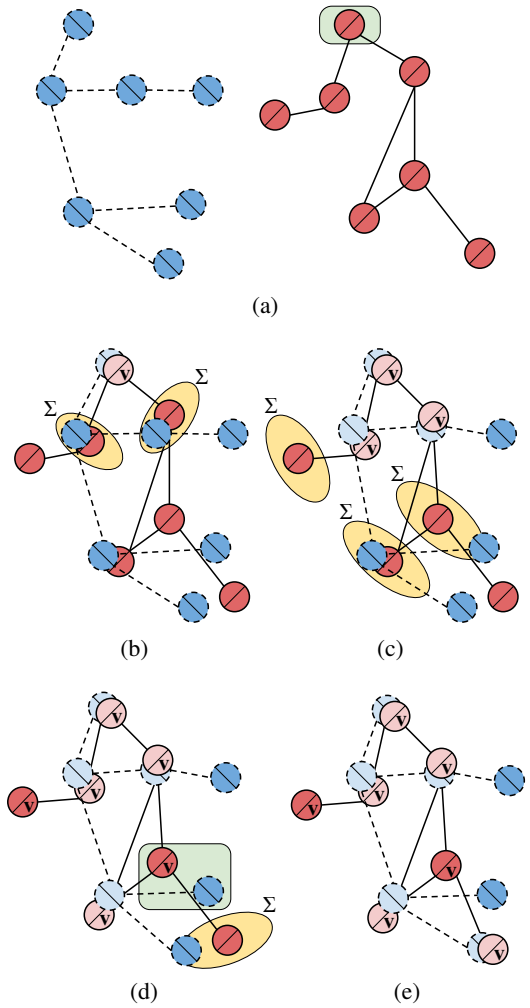


Fig. 2: Illustration of our merging procedure. **(a)** shows *reference* (left) and *matchable* (right) graphs. The merging starts from a random node (highlighted), by matching it against all the reference nodes. **(b)** In case of a successful match, we merge the poses (lighter color). We expand the matched node and mark it as visited (with a **v**). For each neighbor, we seek new matches within a search region Σ (size is approximate). **(c)** We expand each level of the graph. After each merge we update the poses, which are slightly moved. **(d)** In case of an unsuccessful match (highlighted), the poses remain in the original position. **(e)** The procedure terminates when all the matchable nodes are visited.

Figure 1 shows a motivating example on a fragment of data acquired in the *Priscilla Catacombs* of Rome in two different sessions. It displays the outcome of a procedure that rigidly merges the maps along with the results provided by our approach that applies local deformations.

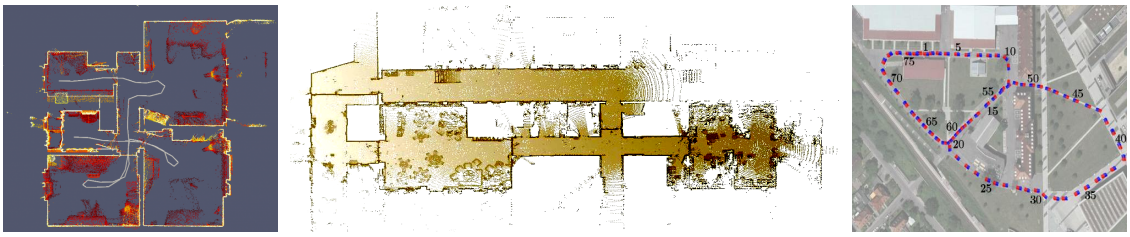
II. RELATED WORK

The earliest map merging techniques operate in 2D on raster maps such as occupancy grids. In this context, Birk and Carpin were among the first to address the map merging problem [3]. They propose a similarity metric for the registration of partial maps that maximizes the consistency of overlapping regions, while neglecting the contribution of non overlapping areas and penalizing the dissimilarities. The method is effective in case of small distortions in the partial maps while it tends to fail in case of heavier deformations. To find the best transform, the approach uses a brute force correlative search where a set of candidate transformations are returned. This time-consuming procedure has been replaced with a more effective matching in the Hough domain by Carpin [4]. This method restricts the search space by exploiting the dominant directions and displacements of straight regions in the maps and retrieves the candidate alignments by correlating the Hough spectra. This technique has been further improved by Saeedi *et al.* [5], by adapting the new cross correlation operator rendering it more sensible to the case of small overlaps.

Park *et al.* [6] operate on input grid-maps whose orientation and scale are unknown. The method is also robust to severe changes in the layout of the objects. The core idea is to approximate the empty regions with rectangles and describe the scene as the connectivity of these rectangles. This approach is rather promising, however the solution of the merging comes as a set of similarity transforms that do not contain enough degrees of freedom to express arbitrary deformations.

Neglecting the metric information in map merging has the obvious advantage of focusing on the topology that is not subject to geometric deformations. Topological map merging systems aim at providing a topological map instead of a metric one. Huang and Beevers [7] perform sub-graph merging by exploiting geometry and structure of topological maps. The problem is approached by merging topological graphs representing the partial maps. The method requires the regions to have a high topological salience to find robust similarities between the graphs. Saeedi *et al.* [8] compute probabilistic topological graphs by extracting the Voronoi diagram of each input. As these graphs encode also metric information, they can better capture the local structure of the environment. The diagrams are then merged by maximizing a cross correlation value through edge matching. In the context of visual mapping, Erinc and Carpin [9] propose a method to merge topological graphs that maximizes the algebraic connectivity, that is a spectral property of the adjacency matrix correlated to the number of consistent loops between the two maps.

In typical multi-robot SLAM contexts, each robot takes care of its own map and uses the observations collected from the other robots to enrich it. These perceptions can be either raw measurements or common portions for the environment (local maps). Cunningham *et al.* [10] proposed an effective schema to register pairwise portions of local maps by using RANSAC on a triangular mesh of landmarks. Whereas this approach is more resilient to mild deformations, it cannot operate with severe distortions that might occur when the local maps are large, since it attempts to explain the deformation with a single transform. During on-line SLAM this effect is not dominant, since all maps are constantly optimized thus minimizing the effects of distortions. Our approach is related to this work, since it partitions the maps to be aligned in small regions that are rigidly and recursively matched, but operates on dense point clouds instead of landmark based maps. In this paper, we present an offline map merging procedure for point cloud maps organized in pose graphs. Similarly to [9], we address the merging problem by connecting graphs in case of geometrically similar regions. Moreover, by adopting a non-



	sensor	location	dynamics	planar	trajectory	#scans
Apartment-Pomerleau	2D tilted laser	indoor	lightly dynamic	yes	-	45
AASS-loop	2D tilted laser	indoor	static	yes	111.4m	60
FreiburgCampus360_3D	2D tilted laser	outdoor	static	partially planar	723m	77
Priscilla Catacombs	depth-cameras	indoor	static	no	-	-

Fig. 3: **Top:** ground truth of the datasets used in the experimental evaluation, respectively *Apartment-Pomerleau* (courtesy of François Pomerleau), *AASS-loop* (courtesy of Martin Magnusson) and *FreiburgCampus360_3D* (courtesy of Bastian Steder). **Bottom:** summary of the characteristics of the datasets.

rigid merging formulation, our solution is able to increase the global consistency of the merged output against partially inconsistent input data. To this end, our algorithm applies local deformations to shared regions, to progressively morph pairs of maps to register the 3D points. To the best of our knowledge, this work is the first to approach 3D map merging with point clouds, especially in case of distorted data.

III. POSE GRAPHS

As their name suggests, pose graphs model a map as a graph. Let $\mathcal{M} = \langle \mathcal{X}, \mathcal{C} \rangle$ be one of such graphs. A node $\mathbf{x}_i \in \mathcal{X}$ represents a pose where the robot acquired an environment measurement. Let $\mathbf{X}_i$ be the homogeneous matrix representing the pose of $\mathbf{x}_i$. Each edge $\mathbf{e}_{ij} \in \mathcal{C}$ is labeled by a transform $\mathbf{T}_{ij}$ that expresses the relative spatial transformation between the connected nodes $\mathbf{x}_i$ and $\mathbf{x}_j$. A transformation $\mathbf{T}_{ij}$ is usually determined through a SLAM system by either registering the corresponding sensor measurements or computed from odometry integration. The unavoidable noise in the measurements will affect also the registration procedure, thus the transformation labeling an edge is subject to uncertainty. Under the assumption that the error is normally distributed, each edge is also labeled with a covariance matrix $\mathbf{\Sigma}_{ij}$ (or equivalently the inverse covariance $\mathbf{\Omega}_{ij}$) characterizing the uncertainty of the transformation. The most likely map $\mathcal{X}^*$ of a pose graph is obtained by computing the spatial configuration of the nodes that better satisfies the constraints induced by the edges. More formally:

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \| \mathbf{X}_i^{-1} \mathbf{X}_j - \mathbf{T}_{ij} \|_{\mathbf{\Omega}_{ij}} \tag{1}$$

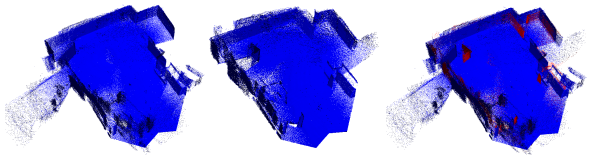
Here $\|\mathbf{a}\|_{\mathbf{\Omega}} = \mathbf{a}^T \mathbf{\Omega} \mathbf{a}$ is the $\mathbf{\Omega}$ norm of the vector $\mathbf{a}$. Typical SLAM engines rely on optimization packages such as g2o [11] or iSAM [12] to quickly retrieve the optimal arrangement of the nodes. The optimization of a pose graph and its connectivity are independent from the sensors used to acquire the data. This consideration allows us to use the nodes of a pose graph to represent local maps. By doing so, the global map can be seen as the union of all the local maps in the graph, each of them translated according to the node position. In our system, each node of the graph stores a local map in the form of a point cloud acquired by processing a contiguous sequence of depth or laser measurements.

IV. MAP MERGING USING POSE GRAPHS

If maps are represented as pose graphs, merging two maps can be seen as combining the input graphs in the most consistent manner. To this extent, the resulting map should contain all the information in the input maps, and portions of the two inputs that represent the same locations should appear in the same point in space. Let $\mathcal{M}^{\mathrm{r}}$ and $\mathcal{M}^{\mathrm{c}}$ be the partial maps that constitute our input. In the remainder of this paper we will refer to $\mathcal{M}^{\mathrm{r}}$ as the *reference* map, and $\mathcal{M}^{\mathrm{c}}$ as the *current* map. If the maps are free from deformations, it is sufficient to apply to the whole graph $\mathcal{M}^c$ a fixed transformation $\mathbf{T}$ that moves the current map onto the reference. This is what earliest map merging systems do. However, if the maps are subject to deformations we need to apply to each node of $\mathcal{M}^{\mathrm{c}}$ a potentially different transformation to justify the registration. In doing this operation, however, also the nodes of the reference map $\mathcal{M}^{\mathrm{r}}$ might be moved to compensate for the new information in the edges of $\mathcal{M}^{\mathrm{c}}$. In fact, we can delegate the operation of finding the most likely configuration of nodes to an optimization system. All we need to provide is a new set of constraints that expresses the relative location between nodes of the two maps. Unfortunately, this requires to carry on an expensive similarity and registration step between all pairs of local maps from $\mathcal{M}^{\mathrm{r}}$ and $\mathcal{M}^{\mathrm{c}}$. More formally, we can express map merging on pose graphs as:

1) Retrieving all constraints $\mathcal{C}^{\mathrm{r+c}} = \mathbf{e}_{ij}^{\mathrm{r+c}}$ between the maps $\mathcal{M}^{\mathrm{r}}$ and $\mathcal{M}^{\mathrm{c}}$
2) Finding the most likely configuration of the global map $\mathcal{M} = \langle \mathcal{X}^{\mathrm{r}} \cup \mathcal{X}^{\mathrm{c}}, \mathcal{C}^{\mathrm{r}} \cup \mathcal{C}^{\mathrm{c}} \cup \mathcal{C}^{\mathrm{r+c}} \rangle$ through non-linear optimization

Our merging procedure is built upon a graph traversal routine, in Algorithm 1, that visits all the nodes of $\mathcal{M}^c$ and searches for nodes in $\mathcal{M}^{\mathrm{r}}$ with similar local maps to establish inter-graph connections. Retrieving these inter-map constraints could lead to wrong associations, in case of environment aliasing. To



(a) (b) (c)

Fig. 4: Alignment example. **(a)** and **(b)** are two clouds captured in the same room at different times. Some changes are visible: the door on the left has been closed and two objects have been positioned in the middle of the room. These differences, highlighted in red in the resulting alignment shown in **(c)**, result as outliers.

avoid these false associations, one needs to consider the entire topological structure of the map during the matching. This can be done by delaying the decision that two nodes represent a similar location until sufficient evidence about the neighborhood supports the match. To gather this evidence, once a pair of nodes is said to match, our approach attempts to recursively match the neighbors of a node. In doing this operation, the current map is morphed onto the reference. This simulates a robot localizing in the reference map by using the data from the current map. To bootstrap the map merging procedure, we preliminarily rely on a place recognition routine that reports matching pairs of local maps if the free-space skeletons extracted from their traversable surfaces look similar. More in detail, Algorithm 1 starts from a random node $\mathbf{x}^{\mathrm{c}} \in \mathcal{X}^{\mathrm{c}}$ and looks for possible matching nodes of the reference graph that fall in the current dynamic search region (at the beginning, the search region will contain all the nodes of $\mathcal{M}^{\mathrm{r}}$). Once a match $\mathbf{x}^{\mathrm{r}*}$ in the reference map is found, we take all the neighboring nodes of $\mathbf{x}^{\mathrm{c}}$ (Alg. 1, line 18) and initialize their new positions by applying the transformation encoded in the edges (Alg. 1, line 20). Furthermore, we update the covariance through first-order error propagation (Alg. 1, line 21). This will expand the search region as the visit progresses: if it becomes too large (Alg. 1, line 22), we stop the expansion. Otherwise, we progress either until a failure occurs (see Section V) or when all the nodes in $\mathcal{M}^{\mathrm{c}}$ are visited. The graph traversal is executed through a breadth-first expansion of the neighbors using a queue to keep the current frontier. Each time a new node is extracted from the queue, we seek for the potential nodes in the reference map that fall in its neighborhood, and try to register them. In Figure 2 we illustrate the evolution of our procedure on a simple instance of the merging problem.

The execution time of our approach is dominated by the relatively expensive pairwise registration and verification of point clouds. This time can vary between 20 to 500 ms, depending on the size and the density of the local maps. Let c and r be respectively the number of nodes in the current and the reference graph. At the initial stage the algorithm has to register all reference nodes with all current nodes. Thus it might require rc matches. Execution time might be greatly reduced by using some efficient appearance-based matching strategy, not addressed in this work. Once the algorithm has found an initial match, it proceeds by visiting the current graph. It thus expands a number of nodes proportional to the size of the current graph, trying to match the evaluated node with all nodes in the reference that fall inside the search region.

V. EVALUATING THE QUALITY OF AN ALIGNMENT

As previously discussed, our system relies on a robust routine for finding the relative alignment between two point clouds. We carry on this operation within Algorithm 2 and we use NICP [13], an effective variant of ICP. When aligning two point clouds, NICP returns the following quantities:

- $\mathbf{T}^*$: a transform that best aligns the two clouds
- $\mathbf{\Omega}^*$: the information matrix describing the uncertainty of the transform
- n_{inl} number of points that are regarded as inliers
- n_{out} number of points that are regarded as outliers

Inliers and outliers are computed by reprojecting the registered cloud on spherical depth images, acquired from the same viewpoint. Intuitively, if two clouds are the same, the depth values of corresponding pixels in the two depth images are equal. Areas in the depth images with rather different depth values correspond to regions that are visible in one cloud, but are not in the other. This occurs, for instance, when one removes or adds an object from a scene. Conflicting pixels with different depth values are the "outliers", see Figure 4 for a graphical example. During the comparison, NICP ignores all those pixels where the depth value is not defined in both clouds. The number of inliers and outliers is combined into a score

$$s^* = \begin{cases} -\frac{n_{\mathrm{out}}}{n_{\mathrm{inl}}}, & \text{if } (n_{\mathrm{inl}} - \alpha * n_{\mathrm{out}}) \leq 0 \\ \frac{n_{\mathrm{out}}}{n_{\mathrm{inl}}}, & \text{otherwise} \end{cases}$$

where $\alpha > 1$ weights the cautiousness of the algorithm: the higher, the more conservative.

We use the score to discriminate between the outcomes of a registration attempt:

Success: if case of few outliers and many inliers, the registration algorithm returns a positive score. If this value is lower than a certain threshold, the matching is considered successful (Alg. 1, line 14). In this case, the algorithm resets the search region to an initial small value (Alg. 1, line 15), repositions the current node onto the reference map by applying the transform reported by the registration routine (Alg. 1, line 16) and inserts a new edge in the set of inter-map constraints (Alg. 1, line 17).

Conflict: if there is a high number of both inliers and outliers, the registration algorithm returns a negative score to indicate a possible conflict. This occurs when two local maps are partially consistent, but structures in one map that should be visible in the other are not. We handle the conflict in two possible ways, depending on the outcome of previous attempted alignments: if no successful matches occurred before the conflict, we treat it as a "no match". Otherwise, we assume that the nearest previous successful match was wrong. In this case, we cancel the relative edge, rollback and return a failure (Alg. 1, line 10).

No Match: NICP directly returns a no match, if the number of inliers is less than the minimum required. This happens when the evaluated local maps have limited/no overlap or are too different. In this case the search region is not reset.

Algorithm 1

Input: $\mathcal{M}^{\mathrm{r}}$, $\mathcal{M}^{\mathrm{c}}$, $\mathbf{x}_j^{\mathrm{c}}$
Output: $\mathcal{C}^{\mathrm{r+c}}$

function TRYMERGE($\mathcal{M}^{\mathrm{r}}, \mathbf{x}^{\mathrm{c}}$)
  $\mathbf{\Sigma}_j \leftarrow \mathbf{\Sigma}_{\mathrm{init}}$ ▷ initialize search region
  $\mathcal{Q} \leftarrow 0$ ▷ clear the queue
  $\mathcal{V} \leftarrow \mathcal{X}^{\mathrm{c}}$ ▷ initialize $\mathcal{V}$ to the nodes of current graph
  $\mathcal{C}^{\mathrm{r+c}} \leftarrow 0$ ▷ clear the edges between the two maps
  $\mathcal{Q}$.push($\mathbf{x}^{\mathrm{c}}$)
  while !$\mathcal{V}$.empty() **do**
    $\mathbf{x}^{\mathrm{c}} \leftarrow \mathcal{Q}$.pop()
    $\langle \mathbf{x}^{\mathrm{r}*}, s^*, T^*, \mathbf{\Omega}^* \rangle \leftarrow$ findBestMatch($\mathcal{M}^{\mathrm{r}}, \mathbf{x}_j^{\mathrm{c}}, \mathbf{\Sigma}_j^{-1}$)
    if $s^* < 0$ **then** ▷ Conflicting information
      rollback()
      return fail
    else
      if $s^* < \tau_{\mathrm{match}}$ **then** ▷ Successful match
        $\mathbf{\Sigma}_j \leftarrow \mathbf{\Sigma}_{\mathrm{init}} + {\mathbf{\Omega}^*}^{-1}$ ▷ Search region reset
        $\mathbf{X}_j^{\mathrm{c}} \leftarrow \mathbf{X}_j^{\mathrm{c}} \mathbf{T}^*$ ▷ Move $\mathbf{x}_j^{\mathrm{c}}$ to the reference
        $\mathcal{C}^{\mathrm{r+c}} \leftarrow \mathcal{C}^{\mathrm{r+c}} \cup \langle \mathbf{T}^*, \mathbf{x}^{\mathrm{r}*}, \mathbf{x}_j^{\mathrm{c}} \rangle$ ▷ new edge
      $\mathcal{N}^{\mathrm{c}} \leftarrow$ findNonVisitedNeighbors($\mathbf{x}_j^{\mathrm{c}}$)
      for all $\mathbf{x}_k^{\mathrm{c}} \in \mathcal{N}^{\mathrm{c}}$ **do**
        $\mathbf{X}_k^{\mathrm{c}} \leftarrow \mathbf{X}_j^{\mathrm{c}} \mathbf{T}_{jk}$ ▷ Transform
        $\mathbf{\Sigma}_k \leftarrow \mathbf{J}_{jk} \mathbf{\Omega}_{jk}^{-1} \mathbf{J}_{jk}^T + \mathbf{\Sigma}_j$
        if $det(\mathbf{\Sigma}_k) < \tau_{\mathrm{largeRegion}}$ **then**
          $\mathcal{V}.push(\mathbf{x}_k^{\mathrm{c}})$
  return $\mathcal{C}^{\mathrm{r+c}}$

Algorithm 2

Input: $\mathcal{M}^{\mathrm{r}}$, $\mathbf{x}^{\mathrm{c}}$, $\mathbf{\Omega}_{s.r.}$
Output: $\langle \mathbf{x}^{\mathrm{r}*}, s^*, T^*, \mathbf{\Omega}^* \rangle$

function FINDBESTMATCH($\mathcal{M}^{\mathrm{r}}, \mathbf{x}^{\mathrm{c}}, \mathbf{\Omega}_{s.r.}$)
  $\mathcal{R} \leftarrow \mathbf{x}_i^{\mathrm{r}} : \|\mathbf{X}_i^{\mathrm{r}} - \mathbf{X}^{\mathrm{c}}\|_{\mathbf{\Omega}_{s.r.}} < \tau_{\mathrm{neighbor}}$ ▷ Neighbors
  $s^* \leftarrow \infty$ ▷ Initialize the matching score
  $\mathbf{x}^{\mathrm{r}*} \leftarrow$ null ▷ Initialize the best reference node
  $\mathbf{T}^* \leftarrow \mathbf{I}$ ▷ Initialize the reference matrix
  $\mathbf{\Omega}^* \leftarrow \mathbf{I}$ ▷ Initialize the information matrix
  for all $\mathbf{x}_i^{\mathrm{r}} \in \mathcal{R}$ **do**
    $\langle s, \mathbf{T}, \mathbf{\Omega} \rangle \leftarrow \mathrm{register}(\mathbf{x}_i^{\mathrm{r}}, \mathbf{x}^{\mathrm{c}})$
    if $s < s^*$ **then** ▷ Update the best match
      $s^* \leftarrow$ s
      $\mathbf{x}^{\mathrm{r}*} \leftarrow \mathbf{x}_i^{\mathrm{r}}$
      $\mathbf{T}^* \leftarrow \mathbf{T}$
      $\mathbf{\Omega}^* \leftarrow \mathbf{\Omega}$
  return $\langle \mathbf{x}^{\mathrm{r}*}, s^*, T^*, \mathbf{\Omega}^* \rangle$

VI. EXPERIMENTS

We validate the performance of our 3D map merging algorithm through extensive experiments. To this end, we use several publicly available datasets, collected with depth-sensors and 3D laser scanners. The selected datasets are:

- *Apartment-Pomerleau* [14], acquired with a tilted laser range scanner in an apartment
- *AASS-loop* [15], acquired with a tilted laser scanner in the AASS center of Örebro University
- *FreiburgCampus360_3D* [16], acquired with a pan-tilted SICK LMS laser range scanner in the Campus of Freiburg University
- *Priscilla Catacombs* [17], acquired during the Rovina project in the Priscilla catacombs of Rome with depth-sensors (Asus Xtion)

Figure 3 shows the main characteristics of each dataset and, if available, the robot trajectories. For each dataset, we tested two different merging techniques. First, we computed a rigid merge with an ICP-based procedure. Then, we applied our merging algorithm and compare the outcomes.

We provide both qualitative and quantitative results. The former are obtained by visual inspection of the different figures depicted. The latter are expressed by comparing the entropy values of the different outputs and, where possible, computing the root mean square error. We measure the entropy of a 3D map as in [18], while we estimate the relative translational and rotational errors as described in [19]. Beforehand, we apply a voxelization of the data, with a voxel resolution of 0.01 meters. Since the entropy of a point cloud grows with the density of points, and these datasets are characterized by very different point densities, we report also the map densities, expressed as number of points per 1mt voxel. We found that normalizing the entropy to the density of the cloud provides a metric that better reflects the results of visual inspection. In these experiments, we selected NICP [13] as cloud matching algorithm. To discriminate whether to accept or to reject a candidate alignment, we used a thresholding value τ_{match}. Throughout the experimental evaluation, we adopted different values for τ_{match} depending on the density of the 3D clouds. In the following we describe for each dataset how we constructed the pose graphs and the results obtained.

A. Apartment-Pomerleau

This benchmark dataset contains 45 3D scans, obtained with a tilted 2D laser in an indoor environment. A ground-truth for the poses is provided, which we used to build the maps. We generated two chunks by dividing the set of laser scans so as to guarantee a small overlapping region. We built each graph representing the positions where the scans were acquired as nodes and then we connected consecutive poses with edges. The graphs, shown in Figure 6, share an overlap in the central corridor of the apartment. For this experiment, we set the acceptance threshold τ_{match} to 0.15. As shown in the images, there are no notable differences between a rigid merging procedure and our approach. This is due to the high quality of the ground truth data, allowing for the creation of input chunks without any relevant inconsistency. Therefore, in this case both approaches provide consistent results, as confirmed in Table I, even if our approach provides a slightly better result. Same holds for the error estimation in Table II, where no significant improvements are obtained with the non rigid approach.

B. AASS-loop

This benchmark dataset is composed by 60 3D laser scans. Similarly to the previous dataset, it was recorded in an indoor structured environment and the author provides the ground truth. We built two chunks halving the whole set of scans, to

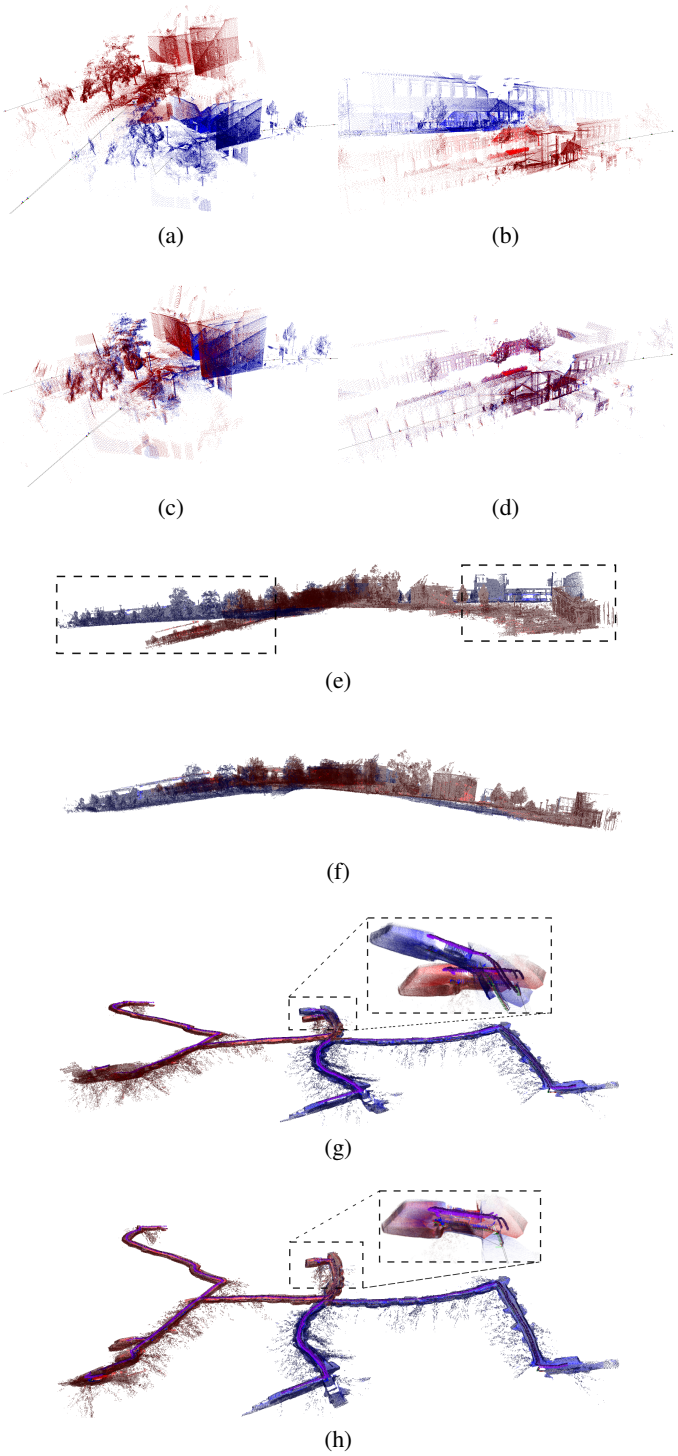


Fig. 5: Analysis of merging results. **(a)** and **(b)** are two misaligned areas in the reconstruction of *FreiburgCampus360_3D* using ICP merging. **(c)** and **(d)** show the same areas, this time correctly merged using our approach. **(e)** is the full reconstruction of *FreiburgCampus360_3D* obtained trough ICP merging. Note how the highlighted ends of the two chunks cannot be aligned because of local distortions. **(f)** is our reconstruction of *FreiburgCampus360_3D*. In this case, the chunks are properly merged by deforming the ends onto one another. **(g)** Detail of *Priscilla Catacombs* reconstructed using ICP merging. The rigid transformation does not merge the highlighted rooms, due to misalignments in the corridors. **(h)** Reconstruction of *Priscilla Catacombs* using our approach. The merging through deformation allows to completely merge the rooms.

obtain the input trajectories shown in Figure 6. We created the chunks as we did for the previous dataset. Referring to the complete map in Figure 3, the chunks share the rooms at the right-hand side of the image and the small corridor. For this dataset, setting $\tau_{\text{match}} = 0.15$ yield to satisfying results. The use of ground truth data, as well as for *Apartment-Pomerleau* dataset, reduces the chance of relevant inconsistencies. Both approaches provide good results as shown in Figure 6 and comparing the entropy values of the respective outputs. Similarly to the previous case, our approach outperforms the rigid one by a small value in terms of entropy with no important differences in the error estimation.

C. FreiburgCampus360_3D

This benchmark dataset consists of 77 scans. It differs substantially from the previous ones since it captures an outdoor environment. The author provides the odometry of the robot equipped with a 2D tilted laser scanner, and the optimized trajectory obtained through a SLAM engine. We built the input chunks starting from the SLAM trajectory and created the graphs taking the scans as nodes and creating the edges from the optimized trajectory. The scans are distributed so that the chunks share an overlap in the center of the original map and on their respective first and last poses, since the robot trajectory forms an eight-shape. It is important to notice that, despite representing the same place, first and last scan lie at different elevations, resulting in a heavy inconsistency. In this case, we set $\tau_{\text{match}} = 0.15$. Figure 6 shows a top-view of both merged maps. Some discrepancies are easily visible, such as the non connected ends of the chunk trajectories. The rigid merging procedure identified the center of the original map as the most promising matching region, and from there it is unable to complete the merge. In the side view in Figure 5e, it is possible to observe how the planar inconsistencies of the input chunks are not corrected by the rigid approach. On the contrary, our solution correctly recognizes the aforementioned overlapping areas as merging points. In the side-view in Figure 5f, it is possible to notice how the planar inconsistencies are corrected by our merging procedure. This is visually evident in the images provided. Figure 5a and Figure 5b show details of the uncorrected misalignments obtained with a rigid merge. Figure 5c and Figure 5d display the respective results obtained with our approach. Also in this case, our algorithm outperforms the rigid approach as demonstrated by the relative errors shown in Table II where, especially in the translational part, we obtain an improvement of one order of magnitude. Moreover the entropy values in Table I confirm this advancement if compared with the low map density.

D. Priscilla Catacombs

This dataset consists of two acquisitions performed in the *Priscilla Catacombs* of Rome at different days, using two Asus Xtion depth-sensors. We independently built the input maps with NICP [13]. To be consistent with the previous datasets, and taking into account the different fields of view of laser and depth-cameras, we generated a node approximately every 5 meters. The input maps share a single corridor with a small

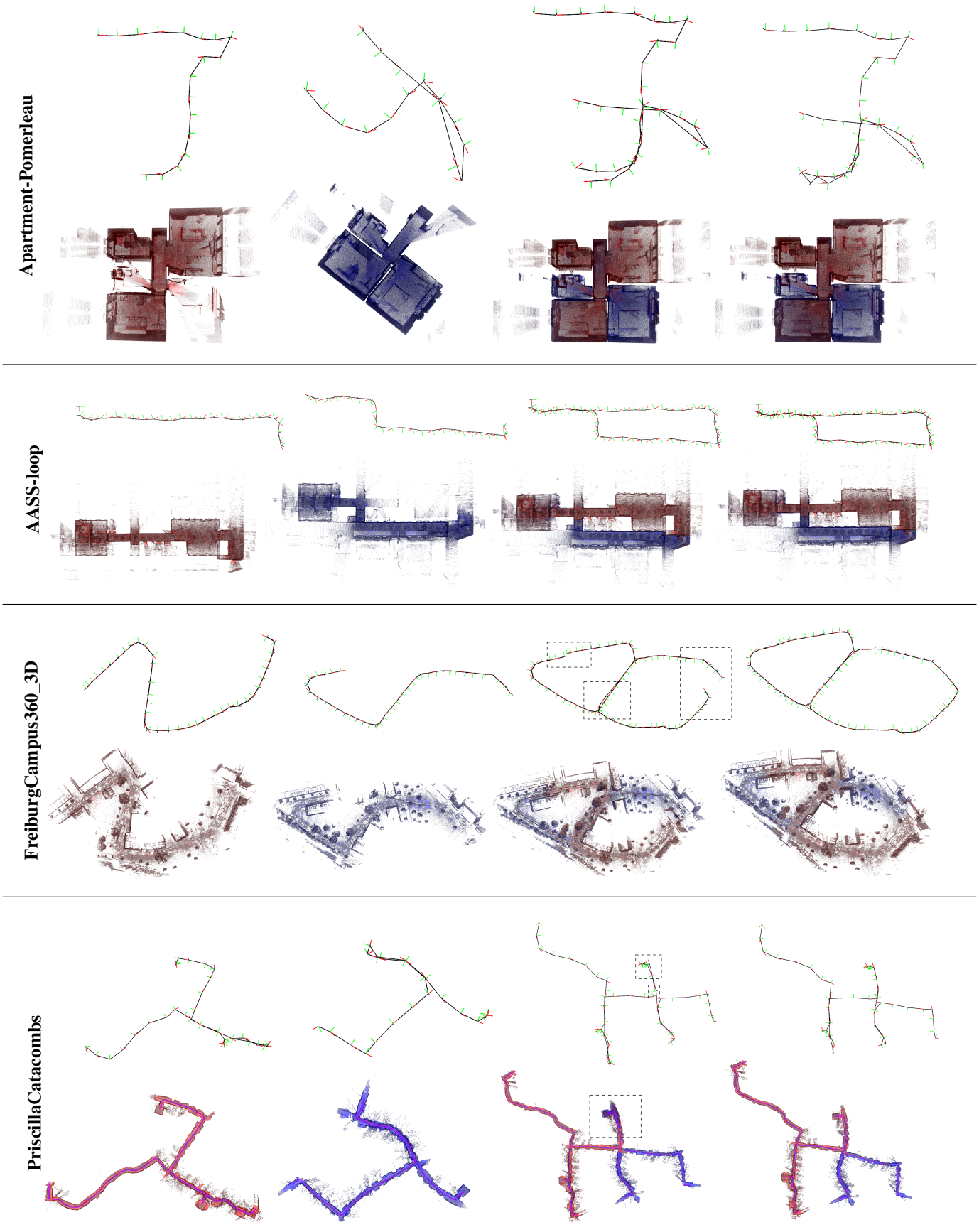


Fig. 6: Summary of the experiments. For each dataset, two rows display respectively trajectory and 3D point cloud. Column-wise, the order is: **(a)** and **(b)** are first and second input chunk; **(c)** is the ICP-based merged output; **(d)** is our merged output. An interactive webgl 3D viewer to better observe the results can be found at www.dis.uniroma1.it/~bonanni/mm-webgl.html.

	~density	chunk 1	chunk 2	sum	rigid merge	our approach
Apartment-Pomerleau	10000	2303870	2262639	4566509	4398299	**4327222**
AASS-loop	600	1307095	1349825	2656920	2576875	**2564753**
FreiburgCampus360_3D	76	2645846	2936904	5582750	5355552	**5296621**
Priscilla Catacombs	3200	2840765	1893129	4733894	4728827	**4630378**

TABLE I: Entropy Computation

	Relative Translational Error		**Relative Rotational Error**	
	rigid merge	**our approach**	**rigid merge**	**our approach**
Apartment-Pomerleau	1.6703e-03	**1.6673e-03**	**1.7146e-04**	4.1833e-04
AASS-loop	8.4202e-03	**2.9648e-03**	**1.2610e-04**	5.6036e-04
FreiburgCampus360_3D	4.9436e-01	**1.1916e-02**	1.8166e-03	**1.5684e-03**

TABLE II: Relative Translational and Rotational Error

square room having an area of about $16m^2$. Due to the higher point density of the data, we set the threshold to a lower value, *i.e.* $\tau_{\text{match}} = 0.05$. While the top-view results in Figure 6 do not show any substantial difference, the rigid merge shown in Figure 5g highlights inconsistencies similar to the ones observed in the reconstruction of *FreiburgCampus360_3D*. Due to the distortions affecting one of the maps, the small room is not merged. Instead, our solution corrects the misalignments (Figure 5h) by applying merging and optimization procedures. The entropy values in Table I confirm these differences, with a reasonable improvement obtained with our approach. Not having a ground-truth for this dataset, it is not possible to evaluate relative translational and rotational errors.

VII. CONCLUSIONS

In this paper we presented a map merging solution designed for 3D maps. Our approach can deal with deformed input maps and addresses the problem by simulating a robot localizing on the reference map by using the data from the map to be merged. We limit the computation of our algorithm by dynamically adapting its search region, depending on the state of the simulated localization. Quantitative experiments performed on publicly available datasets confirm the applicability of our approach. We are not aware of other map merging algorithms that can operate on 3D point clouds. We also provide an interactive WebGL 3D viewer at www.dis.uniroma1.it/~bonanni/mm-webgl.html.

REFERENCES

[1] M. Lazaro, L. Paz, P. Piniés, J. Castellanos, and G. Grisetti, "Multi-robot slam using condensed measurements," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1069–1076.

[2] L. Carlone, M. K. Ng, J. Du, B. Bona, and M. Indri, "Rao-blackwellized particle filters multi robot SLAM with unknown initial correspondences and limited communication," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 243–249.

[3] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1384–1397, 2006.

[4] S. Carpin, "Fast and accurate map merging for multi-robot systems," *Autonomous Robots*, vol. 25, no. 3, pp. 305–316, 2008.

[5] S. Saeedi, L. Paull, M. Trentini, M. Seto, and H. Li, "Map merging using Hough peak matching," in *Intelligent Robots and Systems, International Conference on*. IEEE, 2012, pp. 4683–4688.

[6] J. Park, A. J. Sinclair, R. E. Sherrill, E. A. Doucette, and J. W. Curtis, "Map merging of rotated, corrupted, and different scale maps using rectangular features," in *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE, 2016, pp. 535–543.

[7] W. H. Huang and K. R. Beevers, "Topological map merging," *The Int. Journal of Robotics Research*, vol. 24, no. 8, pp. 601–613, 2005.

[8] S. Saeedi, L. Paull, M. Trentini, M. Seto, and H. Li, "Group mapping: A topological approach to map merging for multiple robots," *IEEE Robotics & Automation Magazine*, vol. 21, no. 2, pp. 60–72, 2014.

[9] G. Erinc and S. Carpin, "Anytime merging of appearance-based maps," *Autonomous Robots*, vol. 36, no. 3, pp. 241–256, 2014.

[10] A. Cunningham, K. M. Wurm, W. Burgard, and F. Dellaert, "Fully distributed scalable smoothing and mapping with robust multi-robot data association," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1093–1100.

[11] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *Robotics and Automation, Int. Conf. on*. IEEE, 2011, pp. 3607–3613.

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

[13] J. Serafin and G. Grisetti, "NICP: Dense normal based point cloud registration," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 742–749.

[14] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms," *The International Journal of Robotics Research*, vol. 31, no. 14, pp. 1705–1711, Dec. 2012.

[15] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, "Automatic appearance-based loop detection from three-dimensional laser data using the normal distributions transform," *Journal of Field Robotics*, vol. 26, no. 11-12, pp. 892–914, 2009.

[16] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard, "Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1249–1255.

[17] V. A. Ziparo, M. Zaratti, G. Grisetti, T. M. Bonanni, J. Serafin, M. Di Cicco, M. Proesmans, L. J. Van Gool, O. Vysotska, I. Bogoslavskyi, *et al.*, "Exploration and mapping of catacombs with mobile robots." in *SSRR*, 2013, pp. 1–2.

[18] J. M. Sáez, A. Hogue, F. Escolano, and M. Jenkin, "Underwater 3D SLAM through entropy minimization," in *International Conference on Robotics and Automation*. IEEE, 2006, pp. 3562–3567.

[19] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, 2009.