Robust Homing for Autonomous Robots

Igor Bogoslavskyi Mladen Mazuran Cyrill Stachniss

***Abstract*— In autonomous exploration tasks, robots usually rely on a SLAM system to build a map of the environment online and then use it for navigation purposes. Although there has been substantial progress in robustly building accurate maps, these systems cannot guarantee the consistency of the resulting environment model. In this paper, we address the problem of robustly guiding a robot back to its starting location after exploring an unknown environment—even if the mapping system fails to produce a consistent map. To tackle this problem, we propose a two-step procedure. First, we check if the current map is consistent using a statistical test. If the map is consistent, we navigate the robot back to its starting location using a standard navigation system. In case of an inconsistent map, however, we propose to rewind the trajectory from the current location to the start without relying on a map. We implemented the proposed system in ROS and showcase its effectiveness on an autonomous exploration robot in real underground and office environments.**

I. Introduction

The ability to robustly operate without user intervention is an important capability for exploration robots, especially if there are no means for communication between the robot and an operator. State estimation processes in the robot's navigation system, however, may fail and there is no guarantee that they will operate flawlessly throughout the time the robot navigates. Thus, it is of great importance to design a system that enables the robot to navigate back to its starting position without the risk of it getting lost, even if a component as central as the mapping system fails. This is the problem that we address in this paper.

Our work is motivated by the autonomous exploration robot depicted in Figure 1. It is used to explore difficult-to-access underground environments such as ancient Roman catacombs. As no communication between the robot and an operator is possible most of the time, the platform has to be truly autonomous. During the course of an exploration task, the robot constructs a map of the environment via a graph-based SLAM system. As long as this map is consistent, the robot can perform autonomous navigation by planning the shortest path—for example using A*—from its current location to its starting point using the map. Although modern SLAM systems are fairly robust, they may still fail, for example, due to wrong data associations generated by the SLAM front-end. Even current state-of-the-art SLAM system cannot *guarantee* the consistency of the resulting map, nor do they provide robust means to decide whether the constructed map is consistent or not. Computing a path based on an inconsistent map, however, is likely to lead to a failure and a possibility of losing the robot if operating in a hazardous environment.

Igor Bogoslavskyi and Cyrill Stachniss are with Institute for Geodesy and Geoinformation, University of Bonn, Germany. Mladen Mazuran is with the Autonomous Intelligent Systems Lab, University of Freiburg. This work has partly been supported by the European Commission under the grant number FP7-ICT-600890-ROVINA.

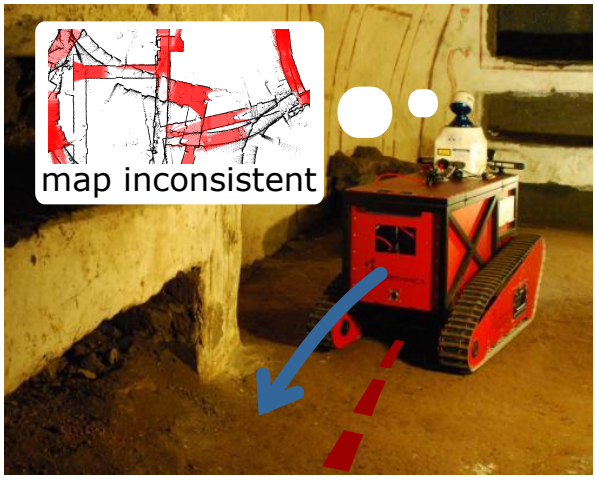

Fig. 1. When the statistical map consistency tester provides the robot with the information that the map is not consistent anymore the robot starts rewinding the trajectory using our method.

To avoid that the robot gets lost, we propose a novel robust homing system consisting of two distinct parts. The first part performs a statistical analysis of the map and thus provides information about its consistency. We build upon our previous work [7] for performing a cascade of pair-wise consistency checks using observations perceived from the same areas. To avoid performing such checks on the overall map, we reduce the area to analyze by planning the shortest homing route for the robot assuming a consistent map. We then analyze the map consistency only along that path and can estimate online if the map around this path is consistent or not with a given confidence level. The second part of our approach is responsible for driving the robot back to its starting location by rewinding the trajectory that the robot took to reach its current pose. This operation is performed without any map knowledge (as the map is inconsistent). If the motions of the robot were perfect, i.e., if they would lead *exactly* to the desired robot pose in the world frame, we would be able to simply invert the motion commands performed by the robot and could safely reach the start location. Both motion execution and odometry, however, are often noisy and can generate erroneous estimates. As a result, simply following inverse motion commands will not bring the robot to the starting location in the real world. Therefore, we take into account the sensor information to

guide the robot back by matching the observations with the measurements taken in the past.

II. Related Work

In robotics, most approaches to exploration focus on selecting view points that minimize the time needed to cover the terrain, such as the method of Yamauchi [16]. In order to address the robots pose uncertainty, extension of this exploration method have been proposed. For example information-theoretic methods that seek to minimize the uncertainty in the belief about the map and the trajectory [14], [6]. Similarly, Sim *et al.* [12] select vantage points to optimize map accuracy by striving for loop closures. Carillo *et al.* [1] use the Shannon and Rényi entropies to employ an exploration strategy that jointly reduces the map and the localization uncertainties. All these methods seek to build consistent maps and allow the robot to navigate in the maps during exploration.

A key problem in autonomous exploration, however, is that in case of a SLAM failure, the map becomes inconsistent. This can prevent the robot from continuing its exploration mission and—even worse—from being able to navigate back. It is therefore important to be able to perform reliable navigation without relying on a map. We are, however, not aware of any robotic navigation system that monitors the map to detect a SLAM failures to trigger a homing action if the map becomes inconsistent. Our idea of homing has been proposed in brief as part of an article on exploration [15], however the work lacks a real world evaluation.

Rewinding a trajectory is related to teach-and-repeat navigation systems. One of important outposts of teach-and-repeat is visual teach-and-repeat (VT-R). Works of Furgale *et al.* [3], [2], Nitsche *et al.* [8] and Ostafew *et al.* [9] show that this approach is indeed suitable to steer a robot in complicated environments based solely on visual sensors without relying on a consistent map. These visual methods, however, need substantial adaptation in order to be used in a setup similar to ours: using monocular cameras to localize through feature detection relies on having enough visual information, which ancient catacombs typically do not possess.

Teach-and-repeat approaches have also been used in lidar-based settings. Sprunk *et al.* [13] present an approach that relies on precise localization of the robot based on lidar measurements recorded during a trajectory demonstration. Similarly, Furgale *et al.* [4] perform an ICP-based teach-and-repeat approach on an autonomous robot equipped with a high precision 3D spinning lidar. They extend the standard teach-and-repeat approach by adding a local motion planner to account for dynamic changes in the environment. Our method to rewind the trajectory is similar to the teach-and-repeat setup in this formulation. However, in contrast to the mentioned methods, we use a substantially less accurate robot and thus have to cope with somewhat larger deviation from the reference trajectory. Further, we are rewinding a trajectory, not repeating it, and the sensor suite also differs significantly: Sprunk *et al.* use a high-precision 2D SICK laser range finder, which allows for accurate position correction. Furgale *et al.* also rely on a precise lidar sensor which is able to provide them with more information than our RGB-D camera setup, thus simplifying scene matching.

III. Robust Homing Using Map Consistency Checks

Under the assumption of map consistency, homing can be solved with existing tools. This can be done by computing a collision-free path from the current to the start location and executing this path with a standard navigation pipeline. Such a navigation system would, for instance, localize the robot in the map built so far and plan the shortest path towards home using A* or a similar approach. If the map, however, is not consistent due to a failure in the underlying SLAM system, this approach is likely to lead to a deadlock situation from which the robot cannot escape easily.

We address the problem of robust homing in a two-stage approach. First, we plan a path from the current location towards home under the assumption of a consistent map. Then, we apply our previously proposed map consistency estimation approach [7] to evaluate if the map along the planned path is consistent with a given confidence, but use a Kinect-style sensor and no laser range finder. In the second stage, we navigate the robot back home. If the path towards the start is consistent, we simply execute this plan. Otherwise, we aim at reversing the trajectory that the robot has executed so far by aligning the current observation with the observations obtained on the way from the starting location to the current one. As we show throughout this paper, this yields a robust strategy to bring a robot home to its starting location.

A. Map Consistency Test

Our previously proposed map consistency estimation approach [7] builds upon a pose-graph representation of the environment, containing the viewpoints of the robot from which individual observations have been taken. We start with evaluating the consistency of pairs of range readings. The approach of Mazuran *et al.* has been formulated for 2D laser range scans and describes the discrepancy between two scans by computing how much the two range scans occlude each other's free space.

To estimate the occlusion of the free space, the approach computes for each scan the polygon over the area covered by the scan at the robot pose, which it then checks against the end points of another scan. The intuition in the noiseless case is that if both scans are consistent with each other, all end points should lie outside or at the boundary of the polygon. A so-called inconsistency distance $d(p)$ is then defined for a point p, which lies inside the polygon of another scan, as the Euclidean distance of p to the closest point on the polygon boundary of the other scan. Intuitively speaking, for a consistent map, we assume that the inconsistency distances $d(p)$ are in line with the sensor noise of the proximity sensor. Substantially larger values for $d(p)$ typically indicate that

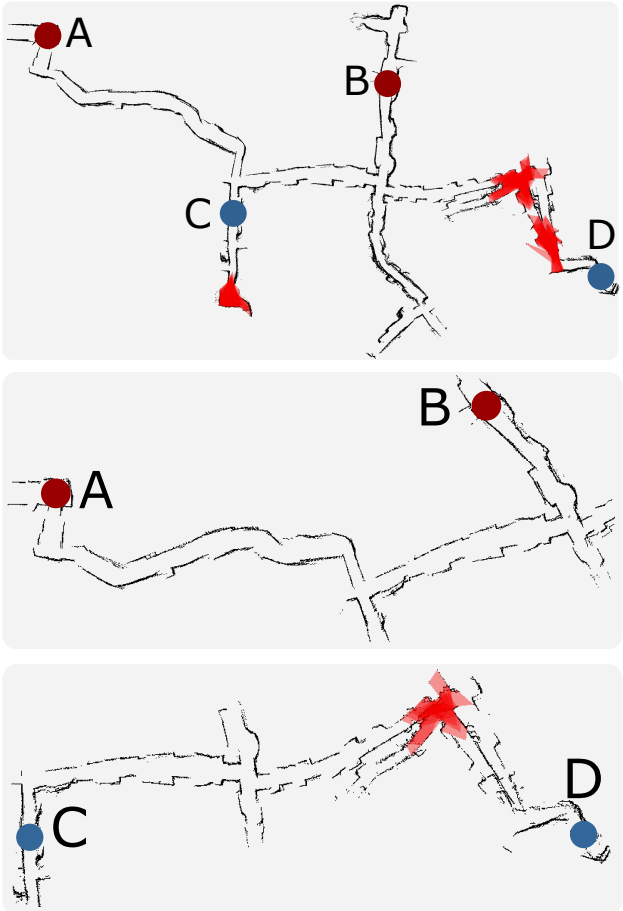

Fig. 2. The top image shows the map built so far with the detected inconsistencies (inconsistent scans are shown in red). The middle one shows a submap that is built using only the scans recorded around the A* path from A to B computed in the full map. In this example, no inconsistencies are present and none are detected. The bottom image is done in the same way as the middle one, but the A* path is computed from C to D and here, the map inconsistencies are correctly detected.

the scans are not properly aligned and the map may be inconsistent in a local neighborhood of the scans.

More concretely, we can expect that, under the assumption of a correct alignment of two scans, on average half of the end points from the first scan have an inconsistency distance $d(p) > 0$ in the second scan and vice versa. This is due to the sensor noise in the range measurements. According to [7], we can formulate a statistical test for the sum of inconsistency distances $d(p)$. This test evaluates if pairs of scans are consistent given the sensor noise or reveal a larger error and thus an inconsistency.

To assess global map consistency, we could conduct this test for all pairs of scans and consider a map to be consistent if all tests are successful. The problem, however, is that a single statistical test will produce a wrong result with probability α. Thus, if we test a single scan, which overlaps with r other scans, this yields a type I error probability of $1-(1-\alpha)^r$ and thus renders the direct application of the pairwise approach unsuitable. The key trick is to model the outcome of the pairwise hypothesis test as a Bernoulli-

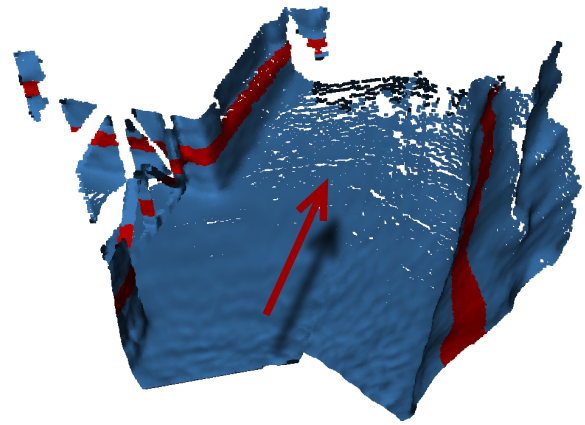

Fig. 3. An example of the point cloud from the double Xtion system we use to log data in the Priscilla catacomb. The red line shows the "scan" line to generate the simulated 2D scan from 3D point cloud. We discretize all the points within the red line into bins according to their horizontal dispersion from the camera viewing direction shown here as a floating red arrow.

distributed random variable with parameter α. As a result of that, the number of failed tests follows a binomial distribution with parameters α and r. Given that, we can compute the maximum number $\hat{\xi}$ of tests that are allowed to fail at a confidence level $1-\alpha'$ as

$$\hat{\xi} = \min_{0\leq\xi\leq r}\left\{\xi \;\middle|\; \sum_{i=\xi+1}^{r}\binom{r}{i}\alpha^i(1-\alpha)^{r-i} \leq \alpha'\right\}. \quad (1)$$

This allows for computing a cascaded hypothesis test for all overlapping scans: We perform all pairwise hypothesis tests and if the number of failed tests is smaller than $\hat{\xi}$, the overall consistency test is positive otherwise negative. For more details, we refer to reader to our previous work [7].

B. Map Consistency Estimate for the Way to Home

Given the consistency test presented above, we can perform a mathematically sound statistical test to evaluate if a map is consistent or not. However, what the robot really needs to know is not the consistency of the full map. Instead, it is sufficient to know if it can safely move along a specific path through the environment to the start location. Thus, we plan a path with A* *assuming* that the current map is consistent and extend our previous statistical consistency check to consider only the scans along that path. To achieve this, we select all recorded locations that were closer than twice the maximum sensor range away from the trajectory planned with A*. Examples of such partial maps can be found in Figure 2. The top image shows an inconsistent 2D map of the Priscilla catacomb. Directly applying our previous work [7] would label the whole map as inconsistent. In contrast to that, if the robot only takes into account the shortest route from A to B, it can still safely perform the navigation task, as shown in the middle image of the same figure. This is not the case if the robot wants to go from C to D as it will encounter an inconsistent part of the map on its way.

We make use of the statistical consistency check algorithm designed for 2D laser scans. There are two different ways of how this method can be applied in our setting. It can either be extended towards 3D data by substituting the polygons with triangulations of the full Kinect-generated 3D scan or, alternatively, the 3D scans can be reduced to a 2D scan and analyzed in a similar way to the original approach. We argue that there is no need for the more complex 3D-based approach[1] as the robot is a track-based platform that is—at least locally—operating roughly on a plane. We found that the 2D solution is well suited for the task at hand, at least for our type of environment.

On our robot, each 3D point cloud is created by combining the point clouds generated by two Asus Xtion cameras, see Figure 3. For every local 3D point (x, y, z) that is within a small margin from the height of the scan line illustrated by the red stripe in the image, we compute its bearing from the center of the robot (red arrow in the figure). The relative bearing can directly be computed through $\alpha = \operatorname{atan2}(y, x)$ and the virtual range reading by the Euclidean distance from the robot's center to (x, y). For our setup, this results in an approximate field of view of $[-\pi/4, \pi/4]$. Such virtual range scans are used for the statistical consistency check described above. A map combined from the scans generated in this manner can be seen in the Figure 2.

C. Homing by Rewinding the Trajectory

Once the consistency check has identified that the submap including the path is inconsistent, we need to perform the trajectory rewinding to bring the robot home safely. We can view the robot's forward trajectory as a series of poses of the robot $\mathrm{P} = \{p_0, \ldots, p_n\}$, where $p_i = [x_i, y_i, \theta_i]^\top$. The trajectory is expressed as a sequence of 2D positions (x_i, y_i) and orientations (θ_i) as we can only steer the robot only on a plane. The task of rewinding the trajectory is to drive the robot from p_n to p_0 and correcting for the error in odometry. The correction of the odometry error is done by aligning the sequence of point clouds that the robot perceives with the ones corresponding to $p_n, \ldots, p_0$. Note that we subsample the trajectory in such a way that each pose P is either $1\,\mathrm{m}$ away from the previous one or that there is a rotation of at least $10°$ between two poses.

Without loss of generality, let us consider that the robot has to carry out the action to move from p_i to p_j and to compensate for the error in odometry. To do so, at each time step t, the robot exploits the point cloud c_t obtained after executing the movement from p_i to p_j. In an ideal scenario, the command should have brought the robot to the pose p_j. In reality, there is an error introduced by slippage, uneven ground, etc. Thus, we align current cloud c_t with the expected one c_j. To achieve that, we use a recently proposed, highly robust variant of the ICP algorithm called NICP [11] to find the discrepancy between the point cloud that the robot *expects* to perceive and the one that it *actually* perceives. The NICP method extends the point-to-plane error metric proposed in Generalized ICP [10] by accounting not only for the metric distance between the points but also the curvature of the underlying surface. The transformation between the point clouds provided by the matching algorithm is a $\mathrm{SE}(3)$ transformation between the poses at which the two point clouds c_t and c_j were taken. As our robot moves on the ground, we project the poses onto the ground plane in order to obtain a $\mathrm{SE}(2)$ transformation. This transformation is parameterized by $\Delta p = [\Delta x, \Delta y, \Delta\theta]$. The coordinates Δx and Δy are the coordinates of the point cloud c_t expressed in the local coordinate frame defined by the position where the robot expects to arrive. Knowing the pose p_j and the local position of c_t enables us to compute the current position of the robot in the global odometry frame. Using homogeneous coordinates, this is $[x_t, y_t, 1] = T_j[\Delta x, \Delta y, 1]^\top$, where T_j is a transformation matrix:

$$T_j = \begin{bmatrix} \cos\theta_j & -\sin\theta_j & x_j \\ \sin\theta_j & \cos\theta_j & y_j \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

In addition to a 2D pose, we find the orientation of the robot θ_t as a sum of θ_j and $\Delta\theta$ normalized to $(-\pi, \pi]$. We use this new pose p_t to generate a motion command to reach the next pose chosen from the recorded trajectory. We continue this process until the robot is within $d_{\max} = 1\,\mathrm{m}$ from the starting pose p_0.

Note that our method relies on matching point clouds, typically seen from similar view points, i.e., no global search. The vanilla ICP algorithm may converge to a local minimum while performing the optimization of the objective function. This usually happens in either very cluttered environments (where the objective function has very high variation with multiple local minima) or, on the contrary, in the ones that are very feature-scarce (few distinct very narrow local minima). We found that using the NICP variant of ICP avoids such shortcomings in most practical situations, particularly when dealing with small view point changes, as is the case for our homing strategy. Refer to the work of Serafin and Grisetti [11] for a robustness analysis.

IV. Experiments

The evaluation is designed to illustrate the capabilities of our approach and to support the claims made in this paper. Our *two main claims* are that (i) the map consistency check works on the virtual 2D scans and is able to evaluate map consistency in an online fashion and (ii) we can rewind trajectories in case of failures of the mapping system and yields a homing behavior not reliant on a map. All experiments have been conducted based on real world data and on a real robot. All components are ROS nodes and operate on the notebook computer installed on the robot.

The robot is controlled through an own navigation system that uses the ROS communication infrastructure. Its SLAM system is a pose graph-based system [5] that aligns the depth images from the cameras. The optimization is done with

[1] We refer only to the consistency check and not to the SLAM system, which takes into account all six degrees of freedom.

g^2o and loop closure hypotheses are generated based on the similarity of local maps stored in the nodes of the graph.

In terms of the persistent data structure that is used to store all the information, we use a generalization of a pose graph. Each node in the graph corresponds to a pose of the robot at time t. In addition to that, each node stores the original odometry pose p_t and the corresponding 3D point cloud c_t from the RGB-D cameras. To efficiently handle this data structure even for large environments, the pose graph with the nodes p_t itself is kept in memory but the corresponding point clouds c_t are stored on disk and are loaded on demand.

The data is gathered using a setup with two Asus Xtion RGBD cameras. Both cameras point forward, one slightly rotated to the left and the other one to the right with a minimal overlap in the middle. The robot computes the point clouds and generates 2D scans from the point clouds for the consistency check as described above.

First, Figure 2 illustrates an example of the statistical map consistency check performed on the real data (virtual 2D scans from Kinect-generated point clouds) from the Priscilla catacomb in Rome. The partial maps computed around the shortest path are usually substantially smaller than the map of the whole environment, especially if the environment has multiple alternative branches and forms a complicated network of corridors or rooms. This is often the case in catacombs or underground mines. Testing smaller maps results in a significant speed-up of the statistical consistency evaluation procedure. The timings for the maps presented in Figure 2 are as follows, top map: $2.93\,\mathrm{s}$; middle map: $0.14\,\mathrm{s}$; bottom map: $0.17\,\mathrm{s}$. The maps in the figure contain, respectively, 1101, 137 and 164 different scans. The computational time depends on the number of scans to analyze and the gain in speed grows with the reduction in size of the tested map, and with the reduction of overlap between the scans. We performed the map consistency test on 5 different maps recorded in real catacombs and the consistency check always generated correct results. In sum, we prove that it is feasible to test a map in less than $200\,\mathrm{ms}$, and therefore also to execute the algorithm online. Furthermore, if needed, most of the computations can be cached during navigation. This is the case when dealing with huge maps. In such instances, the test would require a recomputation only if the SLAM back-end changes the configuration of the pose graph substantially.

As a second step, we need a robust method to return the robot to the starting location if the statistical consistency check claims the map to be inconsistent. Thus, the next set of experiments is designed to backup our second claim, i.e., that our approach can robustly rewind trajectories yielding a homing behavior that does not rely on a map. We evaluate the ability of our approach to rewind the trajectory by carrying out 20 experiments in our lab environment as well as 10 experiments in a catacomb-like, man-made cave in Niederzissen near Bonn. The latter consists of long tunnels and several small room-like structures.

In Figures 4-7 we dipict different tracks from our homing procedure. The original odometry measurements from the *forward path* are drawn in black (hardly visible as the red trajectory overlays it). The red line illustrates the subsampled trajectory that the robot has selected as its sequence P for rewinding the trajectory. Both trajectories overlay because the robot does not use any map (it would be inconsistent) and relies solely on the observations and poses it recorded on the forward path to navigate back. The blue line shows the poses from P (poses on the red line) after the alignment by NICP, thus yielding an estimate of the robot's real position in the odometry frame.

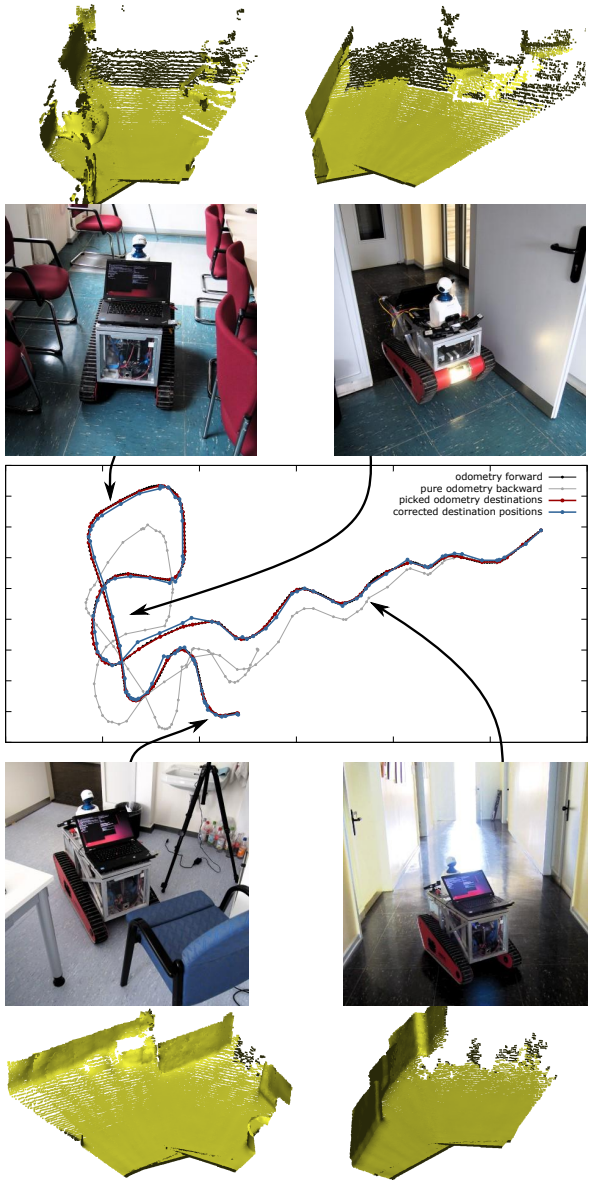

Fig. 4. Illustration of rewinding the trajectory through the office environment. The robot is steered from the bottom "tail" of the depicted trajectory to the upper-right one. The black line denotes the odometry poses saved while the robot is steered, gray denotes the odometry on the way back, red shows the temporary destination poses picked from the odometry and blue shows the same poses after the ICP correction. The pictures show several example locations visited by the robot. These feature tight doors to rooms as well as feature-scarce corridors.

One of the lab experiments is illustrated in Figure 4. Here,

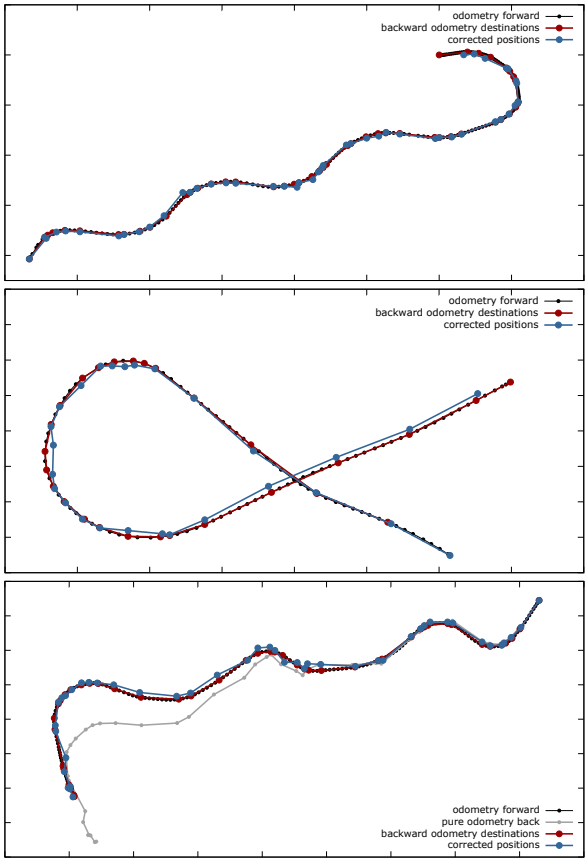

Fig. 5. Three experiments performed in different settings. The meaning of the lines is the same as in Figure 4. The average deviation from the original trajectory is between 4 cm (top dataset) and 6 cm (middle dataset).

we steered the robot on a trajectory through an obstacle parkour containing narrow passages as well as areas with numerous flat walls. The robot activated the "trajectory rewind" behavior after we manually introduced a fault in the SLAM system (by adding incorrect constraints). Consequently, the robot followed the way in reverse order using pair-wise point-cloud aligment with the NICP-based pose correction. Three examples are illustrated in Figure 5. In all 20 experiments, we varied the trajectory as well as the obstacle parkour and we did not encounter any failure of the homing behavior.

We also executed the same system in the cave of Niederzissen, see Figure 6 and Figure 7. Here, the floor is covered with dust and dirt and it is quite uneven, which causes substantial track slippage and a comparatively poor odometry. Even under such conditions, the robot is able to rewind the trajectory as illustrated in Figure 6. Again, we varied the trajectory 10 times, a second longer experiment is shown in Figure 7 and we were always able to robustly drive the robot back to the start location.

As can be seen from the close similarity between the rewinding and corrected trajectories in Figures 4-7, our approach ensures that the robot is able to accurately rewind its trajectory. The deviation of the rewinding trajectory (blue line) from the original trajectory (black line) is approximately 5 to 7 cm in the shown datasets and is largely caused by the imprecise motion command execution and strong track slippage. The gray line depicts the pure odometry measurements recorded while rewinding the trajectory. From the gray trajectory, we can see that the matching of observations must be taken into account—otherwise, the robot would deviate substantially from the reference path (and would collide with walls and obstacles).

Overall, our evaluation suggests that our robot is able to rewind its trajectories through different environments. While rewinding the trajectory, the robot moved backwards most of the time and thus it cannot observe obstacles on the path before it fully passed them. Only by following the reference trajectory precisely, the robot can return. We illustrated here that even when there is a substantial amount of track slippage on the ground, our method is still capable of robustly handling corridor-like environments with multiple narrow passages such as the doorways or narrow winding corridors.

V. Conclusion

The ability to robustly operate without user intervention is an important capability for exploration robots and safely returning after the mission is a key requirement in real-world settings. In this paper, we presented a complete homing system that addresses the problem of returning a robot operating in an unknown environment to its starting position even if

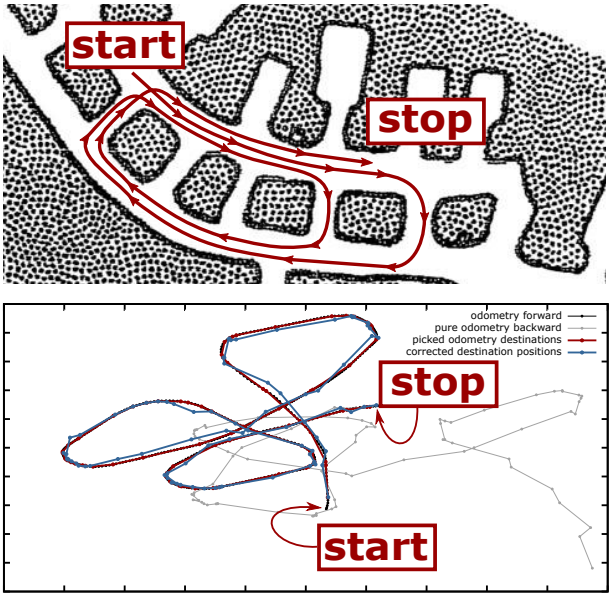

Fig. 6. An experiment in a man-made cave in Niederzissen near Bonn that consists of long tunnels and several small room-like structures. The narrow passages allow for approx. 10 cm margin for fitting a robot. The schematic drawing (top image) shows the approximate robot path drawn on top of a map. The bottom image shows the actual movement of the robot in the odometry frame while being steered from "start" to "stop" (black lines) as well as the waypoints the robot has chosen returning from "stop" to "start". The blue lines show the positions of the robot reported after ICP registration. The gray lines show pure odometry while performing homing.

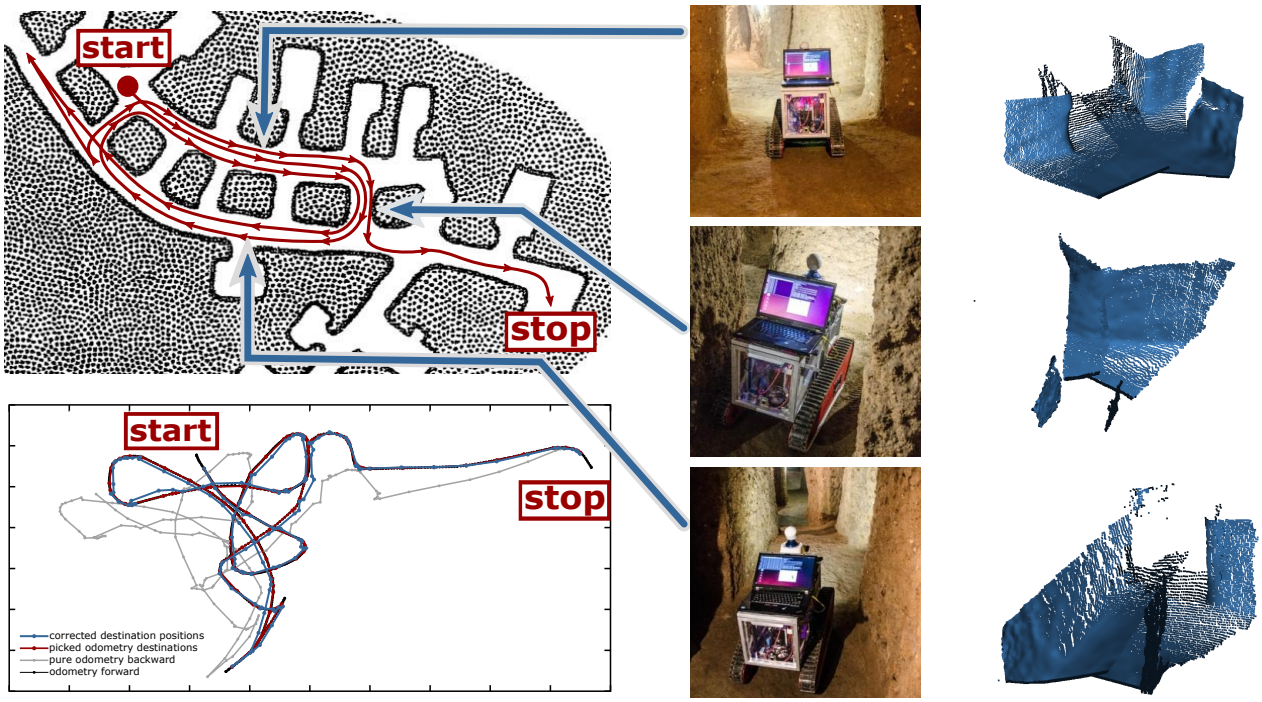

Fig. 7. Second experiment the cave as in Figure 6. The robot was steered from "start" to "stop" and has performed homing autonomously from "stop" to "start". In the left corner of the top picture, the robot was steered forward and then backwards. It has repeated the same path during the homing route as can be seen in the bottom of the second image. The length of the trajectory in this experiment is approx. 70 m.

the underlying SLAM system fails. We combined a statistical map consistency test with a scan alignment approach to rewind a previously taken trajectory. The approach can be executed online on a notebook computer with two Asus Xtion RGBD cameras recording 3D point cloud information. We implemented our approach in ROS and executed it on a real autonomous robot designed to explore and map hard-to-access underground environments. We evaluated the consistency check in our lab environment as well as on real-world data acquired in the Priscilla catacomb in Rome and in a cave in Nierderzissen near Bonn. The evaluation suggests that our approach is well suited for homing in situations in which the mapping system failed. We illustrated that our proposed method can accurately rewind trajectories guiding the robot back to its starting location. It does so in challenging real-world settings in which the robot has to navigate through narrow passages and over uneven ground.

References

[1] H. Carrillo, P. Dames, V. Kumar J.A., and Castellanos. Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

[2] P. Furgale and T. Barfoot. Stereo mapping and localization for long-range path following on rough terrain. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 4410–4416, 2010.

[3] P. Furgale and T.D Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal on Field Robotics*, 27(5):534–560, 2010.

[4] P. Furgale, P. Krüsi, F. Pomerleau, U. Schwesinger, F. Colas, and R. Siegwart. There and back again-dealing with highly-dynamic scenes and long-term change during topological/metric route following. In *ICRA14 Workshop on Modelling, Estimation, Perception, and Control of All Terrain Mobile Robots*, 2014.

[5] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Transactions on Intelligent Transportation Systems Magazine*, 2:31–43, 2010.

[6] A.A. Makarenko, S.B. Williams, F. Bourgoult, and F. Durrant-Whyte. An experiment in integrated exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[7] M. Mazuran, G.D. Tipaldi, L. Spinello, W. Burgard, and C. Stachniss. A statistical measure for map consistency in slam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

[8] M. Nitsche, T. Pire, T. Krajnik, M. Kulich, and M. Mejail. Monte carlo localization for teach-and-repeat feature-based navigation. In *Advances in Autonomous Robotics Systems*, volume 8717 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2014.

[9] C.J. Ostafew, A.P. Schoellig, and T.D. Barfoot. Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 176–181, Nov 2013.

[10] A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Proc. of Robotics: Science and Systems (RSS)*, volume 2, 2009.

[11] J. Serafin and G. Grisetti. NICP: Dense normal based point cloud registration and mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015. Currently under review.

[12] R. Sim and N. Roy. Global a-optimal robot exploration in slam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Barcelona, Spain, 2005.

[13] C. Sprunk, G.D. Tipaldi, A. Cherubini, and W. Burgard. Lidar-based teach-and-repeat of mobile robot trajectories. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[14] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robotics: Science and Systems (RSS)*, pages 65–72, Cambridge, MA, USA, 2005.

[15] D. Perea Ström, I. Bogoslavskyi, and C. Stachniss. Robust exploration and homing for autonomous robots. *Robotics & Autonomous Systems*. Conditionally accepted for publication.

[16] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proc. of the Int. Conf. on Autonomous Agents*, pages 47–53, 1998.