
EU Horizon 2020 Research & Innovation Program

Advanced Robot Capabilities & Take-Up

ICT-25-2016-2017


# Safe Robot Navigation in Dense Crowds

http://www.crowdbot.org

# Technical Report

# D 3.2: Robust Localization & Mapping

Work Package 3 (WP 3)

Navigation

Task Lead: Swiss Federal Institute of Technology (ETHZ), Switzerland

WP Lead: Swiss Federal Institute of Technology (ETHZ), Switzerland

# Table of Contents

# Table of Figures

## Executive Summary

This report details the robust localization and mapping algorithms developed for the Crowdbot project between months M1 and M30. Our proposed solutions are designed with the explicit goal of achieving robot navigation in *crowded* environments, where many existing methods struggle due to the high degree of dynamic motion around the robot. This report primarily serves as an update on Crowdbot D31 First Release of Localization, Mapping and Local Motion Planning, which covers the work carried out in months M1 to M20. Specifically, we build on Sections 3 and 4 of the previous deliverable and include relevant algorithmic details from D31 of the Crowdbot mapping and localization algorithms to ensure the completeness of the report. As stated in D31, the main challenges that we address are:

- Generating clean and coherent maps of the static environment despite the presence of dynamic obstacles during mapping;
- Achieving fast, and accurate localization when prior information on the robot pose is unavailable.

The main improvements to our localization and mapping strategy from D31 are:

- Explicit removal of LIDAR points that return from pedestrians around the robot, identified using the pedestrian detection algorithms provided by RWTH, prior to map generation, and
- Refinement of our localization precision through algorithmic improvements to our branch and bound search technique that enable maintaining real-time operation.

The updates to our localization solution have been implemented in our open source library at https://github.com/danieldugas/map_matcher.

# 1. Introduction

Safe navigation through dense crowds necessitates robust localization and mapping techniques that are able to cope with highly dynamic environments. Unlike in static environments, where all detected landmarks can be used for localization, the CROWDBOT target scenarios feature densely occupied dynamic environments. This raises the difficulty of achieving robust and accurate mapping and localization since the majority of the robot's field of view may be occluded by other traffic participants, while simultaneously increasing the need for precise solutions to guarantee safe navigation as the robot will be in close proximity to humans.

Our developed framework for mapping and localization in challenging crowded environments is modular and can be adapted to different robot setups, provided that similar sensor arrangements (ankle- to knee-height range measurements, such as from a LIDAR) are available cross-platform. Our solution also exploits the person detection and tracking algorithms developed in CROWDBOT WP2 to filter detected humans from the robot sensor returns and enable robust mapping and localization even in densely crowded environments. In Section 2 we provide details on the CROWDBOT mapping approach and our map matching localization algorithm is described in Section 3. Thorough evaluations of each module are provided in the corresponding sections.

# 2. Robust Mapping in Crowded Environments

Here we focus on two strong requirements for robust motion planning: 1) accurate mapping and 2) localization. Many well-developed approaches exist which allow solving these two objectives in mostly-static environments, despite sensor and odometry noise. However, higher densities of dynamic objects around the robot lead to a drop in robustness and often, mapping/localization failure.

Our chosen localization method, described in D31, allows us to combine the localization and mapping objectives into one: using map matching methods for localization means that accurate mapping leads to robust localization. In this section, we present an improvement to our previous mapping technique which attempts to deal with the presence of dynamic obstacles and the resulting deterioration of map quality.

## 2.1. Effect of Crowd on Mapping

In the case of a particle filter-based LIDAR mapping method such as used by the popular gmapping[1] framework, the uncertainty in the pose update as the robot moves is corrected by evaluating the best fit between the new sensor data (laser scan) and current map. This fit is based on the assumption of a static environment, or in other terms, the assumption that the current map and latest scan are samples of the same underlying geometry. Dynamic obstacles, however, break this assumption: the movement of the obstacle implies that the underlying geometry has changed. As a result, the robot movement can be falsely estimated, increasing the distortion errors in the resulting map (see Figure 4).

Typically, once the pose update is corrected, the latest laser scan is added to the current map. Here lies the second effect of dynamic obstacles on LIDAR mapping: transient observations of dynamic obstacles in the latest scan end up etched into the current map, which is supposed to represent static (or very slowly changing) geometry. In practice this can be seen through noise, falsely occupied points in areas of free space as shown in Figure 5.

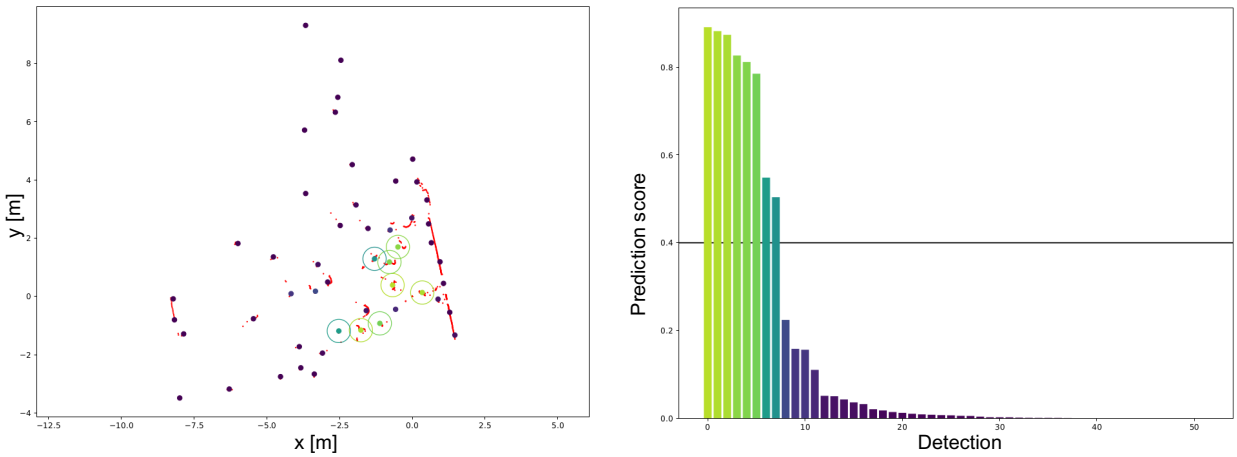
Though there are methods attempting to address these issues, by for example i) making the fit function more robust to dynamic obstacles (e.g. RANSAC), or ii) eliminate false occupancy through voting, using several scans. The method used here attempts to address the issue as early as possible in the processing pipeline: detecting dynamic obstacles using a separate state-of-the-art learned classifier, and using these detections in preprocessing to clear LIDAR returns due to dynamic obstacles in sensor observations. This semantic understanding has the advantage of allowing removal of dynamic obstacles which are temporarily static, such as humans standing still, from the map.

In the scenarios targeted by the Crowdbot project, an overwhelming majority of dynamic obstacles are humans using light form of transport (walking, cycling, wheelchair use, etc.), therefore, we focus our semantic classification effort on detecting the presence of humans around the robot.

## 2.2. Semantic LIDAR Scrubbing

Several sensors are available in order to detect persons in the robot's vicinity. Due to the stated goal of preprocessing LIDAR scans, it would be ideal to perform classification directly on LIDAR data. In our mapping routine, we have integrated the state-of-the-art person detection algorithm DR-SPAAM [1], which was developed by RWTH Aachen as part of CROWDBOT WP2. Details of the DR-SPAAM algorithm can be found in their paper [1].

[1] http://wiki.ros.org/gmapping, (Accessed on 20.08.2019)




(a) (b)

**Figure 1. DR-SPAAM human classification predictions for a single LIDAR scan on a real dataset recorded for the CROWDBOT project. (a) Circled detections lie above the fixed threshold of 0.4, the lighter the circle, the more confident the human classification prediction. (b) All detections ordered by prediction score.**

Human classification predictions output by the DR-SPAAM framework are selected as positive if above a constant threshold of 0.4, as suggested by the authors. A circle of radius 0.3 is centered at the x-y coordinates of the detection in the LIDAR sensor frame, and all LIDAR points falling within this circle are scrubbed.

In order to correct for missed detections, we use a naïve caching approach, where the circle scrubbing from detections on a LIDAR scan are also applied to the next few frames. This results in more consistent yet more aggressive scrubbing. An example of the scrubbed LIDAR scan from Figure 1(a) is shown in Figure 2, where the scrubbed LIDAR points are colored in grey.

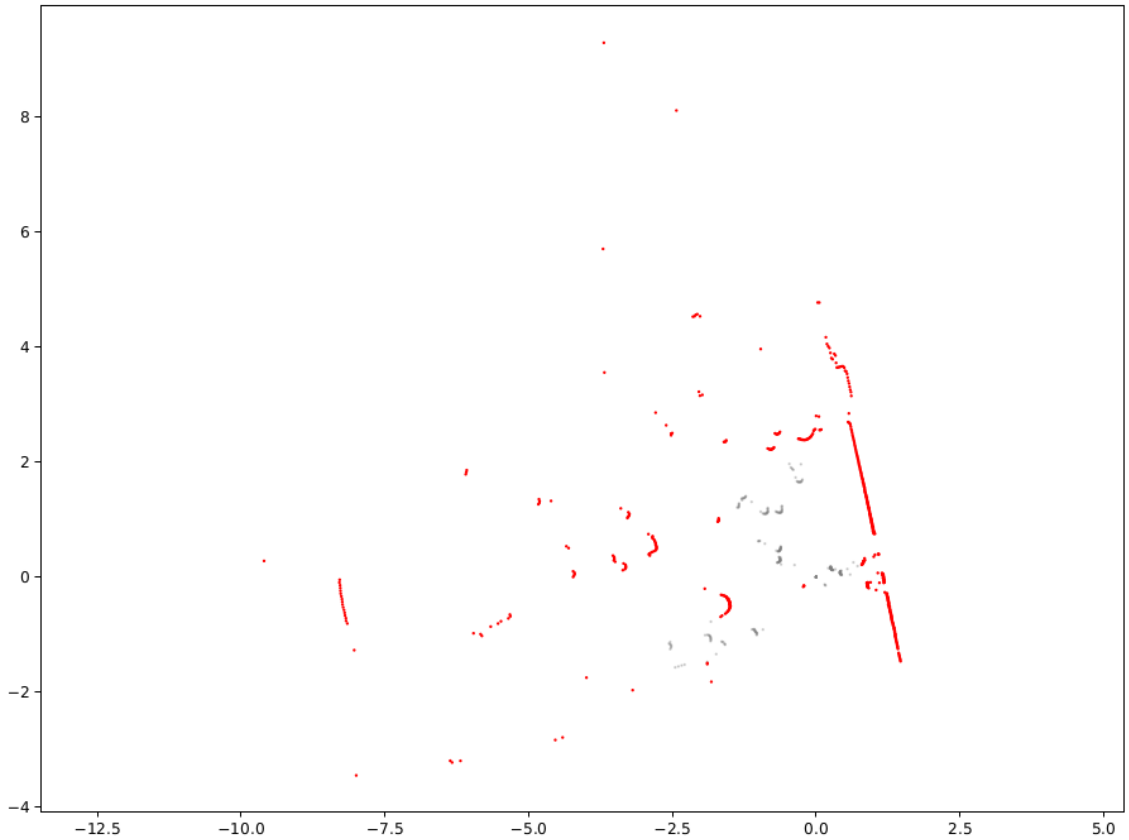


**Figure 2. Resulting scrubbed LIDAR scan, scrubbed LIDAR returns colored in grey.**

## 2.3. Results and Analysis

To evaluate the effect of LIDAR scrubbing on online mapping, we performed comparisons of the gmapping performance with and without scrubbing. We tested on rosbag data collected during our open lab day event which attracted over 500 attendees. As such, the test environment included crowd densities in excess of 2p/$m^2$ with a mixture of static crowds (people standing around the main open spaces to look at exhibits) and dynamic pedestrians (moving along the corridors between the open spaces). Figure 3 shows the two occupancy maps generated without (original) and with (scrubbed) person removal in the LIDAR scans.

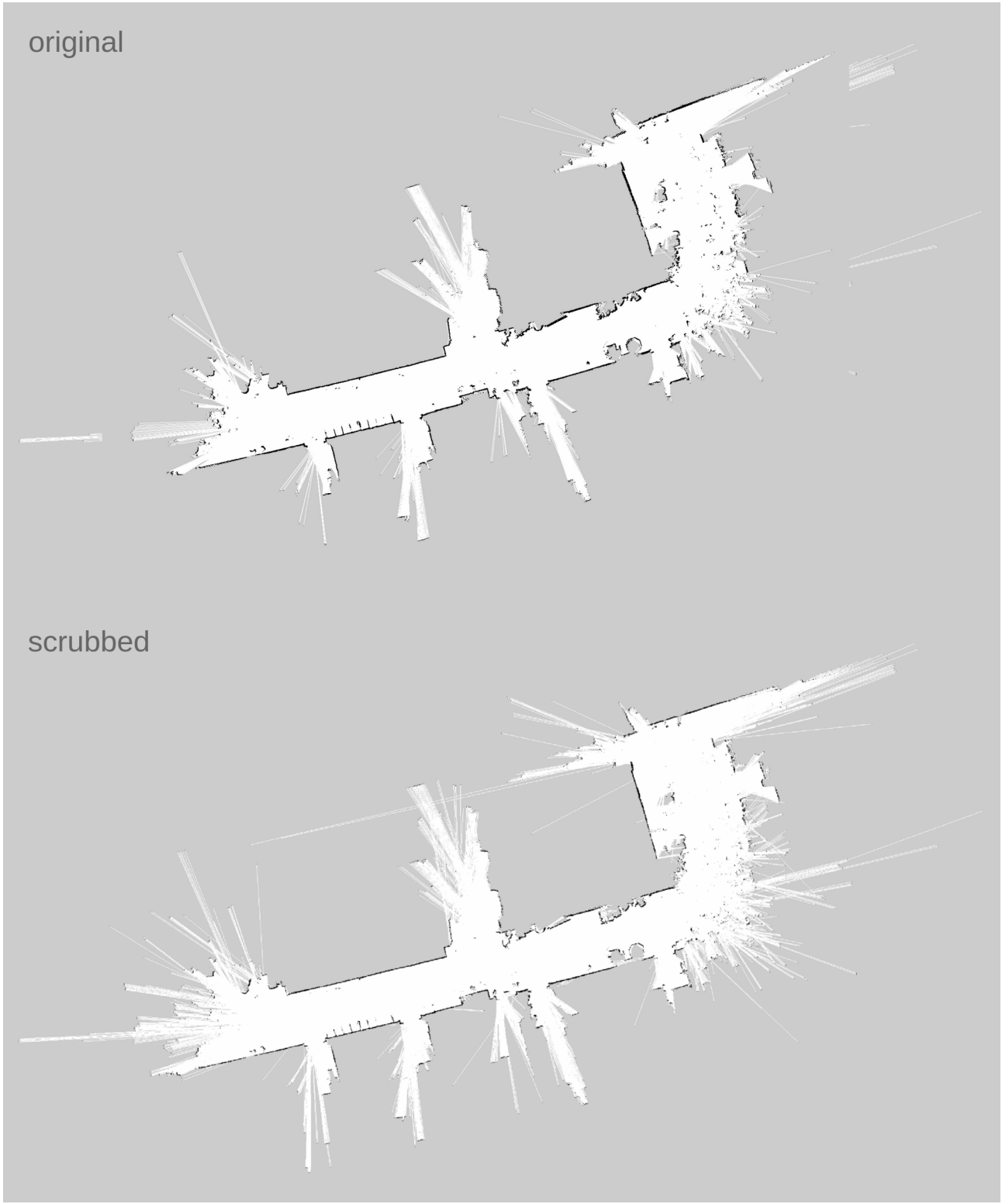



**Figure 3. Original (top) and scrubbed (bottom) occupancy maps.**

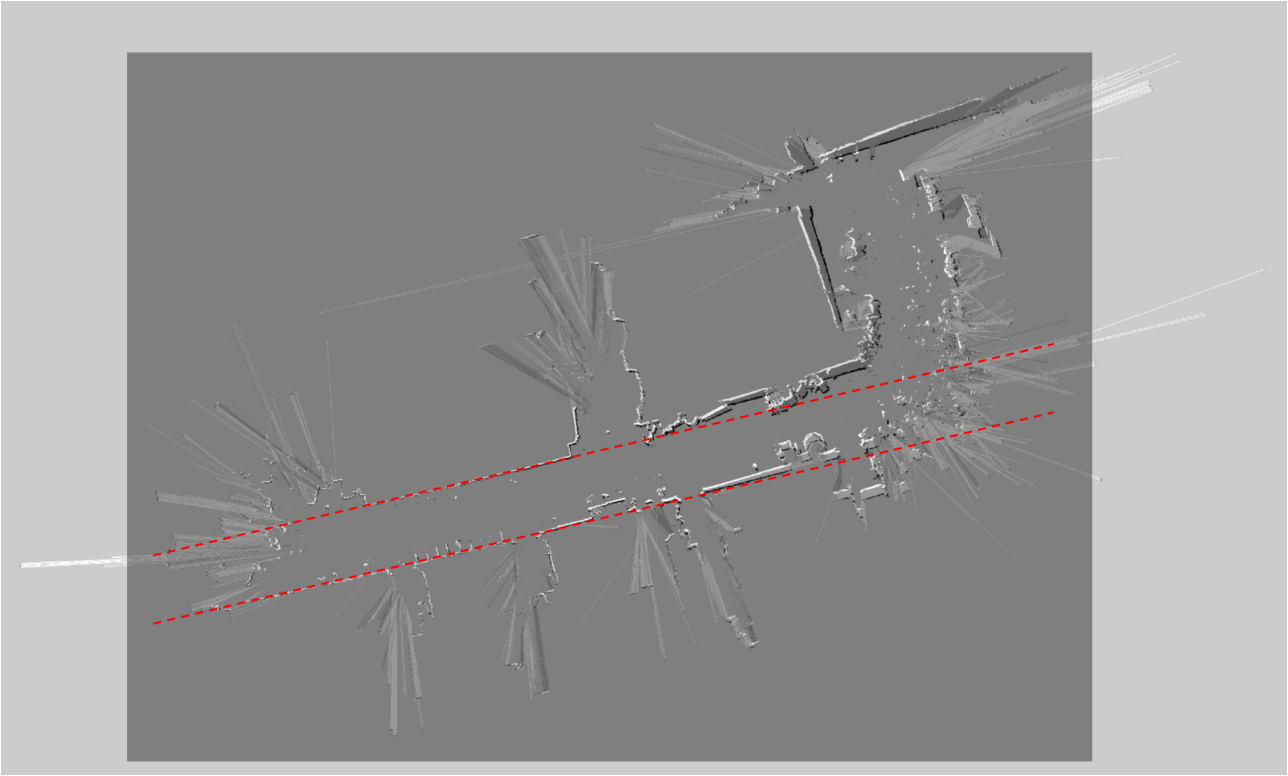


**Figure 4. Comparison of distortion effect between original (white) and scrubbed (black) occupancy. Red straight lines are added for reference.**

(a) (b)

**Figure 5. Comparison of the false-occupancy effect for original (a) and scrubbed (b) maps.**

Figure 4 overlays the original and scrubbed maps to highlight the benefits of removing people from the LIDAR scans before sending the sensor data to the gmapping. Dashed red lines are also drawn in to provide a reference for the parallel walls of the corridor. Without scrubbing, occlusions from moving people in the LIDAR scans results in substantial rotational distortion of the map. Figure 5 zooms in on a portion of the map in the open area where crowd density was especially high. It shows that semantic scrubbing has a positive outcome for both the distortion and false-occupancy effects mentioned in Section 2.1. However, neither effect is completely negated, in part due to the imperfect semantic detection, which leads to false positives and true negatives in the LIDAR scrubbing process.

# 3. Robust Localization in Crowded Environments

The focus of our localization algorithm design is to achieve fast, accurate and prior-free initial localization. Existing open source solutions, such as adaptive Monte Carlo localization (AMCL)[2], require a prior on the robot pose at initialization and demonstrate poor performance when the prior pose is poorly set. Similarly, scan matching solution such as the one used in our Crowdbot active SLAM routine described in D31 and [2] also requires a prior on the robot pose to initialize the point-to-line optimization routine.

The main methodology of our localization solution lies in finding the pixel-wise alignment of the current session map to a provided global reference map of the environment. The current session map can be constructed using any mapping algorithm, e.g. the occupancy grid mapping algorithm described [2], or the *slam_gmapping*[3] ROS node. The only requirement is that the session map resolution is the same as that of the provided global map. By performing a *map* matching routine rather than simply a *scan* matching routine, we are able to improve the accuracy of the localization solution over methods that only use the current LIDAR scan. Furthermore, our method can maintain good localization in highly crowded environments where sensor occlusions may be severe.

We implemented a branch and bound depth first search inspired by the Google Cartographer[4] package [3]. Our prior implementation described in Crowdbot D31 was resolution-limited by the search depth that could be achieved while maintaining real-time localization performance. Our latest improvements to the localization routine now enable map matching at the original resolution of the map (i.e. 2-5cm grid cells compared to 20cm grid cells in our earlier implementation) while maintaining a similar solution speed as before.

Details of our implementation are provided in the following sections followed by a quantitative analysis of the localization performance in crowded environments (both simulated and recorded from real robot trials) compared to our prior implementation.

## 3.1. Branch and Bound Map Matching

Our branch and bound routine maintains computational efficiency by searching for matches between the occupied cells of the session map and those of the global reference map at increasing map resolutions. The lowest map resolution is computed as a function of the size of the session map.

$$H = \log_2\left(\max_{k\in i,j} 2k\right) - 1, \tag{4.1}$$

where $i$ and $j$ are the dimensions of the session map and $H$ is the number of downsampling steps.

For each search query, the global reference map is first downsampled using a max operator, halving the resolution at each layer $h$ until the lowest map resolution, $h = H$, is reached. The global map at each resolution (including the original map resolution, $h = 0$) is stored for fast lookup during the search. See Figure 1 for an example of the precomputed maps at decreasing resolutions.

In addition, the occupied cells in the session map are stored for $N$ discrete rotation transformations, where $N$ is determined according to the session map size, the original map resolution and a user-provided $rotation\_downsampling$ parameter which restricts the maximum pixel translation of a single discrete rotation step. That is,

$$dr = rotation\_downsampling \cdot \frac{1}{\sqrt{i^2+j^2}}, \tag{4.2}$$

$$N = \frac{2\pi}{dr} - 1. \tag{4.3}$$

[2] http://wiki.ros.org/amcl, (Accessed on 20.08.2019)

[3] http://wiki.ros.org/slam_gmapping, (Accessed on 20.08.2019)

[4] https://google-cartographer.readthedocs.io/en/latest/, (Accessed 20.08.2019)

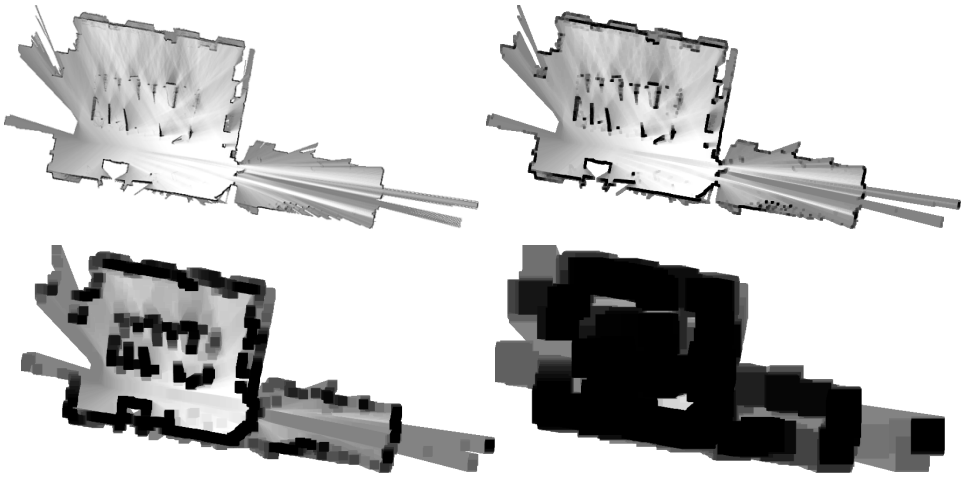


**Figure 6. Precomputed grids at decreasing map resolution. Figure from [45].**

Given the setup described above, the nodes in the search tree contain the (global map) resolution, (session map) rotation and translational information. The initial node list contains all the roto-transformations at the lowest global map resolution. An additional prior sorting according to available heading information can also be performed to further speed up the search.

Each node is scored and sorted according to the percentage of occupied cells in the rotated session map that are also deemed occupied in the current search resolution of the global map. A minimum match threshold is set initially to 50% such that any nodes that fall below this threshold are immediately pruned. Non-leaf nodes that pass this threshold score are expanded by increasing the global map resolution by one depth, effectively adding four more child nodes to the node list. Once a leaf node is reached (i.e. a node at the original map resolution) the existing minimum match threshold is updated to its node score. The search process continues until the node list is empty.

Algorithm 1 provides the pseudocode for our branch and bound depth first search routine.

**Algorithm 1: Branch and bound depth first search**

*Initialise search problem*
Compute and store $H + 1$ downsampled global maps
Compute and store $N$ rotations of session map
$node\_list \leftarrow \emptyset$
$threshold \leftarrow 0.5$
**For** $n = 1{:}\,N$
  $new\_node(H, n, 0, 0)$
  $new\_node.compute\_score()$
  $node\_list.insert(new\_node)$
**End For**

*Depth first search*
**While** ! $node\_list.empty()$
  $node \leftarrow node\_list.pop()$
  **If** $node.score < threshold$
    **continue**
  **End If**
  **If** $node.is\_leaf()$
    $threshold \leftarrow node.score$
    $best\_node \leftarrow node$
  **Else**
    $child\_nodes = node.expand()$
    **ForEach** $child$ **in** $child\_nodes$
      $child.compute\_score()$
      $node\_list.insert(child)$

25 **End ForEach**
26 **End If**
27 **End While**
28 **Return** *best_node*

## 3.2. Updates to Improve Matching Precision

Despite the efficiency of the branch and bound search routine, the algorithm still scales with the depth of the search. Thus, under real-time operating requirements, our prior implementation detailed in Crowdbot D31, was only able to search to a coarse map resolution of around 20cm×20cm. Furthermore, we used a rotation downsampling factor of 2 in equation (4.2).

We have now improved our map matching routine to enable matching at the original resolution of the map. The key step to achieving this precision is to hot-start a second branch and bound search using the initial solution provided by the low-resolution matching. This high-resolution refinement step is substantially faster than the initial search and allows matching to occur over the original map resolution, in our case up to 2cm×2cm grid cells. In addition, due to these speedups, we can also avoid downsampling the rotation resolution during this refinement stage, thus, $rotation\_downsampling = 1$ in equation (4.2).

## 3.3. Results and Analysis

### 3.3.1. Evaluations on Simulated Static Environment

We first evaluate the internal algorithm properties, that is, computation speed and breakdown as well as the map matching performance, when run under controlled conditions within the simulated static environment shown in Figure 7. The results here represent the performance of the localization algorithm under ideal environmental conditions (clean reference map, no dynamic obstacles, no obstacles with noisy reflectance properties). The robot, a Pioneer P3DX differential drive robot, is equipped with a 2D LIDAR with a 270° field of view, 20m range and noise drawn from $\mathcal{N}(0, \sigma^2)$. The sensor parameters were selected to mimic the LIDARs used on the Crowdbot Pepper robot.

We conducted several runs where in each run the robot was initialized at a different location and then tasked with exploring the map using a standard frontier exploration routine[5]. Since there are no dynamic obstacles in this test environment, we use the *slam_gmapping* ROS node to generate the session map. Here we show results from two such runs, where we set $\sigma = \{0.03, 0.05\}$, highlighting the key properties of the map matching routine with varying degrees of sensor noise.

Figure 8 shows a top down view of the localization solution at the start (top row) and end (bottom row) of exploration for $\sigma = 0.03$. The first column shows the low-resolution localization solution while the right column shows the refined solution after the second branch and bound search to the original map resolution. Matching cells are colored red. Even at the start of exploration, the robot is able to accurately localize without needing a prior on the initial robot pose in the global reference map. This reinforces the results we showed in Crowdbot D31, where we also demonstrated superior performance when compared to AMCL, which was unable to localize without accurate prior pose information.

[5] https://github.com/JenJenChung/frontier_waypoint

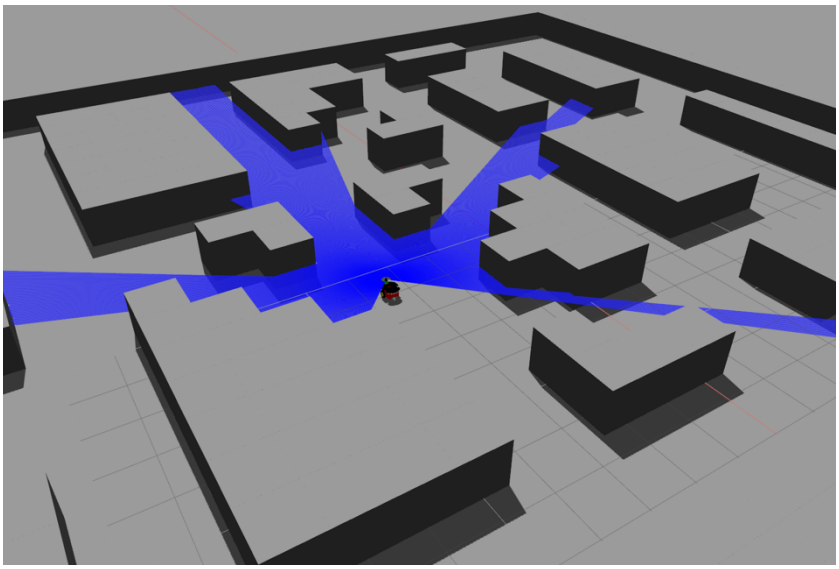
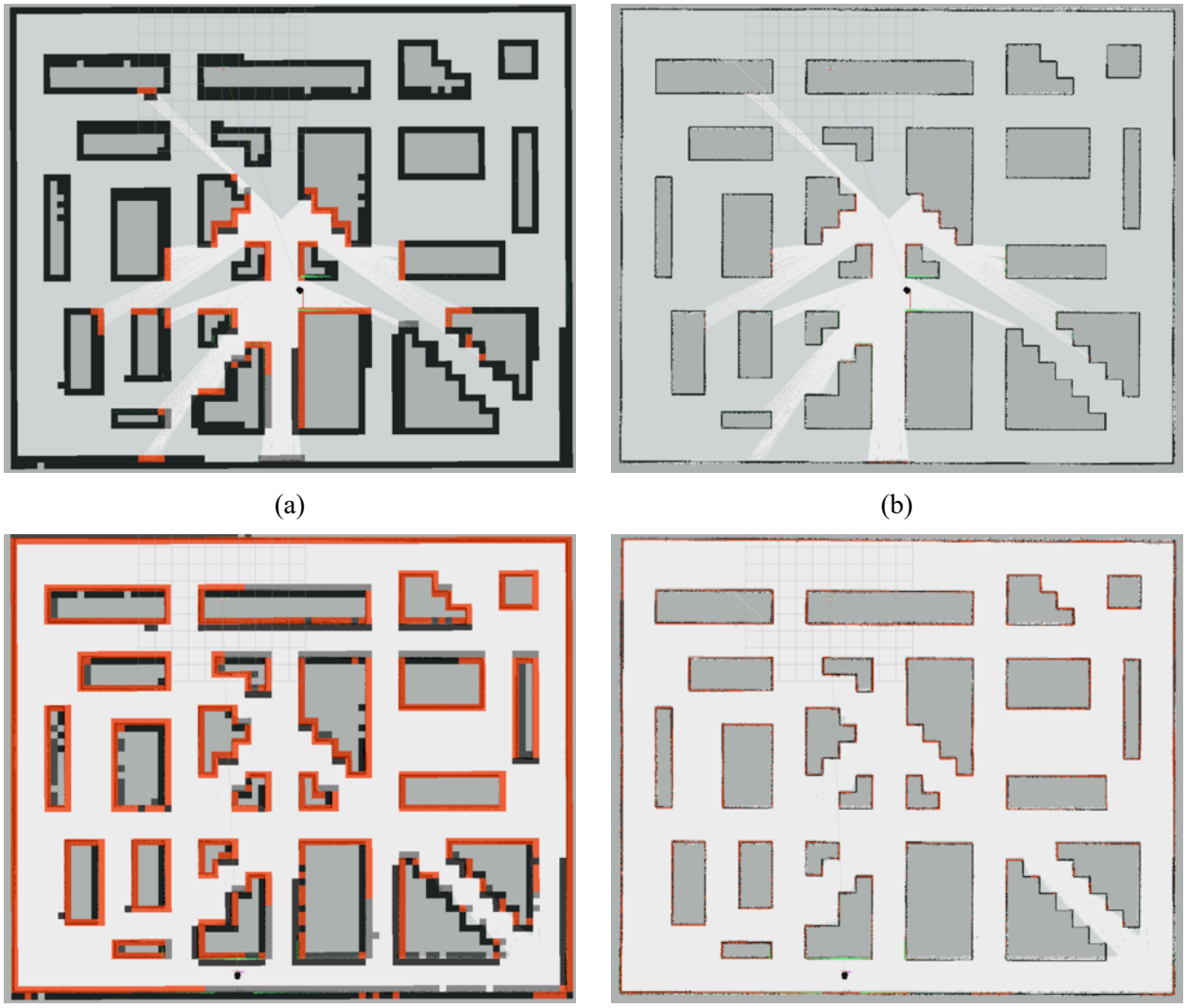


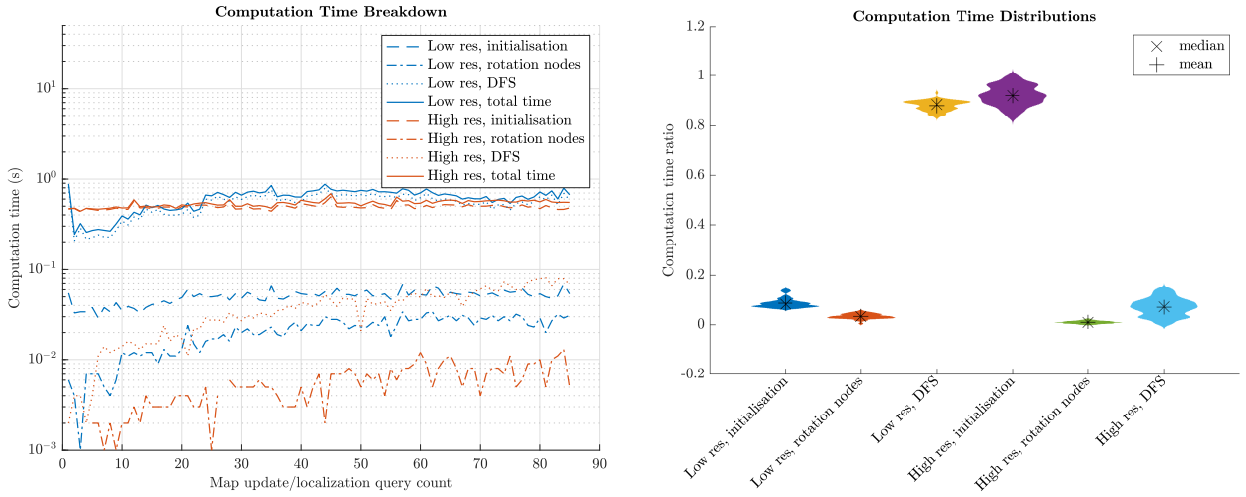
**Figure 7. Gazebo simulation environment for evaluating the internal algorithm properties of our proposed map matching localization routine.**



(a) (b)

(c) (d)

**Figure 8. Map matching progression from the start of exploration (top row) to the end of exploration (bottom row). The left and right columns show the low- and high-resolution matches highlighted in red, respectively.**

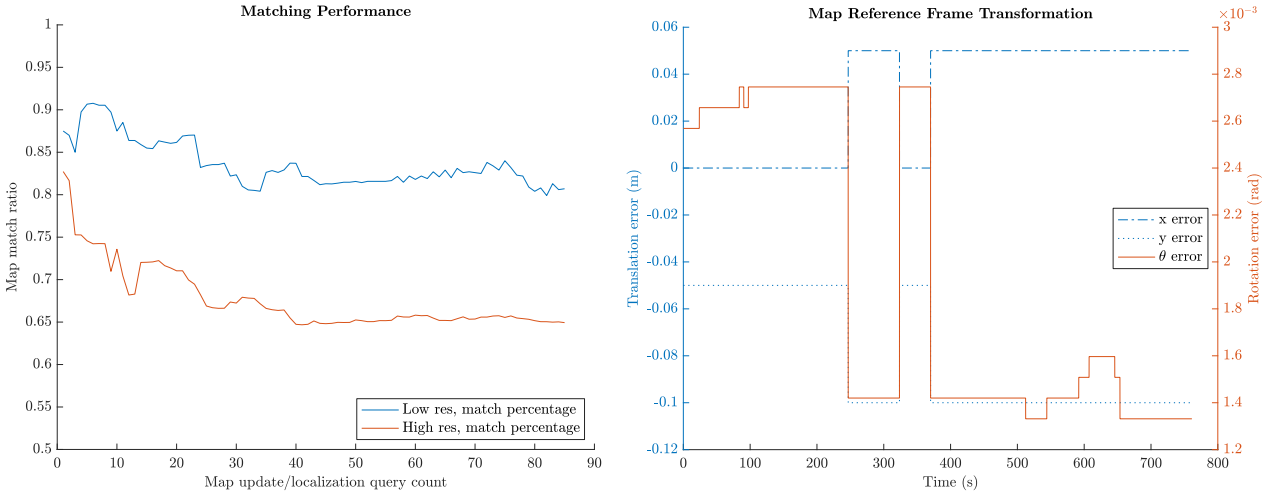



(a) (b)

**Figure 9. (a) Total computation time for each localization query as well as the breakdown into each of the main steps of the branch and bound search. (b) Distribution plots showing the proportion of computation time taken by each component of the low-resolution and high-resolution localization queries.**

Figure 9a shows the total computation time for the low-resolution and high-resolution branch and bound searches. It also breaks down the computation time into the individual components of the search routine, namely, the problem initialization, the expansion of the rotation nodes and the time taken to perform depth first search. In this run, the branch and bound search takes less than 1s for all queries (both at low and high search resolution). What is interesting to note is that for the low-resolution query, the majority of the computation time is taken by the depth first search, whereas for the high-resolution query, it is the problem initialization that takes up most of the computational effort.

Figure 9b highlights this by plotting the proportion of the total computation time taken for each component. For the low-resolution search, fewer maps need to be computed and stored, however the search needs to take place over all possible translation and rotations. In the high-resolution search, the opposite occurs, where larger maps must be stored (since they each at a higher resolution) yet the search is constrained by the solution from the initial low-resolution search.



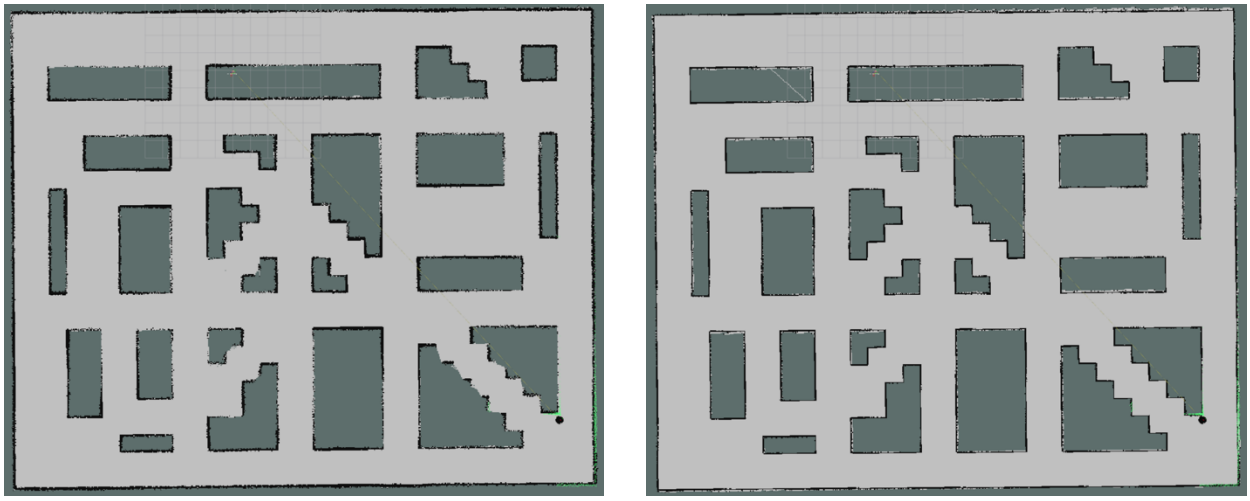

(a) (b)

**Figure 10. Map matching performance (a) computed as percentage of matching occupied cells. (b) Difference between the current session map and global map reference frames.**

Figure 10 plots the map matching performance over the run. A match acceptance ratio of 0.5 was used for the low-resolution queries while a ratio of 0.1 was used for the high-resolution queries. Note that since the high-resolution refinement is already constrained by the low-resolution match, a lower acceptance ratio can be used at the refinement stage to further reduce computation time without substantial risk to the final solution quality.

If no solution is found during the refinement stage, the algorithm reverts to the map match found by the low-resolution search. Figure 10a, shows that the ratio of matching occupied cells for the low-resolution search starts out around 0.87 and stabilizes at around 0.80. Similarly, the high-resolution refinement, starts out at around 0.83 and converges to approximately 0.65, with no significant drop in matching performance after query 40. Figure 10b plots the transformation between the current session map and the global reference map over the course of the full run (just under 800s). This plot shows that the two maps remain aligned within 5cm (x translation error), 10cm (y translation error) and with negligible differences in rotation after over 12 minutes of exploration.

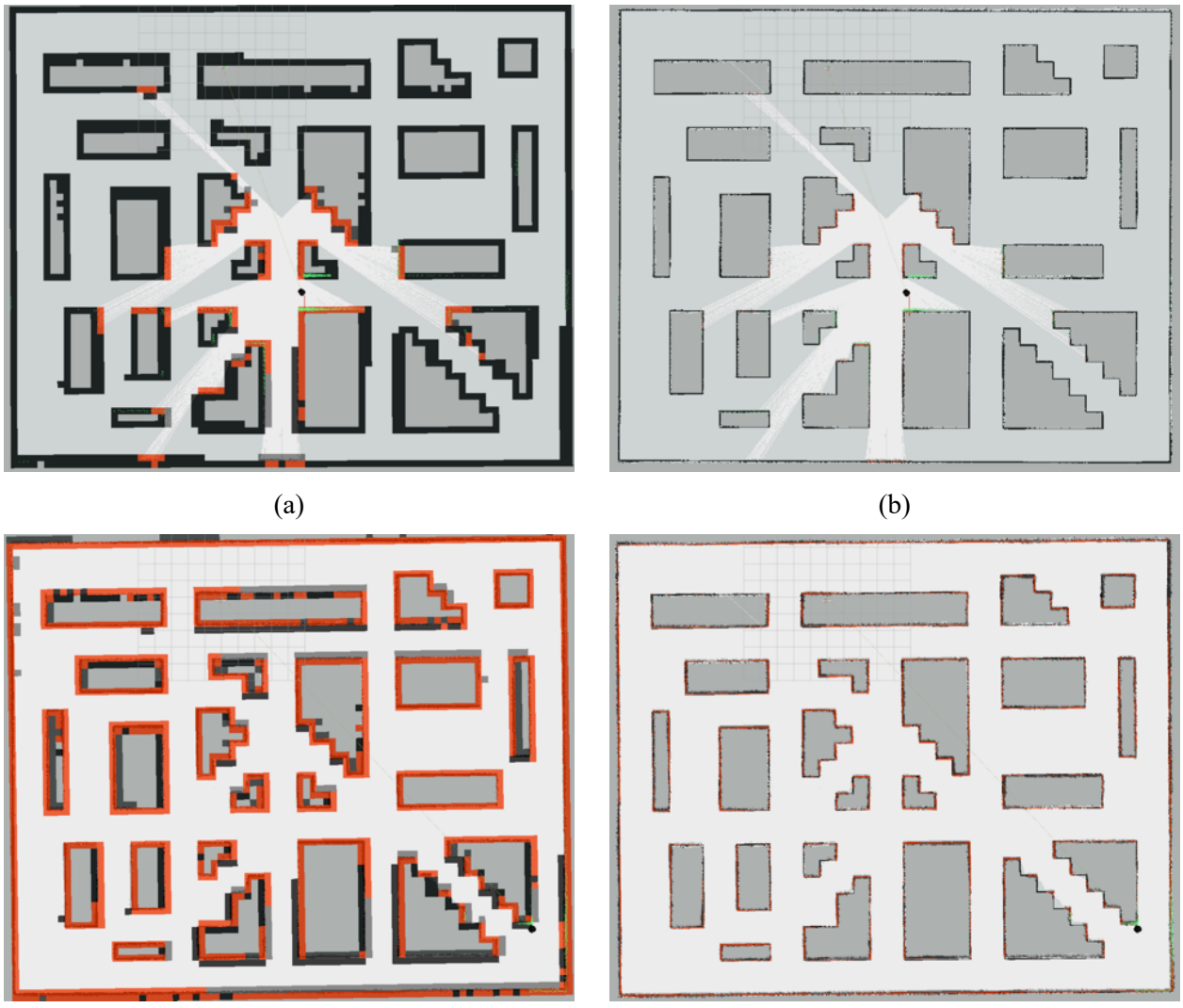
It is important to also investigate the robustness of our map matching localization routine with respect to the sensor noise. Sensor noise influences the matching performance through the construction of the current session map. If the session map is noisy, this may lead to longer computation times in the search as well as poorer overall matching quality. Figure 11 compares the quality of the session map and the global reference map, which were constructed using $\sigma = 0.05$ and $\sigma = 0.01$, respectively.



(a) (b)

**Figure 11. Comparison of the map quality with varying sensor noise. In (a), the map is constructed using a sensor with noise drawn from $\mathcal{N}(0, 0.05^2)$, whereas the global reference map in (b) is constructed from a sensor with noise drawn from $\mathcal{N}(0, 0.01^2)$. In some places the walls are much thicker in (a), spanning several grid cells, while other regions have many more gaps. In contrast, the map in (b) is much cleaner, more complete and with fewer artifacts.**

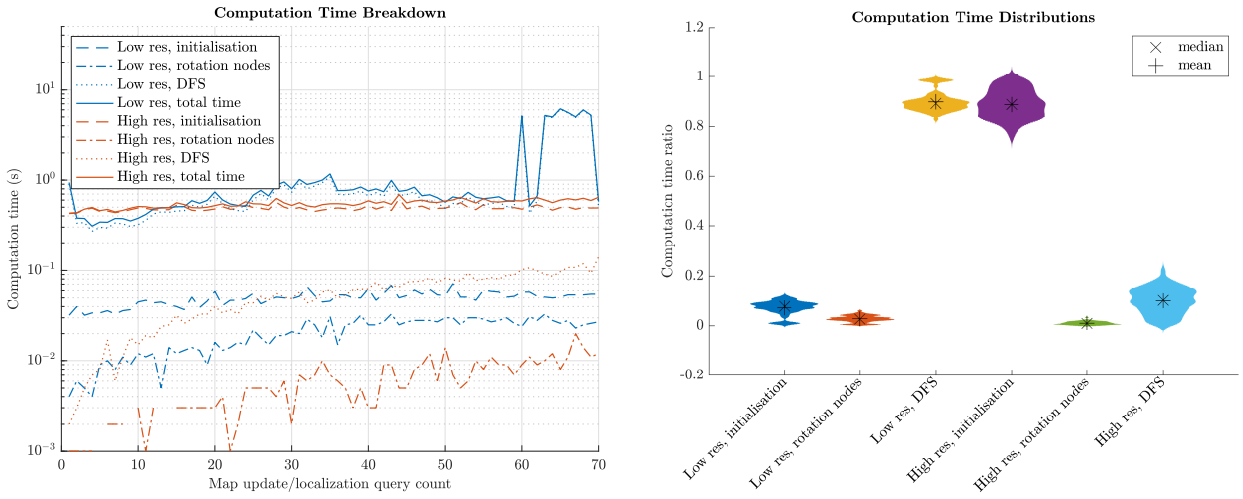
Figures 12-14 show the same metrics evaluated on a run where we increase the sensor noise such that it is drawn from a standard deviation of $\sigma = 0.05$. While the ratio of matched cells in the high-resolution refinement is generally lower than in the previous case, there is no substantial reduction in localization quality as shown in Figure 13b where the transformation between the two map reference frames remains bounded over the 12-minute exploration. The computation time for the branch and bound search also remains less than 1s for the majority of queries, with only a handful of low-resolution search queries requiring up to 6s.



(a) (b)

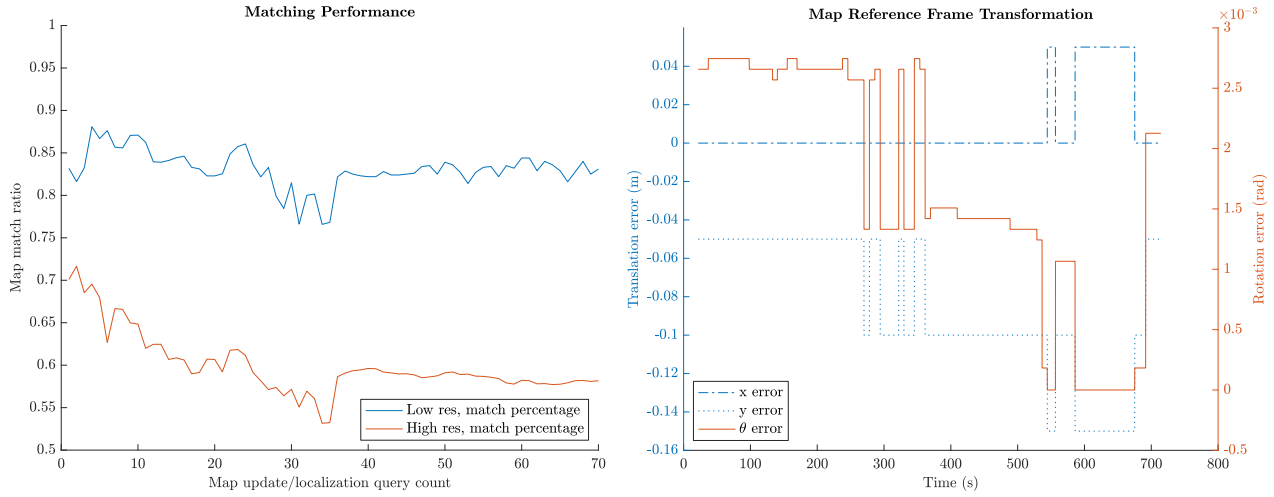
(c) (d)

**Figure 12. Map matching progression from the start of exploration (top row) to the end of exploration (bottom row). The left and right columns show the low- and high-resolution matches highlighted in red, respectively.**




(a) (b)

**Figure 13. Computation time of the map matching routine. (a) Total time for each query, also broken down into each of the main steps of the branch and bound search. (b) Distribution plots showing the proportion of computation time taken by each component of the localization queries.**



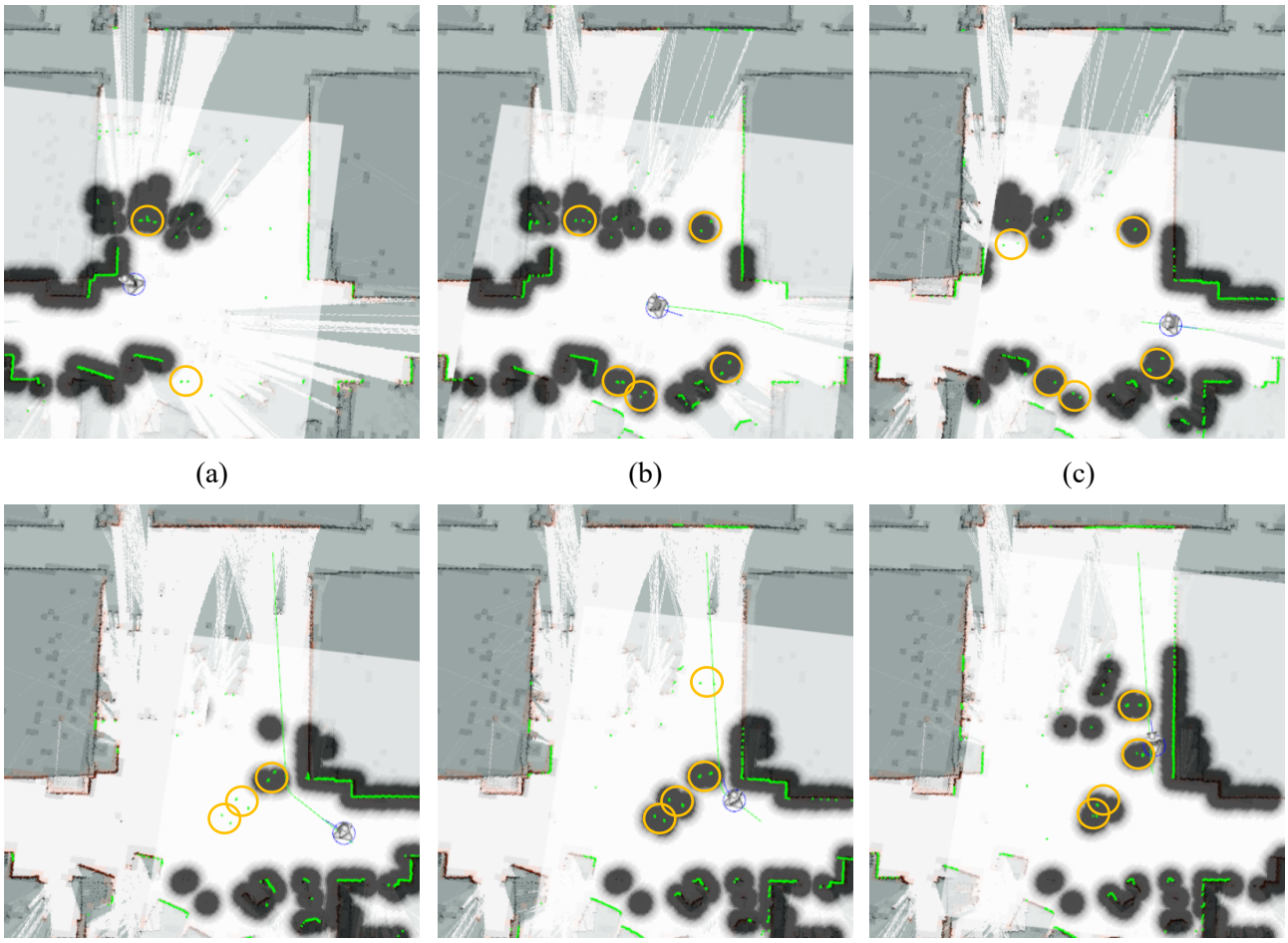

(a) (b)

**Figure 14. Map matching performance (a) as number of matched cells, and (b) as an absolute transformation between the two reference frames.**

### 3.3.2. Evaluations on Simulated Noisy, Crowded Environments

The following set of results demonstrate the localization performance in an exemplar Crowdbot environment. The simulation environment is created from a 2cm×2cm resolution map of the ETH ASL office that was generated by teleoperating the Crowdbot Pepper robot around the physical office space. The map contains various artifacts due to noise inherent in the LIDARs as well as from elements in the environment such as reflections from glass or metallic objects. These artifacts were kept in the map such that we could accurately simulate the noisy sensing that the robot would experience in a real-world setting. Note that the map is also of a finer resolution than that used in the previous set of experiments and so by comparing these results we can investigate how the multi-step map matching algorithm scales with the size of the search space.

In this set of experiments, we also simulate several pedestrians in the static map as pairs of "legs" that move according to a sinusoidal gait pattern. The pedestrians interact with the robot as it navigates through the environment such that the robot may be blocked by their presence. Figure 15 highlights the simulated pedestrians in the LIDAR scans across several consecutive frames as the robot navigates from its starting position in the bottom left of the map to the bottom right and then to the top of the map. We conducted ten trials, each lasting approximately 5 minutes, with the robot starting in different locations in the map, with different numbers of pedestrians and commanding different sets of waypoints around the environment. Here we present one case study whose results are indicative of those observed in each of our tested cases.



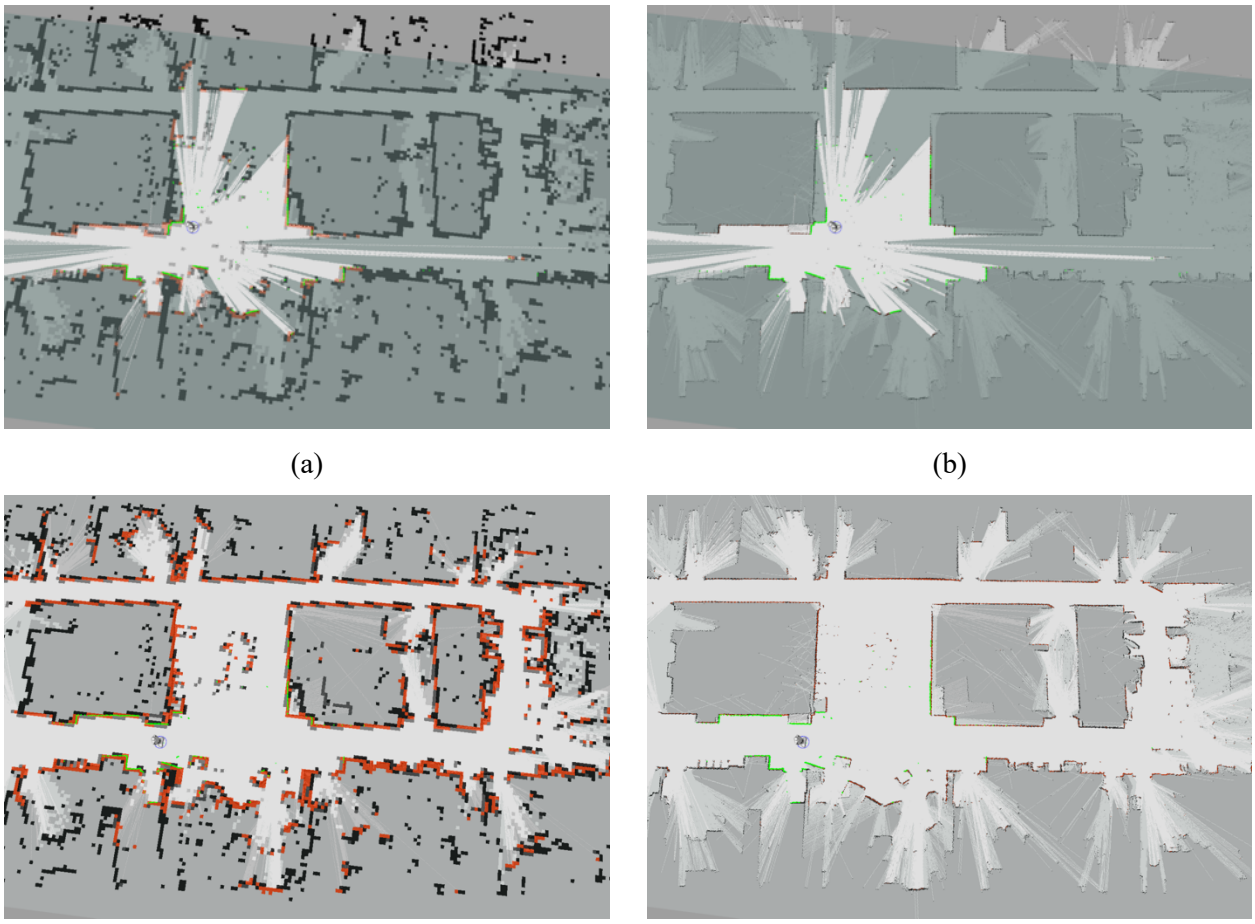
(a) (b) (c)

(d) (e) (f)

**Figure 15. Several simulated pedestrians, circled in yellow, are included in the test environment and can be observed by the robot through its LIDAR scans. The pedestrians are randomly assigned waypoints in the map and move between them, interacting with the robot (e.g. not allowing it to pass, etc.) in the process.**

Figure 16 shows the map matching solutions at the start and end of one experimental run. As with the experimental results from the simulated static environment, our map matching algorithm is able to accurately localize the robot at the start of the run without needing prior pose information. It is also able to maintain good localization throughout the run as the robot navigates around the environment. Comparing Figure 16 to Figure 8, it is clear that this map presents a much more challenging localization environment as the global reference map contains many reflection artifacts and is itself incomplete in places. Nevertheless, our map matching algorithm is still able to robustly localize the robot.

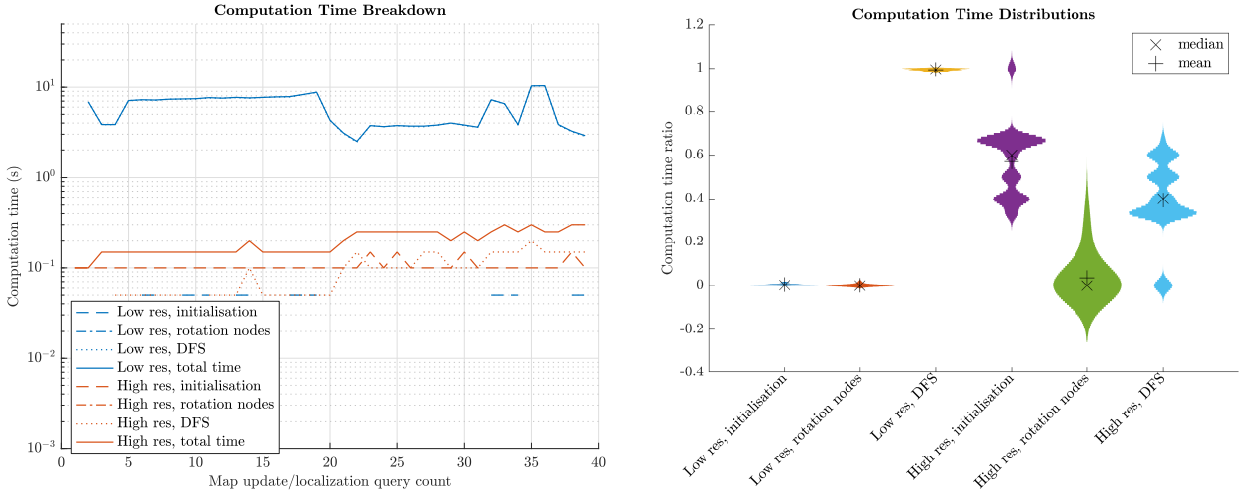
Figure 17 shows the corresponding computation time statistics for this experimental run. Compared to the simulated static environment experiments, the initial low-resolution matching step is slower, with each query taking between 2s and 10s. On the other hand, the high-resolution refinement step is actually faster in this map, taking less than 0.3s for each query in this run. On closer inspection of the computation time breakdown, we see that the main computation time savings come from a faster problem initialization, which overcomes a slightly longer DFS, during the refinement stage.



(a) (b)

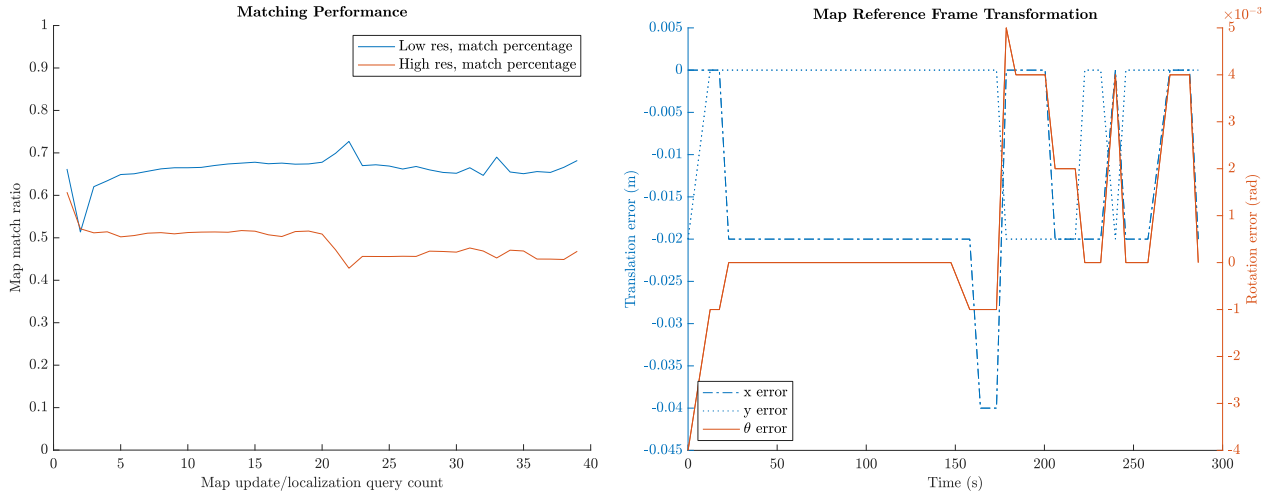
(c) (d)

**Figure 16. Low- (left column) and high-resolution (right column) map matching solution at the start (top row) and end (bottom row) of the experiment. The robot autonomously navigates through a sequence of commanded waypoints that send it on a clockwise loop around the right side of the map, ending back at its starting location. Red indicates matched cells. Qualitatively, the session and global reference maps align well despite the challenging sensing environment.**




(a) (b)

**Figure 17. Computation times for the low- and high-resolution steps in the map matching algorithm. In the recorded office map, the low-resolution matching takes longer than in the simulated static environment mainly due to the increased time to complete the DFS. However, the high-resolution refinement step tends to be faster, with the main savings coming from a faster problem initialization.**




(a) (b)

**Figure 18. Map matching performance over time, measured as (a) ratio of matched map cells and (b) the absolute transform error between the two map frames.**

The map matching performance is shown in Figure 18. As expected, the ratio of matched map cells is lower in this environment compared to the simulated static environment. Spurious LIDAR measurements incorporated into the current session map, as well as the presence of artifacts in the global reference map both result in a higher number of unmatched occupied cells between the two maps. However, analysis of the transform error between the two map frames shows that we still achieve very precise localization (less than 4cm absolute translation error and less than $5e^{-3}$ rad absolute rotation error).

Figure 19 shows the map matching ratios across all ten runs while the aggregated performance statistics are shown in Table 1. In particular, note that even without prior pose information, the average time to achieve an initial localization solution is less than 2s. Furthermore, the maximum localization translational error never exceeds 0.3m (the robot base footprint radius).


**Figure 19. Map matching performance at each localization query for all ten runs in the recorded ETH ASL office environment.**

**Table 1. Aggregate performance statistics for our map matching localization algorithm.**

| | Mean | Std. dev. | Min | Max |
|---|---|---|---|---|
| Computation time, low resolution (s) | 2.898 | 2.398 | 0.100 | 12.100 |
| Computation time, high resolution, (s) | 0.622 | 1.025 | 0.100 | 7.950 |
| Absolute transform error x (m) | 0.037 | 0.033 | 0 | 0.160 |
| Absolute transform error y (m) | 0.037 | 0.046 | 0 | 0.240 |

| | | | | |
|---|---|---|---|---|
| Absolute transform rotation error (rad) | $2.92e^{-3}$ | $2.55e^{-3}$ | 0 | 0.013 |
| Time to first localization solution (s) | 1.910 | 0.265 | 1.500 | 2.350 |

### 3.3.3. Evaluations on Robot Datasets

Finally, we evaluate the performance of our localization solution on rosbag data collected while running Pepper in our office environment during an open lab day event (this is from the same set of rosbags used to evaluate the effect of LIDAR scrubbing on mapping in Section 2.3). The number of attendees at the event numbered in the hundreds with crowd densities exceeding $2p/m^2$ in many areas of the map. Here we combine the LIDAR scrubbing described in Section 2 for generating the session map and compare the performance of cold-start localization using the original and scrubbed maps. When replaying the rosbag, we periodically reset the session map and allow the localization system 20s to establish a solution. Results are shown in Figure 20 where correct localization solutions (<1m error) are colored green and incorrect matches (>1m error) are colored red.


**Figure 20. Side-by-side comparison of localization cold-starts in a real, mixed-density CROWDBOT experiment using original (left) and scrubbed (right) LIDAR scans.**

Though the scrubbing leads to an increased amount of cold-start localizations in real examples (17.5% more correct localizations), the positive effect is limited. Our interpretation is that in very dense areas ($>2p/m^2$), almost all of the sensor field is covered by humans, and scrubbing does not allow the robot to localize as there is too little usable information on static environment geometry remaining. In addition, for a given true negative detection rate, very dense areas should result in a higher absolute amount of missed detections, and thus higher map and localization error. As visible above, the main improvement of scrubbing for localization occurs in medium density areas (i.e. along the corridors). However, this also shows that our solution is robust to dynamic pedestrians as these areas had the greatest movement of people traveling between the two open areas of the office.

# Conclusion

The CROWDBOT mapping and localization solutions presented in this report address the main challenges posed by navigating through dense human crowds. By integrating the RWTH DR-SPAAM LIDAR person detection algorithm into our perception pipeline, we are able to remove LIDAR returns from humans in the environment. This allows us to pass a "scrubbed" LIDAR scan to our online mapping routine, which contains far fewer confounding returns from dynamic obstacles, resulting in cleaner and more coherent maps of the static environment. In turn, this enables our fast, accurate and precise localization solution, which performs online, prior-free map-matching between the current session map and a global reference map. Our localization algorithm relies on a two-stage branch-and-bound search, which allows matching to occur at the original map resolution (2cm×2cm grid cells over a map size of approximately 55m×15m) while maintaining real-time performance. Our evaluations demonstrate that the mapping and localization system is performant up to crowd densities of approximately $2p/m^2$, however, beyond this, the ability of the LIDAR to perceive the static environment becomes highly limited and thus cold-start localization also begins to breaks down.

# References

[1] D. Jia, A. Hermans and B. Leibe, “DR-SPAAM: A spatial-attention and auto-regressive model for person detection in 2D range data,” arXiv preprint arXiv:2004.14079, 2020.
[2] D. Mammolo, “Active SLAM in Crowded Environments”, Master Thesis, ETH Zürich, 2018.
[3] W. Hess, D. Kohler, H. Rapp and D. Andor, “Real-time loop closure in 2D LIDAR SLAM,” in Proceedings of the 2016 IEEE International Conference on Robotics and Automation, pp. 1271–1278, 2016.