2020 IEEE International Symposium on
Safety, Security, and Rescue Robotics (SSRR)
November 4-6, Abu Dhabi, UAE

A Reliable Localization Architecture for Mobile Surveillance Robots

David Portugal[†], André Araújo[◇] and Micael S. Couceiro[◇†]

Abstract— **The operation of a mobile robot relies significantly on the robustness of its hardware and software, in which the self-localization system plays a crucial role. However, keeping a mobile robot localized at all times is not trivial, as there are diverse sources of error which may cause mislocalizations, affecting the whole system with unforeseen consequences. In this work, we describe an innovative and reliable localization architecture for mobile robots grounded on adaptive Monte Carlo localization (AMCL), sensor fusion and scan-matching techniques for mobile robots in large indoor areas. To validate the system, we run two 24-hour-long surveillance demos with an in-house developed mobile robot, comparing and analyzing a standard and widely used AMCL localization architecture against the proposed system. While the standard AMCL leads to 85% of operation time with accurate localization, the robot under the proposed approach does not depict a single localization issue during the 24-hour pilot.**

I. INTRODUCTION

Proven autonomy is the ultimate achievement for many researchers in mobile robotics [1]. However, several challenges hinder this goal. Seminal works addressing the operation of robots in human-populated environments over the years [2–8] point out to diverse factors, such as the dependency on humans to make occasional modifications to the environment, sensor limitations at close range (*e.g.* dead zones), unexpected collisions with dynamic obstacles, navigation relying on off-board resources or limited to safe routes, uncontemplated user intervention (*e.g.* robot kidnapping), hardware failures and software crashes, and, last but not least: maintaining accurate localization at all times, often resorting to artificial landmarks to correct the robot's pose.

In this work, we focus on conceiving a reliable localization architecture for mobile robots in human-populated environments, in the context of the STOP (*Cooperative Robotic Securities*) R&D project[1]. In brief, STOP is an industry-academia partnership project, which proposes the deployment of mobile robots in large human-populated indoor spaces, such as shopping malls, airports, offices, museums, and others, to carry out surveillance missions independently [9]. The main objective of STOP is to assist in the monotonous or repetitive tasks associated with the supervision, monitoring and surveillance of infrastructures, which have low-added value, and may ultimately pose a risk to human guards.

We consider localization as a fundamental capability for a mobile robot engaged in a monitoring task. It is a paramount requirement for reliable navigation, perception, interaction, and generally operating in the environment, thus enabling advanced technological innovations on service robotics, towards a robotic society. Although not addressed here, other aspects, such as autonomous charging, detecting human actions and changes in the environment, deploying teams of cooperative robots, *etc.*, are also relevant within the context of STOP [10].

The overall performance of a mobile robot is severely constrained by its fundamental localization system. Several advancements have been proposed in this domain over the years, from the early work of Drumheller [11], to the proposal of Kalman filter-based techniques [12], other Bayes filter variants [13], Monte Carlo localization [14] and associated variants [15]. Recent work dealing with robots for human-populated environments [18] propose a system architecture that supports recovery modes and involve both autonomous and human-aided recovery strategies to overcome the limitations of the system. The same authors conduct in [1] a 13-day uninterrupted run with a PR2 robot navigating in an unmodified office environment, and presented their views on how to increase reliability at the software engineering, autonomy, and environment level.

The CoBot robots also follow a symbiotic autonomy strategy, in which the robots are aware of their perceptual, physical, and reasoning limitations and pro-actively ask for help from humans, for example for object manipulation actions. The CoBots were able to overcome the 1,000-km challenge, demonstrating the robustness and accuracy in localization over long-term deployments [19], in the presence of variations and changes in the environments. More than 93% of robot deployments did not require human intervention to reset the robot localization, and a monitoring script running on the robots would track the tasks' execution progress and email researchers in the cases when the robots needed assistance.

In the Spatio-temporal Representations and Activities for Cognitive Control in Long-Term Scenarios (STRANDS) project, the focus was placed on enabling autonomous operation in everyday environments and allowing the robots to use their long run times to improve their performance [20]. The authors tackle the problem by providing design-time and run-time monitoring. Using a predictive model, the system's robustness is improved by avoiding situations where it previously encountered navigation issues.

*This work was supported by the **S**eguranças robó**T**icos co**OP**erativos (STOP) research project (CENTRO-01-0247-FEDER-017562), co-funded by the "Agência Nacional de Inovação" within the Portugal2020 programme.

[†]D. Portugal and M. Couceiro are with the Institute of Systems and Robotics, University of Coimbra - Pólo II, 3030-290 Coimbra, Portugal,{davidbsp, micaelcouceiro} at isr.uc.pt.

[◇]A. Araújo and M. Couceiro are with Ingeniarius, Ltd., R. Coronel Veiga Simão, Edifício CTCV, 3025-307 Coimbra, Portugal, {andre,micael} at ingeniarius.pt.

[1]http://stop.ingeniarius.pt

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on January 04,2021 at 09:38:01 UTC from IEEE Xplore. Restrictions apply.

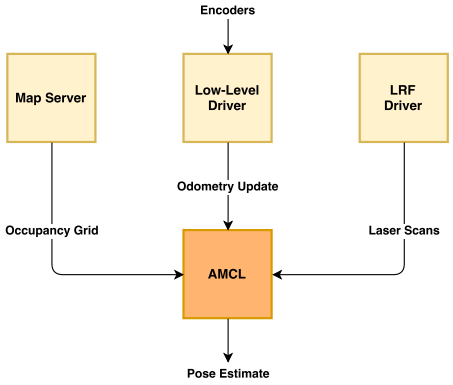


Fig. 1. Initial Localization Architecture (*Standard AMCL*).

Like CoBot, the STRANDS robots ask for help when they are unable to recover from an unexpected situation, *e.g.* if the robot's bumper is pressed, a hardware cut-off prevents it from moving forward, so the robot asks to be pushed away from obstructions by nearby humans.

Instead, our contributions in this work is twofold: *i)* to provide a robust and robot-independent localization system based on Monte Carlo Localization, sensor integration and scan-matching techniques, exploiting the estimation redundancy of laser scans, compass and odometry information; and *ii)* to validate the proposed architecture against a widely used localization system (*cf.* AMCL [16]) during two 24-hour-long surveillance demos in a large indoor facility.

In the next section, we describe the localization system devised. The experimental validation of the surveillance application using an in-house developed mobile robot is presented in section III, and results are discussed in section IV. The article ends with conclusions and plans for future work.

II. AMCL-BASED LOCALIZATION

With the increased use of common robotics middlewares (*e.g.* ROS[2]) and their tools, several research groups have turned their focus to particular and relevant high-level scientific goals and applications scenarios, dealing with the underlying issues, *e.g.* localization, navigation, *etc.*, by making use of the recognized and proven algorithms integrated by the robotics community. For the case of mobile robots operating in indoor environments, the localization system is typically the one presented in Fig. 1.

This architecture, which we deem as *Standard AMCL* makes uses of a 2D occupancy grid map representation, which can be derived from a simultaneous localization and mapping (SLAM) algorithm [21] ran previously in the indoor area, where the robot operates. Briefly, this probabilistic localization system receives the odometry estimates of the robot and laser scan readings, and uses a particle filter to track the planar pose of the robot (x, y, θ) against the known map, to output the most likely robot localization in the map. For more details on AMCL, one should refer to [16].

During the development of the STOP robotic platforms for indoor monitoring and surveillance, the *Standard AMCL* was the initial localization system adopted. As expected, it proved to be a solid approach, especially in short-term experiments. However, for longer runs, we felt the need to propose a more robust localization architecture to guarantee that our robots can be precisely localized at all times, and recover in case of unexpected situation. Table I clarifies the rationale behind our decision: the *Cons* column justifies proposing changes to the *Standard AMCL* system's localization architecture, and the *Pros* column shows that the advantages of AMCL are immense, and it is highly desirable to still guarantee these benefits. For this reason, and inspired by the success of previous initiatives [17], in the following subsection we propose a new and more reliable localization architecture based on AMCL.

TABLE I
STANDARD AMCL PROS AND CONS.

Standard AMCL	
Pros	**Cons**
Widely used and proven localization approach. Allows different input sensory sources. Easily parameterized with filter, laser and odometry models. Adaptive: KLD sampling adjusts the number of particles to use based on how sure the robot is of its pose, being computationally efficient (linear time complexity with respect to the number of particles). Alows filter re-initialization with a known pose.	Lack of robustness to poor odometry estimates due to sliding wheels, weak rotational estimates, low-level errors, and possibly other situations. Assumes low error in laser scan data, which can be severely obstructed momentarily, possibly leading to global estimation errors (*i.e.*, positional jumps).

Proposed Localization Architecture

As pointed out in [1], hardware failures rarely have a realistic recovery strategy. Thus, it is crucial that robotic software architectures are aware of hardware limitations to minimize their impact on system functionality. In order to uncover and diagnose the causes of localization failures, several tests were conducted with our in-house developed robot (*cf.* Fig. 2) navigating in a large indoor environment, while low-level sensor data was continuously logged. Our main findings were: the robot wheels slide in some areas of the environment, especially after the floor has been recently cleaned and when the robot rotates over its own axis, which leads to large odometry errors; also, the low-level data of the robot is not always consistent, having sporadic encoder jumps or "frozen" readings, *i.e.* consecutive repeated encoder values while in motion.

When wheel slippage occurs, the odometry estimate incorporates encoder ticks which do not correspond to actual robot movement, feeding the localization system with erroneous data. Moreover, inconsistent data coming from the robot encoders also have a huge impact on localization, and although the robot is likely to recover when in a translational motion, this is not the case in most rotations. In Fig. 3, we present the proposed localization architecture to overcome

[2]http://www.ros.org

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on January 04,2021 at 09:38:01 UTC from IEEE Xplore. Restrictions apply.

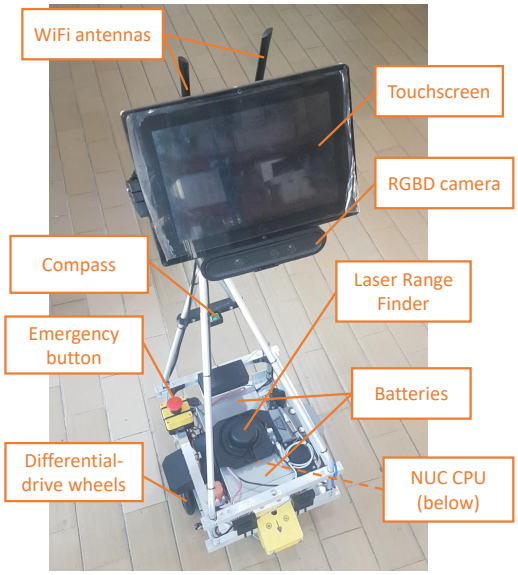


Fig. 2. In-house developed mobile robot for the STOP project.

the abovementioned issues. Changes to the low-level driver were conducted, with the integration of a new sensor (a digital compass), and three blue boxes highlight new components that were added to the localization architecture, when compared to the *Standard AMCL*. We now describe the key features of the proposed localization architecture.

a) Low-level driver: A digital compass (see Fig. 2) was incorporated in the robot to acquire an independent measurement of the yaw angle (θ orientation) of the robot. Moreover, a simple filtering method was implemented in the low-level driver to filter out encoder inconsistencies, thus avoiding the propagation of wrong encoder values to the upper level. The pseudo-code of the encoder filtering process is presented in Algorithm 1. In short, encoders values are only considered correct when the encoder differences and the velocities generated by these differences are within the limits of the robot specifications. Commonly wrong values, which arise frequently, are also filtered out.

b) Hector SLAM: Hector is a 2D SLAM system based on robust laser scan matching proposed by Kohlbrecher *et al.* [22]. The estimation of the robot motion in real-time uses the high update rate and the low measurement noise from modern LIDARs, and does not consider the robot's odometry. The 2D pose estimation is based on optimization of the alignment of beam endpoints with the map obtained from the SLAM system so far. The endpoints are projected in the map and the occupancy probabilities are estimated. Scan matching is solved using a Gaussian-Newton equation, which finds the rigid transformation that best fits the laser beams with the map. In this work, we make use of *Hector SLAM* to provide a redundant estimate of motion of the robot in the world, which we deem as *Scan Matching Odometry*.

c) Intermediate Driver: A new component of the architecture, the *Intermediate Driver*, is responsible to provide an accurate measurement of the robot's motion to feed AMCL. To this end, it aggregates and fuses the redundant x, y and

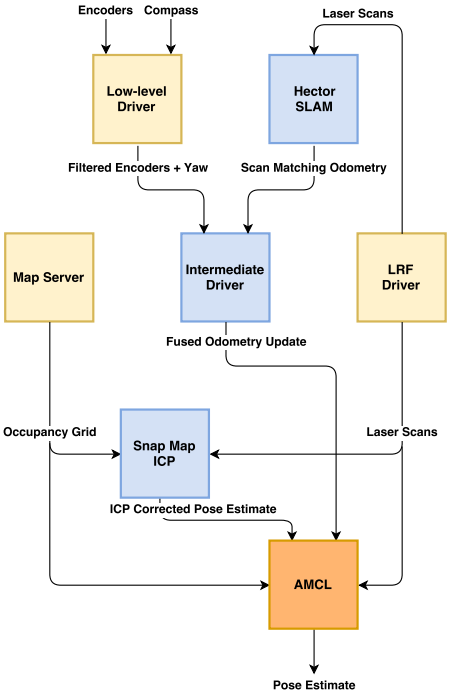


Fig. 3. Proposed Localization Architecture.

θ data from the filtered encoder readings, the compass, and *Hector SLAM*. This allows to compare the yaw angle given by the compass, which is immune to wheel slippage, against the yaw angle estimated by encoder calculations, as well as the translation derived from the encoders against that estimated by the scan matching odometry from Hector SLAM, which is known to fail due to the lack of discriminative features in corridors. Odometry fusion is performed based on simple rules: it is assumed that the odometry calculated from the encoders is generally precise. However, wheel slippage and rotational drift are explicitly identified by checking the data from the encoders against that reported by the compass and Hector SLAM. In Algorithm 2, we present the pseudo-code that implements the odometry fusion.

d) Snap Map ICP: AMCL allows the re-initialization of the particle filter at any time with a known pose. This is a particularly interesting feature; for instance, when the robot is mislocalized, as it allows the users to manually provide a reliable estimate back to the robot. In brief, the *Snap Map ICP*[3] component re-initializes the filter automatically when the software detects a high degree of confidence that the robot is in a new pose. For this, the component compares laser readings with the occupancy grid map in locations around the current pose reported by the robot, and uses an Iterative Closest Point (ICP) method [23] to estimate the most likely pose transformation with different degrees of confidence, based on Singular Value Decomposition (SVD). When the match quality is above a similarity threshold of 90%, the component triggers a pose re-initialization of the filter, preventing the robot to lose track of its pose.

[3]The implementation of *snap_map_icp* for ROS can be found in the following link: `https://github.com/davidbsp/snap_map_icp/`.

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on January 04,2021 at 09:38:01 UTC from IEEE Xplore. Restrictions apply.

```
Input  : repeated_encs, wrong_enc_times
Output : encRight, encLeft

// Initializations:
correct_encoders ← true;
rep_factor ← 1.0;
encRight ← get_encRight ();
encLeft ← get_encLeft ();

// 1.Repeated reading?
if repeated_encoders then
    repeated_encs ++;
else
    // larger tolerance if the encoder values were
       repeated just once (encoder skip):
    if repeated_encs = 1 then
        rep_factor ← 1.25;
    end
    repeated_encs ← 0;
end

// Encoder Differences:
encRight_diff ← encRight − last_encRight;
encLeft_diff ← encLeft − last_encLeft;

// 2. Check if differences are realistic:
max_diff = 2 · rep_factor · MAX_ENC_UPDATE · wrong_enc_times;
if abs (encRight_diff) > max_diff ∪ abs (encLeft_diff) > max_diff
  then
    correct_encoders ← false;
end

// 3. Velocities in (x,y,yaw) that would result from
   the encoder differences:
v_x = calc_pose_diff_x (encRight_diff, encLeft_diff) /
  ODOM_UPDATE_SECS;
v_y = calc_pose_diff_y (encRight_diff, encLeft_diff) /
  ODOM_UPDATE_SECS;
v_yaw = calc_pose_diff_yaw (encRight_diff, encLeft_diff) /
  ODOM_UPDATE_SECS;
linear_factor ← 2 · rep_factor · MAX_LIN_VEL · wrong_enc_times;
ang_factor ← 2 · rep_factor · MAX_ANG_VEL · wrong_enc_times;
if abs (v_x) > linear_factor ∪ abs (v_y) > linear_factor then
    correct_encoders ← false;
end
if abs (v_yaw) > ang_factor then
    correct_encoders ← false;
end

// 4. Filter out commonly wrong values (conservative
   filtering):
if commonly_wrong_enc_values then
    correct_encoders ← false;
end

// Allow encoder values to propagate:
if correct_encoders then
    wrong_enc_times ← 1;
    last_encRight ← encRight;
    last_encLeft ← encLeft;
    return (encRight, encLeft);
else
    // do not update last encoder values
    wrong_enc_times ++;
    return (-1, -1); // do not propagate encoder values.
end
```

Algorithm 1: Encoder Filtering (Low-level Driver).

As in the *Standard AMCL*, the particle filter is fed with the laser scans and the known map. Yet, the odometry updates are now provided by the *Intermediate Driver*, and the filter may occasionally be re-initialized by *Snap Map ICP*.

```
Input  : encoders, cmp_yaw, scan_matching_odom, N, h, c
Output : fused_odom_update

// Initializations for our robot:
N ← 0.8, h ← 1.715, c ← 3.0;
cmpYaw_diff ← cmp_yaw − prev_cmp_yaw;
encYaw_diff ← calc_pose_diff_yaw (encRight_diff,
  encLeft_diff);
using_encs ← true;

// Encoders are changing but compass reports (almost)
   no rotational movement:
if abs (encRight_diff) + abs (encLeft_diff) > 0 ∩ abs
  (cmpYaw_diff) ≤ (ODOM_UPDATE_SECS/c) ∩ abs (cmpYaw_diff)
  < abs (encYaw_diff) then
    using_encs ← false;
end

// Get Hector SLAM and encoders displacement in the
   last N seconds. Compute the translation
   corresponding to the poses' (x,y) centroid.
hector_displacement ← get_displacement (hector_queue_x,
  hector_queue_y, N);
encs_displacement ← get_displacement (enc_queue_x,
  enc_queue_y, N);
hector_translation ← translation (hector_x, hector_prev_x,
  hector_y, hector_prev_y);

// Hector SLAM reports much less movement than
   encoders:
if using_encs ∩ abs (encRight_diff) + abs (encLeft_diff) > 0 ∩
  abs (hector_translation) < (ODOM_UPDATE_SECS/h) ∩
  hector_displacement ·1.5 < encs_displacement ∩
  hector_displacement < (ODOM_UPDATE_SECS/h) then
    using_encs ← false;
end

// Prepare queues for next iteration:
enc_x ← calc_pose_x (enc_prev_x, encRight_diff, encLeft_diff)
enc_y ← calc_pose_y (enc_prev_y, encRight_diff, encLeft_diff)
push_back (hector_queue_x, hector_x);
push_back (hector_queue_y, hector_y);
push_back (enc_queue_x, enc_x);
push_back (enc_queue_y, enc_y);
if queues are filled then
    pop_front (hector_queue_x)
    pop_front (hector_queue_y)
    pop_front (enc_queue_x)
    pop_front (enc_queue_y)
end

// Return fused odometry:
if using_encs then
    // Encoders:
    enc_translation ← translation (enc_x, enc_prev_x, enc_y,
      enc_prev_y);
    fused_odom_update ← calc_odom_from (prev_pose,
      enc_translation, encYaw_diff);
else
    // Hector SLAM Translation + Compass Yaw:
    fused_odom_update ← calc_odom_from (prev_pose,
      hector_translation, cmpYaw_diff);
end
return fused_odom_update;
```

Algorithm 2: Fused Odometry (Intermediate Driver).

III. EXPERIMENTAL VALIDATION

In order to validate the proposed localization architecture, two 24-hour long pilots were planned, using the mobile robot unit of Fig. 2, which was continuously monitoring the floor of an indoor building, by following a patrolling approach to visit a set of distributed waypoints [24]. Fig. 4 illustrates the map of the floor with several pictures of the test site. In the first

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on January 04,2021 at 09:38:01 UTC from IEEE Xplore. Restrictions apply.

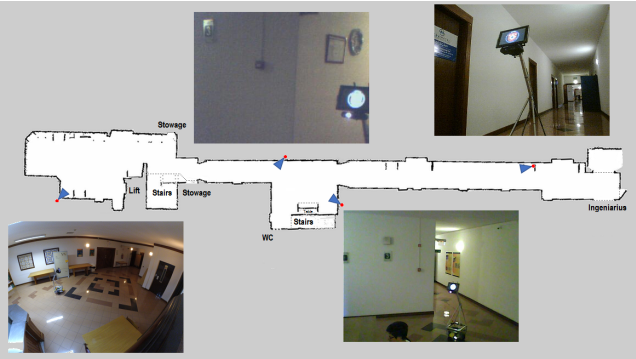


Fig. 4. Illustrative pictures of the test site during the pilots.

pilot, we have used the standard AMCL system, and, in the second pilot, the proposed localization architecture, allowing us to compare the results of both approaches. All the software is integrated in ROS and the navigation stack[4] is used in both pilots to navigate between waypoints. In terms of sensors, we have used a RPLIDAR A2, 360° laser range finder, in both pilots, and a CMPS11 tilt compensated compass in the second pilot. The pilots were overseen to guarantee the continuous operation of the robot in the environment. However, manual actions were kept to the minimum, limited only to re-localize the robot whenever lost, pushing it in indefinite sliding situations, and to recharge its batteries, since the autonomous charging system of the project is not ready yet.

IV. RESULTS AND DISCUSSION

Both 24-hour pilots allowed to validate the basic functional features of the robot and plan subsequent adjustments. The robot operated continuously for at least 8-hour intervals, and needed around 1 hour and 45 minutes for a full recharging cycle. So, it only needed to charge twice during these runs.

Using the *Standard AMCL*, the robot was able to successfully carry out the surveillance demo, safely navigating within the environment most of the time while avoiding obstacles, and correctly planning its trajectories, being perfectly localized 85% of the time. Fig. 5 presents an overview of the results collected during the 24-hour pilot with the *Standard AMCL*. One of the most important aspects in this experiment is the proliferation of localization issues, *i.e.* situations that lead to permanent robot mislocalization, as time goes by (red regions in the first bar of Fig. 5). We were not expecting so many localization issues during the pilot, possibly because 24-hour long tests were never conducted previously with this robot. Interestingly enough, during the first hour of experiments no localization failure was reported, and towards the end of the experiment, these failures became more frequent. The robot spent a total of almost three hours time with localization issues. As a consequence, a total of 8 independent collisions with obstacles occurred during the experiment (red regions in the second bar of Fig. 5). In the video of the experiment, which illustrates the robot's perspective, we can visualize its position in the map, the laser sensor readings, recent positional data of the robot, *i.e.* its trajectory, the battery and wifi connectivity state, and the onboard camera image in the visualization software, *rviz*.

A detailed analysis of the results for *Standard AMCL* allowed us to identify the source of errors revealed previously, which served as motivation for proposing the new localization architecture. Namely, wheels slippage and encoder jumps generate poor odometry estimates, which have a high impact on AMCL, and resulted in 67 manual re-initialization of the pose of the robot.

Fig. 6 presents an overview of the results collected during the 24-hour pilot with the proposed localization architecture. The integration of the digital compass, and the software modifications with the inclusion of the three new components resulted in a pilot without manual pose re-initializations, and no permanent localization failures at all, as can be seen in the video of the experiment. The localization system is now aware of the limitations of the robot hardware and is able to filter out erroneous encoder data (*low-level driver*), and fuse the odometry (*intermediate driver*) from redundant sources (encoders, digital compass and *HectorSLAM*) to feed the AMCL with more accurate odometry data. Moreover, the occasional automatic re-initialization of the localization filter (*Snap Map ICP*) does not allow errors from more complicated slippage situations, such as indefinite sliding, to propagate. The success of the pilot provided an initial validation of the localization architecture, showcasing its reliability and potential.

V. CONCLUSION AND FUTURE WORK

An innovative and reliable localization architecture, following a paradigm of software aware of hardware limitations, is presented in this work. The system benefits from sensor integration and scan matching techniques, and it is grounded on AMCL to accurately localize mobile robots operating in large indoor facilities.

In the short-term, we intend to further exploit the graceful degradation system ability of our multi-robot surveillance solution. To that end, we will study the effect of obstructions in the laser sensor, as the localization software may not be able to match features against the known map, *e.g.* due to people around the robot or severe map changes. We intend to further test and increase the robustness against these situations, by monitoring whether AMCL selects particles far away from recent areas where the robot operates and preventing this, as well as triggering global relocalization based on ICP and compass orientation. Still, the initial tests with the proposed localization architecture (see video at 14:35) show that the robot was able to remain localized in a densely human-populated area, despite occasional disturbance and localization drifts, which were overcome by the system automatically.

Moreover, we plan to conduct longer runs with our surveillance robots to assess whether failures occur under specific circumstances, and work on the remaining goals of the STOP project, as described in section I.

[4] http://wiki.ros.org/navigation

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on January 04,2021 at 09:38:01 UTC from IEEE Xplore. Restrictions apply.

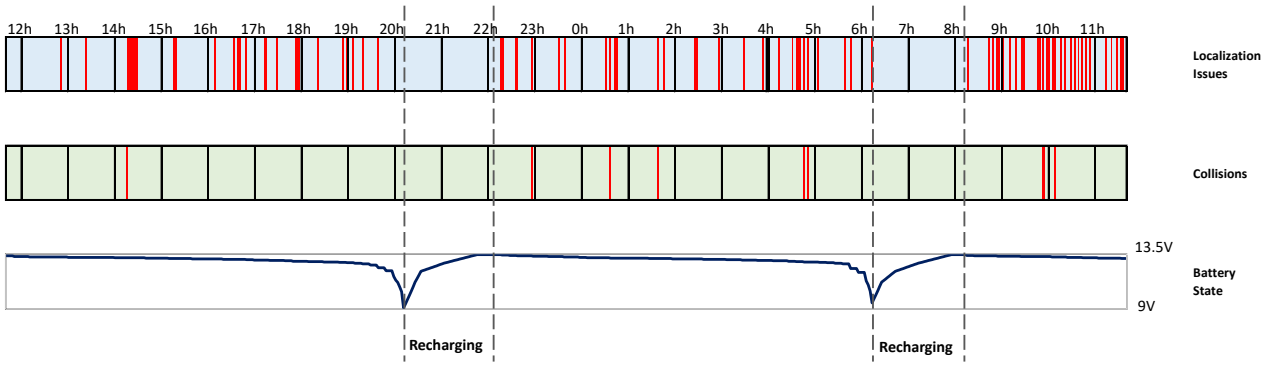


Fig. 5. 24-hour pilot with the *Standard AMCL*. Video available at: `https://goo.gl/5rt8MB` (60x faster).

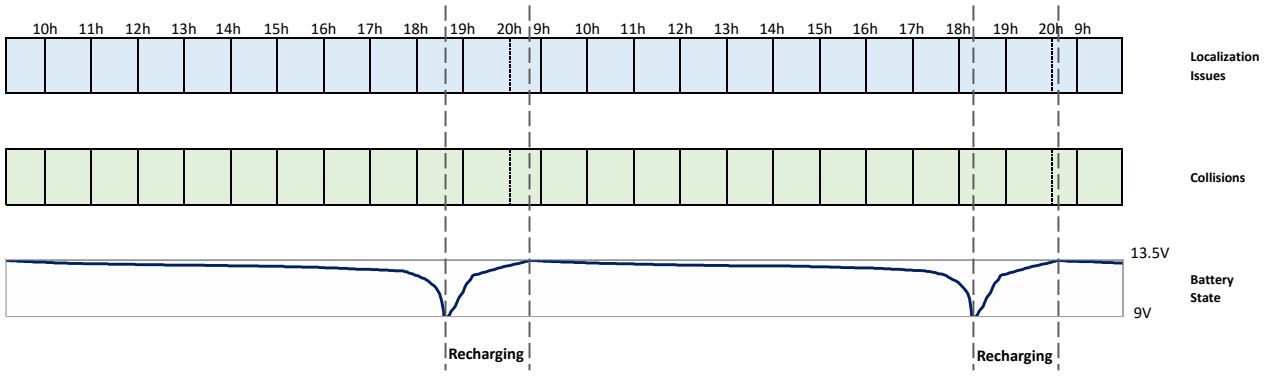


Fig. 6. 24-hour pilot with the proposed localization architecture. Video available at: `https://goo.gl/b7iCuc` (60x faster).

REFERENCES

[1] W. Meussen, E. Marder-Esppstein, K. Watts & B. Gerkey, "Long Term Autonomy in Office Environments". *Robotics: Science and Systems (RSS 2011)* Los Angeles, CA, USA, 2011.

[2] J. Evans, B. Krishnamurthy, W. Pong, R. Croston, C. Weiman & G. Engelberger, "HelpMate^{TM}: A robotic materials transport system". *Robotics and Autonomous Systems*, 5(3), pp. 251-256, Elsevier, 1989.

[3] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyerm D. Schulz, W. Steiner & S. Thrun, "The Interactive Museum Tour-Guide Robot". *Conference of the American Association for Artificial Intelligence (AAAI-98)*, pp. 11-18, Madison, WI, USA, 1998.

[4] S. Koenig & R. Simmons, "Xavier: A Robot Navigation Architecture Based on Partially Observable Markov Decision Process Models" . *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pp. 91-122, MIT Press, 1998.

[5] S. Thrun *et al.*, "MINERVA: A Second-Generation Museum Tour-Guide Robot". *International Conference on Robotics and Automation (ICRA 1999)*, Detroit, MI, USA, May 10-15, 1999.

[6] I. R. Nourbakhsh, "The Mobot Museum Robot Installations: A Five Year Experiment". *Int. Conf. on Intelligent Robots and Systems (IROS 2002)*, Lausanne, Switzerland, 2002.

[7] N. Tomatis *et al.*, "Designing a Secure and Robust Mobile Interacting Robot for the Long Term". *Int. Conf. on Robotics and Automation (ICRA 2003)*, pp. 4246-4251, Taipei, Taiwan, 2003.

[8] G. Kim *et al.*, "The Autonomous Tour-Guide Robot Jinny". *Int. Conf. on Intelligent Robots and Systems (IROS 2004)*, pp. 3450-3455, Sendai, Japan, 2004.

[9] D. Portugal, S. Pereira & M. S. Couceiro, "The Role of Security in Human-Robot Shared Environments: A Case Study in ROS-based Surveillance Robots". *Int. Symp. on Robot and Human Interactive Communication (RO-MAN 2017)*, Lisbon, Portugal, 2017.

[10] R. Prykhodchenko, R. P. Rocha & M. S. Couceiro, "People Detection by Mobile Robots Doing Automatic Guard Patrols," *Int. Conf. Autonomous Robot Systems & Competitions*, P. Delgada, Portugal, 2020.

[11] M. Drumheller, "Mobile Robot Localization Using Sonar". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2), 1987.

[12] J. Leonard, & H. Durrant-Whyte. "Mobile robot localization by tracking geometric beacons". *IEEE Transactions on Robotics and Automation*, 7(3), pp. 376-382, 1991.

[13] H. Badino, D. Huber & T. Kanade, "Real-Time Topometric Localization". *Int. Conf. on Robotics and Automation*, St. Paul, USA, 2012.

[14] F. Dellaert, D. Fox, W. Burgard & S. Thrun. "Monte Carlo Localization for Mobile Robots". *Int. Conf. on Robotics and Automation (ICRA 1999)*, Detroit, MI, USA, 1999.

[15] J. Biswas & M. Veloso, "Localization and navigation of the CoBots over long-term deployments". *The International Journal of Robotics Research*, 32(14), pp. 1679-1695, SAGE, 2013.

[16] S. Thrun, W. Burgard & D. Fox, *Probabilistic Robotics*, 2005.

[17] G. Peng *et al.*, "An improved AMCL algorithm based on laser scanning match in a complex and unstructured environment", Complexity, 2018.

[18] E. Marder-Esppstein *et al.*, "The Office Marathon: Robust Navigation in an Indoor Office Environment". *IEEE Int. Conf. on Robotics and Automation (ICRA 2010)*, Anchorage, AK, USA, 2010.

[19] J. Biswas & M. Veloso, "The 1,000-km Challenge: Insights and Quantitative and Qualitative Results". *IEEE Intelligent Systems*, 2016.

[20] N. Hawes *et al.*, "The STRANDS Project: Long-Term Autonomy in Everyday Environments". *IEEE Robotics & Automation Magazine*, 24(3), pp. 146-156, September 2017.

[21] G. Grisetti, C. Stachniss & W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters". *IEEE Transactions on Robotics*, 23 (1), pp. 34-46, 2007.

[22] S. Kohlbrecher, J. Meyer, O. Von Stryk & U. Klingauf, "A Flexible and Scalable SLAM System with Full 3D Motion Estimation". *Int. Symp. on Safety, Security and Rescue Robotics* Kyoto, Japan, 2011.

[23] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu & S. Behnke, "Registration with the Point Cloud Library: A Modular Framework for Aligning in 3-D". *IEEE Robotics & Automation Magazine*, 22(4), 2015.

[24] D. Portugal, M. S. Couceiro & R. P. Rocha, "Applying Bayesian Learning to Multi-Robot Patrol". *IEEE SSRR*, October 2013.

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on January 04,2021 at 09:38:01 UTC from IEEE Xplore. Restrictions apply.