 DOI: 10.3217/978-3-85125-663-5-19

A case study on working with a commercial mobile robotics platform

Clemens Koza[1] and Wilfried Lepuschitz[2]

***Abstract*— During a period of roughly one and a half years, the authors had the opportunity to work with a commercial mobile robotics platform, namely the "Apollo" platform by Slamtec, to implement real-world use cases in the domain of interactive entertainment installations. During the course of this development, the authors have gained insight into the strengths and limitations of the platform in realistic scenarios. This work will present the use cases, and discuss the experiences with and insights into working with the Apollo platform.**

I. INTRODUCTION

In the domain of interactive multimedia entertainment systems, there are many established forms of installations, from ordinary touch screens, to presence and gesture sensitive spaces, to multisensory "5D" cinemas. Mobile robots present an additional form of visitor interaction. However, entertainment installations bring their own set of challenges for a mobile robot, including an environment with a highly variable number of untrustworthy actors such as kids, and generally a lack of specialized personnel for supervision and upkeep.

II. APOLLO PLATFORM

The authors used the "Apollo" platform by Slamtec [1] to implement two real-world entertainment use cases. The cylindric robot weighs 40kg and has a diameter of 500mm. It has two driven wheels and four smaller casters for support. Payloads up to 35kg can be mounted on the circular top.

Among the robot's sensor systems is a LIDAR with a 270° field of view and 15m maximum scan radius, a depth sensor and six front-facing ultrasonic distance sensors. Hardware interfaces include WiFi and Ethernet, and a 20-25.2V 5A DC power interface to power additional devices on the robot.

Apollo has built-in simultaneous localization and mapping (SLAM), path planning with obstacle avoidance, and can dock with its charging base autonomously [1]. The "RoboStudio" software allows to view and edit the robot's map, edit the position of the home dock, inspect the robot's sensors, and to move the robot to specific positions [2].

The robot is controlled using the "Slamware SDK" (software development kit), which is provided as precompiled binaries for Windows, Linux, iOS and Android. The authors have only worked with the Linux SDK, although working with the Windows version was attempted as well.

The authors are with the Practical Robotics Institute Austria, Wexstraße 19-23, 1200 Vienna, Austria; [1]koza@pria.at; [2]lepuschitz@pria.at

III. DESCRIPTION OF USE CASES

A. Robot Swarm

In a first scenario, four robots decorated as animals were programmed to roam a small area of 3.5m x 5m. Three sides of the site were walled and one side open, providing clear landmarks for SLAM. Charging docks were placed against the three walls. Figure 1 shows a map of the space created by Apollo's SLAM system. The robots showed three basic behaviors:

- When a visitor is present in the area of operation, the robots head towards them. After a randomized timeout, they "lose interest" in visitors and continue roaming.
- When no visitors are present, or for a minimum amount of time after a robot "lost interest", the robots roam around the area in a random fashion.
- When their battery runs low, robots return to their charging stations.

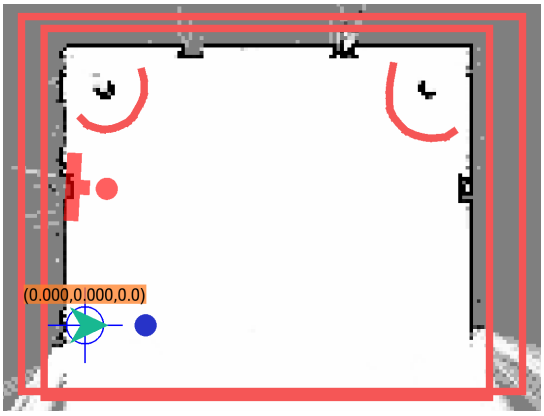

Fig. 1. A SLAM map of the robot swarm operation area. Displayed in red are virtual walls around the area and two hard to recognize chairs, and the first robot's charging dock on the left wall.

B. Greeting Robot

In the second scenario, two robots were used as mobile terminals, with touch screens mounted on top of the platform. These robots operated outside in a roofed area, on a rough concrete floor with some pebbles left on the ground.

Apart from some columns around the area, only the glass walls of the neighboring building were available as landmarks. Each robot had its own area of operation where it executed one of three behaviors:

- When a visitor is present in the area of operation, the robot heads towards that person and greets them.

- While a visitor is interacting with the touch screen, the robot does not move at all.
- Otherwise, the robot drives along a fixed path.

Automatic charging was of no concern, as the chargers were kept indoors and the duration of operation was short.

IV. FINDINGS

A. General findings

Some robots were found to turn in circles or sporadically change direction for no apparent reason. Testing showed that this happened because of faulty ultrasonic distance sensors. Some sensors did not recognize obstacles, while others reported false positives. False negatives did not generally impact the robot's behavior, as the LIDAR readings were accurate and reliable enough for obstacle avoidance.

While Apollo incorporates a depth camera, the camera's data can not be accessed by client applications. It is used solely for recognizing obstacles such as desks that are not visible to the LIDAR, which is mounted at a height of roughly 240mm. The depth camera would be a great tool for obstacle identification or human pose recognition [3], which makes it regrettable that it is not usable for these tasks.

Some features in the library turned out not to be implemented. For example, movement commands support specifying the desired speed. The speed option would be ignored however; support suggested to set the system-wide speed prior to any movement command instead.

B. Findings from the "Robot Swarm" use case

The defining requirement of this scenario was the need for coordination between robots: with detecting visitors limited to Apollo's LIDAR sensor, robots had to be aware of each other to not mistake each other for humans. An architecture with a centralized controller was used. That controller had an ethernet connection to a WiFi router, which would then connect to the robots. As Apollo has two strong WiFi antennas, this allowed for a reliable connection.

Although the Android Slamware SDK includes functions for Domain Name System Service Discovery (DNS-SD), the authors were not able to find robots by a service identifier or host name on the local network. This necessitated a more complicated design than was originally anticipated.

Apollo does not support coordinated path planning, e.g. as proposed in [4]. There is no shared coordinate system among robots and each robot is only aware of itself. It was observed that two robots driving towards each other would not recognize each other in time to prevent a collision. This was largely addressed by avoiding "risky" goals on the controller. Still, as Apollo's path planner may choose a different path from what the controller anticipated, this risk assessment could not be perfect and additional measures had to be taken.

While obstacle avoidance is a built-in feature of Apollo, target tracking had to be implemented manually. The authors first tried to let Apollo plan a path to the target. This did not work, as each update to the target position would stop the robot before potentially trying a vastly different path. Instead, using direct motion commands was necessary to track targets. For this, straight motion and stationary rotations are available; driving circular arcs is not supported. Targeting using direct motion commands worked very well, but these bypass obstacle avoidance, requiring greater care in the client code.

Finally, the presence of four robots, plus potentially multiple visitors, meant that a significant amount of stationary SLAM landmarks were obscured at any point in time. This impacted robot localization, which led to robots occasionally mistaking other robots for visitors. Fortunately, the correct location would be recovered eventually, and the swarming of robots around another did in fact look very fitting.

C. Findings from the "Greeting Robot" use case

The greatest challenge of the second scenario was the outdoor environment. As rough ground, pebbles and weather are hard to predict, commercial mobile robots are generally not advertised as suitable for outdoor operation. Apollo is no exception, but it operated fairly well, as long as the operation area was cleaned from pebbles every day before starting the robots. Apollo was able to recover from getting stuck on the rough ground fairly reliably, although getting stuck frequently led to jagged motions. Larger casters would be necessary to improve Apollo's outdoor driving performance.

The adjacent buildings had glass fronts with wall decals; the transparency made the fronts unsuitable as SLAM landmarks. An opaque tape was added to the fronts, which immediately remedied the problems.

It was observed that the low sun disturbed the operation of the LIDAR. This was no problem during operation, as the desired operation hours were after sundown, but limited the realism of testing conditions during the rest of the day.

V. CONCLUSIONS

Although Apollo showed some shortcomings, especially regarding multiple robots operating in the same space, it is a capable platform. The authors found that the actual tasks were relatively easy to handle, but details around the setup, such as discovery on the local network and map management, could be presented to application developers in an easier fashion.

The fact that Apollo is closed-source and is not using Robot Operation System (ROS) must be considered when evaluating the platform, especially if Apollo is to be added to an environment that is already based on ROS.

REFERENCES

[1] Apollo medium robot development platform parameters. [Online]. Available: https://slamtec.com/en/Apollo/Spec

[2] Robostudio extendable robot management and development software. [Online]. Available: https://slamtec.com/en/RoboStudio

[3] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2011, pp. 1297–1304.

[4] T. Siméon, S. Leroy, and J.-P. Lauumond, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, 2002.