Proceedings of the Austrian Robotics Workshop 2025

ROS with LEGO Spike

Jakob Buchsteiner[1], Daniel Marth[2], Moritz Taferner[1] and Markus Bader[1]

***Abstract*— Teaching mobile robotics algorithms through hands-on hardware exercises can be both costly and resource-intensive. This work addresses this challenge by introducing an affordable differential drive vehicle constructed from LEGO components. An onboard Raspberry Pi, equipped with a camera and a Build HAT, provides standard ROS2 interfaces. An outstanding feature of the design is the calculation of laser ranger data from camera images, which enables the investigation of sensor and motion models, as well as probabilistic approaches for self-localization and mapping. The paper presents a prototype together with statistical results on the motion and sensor models within the real and simulated environment.**

***Index Terms*— ROS2, Mobile Robot, Self-Localization**

I. Introduction

Robot Operating System (ROS) plays a major role in the growing field of robotics, especially in education, such as teaching mobile robotics. Integrating affordable hardware with a software platform like ROS enables undergraduate students to gain hands-on experience in robotics.

This paper explores the possibilities of integrating the Lego Spike PRIME robotics kit into the latest version of ROS2 [1], utilizing a Raspberry Pi 4 single-board computer. For tight integration with the ROS2 ecosystem we employ pre-existing components such as *ros_control* [2], the default ROS2 implementation of AMCL (Adaptive Monte Carlo Localization) and the simulation tool Gazebo [3]. The experimental evaluation shows the viability of the presented approach as a base platform for simple localization tasks.

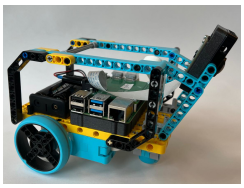

Fig. 1: The assembled robot used for the evaluations of this paper.

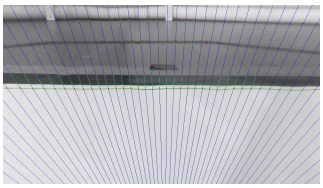

Fig. 2: Generating laser range data for localization using low-cost camera images.

II. Related Work

Commercial platforms such as the *Turtlebot* [4], which was specifically developed for ROS, are also frequently used in education to practice hands-on mobile robotics. However, the acquisition costs play a very large role and therefore make it a less accessible option for institutions with limited resources. [5] also looks at using Lego Spike based robots in combination with ROS2 and Gazebo, however does not address localization. Our design fills this gap by extracting laser ranger data from camera images, allowing direct application of textbook algorithms like [6].

*This work was not supported by any organization

[1]Jakob Buchsteiner, Moritz Taferner and Dr. techn. Markus Bader (first.last@tuwien.ac.at) are with Faculty of Informatics, Vienna University of Technology.

[2]Daniel Marth (daniel.marth@tum.de) is with the Department of Computer Science, Technical University of Munich.

III. Implementation

To provide a base platform for teaching, we developed three primary components: hardware support, localization system using a camera and a simulation environment. The components draw their information from a shared robot description in the Unified Robot Description Format (URDF).

A. Hardware Support

One challenge of hardware integration is retaining reusability for different robot designs. Thus, we leverage existing motion controllers of the *ros2_control* framework, by providing the robot description and a plugin serving as a hardware abstraction layer for the Lego hardware. This layer controls the actuated wheels connected to the Raspberry Pi Build HAT, communicating using the documented serial protocol.

For evaluation, we use a differential drive platform with two independently driven wheels on each side of the robot and a caster wheel, allowing the robot to move in both linear and angular directions (Fig. 1). The differential drive controller included in *ros2_control* then translates motion commands to wheel velocities and performs odometry with the data from the wheel encoders.

B. Localization

Classic implementations of AMCL and SLAM (Simultaneous Localization and Mapping) operate on laser range data [6], however such sensors are expensive. We solve this issue by developing an intermediate layer, which extracts distance measurements from camera images using line markings on the floor.

Using the cameras intrinsic and extrinsic calibration parameters we construct a projective transform $H \in \mathbb{R}^{3\times3}$, mapping points on the ground plane to points on the camera plane. We determine lines in the image corresponding to radial lines around some ray center point on the ground plane. Along each ray, we apply a simple edge detection kernel, and estimate the width of the line using two pairs of line entry and exit points. After checking against the width threshold to isolate line markings from other line features, entry points are reported as the distance measurement in the respective direction (Fig. 2).

https://doi.org/10.34749/3061-0710.2025.20

C. Simulation

Since many educational robotic tasks can be prototyped and evaluated in simulation, we set up a simulation environment to support development and preliminary testing. The same robot description used for hardware support is enriched by physics parameters specific to the simulator Gazebo. Most noteworthy are mass, rotational inertia and friction. We approximated inertia by dividing the robot into subcomponents, each of which were approximated as cuboids and cylinders with evenly distributed mass. The inertia calculation is then automatically performed by Gazebo. Friction parameters were manually tuned to prevent the robot from slipping in the simulation ($\mu_1 = \mu_2 = 1.0$).

IV. Results

We evaluated and compared the vehicle's pose estimation using only the implemented motion model (odometry) and using AMCL with our emulated laser ranger data. Ground truth data was acquired from an OptiTrack motion tracking system.

A. Experimental Setup

The map used for evaluation is roughly a square with a side length of one meter and black, 5 cm thick tape markings used for localization. The robot is instructed to follow a figure-eight trajectory using open-loop control, and the trajectories estimated using odometry and AMCL are then compared against ground truth. Performance is analyzed in both simulation and real world environments according to [7].

B. Analysis

For a qualitative analysis of self-localization accuracy, we plot the trajectories for both simulation and real-world environments (Fig. 3). The convergence of the estimated AMCL trajectory towards the ground truth can be observed in Figs. 3a, 3b and 3d. We can also observe reasonably accurate odometry in Fig. 3c.

In simulation and without an offset in the initial pose estimate, the estimation using AMCL shows an absolute trajectory error of 9.6 mm, while the accuracy of the odometry trajectory has absolute trajectory errors above 50 mm.

Since [7] disregard offsets in the trajectory's starting pose, the numeric evaluation suggests that the estimated AMCL trajectories in the real world are worse than the raw odometry. However, Fig. 3d shows the improvement the localization system achieves: While the odometry can never match the ground truth trajectory, the AMCL particle filter relatively quickly converges to the correct position.

V. Conclusion

The robot's performance under real-world conditions demonstrated its potential as a suitable platform for teaching fundamental robotics concepts, such as navigation and localization. Robot control was proved to be precise in both simulation and real-world environments. Although self-localization solely derived from the wheel encoders deteriorates over time due to drifting, the AMCL particle filter did not only achieve smaller pose errors but could also recover from incorrect initial pose estimates.

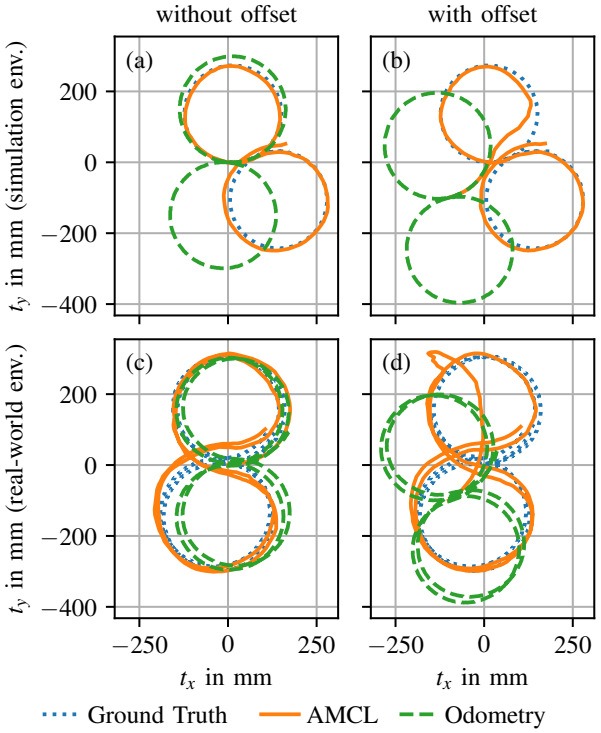

Fig. 3: Ground truth trajectories vs. the trajectories estimated by odometry and AMCL in simulation and real world environments. The right column shows that AMCL with emulated LIDAR data is able to recover from an initially incorrect pose estimate ($\Delta x = -100$ mm, $\Delta y = -100$ mm, $\Delta\theta = 0.3$).

References

[1] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[2] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ROS," *The Journal of Open Source Software*, 2017.

[3] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[4] R. Amsters and P. Slaets, "Turtlebot 3 as a robotics education platform," in *Robotics in Education*, M. Merdan, W. Lepuschitz, G. Koppensteiner, R. Balogh, and D. Obdržálek, Eds. Springer International Publishing, 2020, pp. 170–181.

[5] O. Gervais and T. Patrosio, "Developing an Introduction to ROS and Gazebo Through the LEGO SPIKE Prime," in *Robotics in Education*, M. Merdan, W. Lepuschitz, G. Koppensteiner, R. Balogh, and D. Obdržálek, Eds. Cham: Springer International Publishing, 2022, pp. 201–209.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent robotics and autonomous agents series. MIT Press, 2006.

[7] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

https://doi.org/10.34749/3061-0710.2025.20