RoboCup Junior Rescue Simulation 2024

Team Description Paper

“HTL”

Abstract

- In the RoboCup Junior Rescue Simulation, the virtual robot is assigned the mission of navigating through a complex maze, mapping out its pathways and various chambers, while also detecting and locating victims along its route.
- In this case, the robot uses the 360° field of view from the LiDAR scanner alongside a GPS and a Gyro to achieve non-tile-based navigation. Furthermore, three cameras make it possible to identify each fixture even if the images are incomplete.

1. Introduction

We are team "HTL" from Slovakia, affiliated with SPŠE Prešov. Our team was formed in late 2023 and comprises four members.

Members:

- Adrian Paľa (tester)is the most experienced member of our team. Last year, he won the second prize in the Slovak round of RoboCup OnStage competition and advanced to the European level. Also, he participated in some minor competitions.
- Patrik Hric (debugger) has participated in minor competitions.
- Tomáš Pribula (movement) is an expert in programming, he also develops software for companies and participates in minor competitions.
- Marek Boháč (wall tokens detection) has participated in minor competitions.

2. Project Planning

a. Overall Project Plan

Building a map-exploring robot was a rigorous learning experience, pushing our boundaries in programming, critical thinking, and strategic planning.

- Programming: We meticulously designed and implemented a robot, tackling advanced algorithms, sensor integration, and real-time decision-making.
- Critical Thinking: Through continuous analysis and refinement, we overcame unforeseen challenges and optimized performance.
- Strategic Planning: We allocated resources, prioritized tasks, and adapted methods for maximum efficiency, anticipating obstacles and aligning efforts with project goals.

We divided the project into several key milestones, focusing on robot design, software development, and testing. Each member utilized their expertise to contribute effectively throughout the process.

Milestone	Description	Function/Team Member	Deadline
M0: Refine our Python skills	Learning how to use OpenCV, NumPy, other libraries	All Team Members	October 25,2023
M1: Design and Component Selection	Research and select optimal components based on budget and functionality.	All Team Members	October 27, 2023
M2: Sensor Integration and Software Development	Integrate sensors, develop core functionalities like navigation and object detection.	All Team Members	December 8, 2023
M3: Testing and Refinement	Test functionalities, identify and address challenges, refine algorithms and strategies.	Debugging team (Adrian, Paťo)	January 12, 2024
M4: Finalization and Documentation	Finalize the robot, prepare documentation and presentation materials.	All Team Members	March 3, 2024

b. Integration Plan

Our solution integrates various components seamlessly to achieve mission objectives. Symmetric sensor placement mitigates environmental noise, ensuring consistent performance. Motors enable movement, while cameras and LiDAR facilitate visual recognition and mapping. Software architecture divides tasks between long-term planning and short-term action, enhancing efficiency and adaptability.

3. Robot Design

- The design considers the most efficient solution for each task while keeping in mind the budget on the robot customization tool.
- To prevent noise generated by the simulated environment from affecting the robot's functionality, we placed the sensors symmetrically. This way, the interference is the same on each side and does not cause any malfunctions.

Components

- Motors (2x): used for movement throughout the maze, possess individual speed and directional control. Each motor is connected to a wheel.
- Cameras (3x): utilized for visual recognition of fixtures, obstacles, and floor type. They replaced color sensors to economize (by using color filters). Two cameras are positioned on each side, with one in the front, allowing an almost 90° field of view.
- GPS: obtains the robot's coordinates, guides it throughout the navigation and re-orientates it in case of LoPs (teleportation). It is in the center to get accurate measurements.
- Gyroscope: measures the robot's orientation, working as a detector if it is not advancing in a straight line. It is in the center to get accurate measurements.
- LiDAR: creates a points cloud that detects objects and obstacles surrounding the robot. In contrast to distance sensors, it has no trouble identifying curved walls, calculating distances, or sampling angles. It is placed on top of the robot, taking full advantage of its 360° detection.

*The LiDAR can also get raw detection, mainly used to correct the robot's orientation.

4. Software

Our program is divided into two:

- Explorer (Long-Term Planner): This component analyzes a 2D array of nodes to determine the most efficient path to a specific point in the maze (see section 3b). It provides instructions to the Navigator (Short-Term Actor).
- Navigator (Short-Term Actor): This component processes sensor data to update the Explorer's array and executes simple movements based on the Explorer's instructions.

a. General software architecture

To enhance performance, the robot's tasks are divided into two sections:

- Main Loop: This continuously handles core functionalities like navigation, mapping, and detection. It constantly gathers data from the Explorer and feeds the Navigator with instructions to complete actions. Additionally, it monitors for critical states like "Stuck," "End," or "Report_Victim" that could lead to malfunctions. If a change is detected, it triggers the state machine.
- State Machine: This manages different situations or changes identified by the main loop. Each situation has a dedicated state:
- Init: Calibrates the robot's position and rotation and performs startup tasks (only runs once at program start and transitions immediately to "Explore").
- Explore: Continuously sends the LiDAR grid data to the Explorer for instructions while applying minor position adjustments. The grid updates upon receiving instructions.
- Report Victim: Stops the robot for a second upon detecting a victim, reports the find to the supervisor (updating the victim's status in the map), and resumes exploration.
- Stop: A temporary state for debugging (no clear cause), returning to exploration upon resolution.
- Stuck: A temporary state for debugging situations where the robot is blocked (excluding wheels), returning to exploration after being unstuck.
- Teleported: A temporary state for handling Lack of Progress (LoP), recalibrating the position, and resuming exploration.
- End: Signals the completion of navigation, mapping, and detection, and ends the program.

b. Navigation

To efficiently navigate the maze, the robot employs two key components:

- Explorer: This component utilizes information from the map to plan optimal routes. It employs a Breadth-First Search algorithm to efficiently explore nodes level by level, prioritizing the goal node. For finding the shortest path, the A* algorithm is used. It considers both the traveled distance and an estimated distance to the goal, ensuring efficient traversal while minimizing travel time.
- Navigator: This component executes the movement instructions provided by the Explorer. It handles pre-navigation calibration and post-navigation data collection. Notably, the robot's movement is no longer restricted to tiles, allowing for smoother exploration.

- Pathfinding Details:
  - The Explorer utilizes a 2D array representing the environment, obtained from the mapping process.
  - A Breadth-First Search algorithm efficiently explores nodes, prioritizing the goal.
  - The A* algorithm finds the shortest path considering both traveled and estimated distances.
  - Heuristics in A* ensure the estimated distance is never overestimated, guiding the search effectively.

- Movement Details:
  - The Navigator translates Explorer instructions into movement commands.
  - Pre-navigation calibration ensures sensor accuracy before pathfinding begins.
  - Non-tile-based navigation allows for more fluid exploration.
  - The robot continuously gathers LiDAR data during movement for map updates.

c. Wall Token detection

Our robot utilizes cameras, controlled by the Explorer, for object detection based on color, shape, and size, eliminating the need for color sensors. Regarding color detection, three filters are applied to the camera (red, white, and yellow). As for shape and size, the Explorer checks the number of pixels in the image –in the perimeter and the area– and classifies them by color.

- Fixtures:
  - Continuously analyzed during navigation using color filters (red, white, yellow) for differentiation from the environment.
  - OpenCV library functions are used for perspective correction and shape analysis.
  - Victim and hazard sign recognition relies on pixel proportions within specific image zones and comparison with pre-loaded data.

- Floor Type:
  - Recognized solely by color since shape and size are constant within each area.
  - Color channels determine the floor type (e.g., black hole, swamp) impacting path planning.
  - Information is stored for map creation.
- Obstacles:
  - Identified by color, shape, and size due to camera inclination.
  - Color filters differentiate obstacles from the environment and fixtures.
  - Size calculations and LiDAR data determine distance for path adjustment.
  - Information is stored for map creation.
- Improvements:
  - Non-tile-based navigation eliminates the need for tile-based detection.
  - Optimized comparison matrix improves detection speed and computational efficiency.

d. Mapping

Navigation and detection contribute significantly to map creation. During navigation, the Explorer employs LiDAR data to build a point cloud representing walls and obstacles. This information gets stored in a 2D NumPy array.

- Initial Grid:
  - Created using LiDAR data, storing wall and obstacle positions.
  - Expands dynamically as the robot explores.
- Granular Grid:
  - Combines the initial grid with additional information (floor type, fixtures).
  - Provides a detailed representation but lacks the final format.
- Bonus Grid:
  - Final map format, converted from the granular grid.
  - Rectangular matrix enabling accurate curved wall detection.
- Improvements:
  - Non-tile-based navigation allows for more accurate mapping of curved walls.

5. Performance evaluation

Evaluating our robot's performance against the challenges of the RoboCup Junior Rescue Simulation was crucial for identifying strengths, weaknesses, and areas for improvement. This section dives into the testing procedures implemented, how we analyzed the results, and their impact on the development process.

Testing Procedures:

- To comprehensively assess our robot's capabilities, we implemented a rigorous testing regimen:
- Simulated Environments: We designed various simulated mazes with diverse complexities, mimicking potential competition scenarios. These mazes incorporated different elements like obstacles, varying floor types, and victim placements.
- Metrics and Measurements: We tracked key performance metrics throughout the tests, including:
    - Navigation Efficiency: Time taken to complete the maze and reach the goal.
    - Object Detection Accuracy: Percentage of correctly identified fixtures, victims, and obstacles.
    - Mapping Accuracy: Comparing the generated map with the actual maze layout.
    - Stuck Instances: Number of times the robot became stuck due to various factors.
- Multiple Runs: Each test was replicated multiple times to account for environmental variations and ensure the results were statistically sound.

Analysis and Impact:

- We meticulously analyzed the test results to gain valuable insights into the robot's performance:
- Strengths: The robot excelled in object detection accuracy, consistently identifying fixtures, victims, and obstacles with high precision. The utilization of LiDAR for navigation proved effective, allowing for efficient exploration of complex environments.
- Weaknesses: One of the major concerns identified was the occasional issue of the robot getting stuck. Analyzing the "stuck instances" metric revealed that the problem often stemmed from navigating uneven terrain or encountering narrow spaces.
- Development Impact: The performance evaluation played a pivotal role in shaping the development process. The identified issue of getting stuck prompted us to:
    - Refine the movement algorithms to better handle uneven surfaces and tight spaces.
    - Implement additional sensor data analysis to anticipate potential sticking scenarios.
    - Conduct further testing with specific focus on navigating challenging terrains.

Learning from the Maze:

- While our robot achieved a competitive score of 4000 on one of the largest maps, the identified sticking issue presents an area for further improvement. Through comprehensive testing, meticulous analysis, and continuous refinement, we aim to optimize our robot's performance for future competitions. This experience has emphasized the importance of rigorous evaluation and iterative development, allowing us to learn from challenges and build a more robust and adaptable robot for the RoboCup Junior Rescue Simulation.

6. Conclusion

In conclusion, we have developed a sophisticated map-exploring robot for the RoboCup Junior Rescue Simulation 2024. The robot integrates various components, including LiDAR, cameras, GPS, gyroscope, and motors, to navigate through a maze, detect victims, and create a detailed map of its environment. Our project planning involved meticulous milestones, and our software architecture divided tasks effectively between long-term planning and short-term action. The navigation system utilizes a Breadth-First Search algorithm and A* algorithm for optimal pathfinding, while the robot's non-tile-based movement allows for smoother exploration. We have faced various challenges during performance evaluation, notably occasional issues of the robot getting stuck, leading to adjustments in movement algorithms and sensor data analysis. Despite achieving a competitive score, we acknowledge areas for improvement and emphasize the importance of rigorous evaluation and iterative development for future competitions.

References

[1] Robo Cup Rescue Simulation website

<https://junior.robocup.org/robocupjuniorrescue-league-simulation/>

[2] NumPy library official website

<https://numpy.org/>

[3] OpenCV library official website

<https://opencv.org/>

[4] A* algorithm

https://www.geeksforgeeks.org/a-search-algorithm/

[5] Breadth First Search algorithm

https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

[6] OpenCV Overview

https://www.geeksforgeeks.org/opencv-overview/?ref=header_search

[7] OpenCV library, geometric transformation

<https://docs.opencv.org/4.x/da/d6e/tutorial_py_geometric_transformations.html>

[8] OpenCV library, changing color space.

<https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html>

[9] OpenCV library, contour approximation

<https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html>