Contribution report for project ALIA-DELIVERY

Simulation Environment Development and Enhanced Robot Navigation

Somantha Manuranga, s.mandalawalli-acharige@oth-aw.de[1]

[1]*OTH Amberg-Weiden, Study programme "Industrie 4.0 Informatik ,Project Repository :https://git.oth-aw.de/9923/alia-delivery/-/tree/main"*

Report written on July 4, 2025

Abstract

The project ALIA-DELIVERY targets to develop a simulation environment where delivery robots operate autonomously within a hospital-like floor plan. The simulation serves as a digital twin prototype for testing key use cases such as delivering medication, transporting cafe orders, and supporting assisted living services.
This report shows the development of the Webots simulation world, where the robot's optimized movement within a room was successfully simulated. Further, a path planning method was used to navigate the robot within the Webots floor world.Here, robot movements were tested inside the room such as cell based movement and diagonal movement.

Keywords: *Simulation world, Webots,Navigation*

1. Introduction

Care 4.0 represents significant evolution in healthcare, leveraging digital technologies such as Artificial Intelligence, robotics and cyber-physical systems to increase both the quality of healthcare. This Alia site project aims to create an inclusive living space that will serve as a real-world test for these innovative Care 4.0 technologies. A key part of this project is creating a digital twin building. This virtual model will help to test and develop autonomous robots for care tasks like delivering medication, carrying items, and supporting residents in daily activities. The objective of this project is to simulate a delivery robot operating in a hospital-like floor plan using the Webots robotics simulator. Core tasks included enabling the robot to receive high-level delivery commands, compute paths to destination rooms, and execute movements autonomously. The simulation is visualized as a digital twin prototype, capable of modeling real-world service scenarios such as medication transport or cafe delivery within a healthcare context. The chosen platform, Webots, offered the necessary flexibility in motion and controller to implement various navigation strategies.Webots platform was used as the software material, and Python was used to implement the robot navigation inside the floor world.

2. Scope

Care 4.0 promotes the integration of telemedicine solutions that improve access to care services for people, like patients or adults. The primary objective is to transport and deliver telemedicine to elderly individuals residing in designated rooms at the Alia site. For this purpose, the TurtleBot3Burger robot is used, which is an inbuilt robot in Webots. Here, the selected robot task is to navigate autonomously between simulation world rooms upon receiving a command from a specific room. At this stage task is to develop of simulation environment (floor world) and test the built-in robot within the simulation world. Normally this selected robot moves autonomously between the rooms from avoiding the obstacles on the simulation floor world.Here, my tasks were to build this environment and robot movement optimization inside the patient's room.[1]

3. Design and Technical Details

3.1. Design the Initial Simulation world

Majorly, there are two tasks to achieve. The first one is to create a one-floor simulation world for the Alia site using Webots.This task is primarily involved in modeling and configuration within the existing software. Using Webots editor 3D layout including walls, rooms, doors, and corridors was constructed.The physical properties of this Alia site environment, like wall rigidity, floor friction, and collision boundaries, were accurately configured to ensure realistic robot simulation environment interactions. [3]

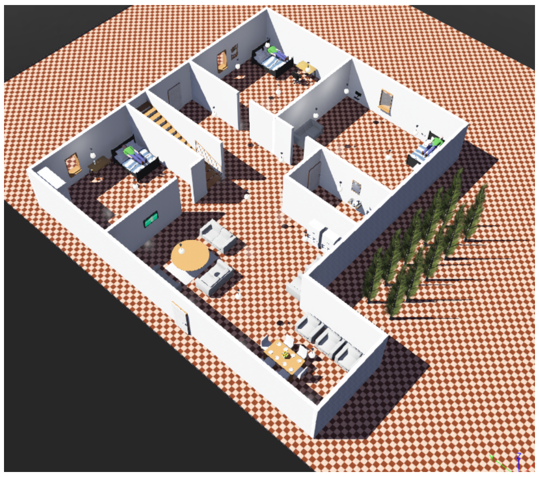

Figure 1. One Floor Alia Site

TurtleBot3Burger comes with a compact design, accurate maneuverability, and an inbuilt 360-degree LIDAR sensor which helps to precisely detect obstacles and environment mapping. Python-based controllers were used to define the robot's behavior, leveraging Webots real-time simulation. During development, smooth navigation through the Webots world was solved through iterative testing and adjustment of the values in the Python controller itself.

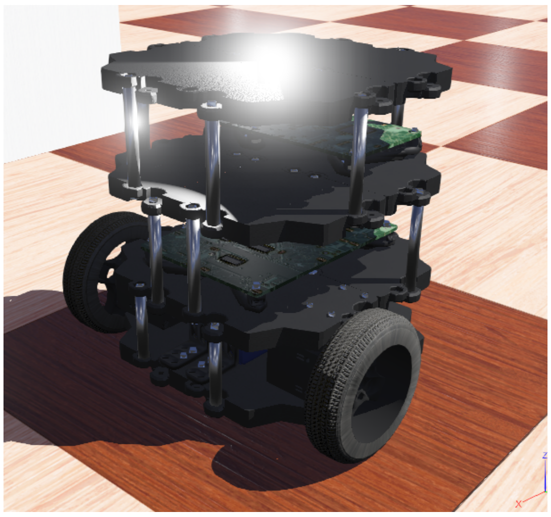

Figure 2. TurtleBot3Burger Robot

3.2. Iterative Path Planning for the Robot

After developing the Webots simulation environment, the next task was to implement diagonal movement for the TurtleBot3 Burger robot. The robot's navigation was initially tested within a single room, moving from one point to another like cell to cell movement. In the first test, the robot started at the door and moved straight ahead, then turned 90 degrees to reach the second point, which was the location of the patient's bed. Initially, the basic movement was implemented, which is basic cell-to-cell movement, and afterward, the diagonal movement approach was improved.[2]

3.2.1. Basic Cell-to-Cell Movement

Initially used the grid system where the robot could only move left, right, forward, or backward. The simulation floor was divided into cells, and the robot moved by going from the center of one cell to the center of a nearby square. The robot continued this scenario step by step. The sequence of the movement is as follows.

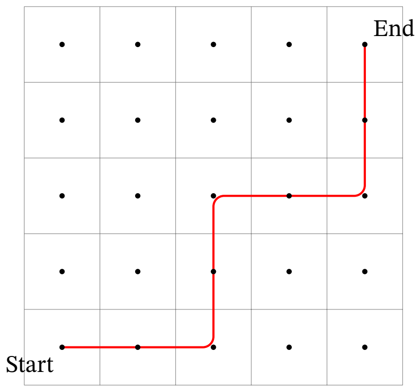

Figure 3. Path illustrating of cell to cell movement.

3.2.2. Improved Diagonal Movement

After cell-based movement, the controller method was improved to move the robot diagonally between cells. Here, the robot finds a new direct path to the next target by calculating a straight vector from its current spot to the target. While moving is keep checking and fixing small direction errors. When the robot gets close enough to the target position, it stops and moves on to the next target point.

Here supervisor access to read the robot's exact position and direction in the simulation.To find where the robot should face the target angle ($\theta_{desired}$) is calculated `atan2` function which is properly handles all quadrands.

$$\theta_{desired} = \text{atan2}(\Delta y, \Delta x)$$

where Δx and Δy are the differences in the x and y coordinates between the target and the robot's current position. The robot then turns until its current orientation (θ_{current}) is aligned with the desired orientation (θ_{desired}) within a defined tolerance.

This movement is smoother with fewer stops and starts than moving square by square. This movement is a little bit complex because the robot should calculate the different angles and keep adjusting its direction while moving. But this is efficient for diagonal routes because it shortens the path and reduces the turns.

In cell-based robot movement total time was calculated as 110 seconds to reach the patient's bed position, but in diagonal movement robot only took 40 seconds.

Code Snippets

The core logic for calculating the target orientation and managing the movement states in the provided Python controller is shown below:

```python

while robot.step(timestep) != -1:

    dx = target_pose["x"] - current_pose["x"]
    dy = target_pose["y"] - current_pose["y"]
    desired_theta = math.atan2(dy, dx)

    angle_error = math.atan2(math.sin(
    desired_theta - current_pose["theta"]),
                             math.cos(
    desired_theta - current_pose["theta"]))

    if abs(angle_error) > ANGLE_THRESHOLD:
        # Turn towards target
        left_motor.setVelocity(-TURN_SPEED if
    angle_error > 0 else TURN_SPEED)
        right_motor.setVelocity(TURN_SPEED if
    angle_error > 0 else -TURN_SPEED)
    else:
        distance = math.sqrt(dx**2 + dy**2)
        if distance > DISTANCE_THRESHOLD:
            # Move diagonally towards target
            left_motor.setVelocity(MOVE_SPEED)
            right_motor.setVelocity(MOVE_SPEED)
        else:
            # Target reached
            left_motor.setVelocity(0.0)
            right_motor.setVelocity(0.0)
            current_waypoint_index += 1
```

Code 1. Core Diagonal Movement Logic

The robot uses a simple two-state system. Which are turning to target and moving to target. Once the robot is aligned, it moves forward until it reaches the target and then finds the next waypoint. This method ensures smooth and efficient diagonal movement.

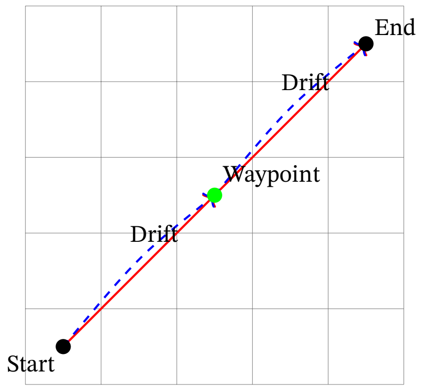

Figure 4. Diagonal movement of the robot with Drift

Since the robot is based on wheel encoders for position tracking, small errors in robot's wheel rotation and surface contact can cause the robot to drift from its desired path. Specially longer diagonal moves drift happen .
Diagonal movement gives a shorter, smoother, and more continuous path compared to cell to cell-to-cell method. This allows the robot to orient and travel toward the target position. From this approach, robots reduce travel distance and time, minimize the acceleration and deceleration cycles, and rotations. This study shows how mobile robots naturally move in open spaces.

4. Conclusion

First task to build the Alia simulation world using Webots simulation software. The next task was to implement and optimize the robot's movement inside the simulation world room. Here, two path planning methods were used: strict cell-to-cell movement and diagonal movement. The cell-based approach provides a fundamental grid-based approach, and the diagonal movement approach enhances the robot's movement. It reduces the stops and turns and makes more movement smoother for the robot. This study shows that the diagonal movement strategy is superior for navigating within open room environments as required by the task.

References

[1] G. A. Vargas, O. G. Rubiano, R. A. Castillo, O. F. Avilés, and M. F. Mauledoux, "Simulation of e-puck path planning in webots", 2016. [Online]. Available: https://www.researchgate.net/publication/329684570_Simulation_of_e-puck_path_planning_in_webots.

[2] K. Yakovlev, E. Baskin, and I. Hramoin, "Grid-based angle-constrained path planning", in *Swarm Intelligence*, ser. Lecture Notes in Computer Science, Springer, Cham, 2016. [Online]. Available: https://arxiv.org/pdf/1506.01864.

[3] O. Michel, "Cyberbotics ltd - webots^{TM}: Professional mobile robot simulation", *International Journal of Advanced Robotic Systems*, [Online]. Available: https://arxiv.org/pdf/cs/0412052.