Report: Simulation of an Autonomous Robot in a Warehouse

1. Introduction

The objective of this project is to simulate an autonomous robot navigating a warehouse, following specific constraints such as speed, stopping behavior, boundary restrictions, and obstacle avoidance. The simulation is written in Python and visualizes the robot's real-time movements from its starting position to its destination within the warehouse.

2. Problem Statement

The robot starts at position (0, 0) in a 10x10 meter rectangular warehouse and needs to navigate to the destination at (7, 9). The robot has a speed of 0.1 m/s and must stop for 2 seconds after moving for 0.1 seconds. Additionally, the robot needs to avoid any obstacles in its path and remain within the warehouse boundaries.

3. Approach

a. Warehouse Representation

The warehouse is represented as a 10x10 grid, where the robot starts at the bottom-left corner and moves towards the destination at (7, 9).

b. Robot Movement

The robot moves step-by-step towards the destination, making a movement every 0.1 seconds and stopping for 2 seconds afterward. The robot calculates its direction based on the destination, and its movement is visualized in real-time.

c. Obstacle Handling (Optional)

The robot checks for obstacles in its path. If an obstacle is detected, the robot changes its path slightly to avoid a collision. Obstacles can be represented by black squares in the warehouse visualization.

d. Visualization

The simulation is visualized using the matplotlib library. The robot's position is plotted in real-time as it moves toward the destination. Obstacles (if any) are represented by black squares, the robot is a red dot, and the destination is marked by a green cross.

e. **code**

```
import numpy as np

import matplotlib.pyplot as plt


# Warehouse dimensions

warehouse_length = 10

warehouse_width = 10


# Robot settings

robot_speed = 0.1

movement_step = 0.1

robot_pause = 2
```

```
# Starting and destination positions
start_position = np.array([0, 0], dtype=float)
destination_position = np.array([7, 9], dtype=float)

# Function to plot the warehouse and robot's position
def plot_warehouse(robot_position, destination):
    plt.imshow(np.zeros((warehouse_length, warehouse_width)), cmap='Blues', origin='lower')
    plt.plot(robot_position[1], robot_position[0], 'ro', label='Robot')
    plt.plot(destination[1], destination[0], 'gx', label='Destination')
    plt.title(f"Robot's Position: {robot_position}")
    plt.legend()
    plt.grid(True)

# Function to move the robot
def move_robot(start, destination):
    current_position = start.copy()

    # Create the plot figure
    plt.figure(figsize=(6, 6))

    # Continue until the robot reaches the destination
    while not np.allclose(current_position, destination, atol=0.1):
        # Calculate the direction vector (normalized)
        direction = destination - current_position
        distance = np.linalg.norm(direction)

        # Normalize the direction vector and move the robot by 0.1 meters per step
        if distance > movement_step:
            direction /= distance
            current_position += direction * movement_step
        else:
            # If the robot is very close, move it directly to the destination
            current_position = destination.copy()
```

```
        # Ensure the robot stays within the warehouse boundaries
        current_position = np.clip(current_position, [0, 0], [warehouse_length - 1, warehouse_width - 1])

        # Visualize the movement
        plot_warehouse(current_position, destination)
        plt.pause(0.1)  # Update the plot for smooth movement

        # Pause for 2 seconds to simulate the robot stopping
        plt.pause(robot_pause)

    print("Robot has reached the destination!")
    plot_warehouse(current_position, destination)
    plt.show()

move_robot(start_position, destination_position)
```

Screenshot