Potential Field Path Planner

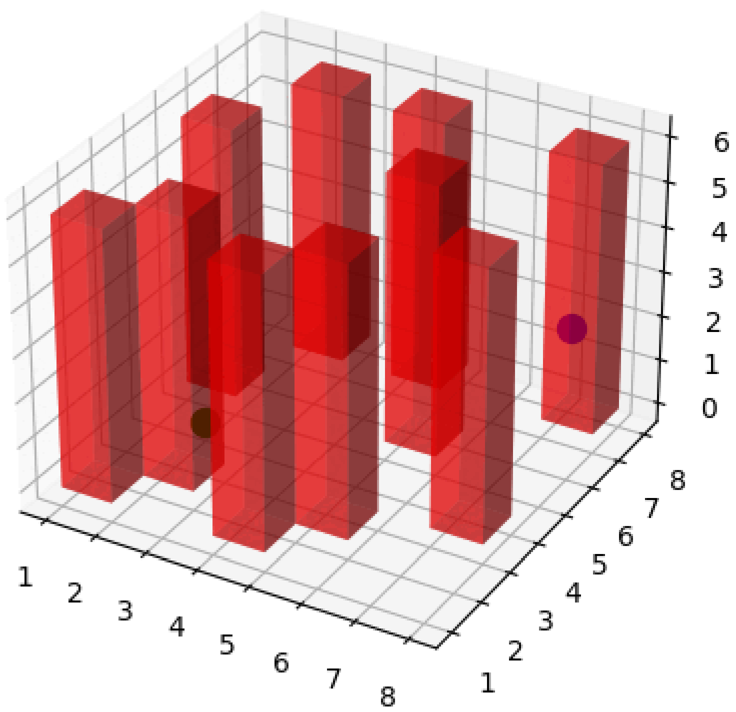
This project involves the design, build, testing, and optimization of a 3D potential field path planner for robotics applications. The planner navigates a robot from a start position to a goal position while avoiding obstacles using attractive and repulsive forces. This project showcases my engineering and programming skills and reflects my passion for robotics and learning.

Features

- **Attractive Force**: Guides the robot towards the goal.
- **Repulsive Force**: Keeps the robot away from obstacles.
- **Path Planning**: Generates a path from start to goal while avoiding obstacles.
- **Visualization**: 3D visualization of the path and obstacles.
- **Parameter Optimization**: Finds the best attractive and repulsive gains for the planner.

Screenshot

The following is the result of the potential field path planner navigating an agent from a start position to a goal position while avoiding obstacles.



Please excuse the fact that the path is obscured by the obstacles in the saved visualization. If I have time, I will fix this issue.

Installation

Ensure you have Python and the required libraries installed:

```
pip install numpy matplotlib
```

Usage

Example Code

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation

# Define the PotentialFieldPathPlanner class and other functions here

# Example usage
obstacles = [
    (2, 2, 0, 1, 1, 6),
    (5, 2, 0, 1, 1, 6),
    (1, 5, 0, 1, 1, 6),
    (5, 5, 0, 1, 1, 6),
    (2, 7, 0, 1, 1, 6),
    (7, 3, 0, 1, 1, 6),
    (4, 1, 0, 1, 1, 6),
    (1, 1, 0, 1, 1, 6),
    (4, 7, 0, 1, 1, 6),
    (7, 7, 0, 1, 1, 6)
]

start = (2, 4, 0)
goals = [(8, 6, 3), (1, 8, 0), (3, 1, 0)]

attractive_gains = np.linspace(1.0, 2.0, 5)
repulsive_gains = np.linspace(0.1, 2.0, 5)

best_gains, best_fitness = find_best_gains(obstacles, start, goals,
attractive_gains, repulsive_gains)
print(f"Best gains: Attractive Gain = {best_gains[0]}, Repulsive Gain =
{best_gains[1]}")
print(f"Best MSE Cost: {best_fitness}")

# Visualize the path for the best gains and the first goal
planner = PotentialFieldPathPlanner(obstacles, start, goals[0], best_gains[0],
best_gains[1])
path, cost = planner.plan_path()
visualize_path(obstacles, path, start, goals[0])
```

```
    # Visualize the path for the best gains and each goal
    for goal in goals:
        planner = PotentialFieldPathPlanner(obstacles, start, goal, best_gains[0],
    best_gains[1])
        path, cost = planner.plan_path()
        visualize_path(obstacles, path, start, goal)
```

Visualization

The `visualize_path` function provides a 3D visualization of the path planned by the robot, including the start and goal positions and the obstacles.

Mathematical Concepts

Attractive Force

The attractive force $\mathbf{F}_{att}$ pulls the robot towards the goal. It is typically modeled as a linear function of the distance to the goal:

$\mathbf{F}{att} = k{att} \cdot (\mathbf{q}_{goal} - \mathbf{q})$

where:

- k_{att} is the attractive gain.
- $\mathbf{q}_{goal}$ is the position of the goal.
- $\mathbf{q}$ is the current position of the robot.

Repulsive Force

The repulsive force $\mathbf{F}_{rep}$ pushes the robot away from obstacles. It is usually modeled to have an effect only within a certain distance d_0 from the obstacle:

$$
\mathbf{F}{rep} =
\begin{array}{ll}k{rep} \left(\frac{1}{\mathbf{q}} - \frac{1}{d_0} \right) \frac{1}{\mathbf{q}^2}
\frac{\mathbf{q} - \mathbf{q}{obs}}{|\mathbf{q} - \mathbf{q}{obs}|}, & \text{if } |\mathbf{q} - \mathbf{q}
{obs}| \leq d_0 \
0, & \text{if } |\mathbf{q} - \mathbf{q}{obs}| > d_0
\end{array}
$$

where:

- k_{rep} is the repulsive gain.
- $\mathbf{q}_{obs}$ is the position of the obstacle.
- d_0 is the influence distance of the obstacle.

Total Force

The total force $\mathbf{F}$ acting on the robot is the sum of the attractive and repulsive forces:

$$\mathbf{F} = \mathbf{F}{att} + \mathbf{F}{rep}$$

Path Planning

The robot's movement is determined by the total force. At each time step, the robot's position is updated based on the force:

$$\mathbf{q}_{new} = \mathbf{q} + \Delta t \cdot \mathbf{F}$$

where Δt is the time step.

Optimization

To find the best gains k_{att} and k_{rep}, we minimize a cost function, typically the Mean Squared Error (MSE) between the planned path and the desired path:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} |\mathbf{q}{i} - \mathbf{q}{goal}|^2$$

where N is the number of steps in the path.

Skills Demonstrated

- **Programming**: Proficient use of Python, including libraries such as NumPy and Matplotlib.
- **Robotics Engineering**: Implementation of potential field path planning, a fundamental concept in robotics navigation.
- **Optimization**: Finding the best parameters for the path planner to minimize the cost function.
- **3D Visualization**: Using Matplotlib to visualize the robot's path and obstacles in a 3D space.

Contributing

Contributions are welcome! If you have suggestions for improvements or new features, please open an issue or submit a pull request.

Contact

For any questions or inquiries, please contact ronen.g.aniti@gmail.com.