# Obstacle Avoidance Using Evolutionary Strategies in a Simulated Gazebo Environment

Zeren Li

*Master of Science*

University of Maryland,
College Park, USA

zli06211@umd.edu

Yi Wei

*Master of Science*

University of Maryland,
College Park, USA

ywei1234@umd.ed

## Abstract

**This study explores obstacle avoidance for mobile robots using an Evolutionary Strategy (ES) algorithm. Leveraging sensor data from LIDAR and IMU, the robot adapts motor speed parameters for optimal navigation while avoiding obstacles. Simulations conducted in the Gazebo environment demonstrated a 40% improvement in fitness scores over just 10 generations, validating the effectiveness of ES in such tasks. The project emphasizes safety and efficiency, showing significant time savings through simulation acceleration techniques.**

**Keywords—Evolutionary Strategy, Obstacle Avoidance, ROS2, Gazebo**

## I. Introduction

Obstacle avoidance is a fundamental capability for robots, enabling them to navigate through environments by detecting and avoiding obstacles. In a real-world environment, a robot would use sensors such as LIDAR, cameras, or ultrasonic rangefinders to detect the objects around it and convert the output into quantifiable data. In simulation platforms like Gazebo, it provides tools to simulate robots with accurate physics and sensors.

Obstacle avoidance can be implemented with many algorithms, such as A* or RRT. In this project, we adopted a reinforcement learning-inspired approach by utilizing Evolutionary Strategies (ES). ES evolves generations of robot control policies, optimizing motor speed parameters to effectively respond to obstacles detected in the robot's vicinity. This approach enables adaptive and efficient navigation by dynamically adjusting motor speeds based on the relative position of obstacles. Each generation in the Gazebo environment will be evaluated by a fitness function. The top-performing generation will be selected for reproduction, and new offspring will be generated through crossover and mutation. Each generation's fitness will be measured by the robot's ability to avoid obstacles while minimizing time and energy spent navigating the environment. Over successive generations, the algorithm will evolve towards an optimal obstacle-avoidance strategy.

The primary objective of this project is to design and train a robot in the Gazebo simulation environment to effectively avoid obstacles using an evolutionary algorithm. The goal is to develop an adaptive, robust, and efficient navigation strategy that improves over time based on performance metrics such as turning time used on avoiding obstacles and path efficiency.

## II. Literature Survey

Evolutionary algorithms (EAs) have attracted much attention for their robustness, adaptability, and ability to optimize complex nonlinear problems. This review explores the recent advances and main contributions of evolutionary algorithms for robotic obstacle avoidance applications. Evolutionary algorithms are meta-heuristic optimization techniques inspired by natural selection and biological evolution. The main types of evolutionary algorithms include genetic algorithms (GA), particle swarm optimization (PSO), differential evolution (DE), and evolutionary strategies (ES). These algorithms excel at optimizing high-dimensional spaces and can handle dynamic environments, making them suitable for robot navigation tasks.

Here are some applications of EAs in Robot Obstacle Avoidance.

### 2.1 Genetic Algorithms (GA)

Genetic algorithms are widely used in obstacle avoidance because of their ability to evolve optimal solutions through selection, crossover and mutation. Recent studies have highlighted their effectiveness:

Path planning optimization: Zhang et al. showed that GA can be used to optimize robot trajectories in dynamic environments [1]. Their approach minimizes the path length while avoiding obstacles by encoding the robot path as a chromosome and applying an adaptive mutation rate.

### 2.2 Particle Swarm Optimization (PSO)

PSO is inspired by the social behavior of birds and fish and is particularly effective for multi-robot systems:

Multi-robot coordination: Raibail et al. used PSO for decentralized obstacle avoidance in multi-robot systems [2]. The approach ensures collision-free navigation while maintaining formation by optimizing a shared objective function.

2.3 Bacterial Foraging Optimization Algorithm (BFOA)

Inspired by the foraging behavior of E. coli bacteria, BFOA emphasizes maintaining formation and minimizing computational time during navigation.

It is particularly effective in situations requiring tight formation and precise coordination, but struggles with pathfinding in highly complex environments [3].

## III. Methodology

### Algorithm Design

- Input: LIDAR sensor data (msg.ranges[0:20], msg.ranges[340:359]).
- Output: Motor speed parameters (fast_turn, left_turn, right_turn).
- Policy: Threshold-based navigation, turning left or right when obstacles are detected at right or left respectively. If both sensors detect obstacles, then turn left at a faster speed. It will also be accelerating forward in clear paths.

We use lidar sensor data as input. Since the output of the sensor is an array of 360-degree directions representing the distance to the nearest object one full circle clockwise from directly in front. So we take [0, 20], which is the left side of the front as the value for the left sensor, and [340:359], which is the right side of the front as the value for the right sensor. We define three actions: left turn, right turn, and fast turn. The robot turns to the left when the right sensor detects the obstacle, and vice versa for obstacles detected on the left. When both sensors detect the obstacle, which means the obstacle is directly in front of the robot, it should take a fast turn to avoid the collision. We defined two thresholds for obstacle detection in this scenario: a larger threshold to identify obstacles at a safe distance and a smaller threshold for obstacles that are dangerously close to the robot. The smaller threshold is associated with a high penalty to discourage unsafe maneuvers and prioritize collision avoidance.

### General Workflow

1. **Initialize Population:** Create an initial set of motor speed parameters for each action.
2. **Simulation:** Run each individual (parameter set) in a simulation environment where the robot navigates using the predefined rules.
3. **Evaluate Fitness:**
   - Measure the fitness of each individual based on how well it

avoids obstacles and minimizes turning time while navigating.
    - Collision Penalty: Assign a high penalty if the vehicle collides with an obstacle.
    - Turning Penalty: Apply a small penalty for time spent turning to encourage efficient obstacle avoidance.
    - Forward Movement Reward: Reward continuous forward movement without collisions.

$$\text{Fitness} = \alpha \times \text{Time without collision} - \beta \times \text{Time spent turning} - \gamma \times \text{Collision penalty}$$

4. **Select and Evolve:**
    - Select the top-performing individuals.
    - Optionally apply recombination to combine the best solutions.
    - Apply mutation to create new motor speed.
5. **Repeat:** Iterate through generations until an optimal solution is found or a set number of generations are completed.
6. **Deploy:** Once an optimal parameter set is found, deploy it on the robot for real-world testing.

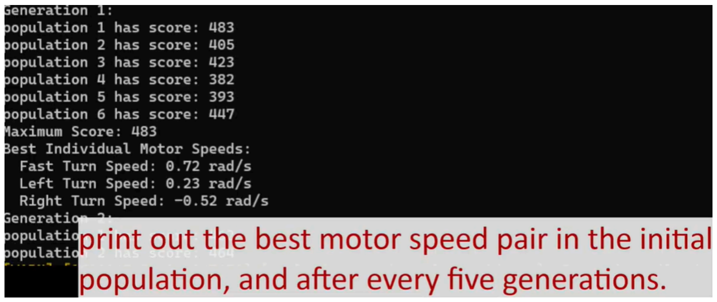
## IV. Contribution

- Repository: GitHub-Zli868-Obstacle_avoidance
- Modified Gazebo configurations for simulation speedup.
- Implemented ES algorithm for parameter optimization.
- Devised a fitness function balancing speed, safety, and efficiency.

## V. Results

### Training model

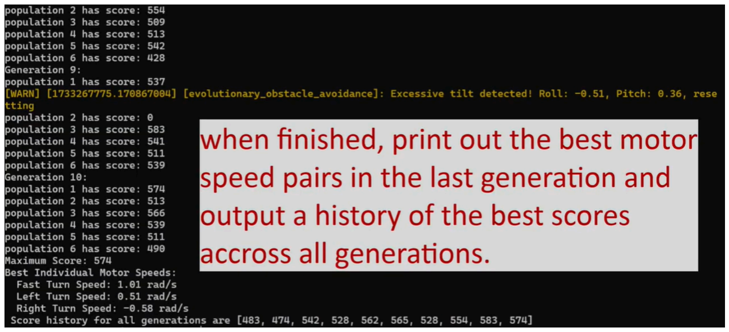
In the simulation environment, we print out the best motor speed in the first generation.



```
eneration 1:
opulation 1 has score: 483
opulation 2 has score: 405
opulation 3 has score: 423
opulation 4 has score: 382
opulation 5 has score: 393
opulation 6 has score: 447
aximum Score: 483
est Individual Motor Speeds:
 Fast Turn Speed: 0.72 rad/s
 Left Turn Speed: 0.23 rad/s
 Right Turn Speed: -0.52 rad/s
eneratio
opulatio
opulatio
```

print out the best motor speed pair in the initial population, and after every five generations.

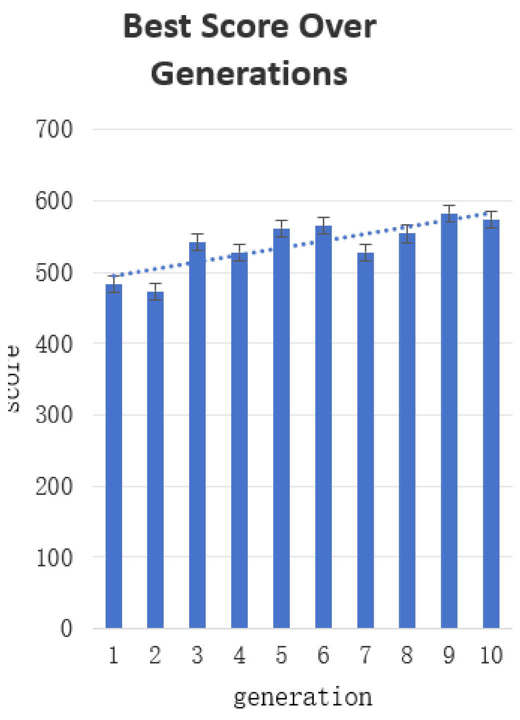
After all iterations of generations, A history of the scores across all generations gets printed out.



```
population 2 has score: 554
population 3 has score: 509
population 4 has score: 513
population 5 has score: 542
population 6 has score: 428
Generation 9:
population 1 has score: 537
[WARN] [1733267775.170867004] [evolutionary_obstacle_avoidance]: Excessive tilt detected! Roll: -0.51, Pitch: 0.36, rese
tting
population 2 has score: 0
population 3 has score: 583
population 4 has score: 541
population 5 has score: 511
population 6 has score: 539
Generation 10:
population 1 has score: 574
population 2 has score: 513
population 3 has score: 566
population 4 has score: 539
population 5 has score: 511
population 6 has score: 490
Maximum Score: 574
Best Individual Motor Speeds:
  Fast Turn Speed: 1.01 rad/s
  Left Turn Speed: 0.51 rad/s
  Right Turn Speed: -0.58 rad/s
 Score history for all generations are [483, 474, 542, 528, 562, 565, 528, 554, 583, 574]
```

when finished, print out the best motor speed pairs in the last generation and output a history of the best scores accross all generations.

It is also important to note that higher turning speeds enhance agility but come with an increased risk of instability. If a robot flips over, the IMU sensor detects abnormal orientation angles and triggers a reset of the robot's position, accompanied by a warning (displayed in yellow). This safety mechanism ensures stability during operation. The algorithm effectively identified motor speed parameters that optimize fitness while adhering to these defined safety constraints.



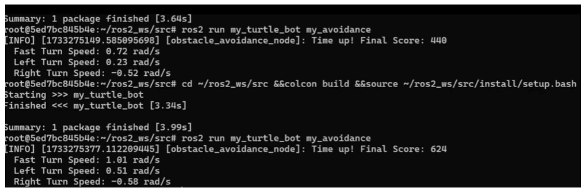
**Best Motor Speed (Last Generation-10)**

- **Fast Turn Speed**: 1.01 rad/s.
- **Left Turn Speed**: 0.51 rad/s.
- **Right Turn Speed**: -0.58 rad/s.

From the plot of the scores, we observe an increasing trend in scores as the generations progress, indicating the algorithm's ability to refine solutions and optimize performance over time.

**Simulation Outcomes**

Finally, we tested the speed parameters obtained from both the initial generation and the last generation in the simulation environment to evaluate performance.



```
Summary: 1 package finished [3.64s]
root@5ed7bc845b4e:~/ros2_ws/src# ros2 run my_turtle_bot my_avoidance
[INFO] [1733275149.585095698] [obstacle_avoidance_node]: Time up! Final Score: 440
  Fast Turn Speed: 0.72 rad/s
  Left Turn Speed: 0.23 rad/s
  Right Turn Speed: -0.52 rad/s
root@5ed7bc845b4e:~/ros2_ws/src# cd ~/ros2_ws/src &&colcon build &&source ~/ros2_ws/src/install/setup.bash
Starting >>> my_turtle_bot
Finished <<< my_turtle_bot [3.34s]

Summary: 1 package finished [3.99s]
root@5ed7bc845b4e:~/ros2_ws/src# ros2 run my_turtle_bot my_avoidance
[INFO] [1733275377.112209445] [obstacle_avoidance_node]: Time up! Final Score: 624
  Fast Turn Speed: 1.01 rad/s
  Left Turn Speed: 0.51 rad/s
  Right Turn Speed: -0.58 rad/s
```

- Initial population fitness: **440**.
- Final population fitness after 10 generations: **624**.
- **Score Improvement**: 40%.

**Observations**

The testing results reveal that the fitness scores closely align with those observed during training, validating the effectiveness and reliability of the Evolutionary Strategy (ES) algorithm. Over the course of ten generations, the fitness score improved significantly by approximately 40%, demonstrating the algorithm's capability to optimize robot control parameters for obstacle avoidance.

## VI. Conclusion

The project demonstrates the effectiveness of ES algorithms in optimizing motor speed parameters for obstacle avoidance tasks. By leveraging Gazebo's simulation environment and accelerating computation, this study highlights the balance between safety and efficiency in robotic navigation. Future work includes testing in real-world environments and extending the fitness function for dynamic obstacles.

## References

[1] Liu, Kaishu & Gu, Jijun & He, Xiaoyong & Zhang, Long. (2024). Optimization algorithms for dynamic environmental sensing and motion planning of quadruped robots in complex environments on unmanned offshore platforms. Measurement Science and Technology. 36. 10.1088/1361-6501/ad8fc2.

[2] Raibail, M., Rahman, A. H. A., AL-Anizy, G. J., Nasrudin, M. F., Nadzir, M. S. M., Noraini, N. M. R., & Yee, T. S. (2022). Decentralized Multi-Robot Collision Avoidance: A Systematic Review from 2015 to 2021. Symmetry, 14(3), 610. https://doi.org/10.3390/sym14030610

[3] D. Roy, M. Maitra and S. Bhattacharya, "Study of formation control and obstacle avoidance of swarm robots using evolutionary algorithms," 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest, Hungary, 2016, pp. 003154-003159, doi: 10.1109/SMC.2016.7844719.