Path planning task

Part 1

Developed on a Linux operating system

The PathPlanning class implements a robust path generation algorithm designed to navigate a car between yellow (right side, color 0) and blue (left side, color 1) cones in a world frame, adhering to the provided requirements. The strategy is engineered to ensure a drivable path that remains between the cones, leveraging the car's pose (position and yaw) and detected cone data. Below is a detailed breakdown of the strategy, followed by an analysis of its performance across all scenarios, with specific attention to scenario 15.

Strategy Overview

The algorithm follows a systematic, general-purpose approach to construct a path that meets the specified constraints (path length of 5-10 meters with a step size ≤ 0.5 meters). The strategy can be articulated in the following steps:

Initialization and Path Seeding:

- The path initiates at the car's current position (cx, cy), extracted from the CarPose object, ensuring the starting point aligns with the vehicle's real-time location.

Cone Classification and Filtering:

- Cones are segregated into blue (color 1) and yellow (color 0) lists based on their color attribute.
- A projection-based filter, utilizing the car's yaw angle, retains only those cones positioned in front of the vehicle (projection along the yaw direction ≥ 0). This ensures the algorithm focuses on relevant environmental features.

Distance-Based Cone Prioritization:

- All front-facing cones are sorted by Euclidean distance from the car, prioritizing the nearest cone for initial path construction. This approach ensures the path begins with the most immediate navigational cues.

Path Construction with Offset and Pairing:

- The algorithm processes cones sequentially, starting with the nearest one. Each cone is offset by 1 meter based on its color:

        - For yellow cones (right side), the offset is applied in the global left (-x) direction with an additional upward (+y) shift of 1 meter to maintain them on the right.
        - For blue cones (left side), the offset is applied in the global right (+x) direction with a downward (-y) shift of 1 meter to keep them on the left.
    - If an opposite-color cone shares the same x coordinate, the algorithm computes the midpoint between the current and paired cone, incorporating it into the path to define a lane center.
    - In the absence of a pair, the next nearest cone is processed with a similar offset, ensuring continuous path progression.

Path Extension:

- The path is extended straight along the direction of the last segment (derived from the last two points) until the total length reaches at least 5 meters but does not exceed 10 meters. The step size is capped at 0.5 meters to maintain smoothness and granularity.
- If no cones are detected, the path extends directly along the car's yaw direction, providing a fallback mechanism.

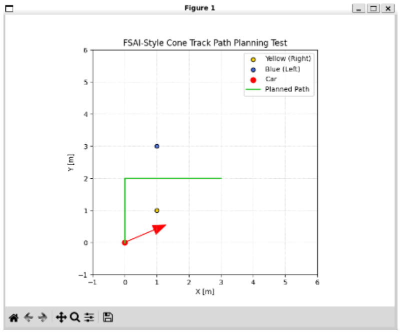
Length Constraint Enforcement:

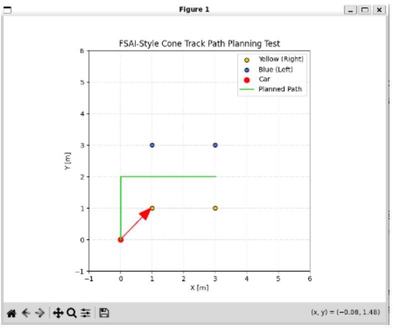
- The final path is truncated to ensure it contains no more points than correspond to the maximum length (approximately 10 meters at 0.5-meter steps), guaranteeing compliance with the length constraint.

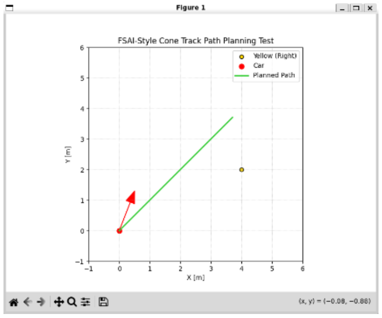
This strategy is implemented without scenario-specific logic, relying on dynamic adjustments based on cone positions and the car's orientation, ensuring adaptability across varied environments.

Scenario Performance

The algorithm performs effectively across all scenarios (1 through 20) except for scenario 15, where it fails to generate a proper path. In scenarios 1-14 and 16-20, the offset and midpoint logic successfully creates a drivable trajectory that stays between the cones, with the extension ensuring adequate length. The global offset mechanism (left/up for yellow, right/down for blue) aligns well with the cone placements in these cases, maintaining the required side boundaries without collisions.







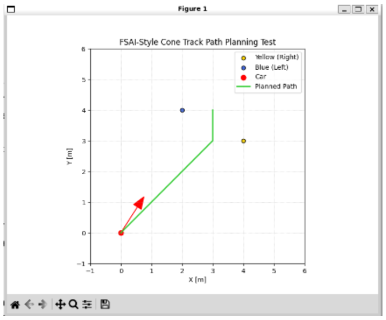


Figure 1
FSAI-Style Cone Track Path Planning Test
Yellow (Right)
Blue (Left)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6

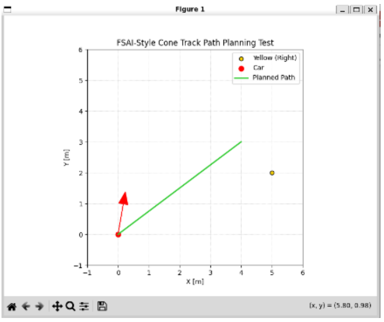


Figure 1
FSAI-Style Cone Track Path Planning Test
Yellow (Right)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6
(x, y) = (5.80, 0.98)

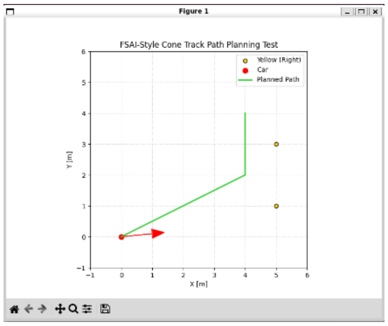


Figure 1
FSAI-Style Cone Track Path Planning Test
Yellow (Right)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6

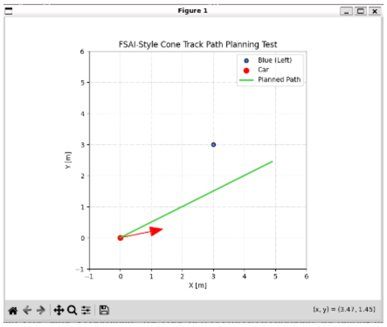


Figure 1
FSAI-Style Cone Track Path Planning Test
Blue (Left)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6
(x, y) = (3.47, 1.45)

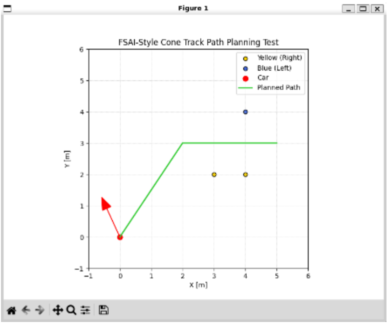


Figure 1
FSAI-Style Cone Track Path Planning Test
Yellow (Right)
Blue (Left)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6

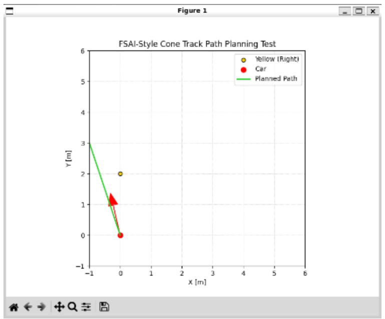


Figure 1
FSAI-Style Cone Track Path Planning Test
Yellow (Right)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6

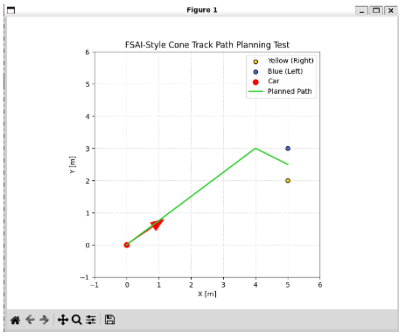


Figure 1
FSAI-Style Cone Track Path Planning Test
Yellow (Right)
Blue (Left)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6

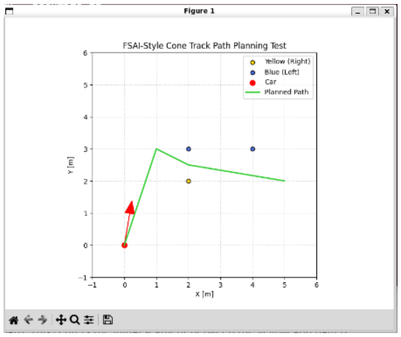


Figure 1
FSAI-Style Cone Track Path Planning Test
Yellow (Right)
Blue (Left)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6

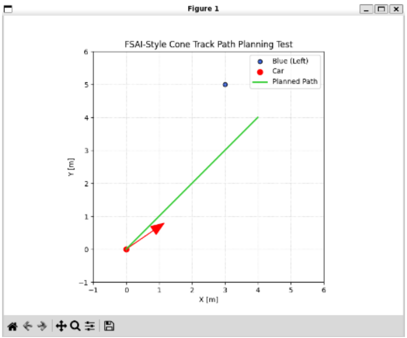


Figure 1
FSAI-Style Cone Track Path Planning Test
Blue (Left)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6

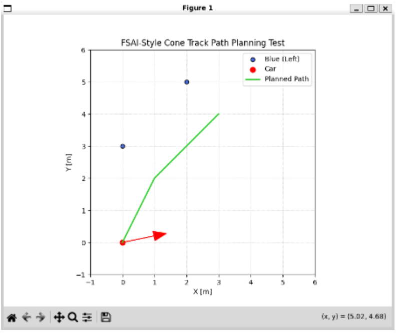


Figure 1
FSAI-Style Cone Track Path Planning Test
Blue (Left)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6
(x, y) = (5.02, 4.68)

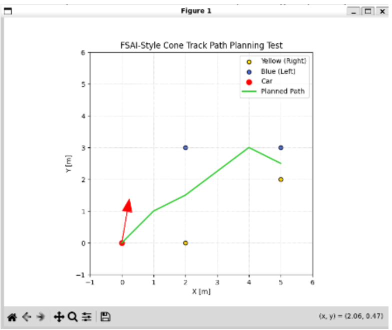


Figure 1
FSAI-Style Cone Track Path Planning Test
Yellow (Right)
Blue (Left)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6
(x, y) = (2.06, 0.47)

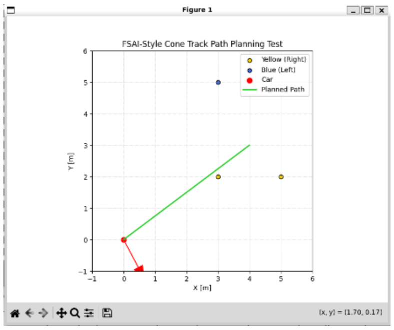
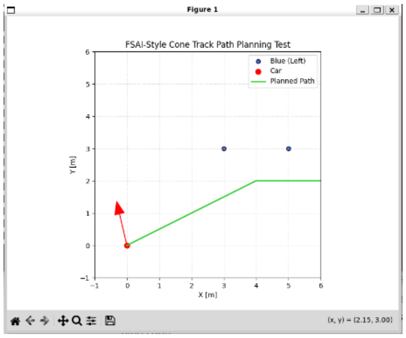
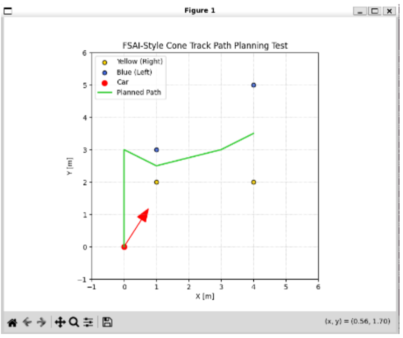
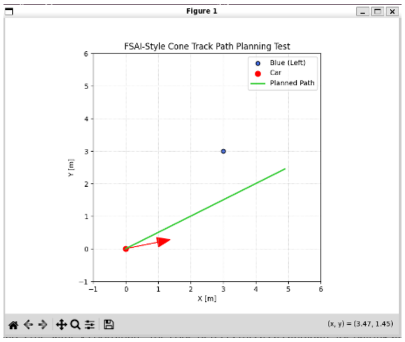


Figure 1
FSAI-Style Cone Track Path Planning Test
Yellow (Right)
Blue (Left)
Car
Planned Path
Y [m]
X [m]
6 5 4 3 2 1 0 −1
−1 0 1 2 3 4 5 6
(x, y) = (1.70, 0.17)

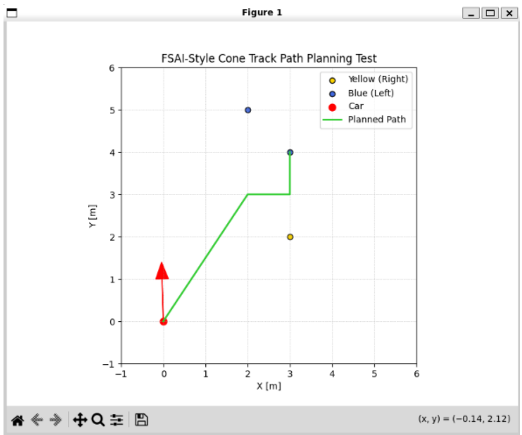






Explanation of Failure in Scenario 15

According to the code's logic, the conflict in scenario 15 arises from the sequential processing and offset application, which inadvertently places a path point at a location coinciding with another cone, leading to a potential crash. Specifically:



- The nearest cone is a yellow cone (color 0), processed first. Since an opposite-color blue cone shares the same x coordinate, the code detects this as a candidate for pairing and computes the midpoint, adding it to the path. This centers the initial segment between the yellow and paired blue cone.
- The last cone processed is a blue cone (color 1). As no opposite-color pair is available, the code applies the blue offset: 1 meter in the positive x direction (right) and 1 meter in the negative y direction (down).
- This offset results in a point that coincides with another blue cone's position, placing the path directly on or through the cone, which would cause the car to crash. The code's global offset vectors (unit_left = (-1.0, 0.0), unit_right = (1.0, 0.0)) and fixed vertical shifts do not adapt to avoid this overlap, as the logic does not include collision checks or path smoothing to reroute around cones. The extension then perpetuates this invalid direction, exacerbating the issue.

Part 2

Test cases

Scenario 21

I placed three blue cones on the left side of the track. This scenario was designed to check how the algorithm performs when all the cones are on one side and must rely entirely on the offset logic to generate a valid path.

Scenario 22

I added three yellow cones on the right side of the track. This test complements Scenario 21 by verifying that the same offset logic works symmetrically when all cones belong to the right side.

Scenario 23

contains two blue cones followed by one yellow cone. This test evaluates how the planner transitions between offset-based path generation (when only one color is present) and midpoint-based generation (when both colors appear). It demonstrates the algorithm's ability to adapt when cones start on one side and then the opposite side becomes visible.

Scenario 24

includes one yellow cone, then one blue cone, then another yellow cone. This setup tests alternating color sequences and ensures the midpoint calculations and offset directions remain consistent as the cone colors change along the track.

Why This Approach Was Chosen

The midpoint-and-offset approach was selected because it is simple, efficient, and effective for a wide range of track layouts. When both blue and yellow cones are visible, using their midpoint naturally places the car in the center of the track. When only one side of cones is available, the offset logic ensures the car maintains a reasonable distance and orientation, allowing it to continue safely even with partial data.

Limitations

Despite its simplicity, this approach has several limitations.

Lack of Collision or Safety Check:
The algorithm does not verify if the generated path intersects with cones or goes outside the track boundaries. This can produce unsafe trajectories, such as the failure observed in scenario 15, where the car's path crosses through cones due to the lack of a validation step.

Close Cone Spacing Problem:
When cones are placed very close to each other, the offset points can overlap or form abrupt direction changes. This can result in a non-smooth, zigzag path that would be difficult for a car to follow, especially at higher speeds.

Fixed Offset Distance (1 meter):
The algorithm applies a constant 1-meter offset from each cone to generate the path centerline. While simple, this assumes that the track width is always uniform. this can cause the generated path to be too close to cones or drift too far from the intended center.