Introduction

The task of finding the boundary of FOV of a car on race track can be formulated as follows:

To simplify the problem, consider the FOV at (s, n) is the same for all n within the boundary of the track, and consider the boundary of FOV is a straight line perpendicular to the center line of the track at position $s_{fov}(s)$.

Denote the position of the point at the center line at s in Cartesian by (x_s, y_s). To find the position $s_{fov}(s)$, one first finds the nearest point (x_{tan}, y_{tan}) on the boundary line of the track such that it is ahead of the position s and the line connecting (x_s, y_s) and (x_{tan}, y_{tan}) is tangent to this boundary line. Then the intersection of the tangent line and the center line is the $s_{fov}(s)$ we are looking for.

In discrete data (i.e., a series of vertices), we must find an optimal vertex to serve as an effective approximation of a "tangent point."

Core Idea: The boundary vertex that best represents the "tangent point" is the most "marginal" point that the line of sight can reach. This "margin" can be described precisely by an angle.

Definition: Inputs and Objective

Inputs

1. **centerline_points**: An ordered list of vertices for the centerline, e.g., $[(x_0, y_0), (x_1, y_1), \dots]$.
2. **boundary_points**: An ordered list of vertices for **one** boundary (e.g., the left boundary), e.g., $[(b_{x_0}, b_{y_0}), (b_{x_1}, b_{y_1}), \dots]$.

Objective

From the centerline_points and boundary_points, find the continuous function $s_{fov}(s)$.

Algorithm Steps:

Step 1: The Angular Maximization/Minimization Principle

This is the most critical step in determining the discrete "tangent point."

1. Iterate through all vertices T_j(from near to far) in left (right) boundary that are ahead of (x_i, y_i).
2. Find the heading vector P_i of (x_i, y_i) by $P_i = (x_{i+1}, y_{i+1}) - (x_i, y_i)$
3. For each T_j, calculate the vector from (x_i, y_i) to itself: $\mathbf{V}_i = T_j - (x_i, y_i)$.
4. Calculate the **signed angle** θ_i between $\mathbf{V}_i$ and P_i. ($\mathbf{V}_i$ on the left of P_i is positive angle)
5. **Apply the rule based on the boundary's position**:
    - If processing the **left boundary**, the tangent point T to be found is the vertex that locally **minimizes** the angle θ_i.
    - If processing the **right boundary**, the tangent point T to be found is the vertex that locally **maximizes** the angle θ_i.

The intuition behind this principle is that as your line of sight sweeps across the boundary, the point with the maximum (or minimum) angle is the critical point where the sightline is about to be occluded by the boundary itself. This is the best representation of a tangent point in a discrete environment.

Step 2: Subsequent Processing

Once found the tangent points T_{left} and T_{right} for the left and right boundaries using the method above, you can continue with the previous algorithm:

1. Construct the rays passing through $((x_i, y_i), T_{\text{left}})$ and $((x_i, y_i), T_{\text{right}})$.
2. Calculate the intersection points of these two rays with the centerline polyline segments, yielding $s_{fov,\text{left}}$ and $s_{fov,\text{right}}$.
3. Compare them to get s_{fov}.

Step 3: Interpolating the boundary of FOV

Once get the discretized correspondence between s and s_{fov}. use linear interpolation to find the continuous function $s_{fov}(s)$.