Assignment: Path Smoothing and Trajectory Control in 2D Space

Objective:

Implement a path smoothing algorithm and a trajectory tracking controller for a differential drive robot navigating through a series of 2D waypoints. The goal is to generate a smooth trajectory from discrete waypoints and ensure the robot follows this trajectory accurately.

Extra Credits: Extend your solution to avoid obstacles

You're encouraged to actively use AI-powered tools—such as GPT, Claude, Cursor AI, Lovable, Replit, and others—to support your development workflow. The task is intentionally challenging but fully manageable within the given timeframe when modern AI solutions and best practices are effectively applied.

Background:

Robots often receive a coarse path from a global planner, consisting of discrete waypoints. To enable smooth and safe motion, this path must be refined into a continuous, smooth trajectory. Moreover, the robot needs a controller that can follow the generated trajectory.

Assignment Tasks:

1. Path Smoothing

Given a list of 2D waypoints:

waypoints = [(x0, y0), (x1, y1), ..., (xn, yn)]

Implement a smoothing algorithm to generate a smooth trajectory

✔️ **Deliverables:**

- A function that takes discrete waypoints and returns a smooth, continuous path.

2. Trajectory Generation

From the smoothed path, generate a time-parameterized trajectory:

- Sample the path at regular intervals
- Optionally assign velocity profiles (e.g., trapezoidal or constant velocity)

✔️ **Deliverables:**

Time-stamped trajectory in the form:

trajectory = [(x0, y0, t0), (x1, y1, t1), ..., (xn, yn, tn)]

3. Trajectory Tracking Controller

Implement a controller that makes a simulated robot follow the trajectory

✔️ **Deliverables:**

- A controller function that outputs velocity commands based on current state and trajectory
- Simulation showing robot tracking performance

Submission Requirements:

1. Code Repository:
    1.1. A self contained C++ (or any other programming language) using ROS2 implementing the assignment tasks with simulation on a differential drive robot (Turtlebot3 or any other platform)
    1.2. Make sure the code is well documented and modular taking into account good architectural practices
2. Documentation:
    2.1. A Readme file with clear setup and execution instructions
    2.2. Explain your design choices, algorithms, architectural decisions
    2.3. Explain how would you extend this to a real robot
    2.4. Brief on the AI tools used (If applicable)
    2.5. Extra Credit: How would you extend this to avoid obstacles
3. Demonstration Videos (3-5 mins):
    3.1. Presenting the system in action
    3.2. Showing the plots/ profiles any other results

Evaluation Criteria:

Component	Sub Component	Points
Code Quality		35
	Trajectory Generation & Smoothing (15)	
	Controller (20)	
Code Architecture & Comments		10
Simulation		20
Testability & QA		20
	Test Case Design (10)	
	Test Automation (5)	
	Error Handling (5)	
Documentation, Reports & Video demonstration		15
Total		**100**