
Lecture 9: Trajectory Optimization

Luca Carlone
Planning vs. Control

- **Planning**
  - Desired trajectory
  - Map, points A, B

- **Controller**
  - Error
  - State estimate
  - Control inputs

- **Robot**
  - Robot's state

- **Sensors**
  - Map

Diagram:
- Map with obstacles:
  - Points A, B
  - Path from A to B
  - Obstacles identified

Flowchart:
- Planning to Controller with error feedback to Robots state estimation to Sensors with map feedback.
Decoupled Trajectory Planning

Map, points A, B → Waypoints → Desired Trajectory → Path Planning → Trajectory Optimization → Controller → Robot → Sensors

Diagram showing a decoupled trajectory planning process with obstacles on the map, and desired waypoints A and B leading to trajectory optimization and controller output for robot movement.
Decoupled Trajectory Planning

- (current) map, start/end points
- Path planning
- Trajectory optimization
- desired trajectory

Decoupled Trajectory Planning
Decoupled Trajectory Planning

Path planning → waypoints → Trajectory optimization → desired trajectory

(current) map, start/end points → Decoupled Trajectory Planning
Decoupled Trajectory Planning

- Need to enforce "continuity" between segments for smooth trajectory
Continuity constraints for trajectory opt.

\[ x_1(t_1) = x_2(t_1) = \bar{x}_1 \]
\[ \dot{x}_1(t_1) = \dot{x}_2(t_1) \]
\[ \ddot{x}_1(t_1) = \ddot{x}_2(t_1) \]

\[ x_2(t_2) = x_3(t_2) = \bar{x}_2 \]
\[ \dot{x}_2(t_2) = \dot{x}_3(t_2) \]
\[ \ddot{x}_2(t_2) = \ddot{x}_3(t_2) \]

\[ x_1(t_0) = \bar{x}_0 \]
\[ \dot{x}_1(t_0) = \bar{v}_0 \]
Decoupled Trajectory Planning

- Need to enforce “continuity” between segments for smooth trajectory
- Need to ensure “minimal” motion
Decoupled Trajectory Planning

- Need to enforce "continuity" between segments for smooth trajectory
- Need to ensure "minimal" motion

\[
\min_{x(t), u(t)} J(t_A, t_B, x(t), u(t)) = \min_{\sigma(t)} J(t_A, t_B, \sigma(t))
\]
Decoupled Trajectory Planning

- Need to enforce "continuity" between segments for smooth trajectory
- Need to ensure "minimal" motion
- Need to ensure feasibility; e.g.:
  - Hit no obstacles
Decoupled Trajectory Planning

- Need to enforce "continuity" between segments for smooth trajectory
- Need to ensure "minimal" motion
- Need to ensure feasibility; e.g.:
  - Hit no obstacles
Decoupled Trajectory Planning

- Need to enforce "continuity" between segments for smooth trajectory
- Need to ensure "minimal" motion
- Need to ensure feasibility; e.g.:
  - Hit no obstacles
Decoupled Trajectory Planning

- Need to enforce **"continuity"** between segments for smooth trajectory
- Need to ensure **"minimal"** motion
- Need to ensure **feasibility**; e.g.:
  - Hit no obstacles (done)

Decoupled Trajectory Planning
Decoupled Trajectory Planning

- Need to enforce "continuity" between segments for smooth trajectory
- Need to ensure "minimal" motion
- Need to ensure feasibility; e.g.:
  - Hit no obstacles
  - Don’t saturate controller

Friday’s lecture...
Decoupled Trajectory Planning

- Need to optimize "continuity" between segments to ensure numerical stability of solution.
- Need to ensure "minimal" motion.
- Need to ensure feasibility; e.g.:
  - Hit no obstacles
  - Don’t saturate controller
- Need to control time-length of trajectory.

Friday’s lecture…