NYU TANDON SCHOOL OF ENGINEERING

Autonomous Mobile Robots

Lecture 8: Motion Planning Algorithms

Dr. Aliasghar Arab

NYU Tandon School of Engineering

Fall 2025

Today's lecture covers the integration of control and planning:

1. **Control Philosophy:** Deterministic vs probabilistic approaches
2. **Nonlinear Model Predictive Control (NMPC):** Optimization and implementation
3. **Local Planning:** Dynamic Window Approach (DWA)
4. **Global Planning:** RRT and RRT* algorithms
5. **System Integration:** Complete autonomous navigation architecture

Three Essential Components

1. **System Modeling**
   - Kinematics, dynamics, mathematical formulation
   - Physical working principles and implementation
2. **Control Methodology**
   - Justification of chosen approach
   - Comprehensive testing and validation
3. **Motion Planning**
   - Integration of planning algorithms
   - Explanation of underlying principles

Fundamental Philosophy

Deterministic methods rely on known physics, not probability distributions

Deterministic Control (NMPC)

System model: $\dot{x} = f(x, u)$

- Physics is reliable and known
- Future states are predictable
- Suitable for well-modeled systems

Probabilistic Methods (RL)

System: $P(s'|s, a)$ (MDPs)

- Environment is uncertain
- Outcomes are stochastic
- Optimizes expected reward

Selection criterion: Confidence in physical model accuracy

Stochastic Environment Scenario

Scenario: Robot navigation on slippery ice

Command: "Move Forward"

Actual outcomes:

- 33% probability: Move Forward
- 33% probability: Slide Left
- 33% probability: Slide Right

Limitation: NMPC assumes deterministic dynamics
Solution: Reinforcement learning handles probability distributions

Optimal Control Problem

At each time step t, solve over prediction horizon T_p:

$$\min_u J = \int_t^{t+T_p} \left[\|x(\tau) - x_{\text{ref}}(\tau)\|_Q^2 + \|u(\tau)\|_R^2\right] d\tau + \|x(t+T_p) - x_{\text{goal}}\|_P^2$$

Subject to constraints:

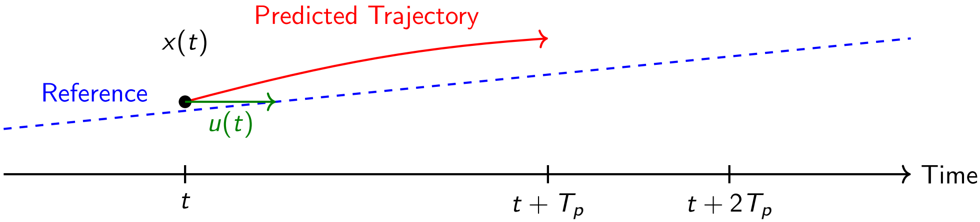
- System dynamics: $\dot{x} = f(x, u)$
- State constraints: $x \in \mathcal{X}_{\text{safe}}$
- Input constraints: $u \in \mathcal{U}_{\text{feasible}}$

Prediction Horizon

T_p determines how far into the future the controller predicts

Key insights:

- Longer $T_p \rightarrow$ Better planning, higher computational cost
- Shorter $T_p \rightarrow$ Faster computation, more reactive behavior
- Typical values: 1-5 seconds for mobile robots
- Must balance prediction quality with real-time requirements



Control Execution

1. Measure current state $x(t)$ **2.** Solve optimization over horizon $T_p \rightarrow$ obtain $u^*(t)$
3. Apply only first control $u^*(t)$ **4.** Shift window and repeat

Tracking Error

$\|x - x_{\text{ref}}\|_Q^2$

- Penalizes path deviation
- Large Q: tight tracking
- Small Q: loose tracking

Control Effort

$\|u\|_R^2$

- Penalizes large inputs
- Large R: smooth control
- Small R: aggressive

Terminal Cost

$\|x(T_p) - x_{\text{goal}}\|_P^2$

- Ensures final state quality
- Improves stability
- Guides to goal region

Engineering Insight

Weights Q, R, P are tuning parameters - balance tracking vs control effort

Abu Dhabi Autonomous Racing League

Winner: Technical University of Munich (NMPC implementation)

Performance: 250+ km/h, zero crashes during race

Critical: Tire friction modeling $F_{\text{lateral}} = \mu(v, \alpha) \cdot F_N$

Key Insight

"At 250 km/h, PID is reactive - by the time you see an error, you've already crashed. You NEED prediction."

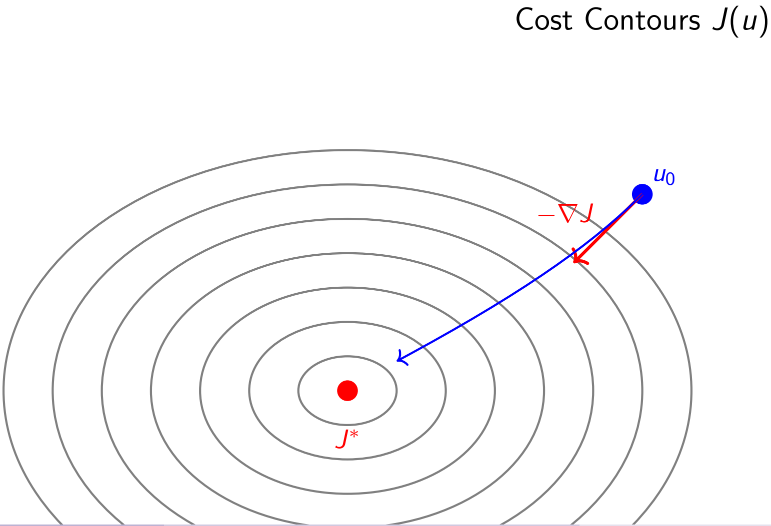
Continuous-Time Update Law

$$\dot{u} = -\alpha \nabla J(u) = -\alpha \frac{\partial J}{\partial u}$$

Understanding the notation:

- $\dot{u}$ represents **algorithmic time derivative**
- Solver "flows" down cost surface toward minimum
- Not physical system time - this is solver iteration

Implementation


Cost Contours $J(u)$
u_0
$-\nabla J$
J^*

Proof of Convergence

Step 1 - Lyapunov Candidate: $V(u) = J(u)$

Step 2 - Time Derivative:

$$\dot{V} = \frac{dJ}{dt} = \frac{\partial J}{\partial u} \cdot \frac{du}{dt} = \frac{\partial J}{\partial u} \cdot \left(-\alpha \frac{\partial J}{\partial u}\right) = -\alpha \left\| \frac{\partial J}{\partial u} \right\|^2$$

Step 3 - Stability: $\dot{V} = -\alpha \|\nabla J\|^2 \leq 0$

Guarantee

Cost decreases monotonically $\rightarrow$ Solver converges stably

Finite Difference Approximation

$$\frac{\partial J}{\partial u} \approx \frac{J(u+\delta u) - J(u)}{\delta u}$$

Implementation Steps

1. Simulate with u, compute $J(u)$
2. Perturb: $u' = u + \delta u$
3. Simulate with u', compute $J(u')$
4. Gradient: $\nabla J \approx \frac{J(u') - J(u)}{\delta u}$
5. Update: $u_{\text{new}} = u - \alpha \nabla J$

Critical Parameter

10× Bandwidth Rule

$$\Delta t_{\text{sim}} \leq \frac{1}{10} \times \tau_{\text{system}}$$

Practical Example

- Motor time constant: $\tau = 0.1$ seconds
- Required: $\Delta t \leq 0.01$ seconds
- Prediction horizon $T_p = 2\text{s} \rightarrow 200$ simulation steps

Consequences of Violation

- Numerical integration errors

Core Principle

Search in **velocity space** (v, ω) instead of position space (x, y)

Dynamic Window

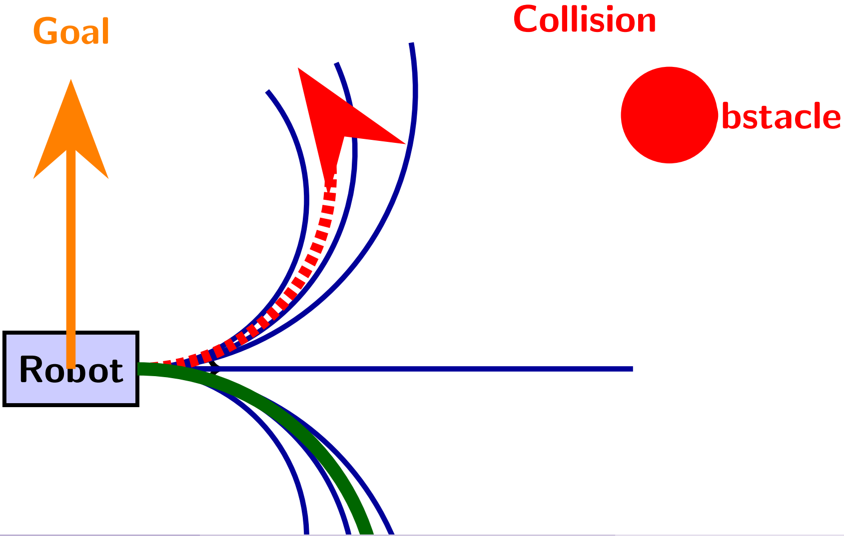
$$V = [V_t - a_{\text{min}} \cdot dt, V_t + a_{\text{max}} \cdot dt]$$

$$\Omega = [\omega_t - \alpha_{\text{min}} \cdot dt, \omega_t + \alpha_{\text{max}} \cdot dt]$$

- Based on acceleration limits
- Defines reachable velocities
- Ensures feasibility

Advantage

- Pre-computed trajectories
- Fast minimization
- Real-time performance


Collision
Goal
bstacle
Robot

Multi-Objective Optimization

$$G(v, \omega) = \alpha \cdot \text{Heading}(v, \omega) + \beta \cdot \text{Clearance}(v, \omega) + \gamma \cdot \text{Velocity}(v, \omega)$$

Component definitions:

- **Heading:** Alignment with goal direction
- **Clearance:** Distance to nearest obstacle
- **Velocity:** Progress toward goal

Typical Configuration

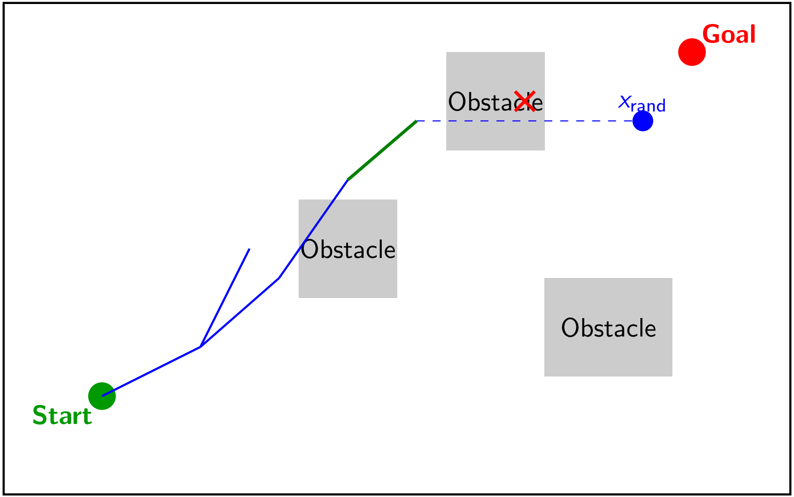
10 linear velocities $\times$ 10 angular velocities $=$ 100 trajectories to evaluate

Algorithm

1. **Sample:** Random state x_{rand} in free space
2. **Nearest:** Find closest node x_{near} in tree
3. **Steer:** Extend from x_{near} toward x_{rand} by Δq
4. **Check:** If collision-free, add x_{new} to tree
5. **Repeat:** Until goal reached or maximum iterations

Key Feature

Rapidly explores high-dimensional spaces by random sampling


Goal
Obstacle
x_{rand}
Obstacle
Obstacle
Start

Computational Efficiency: KD-Trees

NYU

Complexity Challenge

- Naive nearest neighbor: $O(N)$ for N nodes
- With KD-tree: $O(\log N)$ complexity
- Example: $N = 10{,}000 \rightarrow 14$ comparisons vs 10,000

KD-Tree Structure

- Binary space partitioning
- Alternating axis splits
- Essential for real-time RRT

Key Point

Standard RRT

- Finds *any* feasible path
- Fast exploration
- No optimality guarantees
- Good for: Quick solutions

RRT* (Optimal)

- Asymptotically optimal
- "Rewiring" improves paths
- Converges to optimum
- Good for: Quality paths

RRT* Rewiring

- Check if X_{new} provides better paths to neighbors
- Rewire tree to reduce costs
- Continuous improvement over time

Integrated Operation

- Global planning for strategic navigation
- Local adaptation for dynamic obstacles
- Physics-based control for precise execution
- Continuous sensor feedback for real-time updates

1. **Deterministic Control Foundation**
    - NMPC relies on accurate physical models
    - Gradient descent with Lyapunov stability guarantees
    - Suitable for well-characterized systems
2. **Critical Implementation Rules**
    - $10\times$ bandwidth rule: $\Delta t \leq \tau_{\text{system}}/10$
    - Proper finite difference gradient estimation
    - Careful weight selection in cost functions
3. **Hierarchical Planning**
    - Global: RRT/RRT* for strategic paths
    - Local: DWA for dynamic obstacle avoidance

Development Strategy

1. **Start Simple**
   - Basic controller implementation first
   - Incremental complexity addition
   - Early testing and validation

2. **Method Selection**
   - Match complexity to application needs
   - Classical control often sufficient
   - Advanced methods for demanding applications

3. **Validation Process**
   - Comprehensive simulation before hardware
   - Careful time step verification

Summary

Today we covered the complete motion planning pipeline:

- **Control philosophy:** When to use deterministic vs probabilistic methods
- **NMPC implementation:** Optimization formulation, gradient descent, stability analysis
- **Local planning:** Dynamic Window Approach for real-time obstacle avoidance
- **Global planning:** RRT/RRT* for strategic path generation
- **System integration:** Hierarchical architecture combining all components

Next lecture: Advanced topics in perception and sensor fusion

Reference Materials

Planning Algorithms: LaValle, "Planning Algorithms"

RRT Foundation: LaValle (1998)

Optimal RRT: Karaman & Frazzoli, "RRT*"

Digital Control: Various sampling theory texts

End of Lecture 8

Motion Planning Algorithms

Questions?