EECS C106B / 206B Robotic Manipulation and Interaction **Spring 2020**

Lecture 11: Optimization Based Motion Planning and Optimal Control

Scribes: Yibin Li, Mina Beshay

1.1 Overview

1.1.1 Why need path planning?

So far we have learned a lot about control. But in the real world control is just part of the work. Another portion is called path planning. It is a high level work of the robot manipulation. Example: Highway Lane Change Problem. You are approaching to an exit and you need to change lane. But you find two cars on you right occupying the lane. What will you do? You have several options: option 1 cut to the front of these two cars. Option 2 wait and track the second car. In both scenario, you need to consider about several constraints. Safety is the top priority. You definitely cannot go too fast. But not too slow either (occupying the lane for too long and is dangerous in the highway). You also need to consider about engine input and steering input. This is a classic path planning problem.

Several requirements:

- environment: understand the environment around the car (other cars)
- safety: put a safety constraints as the top priority
- goal: change lane

Formally, in this problem, we need to address four issues:

- Policy and Strategy
  - Global: all safe and comfortable
  - Local: need to stay on right side and lane change to another highway
- Vision: detect and analyze neighboring objects with semantic information
  - LIDAR
  - camera
  - Radar
- Planner: plan a collision-free and comfortable trajectory
  - Sample various candidates trajectories
  - Select a best one according to some criteria (desired refresh rate/correctness/complexity)
  - Smooth the speed profiles to make it likely to be dynamically feasible
- Control

  - control the vehicle to follow this path

Planning is the bridge between high level and low level. Frequently we need to consider about:

- Need to consider noise and disturbance from vision module
- Make the trajectory dynamically-feasible for the control module and other low-level ones.

1.1.2 Optimization based path planning VS Optimal control

- optimal control techniques: linear–quadratic regulator (LQR), model predictive control (MPC), sliding mode control, etc.
- optimization based planning considers:
  - speed profile smoothing
  - contact force
  - energy efficiency: snap and jerk.

  for example, we have
  $$C = \dddot{\dddot{X}}^T Q \dddot{\dddot{X}}$$
  for snap and
  $$C = \ddot{\ddot{X}}^T Q \ddot{\ddot{X}}$$
  for jerk, where C is the cost function.
- same mathematics intuition: an optimization problem is formulated to generate control strategy or trajectory/robot's motions.
- optimal control: require high refresh rate and needs to solve really fast.
- optimization based planning: doesn't need to be solved in fast rate, i.e. it can be offline planning.
  - drone example: plan for a while but execute the control in real time.

1.2 Optimization based planning

1.2.1 Optimization problem setup

Generally, an optimal control problem or an optimization-based path planning problem could be described as follows

$$\begin{gathered} u^*(t) := \arg\min_{u(.)} = \int_{t_0}^{t_f} c_t(x(t), u(t))dt \\ s.t.\ \ x(t_0) = x_0 \\ \dot{x}(t) = f(x(t), u(t)) \quad \forall t \\ x(t) \in X_{feas} \quad \forall t \quad (collision - free) \\ u(t) \in U_{feas} \quad \forall t \quad (control\ limits) \end{gathered}$$

- In the view of control: generate feasible control inputs under dynamics constraints
- In the view of planning: generate dynamically-feasible waypoints (which will be tracked with appropriate control methods).

1.2.2 Mathmatics tools

Generally, we can express the optimization problem into familiar equation. Some common solutions are

- Convex quadratic programming (convex QP)
- LQR

These methods have solution explicitly.

However, general problems may not have explicit solution. Two methods to handle those situation or we can formulate solution of these types:

- Shooting methods: we 'shoot' out trajectories in different directions until we find a trajectory that has the desired boundary value.
- Collocation methods: choose a finite-dimensional space of candidate solutions (e.g. polynomials) and a number of points in the domain (called collocation points), and to select that solution which satisfies the given equation at the collocation points.

Common solvers are able to solve those optimization problems.

- convex solver: CVX, OSQP (easy)
- non-convex solver: IPOPT, SNOPT (hard, generally not guaranteed to find a solution.)

Many ongoing researches in this area.

1.2.3 First problem to solve: smoothing the path

Intends to make trajectory generated from the sample-based methods (RRT, PRM, etc) become smoother.

1. polynomial interpolation
2. bezier curve
3. cubic splines
4. B-splined

however in industry, people don't care about those fancy methods. Rather, they want to have something much faster and more reliable (you never know how long these smoothing methods will run). For example, **Dubin's Curve (e.g. lane change in autonomous driving)**.

Smoothness doesn't necessarily guarantee dynamically-feasibility. The optimization-based methods could also deal with trajectory smoothing problem. Some recent research of smoothing focus on

- unknown environment
- noise
- etc.

The pipeline for planning is trajectory generation + trajectory smoothing. In industry, people prefer to not use optimization solver (again, those solvers may take long time to find solution).

1.2.4 Second problem: obstacle avoidance

There are several ways to consider obstacle avoidance as constraints in an optimization-based problem:

- Simple geometric constraints (Ex: $(x - x_{obs})^2 + (y - y_{obs})^2 \geq r^2$):
  - Very rough approximation
  - Nonlinear and non-convex
  - Can be solved with modern solvers but doesn't guarantee solutions
- Divide the map to several convex regions:
  - Makes a problem convex by staying in areas where convex behavior is guaranteed
  - Turns into mixed-integer problem (choose integer combination of labeled areas)
  - Increasing regions improves fidelity but complexity very rapidly increases with number of regions
- Rigid Bodies with Obstacle avoidance:
  - Lets us define the distance between 2 bodies
  - Uses dual variables to represent collision free constraints
- Other research such as:
  - STOMP: Stochastic approach recommended by ROS
  - CHMOP: Considers way-point optimization and trajectory smoothness
  - specialized solutions (swarm path planning for UAV's and Baidu Apollo open source for autonomous driving)

1.2.5 Contact force

Intuition: introduces complementary constraint where contact force is 0 or distance between contact points is 0. Therefore, we can set up a constraints optimization problem.

Find

$$\ddot{q}, \lambda$$

Subject to

$$H(q)\ddot{q} + C(q, \dot{q}) + G(q) = B(q)u + J(q)^T\lambda$$

$$\phi(q) \geq 0 \; (distance)$$

$$\lambda(q) \geq 0 \; (contact\ force)$$

$$\phi(q)^T\lambda = 0 \; (complementary\ constraint)$$

Because of of the complementary constraint, you no longer have to consider mode sequencing. This important for legged robotics and finger contact/manipulation where discontinuous contact dynamics are represented as discrete modes.

1.2.6 Energy efficiency

Intuition: create a cost function to minimize jerk and snap which leads to systems that optimize for energy efficiency and smoothness

- Jerk: third order derivative of position. Performs better with vehicles involving motors because of motor's torque inputs.
$$C = \dddot{X}^T Q \dddot{X}$$
- Snap: One higher degree than jerk. Generally better for human like motion such as legged robotics. Might jerk over snap or vice versa based on performance.
$$C = \ddddot{X}^T Q \ddddot{X}$$

1.3 Optimal control

1.3.1 MPC

LQR can be used to generate a set of optimized Gains to reduce the amount of work required to appropriately tune a controller

$$J^* = \min_{u_i} \sum_{k=0}^{N} x_{k+1}^T Q x_{k+1} + u_k^T R u_k$$

Such that:

$$x_{k+1} = Ax_k + Bu_k$$

However, this only generates constant feedback gains that do not change with time and can not account for data gathered during execution of the path

The intuition for MPC follows: sequentially measure the state and goal and generate a control law and input based on these measurements. Generally it is a QP problem that the system solves for predicting the control required to achieve the next N way-points.

$$J^*(x(t)) = p(x_{t+N}) + \sum_{k=0}^{N-1} q(x_{t+k}, u_{t+k})$$

Such that:

$$x_{t+k+1} = Ax_{t+k} + Bu_{t+k}, k = 0, 1, ..., N-1$$
$$x_{t+k} \in X, u_{t+k} \in U, k = 0, 1, ..., N-1$$
$$x_{t+N} \in X_f$$
$$x_t = x(t)$$

Truncate after a finite horizon N:

- p(x_{t+N}): terminal cost which approximate the 'tail' of the cost.
- $q(x_{t+k}, u_{t+k})$: staged cost

- X_f: approximates the 'tail' of the constraints

When do the predictions occur?

- At each sampling time, solve the constrained optimal control problem accounting for the points [t,t+N].
- Apply the optimal input only during [t,t + 1].
- At t + 1, solve a CFTOC over a shifted horizon [t+1,N+1] based on the new state

Some definitions:

- Open-loop trajectory: generated at each time step from the constrained optimal control problem, which solves a feasible trajectory with predictions
- Closed-loop trajectory: since we solve optimal control problem at each time step to create new control policies, the robot will move along a closed-loop trajectory defined by the changing control policy

Important Notes

- Terminal cost ensures the Lyapunov convergence, but it's only necessary for the mathematics (can ignore in practice).
- Larger horizon brings a bigger controllable set, but needs more time to calculate the optimization because of increased complexity.
- The dynamics constraints could be nonlinear, which will make the application a dynamic programming problem.
- There are many variants of MPC for more specific goals: Robust MPC, Adaptive MPC, MPC with obstacle avoidance, etc.
- Recent work about Learning MPC takes safety constraints into account.