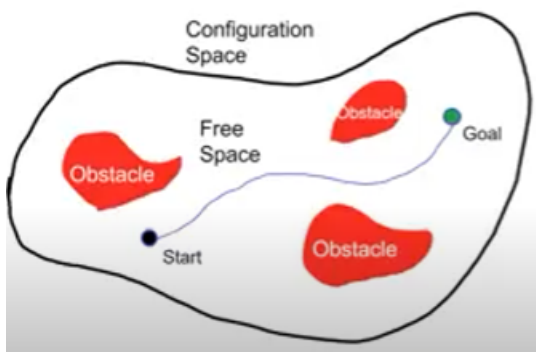


Lecture 4B: (Motion Planning for Non-holonomic systems)

Scribes: Aryaman Jhunjhunwala

The goal of motion planning is to find a path through the free space that links the start and the goal, and avoids any obstacles.



The main challenges we face when doing motion planning are obstacles and constraints.

These notes will cover 3 different methods of path planning: classical methods, optimization based methods, and sampling based methods.

4.1 Classical methods - Steering with sinusoids.

The first classical method is steering with sinusoids.

Advantages of this method are that it is easy(ish) to compute and is constructive and complete.

Some disadvantages are that it doesn't work well with obstacles, and it can be aggressively sub optimal sometimes given certain constraints.

Here is a link to professor Sastry's paper which explains it in more detail.

<https://authors.library.caltech.edu/7315/1/MURieeetac93.pdf>

4.2 Classical methods - Jacobs-Laumond

This method relies on the idea of small time contractility, which is the idea that even though the system is non-controllable, given an infinitesimally small amount of time, you can translate in any direction in the state space.

There are 3 steps to the Jacobs-Laumond method:

1. Ignoring constraints, produce a path which stays at least d away from obstacles.
2. Approximately follow this path with nonholonomic motions
3. Iteratively smoothen/refine.

Advantages: Works reasonably well

Disadvantages: Not complete. Can result in jagged paths if smoothing doesn't work.

4.3 Classical methods - Dubins/Reeds-Shepp paths

These re a collection of lines and arcs of constant radius.

It is an optimal solution but only works on Dubins/Reeds-Shepp cars.

Doesn't consider obstacles.

Dubins Car Moves forward at constant speed

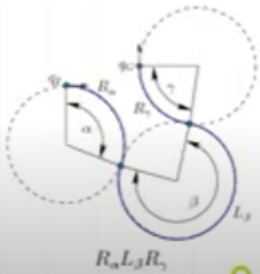
$$\dot{x} = u_1 \cos(\theta)$$

$$\dot{y} = u_1 \sin(\theta)$$

$$\dot{\theta} = u_2,$$

$$u_1 \in \{0, 1\}$$

$$u_2 \in [-1, 1]$$



The graphics for a Dubins care are shown above. The only difference between a Dubins car and a Reeds-Shepp car is the Reeds-Shepp car can move backwards as well, so u_1 could also be -1.

Dubins cars are very simplified systems, and just because a model works on a Dubins car doesn't mean it will work well in real life, or in a more complex situation.

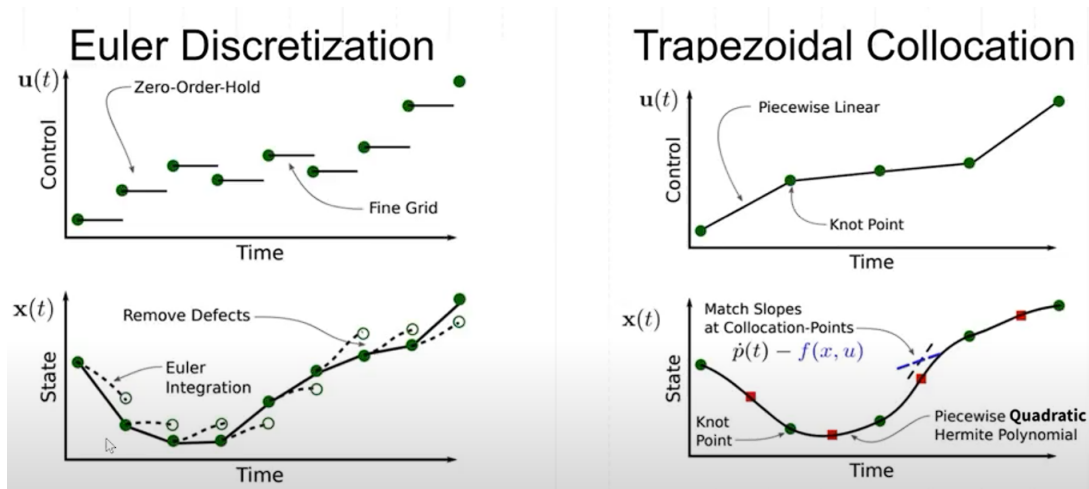
4.4 Optimization based methods.

Path planning can be thought of as the following optimization problem.

$$\begin{aligned} \arg \min_{x(t), u(t)} & \int_0^T \text{cost}(x(t), u(t)) dt \\ \text{s.t.} & \text{constraints}(x(t), u(t)) \leq 0 \\ & \dot{x}(t) = f(x(t), u(t)) \\ & x(0) = x_{\text{start}} \\ & x(T) = x_{\text{goal}} \end{aligned}$$

Doing optimization on continuous paths and functions is very difficult, so our first step will be to discretize the path.

The main ways of doing this are Euler Discretization, trapezoidal collocation (first order systems) and Hermite-Simpson collocation (second order systems.)



This is the new discretized optimization problem:

$$\begin{aligned} \arg \min_{x_t, u_t} & \sum_0^T \text{cost}(x_t, u_t) \\ \text{s.t.} & \text{constraints}(x_t, u_t) \leq 0 \\ & x_{t+1} = f(x_t, u_t) \\ & x_0 = x_{\text{start}} \\ & x_T = x_{\text{goal}} \end{aligned}$$

When we use optimization it is important we define "optimal" for our scenario as it influences how we design our cost function. Different metrics for optimality might include fastest/shortest route, safest route, most fuel efficient etc.

4.5 Optimization - LQR

One way of doing this optimization is to use a linear quadratic regulator.

Suppose $f(x, u) = Ax + Bu$, $cost(x, u) = x^T Qx + u^T Ru$, and $T = \infty$

$$\begin{aligned} \arg \min_{x_t, u_t} & \sum_0^{\infty} x_t^T Qx_t + u_t^T Ru_t \\ s.t. & \quad x_{t+1} = Ax_t + Bu_t \\ & \quad x_0 = x_{\text{start}} \end{aligned}$$

4.6 Optimization - Receding Horizon Control

Very commonly used in the real world. Planning a few timesteps ahead, then recalculating plan.

- 1) Given a system model, figure out the best sequence of system input for the next T timesteps.
- 2) Execute the first input in the sequence
- 3) Repeat

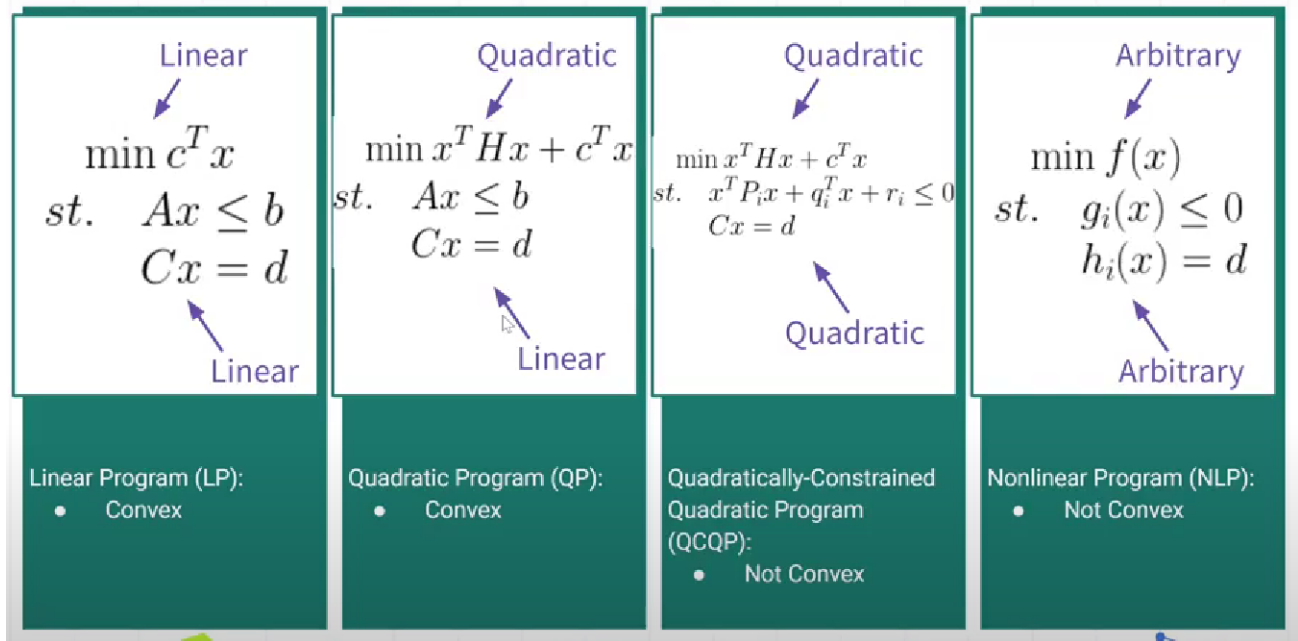
4.7 The fuzzy line between motion planning and control

There is no fixed definition for at what point motion planning becomes control.

In general when people talk about motion planning they are referring to longer time scales, very abstract models, and a larger more detailed search space.

Control tends to refer to shorter timescales and simplified models that enable fast computation.

4.8 Types of Optimization problems.



4.9 Tips and challenges for Optimization problems.

One main challenge with optimization is how do we mathematically represent obstacles.

One way is to model them as spheres, but this forces your constraints to be quadratic, so your problem will be convex.

Another option is to use polytopes, however this can result in a mixed integers problem, which depending on the number of way points and combinations, can be very computationally inefficient.

One final way is to use signed distance fields, which are good up to 3 dimensions but become inefficient after that.

One tip is to "warm start" your optimization problem with a previous solution. If one isn't available use a straight line from the start to the finish.

DO NOT randomly initialise your optimization solver.

4.10 Sampling based planners.

1) build a graph by sampling waypoints.

2) find the shortest path in the graph.

Advantages: Scales well for high dimensional systems. No explicit representation of free space needed.

4.11 Sampling - RRT.

Initialize an empty graph \mathbf{G} and add the start configuration

Until we either succeed or reach the maximum number of iterations, do:

1. Sample a configuration \mathbf{c}_{rand} from the configuration space
2. Find the nearest node \mathbf{c}_{near} in \mathbf{G} to \mathbf{c}_{rand}
3. Construct a local plan \mathbf{p} that steers the robot from \mathbf{c}_{near} to \mathbf{c}_{rand} .
If this plan is in collision with the environment, discard it
4. Take some small prefix of this plan \mathbf{p}' which ends at \mathbf{c}_{new} . Add \mathbf{c}_{new} to \mathbf{G} with \mathbf{p}' as an edge
5. If \mathbf{c}_{new} is within ϵ of the goal \mathbf{g} , then we have success

An issue with this is actually finding the "closest node." If you don't have a simple model like a Dubins car it is very hard to calculate the closest node.

Some solutions are to just use the Euclidean distance, or create some kind of Heuristic.

Tips for RRTs are to use goal biasing, where you periodically just sample the goal, and path biasing, where you concentrate your samples around a nominal path. This is to speed up the algorithm and help it calculate the path more efficiently.

4.12 Conclusions

None of these methods need to be used in isolation. The best methods are going to be a combination of a lot of these methods. For example, a classical method could be used as a primitive for an optimization method.