Motion Planning: Wild Frontiers

(Steven M. LaValle)

★ Introduction

⇒ Figure below shows how a computed collision-free path $\tau : [0,1] \to C_{free}$ is usually brought into alignment with producing a feedback control law.

(Complete geometric model of the world)

↓

Step 1: (Compute a collision-free path $\tau : [0,1] \to C_{free}$)

↓

Step 2: (Smooth τ to satisfy differential constraints $\sigma : [0,1] \to C_{free}$)

↓

Step 3: (Design a trajectory that follows σ $\tilde{q} : [0,t] \to C_{free}$)

↓

Step 4: (Design a feedback controller to track $\tilde{q}$ $\pi : X \to U$)

↓

(Execute π on the robot)

Note: The domain X is State space and U is an action space (Input space).

⇒ These sets appear in the definition of the Control System that models the robot:

$$\boxed{\dot{x} = f(x, u)} \quad \text{where, } x \in X$$
$$\& \; u \in U$$

⇒ One clear problem in this general framework is that a later step might not succeed due to an unfortunate, fixed choice in an earlier step.

- → Even if it does succeed, the produced solution may be horribly inefficient.
- → This motivates planning under differential constraints (i.e. Step 1 & Step 2 together) or Step 1, Step 2 & Step 3 in one shot.

★ Differential Constraints

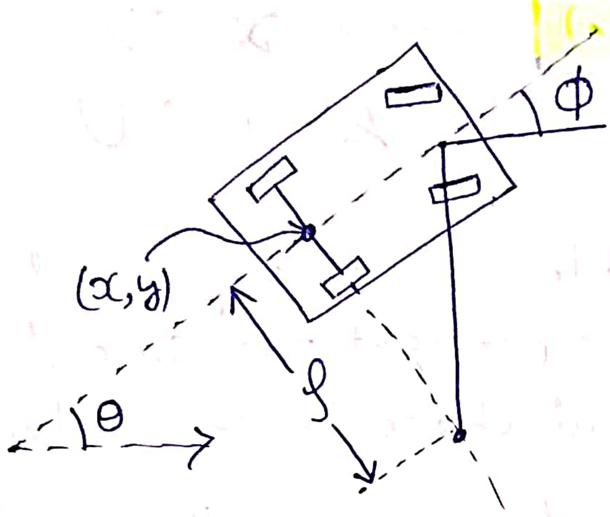
⇒ Differential constraints naturally arise from the kinematics and dynamics of robots.

Ⓐ Modeling the Constraints

$$\boxed{\dot{q} = f(q, u)} \quad \{\text{Configuration transition function}\}$$

⇒ The parameter u is called an action (or input) and is chosen from a predetermined action space U.

⇒ Figure below shows a Car-like robot which has the C-space of a rigid body in the plane $C = \mathbb{R}^2 \times S^1$.



⇒ The configuration vector $q = (x, y, \theta)$.

⇒ Note: It is impossible to move the center of the rear axle laterally because the rear wheels skid instead of roll.

⇒ This induces the constraint $\boxed{\dot{y}/\dot{x} = \tan\theta}$

⇒ This constraint, along with another due to the steering angle can be converted into the following form.

$$\boxed{\begin{aligned} \dot{x} &= u_s \cos\theta \\ \dot{y} &= u_s \sin\theta \\ \dot{\theta} &= \frac{u_s}{L} \tan u_\phi \end{aligned}}$$

in which $u = (u_s, u_\phi) \in U$ is the action:

$u_s \rightarrow$ Forward Speed

$u_\phi \rightarrow$ Steering angle

$\Rightarrow$ Usually, the steering angle is bounded by some $\phi_{max} < \pi/2 \Rightarrow |u_\phi| \leq \phi_{max}$.

$\Rightarrow$ For the possible speed values u_s, a simple bound is often made.

Eg: $|u_s| \leq 1$

$\Rightarrow$ A finite set of values is often used for planning problem that are taking into account only the kinematic constraints due to rolling wheels.

$\Rightarrow$ Letting $U = \{-1, 0, 1\}$ produce what is called the Reeds-Sheep Car.

$\Rightarrow$ By further restricting so that $U = \{0, 1\}$, the Dubins Car is obtained.

$\Rightarrow$ The transition equation f becomes the interface through which solution path must be constructed

B. Moving to the State Space

⇒ We must consider differential constraints that account for both kinematics & dynamics of the robot.

⇒ This allows velocity & acceleration constraints to be appropriately modeled usingly resulting in a transition equation of the form

$$\boxed{\ddot{q} = h(q, \dot{q}, u)}$$

⇒ Le $x = (q^T, \dot{q}^T)^T$ be the state of the system.

⇒ So above equation can be written as

$$\boxed{\dot{x} = f(x, u)} \quad \{ \text{State transition equation} \}$$

⇒ If the first n components of x corresponds to q and if $q \in C_{obs}$ then $x \subset X_{obs}$ regardless of which values are chosen for the remaining components.

C. Sampling based Planning

⇒ Let X be a state space with a given state transition equation $\dot{x} = f(x, u)$ and action space U.

⇒ Task is to compute a function $\boxed{\tilde{u} : [0, t] \rightarrow U}$ that has corresponding trajectory $\tilde{q} : [0, t] \rightarrow X_{free}$ with $\tilde{q}(0) = x_I$ & $\tilde{q}(t) \in X_G$.

⇒ Due to the great difficulty of planning under differential constraints, nearly all planning algorithms are sampling based.

⇒ To develop sampling based planning algorithms in this context, several discretizations are needed.

↳ With differential constraints, the time interval and possibly also U require discretization in addition to X.

⇒ One of the simplest way to discretize the differential constrains is to construct a discrete differential model, which is characterized by three time model, which is characterized by three aspects:

① Time is partitioned into intervals of length Δt.

↳ This enables stages to be assigned, in which stage k indicates the $(K-1)\Delta t$ time has elapsed.

② A finite subset U_d of the action space U is chosen. If U is already finite then the selection of $U_d = U$.

③ The action $\tilde{u}(t)$ remains constant over each time interval.

⇒ From an initial state, a reachability tree can be formed by applying all sequence of discretized action

⇒ For a general system, each trajectory segment in the tree is determined by numerical integration of $\dot{x} = f(\tilde{x}(t), \tilde{u}(t))$ for a given $\tilde{u}$.

⇒ Sampling-based planning algorithms proceed by exploring one or more reachability trees that are derived from discretization.

⇒ Region of inevitable Collision $\{X_{ric}\}$

↳ Set of all states from which, no matter what action history is applied, entry into X obs is unavoidable.

★ Feedback Motion Planning

⇒ This becomes necessary because of imperfections in the transition equation.

⇒ Rather than computing a path or trajectory, we need representations that indicate what actions to apply when the robot is at various places in the C-space.

↳ If dynamics are a concern, then we should know what action to apply from places in the state space X.

⇒ In these cases, we must "feed" the current estimated configuration or state "back" into the plan to determine which action to apply.

A. Feasible feedback planning

$\pi : X \to U$ {feed back plan}

⇒ $x' = f(x, \pi(x))$ {Next state can be obtained using this}

⇒ Another way to represent a feedback plan is through an intermediate potential function
$$\phi : X \to [0, \infty]$$

⇒ Given f and ϕ a plan π is derived by selecting u according to:

$$u^* = \underset{u \in U(x)}{\arg\min} \{\phi(f(x,u))\}$$

⇒ A potential function ϕ, is called a navigation function if

1) $\phi(x) = 0 \;\; \forall x \in X_G$

2) $\phi(x) = \infty$ if & only if no point in X_G is reachable from x

3) For every reachable state, $x \in X \setminus X_G$, applying u^* produces a state x' for which
$$\phi(x') < \phi(x)$$