C106B - Robotic Manipulation and Interaction **(Week 5)**

Discussion #5

Author: Amay Saxena

Problem 1 - Sinusoidal Inputs

1. Consider the sinusoidal function
$$f(t) = \sin(\frac{2\pi}{T}t + \phi) \tag{0.1}$$
where T, ϕ are constants. For some arbitrary constant $c \in \mathbb{R}$, evaluate
$$\int_c^{c+T} f(t)dt \tag{0.2}$$

2. Describe some some cases in driving when you apply sinusoidal inputs to keep some state variables the same before and after your control but change others.

Problem 2 - Randomly Exploring Random Trees (RRT)

The RRT algorithm is as follows:

Algorithm 1 The RRT algorithm

Graph.add_node(q_{init})
while *goal* $\notin$ *Graph* **do**
  $q_{rand} \longleftarrow$ Sample_Configuration()
  $q_{near} \longleftarrow$ Nearest_Vertex(q_{rand}, Graph)
  $q_{new} \longleftarrow$ Local_Planner(q_{near}, q_{rand})
  if NOT Check_Collision(q_{new}) **then**
   Graph.add_node(q_{new})
   Graph.add_edge(q_{near}, q_{new})
  end if
end while
return Graph

1. What parts of this algorithm would need to be designed/changed in order to use it with the bicycle model car?

2. You are using the RRT planner to plan a path for the Baxter robot. How would you go about implementing collision-checking? Would you expect an RRT to work better or worse for an open-chain manipulator like Baxter as opposed to a bicycle model car? (*Hint: what velocity constraints exist for the baxter in jointspace? workspace?*

3. Assume we are using the RRT algorithm for a turtlebot to plan from some starting configuration to an ending configuraiton with state limits

$$-10 \leq x \leq 10 \tag{0.3}$$
$$-10 \leq y \leq 10 \tag{0.4}$$
$$-\pi \leq \theta \leq \pi \tag{0.5}$$

   Assuming that we sample states uniformly in this state space, how many times do we need to sample until we expect to sample a state within 1 unit of the goal configuration?

Problem 3 - Optimization based path-planning

We will be converting the path planning problem into the following nonlinear optimization problem:

$$
\begin{aligned}
q^*, u^* = \text{argmin} \sum_{i=1}^{N} \left((q_i - q_{goal})^\top Q (q_i - q_{goal}) + u_i^\top R u_i \right) + (q_{N+1} - q_{goal})^\top P (q_{N+1} - q_{goal}) & \\
s.t.\ q_{min} \leq q_i \leq q_{max}, & \quad \forall i \\
u_{min} \leq u_i \leq u_{max}, & \quad \forall i \\
q_{i+1} - F(q_i, u_i) = 0, & \quad \forall i \leq N \\
(x_i - \text{obs}_{j_x})^2 + (y_i - \text{obs}_{j_y})^2 \geq \text{obs}_{j_r}^2, & \quad \forall i, j \\
q_1 = q_{start} & \\
q_{N+1} = q_{goal} &
\end{aligned}
$$

For Q, R, P all positive semi-definite matrices. Here, we *discretize* the problem over $N+1$ timesteps, with the first indicating the current time, and the last indicating the time at which we intend to reach the goal. q is an array of $N+1$ states (x, y, θ, ϕ) with $q_i = (x_i, y_i, \theta_i, \phi_i)$ for $i = 1, ..., N+1$, and u is an array of N inputs (v, ω). We don't have an input at the last state. obs is an array of obstacles. In order to simplify computation, we assume that all obstacles are circles with center (x, y) and radius r. We also assume that our robot is circular, and that its radius has been incorporated into the radii of all obstacles. $F(q, u)$ is the discrete dynamics of our system, $q_{k+1} = F(q_k, u_k)$.

1. Explain, in words, what each constraint in the above formulation does.

2. Is the cost function we are using above convex?

3. Which constraints above are convex, and which are not?

4. Would it be possible to use a formulation like the above to path-plan for a 7DOF robot arm like the Baxter through an obstacle-rich workspace? Why or why not?