ECE276B Project 3: Infinite-Horizon Stochastic Optimal Control

Shivani Bhakta
Dept. of Electrical and Computer Engineering
University of California, San Diego
s9bhakta@ucsd.edu

I. Introduction

In this paper we consider safe trajectory tracking for a ground differential-drive robot. The goal of the this project is to design a control policy for this differential-drive robot to track the desired reference position and orientation trajectory, while avoiding collisions (two circular obstacles). We will explore two different ways to find the control policy, first one is receding-horizon certainty equivalent control (CEC) and second one is generalized policy iteration (GPI). This problem is important to address and can have applications in navigation, autonomous systems like cars, and robots to perform different tasks in a 3D space, while avoiding obstacles in the path. Orientation and position plays an important role in this application as our robot can be of any shape and size and we need to plan the path accordingly.

In this paper, we will use CEC (sub-optimal control scheme) to convert the stochastic optimal control problem to a deterministic optimal control problem, which can be solved more effectively. Since CEC problem will be reduced to a Non-liner program (NLP), we will use the NLP solver such as CasADi. In the second part, we will use policy iteration algorithm to compute the control policy. For this part, we will discretize the space and get the transition probabilities of the MDP for all states and control. We compare their performance in this paper. We also included the results of the different 3D environment with different obstacles.

II. Problem Formulation

In this project, we consider a differential-drive robot with state $x_t := (\mathbf{p}_t, \theta_t)$ at discrete-time $t \in \mathbb{N}$ where $\mathbf{p}_t$ is its position in $\mathbb{R}^2$ and θ_t is its orientation in $[-\pi, \pi)$. The robot is controlled by a velocity input $\mathbf{u}_t := (v_t, w_t)$ consisting of linear velocity $v_t \in \mathbb{R}$ and angular velocity (yaw rate) $w_t \in \mathbb{R}^3$. The discrete-time kinematic model of the differential-drive robot obtained from Euler discretization of the continuous-time kinematics with time interval $\tau > 0$ is:

$$\begin{aligned} x_{t+1} &= \begin{bmatrix} \mathbf{p}_{t+1} \\ \theta_{t+1} \end{bmatrix} = f(x_t, u_t, w_t) \\ &= \underbrace{\begin{bmatrix} \mathbf{p}_t \\ \theta_t \end{bmatrix}}_{x_t} + \underbrace{\begin{bmatrix} \tau cos(\theta_t) & 0 \\ \tau sin(\theta_t) & 0 \\ 0 & \tau \end{bmatrix}}_{G(x_t)} \underbrace{\begin{bmatrix} v_t \\ w_t \end{bmatrix}}_{u_t} + \mathbf{w}_t, t = 0, 1, 2, ... \end{aligned} \tag{1}$$

Here $w_t \in \mathbb{R}^3$ models the motions noise with the Gaussian distribution $\mathcal{N}(\mathbf{0}, diag(\sigma)^2)$ with standard deviation $\sigma = [0.04, 0.04, 0.004] \in \mathbb{R}^3$. The motion noise is independent across time and of the robot state $\mathbf{x}_t$. The kinematics model in (1) defined the probability density function $p_f(x_{t+1}|x_t, u_t)$ of $\mathbf{x}_{t+1}$ conditioned on $\mathbf{x}_t$ and $\mathbf{u}_t$ as the density of a Gaussian distribution with mean $\mathbf{x}_t + \mathbf{G}(x_t)\mathbf{u}_t$ and covariance $diag(\sigma)^2$. The control input $\mathbf{u}_t$ of the vehicle is limited to an allowable set of linear and angular velocities $\mathcal{U} := [0, 1] \times [1, 1]$.

The objective is to design a control policy for the differential-drive robot in order to track a desired reference position trajectory $\mathbf{r}_t \in \mathbb{R}^2$ and orientation trajectory $\alpha_t \in [\pi, \pi)$ while avoiding collisions with obstacles in the environment. There are two circular obstacles $\mathcal{C}_1$ centered at $(2, 2)$ with radius 0.5 and $\mathcal{C}_2$ centered at $(1, 2)$ with radius 0.5. Let $\mathcal{F} := [3, 3]^2 \backslash (\mathcal{C}_1 \bigcup \mathcal{C}_2)$ denote the free space in the environment.

It will be convenient to define an error state $\mathbf{e}_t := (\tilde{p}_t, \tilde{\theta}_t)$, where $\tilde{p}_t := \mathbf{p}_t - \mathbf{r}_t$ and $\tilde{\theta}_t := \theta_t - \alpha_t$ measure the position and orientation deviation from the reference trajectory, respectively. The equations of motion for the error dynamics are:

$$\begin{aligned} e_{t+1} &= \begin{bmatrix} \tilde{\mathbf{p}}_{t+1} \\ \widetilde{\theta}_{t+1} \end{bmatrix} = g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{w}_t) \\ &= \underbrace{\begin{bmatrix} \tilde{\mathbf{p}}_t \\ \widetilde{\theta}_t \end{bmatrix}}_{e_t} + \underbrace{\begin{bmatrix} \tau cos(\widetilde{\theta}_t + \alpha_t) & 0 \\ \tau sin(\widetilde{\theta}_t + \alpha_t) & 0 \\ 0 & \tau \end{bmatrix}}_{\tilde{G}(e_t)} \underbrace{\begin{bmatrix} v_t \\ w_t \end{bmatrix}}_{u_t} + \begin{bmatrix} \mathbf{r}_t - \mathbf{r}_{t+1} \\ \alpha_t - \alpha_{t+1} \end{bmatrix} + \mathbf{w}_t. \end{aligned} \tag{2}$$

We formulate the trajectory tracking with the initial time τ and initial tracking error $\mathbf{e}$ as a discounted infinite-horizon

stochastic optimal control problem:

$$V^*(\tau, e) = \min_{\pi} \mathbb{E}\left[\sum_{t=\tau}^{\infty} \gamma^t \left(\tilde{\mathbf{p}}_t^T \mathbf{Q} \tilde{\mathbf{p}}_t + q(1 - cos(\widetilde{\theta}_t))^2 + \mathbf{u}_t^T \mathbf{R} \mathbf{u}_t\right) | e_\tau = e\right] \quad (3)$$

s.t.

$e_{t+1} = g(t, e_t, \mathbf{u}_t, \mathbf{w}_t), \mathbf{w}_t \; \mathcal{N}(\mathbf{0}, diag(\sigma)^2)$
$t = \tau, \tau + 1, ...$
$\mathbf{u}_t = \pi(t, e_t) \in \mathcal{U}$
$\tilde{\mathbf{p}}_t + \mathbf{r}_t \in \mathcal{F}$

where $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ is a symmetric positive-definite matrix defining the stage cost for deviating from the reference position trajectory $\mathbf{r}_t, q > 0$ is a scalar defining the stage cost for deviating from the reference orientation trajectory α_t, and $\mathbf{R} \in \mathbb{R}^{2\times 2}$ is a symmetric positive-definite matrix defining the stage cost for using excessive control effort. We will compare two different approaches for solving the problem in (3): (a) receding-horizon certainty equivalent control (CEC) and (b) generalized policy iteration (GPI).

III. Technical Approach

A. Receding-horizon certainty equivalent control (CEC) control

CEC is a suboptimal problem control scheme that applies, at each stage, the control that would be optimal if the nouse variable $\mathbf{w}_t$ were fixed at their expected values (zero in our case). The main attractive characteristic of CEC is that it reduces a stochastic optimal control problem to a deterministic optimal control problem, which can be solved more effectively. Receding-horizon CEC, in addition, approximates an infinite-horizon problem by repeatedly solving the following discounted finite-horizon deterministic optimal control problem at each time step:

$$V^*(\tau, e) = \min_{u_\tau, ..., u_{\tau+T-1}} q(\mathbf{e}_{\tau+T}) + \sum_{t=\tau}^{\tau+T-1} \gamma^t \left(\tilde{\mathbf{p}}_t^T \mathbf{Q} \tilde{\mathbf{p}}_t + q(1 - cos(\widetilde{\theta}_t))^2 + \mathbf{u}_t^T \mathbf{R} \mathbf{u}_t\right) \quad (4)$$

s.t.

$e_{t+1} = g(t, e_t, \mathbf{u}_t, 0), t = \tau, ..., \tau + T + 1$
$\mathbf{u}_t \in \mathcal{U}$
$\tilde{\mathbf{p}}_t + \mathbf{r}_t \in \mathcal{F}$

Here $q(e)$ is a suitably chosen terminal cost. I chose my terminal cost as:

$$q(\mathbf{e}_{\tau+T}) = \tilde{\mathbf{p}}_t^T \mathbf{Q} \tilde{\mathbf{p}}_t + q(1 - cos(\widetilde{\theta}_t))^2 \quad (5)$$

Thus with the above equations, the receding-horizon CEC problem is now a non-linear program (NLP) of the form:

$$\min_{\mathbf{U}} c(\mathbf{U}, \mathbf{E})$$

s.t. $U_{lb} \leq U \leq U_{ub}$
$h_{lb} \leq h(\mathbf{U},\mathbf{E}) \leq h_{ub}$

where $\mathbf{U} := [\mathbf{u}_\tau^T, ..., \mathbf{u}_{\tau+T-1}^T]^T$ and $\mathbf{E} := [\mathbf{e}_\tau^T, ..., \mathbf{e}_{\tau+T}^T]^T$.

In my controller, I use NLP solver CasADi to find the control policy. First, I initialize my parameters to $\gamma = 0.99$ (recall $\gamma \in (0, 1)$), $T = 5, Q = [1515; 1515], R = I \in \mathcal{R}^2, q = 1$ and define my NLP for u. Previously, I had defined $T = 10, Q = [11; 11], Q = [1010; 1010], etc, R = [11; 11], etc$, but the first mentioned ones are the optimitized to the best of my knowledge. I use equations (2) and (3) to compute the stage cost for the next state for the current timestep to the T + timestep. This will give us the new position and orientation deviation, which can be used to find the terminal cost in eq(5), that we defined. Once we formulate the NPL question using these computes costs, we pass in the NLP declaration and the constriants and let the solver do its work.

The constraints for the NLP solver can be seen as Collision checking in PA 2, here our g and h constrained for NLP solver are the circle formula using the radius and center position of our two circular obstacles. Then, we also pass in the upper and lower bounds for the controls (left same as given in the start code). Note that when defining the NLP solver such that we get the optimal control u. We defined u as NLP parameter at the beginning before finding the stage cost in a for loop.

B. generalized policy iteration (GPI) control

In this part, we use the GPI algorithm to solve (3) directly. First, we discretize the state space to $\mathcal{X} = (n_t, n_x, n_y, n_\theta)$ and the control space $\mathcal{U} = (n_v, n_w)$ number of grid points. Here $n_t = 100$ for the provided reference trajectory with a period of 100. The position error $\tilde{p}$ and control input $\mathbf{u}$ is discretized over the regions $[-3, 3]^2$ and $\mathcal{U}$.

We will create the transition probabilities in the discretized markov decision process, for each discrete state **e** and each discrete control **u**, we can choose the next grid points **e** around the mean $g(t, e, u, 0)$, evaluate the likelihood of $\mathcal{N}(0, diag(\sigma)^2)$ at the chosen grid points, and normalize so that the outgoing transition probabilities sum to 1.

We will than also introduce an additional stage-cost for all the places that are in collisions with the two obstacles. Once we have the stage costs and state and control space defined as per the above criteria, we will use policy iteration or value iteration to get the solution for (3). This will work little similar to part 1, in a sense that we will have to compute P matrix for each iteration for each state.

I didn't have enough time to work on this part of the project. However above is how I would go above solve the problem in this part.

IV. Results

A. Receding-horizon certainty equivalent control (CEC) control

For this part of the problem, it can be seen from pictures the control policy is computing the orientation of the robot correcting and it is changing the orientation as per the projected trajectory. However, I wasn't able to figure out why the velocity was not being computed properly and why the robot was not moving or following the trajectory closer, as it changes the orientation during the cycle. Moreover, even though it doesn't follow the trajectory closely, it avoids the obstacles.

For CEC Quantitative results:

Objective
(scaled): 3.7820980228688779e+02
(unscaled): 3.7820980228688779e+02

Dual infeasibility:
(scaled): 8.6490004909107429e-11
(unscaled): 8.6490004909107429e-11

Constraint violation
(scaled): 0.0000000000000000e+00
(unscaled): 0.0000000000000000e+00

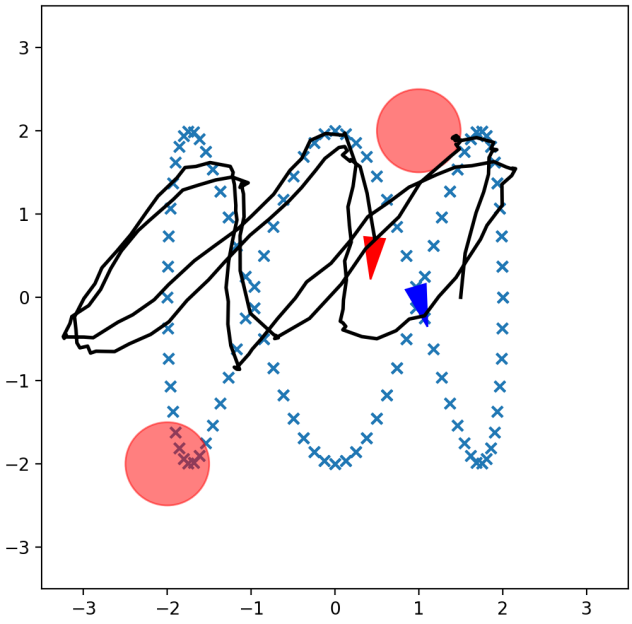
Complementarity
(scaled): 3.3619637758957417e-09
(unscaled): 3.3619637758957417e-09

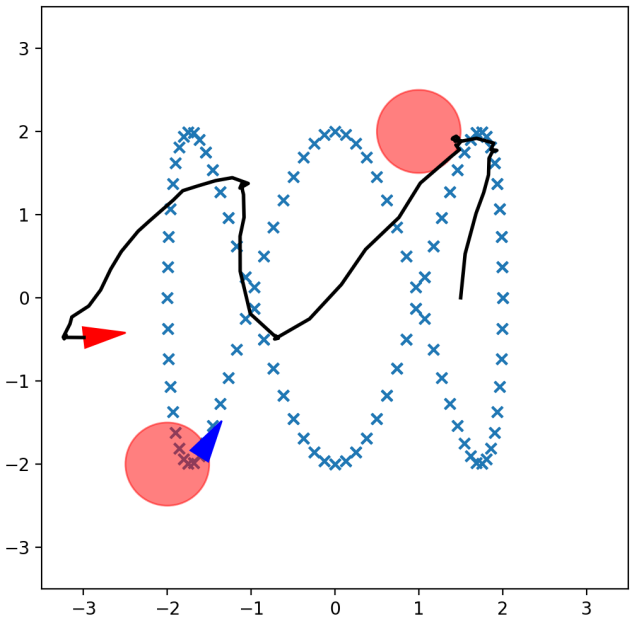
Overall NLP error:
(scaled): 3.3619637758957417e-09
(unscaled): 3.3619637758957417e-09

Evaluations
Number of objective function evaluations = 8
Number of objective gradient evaluations = 8
Number of equality constraint evaluations = 0
Number of inequality constraint evaluations = 8
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 8
Number of Lagrangian Hessian evaluations = 7
Total CPU secs in IPOPT (w/o function evaluations) = 0.002
Total CPU secs in NLP function evaluations = 0.000

Total time: 2.20855712890625

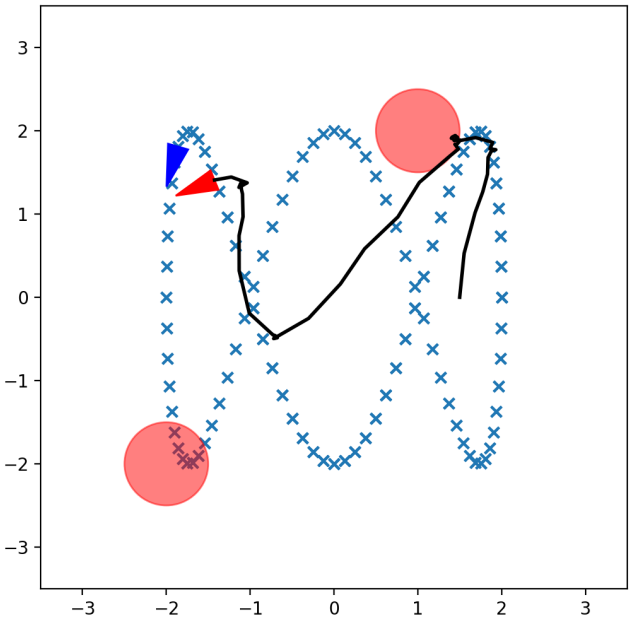
As it can be seen from the computation times, the NLP doesn't take that long to compute the optimal control policy and it is a pretty efficient in terms of computation. The error is also pretty low, even though we are ignoring the noise in the CEC formulation.

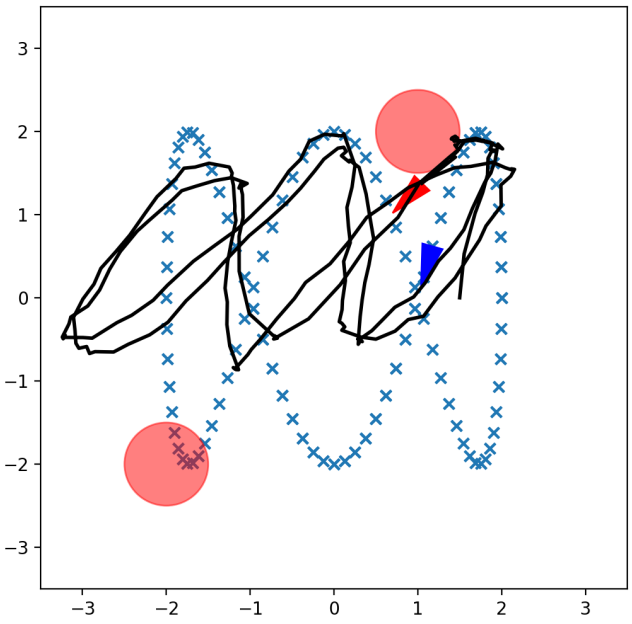




References

[1] For CasADi: https: //web.casadi.org/
[2] I used class lecture slides to write this report and work on the project.


3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3


3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3