ECE276B Project 3: Infinite Horizon Stochastic Optimal Control

Mustafa Shaikh
Department of Electrical & Computer Engineering
University of California, San Diego

I. Introduction

An optimal control problem is a problem in which a real world system is formulated as a mathematical problem, and for which the goal is to determine the best control strategy such that some measure of performance is maximized or minimized. There are often constraints on the system dynamics which also must be considered when finding this optimal policy.

In general, a physical system's behavior is described in continuous time by a set of differential equations. The control input, also continuous, represents the action applied to the system at each time to determine its next state. To simplify analysis, the continuous time kinematics, or dynamics, are often discretized and the resulting discrete time system is solved over a specified time horizon.

Optimal control problems can also be categorized at a high level into two distinct categories; deterministic and stochastic. The deterministic case is generally simpler to work with, as the lack of noise, or randomness, in the evolution of the system state allows us to determine the exact state trajectory based on an initial condition and a sequence of control inputs. Special cases of this include the linear quadratic regulator which even has a closed form solution for the optimal control sequence. The stochastic case requires iterative methods that take into account expected evolution of the state given a control input. In this project, we consider the stochastic case, but reduce it to a deterministic case to simplify analysis.

More specifically, our problem is to design a control policy for a differential-drive robot in order to track a desired reference trajectory while avoiding collisions with obstacles in the environment. We use 2 different approaches for this problem. The first is a receding-horizon certainty equivalent control problem (herein referred to as CEC). This approach directly solves an optimization problem. The second approach is generalized policy iteration (GPI) which solves the stochastic optimal control iteratively. The rest of the paper is organized as follows: the formulation of the problem is done in Section II, the details of the approaches taken to solve the problem and the algorithms are discussed in Section III, and finally, the performances of different algorithms are discussed in Section IV.

II. Problem Formulation

The objective is to generate an optimal trajectory for a differential-drive robot. It is important to ensure that this optimal trajectory avoids the obstacles in the environment and respects the constraints on the system dynamics. These constraints are represented by the motion model of the robot, as well as by the set of allowable control inputs.

The state of the robot consists of position, $p_t \in \mathbb{R}^2$ and orientation, $\theta_t \in [-\pi, \pi)$ and can be represented as a tuple, $x_t = (p_t, \theta_t)$ at a discrete time t. The control input, u_t can also be expressed as a tuple, (v_t, ω_t), where $v_t \in \mathbb{R}$ is the linear velocity and $\omega_t \in \mathbb{R}$ is the angular velocity of the agent. The discrete-time kinematic model of the differential-drive robot obtained from Euler discretization of the continuous-time kinematics with time step Δ is given as:

$$x_{t+1} = \begin{bmatrix} p_{t+1} \\ \theta_{t+1} \end{bmatrix} = f(x_t, u_t, w_t) \tag{1}$$

$$= \begin{bmatrix} p_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \Delta \cos\theta_t & 0 \\ \Delta \sin\theta_t & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} + \mathrm{w}_t \tag{2}$$

where $t \in \mathbb{N}$, and the motion noise $\mathrm{w}_t \in \mathbb{R}^3$ is modelled as a Gaussian distribution $N(0, \mathrm{diag}(\sigma)^2)$ where $\sigma = [0.04, 0.04, 0.004] \in \mathbb{R}^3$. Finally, the control input, u_t is restricted to a set of linear and angular velocities $U := [0, 1] \times [1, 1]$.

The kinematics model above describes the probability density function $p_f(x_{t+1}|x_t, u_t)$ as a Gaussian distribution with mean $x_t + G(x_t)u_t$, and covariance $\mathrm{diag}(\sigma)^2$. We also define an error state, $e_t = x_t - y_t$ where $y_t = (r_t, \alpha_t)$ is the reference state to be followed at time t. Then the equations of motion dynamics for the error states can be formulated as follows:

$$e_{t+1} = \begin{bmatrix} \tilde{p}_{t+1} \\ \tilde{\theta}_{t+1} \end{bmatrix} = g(t, e_t, u_t, \mathrm{w}_t) \tag{3}$$

$$= \begin{bmatrix} \tilde{p}_t \\ \tilde{\theta}_t \end{bmatrix} + \begin{bmatrix} \Delta \cos\tilde{\theta}_t + \alpha_t & 0 \\ \Delta \sin\tilde{\theta}_t + \alpha_t & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} + \begin{bmatrix} r_t - r_{t+1} \\ \alpha_t - \alpha_{t+1} \end{bmatrix} + \mathrm{w}_t \tag{4}$$

where $\tilde{p}_t = p_t - r_t$ and $\tilde{\theta}_t = \theta_t - \alpha_t$ with $r_t \in \mathbb{R}^2$ and $\alpha_t \in [-\pi, \pi)$.

With these definitions, we can express trajectory tracking problem with initial time τ and initial error e, as a discounted infinite-horizon stochastic optimal control problem as follows:

$$V^*(\tau, e) = \min_{\pi} \mathbb{E}[\sum_{t=\tau}^{\infty} \gamma^{t-\tau}(\tilde{p}_t^T Q \tilde{p}_t + q(1-\cos\tilde{\theta}_t)^2 + u_t^T R u_t)] \tag{5}$$

$$\begin{aligned} \text{s.t.} \quad & e_\tau = e \\ & e_{t+1} = g(t, e_t, u_t, \mathrm{w}_t) \\ & \mathrm{w}_t \sim N(0, \operatorname{diag}(\sigma)^2), \quad t = \tau, \tau+1, .. \\ & u_t = \pi(t, e_t) \in U \\ & p_t = \tilde{p}_t + r_t \in F \end{aligned}$$

where F is the free space in the environment, which is the remaining space when two obstacles are removed from the environment. The first obstacle, C_1, is centered at $(-2,-2)$, while the second obstacle, C_2 is centered at $(1,2)$, and the environment ranges between $[-3,3]^2$. Therefore, $F := [-3,3]^2 \setminus (C_1 \cup C_2)$. $Q \in \mathbb{R}^{2\times 2}$ is a positive definite matrix defining the cost of deviating from the reference trajectory, $q > 0$ is a scalar representing the cost of deviating from the reference orientation, and $R \in \mathbb{R}^{2\times 2}$ is a positive definite matrix representing the cost of applying excessive control input.

III. Technical Approach

Now that the problem has been formulated as an infinite-horizon stochastic optimal control problem, we explain our approach for generating the optimal control sequence that tracks the reference trajectory while avoiding the obstacles. We first present the CEC method, followed by the GPI method.

A. *Receding-horizon Certainty Equivalent Control (CEC)*

CEC converts the stochastic problem into a deterministic one by assuming that the value for the noise in the motion model is set to its expected value at all times, in this case 0. This allows us to solve for the optimal control trajectory by formulating the problem as a non linear program subject to constraints in the form of the (now) deterministic motion model, the bounds of the environment, and the bounds of the control input. Furthermore, the problem is considered a finite horizon case over the time span of the dataset (240 time steps with a 0.5 time interval).

The receding horizon concept refers to solving the problem repeatedly for each time step, picking out the first control input, applying it to the system, and recomputing the optimal policy at the next time step. The problem now becomes of the form:

$$V^*(\tau, e) = \min_{u_\tau,\dots,u_{\tau+T-1}} \sum_{t=\tau}^{\tau+T} \gamma^{t-\tau}(\tilde{p}_t^T Q \tilde{p}_t + q(1-\cos\tilde{\theta}_t)^2 + u_t^T R u_t)$$

$$\begin{aligned} \text{s.t.} \quad & e_{t+1} = g(t, e_t, u_t, 0) \quad t = \tau, \tau+1, .. \\ & u_t \in U \\ & p_t = \tilde{p}_t + r_t \in F \end{aligned}$$

Note that we have chosen not to use a terminal cost, and instead incorporated it into the stage cost of each time step since we assume it takes the same form as the stage costs.

The non-linear program discussed above becomes of the form:

$$\begin{aligned} & \min_U c(U, E) \\ \text{s.t.} \quad & U_{lb} \leq U \leq U_{ub} \\ & h_{lb} \leq h(U, E) \leq h_{ub} \end{aligned}$$

where $U = [u_\tau^T, ..., u_{\tau+T-1}^T]^T$ and $E = [e_\tau^T, ..., e_{\tau+T-1}^T]^T$.

We apply several constraints:

$$\begin{aligned} 0 \leq U_v \leq 1 \\ -1 \leq U_\omega \leq 1 \end{aligned}$$

where U_v represents the linear velocity component and U_ω represents the angular velocity component. To ensure that the robot avoids the obstacles, we set the l2-norm of the distance between the robot position and the obstacle centre to be greater than the radius of the obstacle:

$$\begin{aligned} h(u_t, e_t) = e_{t+1} = g(t, e_t, u_t, 0) \\ ||\tilde{p}_t + r_t||_2 > 0.5 \\ e_\tau = e_i \\ \tilde{\theta}_t \in [-\pi, \pi) \end{aligned}$$

The first equation represents the motion model, while the third equation represents the initial error, i.e. the error state of the robot after applying the motion model to the previous state and observing the new position of the robot and the last equation represents the angle wrap around constraint, i.e. the angle portion of the error state always must remain between $-\pi$ and π. To implement this, we use the following equation: $(\theta + \pi) \mod 2\pi - \pi$. Note that we only consider sliding windows of 10 time steps into the future to plan the optimal control sequence, rather than look at the entire remaining trajectory. This is to reduce the computational load while obtaining the solution.

To solve this problem, we use CasADi, which is an NLP solver available in python.

B. *Generalized Policy Iteration (GPI)*

Rather than solving a sub-optimal problem at each time step for the entire control sequence, then iteratively applying the control input to the system, we can directly optimize the stochastic system using GPI. GPI learns the long term value of being in each state and chooses a policy based on the best action to take in each state once this long term behaviour is learned.

Before applying GPI, we must first discretize the state space and control space. Our state space is of the form $(n_t, n_x, n_y, n_\theta)$ number of grid points with $n_t = 100$ since the reference trajectory is periodic with $T = 100$. We use a 30x30x30 grid for the x, y, θ variables. The resulting state space is quite large, with 2,700,000 total states. Similar to first approach, angle enforcement is applied, meaning we keep

$\theta \in [-\pi, \pi)$. In order to prevent collisions we only consider states in the free space.

The control space is discretized into (n_v, n_ω) points where n_v is discretization in linear velocity and n_ω is discretization in angular velocity. We chose $n_v = 10$ and $n_w = 20$ to ensure good resolution. The total number of allowable actions at each time step is therefore 200.

Due to time constraints, and to reduce computation time, we consider only the noiseless case in GPI. This means that the motion model is again deterministic, and applying a given control input leads to a particular next state with probability = 1. Due to our large state space, even computing a deterministic GPI solution takes significant time, and this will be discussed more later.

The resulting equation that represent the optimal policy under the GPI scheme is as follows:

$$V^*(e) = \min_{u \in U}(l(e,u) + \sum_{e' \in E'} p_f(e'|e,u)V^*(e')) \tag{6}$$

$$s.t. \quad e_{t+1} \sim p_f(.|e_t, \pi(e_t)) \tag{7}$$

$$e_t \in E \tag{8}$$

Where E is the state space, U is the action space, l(e,u) is the stage cost for the pair (e,u) and $p_f()$ is simply an indicator variable in the noiseless case that we consider.

To solve this problem, we use the value iteration method. The general form of the deterministic value iteration algorithm is shown below.

```
Data: State Space E, Action Space U, Stage Cost L,
      State Transition Matrix P
Result: Optimal Value Function $V^*(e)$, optimal policy
        $\pi(e)$
$V_0(e) \leftarrow 0 \ \forall e \in E$, i $\leftarrow$ 0
while True do
    i $\leftarrow i + 1$
    for state e in E do
        for action u in U do
            Q(e,u) $\leftarrow L(e,u) + V_{i-1}(e')$
        end
        V$_i(e) \leftarrow \min_u Q(e,u)$
        $\pi(e) \leftarrow \text{argmin}_u Q(e,u)$
    end
    if $|V_i(e) - V_{i-1}(e)| < \epsilon \quad \forall e \in E$ then
        break
    end
end
```

Algorithm 1: Value Iteration

The stage cost calculation is vectorized, but due to time constraints, the motion model is computed in a loop. Since the motion model and stage cost are both static, they are saved to disk and loaded in once for each iteration of Value Iteration (VI), rather than computing both each time. This saves significant computation time. However, we observed that the computation time for computing even 20 iterations of value iterations took over 3 hours. This is likely due to the large state space, and this made it infeasible to train value iteration for over 100 iterations as is required to obtain good convergence. We present the convergence results along with the time taken for the 20 iterations in the next section.

Since the optimal control may result in the evolution of the system state to a state that is not on the grid, we perform a (vectorized) search through the grid and snap the next state to the nearest point on the grid. We do this for each dimension x, y, θ separately. This means that the resulting trajectory will likely not be smooth due to minor fluctuations at each time step.

IV. Results

In this section we analyze 2 different algorithms and compare their performances. For testing we use the same environment for both of them. The robot is expected to follow the reference trajectory which is periodic by T=100.

A. Receding-horizon Certainty Equivalent Control (CEC)

As mentioned above, CEC algorithm tries to retrieve path online, meaning that it solves a separate problem at each step using the estimates for the future. One thing to note is that when we remove the noise from the system, CEC algorithm is able to smoothly track the trajectory. However, since it uses estimates for the future states, constraints and costs, it may deviate from the real future state values in the case of the noise. Some visualizations of the path followed with and without the noise can be found below.

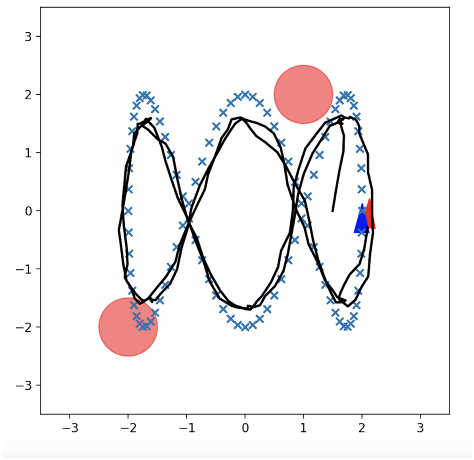

Fig. 1. The path followed by CEC algorithm when Q=R=I and q=1.

From the figure above, we see that the tracking of the reference trajectory is quite accurate. Furthermore, the optimal trajectory successfully avoids the obstacle in the upper right corner, and tangentially touch the obstacle in the lower left corner. We had to relax the constraint that the distance to the centre of the norm is strictly less than 0.5 since we ran into infeasible solution set issues during optimization. This constraint was relaxed to less than or equal to 0.5, and this is likely why the agent collides with the obstacle. The optimal solution should not be unstable in this way, and the reason for this must be investigated. It should also be noted that we applied the motion model constraints separately for each dimension x, y, θ rather than in a matrix equation form, and this could be one source of the issue. Additionally, due to time constraints we only experimented with

The early path figure shows an interesting result at the early stage of the trajectory when the agent approaches the upper left corner obstacle. Rather than follow the target to the apex of its trajectory and then turn, it actually turns before the target turns in order to avoid the obstacle. Since our planning horizon is the next 10 time steps, the optimization algorithm is aware of the future points of the target's trajectory and therefore turns before the target turns in order to stay close to the reference trajectory.

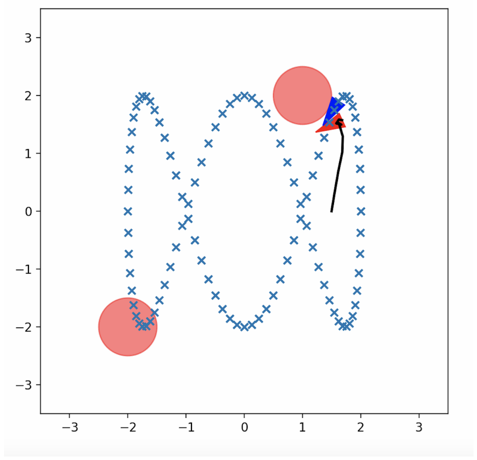

Fig. 2. Early path followed by agent near the first obstacle shows successful avoidance of the obstacle.

The table below shows that the CEC controller vastly outperforms the simple controller in terms of the error to the reference trajectory, although it takes significantly longer to plan for. This trade off is important and should be weighed carefully for various applications.

Table - 1: CEC Experiment Summary

Experiment description	Time taken	Error (L2-norm to ref)
Simple Controller	1 min	2180
CEC Controller	6 mins	785

B. Generalized Policy Iteration (GPI)

The GPI approach is offline, meaning that the optimal policy is completely calculated before running the system simulation. Discretization plays an important role in the performance of the algorithm; a finer resolution can lead to better results but comes at the cost of increased computation time. In fact, as mentioned earlier, it took over 3 hours to compute 20 iterations of value iteration. The motion model also took 30 minutes to compute for 2,700,000 states. Reducing the size of the state space is a good next step to explore algorithm performance. Furthermore, we can explore adaptive discretizations, in which the grid is finer closer to the mean of the next time step's error state under the motion model. This would help reduce the size of the state space and allow for more iterations of the VI algorithm.

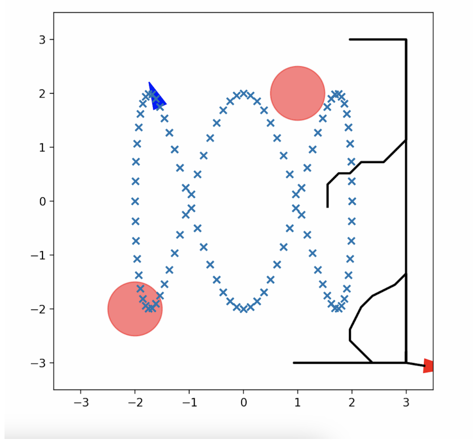

Fig. 3. Path followed by GPI with Q=R=I and q=1

As we can see from the image above, the optimal trajectory looks nothing like the reference trajectory. One possible issue is that VI was only run for 20 iterations, and this is not nearly enough to ensure convergence. Although the stage costs and motion model matrices were checked for accuracy, there could be bugs in there which caused this issue. Another source of error could be inappropriate mappings to valid grid states after applying the motion model with the optimal control policy. If the selection of the closest grid point does not obey the angle wraparound properly, the trajectory can be adversely affected. There could also be issues with the application of the motion model. The exact issue is unknown and further investigation would have to be done. The plot below shows convergence for 20 iterations. The average iteration time was 320s.

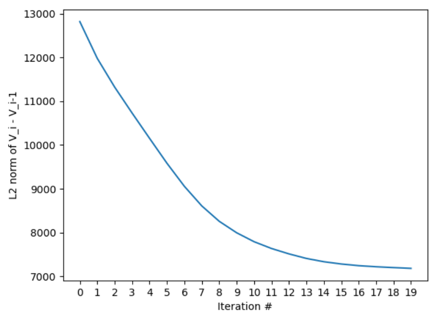

Fig. 4. Convergence of value function over 20 iterations. L2-norm of value function at time t - value function at time t-1

C. Conclusion

We have shown the performance of the CEC and GPI trajectory tracking with obstacles. The results show that CEC algorithm performs well when there in the noiseless case. The CEC algorithm is quite fast to compute the optimal policy.

The GPI algorithm on the other hand, takes quite long to compute the optimal policy. Discretization has a major effect on the performance of the algorithm. As the number of states increase the algorithm performs better but it takes longer time and memory to compute. We must conduct further work into the GPI algorithm to discover why it is not working as expected.

Overall, optimal control is an important problem in many fields of robotics. There are multiple approaches that can yield the optimal control sequence, and the various trade-offs between the methods must be considered when choosing the best approach.

References

[1] N. A. Atanasov, " Stochastic Shortest Path," in ECE 276B Lecture 10.

[2] N. A. Atanasov, "Bellman Equations," in ECE 276B Lecture 11.

[3] "Optimal control," Wikipedia, 24-Mar-2022. [Online]. Available: https://en.wikipedia.org/wiki/Optimal_control. [Accessed: 06-Jun-2022]