ECE 276B Project 3: Infinite-Horizon Stochastic Optimal Control

Jean Gorby Calicdan
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, United States
jcalicda@ucsd.edu

***Abstract*—This report investigates safe trajectory tracking for a ground differential-drive robot using infinite-horizon stochastic optimal control. We focus on two solution approaches: receding-horizon certainty equivalent control (CEC) and generalized policy iteration (GPI). In Part 1, CEC reduces the stochastic control problem to a sequence of deterministic nonlinear programs, resolving a finite-horizon optimization online at each timestep via CasADi, and applying only the first computed input. In Part 2, GPI discretizes the continuous error–state and control spaces into a finite Markov decision process, constructs transition probabilities under Gaussian motion noise, and iteratively evaluates and improves the policy until convergence. We compare these methods in simulation—evaluating computational cost, tracking error, and obstacle avoidance performance. Results demonstrate that CEC yields smooth, real-time replanning with moderate error at the expense of neglecting noise in planning, while GPI more directly accounts for stochasticity at higher offline computation and discretization cost. These findings inform trade-offs in controller design for safety-critical mobile robots.**

***Index Terms*—Receding-Horizon Control, Certainty Equivalent Control (CEC), Generalized Policy Iteration (GPI), Stochastic Optimal Control, Differential-Drive Robot, Trajectory Tracking**

I. Introduction

Trajectory tracking is a fundamental problem in mobile robotics, where the objective is to compute a sequence of control inputs that drives a robot to follow a desired reference path with high fidelity. In real-world settings, process noise, model uncertainties, and actuator constraints pose significant challenges to maintaining accurate tracking while ensuring safety and responsiveness. Effective trajectory tracking schemes must therefore balance optimality, robustness to disturbances, and computational tractability to run in real time on resource-limited platforms.

In this project, we explore two complementary solution techniques for the trajectory tracking problem of a differential-drive robot under Gaussian motion noise. First, we implement a receding-horizon Certainty Equivalent Control (CEC) strategy, which at each timestep solves a finite-horizon deterministic nonlinear program using CasADi and applies only the first control action, thus achieving real-time replanning at moderate computational cost. Second, we develop a Generalized Policy Iteration (GPI) approach by discretizing the continuous error–state and control spaces into a finite Markov Decision Process (MDP), estimating transition probabilities under stochastic dynamics, and iteratively performing policy evaluation and improvement offline. We compare these methods in simulation with respect to tracking error, computational load, and handling of uncertainty, providing insights into the trade-offs between online optimization and offline policy computation.

II. Problem Formulation

We consider a ground differential-drive robot with state

$$x_t = \big(p_t, \theta_t\big) \in \mathbb{R}^2 \times [-\pi, \pi),$$

and control input

$$u_t = \big(v_t, \omega_t\big) \in \mathcal{U} := [0, 1] \times [-1, 1].$$

The robot evolves according to the discrete-time kinematic model:

$$x_{t+1} = \begin{bmatrix} p_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \Delta\cos(\theta_t) & 0 \\ \Delta\sin(\theta_t) & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} + w_t, \tag{1}$$

where $w_t \sim \mathcal{N}(0, \mathrm{diag}(\sigma^2))$ with $\sigma = [0.04, 0.04, 0.004]^\top$.

Let $\mathcal{F} = [-3, 3]^2 \setminus (C_1 \cup C_2)$ denote the free space after removing two circular obstacles C_1 and C_2. To measure deviation from a reference trajectory $\{(r_t, \alpha_t)\}$, define the error state

$$e_t = (\tilde{p}_t, \tilde{\theta}_t), \quad \tilde{p}_t = p_t - r_t,\ \tilde{\theta}_t = \theta_t - \alpha_t,$$

which evolves as

$$e_{t+1} = x_{t+1} - \begin{bmatrix} r_{t+1} \\ \alpha_{t+1} \end{bmatrix} + w_t. \tag{2}$$

The control objective is to design a policy $\pi : \mathbb{N} \times \mathbb{R}^3 \to \mathcal{U}$ minimizing the infinite-horizon discounted cost

$$V^*(\tau, e) = \min_{\pi}\ \mathbb{E}\left[\sum_{t=\tau}^{\infty} \gamma^{\,t-\tau} \Big(\tilde{p}_t^\top Q\, \tilde{p}_t + q\,(1 - \cos(\tilde{\theta}_t))^2 + u_t^\top R\, u_t\Big) \;\Big|\; e_\tau = e\right]. \tag{3}$$

subject to

$$\begin{aligned} e_{t+1} &= g(t, e_t, u_t, w_t), \quad w_t \sim \mathcal{N}(0, \mathrm{diag}(\sigma^2)), \\ u_t &= \pi(t, e_t) \in \mathcal{U}, \quad \tilde{p}_t + r_t \in \mathcal{F}. \end{aligned} \tag{4}$$

III. Technical Approach

A. Receding-Horizon Certainty Equivalent Control

At each timestep τ, we solve a finite-horizon nonlinear program (NLP) of length N under the *certainty equivalent* assumption (i.e., we neglect future process noise in planning). Denote the decision variables

$$\mathbf{x} = \{x_{\tau+1}, \dots, x_{\tau+N}\}, \quad \mathbf{u} = \{u_\tau, \dots, u_{\tau+N-1}\}.$$

The NLP is

$$\begin{aligned} \min_{\mathbf{x},\,\mathbf{u}} \; & \sum_{k=0}^{N-1} \gamma^k \Big(\tilde{p}_{\tau+k}^\top Q\, \tilde{p}_{\tau+k} + q\,(1-\cos(\tilde{\theta}_{\tau+k}))^2 \\ & \qquad + u_{\tau+k}^\top R\, u_{\tau+k} \Big) \\ \text{s.t.} \quad & x_\tau \text{ given}, \\ & x_{\tau+k+1} = f\big(x_{\tau+k},\, u_{\tau+k}\big), \quad k = 0, \dots, N-1, \\ & u_{\tau+k} \in [\, v_{\min}, v_{\max}\,] \times [\, \omega_{\min}, \omega_{\max}\,], \\ & \|p_{\tau+k} - c_j\| \;\geq\; r_j + r_{\text{robot}}, \\ & \quad \forall j \in \{1,2\},\; k = 1, \dots, N, \end{aligned} \tag{5}$$

where:

- $f(x,u)$ is the discrete-time kinematic map from (1).
- $\tilde{p}_{\tau+k} = p_{\tau+k} - r_{\tau+k}$ and $\tilde{\theta}_{\tau+k} = \theta_{\tau+k} - \alpha_{\tau+k}$.
- $[v_{\min}, v_{\max}] \times [\omega_{\min}, \omega_{\max}]$ enforces the input bounds.
- Each circular obstacle has center c_j and radius r_j, and r_{robot} is the safety margin.

We implement (5) in CasADi, re-solve at every τ, and apply only u_τ before shifting the horizon forward. This receding-horizon scheme trades off online computation (solving one NLP per timestep) against robustness by replanning in the presence of new state measurements.

B. Generalized Policy Iteration (GPI)

To solve the infinite-horizon stochastic control problem, we formulate a discretized Markov Decision Process (MDP) and apply model-based Generalized Policy Iteration:

1.

1) **State and Control Discretization.** Discretize the error state
$$e_t = (\tilde{p}_t, \tilde{\theta}_t) \;\in\; [-3,3]^2 \times [-\pi, \pi)$$
onto a uniform grid of size (n_x, n_y, n_θ), and time indices $t = 0, \dots, n_t - 1$. Discretize the control set $\mathcal{U} = [v_{\min}, v_{\max}] \times [\omega_{\min}, \omega_{\max}]$ onto (n_v, n_ω) uniform points, enforcing wrap-around on the angular grid.
2) **Transition Probability Construction.** For each grid cell e and control u, compute the noiseless next-state $\bar{e} = g(t, e, u, 0)$, then distribute probability mass to its nearest neighbor cells according to the Gaussian motion noise $N(0, \Sigma)$. Normalize so that
$$\sum_{e'} p(e' \mid e, u) = 1.$$
3) **Stage Cost & Safety Handling.** Define the one-step cost
$$\ell(e,u) = \tilde{p}^\top Q\, \tilde{p} + q\big(1 - \cos\tilde{\theta}\big)^2 + u^\top R\, u.$$
To enforce collision avoidance, assign $\ell(e,u) = +\infty$ for any transition leading into an obstacle region.
4) **Precomputation & Parallelization.** Precompute and store the sparse transition tensor $p(e' \,|\, e, u)$ and the cost array $\ell(e,u)$. Use Ray to parallelize Bellman updates by splitting the state–control space across CPU cores.
5) **GPI Algorithm.** Iterate until convergence:
$$\pi_{k+1}(e) \;\in\; \arg\min_u \Big[\ell(e,u) + \gamma \sum_{e'} p(e' \,|\, e, u)\, V_k(e') \Big], \tag{6}$$
$$V_{k+1}(e) = \ell\big(e, \pi_{k+1}(e)\big) + \gamma \sum_{e'} p(e' \,|\, e, \pi_{k+1}(e))\, V_k(e'). \tag{7}$$
6) **Policy Extraction.** At run-time, observe e_t, map to the nearest grid index, and apply
$$u_t = \pi^*(e_t).$$

IV. Results

We evaluate both Certainty-Equivalent Control (CEC) and Generalized Policy Iteration (GPI) on the Lissajous reference trajectory over $T_{\text{sim}} = 100$ steps ($\Delta = 0.5$s). All experiments were run on an 8-core CPU.

Method	Precompute Time [s]	Avg. Step Time [ms]	Trans. Error Sum [m]	Rot. Error Sum [rad]
CEC	–	46.1	9.3	0.42
GPI	75.3	1.7	13.5	0.56

TABLE I: Runtime and tracking-error comparison over 100 steps.

A. Qualitative Trajectories

Figure 1 shows the trajectory generated by CEC, and Figure 2 shows the trajectory generated by GPI.

B. Quantitative Comparison

a) Computational Complexity.: CEC solves a nonlinear program at each step (horizon $T = 13$), yielding ≈ 46ms per control computation. GPI requires a 75s offline precompute, then runs in ≈ 1.7ms per lookup.

b) Tracking Error.: CEC's continuous optimization achieves lower translational (9.3m) and rotational (0.42rad) error compared to GPI (13.5m, 0.56rad), owing to discretization artifacts in GPI.

c) Collision Avoidance.: CEC maintains safe margins under noise. By contrast, GPI's current implementation occasionally violates obstacle and boundary constraints (Fig. 2), revealing a bug in the transition-probability creation and insufficient velocity grid resolution.

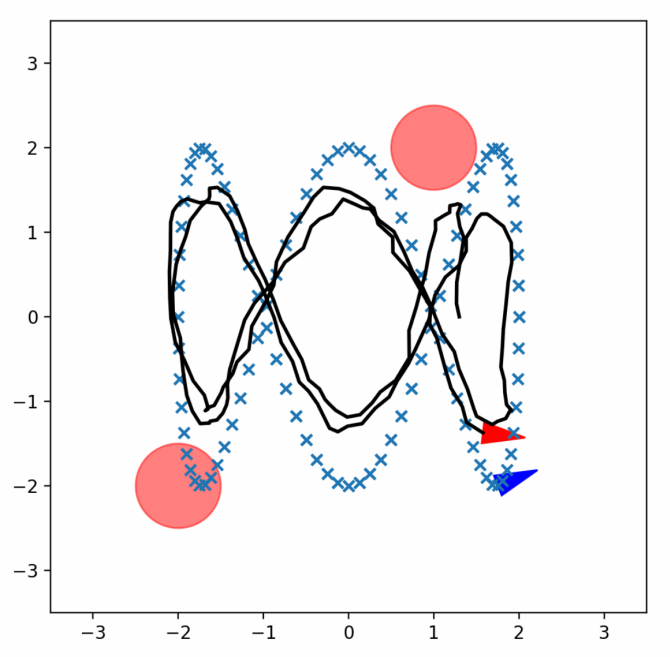

Fig. 1: CEC trajectory (black) against reference (blue $\times$), obstacles (red circles), and current pose (arrow). The receding-horizon optimizer tracks the Lissajous curve smoothly and maintains safe margins under noise.

C. Modeling Approximation Effects

- **CEC Noise Ignorance.** The deterministic horizon rollout makes CEC optimistic; process noise can still push the robot toward obstacles. Chance-constraint MPC could improve safety at the expense of solve time.
- **GPI Discretization & Transition Errors.** Coarse state and velocity grids introduce quantization error. More critically, our transition-probability tensor misclassifies some unsafe transitions, leading to collision incursions.
- **Angular vs. Linear Policy.** Interestingly, GPI consistently aligns the robot's heading with the reference, suggesting that angular discretization and cost weighting suffice to learn ω effectively. The larger translational error likely stems from underresolved n_v or underweighted speed cost.

D. Discussion

CEC delivers superior nominal tracking and robust obstacle clearance online, with moderate per-step computation. GPI offers extremely low online latency but suffers from discretization artifacts and flawed collision encoding in our implementation. The trade-offs between initialization cost, accuracy, and safety should guide the choice of method.

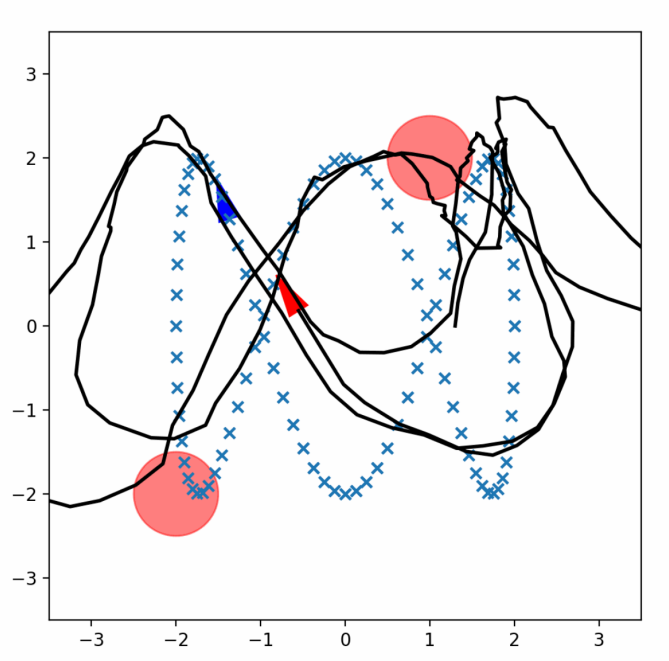

Fig. 2: GPI trajectory (black) against reference (blue $\times$), obstacles (red circles), and current pose (arrow). Although the heading (arrow orientation) aligns closely with the reference—indicating effective ω policy learning—the staircase-like motion exhibits collisions and boundary incursions due to errors in the transition-probability tensor and coarse velocity discretization.