Learning-Adaptive Control of Autonomous Vehicle Motion

Onur Calisir[1], Homayoon Beigi[1,2]

onur.calisir@columbia.edu | homayoon.beigi@columbia.edu

[1]Department of Mechanical Engineering, Columbia University, [2] Department of Electrical Engineering, Columbia University

Motivation

- Many autonomous robots operate in repetitive environments, following the same trajectories every day
- Classical Controllers → Need perfect models, but real-world conditions vary over time
- Adaptive Learning Controllers → Learn to minimize tracking error across repetitions
- Goal: Test Secant Learning-Adaptive Controller[1] and LSTR[2] on a custom-built robot

Learning-Adaptive Controllers

1) Learning-Adaptive Controller using the Generalized Secant Method[1]

Consider a discrete time, LTI system, control input u(k) is split into two components

$$y(k+1) = \hat{A}x(k) + \underbrace{\left[\hat{B}^1 \mid \hat{B}^2\right]}_{\hat{B}} \begin{bmatrix} u^1(k-1) \\ u^2(k) \end{bmatrix} + w(k) \longrightarrow y(k+1) = \underbrace{\begin{bmatrix} \hat{A} & \hat{B} \end{bmatrix}}_{S} \underbrace{\begin{bmatrix} x(k) \\ u(k) \end{bmatrix}}_{v(k)} + w(k)$$

Secant Adaptive Controller: runs in real time solving for $u^2(k)$ and updating the estimated system S_k to improve tracking the desired output $y^d(k+1)$

$$u^2(k) = \hat{B}_k^{2^+} \left(y^d(k+1) - \left[\hat{A}_k \mid \hat{B}_k^1\right] \begin{bmatrix} x(k) \\ u^1(k-1) \end{bmatrix} \right) \quad S_{k+1} = S_k + \frac{(y(k+1) - S_k v_k) z_k^T}{z_k^T v_k}$$

Learning-Adaptive Controller: adds a correction v^k that minimizes the accumulated tracking error along the whole trajectory

$$v^k = -P^{k+} e^k \qquad P^{k+1} = P^k + \frac{(e^{k+1} - e^k - P^k v^k) {z^k}^T}{{z^k}^T v^k}$$

2) Learning Self-Tuning Regulator (LSTR)[2]

The output at each step is modeled as: $\alpha(k) = \phi'(k)\theta(k) + v(k)$

The RLS updates are $\underline{\hat{\theta}}^{r+1}(k) = \underline{\hat{\theta}}^{r}(k) + \underline{L}^r(k)[\alpha^{r+1}(k) - \underline{\hat{\theta}}^{r'}(k)\phi^{r+1}(k)]$

where,

$$\underline{L}^r(k) = \frac{P^r(k)\underline{\phi}^{r+1}(k)}{\gamma + \underline{\phi}^{r+1'}(k)P^r(k)\underline{\phi}^{r+1}(k)} \qquad P^{r+1}(k) = \frac{[I - \underline{L}^r(k)\underline{\phi}^{r+1'}(k)]P^r(k)}{\gamma}$$

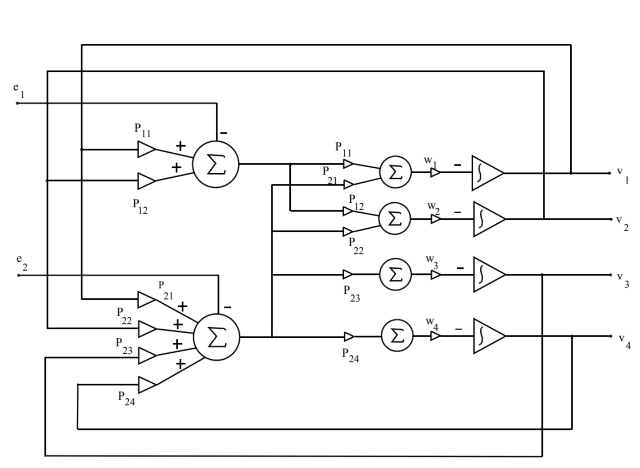

Figure 1. Parallel Network for Learning Adaptive Controller

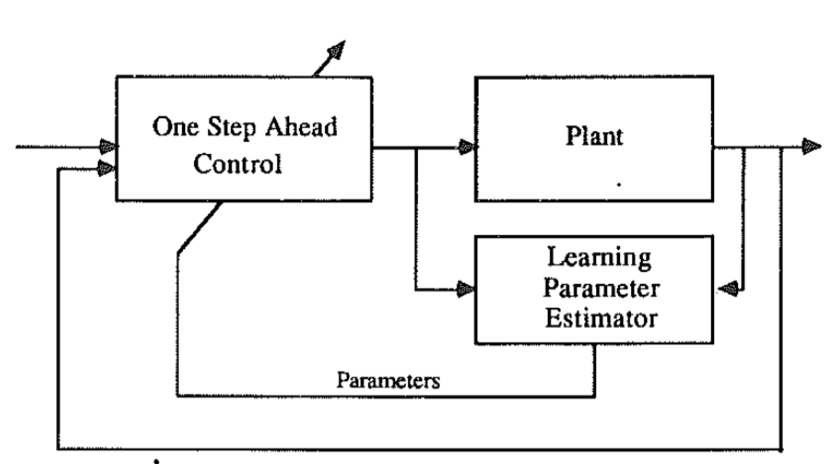

Figure 2. The learning self-tuning regulator

Robot Software and Hardware Stack

Sensor Interface & Measurements Node
Reads raw data: IMU (accel + gyro), encoders (pos + speed), OAK-D (depth + objects)

State Estimation and Robot Localization Node
Combines sensor inputs and previous state to estimate current robot state under noise

Goal Handling and Trajectory Planning Node
Receives desired goal positions and samples a path made from waypoints to the goal

Controller Node
Modular control layer allows switching between different controllers: PID, MPC, Adaptive

Motor Interface Node
Translates the control output into PWM signals to be sent to the H-bridge motor driver

Learning Controller Node
Corrects control sequence via previous repetition history

Components	Description
Car Chassis	ELEGOO Smart Robot Car Kit v4.0
Compute Unit & OS	Raspberry Pi 4B (4GB RAM), running Ubuntu 22.04 & ROS2
Depth Perception Camera	OAK-D Lite stereo depth camera with onboard depth sensing and object-detection NN
Inertial Measurement Unit (IMU)	MPU6050 IMU and BMI270 IMU inside OAK camera (acceleration and angular velocity)
DC Motors and Encoders	Four DC motors each with quadrature encoders (skid-steer model)

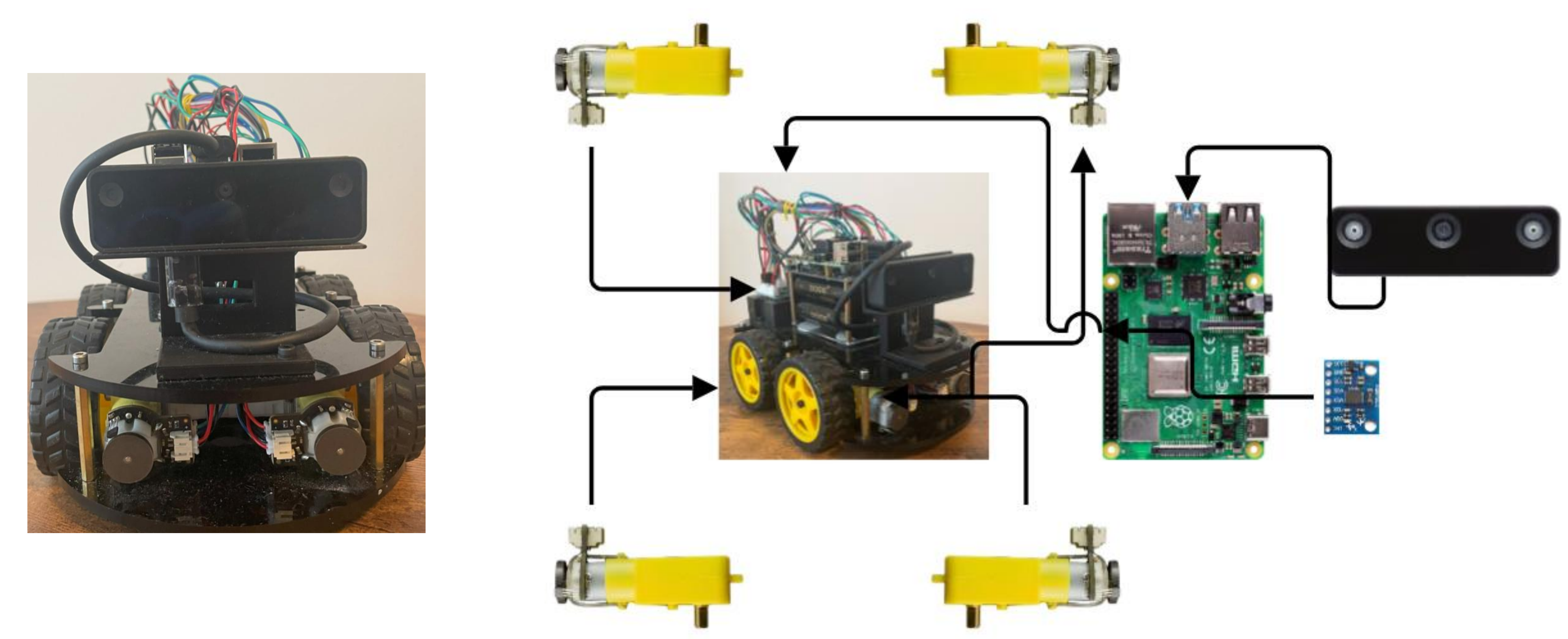

Figure 3. Wiring and component diagram

Current Results

- State vector $x(k) = [x(k), y(k), \theta(k), v(k), \omega(k)]^T \in \mathbb{R}^5$
- Control input $u(k) = [v_L(k-1), v_R(k-1), v_L(k), v_R(k)]^T \in \mathbb{R}^4$
- Measured output $y(k) = [x(k), y(k), \theta(k)]^T \in \mathbb{R}^3$
- ROS 2 system configured and tested on hardware
- PID controller implemented, velocity tracking reference 0.5 m/s, error =± 5 %
- Beigi's Learning-Adaptive not yet implemented but shown in prior work to reduce error significantly[1]

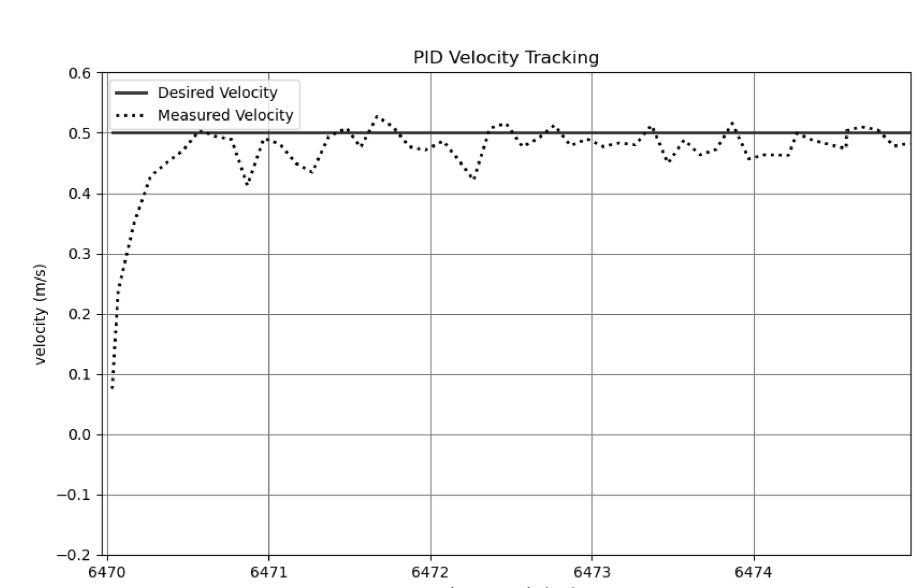

Figure 4. Robot velocity tracking using PID controller

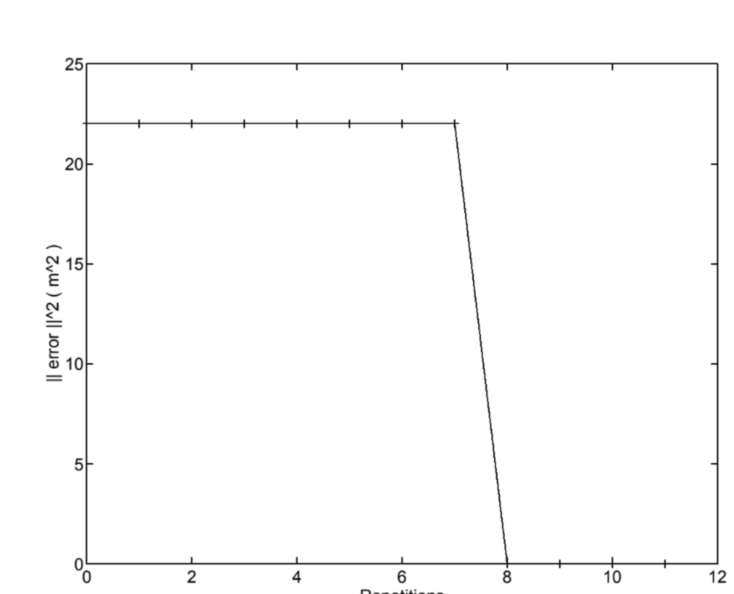

Figure 5. Learning-Adaptive on NLMSD With Rejections[1]

Future Work

- Complete controller integration and trajectory tracking within the ROS2 stack
- Implement the Learning-Adaptive Controller and the LSTR on real hardware
- Explore modern extensions using neural networks
- Experiment with long-memory adaptive structures

References

[1] Beigi, Homayoon S.M. "New Adaptive and learning-adaptive control techniques based on an extension of the generalized secant method." *Intelligent Automation and Soft Computing*, vol. 3, no. 2, Jan. 1997, pp. 171–184, https://doi.org/10.1080/10798587.1997.10750700.

[2] Li, C. James, et al. "Nonlinear piezo-actuator control by learning self-tuning regulator." *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, no. 4, 1 Dec. 1993, pp. 720–723, https://doi.org/10.1115/1.2899203.