ELD431

B.Tech Project

End Semester Presentation

Vision Based Cruise Control

Under the supervision of

Prof. Indra Narayan Kar

Presented By

Kartik Bansiwal (2021EE30743) & Shreysh Verma (2021EE30742)

Motivation

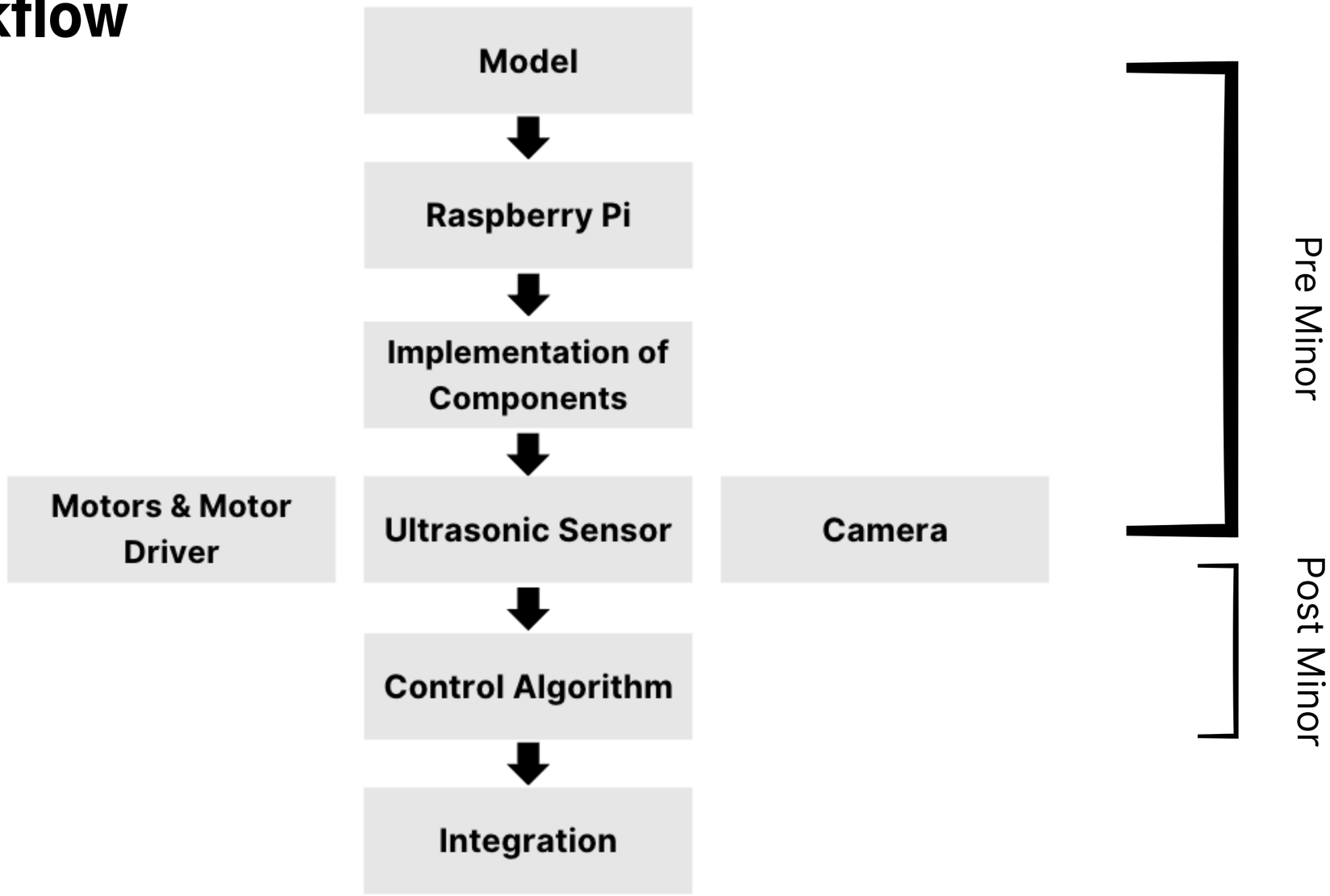
- Conduct applied research on control algorithms by developing an experimental setup from the ground up
- Gain deeper insight into the challenges and intricacies of control engineering in real-world applications
- The system will serve as a platform for future experimental research and testing in control engineering

Problem Statement

Practical Implementation of Adaptive Cruise Control Using 2-Wheeled Differential Drive Robots

- Objective: Explore the application of standard control algorithms on a two-wheel differential drive robot
- Focus: Adaptive cruise control
- Methodology: Implement and validate algorithms experimentally in a practical setup
- Outcome: Understand challenges and limitations of control systems in robots

Proposed Workflow

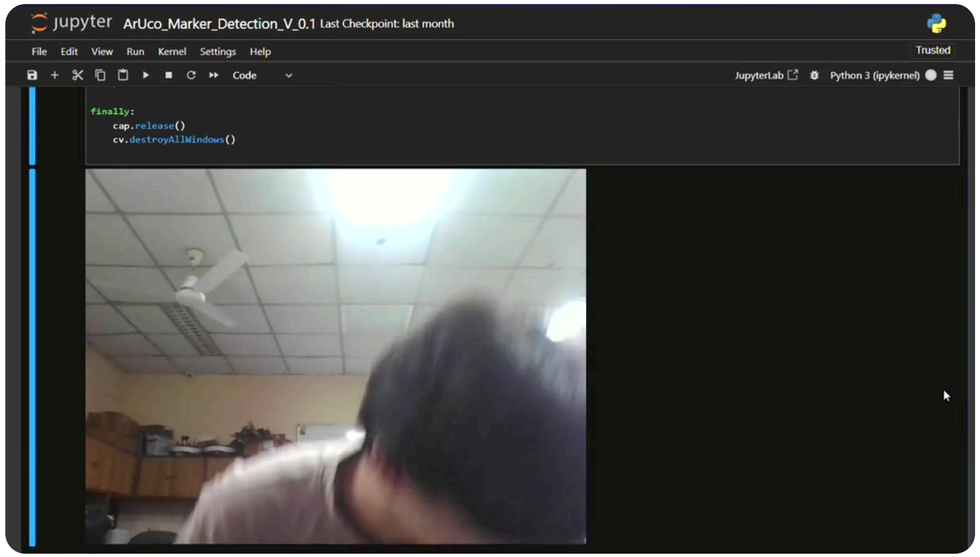


Feedback from Camera:

ArUco Marker Detection and Real Time Depth Estimation

Initialization (Monocular Camera Callibaration):

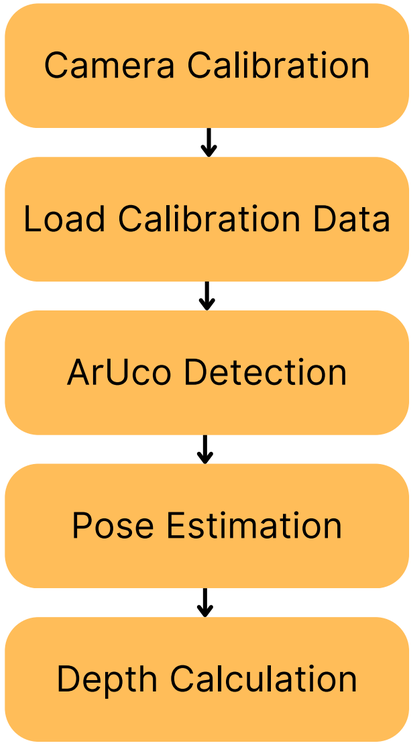
- The camera was calibrated with over 150 images of a chessboard grid.
- The grid was positioned at various locations in space for calibration.
- This calibration accounted for radial and tangential distortions.
- It enabled the extraction of the camera's intrinsic and extrinsic properties.



Implementation:

The implementation involved distance measurements of two ArUco markers, both individually and simultaneously, to evaluate the effectiveness of distance measurement and proper differentiation between the markers.

Accuracy: Accurate measurements within a margin of **+/-5 cm**

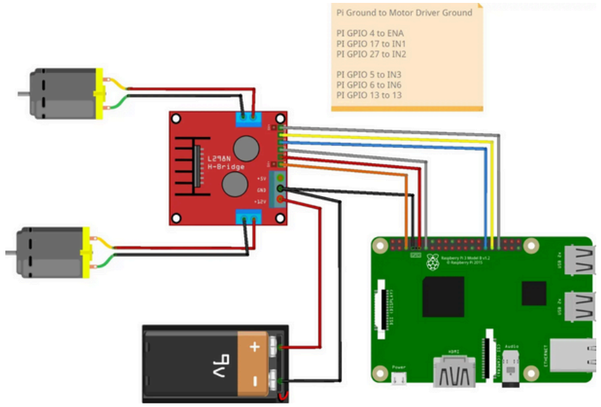


Ref: https://www.youtube.com/watch?v=YOpJrB6bQxo

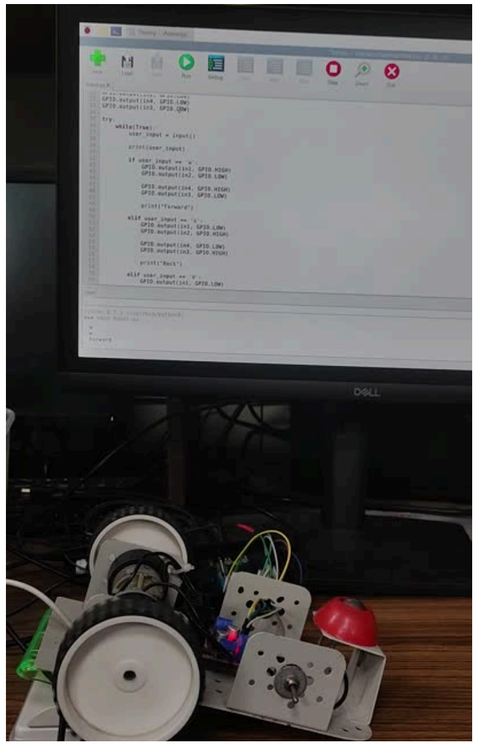
12V DC Motors with L298N Driver

Experimental Setup: Raspberry Pi 4B+, Chassis with two 12V DC motors controlled by L298N driver.
Code: Executed on Thonny IDE & Raspberry Pi 4B+, operated the motors at 75% speed in both forward and reverse directions.

The motors were driven by a **PWM** signal with a frequency of **100Hz** and a duty cycle of **75%**, resulting in **75%** of the motor's maximum speed.



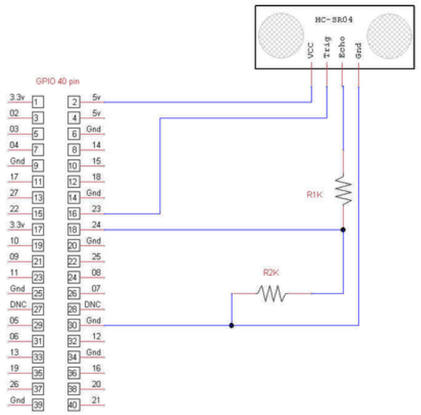
Circuit Diagram



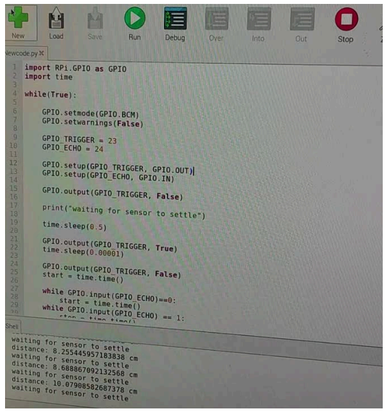
Demonstration + Code + Result

Ultrasonic Sensor

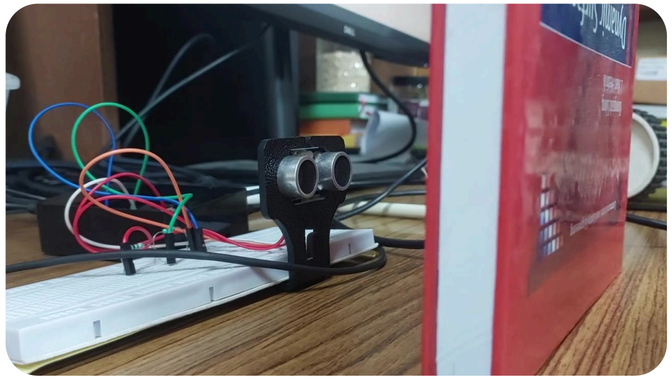
Experimental Setup: HC-SR04 Ultrasonic Sensor, 1k Ω resistor, 2k Ω resistor & Raspberry Pi 4B+
Accuracy: Accurate measurements within a margin of **+/- 1 cm**



Circuit Diagram



Demonstration



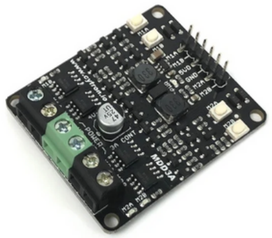
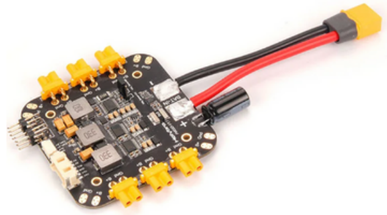
Code + Result

Ref: https://www.collvy.com/blog/controlling-dc-motor-with-raspberry-pi , https://forums.raspberrypi.com/viewtopic.php?t=206751

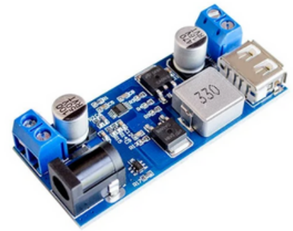
Components Required

Hardware: Two Wheeled Chassis with Castor Wheel, 12V DC Motors with Encoder, L298N Motor Driver, Raspberry Pi 4B+, PiCam, HC SR04 Ultrasonic Sensor, Power Distribution Board, Buck Converter, Battery & Battery Charger, Jumper Wires, Banana Connectors









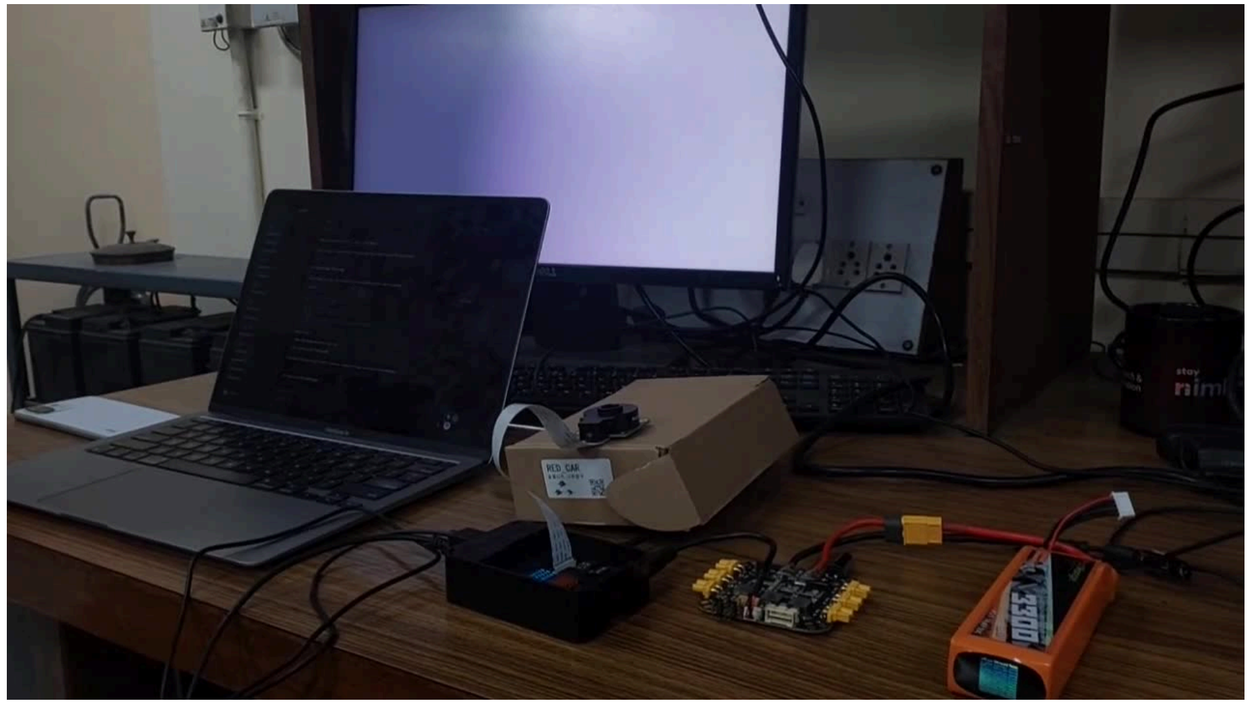
Software: Raspberry OS, MATLAB, Simulink, Jupyter

List of products ordered on ROBU: https://drive.google.com/file/d/1w5brDAPwtRhP3DqFW8URxeseaLbh_wmy/view?usp=sharing

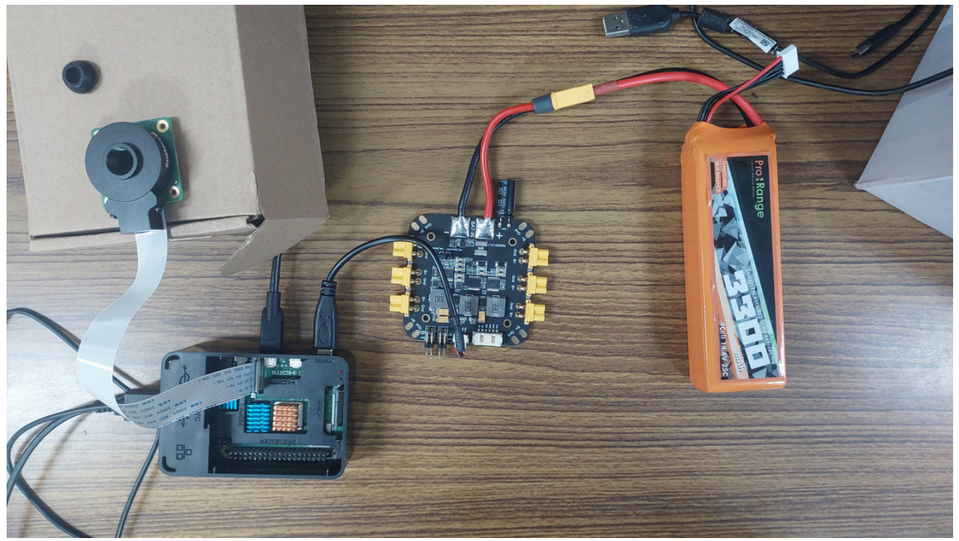
Component Testing

Experimental Setup: Raspberry Pi 4B+, Holybro PM03D Power Module , Orange 14.8V 3300mAh 25C 4S Lithium Polymer Battery Pack , Raspberry Pi HQ Camera - M12 Mount

Code: The program was executed on the Thonny IDE using a Raspberry Pi 4B+. The camera was operated with a 10-second delay, providing a preview of the image for 10 seconds before saving the picture to the system



Demonstration + Code + Result



Connections

Research Paper Implementation: Control Design

Simultaneous Stabilization and Tracking of Nonholonomic Mobile Robots: A Lyapunov-Based Approach

Yaonan Wang, Zhiqiang Miao, Hang Zhong, and Qi Pan

Abstract—A smooth time-varying controller is proposed to simultaneously address the stabilization and tracking problems of nonholonomic mobile robots for most admissible reference trajectories without switching. The controller is developed with the aid of a delicately designed time-varying signal and Lyapunov method. Computational simplification and asymptotic convergence of regulation or tracking errors are achieved by the proposed controller. Our approach provides an interesting way to unify the existing results on point stabilization and trajectory tracking of mobile robots. The simulation and experimental results on a wheeled mobile robot are presented to demonstrate the effectiveness of the proposed controller.

Ref: Wang, Y., Miao, Z., Zhong, H. and Pan, Q. “Simultaneous stabilization and tracking of nonholonomic mobile robots: A Lyapunov-based approach”, IEEE Transactions on Control Systems Technology, 2015.

Robot Kinematic Model

The configuration of the mobile robot can be described by a vector of generalized coordinates

$q = [x, y, \theta]^T$ where (x, y) is the position of the mobile robot and θ is the heading angle.

The nonholonomic constraint subjected to the mobile robot is given as

$$\dot{x} \sin\theta - \dot{y} \cos\theta = 0.$$

Under such a constraint condition, the kinematic model of the mobile robot can be described by

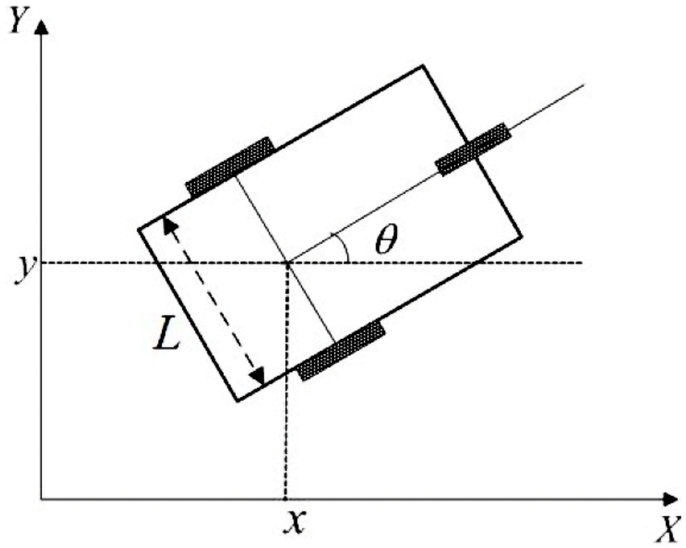
$$\dot{x} = v \cos\theta \quad \dot{y} = v \sin\theta \quad \dot{\theta} = \omega$$

Assume that the reference trajectory is admissible (or feasible), and generated by the following virtual robot:

$$\dot{x}_d = v_d \cos\theta_d \quad \dot{y}_d = v_d \sin\theta_d \quad \dot{\theta}_d = \omega_d$$

where

$$q_d = [x_d, y_d, \theta_d]^T$$



Two-wheeled non holonomic mobile robot

The control objective considered in this paper is to design a (single) continuous feedback control law (v,ω) for the mobile robot that simultaneously solves stabilization and tracking for a desirable reference trajectory, in particular

$$\lim_{t\to\infty} (q(t) - q_d(t)) = 0.$$

As often done in tracking control of mobile robots, consider the following tracking errors as $q_e = T(q)(q - q_d)$

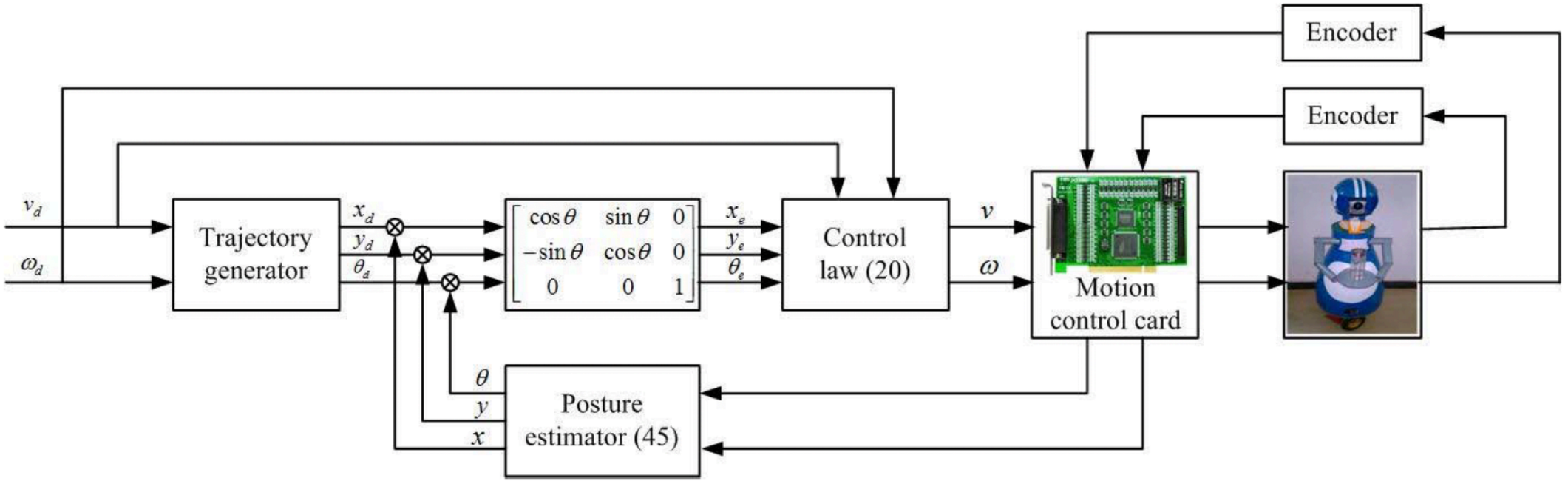
$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_d \\ y - y_d \\ \theta - \theta_d \end{bmatrix}.$$

Then, the following error dynamics can be attained

$$\begin{aligned} \dot{x}_e &= +\omega y_e + v - v_d \cos\theta_e \\ \dot{y}_e &= -\omega x_e + v_d \sin\theta_e \\ \dot{\theta}_e &= \omega - \omega_d. \end{aligned}$$

Ref: Wang, Y., Miao, Z., Zhong, H. and Pan, Q. “Simultaneous stabilization and tracking of nonholonomic mobile robots: A Lyapunov-based approach”, IEEE Transactions on Control Systems Technology, 2015.

Block Diagram



Ref: Wang, Y., Miao, Z., Zhong, H. and Pan, Q. “Simultaneous stabilization and tracking of nonholonomic mobile robots: A Lyapunov-based approach”, IEEE Transactions on Control Systems Technology, 2015.

Control Design

1. Time-Varying Signal

To facilitate the control design, we define a time-varying signal $\alpha = \alpha(t, x_e, y_e)$ as

$$\alpha = \rho(t)h(t, x_e, y_e) \tag{10}$$

ρ(t): Decays over time, modulated by reference velocities:

$$\dot{\rho} = -(|v_d(t)| + |\omega_d(t)|)\rho, \quad \rho(0) = 1$$

h(t,xe,ye): Satisfies properties to ensure smooth transitions and boundedness:

$$h(t, 0, 0) = 0 \quad |h(t, x_e, y_e)| \leq h_0$$

2. Modified Error Dynamics

Substitute α into the dynamics:

$$\begin{aligned}
\dot{x}_e &= \omega y_e + v - v_d \cos\theta_e \\
\dot{y}_e &= -\omega x_e + v_d(\sin\theta_e - \sin\alpha) + v_d \sin\alpha \\
\dot{\bar{\theta}}_e &= \omega - \omega_d - \dot{\alpha}.
\end{aligned}$$

$$\begin{aligned}
f_1 &= \frac{\sin\theta_e - \sin\alpha}{\theta_e - \alpha} \\
&= \frac{\sin\bar{\theta}_e \cos\alpha + (\cos\bar{\theta}_e - 1)\sin\alpha}{\bar{\theta}_e}.
\end{aligned}$$

3. Control Law

$$\begin{aligned}
v &= -k_1 x_e + v_d \cos\theta_e \\
\omega &= -k_2\bar{\theta}_e + \omega_d - k_0 v_d y_e f_1 + \dot{\alpha}
\end{aligned}$$

4. Lyapunov Function

$$V_1 = \frac{1}{2}k_0(x_e^2 + y_e^2) + \frac{1}{2}\bar{\theta}_e^2$$

Ref: Wang, Y., Miao, Z., Zhong, H. and Pan, Q. “Simultaneous stabilization and tracking of nonholonomic mobile robots: A Lyapunov-based approach”, IEEE Transactions on Control Systems Technology, 2015.

Simulation and Experimental Results

The following four cases are simulated.

Case 1: Set-point stabilization: $v_d = 0, \omega_d = 0$.
Case 2: Approach to a point: $v_d = 3e^{-0.2t}, \omega_d = e^{-0.6t}$
Case 3: Tracking a line path: $v_d = 2, \omega_d = 0$.
Case 4: Tracking a circle: $v_d = 2, \omega_d = 1$.

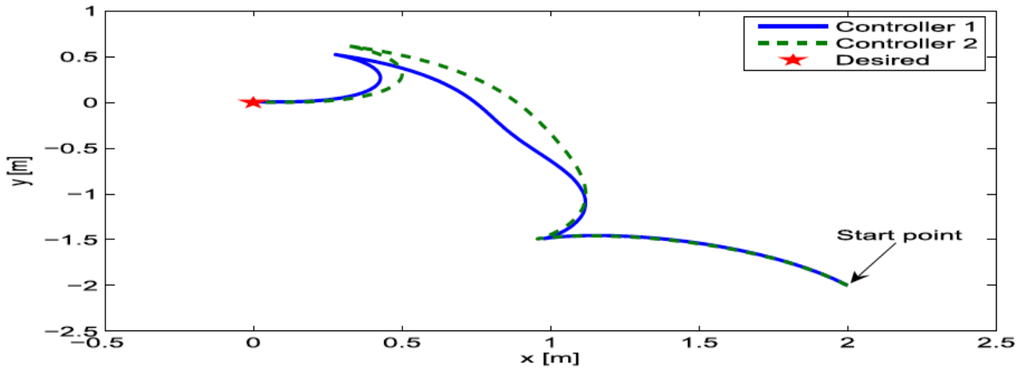
The reference trajectory $q_d(t) = [x_d(t), y_d(t), \theta_d(t)]^T$ is generated by the desired velocities $v_d(t)$ and $\omega_d(t)$ with the initial conditions $q_d(0) = [0, 0, 0]^T$.

In the simulation, the initial positions and velocities of the wheeled robot are picked as $q(0) = [2, -2, -1]^T$ and $[v(0), \omega(0)]^T = [0, 0]^T$. The control parameters are set as $k_0 = 1$, $k_1 = 6$, and $k_2 = 5$. Controllers with two different candidates of the nonlinear time-varying function h are simulated.

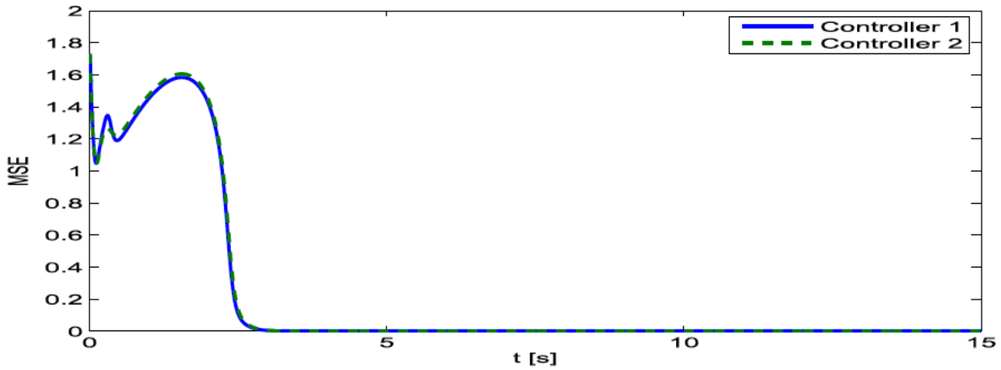
Controller 1: $h = 8 \tanh(x_e^2 + y_e^2) \sin(t)$.
Controller 2: $h = 6 \arctan(x_e^2 + y_e^2) \sin(t)$.

Ref: Wang, Y., Miao, Z., Zhong, H. and Pan, Q. “Simultaneous stabilization and tracking of nonholonomic mobile robots: A Lyapunov-based approach”, IEEE Transactions on Control Systems Technology, 2015.

Simulation and Experimental Results

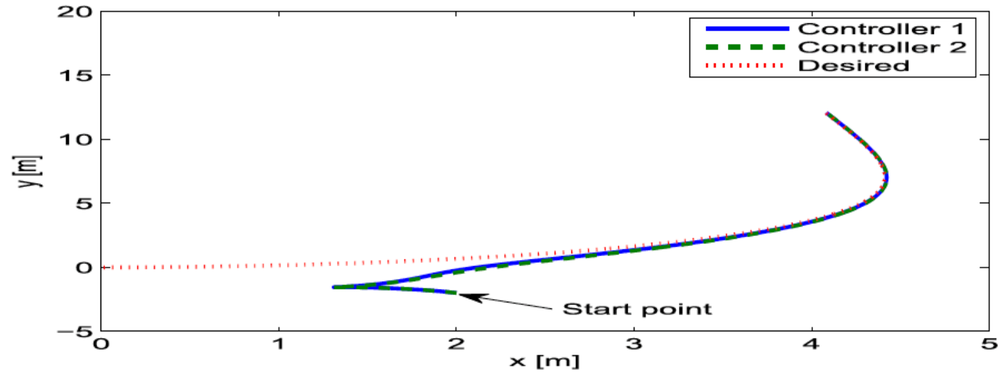


(a)

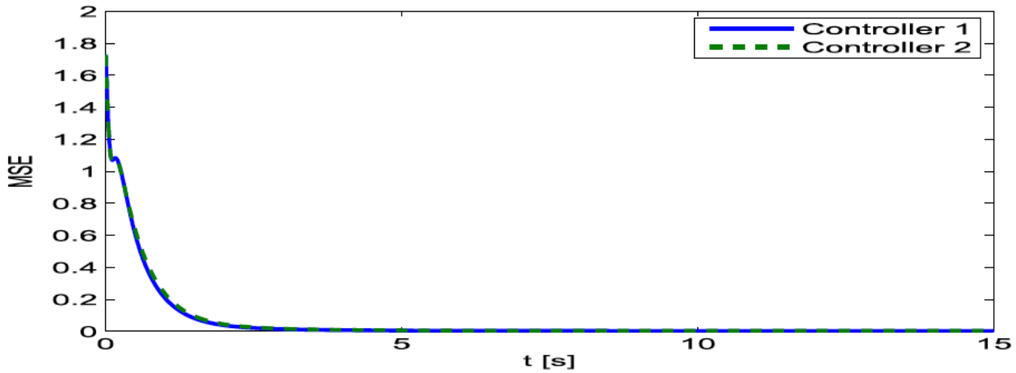


(b)

Simulation results of Case 1. (a) Robot position in the (x, y) plane. (b) MSE control performance.



(a)

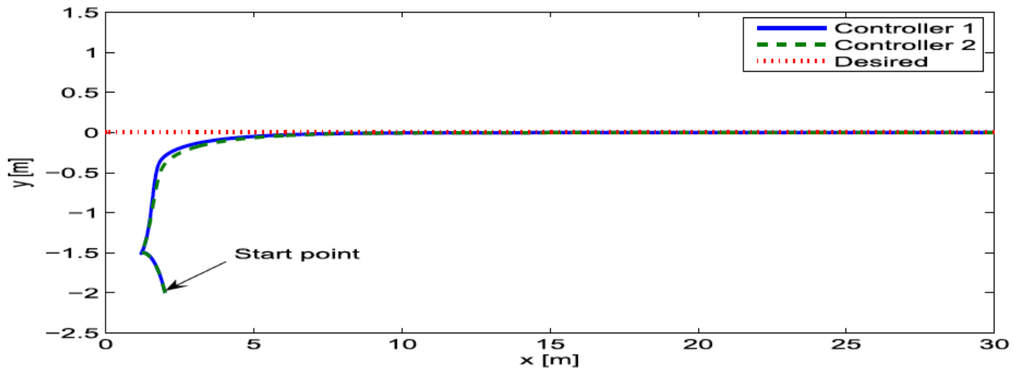


(b)

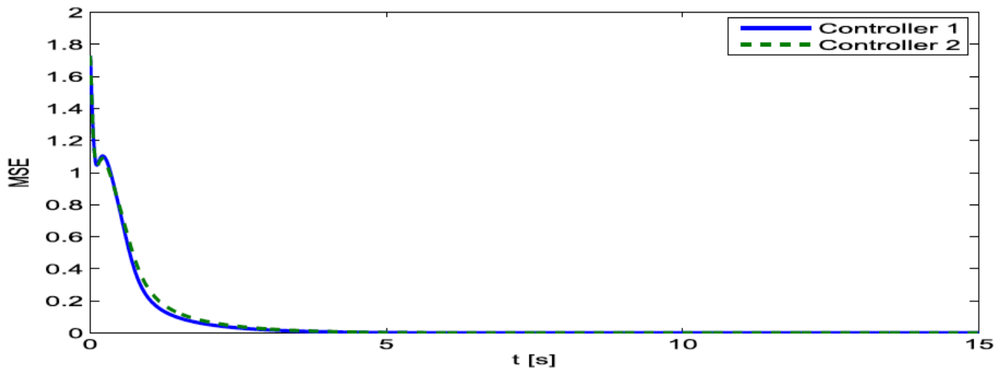
Simulation results of Case 2. (a) Robot position in the (x, y) plane. (b) MSE control performance.

Ref: Wang, Y., Miao, Z., Zhong, H. and Pan, Q. “Simultaneous stabilization and tracking of nonholonomic mobile robots: A Lyapunov-based approach”, IEEE Transactions on Control Systems Technology, 2015.

Simulation and Experimental Results

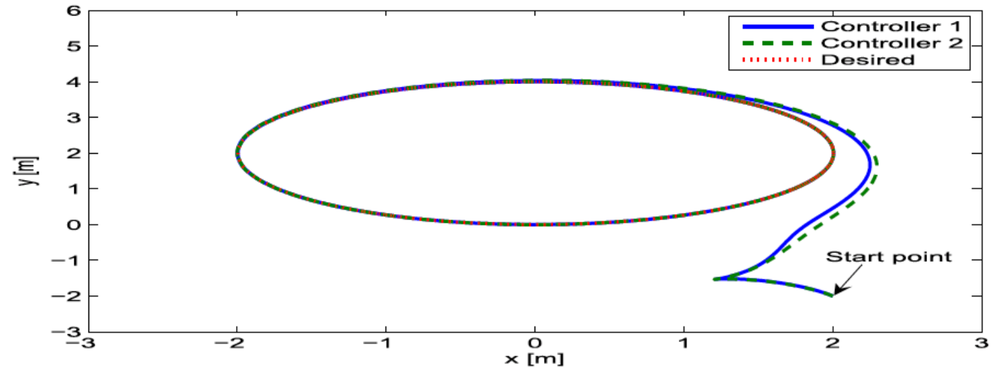


(a)

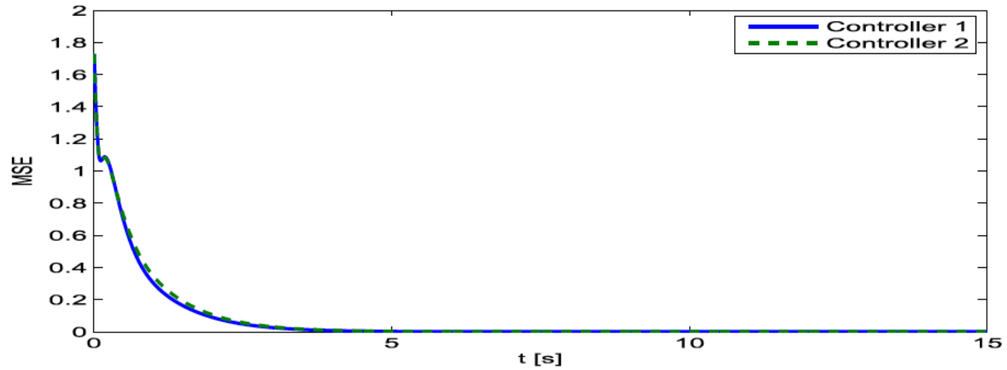


(b)

Simulation results of Case 3. (a) Robot position in the (x, y) plane. (b) MSE control performance.



(a)



(b)

Simulation results of Case 4. (a) Robot position in the (x, y) plane. (b) MSE control performance.

Ref: Wang, Y., Miao, Z., Zhong, H. and Pan, Q. “Simultaneous stabilization and tracking of nonholonomic mobile robots: A Lyapunov-based approach”, IEEE Transactions on Control Systems Technology, 2015.

Conclusion

This project successfully demonstrated the design and implementation of a two-wheeled differential-drive robot capable of adaptive cruise control and obstacle avoidance using vision and ultrasonic sensors. The system combined theoretical control concepts, like Lyapunov-based stabilization, with practical hardware and software integration to achieve trajectory tracking and collision avoidance in real-time.
Through simulations and experiments:

1. The robot achieved trajectory tracking (set-point stabilization, linear, and circular paths) with minimal mean-squared error (MSE).
2. ArUco markers and monocular camera calibration provided precise pose estimation, enhancing the robot's navigation accuracy.

The project highlighted the robustness of Lyapunov-based control laws in managing the trade-off between stabilization and tracking. Despite hardware and sensor limitations, the system demonstrated consistent performance across different scenarios.

Future Prospect

Continuing this work in BTP-2 with the goal of integrating the software and hardware components, which have so far been developed individually, and aiming for a publication upon successful integration

This project serves as a stepping stone for further research into autonomous mobile robot navigation, bridging the gap between theoretical control strategies and practical robotic applications.

ELD431

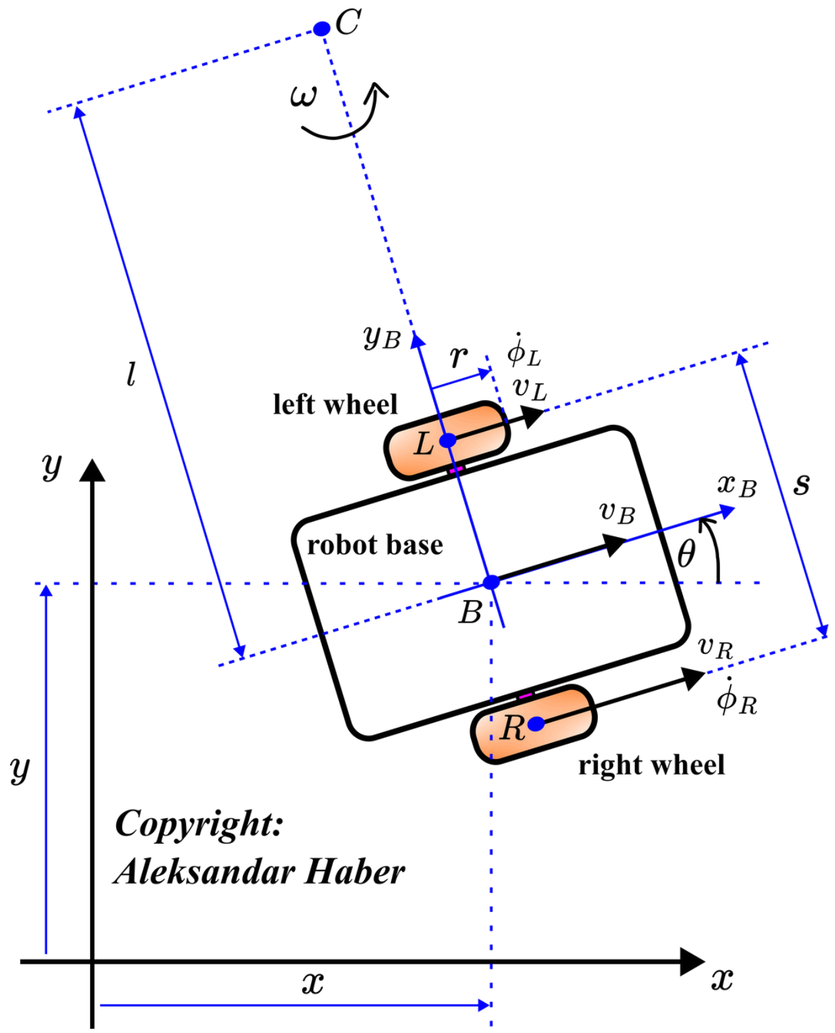
B.Tech Project

End Semester Presentation

Thank You

@Control Group, IIT Delhi

Model Kinematics



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r\cos(\theta)}{2} & \frac{r\cos(\theta)}{2} \\ \frac{r\sin(\theta)}{2} & \frac{r\sin(\theta)}{2} \\ -\frac{r}{s} & \frac{r}{s} \end{bmatrix} \begin{bmatrix} \dot{\phi}_L \\ \dot{\phi}_R \end{bmatrix}$$

$\dot{x}$ is the projection of the velocity v_B on the x axis.

$\dot{y}$ is the projection of the velocity v_B on the y axis.

r is the radius of the wheels.

s is the distance between the points L and R.

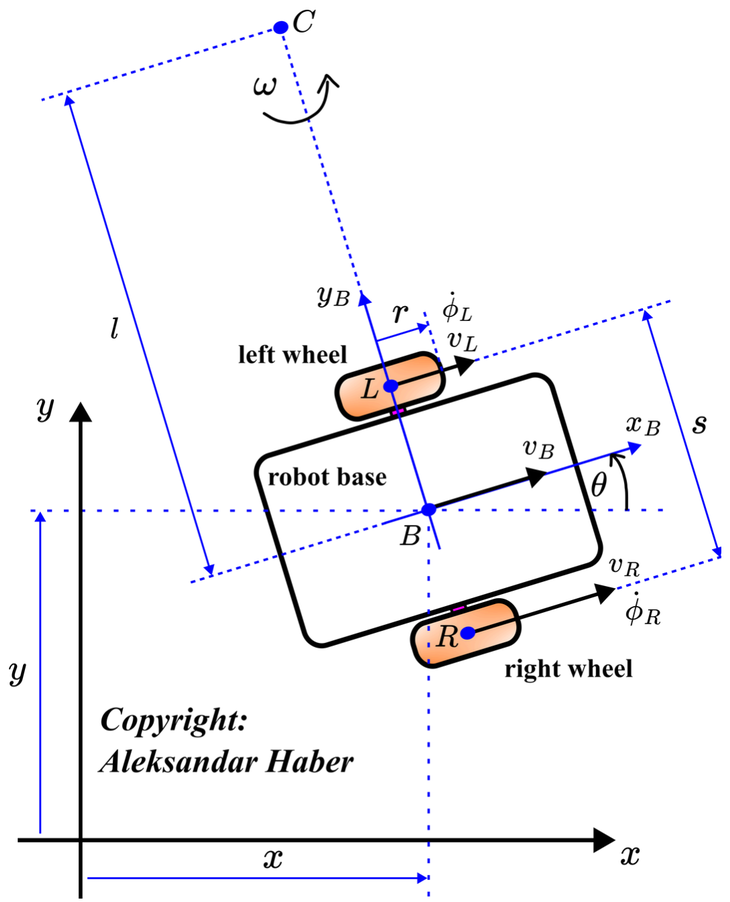
$\dot{\phi}_L$ is the angular velocity of the left wheel.

$\dot{\phi}_R$ is the angular velocity of the right wheel.

θ is the angle of rotation of the robot which is at the same time the angle between the body frame and the inertial frame

Ref: https://aleksandarhaber.com/clear-and-detailed-explanation-of-kinematics-equations-and-geometry-of-motion-of-differential-wheeled-robot-differential-drive-robot/

Model Kinematics: Derivation



That is, we start from the assumption that the following quantities and parameters are known $\dot{\phi}_L$, $\dot{\phi}_R$, s, and r, and we want to determine $\dot{x}$, $\dot{y}$, and $\dot{\theta}$.

$$\begin{aligned} v_L &= \omega(l-\frac{s}{2}) \\ v_R &= \omega(l+\frac{s}{2}) \end{aligned} \longrightarrow \omega = \frac{v_L}{l-\frac{s}{2}} \longrightarrow \begin{aligned} v_R &= \frac{v_L}{l-\frac{s}{2}}(l+\frac{s}{2}) \\ v_R(l-\frac{s}{2}) &= v_L(l+\frac{s}{2}) \\ v_R l - v_R\frac{s}{2} &= v_L l + v_L\frac{s}{2} \\ l(v_R-v_L) &= v_R\frac{s}{2} + v_L\frac{s}{2} \end{aligned} \longrightarrow l = \frac{s(v_R+v_L)}{2(v_R-v_L)}$$

$$\begin{aligned} v_L &= \omega(l-\frac{s}{2}) \\ v_L &= \frac{\omega s}{2}\left(\frac{v_R+v_L}{v_R-v_L}-1\right) \\ v_L &= \frac{\omega s}{2}\left(\frac{2v_L}{v_R-v_L}\right) \end{aligned} \longrightarrow \omega = \frac{v_R-v_L}{s} \longrightarrow \begin{aligned} \dot{x} &= v_B\cos(\theta) \\ \dot{y} &= v_B\sin(\theta) \\ \dot{\theta} &= \omega \end{aligned} \longrightarrow \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos(\theta) & 0\\ \sin(\theta) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v_B\\ \omega\end{bmatrix}$$

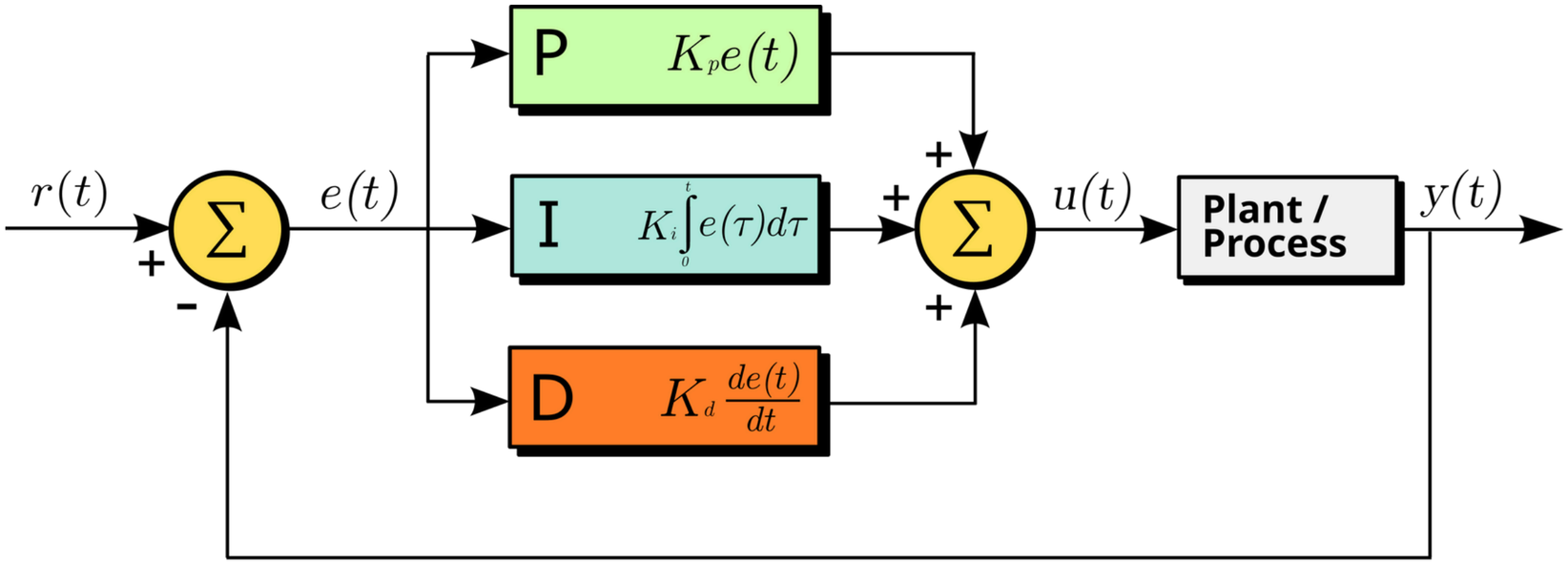
$$v_B = \omega\cdot l \longrightarrow \begin{aligned} v_B &= \frac{v_R-v_L}{s}\cdot\frac{s}{2}\frac{(v_R+v_L)}{(v_R-v_L)} \\ v_B &= \frac{v_R+v_L}{2} \end{aligned} \longrightarrow \begin{aligned} v_B &= \frac{v_R+v_L}{2} \\ \omega &= \frac{v_R-v_L}{s} \end{aligned} \longrightarrow \begin{bmatrix}v_B\\ \omega\end{bmatrix} = \begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\ -\frac{1}{s} & \frac{1}{s}\end{bmatrix}\begin{bmatrix}v_L\\ v_R\end{bmatrix}$$

$$\begin{aligned} v_L &= r\dot{\phi}_L \\ v_R &= r\dot{\phi}_R \end{aligned} \longrightarrow \begin{bmatrix}v_L\\ v_R\end{bmatrix} = \begin{bmatrix}r & 0\\ 0 & r\end{bmatrix}\begin{bmatrix}\dot{\phi}_L\\ \dot{\phi}_R\end{bmatrix} \longrightarrow \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\frac{r\cos(\theta)}{2} & \frac{r\cos(\theta)}{2}\\ \frac{r\sin(\theta)}{2} & \frac{r\sin(\theta)}{2}\\ -\frac{r}{s} & \frac{r}{s}\end{bmatrix}\begin{bmatrix}\dot{\phi}_L\\ \dot{\phi}_R\end{bmatrix}$$

Ref: https://aleksandarhaber.com/clear-and-detailed-explanation-of-kinematics-equations-and-geometry-of-motion-of-differential-wheeled-robot-differential-drive-robot/

PID Controller

The controller attempts to minimize the error over time by adjustment of a control variable u(t)



A block diagram of a PID controller in a feedback loop. r(t) is the desired process variable (PV) or setpoint (SP), and y(t) is the measured PV.

ArUco Marker Detection and Real Time Depth Estimation

The controller attempts to minimize the error over time by adjustment of a control variable u(t)

Radial distortion can be represented as follows:

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Similarly, tangential distortion occurs because the image-taking lense is not aligned perfectly parallel to the imaging plane. So, some areas in the image may look nearer than expected. The amount of tangential distortion can be represented as below:

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

In short, we need to find five parameters, known as distortion coefficients given by:

$$Distortion\ coefficients = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

In addition to this, we need to some other information, like the intrinsic and extrinsic parameters of the camera. Intrinsic parameters are specific to a camera. They include information like focal length (f_x, f_y) and optical centers (c_x, c_y). The focal length and optical centers can be used to create a camera matrix, which can be used to remove distortion due to the lenses of a specific camera. The camera matrix is unique to a specific camera, so once calculated, it can be reused on other images taken by the same camera. It is expressed as a 3x3 matrix:

$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Extrinsic parameters corresponds to rotation and translation vectors which translates a coordinates of a 3D point to a coordinate system.

For stereo applications, these distortions need to be corrected first. To find these parameters, we must provide some sample images of a well defined pattern (e.g. a chess board). We find some specific points of which we already know the relative positions (e.g. square corners in the chess board). We know the coordinates of these points in real world space and we know the coordinates in the image, so we can solve for the distortion coefficients. For better results, we need at least 10 test patterns.

As mentioned above, we need at least 10 test patterns for camera calibration. OpenCV comes with some images of a chess board (see samples/data/left01.jpg – left14.jpg), so we will utilize these. Consider an image of a chess board. The important input data needed for calibration of the camera is the set of 3D real world points and the corresponding 2D coordinates of these points in the image. 2D image points are OK which we can easily find from the image. (These image points are locations where two black squares touch each other in chess boards)

What about the 3D points from real world space? Those images are taken from a static camera and chess boards are placed at different locations and orientations. So we need to know (X, Y, Z) values. But for simplicity, we can say chess board was kept stationary at XY plane, (so Z=0 always) and camera was moved accordingly. This consideration helps us to find only X,Y values. Now for X,Y values, we can simply pass the points as (0,0), (1,0), (2,0), ... which denotes the location of points. In this case, the results we get will be in the scale of size of chess board square. But if we know the square size, (say 30 mm), we can pass the values as (0,0), (30,0), (60,0), ... Thus, we get the results in mm. (In this case, we don't know square size since we didn't take those images, so we pass in terms of square size).

3D points are called **object points** and 2D image points are called **image points.**

Raw Simulation Result

This is the result we got by implementing our own code. However, it needs alterations to get the desired result.