Report on Differential Drive Controllers

Iurii Podkorytov i.podkorytov@innopolis.university

February 17, 2025

All code is available on GitHub

1 Proportional Controller

The Proportional Controller is a simple feedback control mechanism that adjusts the robot's velocity based on the error between the desired position and the current position. In my implementation:

$$v = K_l \cdot \sqrt{(x_{\text{des}} - x)^2 + (y_{\text{des}} - y)^2}$$
$$\omega = K_a \cdot (\arctan 2(y_{\text{des}} - y, x_{\text{des}} - x) - \theta)$$

After calculating linear and angular velocities, they are adjusted to be less or equal to the maximum allowed velocities of the robot.

This controller can be tuned by changing linear and angular gains, I have chosen $K_l = 0.7$ and $K_a = 2$.

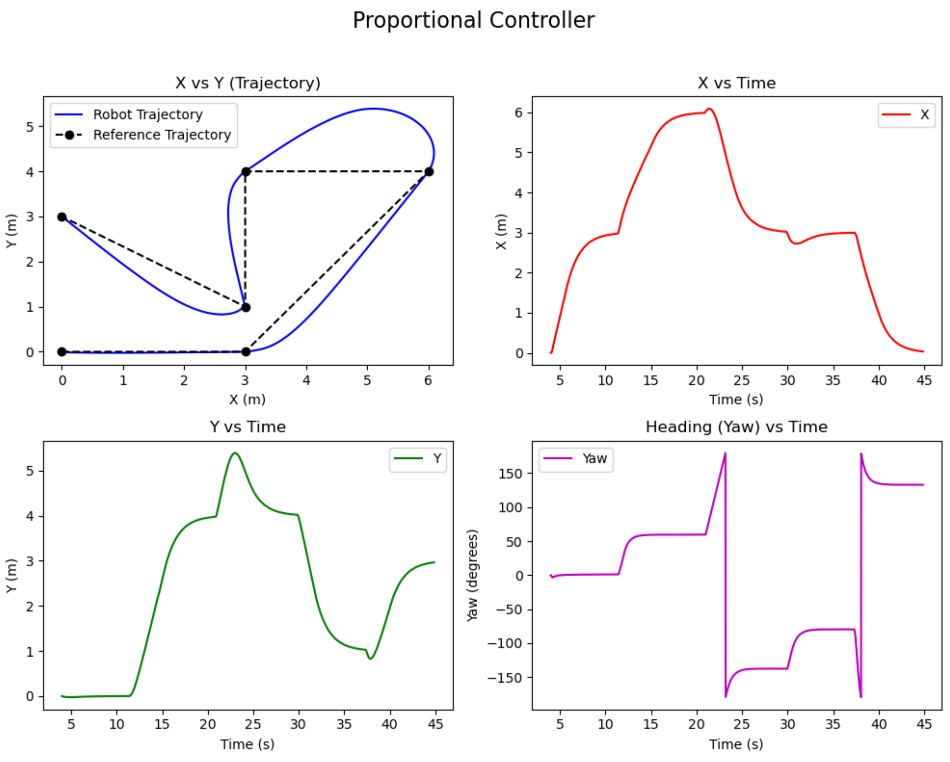

Figure 1: Performance of the Proportional Controller

It is evident that this controller does not handle tight turns well and can deviate quite a bit from the desired path. Also, with my implementation, it comes to a stop at every intermediate point. This can be improved by smoothing the velocity profile.

2 Pure Pursuit Controller

The Pure Pursuit Controller tracks the look-ahead point on the path.

To ensure that the controller does not skip any intermediate point, only the current segment is considered until its end is inside the circle with radius equal to the look-ahead distance.

For this controller, the linear velocity is constant until the last segment of the path, where it is proportional to the remaining distance to the goal. The angular velocity is proportional to the difference between the robot heading and the direction to the target point.

I have used look-ahead distance $R = 1$ and $K_a = 2$.

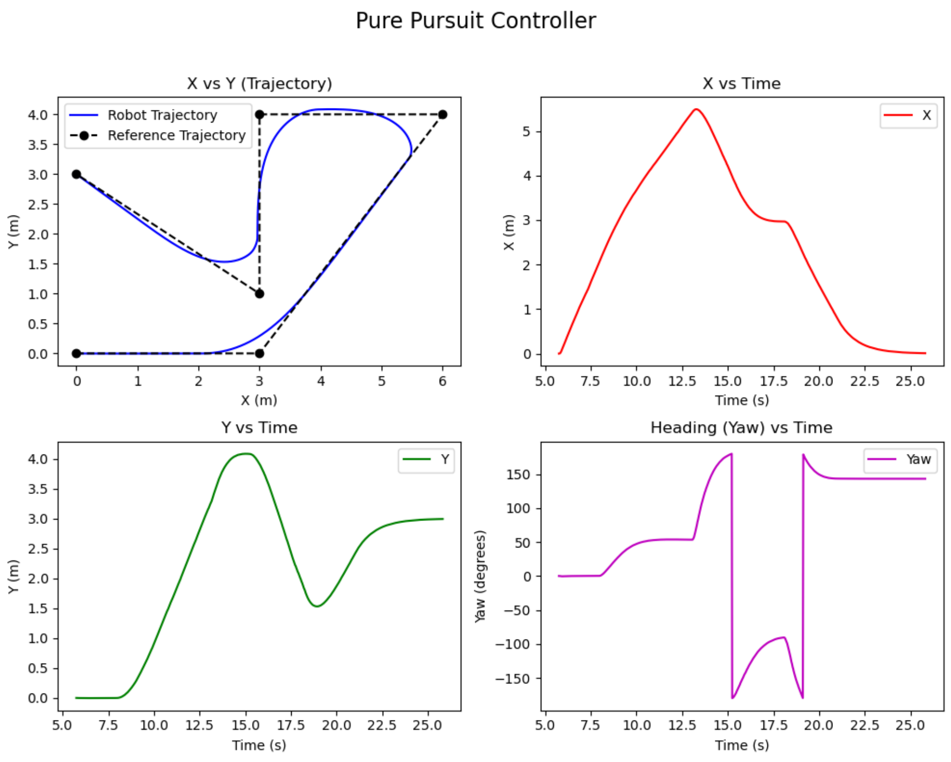

Figure 2: Performance of the Pure Pursuit Controller

This controller performs much better than the previous one and completes the path faster.

3 Stanley Controller

The Stanley Controller combines lateral and heading errors to compute the angular velocity command.

Initially, the linear velocity was set to constant, but it led to a robot overshooting a point by too much and eventually losing the path. To fix this, I made it slow down before a corner in the path.

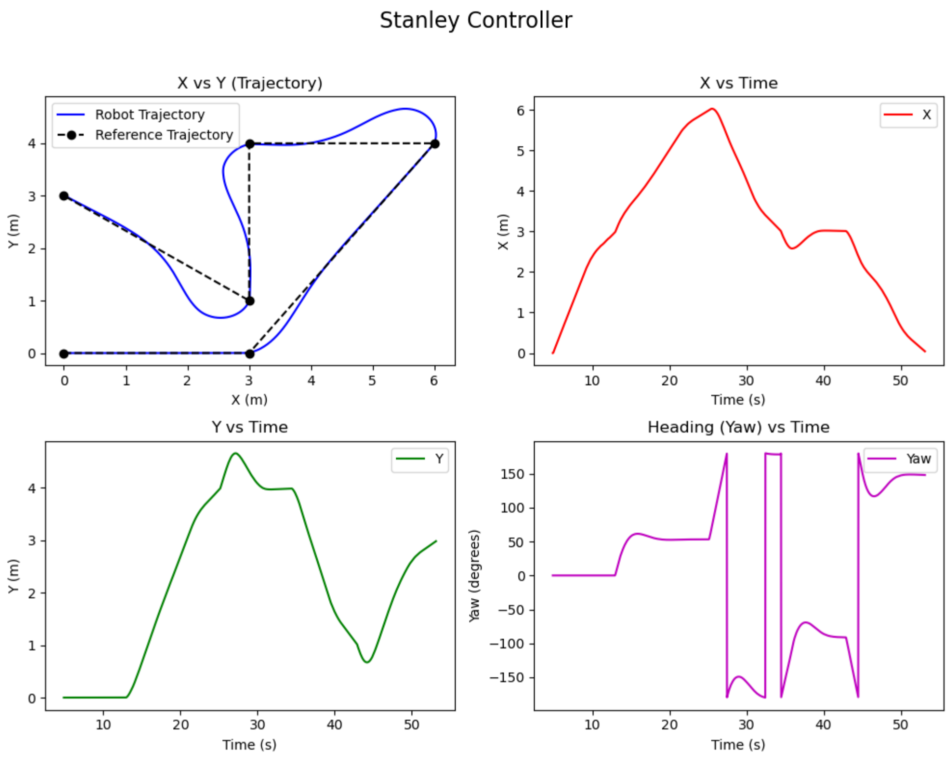

Figure 3: Performance of the Stanley Controller

4 Model Predictive Controller (MPC)

The Model Predictive Controller uses a predictive model to optimize future trajectories by minimizing a cost function over a prediction horizon.

Unfortunately, I was unable to tune the parameters of the cost function so that the robot does not oscillate around the path. However, it is interesting that this is the only controller that allows the robot to drive backwards, in order to make smaller turns.

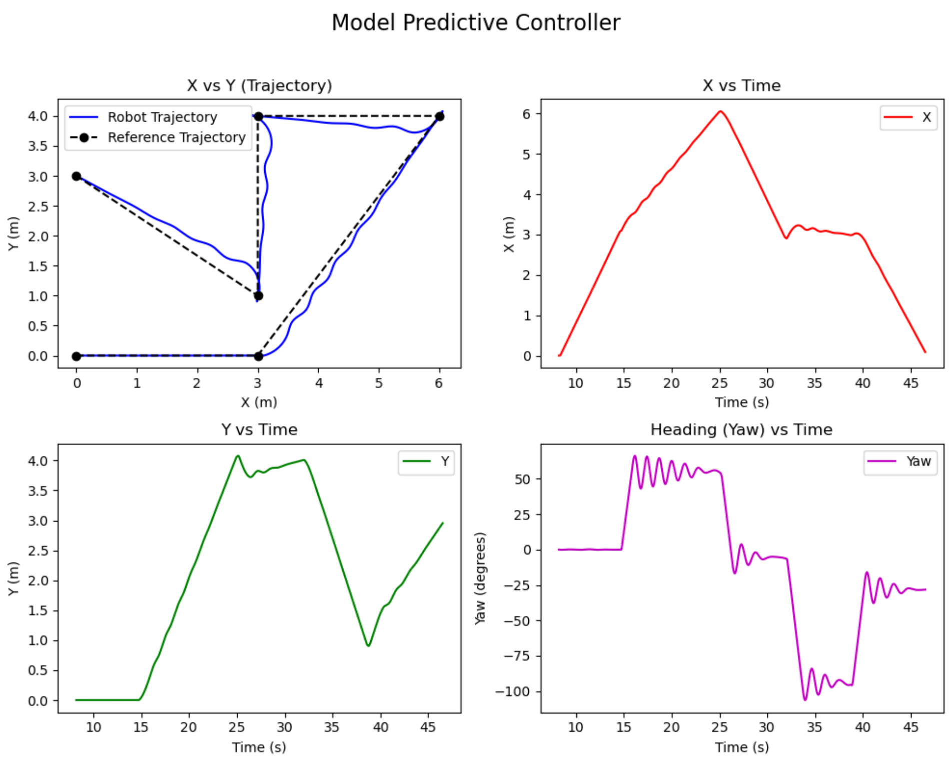

Figure 4: Performance of the Model Predictive Controller