Homework 3 (Tasks 1-19) in EL2450 Hybrid and Embedded Control Systems

Ali Ghasemi	Pierre Haensch	Kinane Obeidine	Angelo Calo'
980422-8352			20030129-T217
aghasemi@kth.se	haensch@kth.se	obeidine@kth.se	acalo@kth.se

Giacomo Vedovato
0209269331
vedovato@kth.se

February 24, 2025

Task 1

From equations 1 and 2:

$$v = \frac{u_r + u_l}{2} \tag{1}$$

$$\omega = u_r - u_l \tag{2}$$

We can solve for u_r and u_l in terms of sum and differences of v and ω:

$$u_r = v + \frac{\omega}{2} \tag{3}$$

$$u_l = v - \frac{\omega}{2} \tag{4}$$

These equations allow the robot's control system to compute the required wheel velocities given the desired translation velocity v and angular velocity ω.

Task 2

a

Proposition	**True in Regions**
Red (R)	R1, R13, R10, R20, R26, R31
Blue (B)	R7, R21, R29, R31
Green (G)	R2, R6, R9, R25, R31
Obstacle (O)	R3, R10, R11, R19, R22, R26, R29 , R31

Table 1: Atomic Propositions for Regions in the Workspace

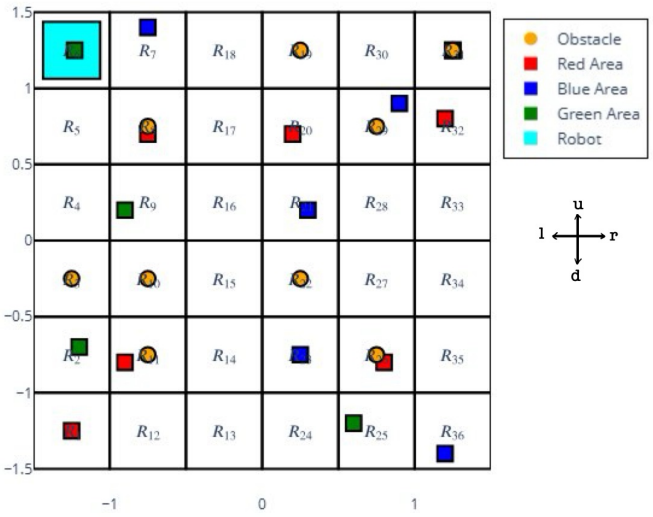


Figure 1: Discretized Workspace

b

The size of one region is (in meters):

$$(0.5, 0.5)$$

c

A finer grid offers a more accurate representation of the environment. This level of detail is especially beneficial when the available space between obstacles is limited (even if, in our case, the dimensions of the regions are constrained by the robot's size), and it allows for a more precise assignment of properties such as "red," "blue," or "green" to specific regions. However, finer discretization increases the number of states that the planning algorithm must manage, resulting in a higher computational load and longer processing times. Furthermore, since our robot's movement is restricted to specific directions (i.e., only orthogonal moves with 90° turns), the advantages of a finer grid may be reduced by the additional number of turns required, possibly leading to a path that appears shorter on paper but requires more time in practice. Another issue with using a finer grid (with cells sized similarly to the robot) arises in region R_{29}, where the obstacle and the "blue" property are divided between different regions. In this case, the obstacle falls exactly at the intersection of four regions, so affecting all four cells. On the other hand, a less fine discretization simplifies the problem by reducing the number of states, which makes planning faster and easier to compute. However, this simplicity comes at the cost of diminished environmental detail, with the additional adverse possibility of merging areas with different properties into a single region (as already happen in region R_{11}).

d

$$S = \{R_1, R_2, \ldots, R_{36}\} \tag{5}$$

$$S_0 = \{R_6\} \tag{6}$$

$$\Sigma = \{r, l, u, d\} \tag{7}$$

$$\mathcal{N}_i = \left\{ R_j \in S \,\middle|\, (c_{x,j}, c_{y,j}) = (c_{x,i} \pm d_x, c_{y,i}) \text{ or } (c_{x,j}, c_{y,j}) = (c_{x,i}, c_{y,i} \pm d_y) \right\} \tag{8}$$

$$\rightarrow = \{(R_i, \sigma, R_j) \in S \times \Sigma \times S \mid R_j \in \mathcal{N}_i\} \tag{9}$$

$$AP = \{\text{red, blue, green, obstacle}\} \tag{10}$$

$$L(R_i) = \{p \in AP \mid R_i \text{ contains a feature labeled } p\} \tag{11}$$

$$L(R_1, R_{20}, R_{32}) = \{\text{red}\}$$
$$L(R_7, R_{21}, R_{23}, R_{36}) = \{\text{blue}\}$$
$$L(R_2, R_6, R_9, R_{25}, R_{31}) = \{\text{green}\}$$
$$L(R_3, R_{10}, R_{19}, R_{22}, R_{31}) = \{\text{obstacle}\}$$
$$L(R_{29}) = \{\text{blue, obstacle}\}$$
$$L(R_8, R_{11}, R_{26}) = \{\text{red, obstacle}\}$$
$$L(R_4, R_5, R_{13}, R_{14}, R_{15}, R_{16}, R_{17}, R_{18}, R_{27}, R_{28}, R_{30}, R_{33}, R_{34}, R_{35}) = \emptyset$$

In the definition of Σ, r stands for a movement to the right ($+d_x$), l to the left ($-d_x$), u upwards (d_y) and d downwards ($-d_y$) with respect to the fixed reference frame in Figure 1.

Task 3

A path that satisfies the required behavior is defined by the following sequence, where r means "right" and d means "down":

$$rrdr(dr)^*$$

Task 4

The hybrid control strategy enables the robot to move directly from the starting region to the goal region without traversing any intermediate areas that may contain obstacles. First, a rotation controller effectively aligns the robot with a straight-line path to the goal. Once correctly oriented, a line-following controller propels the robot forward along this path while maintaining precise alignment. By distinctly separating the rotation and translation processes, the robot confidently avoids unnecessary deviations that could lead it into unintended areas.

Simultaneous control of both orientation and distance can result in the robot taking curved or indirect paths, which heightens the risk of entering adjacent regions. Given that the workspace is organized into predefined regions, as demonstrated in Figure 5, maintaining a direct transition is essential. The hybrid strategy capitalizes on the size of these regions to guide the robot along a controlled trajectory, ensuring it remains on its intended path. This method guarantees structured and predictable motion, positioning it as a highly effective approach for safely navigating from one region to another.

Task 5

For the system to be asymptotically stable, we need to make sure the error term ($e[k]$) converges to zero over time. We can define the error as (12):

$$e[k] = \theta[k] - \theta^R \tag{12}$$

Furthermore, by combining and rearranging the discrete-time update equation (13) and proportional discrete-time controller (14) we can achieve equation (15)

$$\theta[k+1] = \theta[k] + h\omega[k] \tag{13}$$

$$\omega[k] = K_{\Psi,1}(\theta^R - \theta[k]) \tag{14}$$

$$\theta[k+1] - \theta^R = (1 - hK_{\Psi,1})(\theta[k] - \theta^R) \tag{15}$$

Using (15) we can define error dynamics as (16):

$$e[k+1] = (1 - hK_{\Psi,1})e[k] \tag{16}$$

For $\theta[k]$ to asymptotically converge to θ^R, we require:

$$|1 - hK_{\Psi,1}| < 1 \tag{17}$$

Solving this inequality for $K_{\Psi,1}$ will result in:

$$0 < K_{\Psi,1} < \frac{2}{h} \tag{18}$$

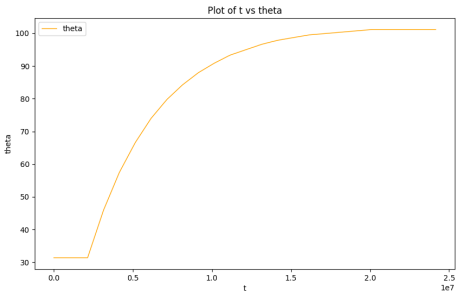
The proportional gain $K_{\Psi,1}$ determines the ratio of output response to the error signal. In general, increasing the proportional gain will increase the speed of the control system response. However, if the proportional gain is too large, the process variable will begin to oscillate. If $K_{\Psi,1}$ is increased further, the oscillations will become larger and the system will become unstable and may even oscillate out of control. Based on this fact we choose:

$$K_{\Psi,1} = \frac{1}{h} \tag{19}$$

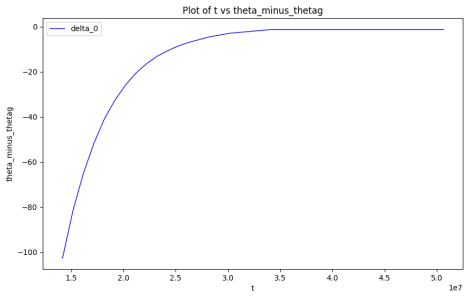
This ensures a reasonable rate of convergence while staying within the stability limits.

Task 6

By setting $v = 0$ and implementing the controller described in Task 5, with the gain $K_{\Psi,1}$ chosen according to equation (36), it is evident that the controller asymptotically stabilizes Θ to values really close to Θ^R. Figure 2a illustrates the temporal evolution of Θ, starting from an initial value of 0° at the origin and converging near the target direction at point 1 (i.e., $\Theta_g = 102.42^\circ$). The final value of convergence of Θ is 101.15°. Figure 2b shows the trend of $\Theta - \Theta_g$ with respect to time.



(a) Θ [◦] with respect to time [s]



(b) $\Theta - \Theta_g$ [◦] with respect to time [s]

Figure 2: Simulation results for task 6

Task 7

Using kinematics, we can define the position update equation in discrete time as(20) :

$$\begin{bmatrix} x[k+1] \\ y[k+1] \end{bmatrix} = \begin{bmatrix} x[k] \\ y[k] \end{bmatrix} + hv[k]v_c[k] \tag{20}$$

where $v[k]$ is the translational velocity at step k and $v_c[k]$ is the unit vector in the direction of motion.

Now using this equation we can define the discrete-time evolution of $\Delta_0[k]$ as:

$$\Delta_0[k+1] = \Delta_0[k] - hv[k]v_c[k] \tag{21}$$

Taking the inner product with $v_c[k]$ and substituting $v[k] = K_{\omega,1}d_0[k]$:

$$d_0[k+1] = (1 - hK_{\omega,1})d_0[k] \tag{22}$$

For $d_0[k]$ to converge to 0 asymptotically we should have:

$$|1 - hK_{\omega,1}| < 1 \tag{23}$$

Solving this inequality for $K_{\omega,1}$ will result in:

$$0 < K_{\omega,1} < \frac{2}{h} \tag{24}$$

A reasonable choice to balance stability and convergence speed is:

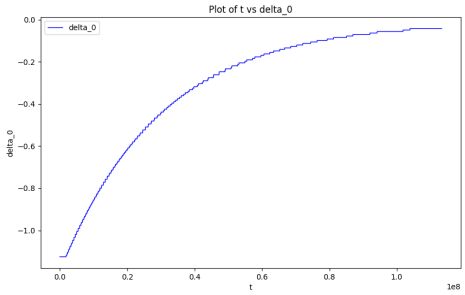
$$K_{\omega,1} = \frac{1}{h} \tag{25}$$

As explained in task 6, this ensures the robot remains close to $[x_0, y_0]$ without oscillating or moving away during rotation.

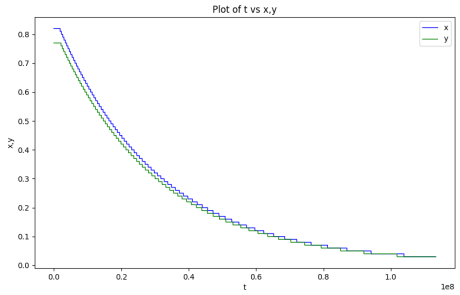
Task 8

Setting $w = 0$ and implementing the controller of Task 7, with $K_{w,1}$ as indicated in equation (32), it is possible to see that the controller is able to stabilize asymptotically $d0$ to 0. Knowing that the robot cannot rotate, but only move along its initial defined Θ, the moment in which it is possible to ensure that $[x[k], y[k]]^T$ stays exactly at $[x0, y0]^T$ is only when the two points are aligned with the direction given by Θ. In other cases, this is not possible. Cases that warrant further consideration are when $\Theta = 0$ and $\Theta = 90°$. If $\Theta = 0$, then the robot will be able

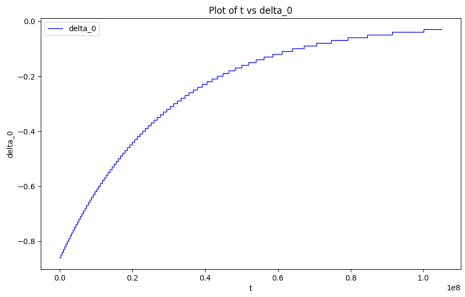
to align its x coordinate with the one of the initial point. This is explainable due to the fact that the second term of v_c, and consequently the second term of Δ_0, will be null. Similarly if $\Theta = 90°$, the coordinates aligned will be the y-coordinates, due to the zeroing out of the first term of v_c, and thus the first term of Δ_0. The simulation results are presented in the figures below. In Figures 3a and 3b is shown the case where the two points are aligned with θ, in particular, respectively the stabilizing of d_0 and the convergence of the robot position to the initial one. Figures 9d and 3c represent the same information in the scenario where $\Theta = 0$.



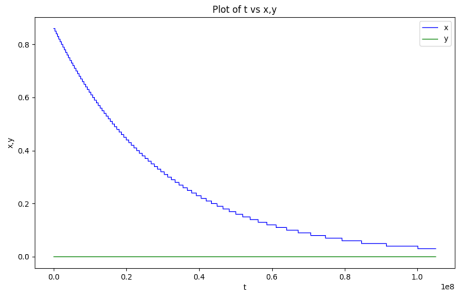
(a) d_0 [m] with respect to time [s] - case points aligned with Θ



(b) Position [m] with respect to time [s] - case points aligned with Θ



(c) d_0 [m] with respect to time [s] - case $\Theta = 0$

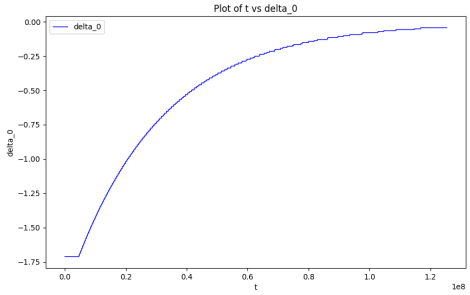


(d) position [m] with respect to time [s] - case $\Theta = 0$

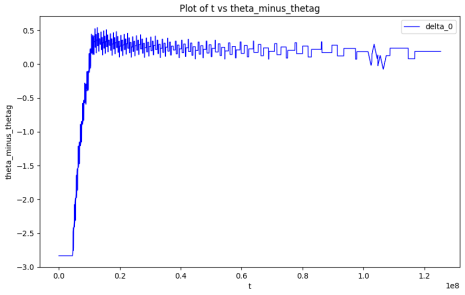
Figure 3: Simulation results for task 8

Task 9

By activating both controllers simultaneously and evaluating the simulation performance, it can be confirmed that the control system effectively aligns the robot with Θ_g while keeping it stable at the starting position. The key difference between the single use of the controllers and the simultaneous one is that in the latter the two controllers interact with each other as the whole system drives the robot back toward its start position. This coupling can induce transient fluctuations or a slower convergence for the Θ controller, since the orientation corrections are affected by the robot's ongoing translation. Similarly, even the distance error is affected by changes in direction of the translational movement perpetrated by the orientation controller. In Figures 4a and 4b are shown respectively the curves of the position error and the one of the angular error with respect to time. The errors converge at the following values: $d_0 = 0.042m$ and $\Theta - \Theta_g = 0.188°$.



(a) position error [m] with respect to time [ms]



(b) orientation error [◦] with respect to time [ms]

Figure 4: Simulation results for task 9

Task 10

we can define the position update equation in discrete time as(26):

$$\begin{bmatrix} x[k+1] \\ y[k+1] \end{bmatrix} = \begin{bmatrix} x[k] \\ y[k] \end{bmatrix} + hv[k]v_g \tag{26}$$

where $v[k]$ is the velocity control input and v_g is the unit vector toward the goal. Subtracting this from (x_g, y_g):

$$\Delta_g[k+1] = \begin{bmatrix} x_g \\ y_g \end{bmatrix} - \left(\begin{bmatrix} x[k] \\ y[k] \end{bmatrix} + hv[k]v_g \right) = \Delta_g[k] - hv[k]v_g \tag{27}$$

Now, if we take the inner product of both sides with v_g^T, we can achieve the dynamics of $d_g[k]$:

$$d_g[k+1] = v_g^T(\Delta_g[k] - hv[k]v_g) \tag{28}$$

After simplification and using the controller equation ($v[k] = K_{\omega,2}d_g[k]$):

$$d_g[k+1] = (1 - hK_{\omega,2})d_g[k] \tag{29}$$

From (29) it can be seen that for $d_g[k]$ to asymptotically converge to 0, we need:

$$|1 - hK_{\omega,2}| < 1 \tag{30}$$

Solving this inequality for $K_{\omega,1}$ will result in:

$$0 < K_{\omega,2} < \frac{2}{h} \tag{31}$$

As explained in previous tasks, a reasonable choice to balance stability and convergence speed is:

$$K_{\omega,2} = \frac{1}{h} \tag{32}$$

Task 11

When only using v as a control, the robot moves to its goal in an almost straight line. The simulation results are presented in figure 5, for three different initial angles. The position of the robot asymptotically converges to the goal position, with an accuracy depending on the margin chosen in the code. For example, in graph (h), position x converges to 1 cm because it is within the chosen margin, which is 1 cm.

Task 12

The orientation $\theta[k]$ updates as:

$$\theta[k+1] = \theta[k] + h\omega[k] \tag{33}$$

Substituting the control law $\omega[k] = K_{\Psi,2} d_p[k]$ and using the approximation of $d_p[k]$:

$$\theta[k+1] = \theta[k] + hK_{\Psi,2} p(\theta_g - \theta[k]) \tag{34}$$

Rearranging and multiplying both sides by p:

$$d_p[k+1] = (1 - hK_{\Psi,2} p) d_p[k] \tag{35}$$

From (35), for $d_p[k]$ to converge to 0:

$$|1 - hK_{\Psi,2} p| < 1 \tag{36}$$

Solving this inequality for $K_{\Psi,2}$:

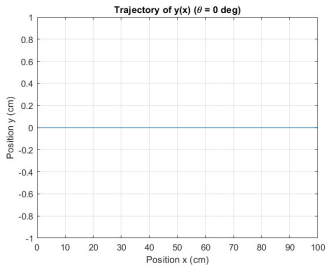
$$0 < K_{\Psi,2} < \frac{2}{hp} \tag{37}$$

As explained in previous tasks, a reasonable choice that ensures the robot follows the straight-line path efficiently without instability is:

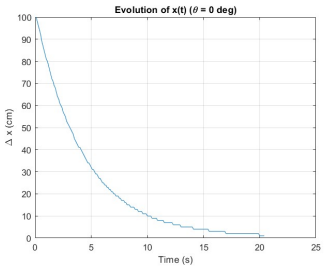
$$K_{\Psi,2} = \frac{1}{hp} \tag{38}$$

Task 13

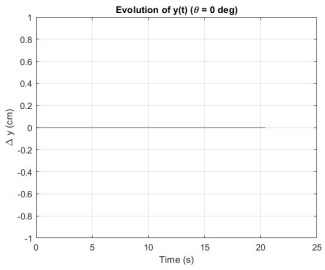
Figure 6 shows the evolution of d_p for some values of p and $K_{\Psi,2}$. One can observe that d_p converges asymptotically towards a final value, though there is a steady state error since d_p is supposed to converge to 0 when the system stabilizes. This steady state error is higher and the settling time is longer when $K_{\Psi,2}$ is kept constant and p decreased.



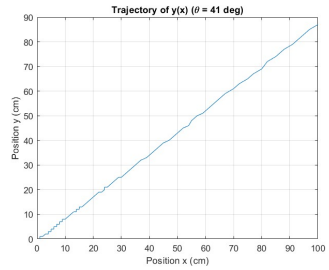
(a) Trajectory of the robot with go-to-goal controller when $\omega = 0$, initial angle $\theta_0 = 0°$



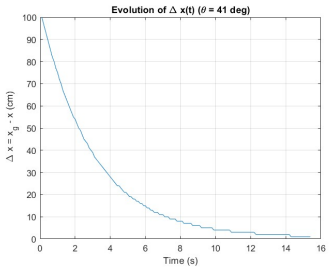
(b) Position x with respect to time with go-to-goal controller when $\omega = 0$, initial angle $\theta_0 = 0°$



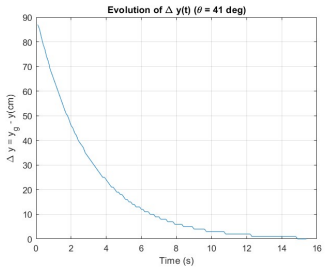
(c) Position y with respect to time with go-to-goal controller when $\omega = 0$, initial angle $\theta_0 = 0°$



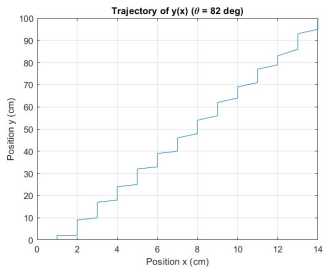
(d) Trajectory of the robot with go-to-goal controller when $\omega = 0$, initial angle $\theta_0 = 41°$



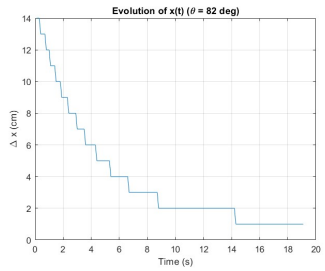
(e) Position x with respect to time with go-to-goal controller when $\omega = 0$, initial angle $\theta_0 = 41°$



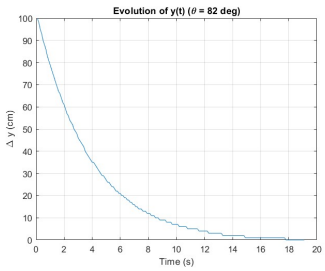
(f) Position y with respect to time with go-to-goal controller when $\omega = 0$, initial angle $\theta_0 = 41°$



(g) Trajectory of the robot with go-to-goal controller when $\omega = 0$, initial angle $\theta_0 = 82°$

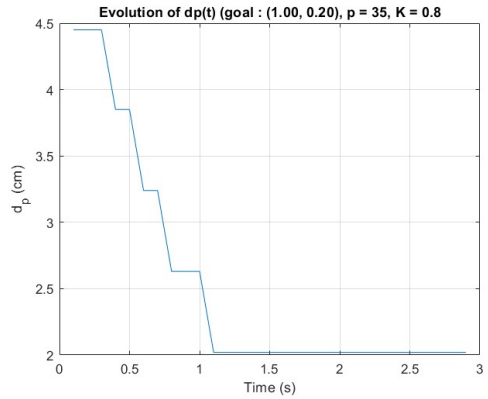


(h) Position x with respect to time with go-to-goal controller when $\omega = 0$, initial angle $\theta_0 = 82°$

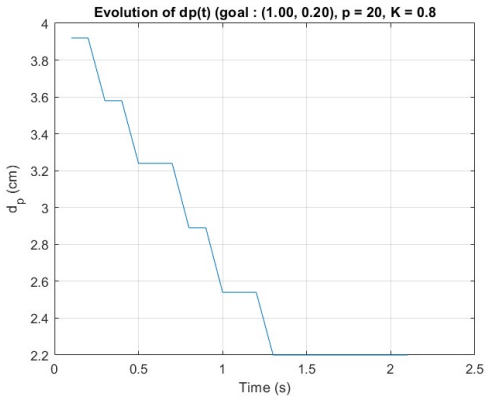


(i) Position y with respect to time with go-to-goal controller when $\omega = 0$, initial angle $\theta_0 = 82°$

Figure 5: Simulation results of go-to-goal controller with $\omega = 0$



(a) Evolution of d_p for $p = 35$cm and $K_{\Psi,2} = 0.8s^{-1}cm^{-1}$



(b) Evolution of d_p for $p = 20$cm and $K_{\Psi,2} = 0.8s^{-1}cm^{-1}$

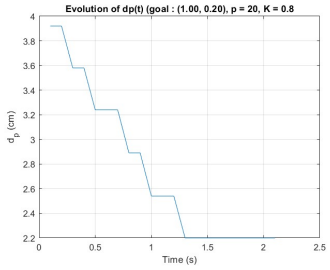
Figure 6: Evolution of d_p and d_g for several p and $K_{\Psi,2}$

Task 14

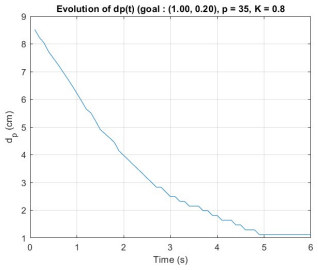
The closed-loop system is given by

$$\dot{\theta} = \frac{R}{L}\omega = \frac{R}{L}pK_{\Psi,2}(\theta_g - \theta)$$

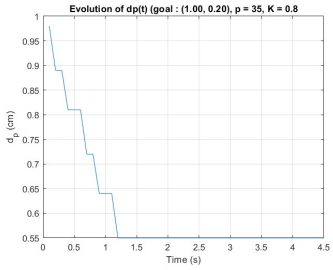
So the transfer function is $\frac{\theta}{\theta_g} = \frac{1}{1+\tau s}$ where $\tau = \frac{L}{pRK_{\Psi,2}}$.
Thus, in order to keep a constant pole of the closed loop, $p \times K_{\Psi,2}$ needs to remain constant. The simulation results for several values of p and $K_{\Psi,2}$, while keeping constant the product of these values, is presented in figure 7.



(a) $p = 20$ and $K_{\Psi,2} = 0.8$



(b) $p = 10$ and $K_{\Psi,2} = 1.6$



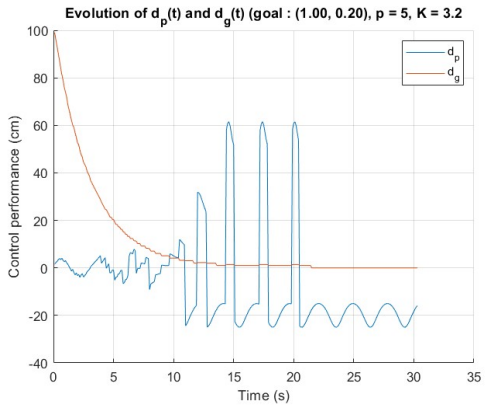
(c) $p = 5$ and $K_{\Psi,2} = 3.2$

Figure 7: Evolution of d_p for several p and $K_{\Psi,2}$

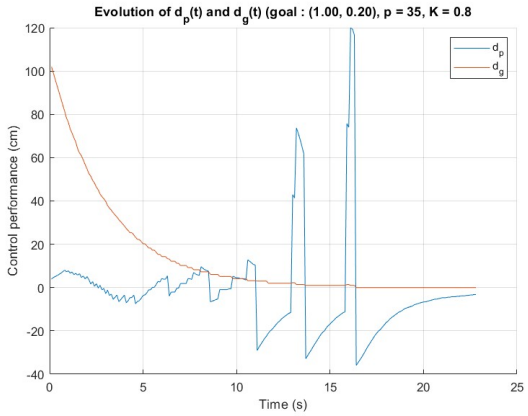
Notice that in all cases, d_p converges asymptotically towards a final constant value with a steady state error. One can observe that when keeping the product constant, decreasing p and increasing $K_{\Psi,2}$ decreases the steady state error.

To understand what p represents, imagine a robot that is out of the line connecting (x_0, y_0) to (x_g, y_g). This could happen for example if there is an obstacle, or if the robot had deviated from its trajectory for any possible reason. Then, to get back to the line, the idea is to aim a point P located on the line and at a distance p from the robot. Therefore, designing p is choosing the angle that the robot should aim in order to get back to its path. The smallest p is, the higher the angle, the quicker the robot gets back to the path and the higher the oscillations around the path are.

Task 15



(a) Both controllers enabled, $p = 20$ and $K_{\Psi,2} = 0.8$



(b) Both controllers enabled, $p = 5$ and $K_{\Psi,2} = 32$

Figure 8: Evolution of d_p and d_g for several p and $K_{\Psi,2}$

As seen in figures 8, d_g stabilizes asymptotically around 0 in both cases, which means that the robot gets to the desired point (x_g, y_g). However d_p does not converge and oscillates around -20 as observed in figure (a). When both controllers are enabled, d_p does not have the same behavior. It makes large oscillations compared to when it is alone. These oscillations increase in amplitude as the robot gets closer to its goal position, because more errors about the angle appear as the distance from the robot to the goal decreases.

Task 16

The robot can be modeled as a hybrid automaton

$$H = \big(Q,\, X,\, \text{Init},\, f,\, D,\, E,\, G,\, R\big)$$

such that:

- Q : set of discrete states,
- X : continuous state space,
- Init $\subseteq Q \times X$: initial states,
- f : continuous dynamics,
- D : domain,
- $E \subseteq Q \times Q$: discrete transitions,
- G : guard conditions,
- R : reset map.

In our situation,

$$Q = \{\, q_0,\, q_1,\, q_2\},$$

where:

q_0 is the *idle* state, q_1 is the *rotation* state, q_2 is the *translation* state.

The continuous state space is:

$$X = \{(x, y, \theta) \mid x \in [-1.5, 1.5],\ y \in [-1.5, 1.5],\ \theta \in [-\pi, \pi]\},$$

where all variables x, y, θ are real-valued.
For each $q_i \in Q$, let $D(q_i) \subseteq X$ be its domain (or region):

$$D(q_0) = \big\{(x, y, \theta) \in X \;\big|\; |x - x_g| + |y - y_g| < 69\big\},$$
$$D(q_1) = \big\{(x, y, \theta) \in X \;\big|\; |x - x_0| + |y - y_0| < 67,\ \ |\theta - \theta_g| > 68\big\},$$
$$D(q_2) = \big\{(x, y, \theta) \in X \;\big|\; |x - x_g| + |y - y_g| > 69\big\}.$$

Indicating initial position and orientation of robot when control phase starts as:

$$\text{Init} = (x_0,\ y_0,\ \theta_0)$$

Finally defining discrete transitions, guards and reset map as follow.

$$E = \{\,(q_0, q_1),\ (q_1, q_2),\ (q_2, q_0)\}.$$

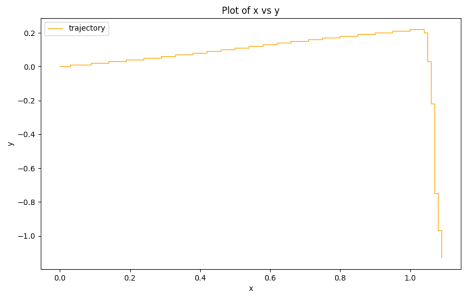
$$G(q_1, q_2) = \big\{\ |\theta - \theta_g| < 68\big\},$$
$$G(q_2, q_0) = \big\{\ |x - x_g| + |y - y_g| < 69\big\},$$
$$G(q_0, q_1) = \text{input} \Rightarrow \text{Set Reference}.$$

R : variables from state to state are kept continuous.

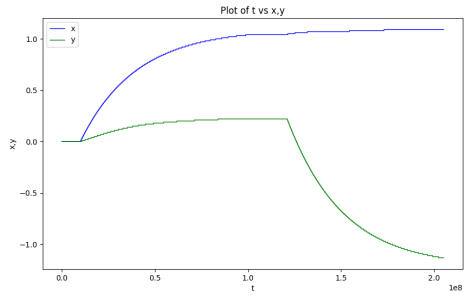
Task 17

The simulation results show that the robot transitions between discrete states in a predictable way, starting in the initial state, then rotating (q1) until it is aligned with the goal, and finally switching to the translation state (q2) to move forward. The plots of x and y over time confirm that the robot's position steadily approaches the goal location, while the plot of theta shows how the robot's orientation changes from its initial angle to align with the target direction. As time progresses, the robot settles into the final region without oscillations, demonstrating that the hybrid controller effectively coordinates the rotation and line-following modes.

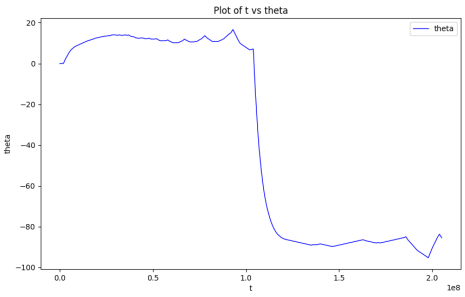
Comparing the discrete state trajectories with the continuous position and orientation data suggests that the system behaves as expected: it first corrects its heading (visible in the theta plot), then advances along a roughly straight path toward the goal (as seen in the x and y plots), and finally stops once it arrives. The fact that each region of motion (q0, q1, q2) appears at the correct times in the state plot indicates that the controller switches modes precisely when needed, confirming the overall stability and correctness of the hybrid approach.



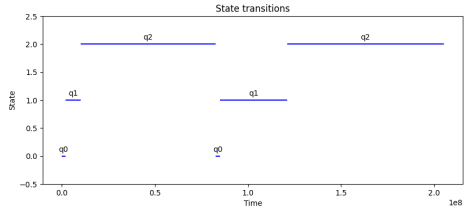
(a) Trajectory of robot



(b) Continues state translations



(c) Continues state Rotation



(d) Discrete state transitions

Figure 9

Task 18

After executing the plan derived in Task 3, the robot successfully traverses each waypoint along the centers of the specified regions. The trajectory plot shows that the robot moves in distinct straight segments between waypoints, turning at each one before continuing to the next. This stepwise path demonstrates that the orientation controller aligns the robot's heading to each new segment, and the line-following controller then drives the robot forward until it reaches the subsequent region center. The final path remains within the intended corridor for each leg of the journey, indicating that the robot follows the plan accurately and avoids neighboring regions. The observed performance is stable, with no significant oscillations, and the robot arrives at the final waypoint as expected.

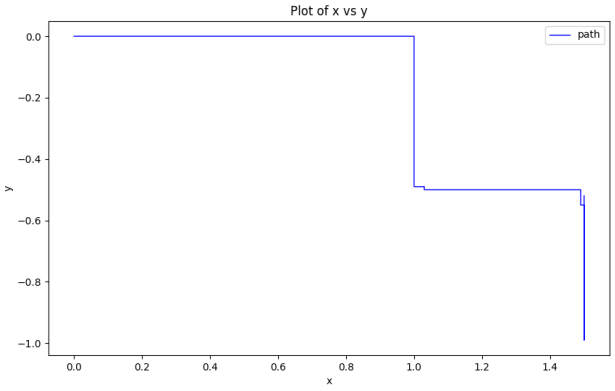


Figure 10: Trajectory

Task 19

The continuous controllers must be tuned so that the robot follows a direct path between two adjacent regions without overshooting or drifting into neighboring ones. If the orientation or line-following gains are too large or too small, the robot can oscillate or deviate significantly, thereby entering unintended regions and breaking the safety requirement (even colliding with obstacles). Stable, predictable control laws ensure the robot's motion remains confined to the corridor that directly connects the start and end regions during each transitions, thereby preserving the property that it stays within those two regions until the transition is complete. For example, If the robot must move from region A to region B along a narrow corridor, an excessively large orientation gain could cause it to overshoot and briefly enter a neighboring region, while a very small gain might not correct drift in time. By choosing balanced gains, the robot stays centered on the path and smoothly completes each transition without straying into unintended areas.