[Robotique mobile]Unicycle and bicycle robot control - Tianchi YU

yu-tianchi

December 2020

1 Unicycle control

1.1 Question 1 - Unicycle control

Here we implement two Proportional controllers, a first one to bring the robot to the goal point and a second one to adjust the orientation at the goal.

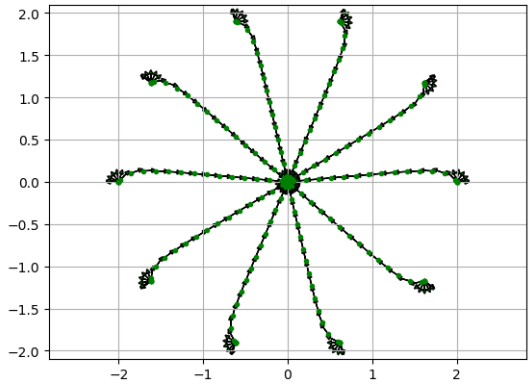


Figure 1: Unicycle To Pose

The *unicycle_to_pose.py* script test my method from several starting positions on a circle to reach the circle center. By adjusting the controller gains and the α_{max} parameter to quickly reach the goal by limiting oscillations, the mean goal reaching steps is 3498.5.

We take $K_\rho = 20$, $K_\alpha = 6$, $K_\beta = 20$ and $\alpha_{max} = \pi/3$.

2 Control of a bicycle

2.1 Question 2 - Control of a bicycle towards a point

Here we write a Proportional controller to guide the robot towards a point.

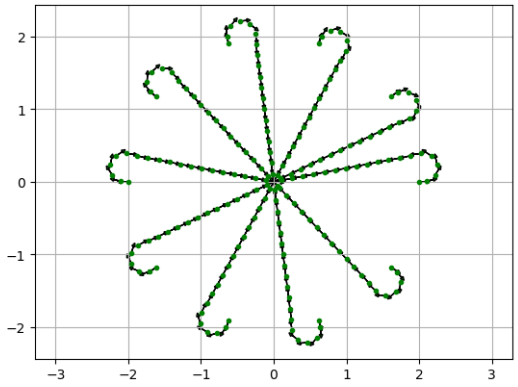


Figure 2: Bicycle To Point

The *bicycle_to_point.py* script test my method from several starting positions on a circle to reach the circle center. By adjusting the controller gains and the α_{max} parameter to quickly reach the goal by limiting oscillations, the mean goal reaching steps is 2769.0.

We take $K_\rho = 10$, $K_\alpha = 6$.

2.2 Question 3 - Control of a bicycle towards a pose

We build a proportional controller that guides the robot to a position with a fixed final orientation.

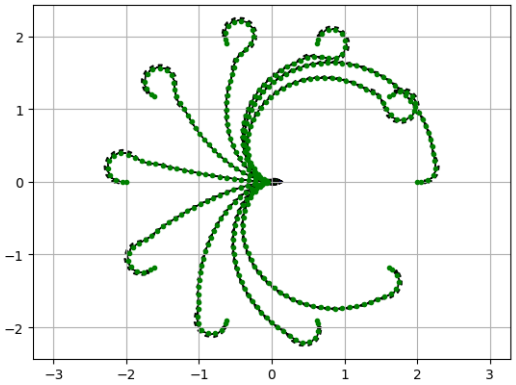


Figure 3: Bicycle To Pose

The *bicycle_to_point.py* script test my method from several starting positions on a circle to reach the circle center. By adjusting the controller gains and the α_{max} parameter to quickly reach the goal by limiting oscillations, the mean goal reaching steps is 3464.7.

We take $K_\rho = 18$, $K_\alpha = 15$, $K_\beta = -8$.

When the K_beta is too closed to 0, the robot wouldn't arrive the final orientation in some of the starting positions.

2.3 Question 4 - Control of a bicycle following a path

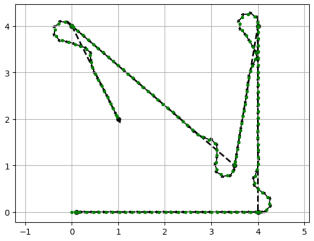
In this part, a controller that guides the robot to follow a path defined by a set of waypoints (Figure 4) is given, using the simple Pure Pursuit approach.

We set the lookahead distance as a fixed value(0.02 at the first example), at each timestep, a point on the trajectory is given at this distance from the robot, and use the proportional controller from question 2 to compute the speed that will guide the robot toward this point.

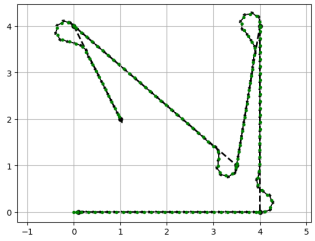
Once the robot's distance to that point is less than the fixed value, the point will move towards the next waypoint with a small step(setted as 0.5 in the first example). Until the point reaches the waypoint, the following waypoint will should be used.

The bicycle_to_path.py script test this method on a fixed path. To optimize the performance, we change the parameters (lookahead and controller gains) to follow the path as closely as possible. Actually, the more interesting thing is the value of the lookahead distance. Without changing controller gains, we take two other distances (), here is the comparison between these results.

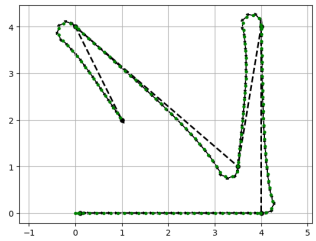
As we can see, the larger the forward-looking distance(lookahead distance) means the smoother the tracking of the trajectory, and the smaller the forward-looking distance will make the tracking more accurate (of course it will also bring more oscillation).



(a) pic1.lookahead 0.05, fixed step 0.5



(b) pic2.lookahead 0.2, fixed step 0.5



(c) pic3.lookahead 1, fixed step 0.5

Figure 4: Bicycle To Path with different lookahead distance