Waypoint Follower Instruction

File info:

1. “waypointfollower.cpp”:
    a. ROS node setup
    b. Joystick control setup: the same with before
2. “waypointfollower.h”:
    a. Functions and variables for “waypointfollower.cpp”
3. “Controller.cpp”:
    a. PID definition
    b. RTR function
4. “Controller.h”:
    a. Functions and variables for “Controller.cpp”

How to run:

rosrun waypointfollower waypointfollower

The output is the same as before (topics that send controlling value to motors)

Task1: PID tuning & Straight line test

1. In “controller.cpp”:
    In function controller_init(): tune the parameters of PID here: the order is P,D,I (check the definition of the function if you want)
2. In waypointfollower.h”:
    a. Line 44: change the vector *square_waypoints* to be size of 1 (because for straight line test there’s only one target position to go)
3. In “waypointfollower.cpp”:
    a. Line 75: uncomment line 78 and comment line 81~84, which is: set the target position here. (the original thing here is a square path)
4. Catkin_make and Run

5. Expected performance:
    a. Since the boat is using RTR algorithm to follow a path, the boat may first just rotate itself to the right direction to the target position, and then move forward to it.
    Note: the rotation is controlled by the angle PID controller (angular_pos_pid)

b. If the boat keeps rotating by more than 180 degree in this case, it may be because the sign of motor is output of angular_pos_pid is not right, just change the sign of outAng at line 37 of “controller.cpp”
c. During moving forward process, the boat is controlled by a cascade PID:

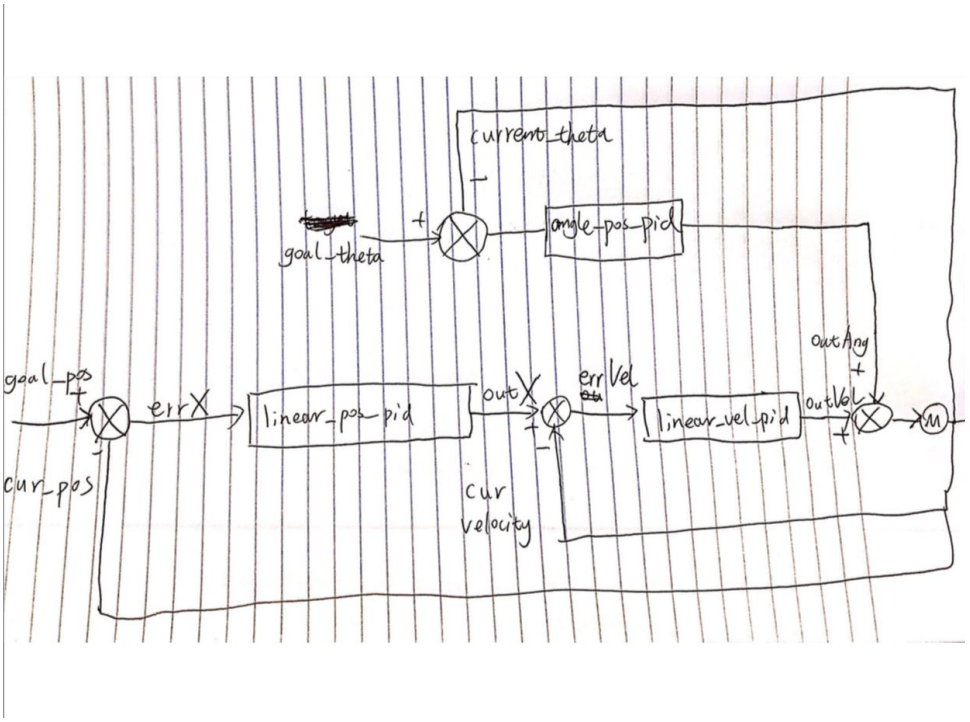

Figure 1: Cascade PID of translation control + PID of rotation control

To tune the pid, start from the inner one to outer one: i.e. set output of the outer one (*outX*) as a constant velocity value in range [-1, 1];

The inner one (linear_vel_pid) is to be tuned in an order: first K, then I, then D. But the order is not strict. You may do like this: change I -> change D -> change I -> change D -> change P -> ….

But don’t be naive, the min/max limitation of D and I should be an importance factor for the success. The min/max limitation is to let the output of D term and I term have similar proportion of K term.

The outer one tuning is the same after the inner one is done. (Don’t forget to set the output back as what it is.) And don’t change any PID params of inner controller when you are tuning the outer one.

After you have a proper configuration of all the parameters, the boat should first rotate to face the target position and then move straightly to it.

Task2: Waypoint following (after task1 is done)

1. In “waypointfollower.cpp”, line 81~84:
    a. Uncomment these lines and comment line 78.
2. Catkin_make and Run

3. Expected Performance:
    a. The boat should follow a path of a square with sides being of 5 meters.