University of Vermont
Autonomy
—
HW 5

Assigned: Thursday October 10, 2024
Due: Thursday Oct. 17, 2024 at 11:59pm.

Similar to the previous assignment, this assignment contains a single [long] problem and will be graded out of 15 points. Start early! Please submit a PDF file of your solutions, and python files as requested in the problems. The figures in your PDF file must be properly labeled. The Python files must be properly commented.

Objective

In this homework, you will build upon the previous assignment to implement the go-to-goal (GTG) and avoid obstacle (AO) maneuvers on the planar drone. The ideas you will employ here parallel those we developed in class for the differential drive ground robot (much of the code will be similar, and some of it may even be identical). This exercise will give you practice in cascaded controller design and GTG/AO maneuvers.

Preparation

Download the files attached to this homework assignment from Brightspace and save them to the same folder. Set up your Python environment by installing the necessary packages listed in `requirements.txt`. Note that the package `shapely` is new; it allows us to work with polygons representing obstacles. You will also find several familiar files from the previous homework assignment: `controller.py`, `main.py`, `PID.py`, `quadcopter.py`, and `utilities.py`. In this homework, you will only modify `controller.py`. **Do not modify any other files.** As before, `main.py` is the main file that runs the Pygame simulation.

Controller

In this homework assignment, the user (i.e. the pilot) specifies a target (goal) location for the drone by clicking anywhere on the screen. The drone will then autonomously navigate to the goal while avoiding obstacles.

As in the previous homework, we assume that both ϕ and $\dot{\phi}$ are perfectly measured. Additionally, x, y, $\dot{x}$, and $\dot{y}$ are perfectly measured and available to the controller. The drone also has access to onboard LIDAR data, which includes the distance to all obstacles within a 3m radius.

The controller you will design builds upon the controller you developed in the previous homework assignment and, similar to the differential drive robot controller, consists of several layers/loops:

- An outer layer with GTG and AO logic to generate a desired velocity vector. This logic will be similar to the one we used for the differential drive robot.
- A middle layer converts the desired velocity vector into a desired drone angle ϕ_{des} and a desired collective thrust u_{col}. The intuition is as follows: to generate horizontal velocity, the drone must tilt, creating a horizontal thrust component that moves it sideways. For vertical velocity, we directly control the vertical thrust. Using these intuitions, we set ϕ_{des} proportional to the error between the actual and desired velocities in the x-direction. Furthermore, we set u_{col} as the sum of two terms: one that compensates for gravity, and another to yield the correct u_y. The first term is simply given by

$\frac{mg}{\cos(\phi)}$. The second term is set proportional to the error between the actual and desired velocities in the y-direction.

- An Inner layer, which includes the angle and angular rate controllers from the previous homework. This layer takes ϕ_{des} and u_{col} from the middle layer and computes the final controller output: the desired motor thrusts u_1 and u_2.

Assignment and deliverables

To complete this assignment, perform the following steps:

1. (3 points) Using the plant and inner loop controllers from the previous homework, along with the middle layer described above and the GTG/AO controllers from class, draw the block diagram of the entire control system. Include the plant (i.e., motor, angle, and position dynamics), the controllers (GTG/AO and angle control), and the feedback paths.

2. Open `controller.py` and modify the `__init__` function, similar to the previous homework, to initialize two PID objects for the left and right motors.

3. You will now modify the `update()` function, which now has the syntax

   ```
   def update(self, phi, phi_dot, x, y, x_dot, y_dot, x_des, y_des, points)
   ```

   Modify the `update()` function to implement the middle and inner layers (not yet the GTG/AO). To do this, under the comment that says "blend U_AO and U_GTG to compute ux and uy", define two variables: `ux`, representing the desired velocity in the x-direction, initialized to 2, and `uy`, representing the desired velocity in the y-direction, initialized to 1.

   Now, based on the description of the middle layer (see above), write the code to convert `ux` and `uy` into `desired_angle` (i.e., ϕ_{des}) and `desired_thrust` (i.e., u_{col}). Use proportional gain 0.5 for ϕ_{des} and 10 for u_{col}. Finally, add your code from the previous homework to compute the motor commands.

4. (3 points) Open `main.py` and run it. Press the spacebar to start the simulation. The drone should begin drifting with a horizontal velocity of 2 and a vertical velocity of 1 m/s. It will crash into the obstacles, but that's expected for now. After some time, pause the simulation by pressing the spacebar again, and click the "Save to CSV" button at the top right. Next, open the `data.csv` file in Python, MATLAB, or Excel, and plot "xdot" and "ydot" (4th and 5th columns). Ensure that xdot tracks 2 and ydot tracks 1 at steady state. Include these plots in your solutions.

5. Now that your velocity controller is working, you can design GTG and AO as we did for the differential drive robot. Bring in the GTG and AO code, and use it to compute `ux` and `uy` (instead of hard-coding `ux` to 2 and `uy` to 1 as you did above). Use the same `Kp_obstacle` function from class, and clip both GTG and AO velocity vectors to limit their magnitude to 2 m/s, as we did with the differential drive robot. Start with a GTG proportional gain of 1.

6. (3 points) Open `main.py` and run the simulation. Click anywhere on the screen and make sure that the drone flies to that point while avoiding obstacles. If the goal is behind a wall, the drone might get stuck and that is normal. We will fix it in a future assignment. After the drone gets to its new location, hit spacebar again to pause and click the button "Save to CSV". Open the csv file with Python, Matlab, or Excel and plot the signals. Include the plots in your solution. Were you able to achieve the desired outcomes?

7. (3 points) Now experiment with the proportional gains of the GTG control logic, the `ux` logic, and the `uy` logic, as well as the `Kp_obstacle` function used in the AO logic. How do each of these affect the performance? Adjust the gains/AO function to improve the tracking and obstacle avoidance performance. Include the final proportional gains and `Kp_obstacle` of your design, and include plots to show the improvement over the base response.

8. (3 points) For the proportional controller that maps `ux - xdot` onto ϕ_{des}, we used a value of 0.5 in the base design (you may have changed it to optimize performance). How can you use the tools you have learned so far (e.g., pidTuner) to design this proportional gain?

Submit the above deliverables as a single PDF file. Please also include your `controller.py`. Make sure your code is thoroughly commented.