AE 483
UAV Navigation and Control
Lab Manual

Lab 4

Navigation and Obstacle Avoidance

4.1 Objectives

In this lab, you will extend basic position and attitude control by adding a high level motion planning algorithm which moves the vehicle from an initial state to a goal state while avoiding obstacles. In particular, you will:

- Modify the Outer in simulation to allow for yaw control.
- Develop a motion planner in simulation to move from an initial state x_{start} to a goal state x_{goal} without obstacles using the gradient descent method.
- Modify the motion planner in simulation to move the vehicle from start to goal while avoiding obstacles.
- Implement the gradient descent in planner.c and run on simulator.
- Implement your motion planner in hardware.

4.2 Tasks

4.2.1 Yaw control

The objective in this task is to control the yaw angle of the vehicle such that the body x-axis points at a fixed 3D point. In the previous lab, you developed a control policy which regulates 3D position and zero yaw angle. In this task, you will modify your control policy by writing your desired roll and pitch angles as functions of yaw.

4.2.2 Gradient Descent

The objective in this task is to simulate translation along a straight line from an initial state to a goal state. In the previous lab, you developed a control policy which regulates 3D position. In this lab you will implement a gradient descent approach, in which you will simply take a step in the direction of the goal at each time step. This approach will form the foundation of later motion planners based on potential function optimization. When your MATLAB simulation and simulator produces stable results, you will implement this in hardware.

4.3 Procedure

4.3.1 Yaw control

1. Use your code from `lab.c` in Lab 3, to modify your outer loop controller `Outer.m`. In that lab, you wrote the desired roll and pitch as functions of yaw. Use this result to point the vehicle x-axis at a fixed location while it moves from start to goal. Note that the desired yaw angle can be computed using straight forward trigonometry in the x-y plane.
   First, we will modify our x-axis and y-axis outer loop controllers to compute desired accelerations.

$$\begin{aligned} a_x &= -K(x_{o_1} - x^*_{o_1}) \\ a_y &= -K(x_{o_2} - x^*_{o_2}) \end{aligned}$$

   Then compute the desired pitch and roll as follows:

$$\begin{aligned} \delta\theta_2 &= a_x \cos\psi - a_y \sin\psi \\ \delta\theta_3 &= a_x \sin\psi + a_y \cos\psi \end{aligned}$$

4.3.2 Gradient Descent: Simulation

1. Let $x_{start} = (0, 0, -1)$ and $x_{goal} = (2, 0, -1)$. Modify your outer loop controller `Outer.m` from Lab 3. Call the `planner.m` function inside `Outer.m` to obtain the desired position and yaw.
2. Modify your planner to take a step in the direction of x_{goal} at every time step of the outer loop. Use the same x_{goal} as above. Modify `planner.m` to compute a step from the vehicle's current position towards the goal using the gradient descent method. We will implement the function `GetGradient.m`

to obtain the total gradient of the attractive and repulsive potential functions. First, in your `planner.m` code, initialize the following parameters near the top of your code:

```
% Set parameters
params.katt = 1;
params.batt = 1;
params.kdescent = 1;
params.bdescent = 1;

% Set goal position
goal.q = [2; 0; -1];
```

Let's recall what these parameters do:

`param.katt`
: If the magnitude of the difference between q and q_{goal} is less than or equal to b_{att}, the gradient is computed as the scaled vector between q and q_{goal}

$$\nabla_q f_{att}(q) = k_{att}(q - q_{goal}) \qquad \text{if} \quad \|q - q_{goal}\| \leq b_{att}$$

`param.batt`
: If the magnitude of the vector between q and q_{goal} is greater than b_{att}, then the gradient is calculated as

$$\nabla_q f_{att}(q) = k_{att} b_{att} \frac{(q - q_{goal})}{\|q - q_{goal}\|} \qquad \text{if} \quad \|q - q_{goal}\| > b_{att}$$

`param.kdescent`
: The "step" that we compute to locally minimize $f(q)$ is computed as

$$\Delta q = -\Delta t(k_{descent} \nabla_q f(q)) \qquad \text{if} \quad \|\Delta q\| \leq b_{descent}$$

`param.bdescent`
: If the step Δq is larger than $b_{descent}$ then we recompute the step to be

$$\Delta q = -b_{descent} \left(\frac{\nabla_q f(q)}{\|\nabla_q f(q)\|} \right) \qquad \text{if} \quad \|\Delta q\| > b_{descent}$$

You will have to find good values for these parameters by trying different values and observing the quadrotor simulation behavior. We suggest that you begin with tuning `param.katt` and `param.batt` equal to 1. Next in `planner.m` compute the small step in position `dq`. Our next desired position is then computed:

```
x_nom = drone.q + dq;
```

where `drone.q` represents the vehicle's current position.

3. Implement repulsive potential function to enable obstacle avoidance. In particular, in `planner.m`, you will need to add the parameters:

   ```
   drone.r = 0.1;
   param.krep = 1;
   param.brep = 1;
   ```

   where `drone.r` can be set by measuring the vehicle, and `krep` and `brep` must be set by tuning in simulation and hardware experiments.
   For our lab we model both the quadrotor and the obstacle as spherical bodies. The repulsive gradient can be computed as follows:

   `param.krep`
   : If the distance between the spheres is less than or equal to b_{rep}, the gradient is computed as:

   $$\nabla_q f_{rep}(q) = -k_{rep}\left(\frac{1}{d} - \frac{1}{b_{rep}}\right)\left(\frac{1}{d}\right)^2 \nabla_q d$$

   where d is the distance between the two spheres after considering the radii.

   `param.brep`
   : If the distance between the spheres is greater than b_{rep}, then the gradient is 0

4.3.3 Gradient Descent: Simulator

1. Place the `Quad_Sim` folder in the MATLAB root directory (C:\MATLABR2016a \).
2. Inside the `matlab_vs2015_c` folder, open the `matlab_vs2015_c.sln` in **Visual Studio 2015**. Open `planner.c`.
3. Navigate to the `PotentialField` function and add your gradient descent code inside the `if` block. All the necessary parameters and variables have been defined at the beginning of `planner.c`.
4. After filling up `planner.c`, build the code and ensure there are no errors.
5. Run the Simulator by pressing `Ctrl + F5`. Show the Simulator output to your TA.
6. In the `matlab_vs2015_c.c` you can change the initial conditions, simulation time and the initial and final obstacle position.

4.3.4 Gradient Descent: GroundStation

1. In `Lab4_codes.zip`, open the `AE483GroundStation.sln` file in `AE483GroundStation_Lab4` with **Visual Studio 2010**.
2. Inside `Source Files` open the `planner.c` file.
3. From your simulator, copy your gradient descent code from the `if (PotFlag == 1)` block inside the `PotentialField` function.
4. Paste this code in the same location inside `planner.c` in the `AE483GroundStation_Lab4` project.
5. Place the `readxyz.txt` in your `U` drive.
6. Show the GroundStation code to your TA and proceed to fly.

4.4 Checkoff Points

In lab, show the TA your:

1. `QuadrotorSim` files `Outer.m`, `planner.m` and `GetGradient.m`.
2. `planner.c` which implements your code in the Simulator and GroundStation.
3. Work with the TA to set obstacles in the workspace. Collect flight data of demonstrating flight from a start position to the goal while avoiding obstacles.

4.5 Report

Write a lab report which answers the following problems.

Problem 4.1. Run your simulation for the case when the drone moves from [0, 0, -1] to [5, 0, -1] while avoiding two spherical obstacles at [2, 0, -0.9] and [3, 0, -1.2]. Assume the obstacles to have a radius of 0.1.

Problem 4.2. Visualize your flight (where simulation parameters have been set to match your hardware flight data) by drawing a 3D animation of the quad-rotor. Copy your code from Lab 1, `lab1_drawquad.m`, name it `lab4_drawquad.m`, and modify it to show a simple point where obstacle is (i.e. you do not need to code the graphics needed to draw a full 3D sphere with correct radius).

Note: Your report should carefully describe anything that went "wrong" with the experiments, or any mismatch between the data and what you saw with your own eyes. It should be **written with your lab group**, and submitted by email to **ae483.fall16+*TA-name*@gmail.com** with the subject line "Lab

4 Report" and the attached report should be named with the last names of all group members "*Lastname1-Lastname2-Lastname3*-Lab-4.pdf".