Activity 2

# Folder: algorithm\_2

## Quickstart - How it works

Line Detection

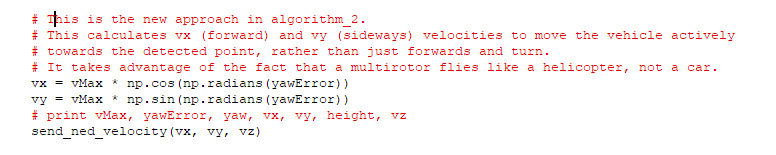
In common with detectline1.py, detectline2.py simply looks for where the red line crosses a band across the middle of the image, indicating the position with a signal between -1 and +1.

Control

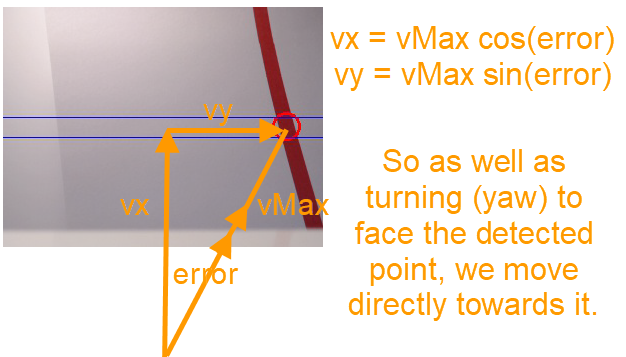
The code in follow2.py also takes the signal and converts it to the ‘error’ angle. It also uses the same method of yaw control to turn towards the detected point on the line.

What it does differently is to do with the ‘forward direction’. Rather than just turn towards the line and go ‘forwards’, it also moves the vehicle directly towards the detected line by introducing a sideways component of velocity. Unlike a car or winged aircraft, it can do this because a multirotor can move in any direction.

The important code is at line 170 in follow2.py



So it is using trigonometry to calculate x (forwards) and y (sideways) components to move more directly to the detected point on the line:



Note: The code snippet above changes the angle yawError from degrees (the units we are working in) to radians (the units used by some of the maths functions). It’s important (1 radian is approx. 57 degrees!), but don’t get distracted by that – just make sure you understand the use of the trigonometry.

## What to Do

**IMPORTANT:**

**Test your code on the simulator every time before real flight.**

**Set the flightHeight variable to -1.0 for real flight, or a different value given to you. DO NOT leave it at   
-5.0 m used for the simulator!**

**Testing algorithm**

*Do the same tests as in the code for Algorithm\_1, starting with the best values of vMax and yawP from those tests. However, you may need to adapt how you change the values. Be prepared to reduce yawP or vMax if necessary, and also to change the amount by which they are incremented.*

*You might also want to turn on flight logging (black box) and save flight images from the camera as indicated below.*

* Start with best value of vMax from Activity 1
* Start with a best value of yawP from Activity 1
* Test that it follows the line with these values
* Repeatedly increase vMax gradually (suggest by 0.25 m/s) until the vehicle can no longer follow the line.
* Repeatedly increase the value of yawP gradually (suggest by 0.05) to a value so that the vehicle still follows the line without ‘oscillating’ too much.

Results

Use the same or similar table to that for Activity 1.

## How to Run Code in the Simulator

1. Start the simulator if necessary and minimise the terminal, console and map windows.
2. Open the folder algorithm\_2
3. Select ‘Tools’ in the menu bar
4. Select ‘Open a terminal’ from the dropdown options
5. Use the up/down arrow keys to find the command ‘python follow2.py –connect “udp:127.0.0.1:5500” and press enter. Nb: The actual address in this line may vary from this.
6. The terminal will indicate a connection has been made and you can see the simulated motion on the map and console.
7. To close the simulation, select the folder terminal and Ctrl-C. Optionally type ‘mode rtl’ in the simulator terminal to have the drone return to land.

## Going Deeper

### Recording Flight Data – the ‘Black Box’

You can request that certain data is written to a text file as the vehicle flies. This is very useful to analyse the performance after each flight. For example, the yaw and velocity values are recorded, so it’s possible to see how well the vehicle turns to the line, or whether it constantly overshoots it. It’s easy to import the data into the spreadsheet app in the virtual environment to graph the motion and look for patterns.

To turn on logging, set the Boolean variable recordLog to True. Data from every 10th loop is recorded. Further information is available separately. However, it is best to delete the contents of the log file (locationdata.txt) before each new flight.

### Recording Flight Images from the Camera

There is code in detectline2.py that will save every 10th image to file. This can be very useful to see what the vehicle itself is seeing. For example, you might spot that it is picking up an object near the track which is red, or that it is not detecting the red line in certain parts of the track due to lighting conditions.

Separate information is available to help you turn on and manage image-saving.

### A Better Way?

Can you think of any ways in which the current approach is limited? How could it be better? If you have some ideas and can would like to consider making limited changes to the code itself, talk to a MAAXX ambassador. It’s fine to consider code changes but they would need to be thoroughly tested on the flight simulator.

Once you consider you have learned all you can, it’s time to go onto Activity 3.