Activity 1

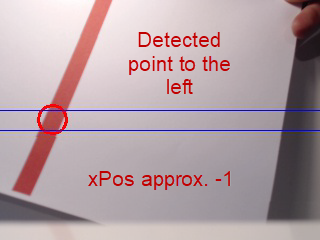
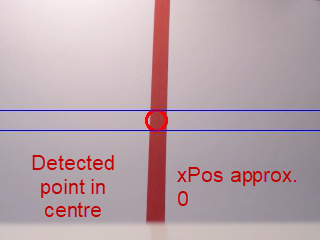
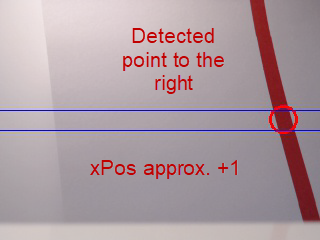
# Folder: algorithm\_1

# Quickstart - How it works

Line Detection

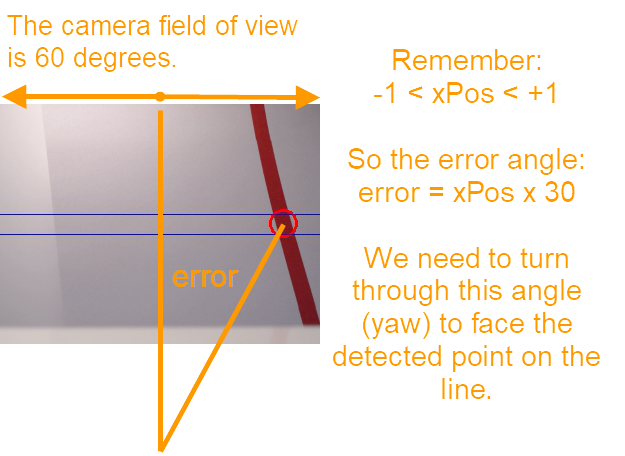
The Python code detectline1.py simply looks for where the red line crosses a band across the middle of the image. This band is called a region of interest (ROI).

The code returns a ‘signal’ which is a number (xPos) between -1 and +1 to indicate where the line crosses the ROI. -1 is extreme left, 0 is centre, +1 is extreme right.

Control

The code in follow1.py takes the signal (xPos) and converts it to an angle. The angle between the current direction faced by the vehicle and the detected line is called the ‘error’ (yawError). The measurable error is approximately between -30 degrees (left) to +30 degrees (right). Here’s how we work it out:



Of course we want the error to be zero. We do this by turning the vehicle in the direction of the error (left if negative, right if positive). The code in follow1.py only controls the direction of the vehicle through yaw.

Ideally, we would simply turn the vehicle through this angle to follow the line. Unfortunately, it’s not that simple! If we tried to turn (yaw) the vehicle through that complete angle as fast as we could, it would overshoot. It would then have to go back a bit… and probably overshoot again. This causes an oscillation which limits performance and could even result in the loss of control of the vehicle.

**Proportional Constant - yawP**

To reduce the chance of this happening, we don’t try to make the whole turn straight away. We just try to correct a fraction of it. For example, we might try to turn half of the error angle, or some other fraction each time. Doing this repeatedly in a loop of code still closes the ‘gap’ but reduces the chance of overshoot and those oscillations.

The fraction is a constant called the P value (for proportional). In follow1.py, this constant is called **yawP** and is a decimal (float) between 0 and 1.0.

In mathematical terms:

Yaw (amount to turn this loop) = yawError (difference between current direction and line) x yawP

By repeating this many times each second, the vehicle will turn towards the line, but will turn more slowly as it gets ‘closer’.

**Forward Velocity - vMax**

The code in follow1.py fixes the forward velocity at vMax. MAAXX-Europe is a race, so we want vMax to be as large as possible whilst still following the line. If vMax is too great, the vehicle will not be able to turn quickly enough to follow the line. Too small and there is not enough forward velocity to effectively control the direction.

## 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**

Start with a low velocity of vMax – say 0.5 m/s.

Start with a low value of yawP – say 0.2

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.

Repeatedly increase vMax gradually (suggest by 0.25 m/s) until the vehicle can no longer follow the line.

Try the straight line in the test arena before attempting the oval track in the race arena.

Results

The following table may be helpful to illustrate some results – your observations will be different!

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Run No. | yawP | vMax  (m/s) | Lap time (s) | Observations  (Does it turn quickly enough to follow the line?  Does it overshoot and oscillate?) |
| 1. | 0.2 | 0.5 | n/a | Does not turn quickly enough to follow the line. |
| 2. | 0.25 | 0.5 | 40 | Follows the line, but only just! |
| 3. | 0.3 | 0.5 | n/a | Keeps turning past the line and getting lost |
| 4… | 0.25 | 0.75 | 37 | Follows the line |

## Finally

The code in folder algorithm\_1 uses a very simple technique. It does not work very well and you may find it difficult to find values of vMax and yawP that enable the vehicle to follow the line reliably.

If this is the case and you have learnt all you can (having thought about and discussed why this is so) take your most promising values of vMax and yawP and go onto Activity 2.

## 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\_1
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 follow1.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 ctl-C. Optionally type ‘mode rtl’ in the simulator terminal to have the drone return to land.