Session 6A: Line Fitting

Objectives

- Learn the basics of line fitting
- Apply line fitting to live image from robocar

Polynomial Curve Fitting

To estimate the line markings from our detected pixels, we'll fit a line using interpolation. Curve fitting is the process of constructing a curve, or mathematical function, that has the best "fit" to a series of data points. The most common definition of a measure of "fit" is the least square error.

Here is an example of the results of fitting a 2nd degree polynomial (red) and a first degree polynomial (green) to a set of data points (blue).

(image source: https://ardianumam.wordpress.com/2017/09/21/polynomial-regression-using-least-square-estimation/)

For the robocar application, we'll fit a first degree polynomial, i.e. a straight line using, to the extracted pixels corresponding to the detected line, using the numpy library's `polyfit` function:

> `numpy.`**`polyfit`**(*x*, *y*, *deg*, *rcond=None*, *full=False*, *w=None*, *cov=False*)
>
> Least squares polynomial fit.
>
> Fit a polynomial `p(x) = p[0] * x**deg + ... + p[deg]` of degree *deg* to points *(x, y)*. Returns a vector of coefficients *p* that minimises the squared error.

(https://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html).

Exercise 6-1a: Line Fitting

Open file line.py in ex-6-1-Line_Fit/line-ex, which is a continuation of the previous openCV exercises.

TO-DO:

- fit a degree one (linear) polynomial with collected points `pts_x` and `pts_y`.
- extract the polynomial into a class/method `p`, where `p(x) = y` (estimate).
- Check that your fitted line is visually optimal by inspecting the output image `05line.jpy`.

Tip: flip axes such that:

- x is the vertical axis (+ve=down), and
- y is the horizontal axis (+ve=across)

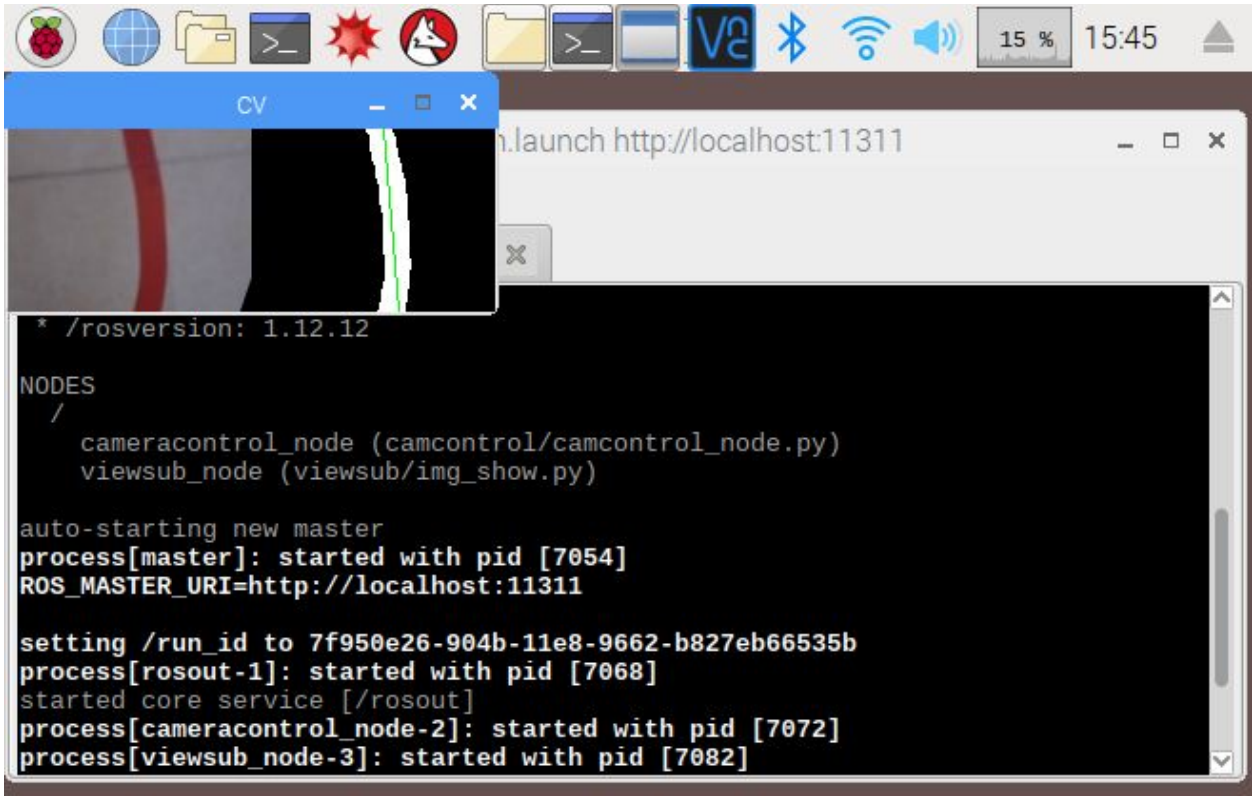
Exercise 6-1b: ROS Line Fitter

Open folder ros-ex in ex-6-1-Line_Fit, which contains a skeleton ROS workspace with two nodes:

- **camcontrol** - Publisher node which does the line-fitting and publish a superimposed image.
- **viewsub** - Subscriber node which reads the published image and display it on the screen.

TO-DO:

- Open the source file `camcontrol_node.py` in the **camcontrol** node and find the set of selected points in function `fit_line_pub()` named `(pts_x,pts_y)`
- fit a degree one (linear) polynomial with collected points `pts_x` and `pts_y`.
- extract the polynomial into a class/method `p`, where `p(x) = y` (estimate).
- Compile and run the ROS workspace using the provided launch file.
- Place the camera over a test red line on the challenge tile. You should see a window with the raw image on the left and the superimposed fitted line (in green) over detected points on the right, like the example below.
- Move the camera around to observe the red line at different angles. What happens when the line is totally outside the camera's view?


15 %
15:45
CV
.launch http://localhost:11311
* /rosversion: 1.12.12
NODES
/
cameracontrol_node (camcontrol/camcontrol_node.py)
viewsub_node (viewsub/img_show.py)
auto-starting new master
process[master]: started with pid [7054]
ROS_MASTER_URI=http://localhost:11311
setting /run_id to 7f950e26-904b-11e8-9662-b827eb66535b
process[rosout-1]: started with pid [7068]
started core service [/rosout]
process[cameracontrol_node-2]: started with pid [7072]
process[viewsub_node-3]: started with pid [7082]