Programming Assignment #1
A Robot Controller for Line Following (and more!)

Nik Swoboda
nswoboda@fi.upm.es

Universidad Politécnica de Madrid

February 25, 2020

Project Code

The code base for this assignment is in Moodle:

- `proyectoRobotica-p27-v3.0.zip` (python 2.7)
- `proyectoRobotica-p36-v1.0.zip` (python 3.6)

Included in those zip:

- `BrainTestLine.py` - A robot Brain with a basic controller which can more or less follow a simple line
- `lineTestWorld.py` - An empty PyrobotSimulator world which loads one of the line images
- `lineBackground-1.png`, `lineBackground-2.png` - Sample line images for use in the PyrobotSimulator
- `runTestLineSim` - one line "script" (unix) to run the code
- `lineSimulation.py` *(Only in the python 2.7 version.)* - Modifications to the PyrobotSimulator to be able to use an image as the "ground" of the world and to "simulate" a *very* simple line detecting algorithm

Setting your PYTHONPATH

Remember the project code directory needs to appear *before* the Pyro code in your PYTHONPATH. (One easy way to do this is to start your PYTHONPATH with "." and to work in the `proyectoRobotica-v3.0` directory.)

Linux and OsX with bash:

```
export PYTHONPATH=.:/home/robotica/pyrobot
```

or

```
export PYTHONPATH=/home/robotica/proyectoRobotica-v3.0:\
    /home/robotica/pyrobot
```

(To make it permanent include it in your .bashrc)

In windows:

See `http://www.java.com/en/download/help/path.xml`

Simulated video processing

The modified PyrobotSimulator contains a very simple line detection algorithm. It works as follows:

- Using a simulated field of view of a forward mounted robot camera (with characteristics similar to the real cameras we will use), extract from the background image the center band of that field of view
- Apply a threshold to separate black and white
- Calculate the center of the black and the distance (positive or negative) between that point and the middle of the field of vision

Simulated video processing

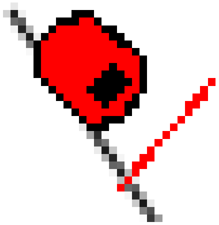
In the case shown on the right, the following code:

```
hasLine,lineDistance,searchRange =
  eval(self.robot.simulation[0].\
    eval("self.getLineProperties()"))
print hasLine,lineDistance,searchRange
```

prints:

```
True -8.39272471214 10
```

(Note: the line search area is shown as a red line. This red line is not shown when running the code.)



Simulated video processing

Note: The simulated video processing will not:

- detect multiple lines in the field of vision
- detect the angle of the line
- deal well with images which are not black and white (grays used to anti-alias the line are OK).
- deal well with lines which are close to perpendicular to the robot (it just returns the ‘center’ of the black pixels which are visible)

The Assignment

For this assignment you must modify the `BrainTestLine.py` file.

You are encouraged to change the world file, `lineTestWorld.py`. You can change the simulator background file (draw your own lines), move the robot's initial position, include obstacles etc. to test your controller.

You should not need to change the simulator file `lineSimulation.py` (but you can if you like). (This file is included in the python 3.6 version of pyrobot.)

However, if you find bugs or have suggestions for improvements, please let us know!

The Assignment

Your basic task is to replace and improve the line-following controller.

At the very least we expect submissions which can follow a curved line like the ones included in the zip.

For a better grade, you should also consider:

- different and “more complicated” lines
- avoiding obstacles placed on or near the line
- ability to follow lines with very sharp curves or angles
- ability to search for a line if not found at the start.
  (Note: The line could be perpendicular to the robot when found! This will not work well with the existing line detection algorithm, but for now you can work around this problem.)
- ...

Changing the Simulation's world

As mentioned previously, please experiment with different lines, robot positions, obstacles etc.

To change the background image (in `lineTestWorld.py`):

```
          # (width, height), (offset x, offset y), scale
    sim = LineSimulation((450,675), (20,650), 32,
           background="line-images/lineBackground-1.png")
```

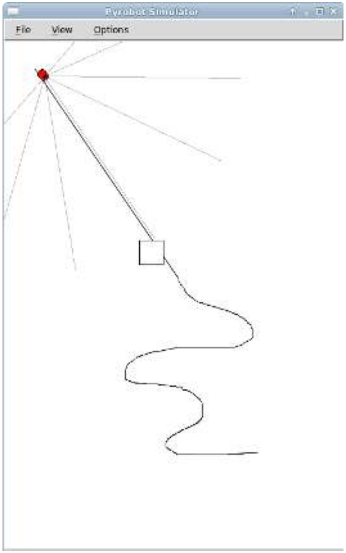
To change the initial position of the robot (also in `lineTestWorld.py`):

```
    sim.addRobot(60000,
      # name, x, y, th, boundingBox
      TkPioneer("RedErratic", 1, 18.9, 4.0,
                  ((.185, .185, -.185, -.185),
                   (.2, -.2, -.2, .2))))
```

Changing the Simulation's world

To add an obstacle (also in `lineTestWorld.py`):

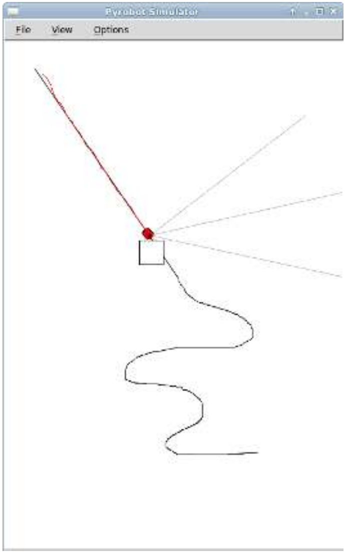
```
# x1, y1, x2, y2
sim.addBox(5, 12, 6, 11)
```



Changing the Simulation's world

To add a trail you can use the menu option: `View->trail` or

```
  # after adding the robot
  sim.robots[0].\
    display["trail"] = 1
```



Some Reminders

- You can work alone or in groups of up to 3 students
- Your work must be submitted through Moodle before 23:55 on March 2nd
- You should submit a zip containing:
  - your Brain and the python files which you modified from the code base
  - a brief report containing:
    - a description of the changes you made
    - examples of tests you ran showing the capabilities of your code (including screen shots is recommended)
- Only one submission per group is needed, but all the group member's names must be in the code and report.

Good luck!