Guertler Fabienne 12-935-508

Robotics Series 5

Exercise 1: Camera Measurements

We took different measures under different circumstances. Mainly there are three experiment setups that we used: different distances under dim light and a right angle, different distances under good lighting and a right angle and different angles under good lighting and a fixed distance. It could be noted that lighting mattered less than angle. We repeated one setup with a different e-puck and a difference could be noted, although not too significant.

Exercise 2: Color recognition

In this exercise the e-puck should give a visual feedback about which color he sees:
1 led → red, 2 leds → blue, 3 leds → green

```
call math.stat(cam.red, minRed, maxRed, meanRed)
```

To calculate *meanRed*, *meanBlue*, *meanGreen* we used the code from the demo presented in the lecture.

```
onevent ir_sensors
    [...]
    elseif color == RED then
        leds[4] = 1
        leds[2] = 0
        leds[6] = 0
    […]
    end
```

The e-puck has a variable *color* which is set to the corresponding color that he sees. He compares this value to the constants *RED, BLUE, GREEN* and *NO_COLOR* and reacts accordingly. We chose to light the leds 2,4,6 as signals. Led 4 is the one on the back of the e-puck and leds 2 and 6 are at the sides. If the color is *RED* he lights *leds[4]*, if color is *BLUE* he lights the ones on the side and if it is *GREEN* all three leds are lit.

```
if meanRed > 24 and meanBlue < 23 and meanGreen < 23 then
    color = RED
elseif meanRed < 22 and meanBlue > 19 and meanGreen < 22 then
    color = BLUE
elseif meanRed < 20 and meanBlue < 20 and meanGreen > 16 then
    color = GREEN
else
    color = NO_COLOR
end
```

The interesting part was to calculate which color he saw. It is not enough to just check which of the three mean values (*meanRed, meanBlue, meanGreen*) is the highest but one has to take into account the ratio they have to each other. For that we tried using the measurements of exercise 1 but we still had to make some adjustments. We did the measurements on an other day than this exercise so that may be a reason. Because of course, it was impossible to recreate exactly the same conditions as when taking the measurements. These ranges of when to set the color to *RED, BLUE* or *GREEN* were determined by experimenting. We started with our measured values and then kept changing them to get a better result. It is still far from perfect, as it sometimes does not see the color it is supposed to or signals colors when it should not.

Exercise 3: Color recognition – Explorer

This exercise is an extension of the previous exercise. Here, the e-puck drives around with the *EXPLORER* behavior and simultaneously gives a visual feedback about the colors he sees. To implement the behavior we used the code from series 3. We would have liked to test other methods of recognizing colors but sadly we did not have enough time left to do that since the measurements took much longer than anticipated. Of course, the color recognition is much easier when the e-puck is standing still so during the implementation of this exercise we also had to tweak the values for the calculation of the color. For example in our calculation he was much too slow to recognize the colors and when driving he often did not show the correct color. By lowering the bounds he signaled colors sooner but this also increases the rate of incorrect signals. With the bounds as they are now, he is much more likely to signal a wrong signal but he is now fast enough to signal the colors when driving around. In the simulation there are no leds so I had to come up with a different method of signaling colors. The system I came up with works like this: e-puck 1 drives around and explores. When he recognizes a color he emits a command with the color as the argument. E-puck 3 reacts to those events by making a certain movement. If the argument he receives is *RED* he drives forward, similarly if he receives *BLUE*/GREEN he starts turning right/left. Whenever e-puck 1 sees none of these colors e-puck 3 receives *NO_COLOR* as an argument. He then stops moving.

```
if avgRed > 6 and avgBlue < 4 and avgGreen < 4 then
    color = RED
elseif avgRed < 4 and avgBlue > 6 and avgGreen < 4 then
    color = BLUE
elseif avgRed < 4 and avgBlue < 4 and avgGreen > 6 then
    color = GREEN
else
    color = NO_COLOR
end
```

The values are also very different in the simulation as opposed to the physical arena. As was explained in the lecture, in the simulator the e-puck sees the colors differently than in reality. Each value in each camera array is either 0 or 100. So of course the boundaries to decide which color he is seeing is different. In the simulation I also used a different method to calculate the color values. Here used calculation of the average of all cameras from the demo. With this method the average is always between 0 and 10 so the boundaries have to be adjusted accordingly.

Note: for the measurements please view the file “05_13_GuertlerF_e_camvalues.pdf”