1 Colors

1.a Calibration script

For this exercise we used the measurements from the previous series. But this time we included a evaluation of our results in percentage to help us find an algorithm to recognize colors.

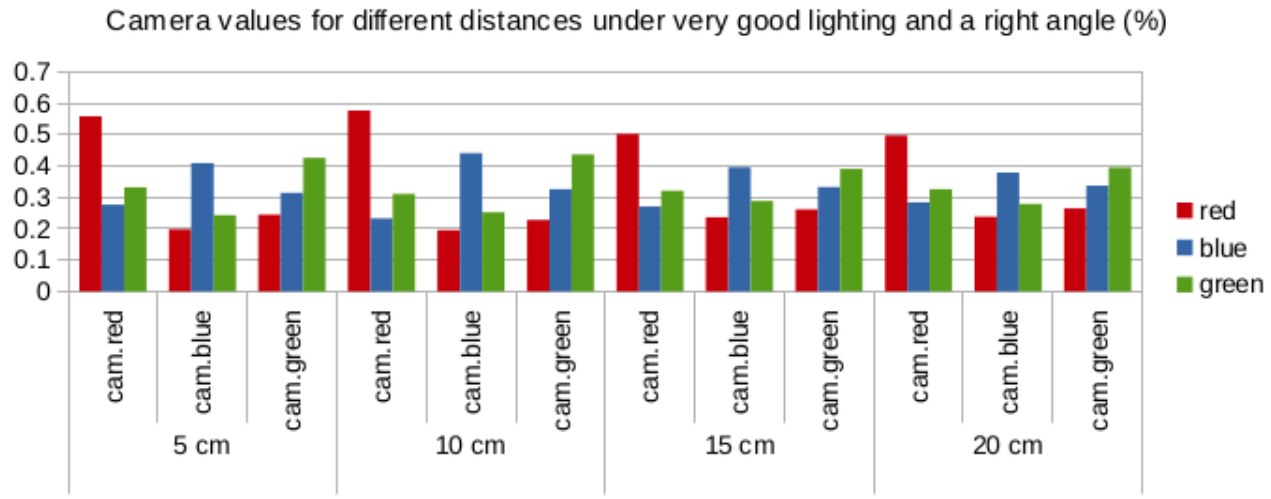

Figure 1: Color measurements in percentage

We used this analysis to come up with the following ranges for the colors:

	cam.red	cam.blue	cam.green
Red	50% - 60%	15% - 25%	20% - 30%
Blue	20% - 30%	35% - 45%	30% - 35%
Green	30% - 35%	20% - 30%	35% - 45%

Table 1: Color ranges in percentage

From the values of Table 1 we figured out algorithms to recognize colors. For example, the ratio of red is generally 2 to 4 times bigger than the ratio of blue.

$$2 \cdot cam.blue \leq cam.red \leq 4 \cdot cam.blue$$
$$2 \cdot cam.green \leq cam.red \leq 3 \cdot cam.green$$

Similarly you can find a calculation for blue and green.

$$14 \cdot cam.red \leq 12 \cdot cam.blue \leq 27 \cdot cam.red$$
$$7 \cdot cam.green \leq 7 \cdot cam.blue \leq 9 \cdot cam.green$$

$$14 \cdot cam.red \leq 12 \cdot cam.green \leq 27 \cdot cam.red$$
$$7 \cdot cam.green \leq 7 \cdot cam.green \leq 9 \cdot cam.green$$

```
for i 0:4 do
        camRed += cam.red[i+28]
end
camRed \= 4
```

We took 5 pixels in the middle of the camera and calculated the average, unlike last time where we calculated the average of every tenth pixels over all the pixels. We used this approach because we felt that when driving around, the pixel right in the front are most important. With this calculation the e-puck's color cognition is similar to a human's in the fact that we also mostly see what is right in front of us and only very little of what is on the sides. Well, like this the e- puck does not recognize any colors on the sides at all. This could be a further improvement of the color recognition algorithm. One could try to find a calculation where the pixels on the front are weighted the most and those on the sides are weighted less. Unfortunately we did not get to this step.

The code that we used to determine the color and the subsequent behavior is not quite like the calculation we came up with in theory but close. As most of the time, when trying to apply theory to the real implementation for the e-puck, we had to adjust some bounds to get a good result. One reason for this might be that we took the measurements over all 60 pixels and for our implementation we only used the center 5 pixels.

```
elseif 7*camRed <= 6*(camBlue +b) and 4*(camBlue -b) <= 9*camRed and
        camGreen -b <= camBlue then
                color = BLUE
end
```

Here, for the calculation of the color blue we deleted one upper bound and used a variable b to adjust. Compared to our calculations of the last series the color recognition worked significantly better. We did similar adjustments for the other colors too.

1.b Approach color

We chose *RED* as the color to approach. Our idea for this exercise was for the e-puck to change behavior according to which color he sees. So in our case, whenever the e-puck recognizes the color *RED* he switches to the *LOVE* behavior. We implemented this as described in Braitenberg's "Vehicules - Experiments in Synthetic Psychology".

```
if color == NO_COLOR then
        leds[4] = 0
        leds[2] = 0
        leds[6] = 0
        behavior = EXPLORER

elseif color == RED then
        leds[4] = 1
        leds[2] = 0
        leds[6] = 0
        behavior = LOVE

        [...]

end
# behavior
if behavior == EXPLORER then
        callsub behaviorExpl

elseif behavior == LOVE then
        callsub behaviorLove
end
```

It always checks the behavior and calls the corresponding subroutine. This approach worked quite good but it needs a good color recognition. If for example the color recognition flickers, the behavior flickers as well. The e-puck also does not actually follow the color. As long as the e-puck sees *RED* he acts as a *LOVER* which results in him following the object. But the *LOVE* behavior does not care about color it works with proximities as seen in series 3. For us this was the implementation with the best results.