RoboCup Final Project Report
Color Detection

Jakob Stimpfl Charlotte Burmeister

May 22, 2018

Contents

1 Introduction

The Nao robot is a humanoid robot produced by Aldebaran. It is 574 mm high and has 26 joints, controlled by servo motors. The environment can be perceived through several sensors. [1]

In this project to goal was to enable the robot to listen to a color, interpret the color heard as RGB values, search for that color in a set distance in front of him and then point to this color. The setup can be seen in fig. 1 and fig. 2

2 Implementation

As predetermined by the course the NAOqi Python Software Development Kit (SDK) was used. This SDK provides the use of C++ modules and enables the user to create own Python modules.[1] The program consists of three main parts: Speech Detection, Color Detection and Movement, which are described in more detail in the following sections. In figure 3 the program organization can be seen.

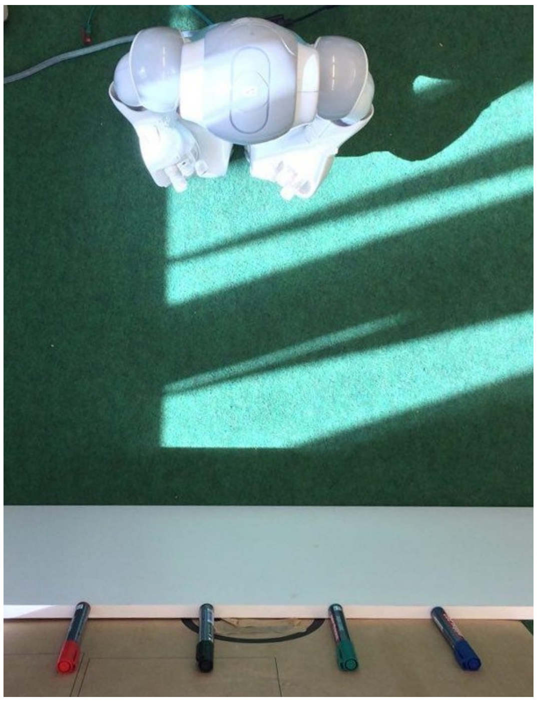

Figure 1: Experimental setup aerial view

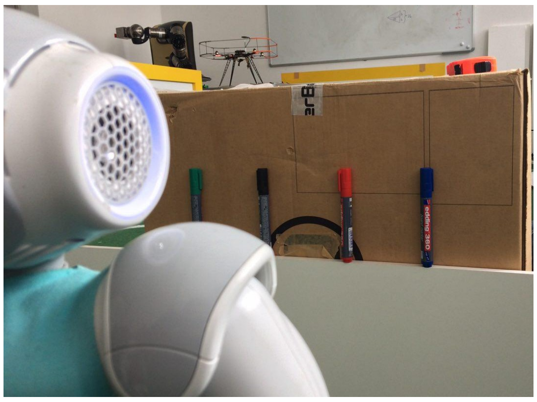

Figure 2: Experimental setup

2.1 Speech Detection

The first part of the project aims at implementing a speech recognition to understand and distinguish the different colors. For this the module “ALSpeechRecognition” was used. It provides the event listener “WordRecognized”. Before subscribing to the listener a vocabulary list has to be specified. Then an event is thrown when a word from the list is recognized. After the event is thrown, the program jumps to the method “OnColorHeard()”, where a simple if-else-query determines the color. Afterwards the next method is called and the Red-Green-

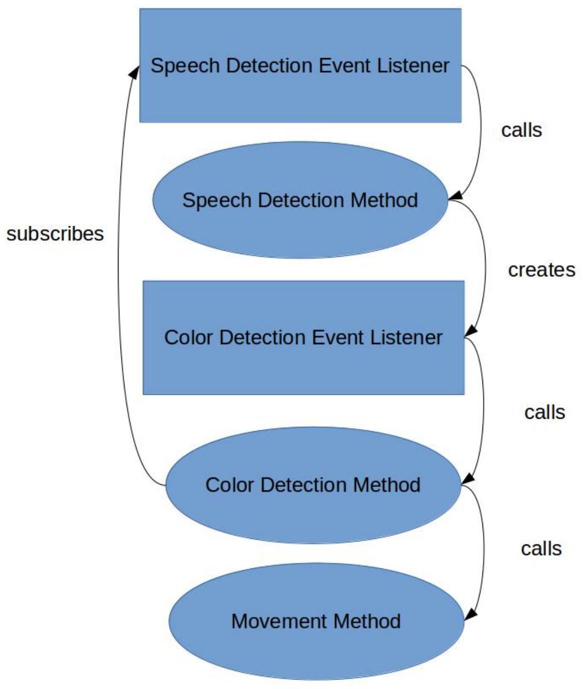

Figure 3: Program structure

Blue values corresponding to the detetected color are passed on.

2.2 Color Detection

The API provides a module “ALColorBlobDetection” that meets the requirements of finding a color spot. To find the color specified by the previous method, the RGB-values and a threshold have to be set through the method “setColor(r, g, b, th)”. By trial and error a threshold of 50 was considered to be working. Additionally the minimal size of the blob was set to 10 pixels and its distance depending on the distance it is search for in: “setObjectProperties(10, distance)”. Then a subscription to the “onColorDetected”-event was made. If that event was thrown in the eventhandler the “getCircle()”-method is used to determine the horizontal distance relative to the whole length of tho blob from the left border of the camera-picture.

2.3 Movement

As described in the previuos section at this point there is an x-value which can be used, to calculate the angle the robot has to move his arm into, to point at the colored object. It has to be noted, that when we talk of the picture the origin is in the left upper corner and x goes from left to right while y goes from top down. If we talk instead of the robot, the robots y-axis is the pictures x-axis and the robots z-axis is in the picture the y-axis, which leaves the robots x-axis to the distance in which an object lays to the robot. As a first distinction it ist to determine whether this x-value is greater than 0.5. If that is the case the right arm is to use otherwise the left. Since the calculation for the left arm is the same then for the right with the small difference, that the pointing-angle has to be negated, subsequently will be explained the calculations only for the right arm.

Having the normalized x-value counted from the left side of the picture, the horizontal distance from the robots center can be calculated. This is done by multiplying it with the horizon-lenght which can be defined as the size, an object could maximal have, in a set distance, so that the robot can still see all of it. For that another parameter is needed, by name, the opening angle α of our camera, defined in the documentation [1]. With the law of sine and the distance d of our robot now the half horizon length can be determined by

$$\frac{horizon}{2} = d \cdot \frac{\sin(\frac{\alpha}{2})}{\sin(90 - \frac{\alpha}{2})}$$

Given the distance robot-to-object and the horizontal distance blob-to-robot-center calculated by

$$distanceToCenter = |xValue - \frac{horizon}{2}|$$

and the set distance center-to-shoulder dy of 98mm now the horizontal distance of blob-to-shoulder can be calculated of the choosen arm:

$$distanceToShoulder = dyShoulder - distanceToCenter$$

Next, the angle in which the arm has to be turned in is:

$$pointingAngle = atan2(distanceToShoulder, distance)$$

in which distance is the distance robot to blob.

As a last thing now one only has to bring the arm in position by using the function angleInterpolationWithSpeed(target, targetAngles, maxSpeedFraction) where maxSpeedFraction is 0.2, target is "RArm" or "LArm", targetAngles is a list of angles with one for each joint where the just calculated pointingAngle is the shoulder angle that is the second one. As it looks more natural the elbow (third angle) can be turned by $-\frac{\pi}{2}$ in case of the right arm and $\frac{\pi}{2}$ in case of the left.

3 Difficulties and future prospects

3.1 Speech Detection

The speech detection which is provided through the API is sometimes returning the wrong color. It is out of scope of this project to implement a new speech detection, but a possible solution could be to compare the probability values for the word list. For values with a small difference the robot could ask for an affirmation of the user if it chooses the right color.

So far only the colors which are specified in the code can be detected by the robot. This limits the interaction possibilities. A more interactive approach would be that any color could be said or that the list of colors can be specified by the person interacting with the robot, e.g. via speech input. But permitting the robot to detect more colors is difficult, as colors appear quite different in changing light conditions. Distinguishing black, red, blue and green is easier as the RGB values are very different.

3.2 Color detection

As a future improvement the minimal blob size should be adjusted depending on the distance if the size of the searched object is given. That might bring better results.

3.3 Movement

Due to the fact that the robot is booting with an expressive listening simulation on, the movement can be a bit tricky to test. That means that it is continuously moving around so one is never sure if everything is working as intended. As a fix, it can be turned off, by subscribing to the proxy "ALAutonomousMoves" and then using its method "setExpressiveListeningEnabled(False)".

Another problem is, that there is no way to get the distance information out of a 2D picture. A good way to avoid that is, to just set it as fixed. Another possibility would be to restrict the form of the searched object so that one can calculate the distance out of changed form.

4 Conclusion

The robot is able to identify a spoken color out of list declared in advance, map this color to its RGB-values, find the color in its field of vision in a fixed distance and point to this color. As a next step it would be desirable to implement a more flexible approach where the colors and the distance don't have to be specified beforehand.

References

[1] *Aldebaran documentation.* May 5, 2018. URL: `http://doc.aldebaran.com/2-1/home_nao.html`.