Humanoid Robots lab course: Soccer

Felix Prahl-Kamps (2782026), Florian Seiler (2782831), and Tridivraj Bhattacharyya (3035538)

Rheinische Friedrich-Wilhelms-Universität Bonn

Abstract. Humanoid Robots performing human tasks is one of the fast expanding areas of research. This includes playing soccer. We implemented an algorithm in order for the Robot Nao to kick a ball and score a goal.

1 Introduction

In recent times, scientific research has achieved in making humanoid robots perform increasing complicated tasks like skiing, soccer etc. Even simple motions like detecting and approaching an object can be non-trivial for a bot in different surroundings. For this lab, we attempt to detect a ball in the environment along with two cylindrical goal posts. The bot needs to approach the ball, position itself according to the goal and kick the ball towards it. There are some constraints applied for the task. We use color to detect objects in the scene like red balls and yellow cylinders as goalposts. Moreover, it is assumed there are no obstacles anywhere in the environment. The code is written in C++. An existing framework is used to interface with the Nao and the motion is created using Choreograph.

2 Background

2.1 Nao

We use the humanoid robot Nao to achieve our goal here. Designed by Softbank Robotics, it has 25 degrees of freedom, two cameras and a humanoid shape. This enables it reproduce human motions with an optimal level of accuracy.

2.2 Existing Framework

We use a C++ framework to program the Nao. QtCreator has been used as the default editor. Once built and run, the code initializes the bot's stance and then carries out any actions programmed into it.
A screenshot is provided in Figure 2, which shows the run window or GUI of the framework. Here sections can be added or removed as needed to make it easier to configure the Nao's actions. It can also help in setting parameters during runtime, like the Hue or Saturation values to detect a specific color. One can also obtain the orientation values of each joint from this interface in real time.

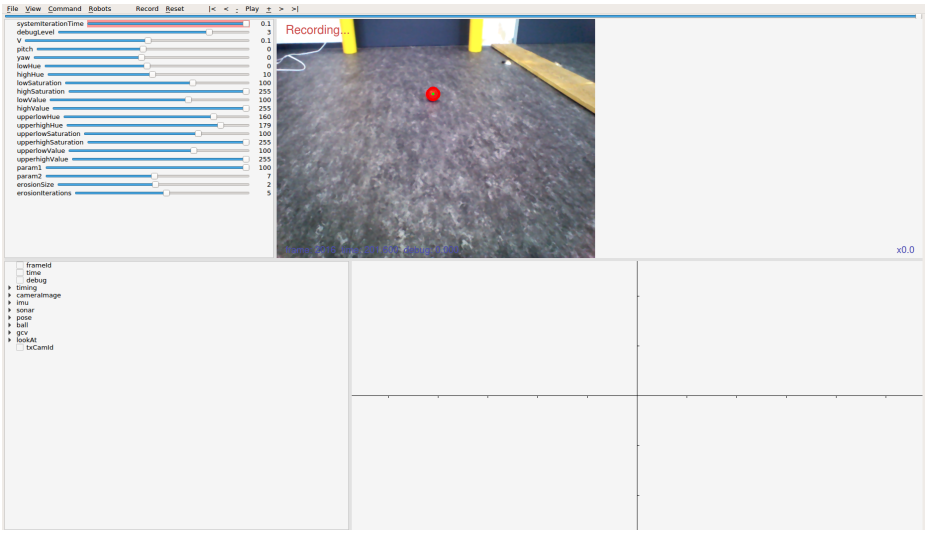


Fig. 1: Screenshot of the Run Window

3 Implementation

The overall task of scoring a goal has to be broken down into small individual steps. This makes it simpler for the bot to fulfill certain conditions before it can perform the kick motion. The basic components of the task include object detection and the state planning based on the position of the ball and goalposts. Each of these are described in detail below:

3.1 Red Ball Detection

To determine the relative position of the ball we use a simple combination of noise reduction, color filtering and circle detection.
The noise in the RGB image is filtered out using the median Blur method. This is then converted to HSV space. A binary mask is obtained from here with the red pixels in a specified range.
The HSV color space consists of two separate ranges of red color: the lower from 0° to 10° and the upper from 170° to 180°. This provides two separate masks which are combined using addWeighted. The output is further improved by applying erosion and dilation.

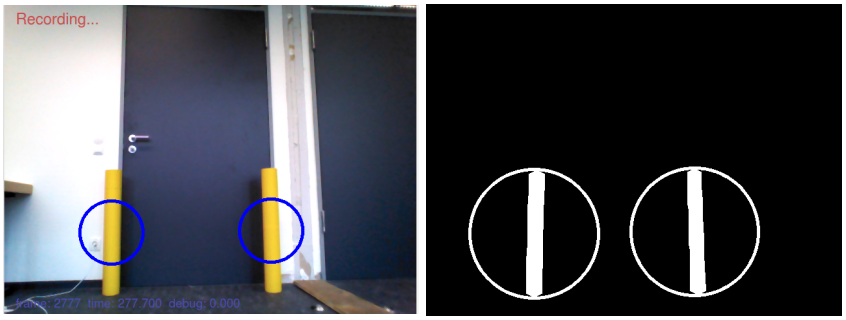
(a) RGB image with different balls (b) Binary Mask of Red Ball

Fig. 2: Ball-Detection

Finally the hough circle transform technique is added to detect circles in the image which in this case is the red ball.

3.2 Yellow Goal Detection

Similar to the Red ball detection, the OpenCV library is used to detect goalpost. The denoised image is converted to the HSV color space from which the range of yellow colors is extracted to create a binary image. Now the range of values for this are highly dependent on the environmental lighting conditions.

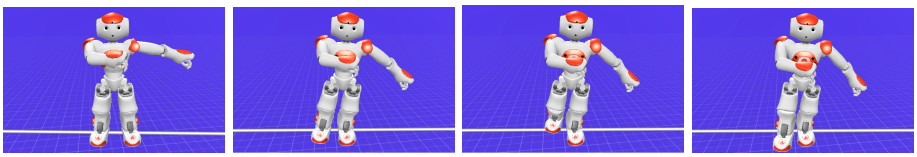


(a) RGB image with Yellow goalposts (b) Binary Mask of Yellow Posts

Fig. 3: Goal-Detection

Here, the two goalposts are cylindrical structures of yellow color. To find these, the method *findContours* is applied on the binary image. This will detect any shapes or contours in the image of the specified color range. To avoid further noise, along with erosion and dilation, contours with radius lower than a certain threshold are filtered out.

3.3 Kick Motion



(a) Alignment (b) Weight Shift (c) Reach Back (d) Kick

Fig. 4: Kick-Motion

The kick motion has been modeled in Choregraphe and it consists of 4 different key frames. First the robot aligns itself and prepares itself for the second key frame in which it does a weight shift in order to lift its right foot. After lifting its foot, the robot is able to execute a kick by swinging the foot back and forth.

3.4 States

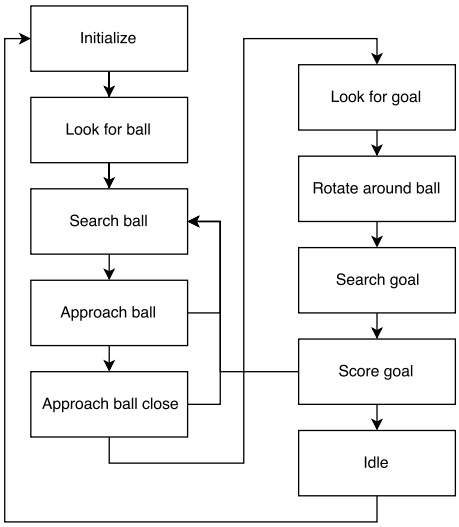


Fig. 5: Basic Flow

Initialize This is the beginning of the series of actions to be carried out by the robot. We implemented an artificial wait time of 100 frames, so that it becomes easier to monitor the robot's actions and movement. Furthermore it becomes possible to test the robot while working without a partner.

Look for Ball In this state, Nao turns its head and looks for the ball in both directions. Now Nao has two cameras positioned vertically on its face. As it rotates its head, it toggles between both cams to detect the ball. Once the ball is detected, the direction of rotation is set to be used in the next step. In case the ball is out of view of the Robot or is not detected, a default direction is set for the robot to rotate towards.

Search Ball First the robot's neck returns to its initial position. Based on the direction of rotation set in the previous step, the robot rotates and tries to detect the ball. Once the ball is detected and the pixel coordinates of the ball are set, it moves on to the next step.

Approach Ball The relative position of the ball influences the rotational movement of the robot while moving towards the ball. This keeps the ball always aligned with the robots center. During all movements, if the robot loses the ball, it will go back to the **Search Ball** state. If the ball is close enough to switch to the bottom camera, it continues with the next state.

Approach Ball Close Once Nao has sufficiently narrowed the distance to the ball, it continues moving towards the ball but slower this time. Again once the ball coordinates cross a certain threshold, the robot stops and moves to the next step. This is to shorten the distance to the ball so that it becomes easier to position itself later for the kick motion.

Look for Goal This is similar to the **Look for Ball** state as it turns it's head and looks for the goal. Once any goal post is detected, the direction of rotation is set and the next step commences.

Search Goal The direction flag from the previous step tells the bot which direction to rotate to. Here the bot starts rotating around the ball till it finds both goalposts. The distance between the ball and Nao had been kept sufficient in the previous so that there is no collision while rotating.

After that, the robot tries to center the ball around the mid section of the goal. The ratio of the x coordinate for the mid-point of the goal and the ball are calculated. Based on this, the robot moves to the left or right till the ball is centered while considering a little offset. Absolute accuracy in any of these steps would possibly make the movements redundant and slow.

Score Goal As soon as the ball is centered, Nao turns its head down for a better view of the ball. Then it moves towards the ball to position itself for the kick motion. The ball is positioned close enough to its right foot for proper contact during the kick. Once the ball's coordinate again crosses the specified threshold, the bot kicks the ball. Right after executing the kick motion the robot changes to the **Idle** state.

Idle In this state the robot waits 15 seconds for the kick motion to complete. To complete the state cycle, the current state is switched to the **Look for ball** state and the whole process starts again.

4 Conclusion & Future Work

In conclusion, the state cycle proved efficient for Nao to carry out the task. The object detection worked reasonably well taking into consideration the surrounding lighting conditions. Moreover throughout all the steps, a certain degree of offset had to be considered for all the movements since perfect accuracy would be near to impossible to gain in the real world.
Presently the robot is only able to detect, move towards and kick the ball towards a goal. To make it play a game of soccer many more such non-trivial tasks would have to be implemented. Obstacle avoidance, collision detection, dribbling or knocking the ball are a few examples among them.

5 References

OpenCV - `opencv.org`
Choregraphe - `doc.aldebaran.com/1-14/software/choregraphe/index.html`