Project RoboCup

April 7, 2019

Contents

1 Goal

Our goal was to let the Nao robot search for a given marker being anywhere around him and to go to this marker and to sit down on it.

2 Approach

The central idea was to spot the marker using opencv on the camera images of the Nao robot. By determining the position of the marker relativ to the middle of the camera image we would guide the nao robot to the marker. In order to make the marker easier to detect, we used opencv's aruco markers. Then when the robot would walk towards the marker until he cant see it anymore, because he was standing on it.

3 Code Architecture

First we had to use opencv's calibrate camera method, so we can use the estimatePoseBoard() method, which estimates the postion the marker. To calibrate we used a chessboard pattern. To get a good calibration we had to have atleast

40 frames where opencv detects the chessboard in diffrent spots of the robots view and varying distances.

The main programm is just 2 simple loops. In the first loop the robot would turn left until the marker is found and then he would turn left or rigth depending on the markers postion until the marker was ahead of him. The second loop would run as long as the marker can still be found, while the robot moves forward. As soon as the marker can not be found, the robot will check an additional time for the marker. If the marker is still being undetected, the robot will assume he is standing on the marker and thatfore completing the program and switching off.

4 What we learned about the Nao

We learned that the modules are running on the nao robot and we are connecting to them via the Naoqi framework in order to make the Nao robot move and get pictures from the camera with our program. During the testing of the robot we learned quite a lot about the cameras as well. We noticed a huge delay loading the camera images via the LAN to the computer. At first we assumed it might be network related, however the delay was not consistent enough to be hardware related. We learned that the cameras are recording in a 1240x840 resolution in a YUV colorspace. As we used a 640x480 resolution and the RGB colorspace in the beginning, we thought changing it more to the native camera settings would speed this up. Adjusting the settings closer to the native camera settings gave roughly a 10% speed boost. When we started getting problems with the movement modules, we assumed this might be network related as well, as most of the time an error would be paired with connection problems. However it turned out the connection problems were merely symptoms and not the reason for our problems. We learned that as the vision module would crash quite often, while the motion module continued to run. So while we could connect to the robot and move it as we liked the camera would return a “Can not connect” error. This empathized with our observations of the loading times of the camera images. While it takes 1-2 seconds to load a picture in the beginning, it will increase rapidly to 15 to 30 seconds or until the vision modul is crashing. We also knew the robot was not in best shape from the beginning, as in rare cases the robot would not be able to execute basic framework movement commands and instead would fall without any obstacles around. This can be seen easy at the start of the program, as the robot would fall in 90% of all program starts performing the first move. While we order the Nao robot to go into a very stable pose in the beginning, the problem continued.

5 Outcome

Starting the program will let the Nao robot turn to the left in 15° steps until the marker is found. While doing this the robot will look up and down to ensure

the marker is not already laying right next to him. When found the robot will adjust its position until the marker is laying right in front of it. Now it will try to move closer to the marker in 15 cm steps, while constantly trying to have the marker right in front of it. At any point the robot would stand on the marker and will no longer be able to detect the marker, as the feets are covering it. Having reached the goal, the robot will go into rest mode.

6 Ways to improve

At the moment we are loading the camera images as close as possible to the native camera settings from the robot via the wired network. As mentioned before this is clearly a bottleneck in the design. One way to speed this up might be to perform the calculations on the robot himself. While the opencv might take longer on the Atom cpu of the Nao robot, it should still provide a speed boost in comparison to loading a raw image file via the network.

The robot is having problems detecting the image around 2 meters and beyond. The main problem is, that the marker lies on a sharp angle and the light is reflected easy on the used paper. While detecting the marker farther away can be improved using machine learning algorithms, the reflecting problem of the paper could be improved by using a non reflecting material. Also we could just use checkboard patterns instead of aruco markers, since the opencv detection method of aruco marker can be quite taxing for the system.

At the moment we are not using the head to turn left and right to look for the marker but instead let the robot rotate until the marker is in the field of vision. This was a design decision as the robot had to adjust its position to the marker anyway. However checking with the head first which direction would be best to turn to, would greatly increase the speed of the robot to find the marker.