Chicken Chicken Chicken

My Penis

January 15, 2014

1 other section

2 other section

3 other section

4 Image Processing

This section's purpose is to explain the mechanisms, algorithms and mathematics behind the processing of images made by the Nao. A brief introduction to the subsections follows before heading into the details.

First the image is preprocessed. In this step the image is converted into a new image consisting only of green, yellow and white pixels. Any background noise is removed.
Second the goalposts get isolated and the location of the foot of each post - if visible - gets determined.
Third the fieldlines get isolated and using Hough Transforms the cornerpoints are extracted.
Lastly the distance gets calculated using the camera angle and height, the resolution of the image and the coordinates of the previously extracted landmarks.

4.1 Image Preprocessing

Before any kind of extraction can be made, the image first needs to be preprocessed. First a random sampling of pixels is made from which the average light intensity is calculated using the luminence from the HSL color-space.
Next the white, green and yellow (goal) pixels are extracted by converting them to the HSL color-space and by using the following thresholds and bounds:

White:
$L_p > 120 + (L_{max}\text{-}120) \times \frac{L_{avg}}{L_{max}}$

Green:
$G_{min} \leq \mathrm{H} \leq G_{max}$

Yellow:
$Y_{min} \leq \mathrm{H} \leq Y_{max}$

Where
L_{max} = maximum luminence (usually 255),
L_p = luminence of pixel,
L_{avg} = previously approximated average luminence,
H = hue of pixel, and
G_{min}, G_{max}, Y_{min}, Y_{max} = lower and upper bounds for the hue of green and yellow.

The value of '120' has experimentally been shown to deliver the best results. The G_{min} and G_{max} are determined at the start of each run by taking a picture of the 'grass'. This image is then scanned for the minimum and maximum hue. G_{min} will be set equal to this minimum hue minus some small number ϵ in order to deal with the occasional over- and underlighted areas on the field. The same is done for G_{max}.
Y_{min} and Y_{max} are determined similarly but without adding/substracting ϵ, because (in the case of this project) there is only one goal and thus the difference in lighting is negligible.
At this point the original image(figure 4.1) now looks something like figure 4.2.

Figure 4.1: Raw image taken by Nao

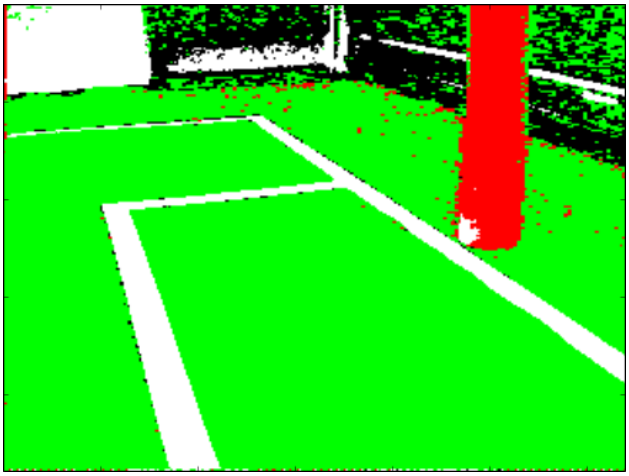

Figure 4.2: Image after color filter

Before continuing preprocessing, the goalposts need to be extracted first.

4.2 Goalpost Extraction

The only parts of the goal that are of interest for this project are the locations of each foot of the posts, because the localisation algorithms use a 2D map. To find these the image is scanned from bottom to top. If a yellow (in this case red because of optimisation) pixel is spotted the algorithm looks a few pixels down to see if there are some green pixels. If there are sufficient green pixels the algorithm continues, if there aren't the algorithm stops scanning this column and proceeds to the next one. This heuristic was added to avoid getting parts of the goal's mast or some random noise in the background.
When the algorithm has decided there are enough green pixels under a yellow one it continues to climb the vertical axis and count the amount of adjacent yellow pixels. If this counter surpasses a certain predefined threshold (somewhere between 10 and 20 percent of the height of the image) the first-encountered yellow pixel is now saved in a list of goalpost-coordinates and the algorithm continues to the next column.
After the entire image is scanned the list of goalpost-coordinates is clustered and averaged to prevent multiple landmarks at one goalpost.

4.3 Fieldline Cornerpoint Extraction

Before any fieldlines can be extracted the background noise needs to be removed. The details of this operation will not be explained here, but the general idea is to scan the color-filtered image from top to bottom for the color green. Once a safe amount of green pixels in a row has been encountered, 'erase' all the pixels above this point and continue to the next column [1]. To finish up we will remove any remaining green and yellow pixels because they aren't needed anymore. At this point the image will look something like figure 4.3.

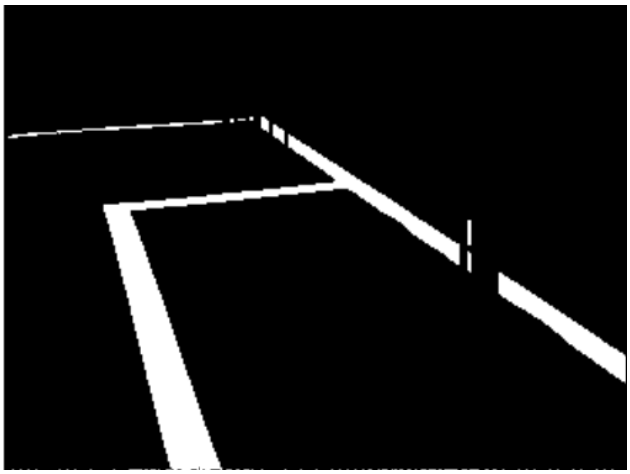

Figure 4.3: Background noise removed

There are quite some ways of obtaining the - in this example - three landmarks, but one in particular seems to be favored in the world of robotics [2][3]: OpenCV's build-in Canny and Hough Transform methods. To illustrate Canny's use see figure 4.4.

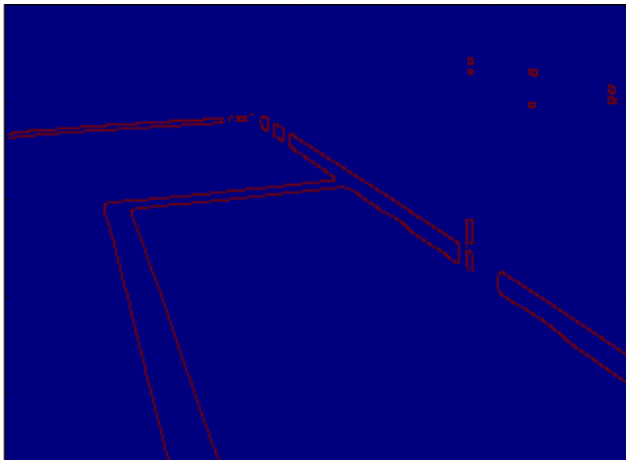

Figure 4.4: Image after applying Canny

After using the Canny method the Hough Transform is applied to the newly obtained image resulting in figure 4.5.

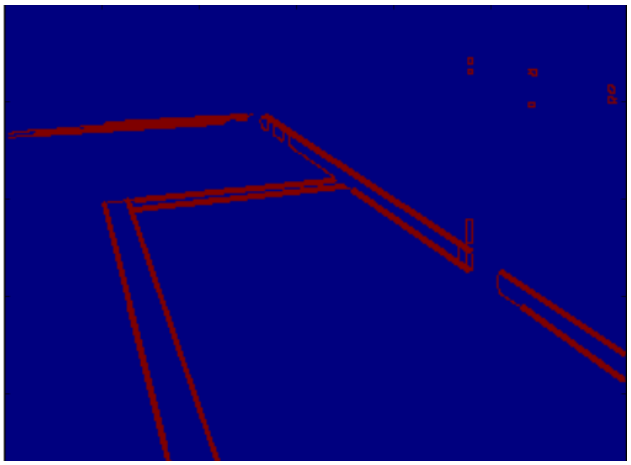

Figure 4.5: Image after applying Hough Transform

The thicker lines are the lines generated by the Hough Transform method. The only thing left to do is finding the intersections of these lines and deleting intersections which are relatively close to each other in order to avoid double landmarks.

Now that all the landmarks have been extracted the distance to these can be calculated.

4.4 Distance Calculation

References

[1] Amogh Gudi, Patrick de Kok, GeorgiosK. Methenitis, Nikolaas Steenbergen, "Visual Goal Detection for the RoboCup Standard Platform League" *X WORKSHOP DE AGENTES FISICOS*, sept 2009.

[2] Jose M. Canas, Domenec Puig, Eduardo Perdices and Tomas Gonzalez, "Feature Detection and Localization for the RoboCup Soccer SPL" *University of Amsterdan*, feb 5, 2013.

[3] Lukasz Przytula, "On Simulation of NAO Soccer Robots in Webots: A Preliminary Report" *Department of Mathematics and Computer Science University of Warmia and Mazury*, sept 2011