NAO Localization

Amogh Gudi, Georgios K. Methenitis, Nikolaas Steenbergen, Patrick de Kok

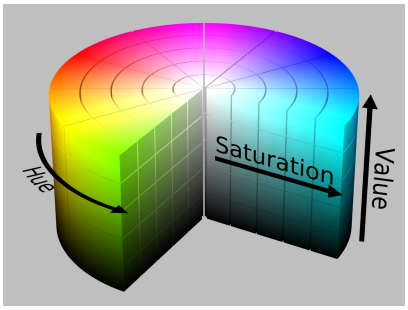
University of Amsterdam

In this Project we propose a basic code for the Robo soccer Standard league Platform for the Dutch national team. We focused on the problem of visual feature recognition and the localization of the robot in a tournament setup. This project contains a detailed description of our implementation together with a conclusion and a brief evaluation. Since this report only covers feature detection and Localization we give an outlook on what parts are to be implemented in addition to make the robots play soccer.

1 Feature Extraction

Visual object recognition is a key ability for autonomous robotic agents especially in dynamic and partially observable environments. A reliable landmark detection process is really crucial for achieving self-localization, which can be easily considered as the stepping stone for having a functional robotic soccer team. In this section, we describe the whole procedure of processing input images from the NAO's camera, outputting features that are informative to be used by the core of our localization scheme. Using a variety of image processing techniques, we successfully detect field landmarks. In general, we have to deal with noisy images, which not only contain the field region but also background noise and useless information which comes from above the horizon of the field view. Furthermore, we had to deal with lighting conditions which may vary significantly. There also real-time constraints which had to be taken into consideration during our algorithmic implementation in order for the whole procedure to be able to execute in real time.

Figure 1: *HSV colorspace. (Image source)*



1.1 Choosing colorspace

The first challenge we came along was to choose the best colorspace representation in order to overcome the variations in lighting conditions. The dataset we had consisted from rgb images. Unfortunately, it was unfeasible to use the default rgb colorspace for most applications due to noise, shadows, etc. We also experimented with normalized rgb colorspace. Theoretically, normalized rgb would have given us the pure chromaticity values of each object in our field of view, overcoming like this possible shadows and lighting conditions. In the contrary, normalized rgb did not help us achieving a light-invariant color representation, resulting into false positives in color segmentation. HSV was the solution to our problem. HSV is a cylindrical colorspace representation, which contains information about hue, saturation, and value. We found hue to be really informative in order to distinguish easily between the colors. Hue define the pure chromaticity of the color and it is independent of the lighting conditions. In figure 1,

we illustrate this cylindrical color representation, we can see that the hue is represented by the angular dimension in the vertical axis of the cylinder.

1.2 Basic Pipeline

Having described the first basic step of our approach, choosing the proper colorspace for our application, it is time to go deeper into the feature extraction procedure. The first step in this procedure is the image input from the NAO's camera in HSV format. As we said before, images not only contain the important region of the field in which we are interested in, but also some background information which is useless in order to detect field features. This background usually extends above the field horizon and it may become really disturbing in the extraction process. This is the reason why background removal is the first step in our feature extraction scheme. Once we have removed, our input image contains information which has to be extracted. Field lines, goals, ball, and other NAO robots are the features located into the field. In this project, we only took into consideration lines, and the goals which are static landmarks, as ball, and other robots are moving and we cannot depend on them in order to self-localize. So, we use information from the HSV values to binarize the image, first in respect to the goals and then in respect to the lines. As we said in the introduction part of this paper, field is colored green, both goals are yellow, and lines are white. Next step of the feature extraction is the the goal detection, making use of the horizontal and vertical histograms of yellow values. Line detection is followed in order to find a good estimation about the lines detected in our field of view. Having these lines' estimations, we can detect feature on them, using geometric properties. Last step in this process' pipeline is to output all these detected features to the localization core process. We can now enumerate all steps in these pipeline, these are:

1. Input HSV image from camera
2. Background removal
3. Image binarization
4. Goal detection
5. Line detection
6. Line feature detection
7. Output features

Table 1: *Threshold values used for color segmentation.*

	Threshold		
Color	**Hue**	**Saturation**	**Value**
Green	$20 \sim 37$	$100 \sim 255$	$100 \sim 255$
Yellow	$38 \sim 75$	$50 \sim 255$	$50 \sim 255$
White	$0 \sim 255$	$0 \sim 60$	$200 \sim 255$

2 Image format

NAO's camera can output images in different colorspaces representations. The images from the dataset we had, were in RGB format, and they had QVGA resolution (320×240). We wanted to keep a low resolution in order to keep the time complexity of our algorithm feasible for real-time execution. **Patrick can add some things here as he experimented better with NAO's camera.**

3 Background Removal

Background removal defines the task of detecting the field's borders in order to exclude uninformative regions in the processing image, saving this way computational cost and helping to subtract only the useful part of the image, which is the one, where we can detect all the features we are interested in. This can be done by a vertical scan of the image and detecting the first green pixel in each column. This method can work efficiently but it is not robust in many cases, as green pixels can be found above the field's horizon as well. Following the same principle in our approach, we consider as background every region upon a column before a considerable amount of green pixels, and not just one, which can lead us to faulty decisions about the horizon. We start in each column considering the pixel in the first row as background. Then, during scanning each column, we stop when we find many green continuous pixels and assign as horizon row in this column the first of this sequence of green pixels. In this process, we are only interested in green pixels. As you can see in table 1, green pixels are considered as those which have HSV values around these thresholds.