Localization for Robot Soccer

Research Project Master AI, Final report

Diener, A. Fuss, F. Polzin, R.

January 25, 2017

Abstract

This report discusses the background and approach taken for the research project Localization for Robot Soccer, the goal of which is to create a NAO robot module capable of providing locational information within a RoboCup environment. The concepts used to derive the locational information are exclusively focused on the use of the robot's video feed and concern object detection techniques.

Contents

1 Introduction **3**
1.1 Requirement analysis . . . 3
1.2 Background . . . 4
1.3 Goals and research questions . . . 4

2 Possible approaches and solutions **4**
2.1 Feature Detection . . . 4
2.2 OpenCV approach . . . 6

3 Prototype **6**
3.1 Ground Detection . . . 7
3.1.1 Color Separation . . . 7
3.1.2 Key Point Detection . . . 9
3.2 Edge Clustering . . . 10
3.2.1 Edge detection . . . 11
3.2.2 Cluster definition . . . 11
3.2.3 Cluster classification . . . 12
3.3 Goals Detection . . . 12

4 Experiments **13**
4.1 Feature Detection . . . 13
4.2 Ground Detection . . . 15
4.3 Goals Detection . . . 15

5 Conclusion **15**

6 Appendix **17**

A How to use the implementation **17**
A.1 Ground detection . . . 17
A.1.1 Python version . . . 17
A.1.2 Java version . . . 18
A.2 Goal detection based on Edge clustering . . . 18

B Options for improving the code **19**
B.1 Ground Detection . . . 19
B.2 Edge detection and clustering . . . 19

C List of recommended improvements **21**

D Images of Experiments **22**
D.1 Ground Detection . . . 23

1 Introduction

The research project which is discussed in this report pursues the goal to create a module capable of enabling the extraction of reliable locational information for a NAO robot in a RoboCup environment. Specifically, the module is developed with object detection techniques in mind, specifically designed to extract the core attributes of a RoboCup field from visual information.

In section 1.2 background on B-Human and its localization approach is given. Section 1.3 goes on to elaborate upon the goals of this project and the localization module it represents. In section 2 possible approaches are discussed, which were considered but abandoned for practical reasons. Section 3 talks about approaches which have been implemented and been explored in depth.

1.1 Requirement analysis

The localization module provides the answer to the question of where and how the robot is currently located on the soccer field. There are three essential variables, the robot's position in both lateral dimensions x and y as well as its angle, which make up its pose. For deriving the robot's pose there are two essential approaches: Using a history of actions or using some kind of processing of landmarks registered by the robot's sensors. The first approach has the downside that the perceived and actual pose can diverge quickly as a state change provoked by an action is not fully deterministic in a real world environment. The second approach has the downside of being more computationally expensive, slower and inherently reliant on outside factors. For the localization the robot can make use of domain knowledge of the soccer field, in particular the lines and how they relate to one another.
The approach discussed in this report is of the second kind. The robot's video feed is used to generate images which in turn are used as the input for an object detection algorithm which can be used as a basis for deriving the robot's location.

The NAO robot is a programmable humanoid robot designed by SoftBank Robotics Corp., a French company offering robot related products and services. The NAO robot was the first humanoid robot designed by SoftBank Robotic Corp. in 2006. After the NAO they released two more humanoid robots called ROMEO and Pepper. ROMEO is designed for helping elderly people in their daily life and Pepper is the first robot actually recognizing human emotions and adapting their behavior accordingly. The NAO has many different sensors to detect its environment and is able to move it arms, legs and its head. It has two sonar range finders to detect obstacles at a distance with the maximum of 70cm. There are two 920p cameras at the robots forehead and its mouth to take pictures or record videos. Additionally, it can listen to the environment with four microphones. Its eyes are infrared emitters and receivers.

1.2 Background

B-Human is a RoboCup team founded in 2006, representing a joint effort of the University of Bremen and the German Research Center of Artificial Intelligence. The team has consistently won the German/European Open and has managed to become world champion 4 times. To the team's knowledge 21 different teams either base their systems on B-Human's platform or use at least parts of it in their own.

Previous iterations of the B-Human platform made heavy use of visual identification of goals in their self-localization algorithms. This was possible due to the smaller field size and unique colouring of the goal posts defined by earlier versions of the RoboCup rules. Using this basic system supplemented by a detector for the white goals on a modern RoboCup field has turned out to require too much computation time and to not be reliably depending in the specific environment.
The current version of B-Human does not consider goals for its localization. Instead it uses a module based on measurements referred to as 'Field Features'. Those measurements are derived from basic field elements observed by the robot. They include the penalty area, center circle, field corners, center corners and the goal frame on the floor.

1.3 Goals and research questions

The goal of this project will be specified in terms of a final product and a number of research questions.

Final Product: A module for localization of NAO robots on soccer field, which supposed to be better then currently existing module of B-Human platform. The following parameters should be considered: quality and robustness of the code as long as computational efficiency of the whole program.

The following research questions are formulated:

- Which techniques can be used for localization?
- What are the problems of using such techniques on NAO robots for soccer?
- How can the existing techniques be adapted to work properly?
- What assumptions need to be made?

2 Possible approaches and solutions

2.1 Feature Detection

Feature Detection refers to a concept of computer vision that allows to detect whether a certain set of feature is given in an image. Such a set of features

often consists of points, edges or colors with a certain relation to each other. A white sheet of paper could be described as a white rectangle with a certain aspect ratio. The task gets increasingly more complicated if the sheet of paper is rotated, scaled or skewed. A feature detection algorithm can extract specific feature points and then use those to provide good detection even if the sheet of paper is transformed.

During research one concept was to detect important aspects of the playing field using such an algorithm. For once the goals could be detected, or the center circle or the penalty area. To evaluate whether such a solution was viable, especially in regards to computational complexity, different algorithm were investigated. The algorithms evaluated are ORB (Oriented FAST and Rotated BRIEF), SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features).

SIFT is an algorithm from 1999 that often uses high contrast regions that are invariant to scale, noise or illumination [12]. The algorithm is often quoted in feature detection and is supposed to offer good results at different scaling, orientation or affine transformations. The library that we use in the algorithm offers support for SIFT feature detection.

SURF is partly inspired by SIFT and was developed in 2006. It is based on the same basic ideas as SIFT, but alters details in the different steps [11]. The algorithm is supposed to be several times faster than SIFT and more robust to certain image transformations. In addition to the SIFT algorithm the SURF algorithm is also supported by the library used in our implementation.

ORB was developed in 2011 [10] as an alternative to SIFT and aims to be faster and more efficient. The release paper states good results and the algorithm is available in the library used.

Figure 1 illustrates the issues we encountered when working with either of the above mentioned algorithms. While most feature detectors detected a high amount of features and often could match some of them back to the object the detection in general was very unreliable. Most of the time no significant amount of matches were found and only with a setup nearly identical to the original learning image the object was detected with sufficient confidence. As not one of the algorithms produced proper results and some are considered standard in computer vision, the lack of good results was certainly based on issues applying the algorithm the library provided. As we decided not to invest more than a week we did not find the reasons for the lack of good results. There were still differences between the algorithms in terms of speed and robustness, but none provided the precision and consistency we required.

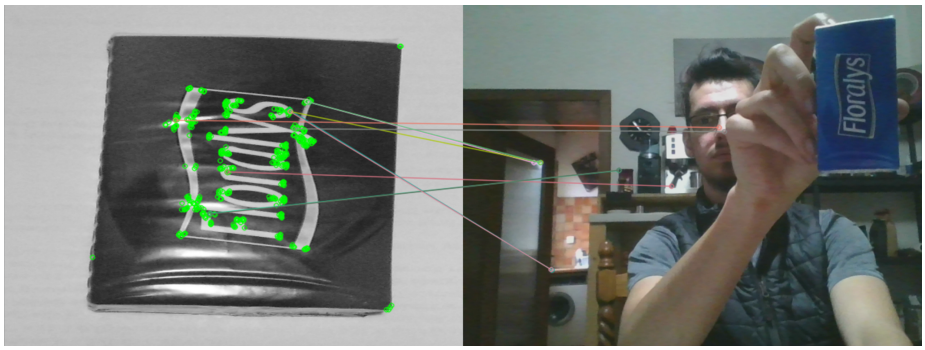


Figure 1: This figure illustrates the issues encountered using existing feature detection algorithms. On the left the learning pictures with some of the key points determined by SIFT can be seen. The right side shows the current image from the notebooks camera. Lines are connecting the detected key points in both images.

2.2 OpenCV approach

The OpenCV tutorial "Real Time pose estimation of a textured object" [3] proposes detection of objects based on their 2D image and 3D model. This approach offers a solution to get an orientation of detected objects.
For purposes of Robot localization on a soccer field, this approach could be useful by application of a 3D model of the soccer field and getting its current orientation based on available features. Proposed 2D image features are only applicable for wholly visible objects, so in our case, some other features can be used instead. Such features could be based on Ground detection and Goal detection methods, discussed below in respective sections. After getting the orientation and position of the soccer field underneath the robot, the position on the robot itself could be easily calculated using given data. The whole algorithm consists 5 steps:

- Create 3D model of the playfield
- Extract features from the visible image
- Match scene descriptors with model descriptors using Flann matcher
- Pose estimation using PnP + Ransac
- Linear Kalman Filter for bad poses rejection

3 Prototype

This section contains information on the techniques we implemented in the prototype.

During the research and evaluation phase we realized that generic feature detection algorithms like ORB (Oriented FAST and Rotated BRIEF), SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features) did not work in our particular field of application. Therefore the focus was shifted towards a more custom-tailored solution.
In the B-Human team a combination of color separation, field and goal detection is currently used [14]. A couple of Bachelor theses from the University of Tartu, Estonia, pursue a similar approach [13]. We analyzed the most relevant techniques described in these theses - Ground Detection and Goal Detection - and implemented them. The following sections will describe those techniques in more detail and elaborate on our results testing them.

3.1 Ground Detection

A solid guideline for object detection and localization is the green area of the soccer field. Being able to determine whether an object is positioned on the field allows for a better separation of relevant object and objects that the algorithm detects, but that are not part of the game. For example with the algorithms we implemented goal posts will be detected as two vertical lines connected by a horizontal line. Such constructions could be detected in many different parts of the pictures for example in door frames or tables. All these false positives could be eliminated if the additional requirement of 'being on the field' is introduced.

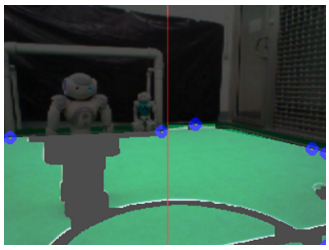
Detecting the ground area of the soccer field is an important step for the self localization algorithm we decided to implement. The algorithm proposed by Tarvas [13] first uses color separation to get the green parts of the image. This is in detail described in section 3.1.1 In the next step erosion and dilation are used to reduce the noise in the detected green regions. The last step of the algorithm consists of the detection of different key points in the image. Those are then connected to form an 8-edge polygon that resembles the detected field area. This step is described in section 3.1.2 The separate steps can be seen in Figure 2.

3.1.1 Color Separation

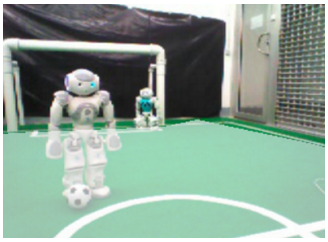
As of 2016 the rules for the RoboCup Standard Platform League state that the carpet that marks the playing field is green. While the exact color of the green is not specified, it 'should not be too dark' [5]. This can be used in localization to more easily detect landmarks that are relevant to the game like goals, the middle circle or specific lines.
In the bachelor thesis of Anastasia Bolotnikova color calibration techniques for the production of reliable green values are presented [9]. We did not find the time to implement them in our algorithm and therefore assume that the colors in the received image mostly resemble the actual colors.

(a) Input Image

(b) Noise Reduced Detected Green



(c) Key Points



(d) Resulting Region

Figure 2: Results of the ground detection algorithm.

A technique that can be used to separate for example the color green from an image is thresholding. To separate the green parts of the image a different color space is used. While in the original image the color of a pixel is described in BGR (Blue, Green, Red) the separation algorithm transforms the image to HSV (Hue, Saturation, Value). In BGR a pixel is described by one value for blue, red and green respectively. The values range from 0 to 255 and a pixel with blue = 255, red = 0 and green = 0 would be a pure blue. In comparison to BGR in HSV no primary colors are used to describe the pixel. Instead hue is used, which describes the shade of the color. Saturation describes the intensity of the color, where 0 is white and 255 is the maximum intensity. To describe the lightness or darkness of a color value is used, where 0 describes a black and 255 a very bright color. Figure 3 visualizes the composition of a color in HSV.

The advantage of HSV over BGR in color separation is the independence of the color's hue from it's saturation and value. In theory a green in HSV representation will be green regardless of different lightning conditions. Even if in practise the results differ they still improve over BGR color separation.

Based on the HSV color space a color separation is achieved by thresholding every pixel of the image against the shades of green we consider relevant. In HSV representation greens range from 81 to 140 on the hue circle. A pure BGR green (blue = 0, green = 255, red = 0) converts to a hue of 120. The image library used by the algorithm halves the hue value to fit into the 0-255 range. Therefore if a hue value of 120 is used you will find 120/2 = 60 in the code.
Based on this 'central green' a possible deviation is defined. In the final algorithm this deviation was 60 (30 in code). A hue is considered by the algorithm if it's in the range of 60 to 180 with value and saturation between 100 and 255, so that dark and white regions are not considered green. As most of the time the color calibration will not be perfect the defined range is slightly larger then the representation of green colors in HSV. Manual testing showed good results with these parameters, but automatic fine tuning for different lightning conditions could help greatly reduce the detection range and thus noise in the detection.

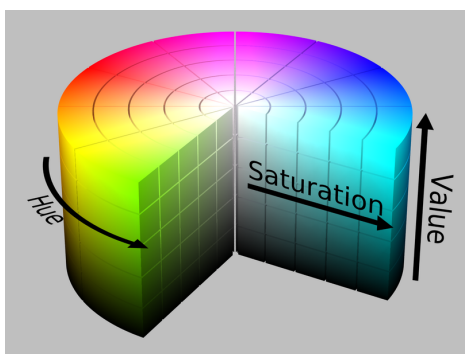


Figure 3: Visualization of the HSV color space.
Photo taken from Wikipedia's HSL and HSV article.

3.1.2 Key Point Detection

While the application of color separation already provides a good detection for the field it suffers from many issues. The returned area, as seen in Figure 2b, does not handle occlusions at all. Many areas will be regarded as 'not in the field' that way and the algorithms based on the ground detection would fail.

Karl Tarvas proposes a geometric approach that constructs a polygon to fit the field in his bachelor thesis [13]. The implemented algorithm is based on this geometric approach and utilizes the same key points. In Figure 4 the different key points can be seen.

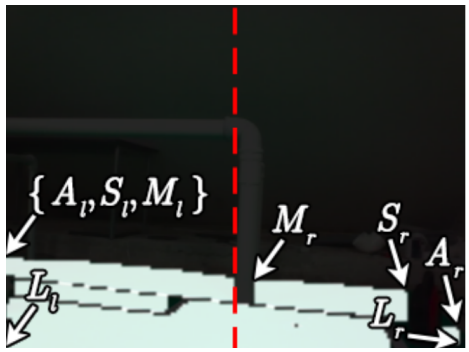


Figure 4: This figure visualizes the different key points that are detected to create the final shape of the ground. Image is taken from Tarvas' bachelor thesis **??**.

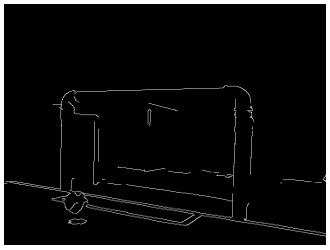
L_l and L_r are first detected and describe the lowest green pixel on the left and right sides. Based on those two points A_l and A_r are calculate. They correspond to the highest point on the side, roughly above L_l and L_r. The next points defined are S_l and S_r. The used implementation diverges from the formula in the thesis at this point. Probably a mistake was made in the thesis at this point, as with the given formula did not provide the results we see in figure 4. In the implementation we use S_l and S_r evaluate to points that try to balance being as far away from the middle of the green section as possible, while still being as high up as possible. Finally M_l and M_r are the highest absolute points on the right and left part of the detected green section. These points are combined into a polygon that is used to describe the field. Most of the time the resulting area is pretty precise and eliminates most of the occlusion. The limits are discussed in section 4 and improvements are proposed in Appendix B.

3.2 Edge Clustering

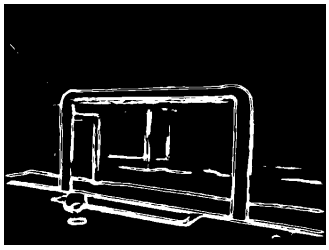
Detection of directed edges can give a lot information about the current position on the field. As example we could use it for detection of the goals by looking for two vertical planks and a horizontal one, bound together, with the bottom on the soccer field. Furthermore by knowing relative size, angle of top bar and position of goals we could estimate current position of the robot.
Edge detection approaches are usually color invariant, so color changes on soccer field will not affect the success of the localization algorithm.

The clustering process consists following steps, according to algorithm proposed by [13]: edge detection, cluster definition, cluster classification. All details on every step are available in respective subsections.

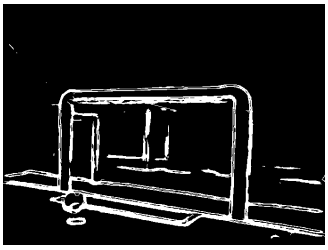
(a) Input image



(b) Canny method



(c) Random Forests method



(d) Combination

Figure 5: Results of Canny and Random Forests edge detection methods

3.2.1 Edge detection

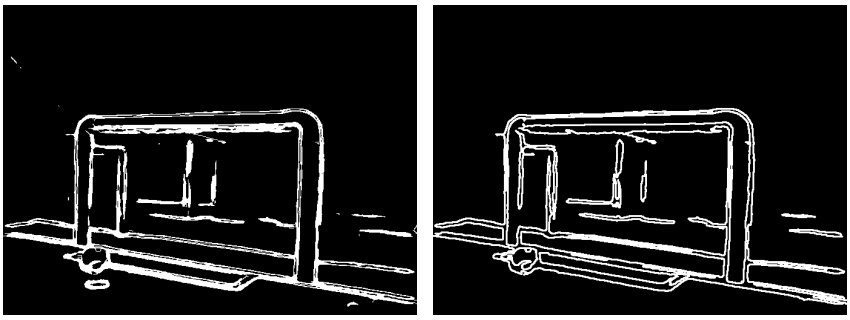
For detection of edges on the image we used two different edge extraction methods to be sure, that important objects to detect will not be missed.
Karl Tarvas [13] gives a good explanation on why Canny and Random Forests edge detectors are ggod methods to choose, so we initially tried to use the same techniques. Our tests for goal detection (Section 3.3) returns good results, so we stayed on using them. Threshold are really case dependent, but for figures on the paper we used next values:

Both methods are compared on figure 5.

3.2.2 Cluster definition

The clustering is working on two levels. First level is distinction of the edge points, according to underlying gradient direction on the image. Second level is

separation of points for 3 types of gradient direction of clusters, using maximal distance between points within one cluster.
Additionally at the beginning we apply a little optimization, by extracting only 8-connected points from the edges. Those points describe contours of the edges and allow to reduces the number of points to be analyzed for clustering, which drastically decreases the calculation time.
All steps are represented on figures 6 and 7.



(a) Original edges (b) 8-connected points / Contours

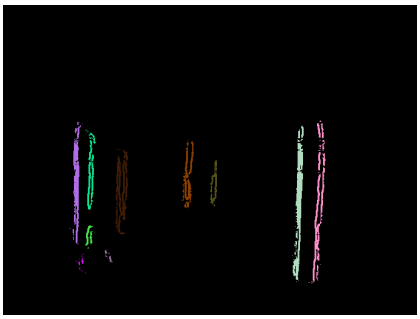
Figure 6: Image before and after contours extraction

3.2.3 Cluster classification

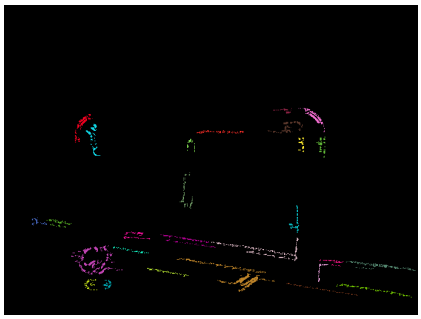
Every cluster is labeled according to the underlying gradient direction. In order to get proper overview of position of the cluster, we select two points, with largest distance within the cluster and one which represents an arithmetic mean of two others. Then both side points are checked, whether they are on the soccer field or not.
Additionally, we also save a distance the side points to know a length of the oriented cluster.

3.3 Goals Detection

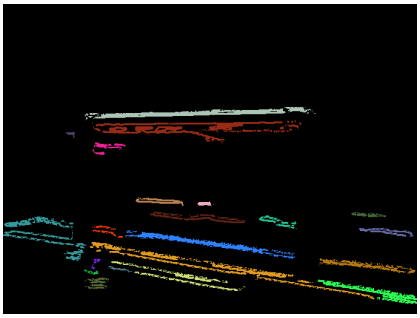
Using the clustering results, described in section 3.2 in combination with section 3.1 we can detect goals in two steps.
We know, that goals are represented as two vertical lines connected by a horizontal one, so the steps are make in order to detect such structure.
First step is to find a pair of vertical clusters with bottom side on the soccer field and top side outside the field and enough distance between them. In order to define the proper minimal and maximal distance between the clusters and also their minimal and maximal length, proper testing on desired environment is needed. Second step is to find a non-vertical cluster, which connects the top sides of clusters from the first step. Those two steps will be repeated until all candidates are found. Both steps are represented on figure 8;



(a) Vertical clusters



(b) Diagonal clusters



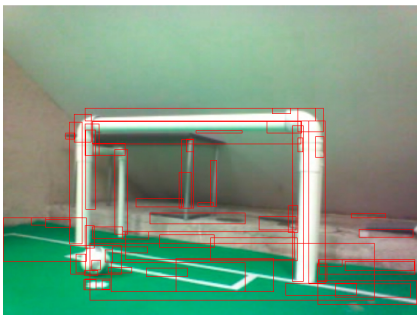
(c) Horizontal clusters

Figure 7: 3 types of clusters, separated by maximal 5 pixels distance between points in cluster. Every cluster is represented with different color. The separation can become invisible on a gray-scale print.

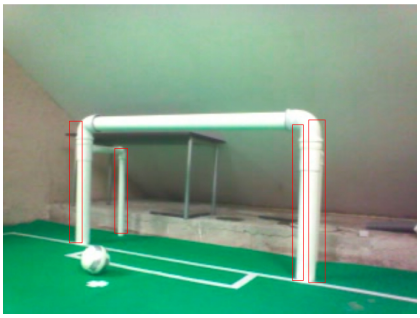
4 Experiments

4.1 Feature Detection

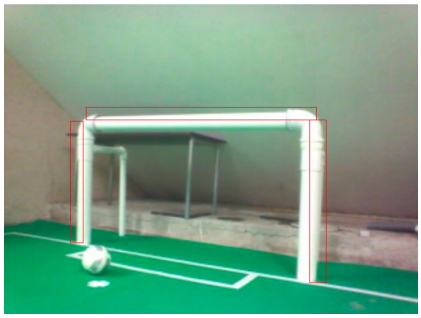
During the initial research phase different feature detection algorithms have been tested. Section 2.1 describes these in detail. The experiments on the algorithms consisted of a Python script that would use the OpenCV library to train the corresponding algorithm on an image of an object. After training a live video from the laptops camera would be analyzed and the algorithm would try to detect the learned object in the video stream.
The advantage of such as setup is the ability to easily rotate, scale and translate the object that is to be detected and to determine the computational requirements and real time capability of the algorithm. Different environments have been simulated using two different backgrounds, a cluttered view with many details and pictures in the back and a white wall, as well as different lighting conditions. The light was modified by using actual sun light from a window or different colored lights from LED bulbs capable to reproduce different colors.



(a) All clusters



(b) Vertical clusters with one side on the soccer field



(c) Goals candidate

Figure 8: Representation of the goal detection process.

Another modification that was made to broaden the covered scenarios was the use of different objects to track. A small glass bottle, a sport foot-bag (hacky-sack) and a package of handkerchiefs have been used.

None of the algorithms yielded satisfying results. Only when exactly recreating the position of object on the training image the algorithm was confident enough in it's results. Illumination changes and a cluttered background had little influence on the algorithm, but the general detection rate was very poor. Based on the algorithms popularity and widespread use it was probably an issue with our usage and implementation of the algorithm, but within the declared time frame no acceptable explanation was found. To sum it up general feature detection was regarded as too complex to fine tune to the specific problem and the focus was shifted towards a more custom tailored solution for RoboCup soccer. A solution based on a modified SURF, SIFT or ORB might still produce better results, but the exact details of these algorithms are complex and understanding and modifying them would probably go beyond the scope of this research project.

4.2 Ground Detection

The ground detection algorithm from Tarvas' bachelor thesis has been thoroughly tested. For this purpose several recordings of a RoboCup field have been made from the perspective of a NAO robot. Those videos show the field under artificial lighting conditions, with yellow and white walls as well as other distracting wall mounted objects, both in relative stand still and in motion. For testing of the algorithm individual frames have been extracted from these videos at distinct points, both in motion and in relative stand still.
A total of twelve frames were tested under these conditions. Eleven of the tested frames showed the desired result, of the ground detection projecting properly onto the field. However one of the tested frames resulted in a projection, which did only cover part of the field visible.
In addition to this, images from different RoboCup SPL soccer games that were available online have been tested. The results for those images can be seen in the Appendix 9. They offer different resolutions as well as different lighting conditions.
To conclude the experiments conducted on the ground detection a satisfying result was achieved most of the time. The resulting polygon offered a good approximation of the playing field and only failed to do so if the first step of color thresholding did not result in a good separation. This critical part of the algorithm could be improved using the techniques described in Bolotnikova's bachelor thesis, which focuses on automatic color calibration [9].

4.3 Goals Detection

The goals detection algorithm, described in section Goals Detection was implemented shortly before this paper was written, therefore it was not properly tested yet. Although positive results are shown on Figure 8
Success of this algorithm depends on proper ground detection, as goals need to have their bottom on soccer field and their top above the field. Edge detection needs to have the right setting of parameters in order to detect vertical and horizontal parts of goals, distinct from background objects.

5 Conclusion

The product of the research project is a way of feature detection based on RoboCup specific image input. Specifically, Ground Detection, Edge Clustering and Goal Detection algorithms have been implemented and tested. While Ground Detection provides a reliable way of determining the space of the RoboCup field, Edge Clustering provides a means of detecting lines in the environment. With the algorithm for Goal Detection this means that the most important features of a RoboCup soccer field can be identified, located and used for the purposes of localization.

In the process of creating the algorithms to use as the basis for localization certain assumptions had to be made.
The video used as the basis for the algorithm is assumed to be recorded in sub-optimal quality in terms of lighting and colour calibration. Furthermore the assumption is that the algorithm will not be run every frame but rather in a relatively infrequent interval of up to a couple of seconds, which allows for more lea-way in terms of performance.
The field is assumed to be coloured according to the specifications in the RoboCup rules, and the assumption is made that there will not be any big clusters of green colour similar to the field colour easily visible for the NAOs.

For the purposes of the project three techniques have been implemented and explored to aid the purpose of localization. Those being Ground Detection, Edge Detection and Goal Detection. All of these algorithms provide feature detection of different aspects relevant to building a model of the field which in turn can be used for the purposes of localization.

As far as existing techniques are concerned, ORB, SIFT and SURF have been tested for the purposes of this project. However, none of these algorithms have yielded any positive results, so they have been abandoned in favor of more specific approaches.

Algorithms ORB, SIFT and SURF have not yielded any positive results. As the algorithms are considered standard in the field of computer vision this is most certainly an issue with the usage of the library, but due to time constraints they have not been evaluated for more than a week.

6 Appendix

A How to use the implementation

The implementation consists of two implementations: one of Ground detection in Python and Java, second of Goal detection, based on edge clustering, written in C++ and using Java library for Ground detection.

A.1 Ground detection

The ground detection algorithm has been written in both Python and Java. This was necessary, since a Java library was needed to be integrated for the goal detection algorithm more easily. Both implementations make heavy use of OpenCV.

A.1.1 Python version

The python code was developed on a Linux OS, therefore following explanations are tailored to a Linux but should work equally under any other OS. OpenCV 3.1.0 is used to work with images and the code was written in Python 2 but should work in Python 3 with little to no modifications.

A single file, 'ground_detection.py', contains the code. It implements all step of the ground detection algorithm from Tarvas' bachelor thesis. The file can be used on it's own to analyse and overlay images or videos or can be used as a library to analyse images and check whether certain points are considered to be on the field. To use the file to analyse video or image files the corresponding call has to be added to the main guard and the image or video file in the called method needs to be adjusted. The most important methods are analyse_image which gets an OpenCV image passed and stores the ground area internally and check_point which receives two coordinates and returns whether the point is considered in the field. To use the python file to analyse and overlay a picture the test_image file can be used and test_video can be used to do the same with a video. For working with a more low level method the tarvas_geometric method does all the heavy lifting. It receives an OpenCV image and the visualize flag. If the flag is set to true the original image with the points and area will be displayed. In addition the method always returns this overlayed image as well as a NumPy array that contains a Boolean entry for every pixel, stating whether it is considered to be part of the image.

Different aspects of the algorithm can be fine tuned in the code. The important parts are highlighted through comments. Furthermore the code tries to ensure consistent naming of variables in regards to Tarvas' bachelor thesis. Some seemingly random values occur in the code, especially in regards to OpenCV functionality. A comment states if they are based on finding in the thesis. When

those values deviate from the values in the paper the modified version yielded better results in our concrete case.

A.1.2 Java version

The implementation of the Java version is similar to the Python version. Since the algorithm is based on OpenCV functionality, the Java OpenCV library has to be included.
To use the algorithm properly, an image has to be converted into an OpenCV Mat and then used to call the method 'detect', which in turn will return 4 Mats. The first of these Mats contains the actual information of ground that has been detected, the second the Mask that contains the information of green on the image, the third the image multiplied with the mask and the fourth is a pure debug image to show the generated ground polygon overlay ed to the original image.
The utility methods 'upload' store these four Mats of the given image statically, so that the method 'inPoly' can be used to check whether the given position lies within the ground in the original image.

A.2 Goal detection based on Edge clustering

The implementation was done under Linux OS, so following explanations are working only for Linux, but could also be ported to Windows OS. For the edge detection purposes, this implementation uses OpenCV 3.2.0. To compile everything properly, the include path need to be set to OpenCV include folder (usually /usr/local/include) and link to all OpenCV libraries (option "-ggdb 'pkg-config –cflags –libs opencv'" for c++) including additional library 'ximageproc'. Also, because of binding to Java, the include path need to be extended with path to java sdk of version 1.8 and link the library 'libjvm.so'. After a proper compilation, you need to have the precompiled Java class from section A.1.2 and Java OpenCV library in main folder of c++ project. Those files are included in archive, that comes together with that report, but tested only for Ubuntu 16.10 (may also work for version 16.04).

To run the program you need to pass a name of image file or video file (relative or absolute) with respective option ('-i' for image and '-v' for a video) to the program by start. (e.g. "main -i goals.jpg").

Description of all the files can be found in file 'Readme.md'

B Options for improving the code

The source code is fully documented and a readme file is available so adding changes should be straightforward.

B.1 Ground Detection

While the general structure of the code is straight forward and coherent with the techniques described in Tarvas' bachelor thesis different techniques could be used to increase the robustness and speed.

- Improve the color thresholding by implementing automated color calibration based on Bolotnikova's [9] thesis.
- With the current implementation A_l and A_r or only detected in a straight line above L_r and L_l, they should be allowed to deviate a little to the left or right.
- The speed can be optimized by trying to handle different calculations at a single iteration over the image. For example computing the different key points in a single loop.
- Speed can also be increased using a compiled instead of a interpreted language
- Test the parameters and determine good defaults for a wide set of different environments
- Group the adjustable parameters in a separate module
- Add a command line option to load images or videos from command line

B.2 Edge detection and clustering

Current implementation needs multiple seconds in order to do clustering on 640x480 image.
It should be possible to make the code more than 10 times faster by applying an optimization on the code.
Here is the list of things, that could be optimized:

- Properly learned model for random forest edge detection, could improve the speed of recognition and also do better edge detection on important parts of the image (like edges of goalposts).
- Setting of different parameters to contour detection function also affect the speed of computation. Quickest method of contour recognition is 'CV_CHAIN_APPROX_SIMPLE', which concentrates only on largest parts of an edges image.

- Looking for two furthest points in a cluster can be somehow done while separation on clusters without spending additional time on iteration through all the points once more.
- Setting of proper minimal and maximal size of goalposts can also reduce time of goal-detection. Those sizes can be defined after proper testing from different angles of robot's view on the soccer field.
- The detection should be possible to do without a need for Canny edge detector, but only with random Forests.
- There is still a big room for optimization on many other aspects of the code.

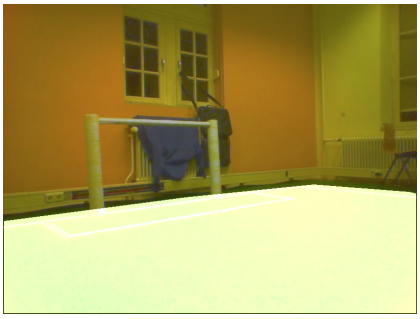
C List of recommended improvements

Even though results found with the approach taken are promising, there are still aspects of the module which can be improved or expanded upon.

- The ground detection algorithm, though stable, can still experience false positives depending on the environment. Resulting in an incorrect model of the field. This could be improved with some color calibration techniques [9].
- Changing the algorithm to take into account the history of the position of the robot in addition to an image from the visual feed would also decrease risk of a misinterpretation of the field.
- Goal detection algorithm need to be extended with detection of partially visible goals. This extension could be done by application of some machine learning techniques. Another possibility would be some usage of history for probability calculation of candidates for partially visible goals.

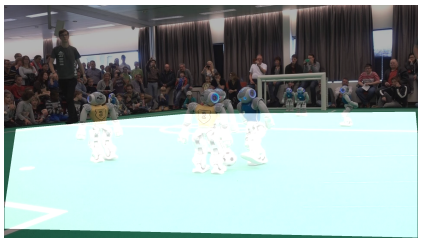
D Images of Experiments

On the following pages you will find images visualizing the results of the algorithms at different step.



(a) Ground Detection from video

(b) Partially covering polygon from Ground Detection



(c) Ground Detection from online image

Figure 9: Results from Ground Detection experiments

D.1 Ground Detection

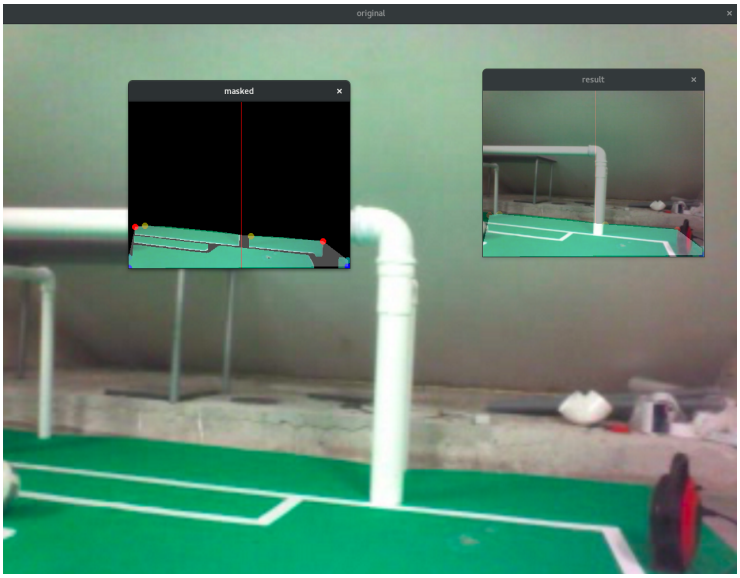


Figure 10: One of the sample image from Tarvas' bachelor thesis. Results of the algorithm are very close to the results in the thesis.

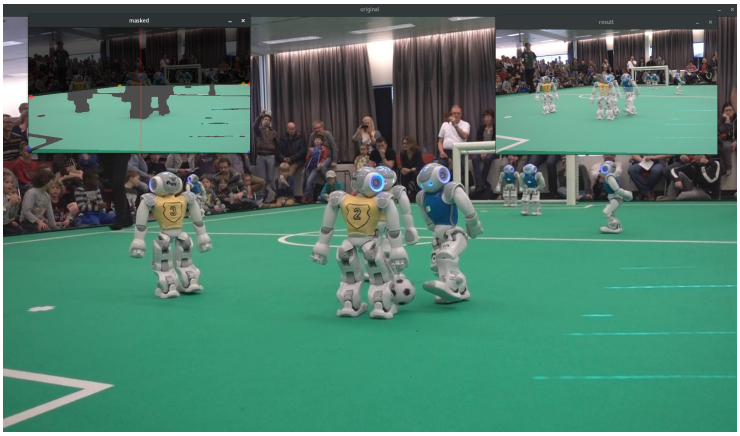


Figure 11: Image taken from the HTWK Robots RoboCup European Open 2016 Youtube video [4]. The algorithm smoothly handles the occlusion by the NAO robots.

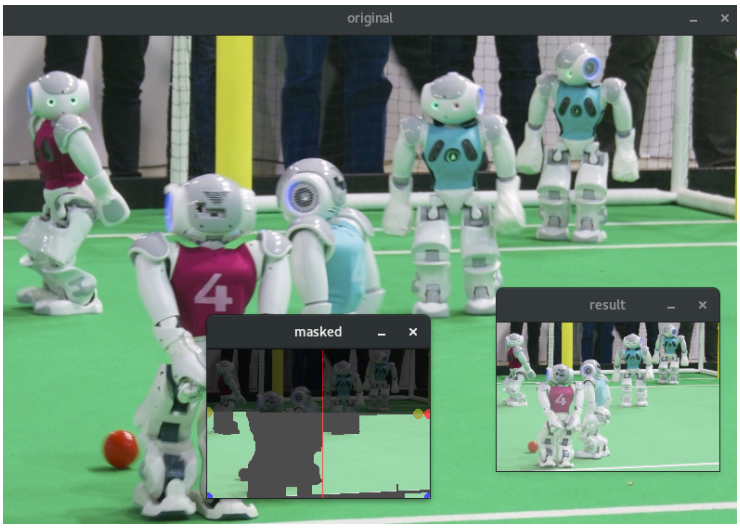


Figure 12: Image taken from the Website of the University of Sydney, Computer Science and Engineering [8]. On this example the color detection does not detect the darker shade of green in the upper right corner. The occlusion is still handled well.

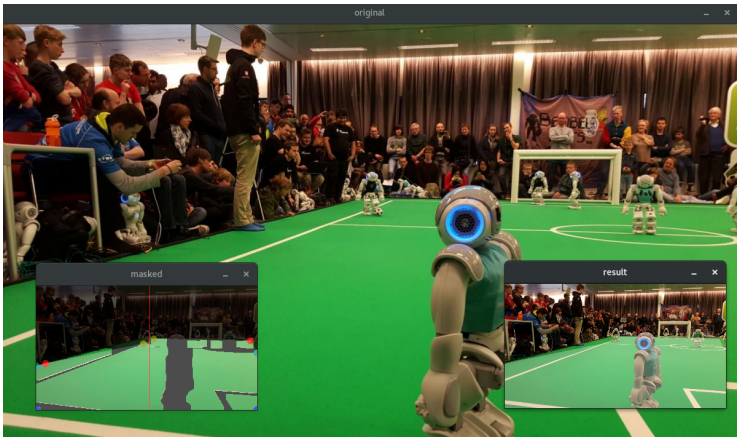


Figure 13: Image taken from the RoboCup European Open Daily Update (2 - Thursday) [2]. While not perfectly matching the field on the left the algorithm still produces a pretty good fit for the ground of the soccer field.

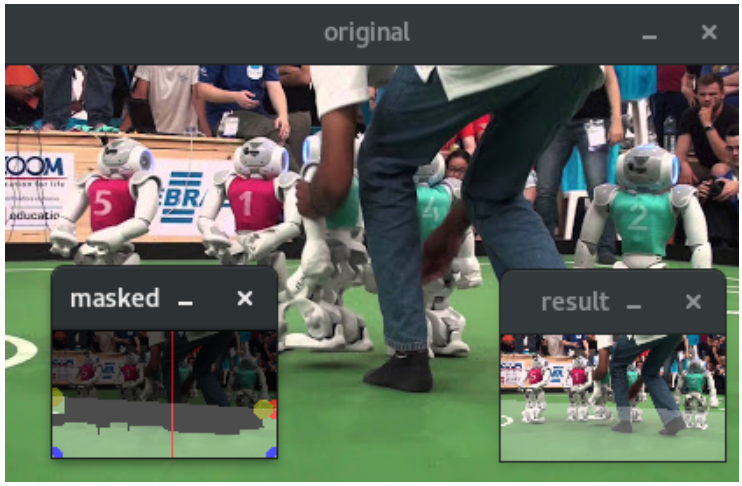


Figure 14: Image taken from the RoboCup 2015 UNSW Quarter Final [1]. Although little of the field is visible a good estimation of the field is generated.

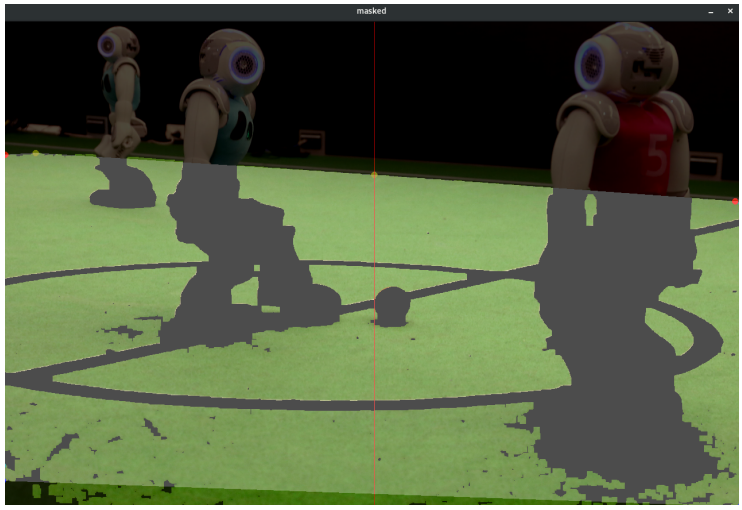


Figure 15: Image taken from ABC news [6]. The color separation as an overlay on the original image.

Figure 16: Image taken from ABC news [7]. The algorithm does not properly detect the green of the field, therefore it fails badly. The detected field is completely off.

References

[1] 2015 RoboCup SPL Quarter Final - UNSW vs NTU RoboPal . https://www.youtube.com/watch?v=QvmN0z6GpBM. Accessed: 25-01-2017.

[2] Daily update 2 - Thursday. https://www.robocupeuropeanopen.org/en/news/daily-update-2-thursday. Accessed: 25-01-2017.

[3] OpenCV Tutorial: Real Time pose estimation of a textured object. http://docs.opencv.org/trunk/dc/d2c/tutorial_real_time_pose.html. Accessed: 25-01-2017.

[4] RoboCup EuropeanOpen 2016 SPL Finals: Nao-Team HTWK vs. B-Human — Second Half . https://www.youtube.com/watch?v=CkRsQ1QCVj8. Accessed: 25-01-2017.

[5] RoboCup Standard Platform League (NAO) Rule Book. http://www.informatik.uni-bremen.de/spl/pub/Website/Downloads/Rules2016.pdf, note=Accessed: 25-01-2017.

[6] RoboCup World Championship: UNSW student engineers take robots to China to defend title. http://www.abc.net.au/news/2015-07-16/robocup-world-championship-unsw-student-engineer-defending-title/6625282. Accessed: 25-01-2017.

[7] Robots Compete in 2016 RoboCup. http://abcnews.go.com/Lifestyle/photos/robots-compete-2016-robocup-40247361/image-40247446. Accessed: 25-01-2017.

[8] Robots Standard Platform League. https://www.engineering.unsw.edu.au/computer-science-engineering/help-resources/students/student-projects/robocup/robots. Accessed: 25-01-2017.

[9] Anastasia Bolotnikova. Melioration of color calibration, goal detection and self-localization systems of nao humanoid robots. 2015.

[10] Kurt Konolige-Gary Bradski Ethan Rublee, Vincent Rabaud. Orb: an efficient alternative to sift or surf. 2011.

[11] Tinne Tuytelaars-Luc Van Gool Hebert Bay, Andreas Ess. Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[12] David G. Lowe. Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision*, 2:1150–1157, 1999.

[13] Karl Tarvas. Edge information based object detection and classification. May 2016.

[14] Jesse Richter-Klug Felix Thielke Thomas Röfer, Tim Laue. B-human team description for robocup 2016. 2016.