

A Monocular Vision System for Playing Soccer in Low Color Information Environments

Hafez Farazi, Philipp Allgeuer and Sven Behnke

Abstract—Humanoid soccer robots perceive their environment only exclusively through cameras. This paper presents a monocular vision system that was originally developed for use in the RoboCup Humanoid League, but is expected to be transferable to other soccer leagues. Recent changes in the Humanoid League rules resulted in a soccer environment with less color coding than in previous years, which makes perception of the game situation more challenging. The proposed vision system addresses these challenges by using brightness and texture for the detection of the required field features and objects. Our system is robust to changes in lighting conditions, and is designed for real-time use on a humanoid soccer robot. This paper describes the main components of the detection algorithms in use, and presents experimental results from the soccer field, using ROS and the *igus*[®] Humanoid Open Platform as a testbed. The proposed vision system was used successfully at RoboCup 2015.

I. INTRODUCTION

RoboCupSoccer is an ongoing effort to develop humanoid soccer robots, with the vision of them being able to win against the FIFA world champion team by 2050. Essential to this effort is the ability of the robots to perceive the game situation in real-time under realistic conditions. Each year, the RoboCup soccer leagues update their rules, to force participating teams to develop more advanced features for their robots, and to make the competitions more similar to real soccer games. For the 2015 RoboCup, numerous changes were made in the Humanoid league rules that affect visual perception, including in particular the reduction of color coding on objects on the field. The ball is now only specified to be at least 50% white, and the goal posts are now white instead of yellow. Moreover, the field lines—a major feature for localization on the field—are now painted onto artificial grass, and are as a result no longer white. Due to these changes, the simple color segmentation and blob detection approaches that were quite popular in the past [1] [2] have become unsuitable. This paper presents a monocular vision system that addresses the new challenges by relying more on relative brightness and texture. It was tested at RoboCup 2015 and found to perform well under the new rules.

II. RELATED WORK

Visual perception is a much researched topic within the RoboCup community, and many approaches have been developed in the past, albeit now for older versions of the rules.

All authors are with the Autonomous Intelligent Systems (AIS) Group, Computer Science Institute VI, University of Bonn, Germany. Email: farazi@ais.uni-bonn.de. This work was partially funded by grant BE 2556/10 of the German Research Foundation (DFG).

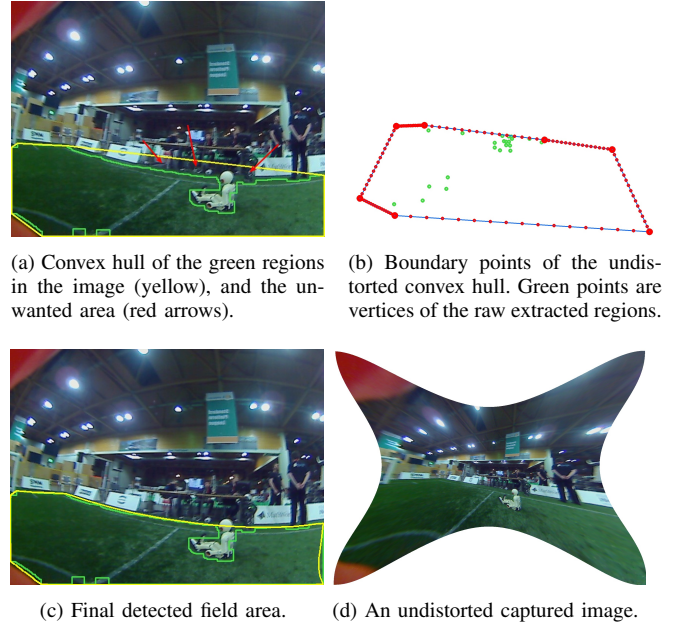


Fig. 1: Stages of field boundary detection.

Strasdat et al. [3], for example, developed a probabilistic method for robot localization relying on many unreliable detections of field features and field lines. Schulz and Behnke [4] demonstrated that the tracking of the robot pose is possible by using field lines only.

Many attempts have been made to reduce the reliance on color classification. For example, Schulz et al. [5] learned to detect the ball by classifying regions of interest with a neural network, based on color contrast and brightness features. Similarly, Metzler et al. [6] learned the detection of Nao robots based on color histograms in regions of interest.

More recent examples include the work of Houliston et al. [7], who introduced an adaptive lookup table in order to deal with changes in image illumination. Schwarz et al. [8] proposed a calibration-free vision system in the standard platform league (SPL), based on heuristics such as that the field is the maxima of the weighted histograms of each channel in the image. Härtl et al. [9] presented work on color classification based on color similarity. To address the recent changes in goal color in the SPL, Cano et al. [10] proposed a method for detecting white goals based on pixel intensities using the Y channel of the YUV color space. In order to cope with variations in illumination, Reinhardt [11] proposed different heuristics that can be applied to color histograms. The middle size league (MSL) has used white goals for a number of years, but most approaches used in the MSL are

not applicable to the humanoid league due to omnivision and active sensors such as Kinects being allowed.

The contributions of this paper include the introduction of several novel algorithms and techniques for performing soccer-related vision detections, and the general development of a soccer vision system that can run in real-time and work under various lighting conditions, and in a low color information environment.

III. SYSTEM OVERVIEW

In our vision system, the image is captured in the RGB color format, and converted to the HSV (Hue, Saturation and Value) color space using the appropriate OpenCV conversion routines [12]. For color classification purposes, we use the HSV color space due to its intuitive nature, and other special characteristics, such as its ability to separate light and chromatic information [13]. To ensure the short term consistency of the colors in the captured images, and to prevent unwanted abrupt image composition changes due to the camera itself, a fixed set of camera parameters, including for example brightness and exposure, are set in the camera device. In some cases the camera may disconnect and reconnect from the USB bus, such as possibly when the robot falls. For this situation, the camera image acquisition thread is designed in such a way that it can automatically detect a disconnect event and reconnect to the camera device.

The vision process nominally acquires camera images at 30 Hz. Each captured image is labelled with a timestamp, along with the local robot position and orientation at that time. Feature detection routines are applied to each image, in the order *field boundary*, *ball*, *lines*, *circle*, *goal posts*, and finally *obstacle*. Due to the unwanted overhead of transferring images across process boundaries, all image acquisition and processing routines are implemented in a single ROS node, referred to as the vision module. The vision module publishes its detections and outputs to suitable ROS topics, which can then be used by other nodes.

IV. CAMERA CALIBRATION

In the igus[®] Humanoid Open Platform we use a wide-angle lens to allow more of the environment to be seen at once. This introduces significant distortion however, which must be compensated when projecting image coordinates into egocentric world coordinates. The OpenCV [12] chessboard calibration procedure is used to determine intrinsic camera parameters that characterise the distortion. The following pinhole camera distortion model, including both radial and tangential distortion components, is used:

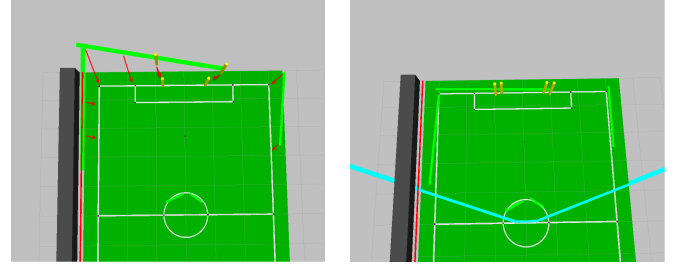
$$a = \frac{x}{z}, \quad b = \frac{y}{z}, \quad r^2 = a^2 + b^2, \quad (1)$$

$$\hat{x} = a \left(\frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} \right) + 2p_1ab + p_2(r^2 + 2a^2), \quad (2)$$

$$\hat{y} = b \left(\frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} \right) + 2p_2ab + p_1(r^2 + 2b^2), \quad (3)$$

$$u = c_x + f_x \hat{x}, \quad v = c_y + f_y \hat{y}, \quad (4)$$

where (c_x, c_y) is the principal point, and f_x and f_y are the focal lengths, all in pixel units. k_1 to k_6 are the radial



(a) Projected line and goal post detections prior to kinematic calibration, with projection errors marked. (b) Projected line and goal post detections after kinematic calibration, with the field of view marked.

Fig. 2: Effect of the kinematic calibration of the camera.

distortion coefficients, and p_1 and p_2 are the tangential distortion coefficients. The input (x, y, z) is a world vector in the camera frame, and the output (u, v) is the corresponding distorted image pixel position. For simplicity reasons, higher order coefficients and thin prism distortions are not included in the model. Due to limitations of the undistortion functionality in OpenCV, producing poor results especially near the corners, a more accurate and efficient undistortion method was implemented based on the Newton-Raphson method. This method is used to populate a pair of lookup tables that allow $O(1)$ distortion and undistortion operations at runtime. The effect of undistortion is shown in Fig. 1d.

The distortion model allows projection to the camera frame, but further extrinsic parameters are required to allow projection to egocentric world coordinates. More specifically, at each point in time the transformation from the egocentric world frame to the camera frame must be known, which is both time- and robot-specific. We use the ROS-native tf2 library [14] for this purpose, which amalgamates joint position and robot kinematics information to produce the required time-varying transforms. Although we have a relatively exact kinematic model of the robot, some variation still occurs in the real hardware, resulting in potentially large projection errors for distant objects. To account for this, we calibrate the kinematics on each robot using a hill-climbing approach that tunes translation and rotation offsets between the torso, which contains the IMU, and the camera. These offsets are crucial for good performance of the pixel to egocentric coordinate projection algorithm. The effect of the kinematic calibration is demonstrated in Fig. 2. The left image shows the detected lines and goal posts projected onto the ground plane prior to the calibration, and the right image shows the results after calibration. For reference, the corresponding raw captured image is shown in Fig. 5b. Note that the calibration procedure is interactive in the sense that the user can select a number of points in the captured image and the corresponding true field locations. This is used as the input to the hill-climbing method, which seeks to minimise the reprojection error of the selected points. This completes the calibration of the camera for the purposes of the vision module.



Fig. 3: Extending one positive sample (leftmost) to ten, by applying rotations and mirroring operations.

V. FEATURE DETECTION ALGORITHMS

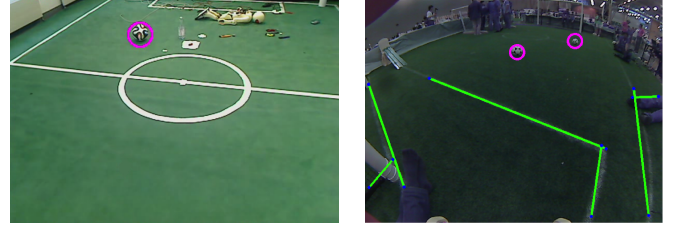
A. Field Detection

The soccer field used in RoboCup competitions is green, so field detection can be based on color segmentation in the HSV color space. Using a user-selected green color range, a binarized image is constructed, and all connected components that appear below the estimated horizon are found. Connected regions that have an area greater than some chosen threshold are taken into consideration for the extraction of a field boundary. If too many separate green regions exist, a set maximum number of regions are considered, prioritised by size and distance from the bottom of the image. Although it is a common approach to find a convex hull of all green areas directly in the image [10] [1], more care needs to be taken in our case due to the significant image distortion. Fig. 1a shows how the convex hull may include parts of the image that are not the field. To exclude these unwanted areas, the vertices of the connected regions are first undistorted before calculating the convex hull, shown in Fig. 1b. The convex hull points, and intermediate points on each edge, are then distorted back into the raw captured image, and the resulting polygon is taken as the field boundary. An example of the final detected field polygon is shown in Fig. 1c.

B. Ball Detection

In previous years of the RoboCup, most teams used simple color segmentation and blob detection based approaches to find the orange ball. Now that the ball is mostly white however, and generally with a pattern, such simple approaches no longer work effectively, especially since the lines and goal posts are also white. Our approach is divided into two stages. In the first stage, ball candidates are generated based on color segmentation, color histograms, shape, and size. White connected components in the image are found, and the Ramer-Douglas-Peucker algorithm [15] is applied to reduce the number of polygon vertices in the resulting regions. This is advantageous for quicker subsequent detection of circle shapes. The detected white regions are searched for at least one third full circle shapes within the expected radius ranges using a technique very similar to Algorithm 3. Color histograms of the detected circles are calculated for each of the three HSV channels, and compared to expected ball color histograms using the Bhattacharyya distance [12]. Circles with a suitably similar color distribution to expected are considered to be ball candidates.

In the second stage of processing, a dense histogram of oriented gradients (HOG) descriptor [16] is applied in the form of a cascade classifier, with use of the AdaBoost technique. Using this cascade classifier, we reject those candidates that do not have the required set of HOG features. The AdaBoost technique is used because it provides a powerful



(a) Ball detection in an undistorted image, with other white objects. (b) Multiple balls detected on a soccer field, with image distortion.

Fig. 4: Ball detections under various conditions.

Algorithm 1 Merge similar line segments

Input: A set of line segments S

Output: A set of line segments N where $|N| \leq |S|$

```

1:  $N \leftarrow S$ 
2: repeat
3:    $M \leftarrow N$ 
4:    $m \leftarrow |M|$ 
5:    $N \leftarrow \emptyset$ 
6:   while  $M \neq \emptyset$  do
7:      $X \leftarrow$  Any element of  $M$ 
8:      $M \leftarrow M \setminus \{X\}$ 
9:      $\hat{m} \leftarrow |M|$ 
10:    for  $Y \in M$  do
11:      if  $MinDistance(X, Y) < 0.25\text{ m}$ 
12:        and  $AngleDiff(X, Y) < 15^\circ$  then
13:           $N \leftarrow N \cup \{Merge(X, Y)\}$ 
14:           $M \leftarrow M \setminus \{Y\}$ 
15:          break
16:    end if
17:  end for
18:  if  $|M| = \hat{m}$  then
19:     $N \leftarrow N \cup \{X\}$ 
20:  end if
21: end while
22: until  $|N| = m$ 
23: return  $N$ 

```

bound on the generalization performance [17]. In contrast to what is suggested in [16] however, which was targeted at pedestrian detection, we do not use a multi-scale sliding window technique. Instead, to save computational time, we only apply the HOG descriptor to the regions suggested by the ball candidates. The aim of using the HOG descriptor is to find a description of the ball that is largely invariant to changes in illumination and lighting conditions. In contrast to other common feature descriptors, such as SIFT [18], HOG produces a more easy output to understand and visualize [19], and has for example been found to be more reliable than SIFT in pedestrian detection [16]. The HOG descriptor is not rotation invariant however, so to detect the ball from all angles, and to minimize the user's effort in collecting training sets, each positive image is rotated by $\pm 10^\circ$ and $\pm 20^\circ$, selectively mirrored, with the resulting images being presented as new positive samples. Greater rotations are not

Algorithm 2 *Merge*(X, Y): Merge two line segments

Input: Two line segments X and Y **Output:** Merged line segment Q

```
1:  $\theta \leftarrow 0$ 
2:  $X_m = \text{Midpoint}(X)$ 
3:  $Y_m = \text{Midpoint}(Y)$ 
4:  $r \leftarrow \|X\| / (\|X\| + \|Y\|)$ 
5:  $P \leftarrow rX_m + (1 - r)Y_m$ 
6: if  $\|X\| \geq \|Y\|$  then
7:    $\theta \leftarrow \text{Slope}(X)$ 
8: else
9:    $\theta \leftarrow \text{Slope}(Y)$ 
10: end if
11:  $Z \leftarrow$  The line through point  $P$  of slope  $\theta$ 
12:  $S \leftarrow$  Orthogonal projections of the endpoints of  $X$  and  $Y$  onto  $Z$ 
13:  $Q \leftarrow$  The largest line segment defined by points in  $S$ 
14: return  $Q$ 
```

considered to allow the cascade classifier to learn the shadow under the ball. So in summary, each positive sample that the user provides is extended to a total of 10 positive samples, shown for example in Fig. 3, where the original sample is the leftmost. With a set of about 400 positive and 700 negative samples that are gathered directly from the robot camera and with 20 weak classifiers, training takes approximately 10 hours.

As demonstrated in Fig. 4, the described approach can detect balls with very few false positives, even in environments cluttered with white and with varying lighting conditions. In our experiments, we found that this approach can detect the FIFA size 4 ball up to 4 meters away. An interesting result is that our approach can find the ball in undistorted and distorted images with the same classifier. A full set of generalized Haar wavelet features [20] was tested on the same data set, and although the result was comparable, the training time was about 10 times slower than for the HOG descriptor. A Local Binary Patterns (LBP) feature classifier [21] was implemented in another comparative test, but the ball detection results were poor.

C. Field Line Detection

Due to the introduction of artificial grass in the RoboCup humanoid league, the lines are no longer clearly visible and identifiable on the field. In past years, many teams based their line detection approaches on the segmentation of the color white. This is no longer a robust approach due to the increased number of white objects on the field, and due to the visual variability of the lines. Our approach is to detect spatial changes in brightness in the image using a canny edge detector [22] on the V channel of the HSV color space. The V channel encodes brightness information, and the result of the canny edge detector on this input is quite robust to changes in lighting conditions, so there is no need to tune parameters for changing lighting conditions. A probabilistic Hough line detector [23] is used to extract line segments

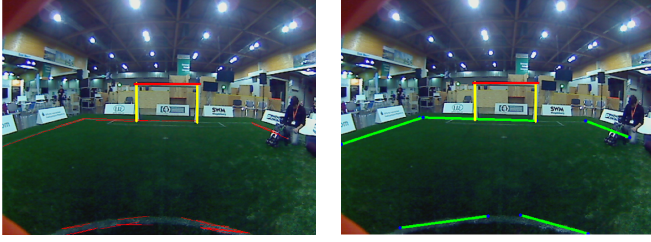
TABLE I: Helper functions used in the algorithms

<i>MinDistance</i> (X, Y)	Returns the minimum distance between two line segments X and Y
<i>AngleDiff</i> (X, Y)	Returns the angle between two line segments in degrees
<i>Merge</i> (X, Y)	See Algorithm 2
<i>Midpoint</i> (X)	Returns the midpoint of a line segment
<i>Slope</i> (X)	Returns the slope of a line segment
<i>MaxPtDistance</i> (S)	Returns the maximum distance between any pair of points in the set S
<i>GetBisector</i> (X)	Returns the perpendicular bisector line of the given line segment
<i>HasIntersection</i> (X, Y)	Returns whether two line segments intersect at a point
<i>GetIntersection</i> (X, Y)	Returns the intersection point of two line segments
<i>DistanceToLine</i> (P, X)	Returns the Euclidean distance between point P and line segment X
<i>Mean</i> (S)	Returns the mean of the points in the set S

of a certain minimum size from the detected edges. This helps to reject edges from white objects in the image that are not lines. The output line segments are filtered in the next stage to avoid false positive line detections where possible. A thresholding technique is used to try to ensure that the detected lines cover white pixels in the image, have green pixels on either side, and are close on both sides to edges returned by the edge detector. The last of these checks is motivated by the expectation that white lines, in an ideal scenario, will produce a pair of high responses in the edge detector, one on each border of the line. Ten equally spaced points are chosen on each line segment under review, and two normals to the line are constructed at each of these points, of approximate 5 cm length in each of the two directions. The pixels in the captured image underneath these normals are checked for white color and green color, and the output of the canny edge detector is checked for high response. The number of instances where these three checks succeed are independently totalled, and if all three counts— V_W , V_G and V_E respectively—exceed the configured thresholds the line segment is accepted, otherwise the line segment is rejected.

In the final stage, similar line segments are merged with each other, as appropriate, to produce fewer and bigger lines, as well as cover those line segments that might be partially occluded by another robot. Prior to merging we project each line segment to the egocentric world coordinates. The merging algorithm is detailed in Algorithm 1, which in turn uses Algorithm 2 to perform the individual merge operations. A list of the simple helper functions that are used in the algorithms in this paper is presented in Table I. Note that the constant thresholds used in Algorithm 1, are indicating the maximum allowed difference in angle and distance between two lines to be merged, and the numeric values are suitable for the current humanoid league field dimensions.

The final result is a set of line segments that relates to the lines and center circle on the field. Line segments that are under a certain threshold in length undergo a simple circle



(a) The output of the probabilistic Hough line detection on the extracted edges (thin red lines). (b) The final field line (green lines) and goal post (yellow lines) detections, marked in the captured image.

Fig. 5: Line detections on a soccer field.

detection routine, detailed in Algorithm 3, to find the location of the center circle. In our experiments, we found that this approach can detect circle and line segments up to 4.5 m away.

We generate a probability estimation for each line detection, so that a localization algorithm for example can assign higher weights to detections of higher probability. For a detected line X of length L , and check counts V_G and V_E , the probability $P(X)$ of the detection is calculated as

$$P(L, V_G, V_E) = \begin{cases} \frac{V_G}{V_{GM}} \cdot \frac{V_E}{V_{EM}} \cdot \frac{L-1}{3} & \text{if } 1 < L < 3, \\ \frac{V_G}{V_{GM}} & \text{if } L \geq 3, \\ 0 & \text{if } L \leq 1, \end{cases} \quad (5)$$

where V_{GM} is the maximum possible value of V_G , and V_{EM} is the maximum possible value of V_E . For a detected circle C , consisting of line segments X_i for $i = 1 \dots n$, with corresponding check counts V_{Gi} and V_{Ei} , the probability $P(C)$ of the detection is calculated as

$$P(C) = \frac{\sum_{i=1}^n V_{Gi}}{nV_{GM}} \cdot \frac{\sum_{i=1}^n V_{Ei}}{nV_{EM}} \cdot \frac{MaxPtDistance(P)}{0.75}, \quad (6)$$

where P is the set of estimated circle centre points, as calculated by Algorithm 3.

D. Goal Detection

In our vision module, detection of the white goal posts is divided into two stages. In the first stage, the image is binarized using color segmentation of the color white, defined as a particular region of the HSV color space by the user. A sample output is shown in Fig. 6a. Horizontal and vertical lines are then extracted from the binarized image using probabilistic Hough line detection [23]. Using a similar approach as for field line detection, the detected vertical and horizontal line segments are merged together to produce fewer, bigger lines. The main idea behind merging is to have one line segment on each goal post, instead of two on each side. Moreover, by merging line segments we can overcome only partially observable goal posts, and motion-blurred images in which the background may blend into some part of the goal post. The result of the merging is shown in Fig. 6b. In the second stage of goal post detection, each of

Algorithm 3 Detect circle from line segments

Input: A set of line segments S

Output: A boolean flag whether a circle is detected

Output: The centre $C \in \mathbb{R}^2$ of the detected circle

```

1:  $P \leftarrow \emptyset$ 
2: for all Line segments  $X, Y \in S$  such that  $X \neq Y$  do
3:    $X_B \leftarrow GetBisector(X)$ 
4:    $Y_B \leftarrow GetBisector(Y)$ 
5:   if  $HasIntersection(X_B, Y_B)$  then
6:      $C \leftarrow GetIntersection(X_B, Y_B)$ 
7:     if  $DistanceToLine(C, X) \approx 0.75$ 
       and  $DistanceToLine(C, Y) \approx 0.75$  then
8:        $P \leftarrow P \cup \{C\}$ 
9:     end if
10:  end if
11: end for
12: if  $|P| \geq 5$  and  $MaxPtDistance(P) < 0.75$  then
13:   return true,  $Mean(P)$ 
14: end if
15: return false,  $(0, 0)$ 

```

*: Note that the number 0.75 in this algorithm is related to the expected circle radius in metres.

the vertical line segments that do not meet the following criteria are rejected.

- The length of the line segment must be within a certain range, dependent on the distance from the robot to the projected bottom point of the line.
- The bottom of the line must be within the field.
- The top of the line must be above the estimated horizon.
- On a kid-size field, if the goal post candidate is more than 2 m from the robot there must be one horizontal line segment close to the candidate.

To reject the vertical line segments that might belong to a standing white robot, we check the homogeneity of the texture of each of the merged vertical candidates. To do this, we calculate 10 equally spaced points on the line, and search locally around each of these points for large changes in the H channel of the HSV color space. If many changes in the H channel are observed, the candidate is rejected. Experiments have shown that this approach can detect goal posts up to 5 m away.

As for field lines, we assign a probability $P(G)$ to detected goal posts, using the equation

$$P(G) = \begin{cases} \frac{3}{4} & \text{if } n = 1, \\ 1 - \frac{1}{6}|D - \|X_1 - X_2\|| & \text{if } n = 2, \\ 1 - \frac{n}{5} & \text{if } 3 \leq n \leq 4, \\ 0 & \text{if } n \geq 5, \end{cases} \quad (7)$$

where n is the number of the detected goal posts, D is the known goal width, and X_1, \dots, X_n are the egocentric world vectors to the n detected goal posts.



(a) White color segmented image in which pixels that are not considered to be white have been painted black. (b) Result after merging all extracted horizontal and vertical white line segments (annotated in blue).

Fig. 6: Goal detection on a soccer field.

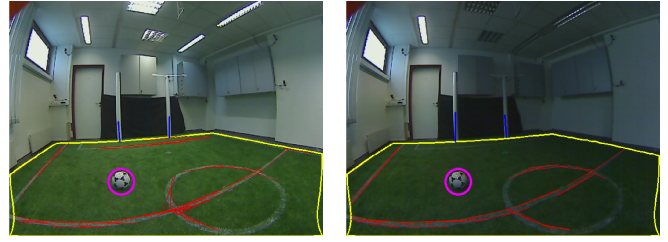
E. Obstacle Detection

The rules of the RoboCup humanoid league state that all robots should have mostly black feet. As such, obstacle detection is based on color segmentation of the black color range that is defined by the user. We search for black connected components within the field boundary, and if a detected component is large enough and within a certain predefined distance interval from the robot, the lowest point of the component is returned as the egocentric world coordinates of an obstacle. To prevent the detection of own body parts as obstacles, an image mask is implemented that is dependent on the position of the head. This allows the regions of the image that are expected to contain parts of the own body to be ignored. Using this approach we are able to detect robots at distances of up to 2 m.

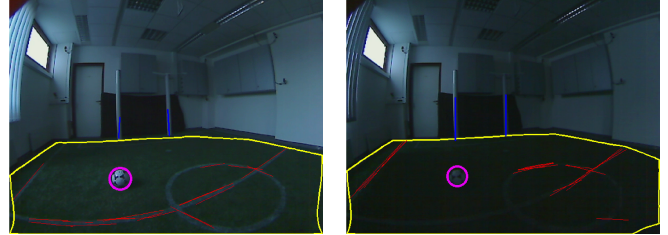
VI. EXPERIMENTAL RESULTS

We used the igus® Humanoid Open Platform [24] in our experiments. This robot is about 90 cm tall and can be used in both the teen-size and kid-size leagues. The robot has a computer with a dual-core AMD E-450 1.65 GHz processor and 2GB of memory. This robot is equipped with two 720p Logitech C905 USB cameras. Each camera is fitted with a wide-angle lens that has an infrared cut-off filter, yielding a diagonal field of view of approximately 150°. We use the Video4Linux2 library to capture images in RGB format at a resolution of 640×480 . So far we have only evaluated the vision system on one camera at a time. During RoboCup 2015, the computation time of the vision system was measured on the robot. Using only one thread an average cycle time of 25 ms was achieved, with a minimum of 15 ms, and maximum of 38 ms, so the proposed vision system can easily run on a single CPU core at 30 Hz. Of all the components of the vision system, the ball detection is the most time consuming, and the robot detection is the least.

We assessed the performance of the proposed vision system both qualitatively and quantitatively, using an approach similar to that used by Schwarz et al. [8]. As inputs, we used recorded data from the RoboCup German Open 2015, which included captured image data from various different field locations and lighting conditions. The recorded



(a) Normal lighting conditions. (b) Poor lighting conditions.



(c) Natural lighting. (d) No ambient lighting.

Fig. 7: Experimental results under various lighting conditions without any changes in the algorithm parameters.

data contained image data, kinematic information and dead-reckoning walking data. Many captured images were affected by motion blur due to walking and head panning motions. The data was evaluated frame by frame by the authors to assess the performance of the vision system in real soccer situations. Table II defines a set of detection criteria for the evaluation of the recorded bags. The success rate of the detections was calculated given the criteria, and the number of false positives was recorded. The results are shown in Table III. Despite the fact that Schwarz et al. tested their vision algorithms in a more color-coded SPL environment [8], we achieve comparable results, and in fact achieve better results when it comes to circle, ball and robot detections. In another comparison of the obtained results, we found that our method outperforms that of Härtl et al. [9] in terms of both line and circle detections, even though their results were obtained in the previous year's more color-coded RoboCup environment. In yet another experiment, we tested our vision system under various different lighting conditions, including normal, very poor, ambient and natural lighting conditions, without tuning any parameters. The results are shown in Fig. 7. Consistent detections can be seen in all scenarios. We claim that the achieved result is sufficient for successful soccer gameplay in the new low color information RoboCup environment.

VII. CONCLUSION AND FUTURE WORK

In this paper, we presented a monocular vision system for humanoid soccer robots that addresses the challenges posed by the new rules of the RoboCup Humanoid League. In order to tackle the challenging task of finding objects in a low color information environment, we proposed a learning approach for ball detection. For detecting goals, field lines, the center circle, and other robots we proposed effective algorithms with low computational cost. Our methods have been

TABLE II: Evaluation criteria used for feature detections

Feature	Criteria for expected observability
Field boundary	The field should be visible in the image.
Ball	At least one third of the ball should be visible. The ball should be completely within the field boundary at ≤ 4 m from the robot.
Field lines	All points of the field line segment should be at most 4.5 m from the robot. The line segment should be at least 1 m in length.
Centre circle	At least one third of the circle should be visible.
Goal posts	The goal post should be ≤ 5 m from the robot. The base of the goal post should be visible. The goal post should have a minimum margin of 30 px from the edges of the image.
Other robots	The feet of the robot should be completely black and visible.

TABLE III: Detection results for each type of feature

Feature	Success rate	False positives	Frames
Field boundary	93%	0	1482
Ball	81%	11	457
Field lines	57%	17	187
Centre circle	63%	7	138
Goal posts	48%	3	140
Other robots	71%	0	239

evaluated on data from a RoboCup competition and in lab experiments. The results indicate good detection performance and robustness to changes in lighting conditions.

In the future, we want to increase the maximum distance of ball detection, and utilize stereo vision for better distance estimation. Porting some of the system to run on a GPU is additional future work, along with developing a learning approach for more general robot detection, as the feet of other robots may not always be visible in the image.

REFERENCES

- [1] T. Laue, T. J. De Haas, A. Burchardt, C. Graf, T. Röfer, A. Härtl, and A. Rieskamp, "Efficient and reliable sensor models for humanoid soccer robot self-localization," in *Proceedings of the Fourth Workshop on Humanoid Soccer Robots*, 2009, pp. 22–29.
- [2] H. Farazi, M. Hosseini, V. Mohammadi, F. Jafari, D. Rahmati, and E. Bamdad, "Baset Teen-Size 2014 Team Description Paper," 2014.
- [3] H. Strasdat, M. Bennewitz, and S. Behnke, "Multi-cue localization for soccer playing humanoid robots," in *RoboCup 2006: Robot Soccer World Cup X*, ser. Lecture Notes in Computer Science, vol. 4434. Springer, 2007, pp. 245–257.
- [4] H. Schulz and S. Behnke, "Utilizing the structure of field lines for efficient soccer robot localization," *Advanced Robotics*, vol. 26, no. 14, pp. 1603–1621, 2012.
- [5] H. Schulz, H. Strasdat, and S. Behnke, "A ball is not just orange: Using color and luminance to classify regions of interest," in *Proceedings of The Second Workshop on Humanoid Soccer Robots, Humanoids Conference*, 2007.
- [6] S. Metzler, M. Nieuwenhuisen, and S. Behnke, "Learning visual obstacle detection using color histogram features," in *Proceedings of the 15th RoboCup International Symposium*, 2011.
- [7] T. Houlston, M. Metcalfe, and S. K. Chalup, "A fast method for adapting lookup tables applied to changes in lighting colour," 2015.
- [8] I. Schwarz, M. Hofmann, O. Urbann, and S. Tasse, "A robust and calibration-free vision system for humanoid soccer robots," 2015.
- [9] A. Härtl, U. Visser, and T. Röfer, "Robust and efficient object recognition for a humanoid soccer robot," in *RoboCup 2013: Robot World Cup XVII*. Springer, 2014, pp. 396–407.
- [10] P. Cano, Y. Tsutsumi, C. Villegas, and J. Ruiz-Del-Solar, "Robust detection of white goals," in *RoboCup Symposium*, 2015.
- [11] T. Reinhardt, "Kalibrierungsfreie Bildverarbeitungs-algorithmen zur Echtzeitfähigen Objekterkennung im Roboterfußball," Ph.D. dissertation, 2011.
- [12] G. Bradski, *Dr. Dobb's Journal of Software Tools*, 2000.
- [13] P. M. Caleiro, A. J. Neves, and A. J. Pinho, "Color-spaces and color segmentation for real-time object recognition in robotic applications," *Electrónica e Telecomunicações*, vol. 4, no. 8, pp. 940–945, 2007.
- [14] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA)*, 2013 IEEE International Conference on, ser. Open-Source Software workshop, April 2013, pp. 1–6.
- [15] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer graphics and image processing*, vol. 1, no. 3, pp. 244–256, 1972.
- [16] N. Dalal and B. Triggs, "Object detection using histograms of oriented gradients," in *Pascal VOC Workshop, ECCV*, 2006.
- [17] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Annals of statistics*, pp. 1651–1686, 1998.
- [18] D. G. Lowe, "Object recognition from local scale-invariant features," in *The proceedings of the seventh IEEE international conference on Computer Vision*, 1999, vol. 2, 1999, pp. 1150–1157.
- [19] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba, "HOGgles: Visualizing Object Detection Features," *ICCV*, 2013.
- [20] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *International Conference on Image Processing*, vol. 1, 2002, pp. I–900.
- [21] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Z. Li, "Learning multi-scale block local binary patterns for face recognition," in *Advances in Biometrics*. Springer, 2007, pp. 828–837.
- [22] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 6, pp. 679–698, 1986.
- [23] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic Hough transform," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119–137, 2000.
- [24] P. Allgeuer, H. Farazi, M. Schreiber, and S. Behnke, "Child-sized 3D printed igus humanoid open platform," in *Proceedings of 15th IEEE-RAS Int. Conference on Humanoid Robots (Humanoids)*, 2015.