Robust and Efficient Object Recognition for a Humanoid Soccer Robot

Alexander Härtl[1], Ubbo Visser[1], and Thomas Röfer[2]

[1] University of Miami, Department of Computer Science,
1365 Memorial Drive, Coral Gables, FL, 33146 USA
{a.haertl,visser}@cs.miami.edu
[2] Deutsches Forschungszentrum für Künstliche Intelligenz,
Cyber-Physical Systems, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
thomas.roefer@dfki.de

Abstract. Static color classification as a first processing step of an object recognition system is still the de facto standard in the RoboCup Standard Platform League (SPL). Despite its efficiency, this approach lacks robustness with regard to changing illumination. We propose a new object recognition system where objects are found based on *color similarities*. Our experiments with line, goal, and ball recognition show that the new system is real-time capable on a contemporary NAO (version 3.2 and above). We show that the detection rate is comparable to color-table-based object recognition under static lighting conditions and substantially better under changing illumination.

1 Introduction

Color-based recognition of geometrically simple objects is a well known problem that has been extensively studied for over two decades [18,5]. Prominent examples for successful image processing methods are edge detection [2,10], region-growth algorithms [21,8], and histogram-based algorithms [12,19,14]. A popular approach is based on edge detection and subsequent Hough transformation [4] in which the authors show a method for line detection that can be used for more general curve fitting. Although the literature shows a broad range of variations and implementations of the mentioned approaches, only a few can be used for embedded systems that are constrained by limited resources such as time, memory, and/or CPU power.

The RoboCup Soccer environment demands efficient real-time object recognition. Many systems are still based on fixed color tables that are similar or based on the *CMVision* system [1]. It is well suited for static lighting conditions but lacks robustness when illumination varies. Röfer [15] improved the robustness by introducing ambiguous color classes and delaying hard decisions to a later processing stage. Reinhardt [14] uses different heuristics applied to color histograms to cope with variations in illumination.

We propose a new object recognition system where objects are found based on *color similarities*. As a first step, a subsampling is created considering the

S. Behnke et al. (Eds.): RoboCup 2013, LNAI 8371, pp. 396–407, 2014.

perspective projection of objects on the field plane (Section 2.1). Based on the subsampling, the image is segmented line-wise considering color similarities of neighboring pixels, similar to region growing (Section 2.2). The actual object recognition (Section 3) is based on the result of this segmentation. Line and ball detection are based on region growing, considering the shape of the objects, whereas the goal detection is histogram-based.

We have conducted our experiments for line, goal, and ball recognition (Section 4) and our results show that it is real-time capable on a contemporary NAO (version 3.2 and above).

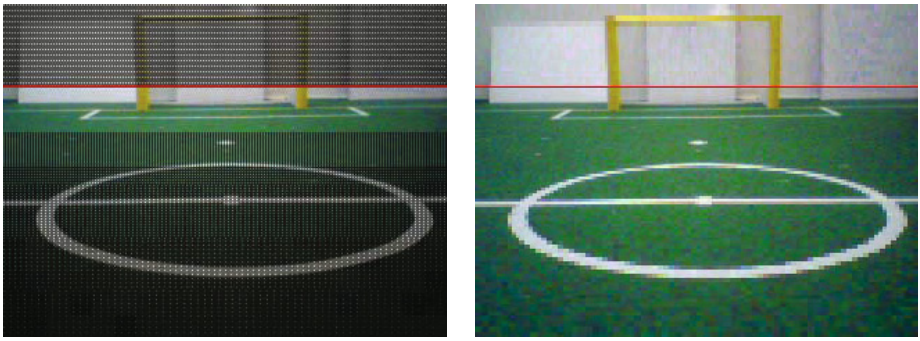
2 Preprocessing

Processing the image in its full resolution (YUV422 640×480) is computationally expensive and would not allow for real-time operation. Therefore, the effective resolution has to be reduced. The naïve approach would be to reduce the resolution for the actual processing globally. This, however, does not account for the perspective projection of objects on the field plane: closer objects appear larger in the image. In order to rectify this effect the image is locally subsampled with different resolutions.

2.1 Subsampling

The image is split into two areas: a) the field plane and b) the area above this plane, for which a static subsampling is applied with a high horizontal and low vertical resolution suitable for goal post detection. The basic idea for a) is to project a homogeneous grid on the field plane into the image [11]. However, the exact projection of a grid is approximated by two means to increase the efficiency of the following segmentation process. First, the rotation about the image axis is ignored because a humanoid robot operates mostly upright and therefore the camera possesses a small rotation about that axis. This has been verified experimentally, which showed that the average deviation while walking and moving the head is only 1.15 degrees [6]. The second approximation limits the sampling points within a line to a fixed interval, i.e., the distance between the sampling points is limited to a fixed number of pixels.

Both approximations decrease the complexity and therefore increase the efficiency of the subsequent segmentation process. The resulting subsampling is reduced to rectangular blocks of homogeneous grids of sampling points, as shown in Figure 1. The computation of the grid is thus reduced to finding the coordinates of the borders of those blocks of different resolutions. Note that the computation of the subsampling requires the camera pose to be known; here it is computed based on sensor fusion of forward kinematics and the IMU (cf. [16]). First, the border between the upper and lower area is found by projecting a point in the distance of the field diagonal into the image plane. This is motivated by the fact that the farthest object to be recognized on the field plane has a



(a) (b)

Fig. 1. Visualization of the subsampling using greyed out pixels (a) and successive supersampling (b). The resolution within the field plane incorporates the perspective projection, whereas the area above the field plane (marked by the red line) is subsampled statically. It can be seen that the shape of all objects remains sound despite the subsampling, regardless of their distance.

distance of the field diagonal. Second, the borders between the blocks of different granularity within the lower area are found using the following formula:

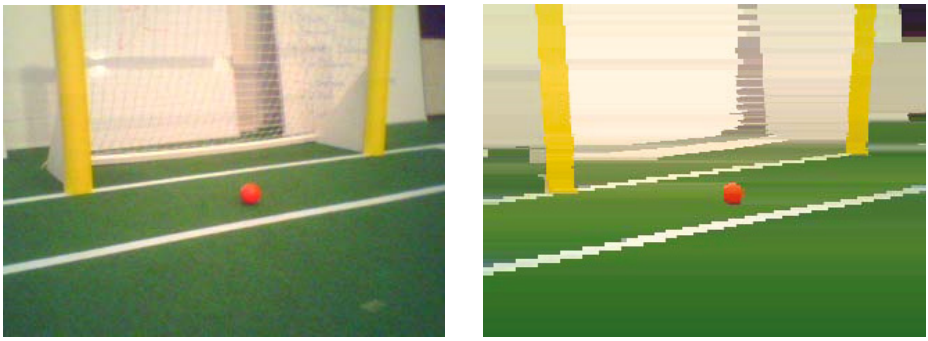
$$y_{\Delta} = \frac{\Delta_f \, h + \Delta \sqrt{\mathbf{R}_{11}^2 + \mathbf{R}_{12}^2}}{\Delta \left(\mathbf{R}_{13} \, f - \mathbf{c}_y \sqrt{\mathbf{R}_{11}^2 + \mathbf{R}_{12}^2} \right)} \tag{1}$$

with Δ being the desired pixel spacing in the image plane, Δ_f the mesh size in the field plane, f the effective focal length of the camera, $\mathbf{R}$ the rotation matrix of the camera, $\mathbf{c}$ the optical center, and h the height of the camera above the field plane. To compute the blocks, y_{Δ} is computed for increasing pixel spacings Δ, beginning with 1, which delivers the vertical positions of the borders of the blocks in the image plane.

The mesh size of the grid in the field plane has a significant influence on both the processing time and the recognition accuracy. A fine mesh size results in a large processing time, whereas a coarse mesh size results in an impaired detection rate and accuracy. Therefore, a good compromise has to be found. Having in mind that the resulting system is supposed to be real-time capable, it is desirable to pursue a constant processing time. Assuming that the processing time is proportional to the number of sampling points, the mesh size should be chosen in a way that a certain constant number of sampling points is generated. We use the secant method [13] as a numerical approximation algorithm to find an appropriate value for Δ_f, which usually converges after approx. 5 iterations.

2.2 Segmentation

After the subsampling is determined, it is used to segment the camera image. The general idea of the segmentation process is based on the popular *region growing* [9], which has been modified to reduce the processing time. It combines



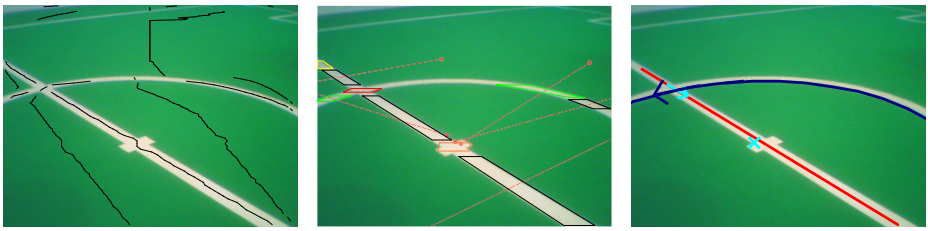
(a) (b)

Fig. 2. Original camera image (a) and the result of the preprocessing stage (b). The segmentation considers the previously determined subsampling.

the efficiency of run length encoding (as e.g. in [1]) with the robustness of region growing. Therefore, the image is traversed linearly in a single pass, and regions can only grow to the right. Also, regions are limited to single rows, so that the approach can be seen as a dynamic programming implementation of region growing. Henceforth, a region within a row is referred to as a *segment*. The similarity criterion used is the comparison with the average color of the hitherto segment. A major advantage of the modified region growing is that the memory is only traversed once in a linear fashion, which significantly reduces random memory accesses and improves the caching efficiency and ultimately the processing time.

Color comparison is performed channel-wise: the absolute color difference of each channel is computed and compared to a predefined threshold, and the pixel is added to the segment if no threshold is exceeded, otherwise a new segment is started. The major advantage of separate thresholds per channel in conjunction with the YUV color space provided by NAO's cameras is that a higher threshold can be chosen for the brightness channel than for the color channels to improve the illumination invariance. Also, the parameter space of the segmentation process is reduced to three single values, compared to the high dimensional configuration space of the common color-table-based approach.

The result of the segmentation process is shown in Figure 2. The amount of data is significantly reduced to an average of 1,250 segments compared to 76,800 pixels ($\approx 1.6\%$). Also, the amount of noise is significantly reduced, because the region growing and the averaging acts as a kind of noise filter. Most importantly, all relevant objects (goal posts, lines, ball) have been clustered into segments within each row. However, the approach has some disadvantages. In some areas, especially around the ball, the object is split into multiple segments due to the shadings of the object. Another disadvantage of this segmentation approach is the inter-line inconsistency that arises from line-wise processing [17]. We can observe this by looking at the goal posts (see Figure 2b). Their expected trapezoid-like shape is far from being perfect and include some awkward bends. However, both problems are treated explicitly in the corresponding object detectors.



(a) (b) (c)

Fig. 3. Stages of the line detection. (a) Adjacent segments of similar color and length are connected. (b) Trapezoid shape fitting and circle center computation. The colors of the trapezoids indicate different shapes. (c) The result of line detection: straight lines, the center circle and line crossings.

3 Object Recognition

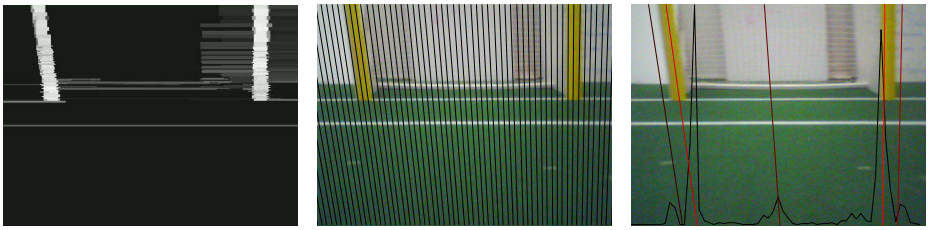
After this preprocessing step, distinct object recognition detectors need to be developed. We have created three detectors for lines, goal posts, and the ball.

3.1 Line Recognition

The line recognition is mainly based on region growing. To improve performance this is done in two steps. First, all segments are traversed once, connecting adjacent segments in adjacent lines if their color difference and length ratio is below a threshold. This can be done very fast as the segments are traversed sequentially in a single pass. It narrows down the potential segments processed in the actual region growing, as two segments must fulfill those properties to belong to the same region. The result of this step is shown in Figure 3a. Instead of considering the absolute color of the segment, this considers color difference only, so that the connected components do not only include lines but all types of objects.

The adjacent segments are then connected to actual regions, the average color of which is compared to the color prototype[1]. The result is a set of arbitrarily shaped whitish regions. To identify line segments among those region, the shape of the regions is taken into account. A line in world space projects to a line in the image [7], and therefore a field line, which is bordered by two lines, projects to a trapezoid in general. A trapezoid shape fitting is then applied to the regions, which basically consists of two simultaneous linear regressions for the segments' slope and width along the image's y-axis. In particular, the regions are split as soon as they deviate too much from the trapezoid shape, which is particularly useful to split the arc-like shaped regions of the center circle into short straight line segments. The result of this step is visualized in Figure 3b. The resulting regions are then assessed based on the shape, exploiting that the expected shape

[1] The color prototypes for the different object classes are currently calibrated manually, as an approximate calibration is sufficient.



(a) (b) (c)

Fig. 4. Goal detection. (a) Visualization of segment rating. (b) Layout of the Hough space. (c) Resulting Hough responses and 5 highest local maxima (the colors of the lines indicate the order of the maxima).

of a line can be computed based on the known camera pose. The different categories for the shapes are *line segments*, *center circle segments* and *non-line segments*. This step reliably sorts out white segments in other robots, since they have a distinctive shape.

After potential line segments have been determined, a simple sanity check tests whether the segments next to the line segment are sufficiently green by comparison with the corresponding color prototype. Then, the line segments are projected onto the field plane for further processing under consideration of the rolling shutter effect (cf. [15]). Collinear line segments are merged by clustering in the Hesse normal form parameter space. Afterwards, the center circle is detected. Therefore, line segments that potentially lie on the center circle are determined based on their length and shape. Assuming that those segments are tangential to the center circle, hypothetical center points are built by computing two points orthogonal to the segment in a distance of the known center circle radius (see Figure 3b). Afterwards, those points are clustered, and if a sufficiently large cluster with at least three supporting segments is found, its centroid is treated as the center circle's center point. Those segments are disregarded in further processing. Finally, field line intersections are computed based on the line segments found, since they are valuable features for the self-localization. The result of the line detection including the center circle and intersections is visualized in Figure 3c.

3.2 Goal Recognition

Due to the complex shape of the goal as a whole, goal detection is simplified to detecting the goal posts; the goal bar only serves as an indicator for the laterality and as a sanity check. The general idea for detecting goal posts is a combination of Hough transformation [4] and region growing. This is related to *histogram-based recognition*, as presented in [19], extended to work with the segments generated in the preprocessing step. The first step is to assign a rating to the segments, i.e. the color difference (sum of channel-wise color differences) to the color prototype, as visualized in Figure 4a. This is implemented efficiently exploiting the MMX instruction set extension available on NAO's CPU.

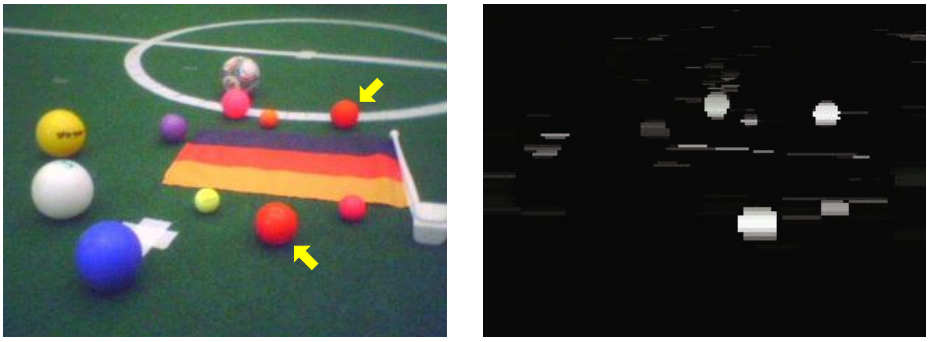
The rated segments are accumulated in a one-dimensional Hough space. The image is split into bins (as a compromise of detection quality and processing time we use 120 bins), the borders of which are lines passing through the vanishing point of the world's vertical axis (cf. Figure 4b). This is motivated by the prior knowledge that the goal posts are cylindrical objects oriented parallel to the world's z-axis. A linearized version of the motion compensation presented in [15] is applied to the Hough space to compensate for the rolling shutter effect. The Hough space is filled by iterating over the segments, determining the bin that corresponds to the center of the segment, and adding its rating to the bin. If a goal post is present in the image, its segments have a high rating and accumulate in few bins, resulting in distinct peaks in the Hough space, as seen in Figure 4c. Finally, the Hough space is smoothed with a 1D Gaussian filter to decrease noise, and the $N = 5$ highest local maxima are determined. No hard thresholds (besides the number of local maxima) have been involved so far to increase robustness.

After promising parts of the image have been identified, region growing is applied. Therefore, the local maxima are processed separately. First, the segments belonging to a local maximum bin are determined. Those are traversed from top to bottom to build regions of similar color (the same similarity criterion as in the run length encoding step is used). To identify goal posts among the regions, several sanity checks are applied. First, it is tested whether the region exceeds a certain minimum height (30 pixels) to ensure not to process arbitrary artifacts. Then, the average color of the region is compared to the color prototype of the goal. Afterwards, the average color of the region below is compared to the color prototype of the field because the goal post resides on the field plane. Then, the relative position of the (potential) goal post is computed based on the border between those two regions. Based on the relative position, the potential goal post is projected into the image, and the projected width is compared to the actual width of the region. Finally, the top of the goal post region is checked for the goal bar by examining the uppermost segments resp. those next to them, which can be used to determine the laterality of the goal post.

After a set of goal posts has been determined, it is finally checked whether the constellation of goal posts is valid, i.e., no goal posts of same laterality, no differently colored goal posts, at most two goal posts.

3.3 Ball Recognition

Ball detection is based on region growing using the formerly detected segments, and a subsequent post-processing step. A simple approach for region growing would be to use all segments for initiating a region. This however, has two major drawbacks regarding the processing time. First, the processing time would be too high if no preselection is performed, and secondly, the variance of the processing time would be very high, since it would highly depend on the actual image contents. Taking this into consideration, the first step of the ball detection is a selection of the segments that are used as a seed for the actual region growing.



(a) (b)

Fig. 5. Preprocessing for ball detection. (a) Test scene including real SPL balls (marked with arrows) along with other objects. (b) The resulting weighted segments. The two actual balls get the highest rating.

This step rates segments based on two measures: the first measure is the sum of the channel-wise color differences to the color prototype for the ball color. The second measure is the deviation of a segment's length from the projected diameter of the ball at that position. This way the center segment of the actual ball (if it is in the image) should get the highest rating. This step does not invoke any hard thresholds, but sorts the segments by an objective measure. The result of the step is visualized in Figure 5 using a complex test scene. It can be seen in Figure 5b that the highest rating is assigned to the actual RoboCup SPL balls. The other objects are downgraded because of the size and/or the color. The number of segments processed is bound to 1,500 beginning at the lower boundary of the image to limit the maximum processing time and its variance.

The $N = 5$ best-rated segments are then used as a seed for region growing. The difficulties that arise from the shades and reflections on the ball motivates the use of an alternative color space for the segmentation that focuses on the color *hue*, similar to the HSV or HSL color space [20]. We decided against HSV because the conversion of YUV to HSV is quite expensive due to the use of trigonometric functions resp. case distinction. Also, a lookup table is not well suited due to its space requirement of 16M entries, which makes cache misses very likely. Therefore, a simpler approximation is used. The color model used consists of four (partly redundant) channels y', s, cb', cr', which are defined as follows:

$$y' = y \tag{2}$$

$$s = \sqrt{cb^2 + cr^2} \tag{3}$$

$$cb' = \frac{cb}{s} \tag{4}$$

$$cr' = \frac{cr}{s} \tag{5}$$

The y' channel is the same as the original y channel. s is comparable to the saturation channel of HSV. cb' and cr' can be seen as normalized color channels, which together have similar properties as the hue channel of HSV. The advantage of this color space is twofold. Firstly, it allows for the effective use of a lookup table, since the conversion function is now $\mathbb{R}^2 \to \mathbb{R}^3$, yielding a lookup table of 64K entries, which significantly benefits from caching. Secondly,

the cyclic property of the hue [20] is avoided, simplifying the actual comparison. Also, the introduction of an additional color channel does not have a negative influence on the processing time because of the implementation using MMX instructions, where it makes no difference whether three or four channels are processed simultaneously.

Based on this color model, the region growing takes place. The segments above and below are processed. To check whether a segment is added to the region, the color difference of the segment's color to the average color of the region is computed and compared to thresholds for each channel. For the ball detection it is reasonable to choose a high threshold for the saturation and luminance considering the shading on and reflections of the ball, and a low threshold for the (normalized) color channels to exploit the uniform color of the ball. Also, due to the known size of the ball, the region growing process can be terminated early if the bounding box of the region exceeds the projected size of the ball. This process is repeated for the $N = 5$ best segments found in the preprocessing stage.

After some potential ball regions have been found, they are post-processed to find the actual ball. First, the average color is compared to the color prototype. Then, the convex hull is determined. Based on the convex hull, a circle fitting is performed to reduce the region found to a circle. We use the *algebraic fit* since it has a closed form solution and the results are neglectably worse compared to the *geometric fit* in this scenario [3]. Due to the convex hull, the circle fitting also handles partially visible balls properly. The RMS error of the data points in the circle fitting is used as a sanity check, since it indicates the "roundness" of the region. Based on the circle center found, the robot-relative position is computed by intersecting the viewing ray with the plane that is one ball radius above the field. Afterwards, the (hypothetical) ball is projected back into the image, and the radius of the projection is compared to the radius found as another sanity check. The first potential ball region that passes all sanity checks is finally selected as the ball.

4 Experiments and Results

The focus of the experiments conducted is twofold. On one hand, the perception rate and quality are of interest. On the other hand, since the processing power of the NAO V3.2 is limited and the resulting system is supposed to be real-time capable, the run time of the modules is examined in greater detail.

For the evaluation of the perception rate and quality, different log files were created and manually annotated, which allows for an automated evaluation. The log files contain 190 frames on average, and were created while the default robot soccer behavior was executed (approaching the ball and kicking). To simulate different lighting conditions, three different scenarios were created, each of which used different light sources, as seen in Figure 6. For comparison purposes the same set of log files was evaluated with a well-established color-table-based vision system that was used by the SPL world champion 2009–2011 in all competitions [16]. For each scene a specialized color table was created which was evaluated

Fig. 6. Different lighting conditions used for the evaluation

Table 1. Detection rate and quality of the newly developed vision system compared to a color-table-based system. The right part of the table represents the color-table-based system. The table shows the true positive rate (TP), the false positive rate (FP) and the average error in image coordinates (err.).

	new system			**matching color table**			**non-matching color table**		
percept type	TP	FP	err.	TP	FP	err.	TP	FP	err.
ball	86.4 %	0.3 %	1.538 px	77.1 %	0 %	1.292 px	68.1%	0.2 %	1.328 px
goal post	84.3 %	0.5 %	2.18 px	81.1 %	0 %	1.377 px	7.7 %	6.4 %	2.223 px
lines	19 %	0.4 %	0.94 px	44.7 %	2.9 %	1.524 px	6 %	3.1 %	1.193 px
intersections	6.7 %	2.7 %	2.517 px	18.6 %	1 %	4.456 px	1.4 %	0 %	3.77 px
center circle	64.5 %	1.8 %	5.323 px	75 %	0.5 %	12.635 px	6 %	3.1 %	1.193 px

with each log file to simulate changing lighting conditions. The new system used a single configuration for all scenes.

The results are summarized in table 1. The evaluation is split into three sections: the newly developed system, the color-table-based system using matching color tables (representing static illumination), and the color-table-based system using non-matching color tables[2] (representing changing lighting conditions). The line detection rate refers to the coverage of lines rather than the line count. The performance of the new system is comparable to the color table system under static lighting conditions; the ball and goal rate is slightly better whereas the line detection has a lower detection rate but is more reliable[3]. The average error of both systems is adequate and similar. Under changing illumination, however, the new system significantly outperforms the color-table-based system with regard to every percept type. The color-table-based system is basically unusable in this scenario except for the ball detection.

Besides the perception *quality*, the run time was evaluated as well. It was measured when the robot was standing while the head control was active, which ensures representative diversity in the camera images that are processed. Again, the new system as well as the color-table-based system are evaluated. The results are shown in table 2. As expected, the run time of the new system is higher compared to the color-table-based system. However, the total run time of approx.

[2] Non-matching color tables refer to color tables that were created for a different lighting situation than they are used in.

[3] Note that the overall line detection rate is low because all lines, even those barely perceivable for a human, were marked for reference.

Table 2. Runtime of the new system in comparison with the color-table-based system

	new system		**CT based system**	
	mean	std. dev.	mean	std. dev.
preprocessing	4.605 ms	0.077 ms	2.65 ms	0.565 ms
ball detection	0.342 ms	0.05 ms	0.05 ms	0.035 ms
goal detection	0.448 ms	0.036 ms	0.085 ms	0.048 ms
line detection	0.6 ms	0.151 ms	0.345 ms	0.092 ms
total	5.995 ms	0.314 ms	3.13 ms	0.74 ms

6 ms is still comparably low and especially allows for real-time operation. More importantly, the variance of the new system is significantly lower, despite the higher total run time. The latter is a crucial property for real-time systems, as it must be able to perform the task in a given time regardless of the actual input.

5 Conclusion and Outlook

We presented an object recognition system based on region growing. Hard decisions based on the color are delayed to the end of processing to increase robustness. Using different modifications and optimizations it is able to process camera images in real-time on the SPL NAO. In our evaluation, we showed that its performance is comparable under static lighting conditions, and far superior under changing lighting conditions compared to a color-table-based system.

Further work on this system may include a robot recognition module to enable reliable obstacle avoidance. To cope with the deficiency in the line detection, the preprocessing step could be modified to scan the image vertically instead of horizontally, which would ease and improve the detection of (almost) horizontal line segments. Due to the comparably low number of parameters compared to a color table, an automatic parameter optimization is very promising to further improve the detection rate. Finally, the system could easily be transferred to other RoboCup humanoid leagues, as they provide similar preconditions such as an upright upper body and a color-coded environment.

Acknowledgements. The authors would like to thank the RoboCup team *B-Human* for providing the software basis of the developed system, as well as the color-table-based vision system used for comparison. Special thanks go to Tim Laue and Udo Frese for supervising respectively reviewing the thesis this paper is based upon. This work has been partially funded by DFG through SFB/TR 8 “Spatial Cognition”.

References

1. Bruce, J., Balch, T., Veloso, M.: Fast and Inexpensive Color Image Segmentation for Interactive Robots. In: 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2061–2066 (2000)
2. Canny, J.: A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence (6), 679–698 (1986)

3. Chernov, N., Lesort, C.: Least Squares Fitting of Circles. Journal of Mathematical Imaging and Vision 23(3), 239–252 (2005)
4. Duda, R.O., Hart, P.E.: Use of the Hough Transformation To Detect Lines and Curves in Pictures. Communications of the ACM 15(1), 11–15 (1972)
5. Gevers, T., Smeulders, A.W.M.: Color-based object recognition. Pattern Recognition 32(3), 453–464 (1999)
6. Härtl, A.: Robuste, echtzeitfähige Bildverarbeitung für einen humanoiden Fußballroboter. Master's thesis, Universität Bremen (2012)
7. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press (2004)
8. Hojjatoleslami, S.A., Kittler, J.: Region Growing: A New Approach. IEEE Transactions on Image Processing 7(7), 1079–1084 (1998)
9. Jähne, B.: Digital Image Processing, 6th edn. Springer (2005)
10. Jain, R.C., Kasturi, R., Schunck, B.G.: Machine vision. McGraw-Hill (1995)
11. Jamzad, M., Sadjad, B.S., Mirrokni, V.S., Kazemi, M., Chitsaz, H., Heydarnoori, A., Hajiaghai, M.T., Chiniforooshan, E.: A Fast Vision System for Middle Size Robots in RoboCup. In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 71–80. Springer, Heidelberg (2002)
12. Otsu, N.: A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics 9(1), 62–66 (1979)
13. Press, W., Teukolsky, S., Flannery, B., Vetterling, W.: Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press (1992)
14. Reinhardt, T.: Kalibrierungsfreie Bildverarbeitungsalgorithmen zur echtzeitfähigen Objekterkennung im Roboterfußball. Master's thesis, Hochschule für Technik, Wirtschaft und Kultur Leipzig (2011)
15. Röfer, T.: Region-Based Segmentation with Ambiguous Color Classes and 2-D Motion Compensation. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007. LNCS (LNAI), vol. 5001, pp. 369–376. Springer, Heidelberg (2008)
16. Röfer, T., Laue, T., Müller, J., Fabisch, A., Feldpausch, F., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Humann, A., Honsel, D., Kastner, P., Kastner, T., Könemann, C., Markowsky, B., Riemann, O.J.L., Wenk, F.: B-Human Team Report and Code Release 2011 (2011), `http://www.b-human.de/downloads/bhuman11_coderelease.pdf`
17. Scharstein, D., Szeliski, R.: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. International Journal of Computer Vision 47(1-3), 7–42 (2002)
18. Swain, M.J., Ballard, D.H.: Color Indexing. International Journal of Computer Vision 7(1), 11–32 (1991)
19. Volioti, S., Lagoudakis, M.G.: Histogram-Based Visual Object Recognition for the 2007 Four-Legged RoboCup League. In: Darzentas, J., Vouros, G.A., Vosinakis, S., Arnellos, A. (eds.) SETN 2008. LNCS (LNAI), vol. 5138, pp. 313–326. Springer, Heidelberg (2008)
20. Zhang, C., Wang, P.: A New Method of Color Image Segmentation Based on Intensity and Hue Clustering. In: 15th International Conference on Pattern Recognition, vol. 3, pp. 613–616 (2000)
21. Zucker, S.W.: Region Growing: Childhood and Adolescence. Computer Graphics and Image Processing 5(3), 382–399 (1976)