B-Human 2016 – Robust Approaches for Perception and State Estimation Under More Natural Conditions

Thomas Röfer[1,2(✉)], Tim Laue[2], and Jesse Richter-Klug[2]

[1] Deutsches Forschungszentrum für Künstliche Intelligenz, Cyber-Physical Systems, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
thomas.roefer@dfki.de
[2] Fachbereich 3 – Mathematik und Informatik, Universität Bremen, Postfach 330 440, 28334 Bremen, Germany
tlaue@uni-bremen.de

Abstract. In 2015 and 2016, the RoboCup Standard Platform League's major rule changes were mostly concerned with the appearance of important game elements, changing them towards a setup that is more similar to normal football games, for instance a black and white ball and white goals. Furthermore, the 2016 *Outdoor Competition* was held in a glass hall and thus under natural lighting conditions. These changes rendered many previously established approaches for perception and state estimation useless. In this paper, we present multiple approaches to cope with these challenges, i. e. a color classification for natural lighting conditions, an approach to detect black and white balls, and a self-localization that relies on complex field features that are based on field lines. This combination of perception and state estimation approaches enabled our robots to preserve their high performance in this more challenging new environment and significantly contributed to our success at RoboCup 2016.

1 Introduction

B-Human is a joint RoboCup team of the University of Bremen and the German Research Center for Artificial Intelligence (DFKI). The team was founded in 2006 as a team in the Humanoid League, but switched to participating in the Standard Platform League in 2009. Since then, we participated in seven RoboCup German Open competitions, the RoboCup European Open, and eight RoboCups and only lost four official games. As a result, we won all German Open and European Open competitions, the RoboCups 2009, 2010, 2011 and 2013. This year, we won the main (indoor) competition and became the runner-up in the newly introduced outdoor competition.

The rules of the competition are changed every year to make the task more challenging and to work towards the RoboCup Federation's 2050 goal. In the past two years, these changes mostly concerned the appearance of the environment, making it look more like a human football environment. In 2015, the yellow goals were replaced by white ones and it 2016, a black and white ball

S. Behnke et al. (Eds.): RoboCup 2016, LNAI 9776, pp. 503–514, 2017.
https://doi.org/10.1007/978-3-319-68792-6_42

replaced the previously used orange one. In addition, over both years, almost all restrictions regarding jersey colors have been removed, too. This combination of changes has made simplistic and purely color-based approaches such as "detect the orange spot" or "find yellow rectangles" useless as most unique color assignments do not exist anymore. Furthermore, to evaluate the performance under natural lighting, the *Outdoor Competition 2016* was held in a glass hall, requiring the adaptiveness of image preprocessing algorithms. In addition, this competition had one more challenging aspect: walking on artificial grass. We solved all vision-related challenges sufficiently well, but were less successful regarding a robust walking implementation, which ultimately resulted in losing the outdoor final. However, we still scored more goals (18) outdoors than all of our seven competitors together (17).

This paper mainly focuses on our vision system and the impact it had on ball localization and self-localization. The robustness of these components strongly contributed to our success. For instance, we were the only team in the competition that never got a *leaving the field* penalty.[1] In comparison, the average number of *leaving the field* calls was 35.45 times per team (5.8 per team per game), which either meant that the robots were delocalized or they were chasing a false ball they detected outside the field.

This paper is organized as follows: Sect. 2 describes our image preprocessing approach, which is capable of handling natural lighting, followed by the algorithms required to detect the new black and white ball in Sect. 3. Finally, the new complex field features, which make the perception of the white goals unnecessary, and their impact on self-localization are presented in Sect. 4.

2 Image Preprocessing

In the Standard Platform League, all teams use the same robot model, the *SoftBank Robotics NAO*. The NAO is equipped with two cameras – one located in the forehead and one in the chin – that are the major source of information about the environment. Processing these images is the most time-consuming task performed on the robot, because they consist of a large amount of data. Before 2016, the basic principle of our vision system was to reach real-time performance, i. e. to process 60 images per second, by analyzing only a fraction of the pixels available. The selection was based on the perspective under which the camera that took the image observed the environment and the expected size objects would have in the different parts of the image. This approach is still followed in our current system, but there are now preprocessing steps that consider the whole image. To keep the real-time performance, the amount of data to process was reduced by using a smaller image resolution and processing was accelerated by employing the SIMD (*single instruction multiple data*) instructions of NAO's processor.

[1] Actually, we got two, but both were the result of human errors, as we confirmed from analyzing video footage and log files.

2.1 NAO's Camera Images

NAO's cameras provide images in the YUV 4:2:2 color space. In this format, two neighboring pixels have separate brightness values (Y), but share the same color information (U and V). This format is a little bit cumbersome to handle, because it always requires a distinction between whether a pixel is the left or the right one of a pair that share the color channels. Therefore in the past, we have acquired images that were twice the resolution than we actually needed. We interpreted two neighboring pixels as a single one, ignoring the second Y channel, and skipped every second row to keep the aspect ratio. As a result, the images were used as if their format would be YUV 4:4:4. However, for using SIMD instructions, it is important to process as much data as possible with a single instruction. Therefore, having an unused Y channel is not desirable. In addition, color information has become less and less important in the Standard Platform League, because most color coding was removed from the setup during the recent years. Hence, it is important to keep the resolutions we used before for brightness information, i. e. 640×480 pixels for the upper camera and 320×240 pixels for the lower one, but the color information can be sparser.

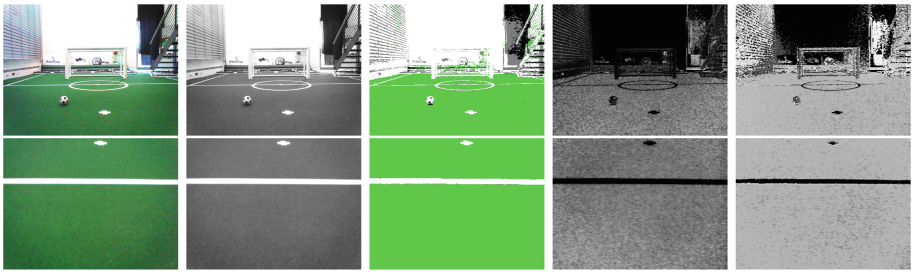
2.2 The YHS2 Image Format

As a result, the YUV 4:2:2 format now appears to be a reasonable compromise, which reduces the amount of data to process by a factor of two. However, its data layout is still impractical. Therefore, the original image (cf. Fig. 1a) is split into two dedicated images: a gray-scale image (cf. Fig. 1b) and a color-classified image (cf. Fig. 1c). The color-classified image assigns one of the following classes to each pixel: *field*, *white*, *black*, and *other*[2]. These two images are used by all further image processing steps instead of the original image. Both images are generated together in a single pass using the SSE3[3] instruction set of NAO's Intel Atom CPU. The gray-scaled image simply consists of all Y values. The color-classified image needs the actual color classification as well as an optional color conversion. A suitable candidate for lighting-independent color classification is the HSI (*hue*, *saturation*, *intensity*) color space. Unfortunately, an implementation of a correct conversion to the HSI color space takes about 7 ms for the whole upper image. This is why we implemented a color conversion to the *YHS2* format[4] instead. It follows roughly the same idea, but requires significantly less computation time. In *YHS2*, a vector that is created from the U and V channels describes the *H*ue (cf. Fig. 1e) as its angle as well as the *S*aturation (cf. Fig. 1d) as its length divided by the luminance component *Y*. The color classification is performed in two steps. First, it is decided, whether a pixel is saturated or not. If the hue value of a saturated pixel is inside a specific range (normally something greenish), it is classified as *field*, otherwise as *other*. An unsaturated pixel is categorized by

[2] Saturated, but not green, i. e. not the field color.

[3] Unfortunately, the AVX extensions are not supported by the NAO's CPU.

[4] The *YHS2* conversion is inspired by a discussion found in the internet at http://forum.doom9.org/showthread.php?t=162053.



(a) original (b) gray-scale (c) color classes (d) saturation (e) hue

Fig. 1. Examples of upper and lower camera images in different formats (Color figure online)

its Y value as either being *black* or *white*. The whole classification takes 4 ms for the upper camera image and only 1 ms for the lower one.

2.3 Basic Feature Detection

The colored images are used by the subsequently executed modules to find initial cues – the so-called *spots* – that indicate interesting features. In a first step, the images are subsampled by traversing precomputed scan-lines to build regions of the same color class. In advance, an initial coarse grid is used to detect the field's boundary, i. e. the polygon that encompasses the green field. This is done to avoid further calculations within areas outside the field (where we do not expect any objects relevant for the game).

After regions have been found, the detection of the lines – which provide the base for all field elements that we currently consider – is realized by fitting white regions via linear regression over their field coordinates. In parallel it is also tried to fit a mid circle into the field coordinates of sets of short white line elements. For this purpose, linear regression is used, too. In case of field lines hitting or crossing each other in an approximately right angle, a field line intersection is detected. Each intersection is either classified as L, T, or X, according to its appearance.

Finally, penalty marks are detected by searching for small white areas which are surrounded by field color. If too much black is found inside the area, it will be discarded, as it might also be a ball.

3 Detecting the Black and White Ball

The introduction of the black and white ball is the major new challenge in the Standard Platform League in 2016. Until the RoboCup 2015, the ball was orange and rather easy to detect. In particular, it was the only orange object on the field. The new ball is mainly white with a regular pattern of black patches, just as a miniature version of a regular soccer ball. The main problem is that the

field lines, the goals, and the NAO robots are also white. The latter even have several round plastic parts and they also contain grey parts. Since the ball is often in the vicinity of the NAOs during a game, it is quite challenging to avoid a large number of false positives.

Playing with a normal soccer ball has also been addressed in the Middle Size League (MSL), e. g. [3,5]. However, the robots in the MSL are typically not white and they are equipped with more computing power than the NAO is, e. g. Martins et al. [8] presented a ball detection that requires 25 ms at a resolution of 640 × 480 pixels on a Intel Core 2 Duo 2 running at 2 GHz. In contrast, the solution presented here is on average more than ten times faster running on an Intel Atom at 1.6 GHz, which allows our robots to process all images that their cameras take.

We use a multi-step approach for the detection of the ball. First, the vertical scan lines our vision system is mainly based on are searched for ball candidates. Then, a contour detector fits ball contours around the candidates' locations. Afterwards, fitted ball candidates are filtered using some general heuristics. Finally, the surface pattern inside each remaining candidate is checked. Furthermore, the ball state estimation has been extended by some additional checks to exclude false positives that cannot be avoided during image processing.

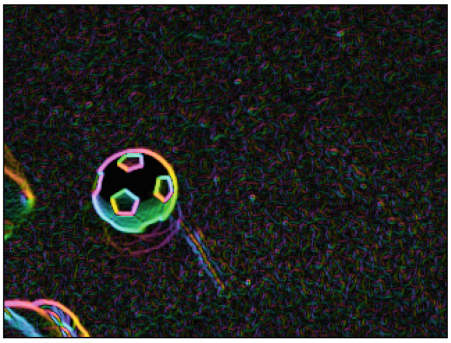
3.1 Searching for Ball Candidates

Our vision system scans the image vertically using scan lines of different density based on the size that objects, in particular the ball, would have in a certain position of the image. To determine ball candidates, these scan lines are searched for sufficiently large gaps in the green that also have a sufficiently large horizontal extension and contain enough white (cf. Fig. 2a). Candidates that are significantly inside of a detected robot are discarded. In addition, the number of candidates is reduced by only accepting ones that are sufficiently far away from other candidates.

3.2 Fitting Ball Contours

As the position of a ball candidate is not necessarily in the center of an actual ball, the area around such a position is searched for the contour of the ball as it would appear in this part of the image given the intrinsic parameters of the camera and its pose relative to the field plane. The approach is very similar to the detection of objects in 3-D space using a stereo camera system as described by Müller et al. [9], but we only use a single image instead. Thereby, instead of searching a 3-D space for an object appearing in matching positions in two images at the same time, only the 2-D plane of the field is searched for the ball to appear in the expected size in a single camera image. For each ball candidate, a contrast-normalized Sobel (CNS) image of the surrounding area is computed (cf. Fig. 2b). This contrast image is then searched for the best match with the expected ball contour (cf. Fig. 2c). The best match is then refined by adapting its hypothetical 3-D coordinates (cf. Fig. 2d).

(a) Vertical scan lines and a detected ball candidate (the cross). Parts of the robot's body are ignored (bottom left).



(b) Contrast-normalized Sobel image. The colors indicate the directions of the gradients.

(c) Visualization of the search space for the ball contour. The actual search is only performed around the ball candidate, but in single pixel steps in both dimensions.

(d) The contour with the highest response (green) and the sample grid to check the ball pattern (pixels classified as black are shown in red, white pixels in blue).

Fig. 2. The main steps of the ball detection (Color figure online)

3.3 Filtering Ball Candidates

The fitting process results in a measure, the *response*, for how well the image matches with the contour excepted at the candidate's location. If this value is below a threshold, the ball candidate is dropped. The threshold is dynamically determined from the amount of green that surrounds the ball candidate. On the one hand, the less green is around the candidate, the higher the response must be to reduce the amount of false positives inside robots. However, if a ball candidate is completely surrounded by green pixels and the response was high enough to exclude the possibility of being a penalty mark, the ball candidate is accepted right away, skipping the final step described below that might be failing if the ball is rolling quickly. All candidates that fit well enough are processed in descending order of their response. As a result, the candidate with the highest

response that also passes all other checks will be accepted. These other checks include that the ball radius found must be similar to the radius that would be expected at that position of the image.

3.4 Checking the Surface Pattern

For checking the black and white surface pattern, a fixed set of 3-D points on the surface of the ball candidate are projected into the image (cf. Fig. 2d). For each of these pixels, the brightness of the image at its location is determined. Since the ball usually shows a strong gradient in the image from its bright top to a much darker bottom half, the pixels are artificially brightened depending on their position inside the ball. Then, Otsu's method [10] is used to determine the optimal threshold between the black and the white parts of the ball for the pixels sampled. If the average brightnesses of both classes are sufficiently different, all pixels sampled are classified as being either black or white. Then, this pattern is looked up in a pre-computed table to determine whether it is a valid combination for the official ball. The table was computed from a 2-D texture of the ball surface considering all possible rotations of the ball around all three axes and some variations close the transitions between the black and the white parts of the ball.

3.5 Removing False Positives Before Ball State Estimation

The major parts of B-Human's ball state, i. e. position and velocity, estimation remained unchanged for many years and consist of a set of Kalman filters. However, the introduction of the new black and white ball required the addition of a few more checks. In previous years, the number of false positive ball perceptions has been zero in most games. Hence, the ball tracking was implemented as being as reactive as possible, i. e. every perception was considered. Although the new ball perception is quite robust in general, several false positives per game cannot be avoided due to the similarity between the ball's shape and surface and some robot parts. Therefore, there must be multiple ball perceptions within a certain area and within a maximum time frame before a perception is considered for the state estimation process. This slightly reduces the module's reactivity but is still fast enough to allow the execution of ball blocking moves in a timely manner. Furthermore, a common problem is the detection of balls inside robots that are located at the image's border and are thus not perceived by our software. A part of these perceptions, i. e. those resulting from our teammates, is excluded by checking against the communicated teammate positions.

3.6 Results

The approach allows our robots to detect the ball in distances of up to five meters with only a few false positive detections. Figure 3 shows the statistics of how well the ball was seen by different teams in terms of how long ago the ball was seen

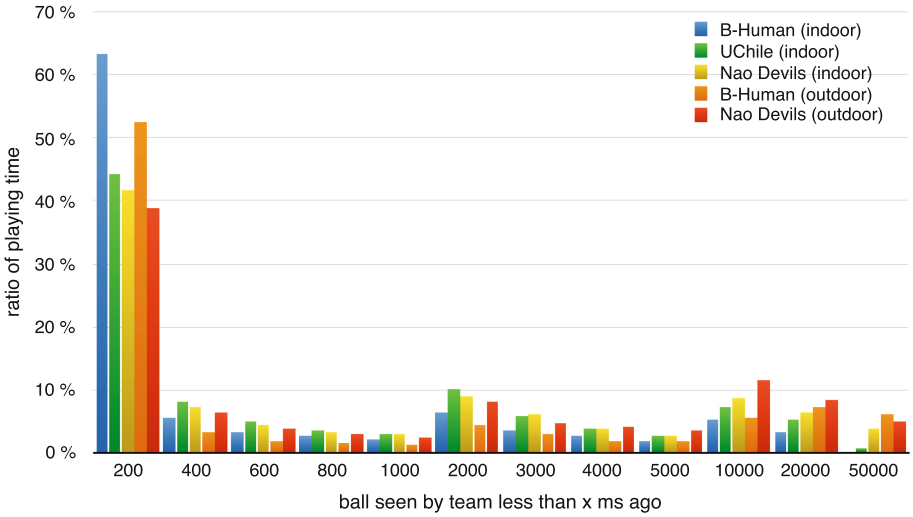


Fig. 3. Analysis of the team communication data of the indoor semifinal B-Human vs. UChile Robotics Team, the quarterfinal Nao Devils Dortmund vs. UT Austin Villa (UT Austin Villa is missing in this chart, because they did not broadcast their team communication.), and the outdoor final B-Human vs. Nao Devils Dortmund (Color figure online)

by the team, i. e. the robot that saw it most recently. The statistics was created from some of the log files recorded by the TeamCommunicationMonitor [11] at RoboCup 2016 that were made available at the website of the league. Since the teams analyzed were some of the best in the competition[5], it is assumed that the number of false positive ball detections, which would also result in low numbers, is negligible. Although the chart in Fig. 3 suggests that the ball detection worked better indoors, it actually benefited from the good lighting conditions in the outdoor competition. However, since our robots were only walking slowly and fell down quite often, the average distance to the ball was a lot higher, which impeded the perception rate. In a rather dark environment, as on Field A in the indoor competition, balls with lower responses had to be accepted in order to detect the ball at all. This resulted in more false positive detections, in particular in the feet of other robots, because they are also largely surrounded by green.

The runtime is determined by the number of ball candidates that are found. For instance, the log file of player number 2 from the second half of the final shows that the search for ball candidates took 0.135 ms on average and took never longer than 0.816 ms. Computing the CNS image for the candidates took 0.285 ms on average and reached a maximum of 5.394 ms. Checking these candidates took 0.951 ms on average, but sometimes took significantly longer.

[5] The third-placed Nao-Team HTWK was not analyzed, because they only provided binary information in the standard communication's field *ballAge*.

The maximum duration reached was 10.604 ms. As it rarely happens that the processing of images from the upper and the lower camera take long in subsequent frames, the frame rate was basically 60 Hz all the time.

4 Complex Field Features and Self-localization

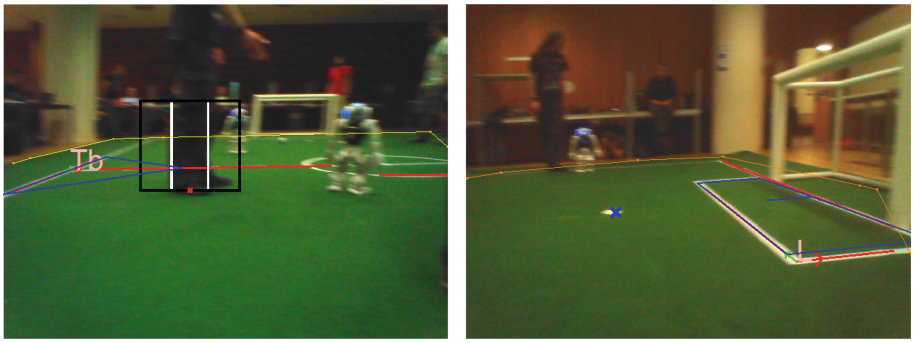
In the past, B-Human used goals as a dominant feature for self-localization. When the field was smaller and the goal posts were painted yellow, they were easy to perceive from most positions and provided precise and valuable measurements for the pose estimation process. In particular the sensor resetting part, i. e. the creation of alternative pose estimates in case of a delocalization, was almost completely based on the goal posts perceived. In 2015, we still relied on this approach, using a detector for the white goals [11]. However, as it turned out that this detector required too much computation time and did not work reliably in some environments (requiring lots of calibration efforts), we decided to perform self-localization without goals but by using complex field features derived from certain constellations of perceived field lines.

4.1 Field Features

The self-localization always used field lines, their crossings, and the center circle as measurements. Since 2015, these features are complemented by the perception of the penalty marks. All these field elements are distributed over the whole field and can be detected very reliably, provided a constant input of measurements in most situations.

Built upon this, the perception of a new category of measurements, the so-called *field features*, has been implemented. They are created by combining multiple basic field elements in a way that a robot pose (in global field coordinates) can be derived directly (the handling of the field symmetry, which leads to actually two poses, is described in the following section). The currently computed features are: the penalty area, the center circle (including the center line that provides the direction), the field corners, the center corners (where the center line touches the outer field lines, cf. Fig. 4a), and the goal frame on the floor. Some of these features can be determined by different field element constellations, e. g. the penalty area can be derived from a subset of its corners as well as from a penalty mark and the penalty area's line next to it (cf. Fig. 4b). All considered lines are preprocessed by classifying them in short and long lines and by determining their relation to the field border (if available). The crossings of the lines are categorized as $L/T/X$ on the one hand and in $big/small$ on the other hand. In this context, *big* means that it is a crossing that results from the intersection of two long lines, such as field corners perceived from a longer distance.

Overall, this approach provides a much higher number of reliable pose estimates than the previous goal-based approach, as the field lines on which it is based can be seen from many perspectives and have a more robust context



(a) Center corner: two long field lines intersect and represent a big T (marked *Tb*).

(b) Penalty area: a penalty mark and a close line allow the detection of this area.

Fig. 4. Two examples for field features, both are depicted as blue lines. (Color figure online)

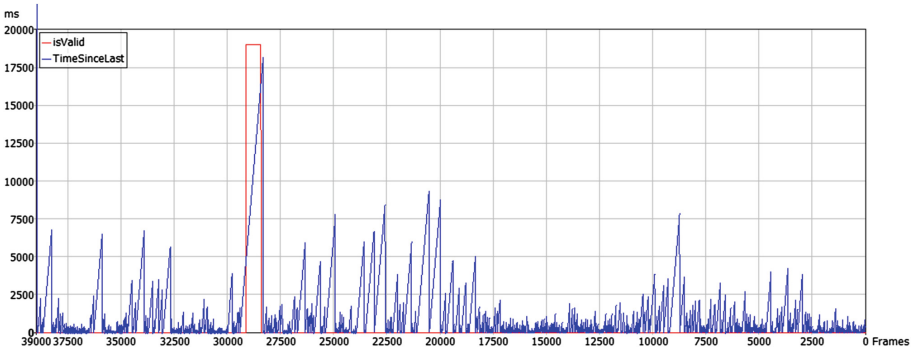


Fig. 5. Detected field features of one robot during the second half of the 2016 SPL final. The blue plot shows the elapsed time since the last feature was detected. The red line marks a period of time during which the robot's camera perspective was invalid, making feature detection impossible. It can be seen that there was never a period of time longer than nine seconds during which no field feature was detected. On average, a field feature was seen every 668 ms. (Color figure online)

(straight white lines surrounded by green carpet) than the noisy unknown background of goal posts. An example of the continuous perception of field features is plotted in Fig. 5.

4.2 Localization Resetting

The self-localization is based on a particle filter [4] with a low number of particles that each include an Unscented Kalman filter (UKF) [6]. Both approaches are straightforward textbook implementations [12], except for some adaptions to handle certain RoboCup-specific game states, such as the positioning after

returning from a penalty. Field features can be used as measurements for these filters but not as a perception of relative landmarks. Instead, an artificial measurement of a global pose is generated, reducing the translational error in both dimensions as well as the rotational error at once. Furthermore, no data association – in contrast to the basic field elements that are not unique – is required.

However, particles only cover the state space very sparsely. Therefore, to recover from a delocalization, it is a common approach to perform *sensor resetting*, i. e. to insert new particles based on recent measurements [7]. The field features provide exactly this information and thus are used by us for creating new particles. As false positives can be among the field features, e. g. caused by robot parts overlapping parts of lines and thereby inducing a wrong constellation of elements, an additional filtering step is necessary. All robot poses that can be derived from recently observed field features are clustered and only the largest cluster, which also needs to contain a minimum number of elements, is considered as a candidate for a new sample. This candidate is only inserted into the sample set in case it significantly differs from the current robot pose estimation.

To resolve the field's symmetry when handling the field features, we use the constraints given by the rules (e. g. all robots are in their own half when the game state switches to *Playing* or when they return from a penalty) as well as the assumption that the alternative that is more compatible to the previous robot pose is more likely than the other one. This assumption can be made, as no teleportation happens in real games. Instead, most localization errors result from situations in which robots lose track of their position and accumulate translational and rotational errors.

Self-localization without goals has already been realized by other teams, starting with Robo Eireann in 2011 [13]. There have also been different solutions for resolving the field's symmetry, e. g. by observing the field's surrounding, an approach that has been used by the two-time world champion UNSW Australia [2] who successfully use a visual compass [1]. However, our recent developments in localization and perception – along with a growing number of robots that have a z-axis gyroscope – enabled us to reduce the number of *Leaving the Field* penalties from 15 (in seven games during RoboCup 2015) to basically zero (in eleven games during the RoboCup 2016). This is a result that – as mentioned in Sect. 1 – significantly outperforms all other teams during a real competition.

5 Conclusion

In this paper, we have presented our vision and state estimation approaches that helped us to cope with the recent changes of the league's environment and significantly contributed to our success at RoboCup 2016. The analysis of data recorded during that competition shows that our robots have been able to frequently see important elements of the game, i. e. the new ball as well as complex field features. This enabled our robots to show great performances during games in the indoor competition as well as in the outdoor competition, in which we scored more goals than all other teams together. Furthermore, in

contrast to all other teams, our robots never accidentally left the field. This indicates a very robust self-localization as well as no false ball positives outside the field. Overall, our system is ready for upcoming competitions, which are supposed to be held under more natural lighting conditions than the past ones.

References

1. Anderson, P., Hengst, B.: Fast monocular visual compass for a computationally limited robot. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS, vol. 8371, pp. 244–255. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44468-9_22
2. Ashar, J., et al.: RoboCup SPL 2014 champion team paper. In: Bianchi, R.A.C., Akin, H.L., Ramamoorthy, S., Sugiura, K. (eds.) RoboCup 2014. LNCS, vol. 8992, pp. 70–81. Springer, Cham (2015). doi:10.1007/978-3-319-18615-3_6
3. Coath, G., Musumeci, P.: Adaptive arc fitting for ball detection in RoboCup. In: APRS Workshop on Digital Image Analyzing, pp. 63–68 (2003)
4. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte-Carlo localization: efficient position estimation for mobile robots. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence, pp. 343–349, Orlando, USA (1999)
5. Hanek, R., Schmitt, T., Buck, S., Beetz, M.: Towards RoboCup without color labeling. In: Kaminka, G.A., Lima, P.U., Rojas, R. (eds.) RoboCup 2002. LNCS, vol. 2752, pp. 179–194. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45135-8_14
6. Julier, S.J., Uhlmann, J.K., Durrant-Whyte, H.F.: A new approach for filtering nonlinear systems. In: Proceedings of the American Control Conference, vol. 3, pp. 1628–1632 (1995)
7. Lenser, S., Veloso, M.: Sensor resetting localization for poorly modelled mobile robots. In: Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000), vol. 2, pp. 1225–1232, San Francisco, USA (2000)
8. Martins, D.A., Neves, A.J., Pinho, A.J.: Real-time generic ball recognition in RoboCup domain. In: Proceedings of the 3rd International Workshop on Intelligent Robotics, IROBOT, pp. 37–48 (2008)
9. Müller, J., Frese, U., Röfer, T.: Grab a mug - object detection and grasp motion planning with the NAO robot. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2012), pp. 349–356, Osaka, Japan. IEEE (2012)
10. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. **9**(1), 62–66 (1979)
11. Röfer, T., Laue, T., Richter-Klug, J., Stiensmeier, J., Schünemann, M., Stolpmann, A., Stöwing, A., Thielke, F.: B-Human team description for RoboCup 2015. In: RoboCup 2015: Robot Soccer World Cup XIX Preproceedings. RoboCup Federation, Hefei, China (2015)
12. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
13. Whelan, T., Stüdli, S., McDonald, J., Middleton, R.H.: Efficient localization for robot soccer using pattern matching. In: Hähnle, R., Knoop, J., Margaria, T., Schreiner, D., Steffen, B. (eds.) ISoLA 2011. CCIS, pp. 16–30. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34781-8_2