B-Human

Team Description for RoboCup 2018

Thomas Röfer[1,2], Tim Laue[2],
Andreas Baude[2], Gerrit Felsch[2], Arne Hasselbring[2],
Bernd Poppinga[2], Felix Thielke[2], Timo Urban[2]

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

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 nine RoboCup German Open competitions, the RoboCup European Open 2016, and in nine RoboCups. We won eight of the German Opens, the European Open, and became world champion six times.

This team description paper is organized as follows: In Sect. 2, we present a method for estimating the different colors on the field. Section 3 describes our new approach to determining the field boundary. Our new framework for efficiently computing neural networks on the NAO robot is outlined in Sect. 4. Section 5 describes the new ball detection that is based on this framework as well as improvements we made in tracking the ball. The modularization of our existing behavior framework is discussed in Sect. 6. Section 7 describes a change to the B-Human streaming architecture that might interest other users of our framework. Our work on the GameController suite this year is outlined in Sect. 8. Finally, we summarize this paper in Sect. 9.

1.1 Team Members

B-Human consists of the following people, most of whom are shown in Fig. 1:

Team Leaders / Staff: Tim Laue, Thomas Röfer.
Students: Andreas Baude, Jan Buschmann, Gerrit Felsch, Jan Fiedler, Marvin Franke, Tryggve Gahrmann, Martin Gerken, Mario Grobler, Paul Luca

Fig. 1: The B-Human team of the 2018 RoboCup season

Habermann, Arne Hasselbring, Jannik Heyen, Markus Ihrig, Jan-Henrik Kasper, Jonah Klöckner, Gregor Kuhn, Philip Reichenberg, Nicole Schrader, Lukas Schulze, Timo Urban, Lars Wimmel.

Alumni: Daniel Krause, Bernd Poppinga, Lukas Post, Jesse Richter-Klug, Enno Röhrig, Alexander Stöwing, Felix Thielke.

Associated Researcher: Udo Frese.

1.2 Publications Since RoboCup 2017

As in previous years, we released our code after the RoboCup 2017 together with a detailed description [10] on our website and on GitHub (`https://github.com/bhuman/BHumanCodeRelease`). To date, we know of 32 teams that based their RoboCup systems on one of our code releases (AUTMan Nao Team, Austrian Kangaroos, BURST, Camellia Dragons, Crude Scientists, Edinferno, GraceBand, JoiTech-SPL, Luxembourg United, Nao Devils, Naova ETS, NimbRo SPL, NTU RoboPAL, RECIFE Soccer, Rinobot, SPQR, TJArk, UChile, Z-Knipsers/NomadZ) or used at least parts of it (Bembelbots, Cerberus, HULKs, MiPal, MRL SPL, Northern Bites, NUbots, RoboCanes, RoboEireann, rUNSWift, UnBeatables, UPennalizers, UT Austin Villa). We also described our team tactics and our approaches to the mixed team competition as well as our path planner in our 2017 Champion Paper [11].

2 Automatic Color Calibration

Currently our vision is based on the computation of a gray-scaled and a color-coded version of the camera image. While the gray-scaled image is computed

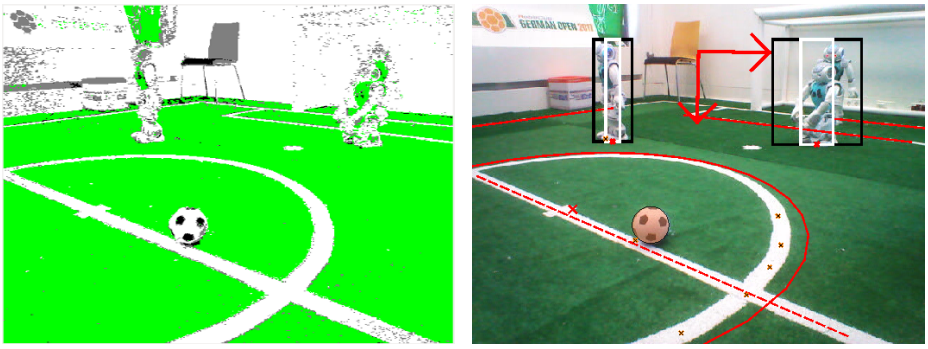


Fig. 2: The color-classified image compared to the original image with recognized field features.

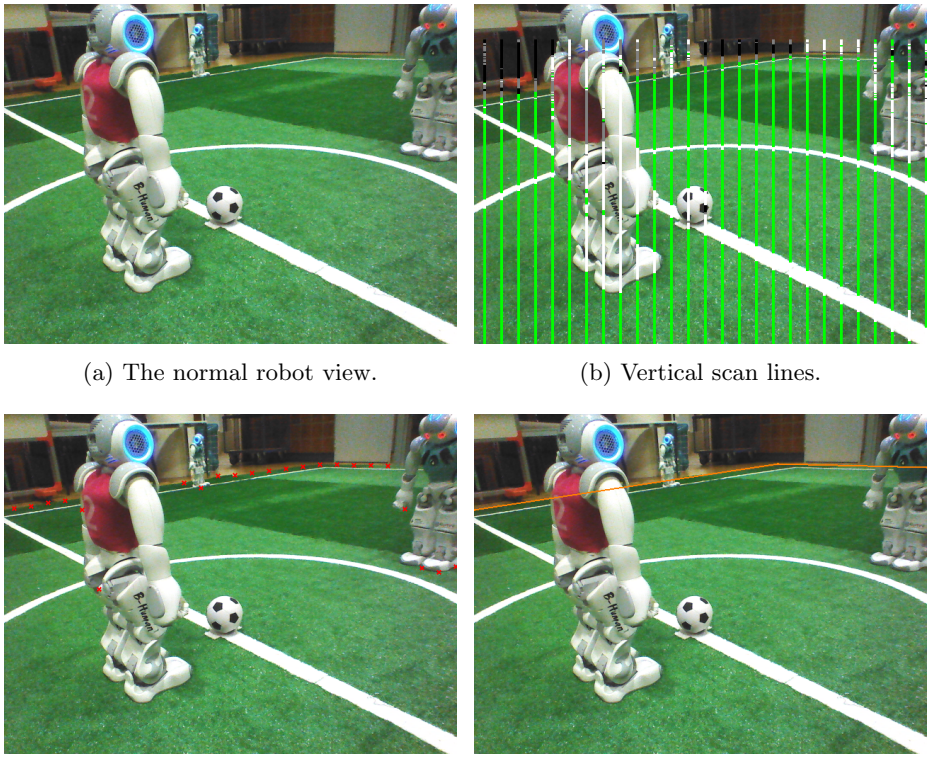
independently, the color-coded version is calibrated manually beforehand using the YHS color space. The values set are the maximum non-color saturation, the black/white delimiter, and the hue range of the field color. To cope with the change of lighting conditions during the game, we chose a new approach that is capable of adapting to these changes. The color classes *black*, *white*, *field color*, and *other* are represented as three dimensional normal distributions in the YHS space as part of a Gaussian mixture model (GMM).

Since expectation-maximization needs to accumulate samples before predicting the normal distributions and prediction is computationally expensive, the adaptation algorithm by Zivkovic [14] is used to update them. Samples are matched to the distributions by two criteria, first the Mahalanobis distance has to be smaller than three times the standard deviation of the distribution and second the Mahalanobis distance of the distribution has to be the minimum distance to any distribution. The GMMs are then updated to match this sample. The weights for all distributions, except the matching ones, are decreased, while their weight is raised and their means and variances are shifted to match the sample. Unfortunately, it is too expensive to update the model with each pixel, therefore scanlines (compare Fig. 3b) are used to identify color regions. Only the middle point of each of these regions is used to update the model.

Figure 2 shows the classification on our test field together with the recognized features drawn into the original image.

3 Field Boundary Detection

All interesting objects, such as the ball and other robots, are located within the field boundary, which is why the robust detection of it has so many advantages. For example, the image only needs to be considered up to the field boundary, saving computing time and avoiding false detections i.e. in the audience and advertisements. On the other hand, it is also important that the detected field



(a) The normal robot view.

(b) Vertical scan lines.

(c) Detected field boundary spots (red crosses).

(d) The detected field boundary with the RANSAC algorithm (orange line).

Fig. 3: The main steps of the field boundary detection

boundary does not cut off too much of the image, as otherwise important image areas would not be scanned.

So far, the field boundary was detected using a convex hull, which was placed through previously determined spots. In concrete terms, these spots are the ends of the most field-like region sequence along vertical scan lines. However, one disadvantage of the convex hull is that it can take on unexpected shapes. Furthermore, robots or other obstacles can cause the field boundary to drop along the object. All in all, the modeling of the field boundary using the convex hull is not optimal.

Since it actually consists only of straight lines that are orthogonal to each other, a RANSAC approach was implemented that uses this previous knowledge about the field boundary.

In the first step, possible field boundary spots are searched for along vertical scan lines, as it was done in our previous approach. In the second step, two lines and their intersection are calculated in each RANSAC iteration. This results in three potential models: the resulting corner and the two individual lines. These

three models are checked based on which one describes the calculated spots best. In the case of two lines, additional conditions apply. The lines projected to the field must be nearly orthogonal to each other, i. e. there must be an angle of 90 degrees between them within a certain tolerance range. Furthermore, the obtuse angle must point to the robot, because otherwise the robot would stand outside the field.

4 Fast Inference of Neural Networks

With the new ball detector using a neural network for classification—and several other experiments utilizing deep learning in the works—it became a necessity to add the possibility of inferring neural networks to the B-Human code base. For this task, other teams have already taken various approaches. As two examples, Nao-Team HTWK linked their code with the Caffe library [8] while the Nao Devils Dortmund programmed a Matlab script to translate the models they train in Matlab to C++ code [4].

For our solution, we wanted to achieve the best possible performance while having a solution that is general enough to support the different architectures of networks we use in our experiments. Because we do not train models at runtime, only inference of pre-trained models needed to be implemented. While using a pre-built library such as Caffe [5] or Tensorflow [1] would cover the generality requirement, these do not offer optimal performance in all cases because they are generally optimized for using GPUs using CUDA. Actually, the CPU implementation of neural network inference in Caffe relies on the installed BLAS library, while Tensorflow uses the experimental Eigen extension for tensor math, both of which are not guaranteed to be optimized for the NAO's limited hardware.

Thus, we decided to build our own lightweight implementation for neural network inference. It is designed to read models from a file at runtime and translate them to pure x86 machine code exposed as C++-functions. That means that using a neural network model in a B-Human module is done using a simple function call. This approach offers very good locality of code and data which is a good prerequisite for caching and allows for low-level optimization of the generated code for the given purposes. Not only can it utilize all SIMD features of the NAO's CPU, but also exploit several static pieces of information about a given model during compilation to generate faster code. For example, depending on the size of kernels, inputs and outputs in a given layer, loops can be unrolled or different strategies for accumulating results can be chosen. Optimizations on a larger scale are possible as well, e. g. if a convolutional or dense layer does not have an activation function and is followed by a batch normalization along the feature axis, the normalization can be integrated into the layer by adjusting its weights accordingly.

For storing neural network models, we use a simple binary file format and created a Python script based on kerasify [12] to export Keras [3] models into that format. In the future, using other libraries than Keras for training models would be possible by adding exporters for these libraries to our format as well.

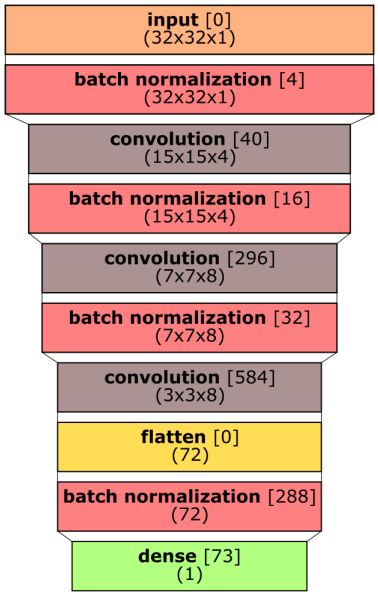


Fig. 4: The developed neural network consists of ten layers. The values in the square brackets describe the number of parameters. The values in the parentheses describe the shape of the corresponding output.

5 Ball Localization

This year, our modules for detecting and tracking the ball saw several improvements.

5.1 Ball Perception

As the SPL moves towards a realistic soccer scenario, the environment becomes more unreliable. This makes it harder to model everything properly. Therefore, even sophisticated methods such as our *CNS BallPerceptor* [10] reached their limits. However, these limitations can be overcome with the application of machine learning. Because Neural Networks, especially Convolutional Neural Networks (CNNs), have shown great success in image classification [7] and are already well established in the RoboCup, for instance by the Nao-Team HTWK [8], we decided to use them, too. However, we did not want to restructure the whole ball perception pipeline, therefore we rely on the existing regions of interest which are calculated conventionally. These so-called spots are used to calculate $32x32$ pixel patches which may contain a ball. By computing the estimated size of a ball that would be at that position, it is possible to scale the patches so that the

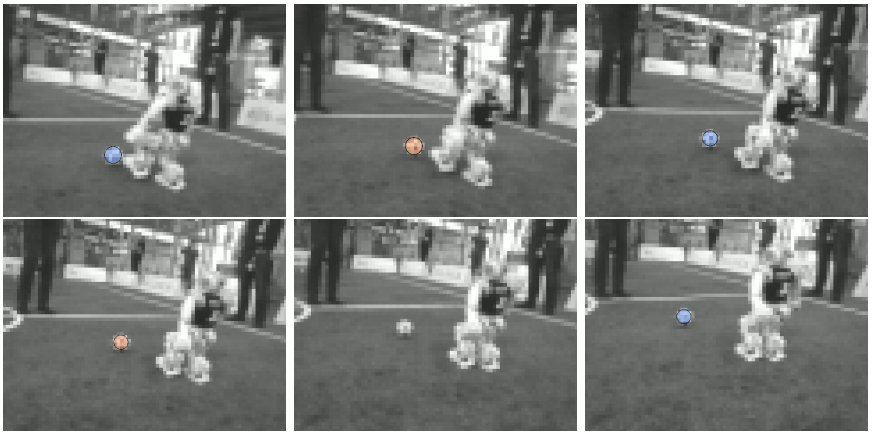


Fig. 5: A sequence of images captured while observing a kicked ball during a test game. Detected balls are denoted by circles of different colors, depending on the detector response: orange means a high probability that the detected object is actually a ball, blue means a medium probability. As one can see in this example sequence, rolling balls cannot be detected robustly. Please note that the images have been significantly compressed for logging purposes.

ball covers always approximately the same area of the patch. For each patch, a probability that it contains an actual ball is calculated by the neural network which is shown in Fig. 4. Because the result of the classification only tells us if there is a ball in a spot, but not where exactly, we use a Hough-based ball perceptor to determine the actual position of the ball in the patch after we have found a patch that is likely to contain a ball.

5.2 Ball Tracking

The currently used ball detection approach as well as the edge-based approach used in previous years, does not only provide a binary decision about something being a ball or not. Instead, it also provides detected objects that *might* be a ball but have a detector response that is too low to be sure. These detections cannot be used directly for the ball tracking process as false positives, e. g. in robot parts, occur frequently. Unfortunately, rolling balls, which are quite important to see to carry out blocking motions, cannot be detected reliably and often lead to low detector responses. One example is given in Fig. 5.

However, the motion of a ball provides enough context to integrate low-quality detections. Given a sequence of ball detections as well as a proper model for ball motion (including friction on the ground), we perform a least-squares estimation to determine if the hypothesis of a rolling ball (that has a realistic velocity) is more plausible than the hypothesis of a stationary ball. If the former is the case, the ball detections are integrated into the actual state estimation process, which is based on a set of Kalman filters. If the latter is the case, the

ball detections have to be ignored as they could be repeatedly seen false positives, e. g. in a robot leg.

6 Behavior Infrastructure

Since 2013, B-Human has used CABSL [9] to describe the robot behavior as a hierarchy of finite state machines. In addition, so-called libraries provide functionality that is called from the state machine but does not fit into the dataflow oriented framework of representations. However, the behavior hierarchy is still completely contained in one state machine. This has some disadvantages: These components cannot be exchanged easily or load their own numeric parameters from configuration files. It is also hard to create coordinated team plans for specific situations, such as set plays. This is especially important due to the new free kick rules.

In the RoboCup Small Size League, an architecture called "Skills, Tactics, and Plays" has been established for many years [2] and recently been employed in the Mid Size League [6]. To implement a similar system based on these ideas and to solve some of the problems mentioned above, another abstraction called *behavior option* has been introduced. Behavior options are declared similarly to modules (cf. "3.3.2 Module Definition" in [10]), such that they can specify their own requirements for representations and parameter sets. However, all behavior options are executed in the context of a single module which inherits the requirements of all of them. A behavior option can contain a CABSL state machine, a simple switch-case-block or a completely different representation of a behavior.

The most important addition to the old system is the PlaySelector. It is invoked one level above the normal role selection (cf. "6.2.1 Roles and Tactic" in [10]), so team-wide special situations can override the normal play. The playbook configures the set of plays among which can be chosen during the game. The selection method is based on binary applicability conditions and a numeric value specified by each play to compare them. Team coordination is achieved by letting each robot send its currently active play in its WiFi message. On each robot, the PlaySelector checks if there is another robot leading a play and, in that case, checks whether that play is possible to execute according to the own world model.

7 Framework Changes

Many datatypes in the B-Human system are streamable, i. e. they can be serialized to and from data streams, e. g. for reading them from a file. This technique was originally developed as part of the GermanTeam framework [13]. Part of the streaming architecture is the ability to determine the specification of datatypes at runtime, for instance, to be able to read them from a structured configuration file. However, in its original implementation, streaming was mainly used as a

debugging feature, e. g. to inspect data structures with an external PC at runtime. Therefore, the approach that was selected to acquire the specification of datatypes did not need to be very efficient, because it was rarely used when not debugging. As a result, the specification was determined during streaming, i. e. while the data was serialized its specification was recorded. In the current B-Human system, streaming is used much more often during the normal execution than in the GermanTeam system. In particular, the robots log a lot of data while they play soccer. It turned out that determining the specification over and over again while logging created a significant overhead. For instance last year, logging took around 1 ms per frame in the thread that performs image processing.[3]

Therefore, in the 2018 B-Human framework, the specification of all streamable datatypes is now determined only once at the beginning of the program. This would normally require a larger manual coding overhead, because a separate method is needed for each streamable datatype to record the specification. However, since most streamable datatypes in the B-Human system are generated from macros by the C++ preprocessor, this change did not require any manual changes for most datatypes. The new approach simplifies the access to the specification of datatypes, because it is now immediately available, while before, it was only available for datatypes that had been streamed at least once. In addition, streaming the data to be logged is now seven times faster, freeing up processing time for other tasks.

8 Infrastructure for the League

As in previous years, we improved the infrastructure of the league. This year, this effort was again financially supported by the RoboCup Federation as a project for league development. As the major change, we integrated the new penalty free kick and goal free kick into the GameController suite. However, at the RoboCup German Open 2018, only 23% of all pushing calls resulted in a pushing free kick.[4] This shows that the referees still have to get used to the new rule. Also, 23% of all balls played out resulted in a goal free kick.

Penalty times only end prematurely at the end of a half or a timeout. They are not counted down in the state Set. Their increase is never reset during a game and is larger now. At the RoboCup German Open 2018, all this only slightly lowered the average penalties per game from 9 in 2017[5] to 8.2 in 2018[4], although robots were longer out of play than in the previous year.

A lot of minor changes were implemented as well. For instance, the process of player substitution was simplified. The support for the coach and manual penalization was removed. The TeamCommunicationMonitor checks for the new,

[3] The actual writing to disk is performed in a separate thread in the background. Its speed is only limited by the write speed of the target medium.

[4] `http://spl.robocup.org/wp-content/uploads/downloads/RoboCupGermanOpen2018Statistics.pdf`

[5] `http://spl.robocup.org/wp-content/uploads/downloads/RoboCupGermanOpen2017Statistics.pdf`

smaller network packages and whether they are only sent three times per second instead of five times. The team communication logs from the RoboCup German Open 2018 were again made available online.[6]

9 Summary

As in most previous years, B-Human once again has many team members who have changed and improved many parts of the system. In this paper, we have described some of the major changes that are planned to come into use during the RoboCup 2018 competition. Among some infrastructure improvements, we again have a strong focus on making the vision parts of our system more robust. This is necessary as the league continuously moves forward towards more realistic (and thus more challenging) lighting conditions. We are about to switch multiple of our object detection modules to a neural network-based solution, which appears to be a good approach in the SPL context and which is already used by multiple other teams. To avoid wasting computing time, we have always limited our object detection to objects that are on the field of play. A detection of the field itself has always been based on its major color: green. Having more dynamic lighting, this process has now become more complex, as described in this paper. Overall, our changes contribute to keep the high performance of our team under even more challenging conditions. Thus, we are looking forward to a successful participation in the RoboCup 2018 in Montréal!

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`, software available from tensorflow.org
2. Browning, B., Bruce, J., Bowling, M., Veloso, M.: STP: Skills, tactics and plays for multi-robot control in adversarial environments. IEEE Journal of Control and Systems Engineering 219, 33–52 (2005)
3. Chollet, F., et al.: Keras. `https://keras.io` (2015)
4. Hofmann, M., Schwarz, I., Urbann, O., Larisch, A.: Nao Devils Dortmund team report 2017 (2018), `https://github.com/NaoDevils/CodeRelease/blob/master/TeamReport2017.pdf`
5. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)

[6] `http://spl.robocup.org/wp-content/uploads/downloads/RoboCupGermanOpen2018TeamCommunicationLogs.zip`

6. de Koning, L., Mendoza, J.P., Veloso, M., van de Molengraft, R.: Skills, tactics and plays for distributed multi-robot control in adversarial environments. In: Obst, O., Tonidandel, F., Akiyama, H., Sammut, C. (eds.) RoboCup 2017: Robot World Cup XXI. Lecture Notes in Artificial Intelligence, Springer (2018), to appear
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012), http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
8. Nao-Team HTWK: Team research report 2017 (2018), http://www.htwk-robots.de/documents/TRR_2017.pdf
9. Röfer, T.: CABSL - C-based agent behavior specification language. In: Obst, O., Tonidandel, F., Akiyama, H., Sammut, C. (eds.) RoboCup 2017: Robot World Cup XXI. Lecture Notes in Artificial Intelligence, Springer (2018), to appear
10. Röfer, T., Laue, T., Bülter, Y., Krause, D., Kuball, J., Mühlenbrock, A., Poppinga, B., Prinzler, M., Post, L., Roehrig, E., Schröder, R., Thielke, F.: B-Human team report and code release 2017 (2017), only available online: http://www.b-human.de/downloads/publications/2017/coderelease2017.pdf
11. Röfer, T., Laue, T., Hasselbring, A., Richter-Klug, J., Röhrig, E.: B-Human 2017 – team tactics and robot skills in the standard platform league. In: Obst, O., Tonidandel, F., Akiyama, H., Sammut, C. (eds.) RoboCup 2017: Robot World Cup XXI. Lecture Notes in Artificial Intelligence, Springer (2018), to appear
12. Rose, R.W.: kerasify. https://github.com/moof2k/kerasify (2016)
13. Röfer, T., Laue, T., Weber, M., Burkhard, H.D., Jüngel, M., Göhring, D., Hoffmann, J., Altmeyer, B., Krause, T., Spranger, M., von Stryk, O., Brunn, R., Dassler, M., Kunz, M., Oberlies, T., Risler, M., Schwiegelshohn, U., Hebbel, M., Nisticó, W., Czarnetzki, S., Kerkhof, T., Meyer, M., Rohde, C., Schmitz, B., Wachter, M., Wegner, T., Zarges, C.: GermanTeam RoboCup 2005 (2005), http://www.informatik.uni-bremen.de/kogrob/papers/GT2005.pdf
14. Zivkovic, Z.: Improved adaptive gaussian mixture model for background subtraction. In: Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. vol. 2, pp. 28–31 Vol.2 (Aug 2004)