RoboEireann Team Description
RoboCup 2017 Standard Platform League

Rudi Villing[1], John McDonald[2], Simon O'Keeffe[1], Louis Gallagher[2]
[1]Department of Electronic Engineering [2]Department of Computer Science
Maynooth University
Maynooth, Co. Kildare, Ireland
http://www.roboeireann.ie

1 Introduction

1.1 Team

RoboEireann is a Robocup team from Maynooth University, and is currently Ireland's only RoboCup Standard Platform League team. RoboEireann is a collaborative effort between the staff and students of the Computer Science and Electronic Engineering Departments of Maynooth University, both of which have strong research records in the wider areas of computer vision, signal processing, control, robotics, and intelligent systems. In previous years we have also hosted visiting researchers from other teams and collaborated with Queen's University Belfast.

We first participated in RoboCup in 2008 as part of the Standard Platform League team *NUManoids* which was a joint effort of Maynooth University and the University of Newcastle, Australia. In that year, the first in which the Aldebaran NAO platform was used, we were Standard Platform League's overall winners. Since 2009 we have competed as an individual team under the name *RoboEireann*.

As RoboEireann we have competed in RoboCup 2009, 2010, 2011, 2013, 2015 and 2016. In RoboCup 2013, Eindhoven, we reached the quarter finals of the competition for the first time. In RoboCup 2011, Istanbul, we achieved first place in the technical challenge for our open challenge "Localisation without goal posts". The approach which was based on an extension to the work of Cox [1] was published in [12] and [13]. We have also competed in the RoboCup German Open every year from 2010 to 2014, achieving 2^{nd} place overall in 2012 and 2^{nd} in the best player award for the drop-in player competition in 2014.

For RoboCup 2017, the team is comprised of the following staff, graduate students, and undergraduate students:

Staff members: Rudi Villing (Team Leader) and John McDonald.
Graduate students: Simon O'Keeffe, Louis Gallagher, and Chervin Ann Dela Cruz.
Undergraduate students: Enrico Marugliano, Pang Yan, Daniel Hopkins, Dylan Stuart, Toby Burns, Robert McCraith, and Martin Akinola.

1.2 Impact

RoboEireann has made a number of technical contributions to the standard platform league since 2008. The distance and accuracy of our strong kick design was a notable contribution in early years. Our localization system developments based on Kalman filters and subsequently based on extensions to the line based registration algorithm due to Cox also contributed to progress within the league. Other technical contributions by the team include: early development of the b-script system for specifying behaviours, development of a light weight modular architecture, and kernel fixes to the Aldebaran Linux kernel to deal with a number of camera driver issues[1]. Over the years we have also hosted members from other RoboCup teams for extended research visits at our lab.

RoboCup and the standard platform league provide an excellent training and development environment for students in the areas of robotics and real time software systems. In Maynooth University we have a number of research groups active in the area of robotics and RoboEireann provides an excellent means for engaging the CS and EE undergraduate community with those groups. Every year, academic staff associated with RoboEireann have supervised undergraduate projects and internships that expose students to both the practical and cutting edge aspects of of robotic software development. The association with RoboCup is a very positive motivating factor that greatly affects the students' desire to get involved and perform well. Since our initial involvement in RoboCup we have had a number of undergraduate students who, as a consequence of their involvement in the team, have completed Masters and PhDs in robotics in our respective labs.

In September 2016, Maynooth University took its first intake of students on a new B.Sc. programme in Robotics and Intelligent Devices. The programme's syllabus draws on the wide array of research and postgraduate level activities in both departments including our participation within RoboCup. The programme will incorporate a focus on modern mobile autonomous robotics, including hands on experiences of modern robotic software and hardware platforms (including the Nao) and laboratory sessions with the RoboEireann codebase in the later years.

In addition to engaging students within the university, our involvement in RoboCup has played a key role in the successful running of a three year Summer Internship in Autonomous Robotics (SIAR) programme [2] and outreach activities to promote engagement of the Irish public with science, technology, engineering, and maths as part of the Discover Programme. Both programmes received financial support from Science Foundation Ireland (SFI). The SIAR programme funded 30 summer internships over its duration for both national and international students at Maynooth University. The Discover Programme has funded outreach activities for robot soccer demonstrations that have now been seen by thousands of children, families, and the wider public as part of the annual National Science Week events.

[1] https://github.com/mp3guy/linux-aldebaran/commits/release-1.12/geode
[2] http://www.roboeireann.ie/siar/siar.php

2 Architecture and key subsystems

2.1 Code Usage

The vast majority of code in the RoboEireann code base has been developed by RoboEireann team members with the exception of the components listed in this section.

Between 2015 and the present we have made use of following B-Human components: the B-Human 2010 walking engine [8], the libbhuman process and shared memory communication code, an Unscented Kalman Filter (UKF) implementation, and some utility classes related to math and poses.

Additional code that was not directly ported from other teams but inspired recent additions to our code base includes a RANSAC line fit from Nao Team HTWK, some aspects of field boundary detection from B-Human, and a ball circle fitting technique from code released by UChile.

Finally, for 2017 we are integrating the Walk2014Generator from the 2016 code release of UNSW Sydney.

2.2 Architecture

The RoboEireann software architecture is based around two main threads: Body and Cognition as shown in Fig. 1. The Cognition thread runs at the vision frame rate (i.e. 2 cameras×30Hz) and implements all the "intelligence" of the robot. The Body thread runs at the Naoqi DCM rate (i.e. 100Hz) and implements the low-level motion control. The Body thread interfaces with the robot hardware through a separate *libagent* process which in turn manages all low level direct communication with Naoqi and isolates Naoqi from crashes in the robot control code. Cognition does not interact directly with Naoqi or libagent. The libagent implementation is a ported and slightly modified version of the equivalent libbhuman component from B-Human.

The design of the system is based on an original blackboard style micro module framework that has been implemented completely in-house. Each independent piece of information that is shared between several modules is represented in a single *MemoryUnit*. Separate modules of functionality are implemented as independent classes. Modules can read data from and write data to the memory units in the shared blackboard but they must declare their dependencies in advance. Each module can write to one or more memory units, but only one module (the owner) is permitted to write to a given memory unit. A module may also declare a read dependency on one or more memory units. The framework is responsible for ensuring that memory units to be read have previously been written so that they contain valid data.

Each thread in the architecture implements the superloop architecture for real time systems such that all activities for a given iteration of the loop must be completed before the loop period expires. In the case of the cognition agent, this loop period is approximately 16 ms. Within the superloop, the module framework calls modules sequentially to perform one frame worth of processing.

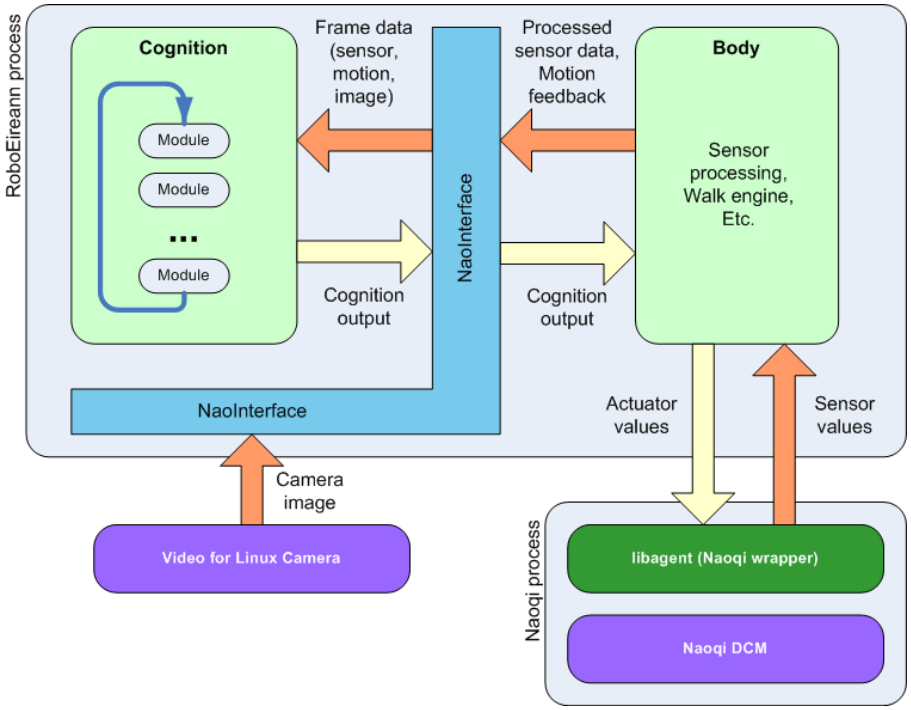


Fig. 1: RoboEireann Architecture

These calls are scheduled in a fixed round robin order determined during framework initialization to ensure that dependencies between modules which read and write individual memory units are satisfied.

2.3 Robot Perception

Currently, perception in the RoboCup environment presents many vision challenges including black and white SPL ball detection, robust white goal detection, and operation under natural lighting conditions.

Ball detection: Our existing heuristic based ball detector relies on colour, shape, and real world dimensions to classify ball candidates. Candidate ball regions are identified by scanning the image for pixels that are not the pitch background (green) and constraining regions by real world size. Once identified, candidate regions are separated into those that are in free space and those that may be on a line or near a robot or goal post. Free space candidates are then subjected to tests for circle edge points, pitch coloured surroundings, and discrimination from the penalty spot. If these tests are passed, the ball has been identified. If no free-space candidates result in a ball detection, then all candi-

dates are subjected to additional heuristics which depend on the black spots in the ball.

This approach detects a ball in free space up to 3 m away and detects a ball on a line or near a robot up to 1.5 m away. However, it fails to consistently detect balls on a line or near a robot more than 1.5 m away. The detector also struggles to detect a moving ball if it is not in free space and cannot detect occluded balls. For this reason we have been developing an alternative approach using Deep Learning and CNNs which is described in 3.1.

Pitch Boundary: The RoboEireann pitch boundary detector was inspired by the pitch boundary detectors of both B-Human and Nao-Team HTWK, as presented in their respective 2015 code releases [9,6]. To extract candidate pitch boundary points we follow the general approach of B-Human, scanning the image vertically and maintaining a score along each scanline. If a pixel is classified as *pitch* the scanline is rewarded, if not the scanline is penalised. Our approach differs from B-Human in that we reward/penalise in a non-symmetric manner, penalising more heavily than we reward, to reduce the effects of noise on pitch boundary detection. A scanline is considered to intersect with the pitch boundary where its score is maximal. To recover the pitch boundary from these candidate points we use RANSAC to fit a line to the candidate boundary points, similar to the HTWK approach.

Obstacle Detection: In the past, obstacle detection was based on ultasonic sensor input only and suffered from a number of problems. We have now integrated a basic visual obstacle detection module. The approach is deliberately simple and shares common elements with pitch boundary detection both for performance and implementation convenience. Specifically, the obstacle detector identifies obstacles as vertical runs of non-pitch pixels that are too big to be a ball. (The pitch boundary is also classified as an obstacle in this scheme.) An occupancy grid of obstacles seen over a short time window is maintained so that the robot can make better path planning and kicking decisions.

Goal Detection: Our goal detector uses the shape, colour, and dimension of the the posts as a basis for detecting individual posts. The image is scanned horizontally, starting at the horizon line, looking for non-white to white transitions. These points are then grouped vertically to form post candidates. The post candidates are subjected to a series of heuristic tests that check the size of the bottom of the post, the colour around the base of the post, and that the candidate is not clearly a robot arm (but see 3.2 also).

Circle detection: Previously our localisation system relied heavily on yellow goals as a global cue. The introduction of white goals resulted in a significant increase in false positive detections and a resultant detrimental effect on localisation. To mitigate this issue we have reduced the influence of goal percepts

on localisation and introduced a new oriented center circle percept. The associated detector is based on a RANSAC fit of line points that have been projected into pitch space. Tight bounds on permitted circle parameters provide increased reliability in detection and lower the rate of false positive detections.

2.4 Motion

Walking: We originally used the Aldebaran walk, but found that it was rather slow in a RoboCup context and had some difficulties stopping near and approaching the ball. Since RoboCup 2011 we have instead been using a modified version of the 2010 B-Human walk [8]. The B-Human walk engine was modified where necessary to utilise features already available in our code in preference to adding unnecessary duplicated code from the B-Human code base.

For RoboCup 2017 we have included the Walk2014Generator from the UNSW 2016 code release [11]. Sensor and joint data were mapped from the RoboEireann representation to the UNSW representation to use the walking engine and a reverse actuator mapping was done to actuate the Nao motors.

Kicking: Our primary kick engine was designed around carefully designed poses for backswing, mid strike, foot lift and recovery. In all cases, key aspects of torso, arm and leg movements were considered in the design so that maximum kicking force and accuracy could be achieved. Using a small number of main kick designs we could independently specify the kick direction within a range of ball placements by interpolating between different kick parameter sets. This kick performed well at numerous RoboCup and Open competitions where tests showed that we could kick more than the length of the old 6 $\times$ 4m field. Our kick design is still relatively unique, but other teams have subsequently achieved similar performance with alternative designs. Recently we have adapted this kick so that it can also be used in passing moves.

As the RoboCup soccer game has improved in recent years, the importance of dealing with the ball quickly has increased. Since 2013 we have been using fast and short in-walk kicks that were part of the B-Human walk engine. Our behaviour system now applies heuristics and a probabilistic element to decide which kick to perform when the ball is within range. These in-walk kicks are now being ported to the UNSW motion generator.

2.5 Behaviour and Team Play

In 2011 our behaviour was based on hierarchical state machines. Since 2012 we replaced this with an early variant of the b-script behaviour engine initially developed in Ireland and subsequently refined in Germany [4]. This system uses a custom scripting language to specify behaviours using hierarchical generators.

Our current behaviour design is based on robot agents which can switch between a number of roles. In five-player SPL teams the roles have typically been striker, supporter, forward, defender, and goalie. Each agent determines

their own role based on information they perceive themselves and information broadcast from their team members in the standard SPL packet. We gradually modified this system in an effort to support greater flexibility and less hard switching between roles. This was particularly applicable to the now-defunct drop-in player competition (where behaviour should be more adaptive to team mates and the overall situation). We see it having relevance to the new mixed teams competition also.

Recently, qe have developed an alternative to the b-script behaviour system which is based on co-routine behaviours implemented in C++ rather than the b-script scripting language. When using b-script, high level behaviours are made simple and made to perform quickly enough by implementing complex logic or calculations in C++ functions that are called from b-script. By using a similar approach with C++ based co-routine behaviours (i.e. separating complex logic and calculations from the high level behaviour logic), implementing the high level behaviours in C++ becomes rather similar to implementation in b-script. The advantages of the C++ implementation are that it runs faster, many more errors are detected at compile time, and there is no requirement to maintain a mapping between C++ identifiers and b-script identifiers.

Having had experience of hierarchical state machines, hierarchical generators, and hierarchical co-routines we have observed that each has different advantages which it would be useful to combine. Our development is currently leading us towards a hybrid implementation.

2.6 Behaviour simulation

For 2017 we have developed a very fast multi-agent behaviour simulator that uses simplified physics and perception. It can simluate a 10 minute half of an SPL 5 vs 5 game in 11.6 s when the simulation frame rate is 60 frames per second. If the simulation frame rate is dropped to 4 frames per second (which is generally reasonable for behaviour simulation but less accurate from a physical perspective) the 10 minute game half can be simulated in just 880 ms on average. These measurements were collected with agents running the most recent RoboEireann b-script game play behaviour.

The main benefit of such a fast simulator is that it allows the collection of behaviour statistics averaged over many matches when evaluating different behaviours or different parameterizations of a behaviour. We are aware of some aspects of the implementation that are currently over simplified so it is likely that the final simulation speed will reduce somewhat. Nevertheless we do not expect the differences to be very significant.

The simulator can be run headless, or with a basic visualization monitor as shown in Figure 2. It has been designed so that it does not depend on any one agent or behaviour implementation. It allows the behaviour engine, the mapping from simulated ground truth to percepts, and the mapping from agent requested actuations to simulated realisation to be customized. In particular, customization of the mapping to and from simulated ground truth enables some mitigation of the so-called "reality gap" that exists between simulation and robots operating

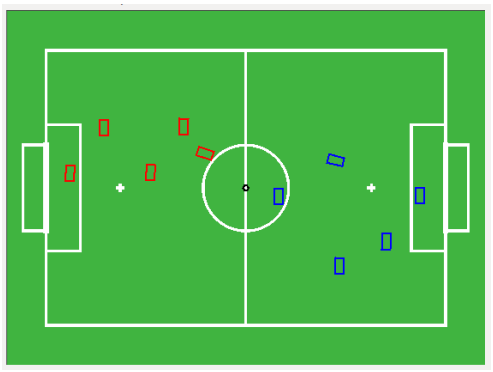


Fig. 2: Behaviour simulator monitor

in a real dynamic environment. We expect that this simulator will be usable by teams other than RoboEireann in the future.

2.7 Debug infrastructure

For 2017 our debugging and logging infrastructure has undergone significant change. The new infrastructure uses Google Protocol Buffers [3] (protobufs) for exchanging data in a language and platform independent manner. Our logging file format also uses protobufs now. In keeping with Google recommendations we use several small messages rather than a single large message to communicate the data associated with each cognition frame from the robot to a debug client. Benchmarking on the robot indicated that large memory allocations and large memory copies were both rather time consuming, so, to eliminate unnecessary copies and memory allocations, the camera image is encoded as a length delimited byte stream with a protobuf compatible length rather than a full protobuf message.

Our main debugger tool has been enhanced so that it can now either connect online to a robot or run offline by replaying logfiles through robot code running in the debug environment to facilitate faster test, debug, and fix development cycles.

3 Research and development

3.1 Ball detection using convolutional neural networks

A major challenge which we faced during 2016 was the development of a robust detector for the new black and white ball. Although we have had some success with traditional approaches such as heuristic based colour filtering, we found that detection rates were poor in the face of clutter and over large ranges. This year we have put considerable research focus on exploiting recent success in

machine learning to address this problem. In particular Convolutional Neural Networks (CNNs) have been shown to be incredibly powerful at image based tasks from object classification to activity recognition. In our work we have evaluated a number of different deep network architectures for both detection and segmentation of the new SPL ball.

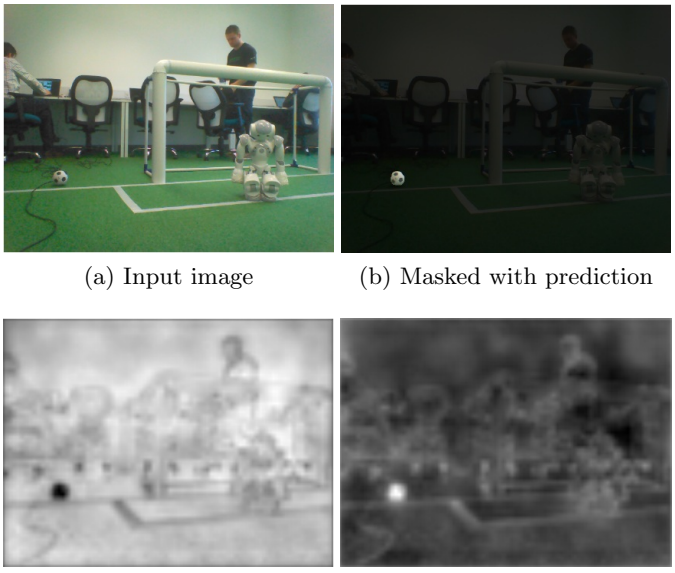
Lightweight CNN based detection: We used the Caffe Deep Learning Framework [5] to evaluate multiple CNN architectures on small image patches (from 12×12 to 32×32 pixels) representing candidate ball regions. We examined architectures which varied the number of layers, number of outputs per layer, kernel size, the use of pooling, and the use of ReLUs. The largest network evaluated had 5 weighted layers. Our results showed that classification performance was largely insensitive to many parameters of the architecture (including notably the number of layers and kernel size). Inference time on the robot did vary however, taking between 1.5 and 4 ms to classify one image patch, depending on network size and architecture.

Such inference times would limit the number of candidate patches that could be processed to 1 or 2 per cognition cycle. For this reason, we also evaluated the XNOR-Net architecture [7] which quantizes the weights and activations of a CNN to binary values. We found that the classification performance of XNOR-Net was around 12% worse than an equivalent network using full precision weights and activations.

Pixel-wise segmentation using FCNNs: Separately we have also investigated the applicability of Fully Convolutional Neural Networks (FCNNs) which give dense pixel-wise prediction of object location [10]. FCNNs can take arbitrary sized images and produce a binary mask depicting the class of every pixel, allowing multiple classes of object to be located in the image. When applied to ball segmentation the results provide a bounding box (as is the case in object detection) and a per-pixel segmentation which allows further morphological level processing to be applied. When evaluated on a dataset of approximately 1000 images, this approach achieved 68% accuracy measured by per pixel correctness. Figure 3 shows the output and intermediate represenation for an example input image. Although this work shows the effectiveness of such architectures, deploying or exploiting the results on the limited compute platform of the Nao is still an open problem within our research.

3.2 Robot detection

With many robots on the field, robot arms are a significant source of false positive goal detections. These can be eliminated by detecting regions containing robots prior to goal detection. This year we have developed a method for visual robot detection based on the histogram of oriented gradients (HoG) algorithm in conjunction with a support vector machine (SVM) based classifier [2]. This algorithm extracts a feature vector using the gradient magnitude orientation in



(a) Input image (b) Masked with prediction

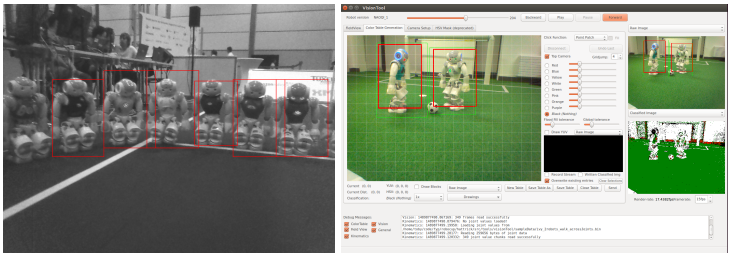
(c) Raw background prediction (d) Raw ball prediction

Fig. 3: Examples of pixelwise segmentation of the ball using a Fully Convolutional Neural Network (FCNN).

local image regions. These features, along with a relevant classification label, are then input to a support vector machine which is trained to perform robust robot detection. To avoid an exhaustive search on the full camera image we used a heuristic for rapidly finding smaller candidate regions in the image. Our detector was trained and tested using 3-fold cross-validation on 200 images with a precision of 95% and a recall of 80%. Figure 4 shows an example output from the detector. Although we have this detector deployed within the robot codebase, we are currently focussed on improving performance to permit it to be used within competitive game scenarios.

4 Conclusions

We have presented the RoboEireann Robocup Standard Platform League team, the software architecture and sub-systems, and the ongoing research and development work both for Robocup 2017 and beyond. This year we have developed a number of new percept detectors in order to maintain robust gameplay in the context of recent rule changes. We have also developed a new fast behaviour simulator and enhanced our debugging infrastructure and debugging tools. A key feature of our research this year has been the examination of machine learning applied to perception in the RoboCup environment.



(a) (b)

Fig. 4: HoG based robot detections

References

1. Cox, I.J.: Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. Robotics and Automation, IEEE Transactions on 7(2), 193–204 (Apr 1991), http://dx.doi.org/10.1109/70.75902
2. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1, pp. 886–893 vol. 1 (June 2005)
3. Google: Protocol buffers. https://developers.google.com/protocol-buffers/
4. de Haas, T.J., Laue, T., Röfer, T.: A scripting-based approach to robot behavior engineering using hierarchical generators. In: ICRA. pp. 4736–4741 (2012)
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. Nao-Team HTWK: HTWKVision code release 2015. https://github.com/NaoHTWK/HTWKVision (2015)
7. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, pp. 525–542. Springer International Publishing, Cham (2016), http://dx.doi.org/10.1007/978-3-319-46493-0_32
8. Röfer, T., Laue, T., Müller, J., Burchardt, A., Damrose, E., Fabisch, A., Feldpausch, F., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Honsel, D., Kastner, P., Kastner, T., Markowsky, B., Mester, M., Peter, J., Riemann, O.J.L., Ring, M., Sauerland, W., Schreck, A., Sieverdingbeck, I., Wenk, F., Worch, J.H.: B-Human team report and code release 2010. http://www.b-human.de/downloads/bhuman10_coderelease.pdf (2010)
9. Röfer, T., Laue, T., Richter-Klug, J., Schünemann, M., Stiensmeier, J., Stopmann, A., Stöwing, A., Thielke, F.: B-Human team report and code release 2015. https://www.b-human.de/downloads/publications/2015/CodeRelease2015.pdf (2015)
10. Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. CoRR abs/1605.06211 (2016), http://arxiv.org/abs/1605.06211
11. UNSW Australia Team Members: UNSW RoboCup standard platform league code release 2016. https://github.com/UNSWComputing/rUNSWift-2016-release (2016)

12. Whelan, T., Stüdli, S., McDonald, J., Middleton, R.: Line point registration: A technique for enhancing robot localization in a soccer environment. RoboCup 2011: Robot Soccer World Cup XV pp. 258–269 (2012)
13. Whelan, T., Stüdli, S., McDonald, J., Middleton, R.: Efficient localization for robot soccer using pattern matching. In: Leveraging Applications of Formal Methods, Verification, and Validation. Communications in Computer and Information Science (2012)