Rhoban Football Club: RoboCup Humanoid Kid-Size 2017 Champion Team Paper

Julien Allali, Rémi Fabre, Loic Gondry, Ludovic Hofer, Olivier Ly, Steve N'Guyen, Grégoire Passault, Antoine Pirrone, Quentin Rouxel

Rhoban Football Club Team,
LaBRI, University of Bordeaux, France
quentin.rouxel@labri.fr

Abstract. In 2017, Rhoban Football Club reached the first place of the Kid-Size soccer competition for the second time, and was awarded best humanoid by the jury[1]. Capitalizing over the hardware design and software basis from previous participations, we improved both robustness and overall behaviors of our humanoid robots. An interesting milestone reached is the opportunistic pass ability ; since such an high-level team play behavior can only work after solving underlying difficulties like having a good field localization, robust and reliable detection of vision features, a repeatable locomotion with odometry etc. This paper describes the most meaningful improvements accomplished over this year along with perspectives of future work.

1 Introduction

During the 2017 competition, Rhoban humanoid robots scored 41 goals. 2 goals were scored against Rhoban, in both cases by our own robots because of a technical bug with our localization system that was fixed during the competition. Indeed, the banishment of magnetic compass[2] in 2017 makes it more likely to observe this kind of failure. We also scored the highest number of points during the drop-in games, which was a new challenge mixing different teams in the 2017 edition. This was a great opportunity for teams to get ready for real game situations. In this paper, we present the main modifications we added to both software and hardware since RoboCup 2016.

2 Hardware

Like every team, Rhoban continuously tries to improve its robots by observing new problems each year at the RoboCup and trying to fix it for the next year. In 2017 our new *Sigmaban 2* robots are 68 cm tall with Dynamixel MX106 on the legs, which has proven an efficient and robust compromise between our now

[1] https://www.robocuphumanoid.org/hl-2017/results/
[2] https://www.robocuphumanoid.org/materials/rules/

old MX64 based 58 cm tall Sigmaban and the various taller prototypes (90 cm in 2015, 80 cm in 2016). The mechanical parts of the robots are now entierely machined and screwed aluminium rather than bent for a better precision. Some other significant improvement have been made this year and are described below.

2.1 Feet Pressure and Load Cells

Sensing pressures under the feet is one of the specificities of our small humanoid robots, as explained in the 2016 champion paper [1] and open-source project[3].

In 2017, we switched to a more robust, full Wheatstone bridge, 40kg rated, off the shelf load cells. These load cells are smaller than the previous ones and are able to handle higher forces because they are made of 3mm thick steel. To improve the solidity of our robots, and to avoid exposing the load cells and especially the strain gauges below the feet, we also switched to another hardware fixation, as depicted on Fig. 1. The load cells are now on the top of the feet plate and the deformation is transmitted through the plate, using a nylon ring to reduce frictions.

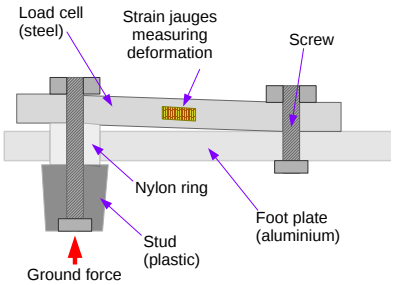


Fig. 1: Fixation of load cells on the feet plate for RoboCup 2017. An exaggerated deformation is represented to understand how the measurement is done.

2.2 Hot-Swap

We designed an hot-swap power board that offers the possibility to switch the batteries without any transition phase where the robot computer is not powered. To achieve this, the new battery is plugged before the old one is unplugged and a *single pole, double throw* switch is used to change which battery powers the motors. Two diodes allow the batteries to be simultaneous power sources of the computer, regardless of the switch position. This system is more efficient than the hard reset that was needed during the half-time break in previous years.

[3] `https://github.com/Rhoban/ForceFoot`

2.3 Open-Sourcing Dynamixel Board

Our Dynamixel board, which includes three parallel USART buses to reduce the low-level delays that was already introduced in 2016 [1] is now available as an open-source project[4].

2.4 Head Protection

In previous years, 3D-printed camera protections were used. This year, the protections were cut into the same aluminum used for the robot's body. This turned out to be a strong downgrade. When a collision occurs with the head of the robot, 3D-printed protections bend, effectively absorbing a portion of the impact. With the aluminum protections, most of the impact is directly transmitted to the motors' shafts. Rotational offsets up to 6 degrees were commonly observed on the head pitch servomotors after a fall and having the head's shaft completely severed became a common issue.

3 Motion

3.1 Walk Engine

From 2014 to 2016, all of our robots were relying on the open-source *IKWalk* engine presented in [15]. For this 2017 edition of the RoboCup competition, this walk engine has been improved to a new version, *QuinticWalk*[5], leading to a smoother and stabler motion. This new walk engine follows the same principles as the old one but with many details updated.

As introduced by [2], the movement is open loop and does not use any ZMP criterion or dynamics modeling. The shapes of target trajectories are built geometrically from a set of parameters in the Cartesian space. All target joint positions are then computed through an inverse kinematics of the legs of the robot. These parameters are manually tuned on the physical humanoid robot (trials and errors) until a fast and balanced motion is achieved. The main changes are the following:

- The trajectories are represented by using 5th order polynomial splines. This splines allow for continuous accelerations. Hence, the computed theoretical torques generated by the motion at each joint are also continuous, as well as the ZMP trajectory on the ground.
- Acceleration continuity is ensured during omnidirectional walk as well as parameter update (in a way similar as done in [9]).

[4] `https://github.com/Rhoban/DXLBoard`

[5] Source code available at: `https://github.com/RhobanProject/Model/tree/master/QuinticWalk`

- Instead of expressing the target trajectories with respect to the moving frame of the *trunk*, all Cartesian positions and orientations (12 degrees of freedom) are expressed with respect to the support foot, fixed on the ground. The overall motion is easier to define but the discontinuity of the representation has to be handle at the support swap.
- The handling of the double support phase is implemented. A low frequency and stable gait can be achieved with the right set of parameter values.
- On the walk startup, the trunk begin to oscillate during an half cycle before the legs in order to initialize the dynamics of weight swigging of the robot.

As a result, the displacement of the robot is qualitatively far stabler, especially on our bigger robots. For example, the lateral steps do not trigger constantly the stabilization module anymore (see [12]) which was required to prevent the robot from falling.

However, the effects of the Runge phenomenon can be observed on some parts of the trajectories. The current implementation could be improved by replacing the polynomial splines by the optimal bang-bang jerk control developed by [4].

4 Vision

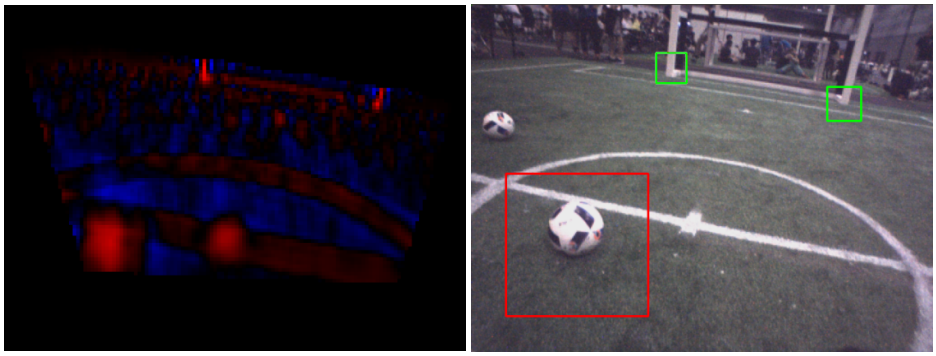
Our main vision system has been completely changed in 2017. The method we used before was mainly based on multiple hand-tailored OpenCV filtering, but for our purpose, this "standard" method has reached its limits in terms of complexity, robustness and maintainability. The new vision pipeline is much simpler and requires much less hand tuning. Moreover, it also appeared to be quite robust to the environments changes in luminosity and color.

Regions of interest (ROI) for the ball and the goal posts are extracted from the full image, using the robot's state (ground plane projection on the camera plane) and a kernel convolution on an Integral Image filter. These ROIs are then classified by a Convolutional Neural Network, see Fig. 2.

4.1 Identification of ROI by Integral Images

The first step of ball and goal post detection is to identify ROI which are likely to contain them. The ROI are squares and their size is computed according to the robot model. A ROI is supposed to measure twice the size of the ball, due to perspectives, the size might vary inside the image.

The score of a ROI is computed based on white and green densities inside the region. Local densities can be obtained at a low computational cost based on integral images, see [5]. For the ball, the density of white at the center of the region should be higher than the average density in the whole region, on the other hand, the density of green in the center of region should be lower than in the whole region. The same pattern applies for goal post detection, but the region which has to contain a high density of white is the top center of the region. A



(a) Score of ROI for goal posts (b) Result of goal posts detection

Fig. 2: Example of ball and goal detection

visualization of the score function is shown for the goal posts in Fig. 2a. Positive scores are shown in red and negative scores in black. Note that if a region of interest should exceed the margin of the image, then its score is zero.

In order to ensure a satisfying frame rate, a maximum of 4 ROI per frame are classified by each neural network. Selection is based on scores and intersection of ROI is forbidden.

4.2 Classification

Convolutional Neural Networks (ConvNet) have become the state of the art methods in various computer vision tasks [10]. Several off-the-shelf very powerful architectures are available such as [13,14] but unfortunately none were usable in the very limited embedded computers of our robots. We thus designed our custom ConvNet using a c++ library with no external dependencies [6]. The aim of the approach was to design a minimal architecture able to classify ball and goal post patches with at least 95% accuracy.

After some hand-tuning we obtained a quite small network (cf. Fig. 3) able to classify small $16 \times 16 \times 3$ patches with good results (cf. Table 1).

	nb training/validation	**Learning rate**	**Accuracy**
Ball	7400/1500	0.013	96.8%
Posts	13500/2500	0.0425	96.92%

Table 1

This architecture was used both for ball and goal post classification, with the small difference that the goal network has only 16 feature maps.

[6] `https://github.com/tiny-dnn/tiny-dnn`

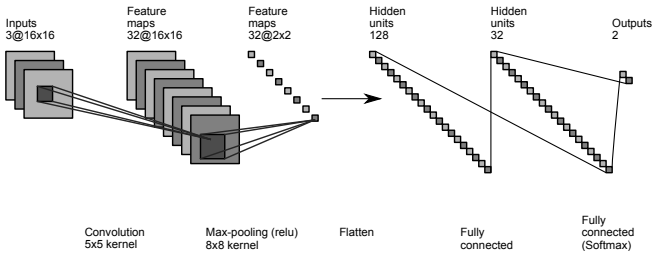


Fig. 3: Architecture of the reduced ConvNet used for ball classification in the RoboCup 2017

4.3 Logging and Image Tagging

One key functionality to this system is the ability to quickly obtain a large quantity of labeled data. To do so, patches extracted from ROIs were uploaded on an online tagging tool[7] accessible to the public. Tagging was made simple with a responsive "*Google ReCaptcha*" style interface. This tool was used by several supporters from outside the team and allowed to get thousands of labeled patches in a matter of a few hours, thus freeing precious time to the team members. In order to ensure the quality of the tagging from non expert users, a consensus based approach on the label was implemented (a patch is considered correctly labeled if it was tagged more than N times and if more than $X\%$ of the tag were consistent). Moreover, new users were first required to correctly label a set of images previously labeled by the team members. To date, approximately 10000 patches of of balls and 4000 of goal posts were positively labeled. Finally, a slight gamification of the system was added through a leaderboard in order to motivate users. The project is open source[8] and the data are available once registered.

5 Strategy

Last year, our strategy to kick the ball was simply to face the center of the goals. However, this is not a satisfying solution for several reasons. For instance, when the ball is located on the corners of the field, it is better to center it before trying to score a goal. Moreover, depending on the ball location, the tolerance on the orientation is not the same, it is more important to face exactly the goals when we are near than if we are in the other half field.

5.1 Kick Model

In order to be able to optimize the kick decisions, we started by establishing a model of the noise on the kick based on real-world data. We used a simple model

[7] `http://rhoban.com/tagger/`
[8] `https://www.github.com/rhoban/tagger`

where the kick distance r and the kick direction θ are sampled from Gaussian distributions.

$$\begin{aligned} r &\sim N(\mu_r, \sigma_r) \\ \theta &\sim N(\mu_\theta, \sigma_\theta) \end{aligned} \tag{1}$$

5.2 Value Iteration for the Kick Strategy

Choosing the decision and the type of kick is modeled as a Markov decision problem, MDP for short.

The current state $s \in S$ can be any position of the ball on the field using discrete grid (for instance every 20 cm), or also *out* or *goal* if the ball goes out of the field or scores a goal. The possible actions $a \in A$ are tuples (k, θ) of the selected kick k (forward kick or lateral kick) and kick orientations θ, which are also discrete (for instance every 5 degrees).

We can then build a model of the stochastic transition function, using random samples, as depicted on Fig. 4. This model is actually the estimation of $P(s'|s, a)$, i.e the probability of reaching the state s' knowing that we were in the state s and we applied the action a.

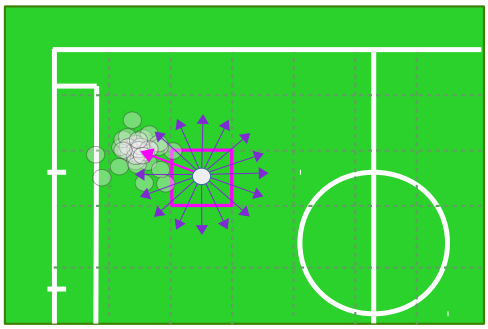


Fig. 4: The transition function is approximated using random sampling for each possible state s and action a.

We then use the value iteration algorithm [3] with the following iterative method:

$$\begin{aligned} &V(s) = \max_a \left[\sum_{s' \in s} P(s'|s, a)(R(s, s') + V(s'))\right] \\ &V(goal) = 0 \\ &V(out) = -1000 \end{aligned} \tag{2}$$

Where $R(s, s')$ is the reward for getting in state s' from state s. In our case, we choose the reward to be the negative number of seconds required to walk from the position of state s to the state s'. Thus, the value of $V(s)$ can here be understood as the (negative) time required to score a goal for the better possible sequence of kicks.

We update the values of each state iteratively until one complete iteration did not changed the value of any state.

The action a that maximizes $V(s)$ for each $s \in S$ is then the kick and orientation that should be used depending on where the ball is located on the field. Moreover, we can also select several actions that produce similar values, using a tolerance threshold. This is an interesting information since it can be used to know how accurate we need to be when trying to approach the ball. Fig. 5 shows an example of result obtained.

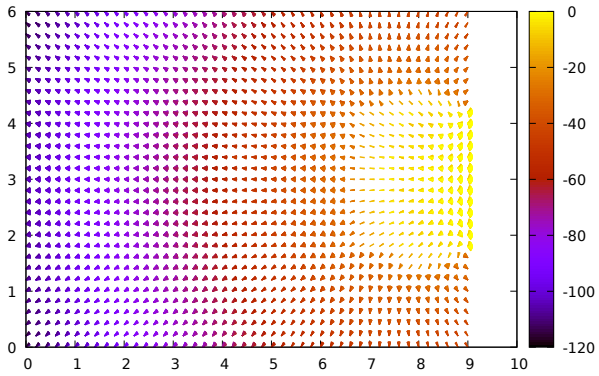


Fig. 5: The values of $V(s)$ (using color gradient) on the field for the better kicks (using a tolerance) for each position on the field

Approaching the kick problem through MDP allowed us to provide satisfying policies without requiring heavy hand-tuning of parameters. However, in order to take into account the position of the other robots into the kick decision process, it would be necessary to add dimensions to the problem, thus leading to the curse of dimensionality. We expect to use approach based on continuous state and action MDP to improve further the quality of our kick decision policies. This approach has already been successful for reducing the time require to reach a satisfying position for kicking [8].

5.3 Team Play

All robots share some information about their state, including their position estimation, the estimation of where the ball is on the field, their game state (for instance if they are playing, fallen or penalized).

A kicker score, which is an estimation of the time that would be required for the robot to reach the ball and the value $V(s)$ depending on the ball position is also shared to choose which robot will handle the ball and kick it. This robot shares the estimation of the position of the ball after its next shoot, using the mean μ of the kick model. We achieved opportunistic passes this way. Indeed, the robots were trying to walk where the ball *would be* in the future, after the next kick of the robot currently handling the ball.

In the future, this might be achieved in a better way, taking in account the positions of other robots to adjust the kick. From a technical point of view, we are also considering electing a virtual team "captain", that would take the decision for the whole strategy, which would considerably simplify the organization of team play.

5.4 Camera Model Calibration

In the current state of the competition, robots still fall frequently. The deformation that is caused on the mechanical parts has been a recurrent issue since the geometric model of the robot is constantly used in the vision pipeline. For example to compute the Cartesian egocentric distance between the robot and the detected ball on the ground. A dismountable, low cost calibration setup was designed specifically for calibration. The markers on the setup are detected with an off-the-shelf algorithm [6] by the robot whose cleats are used to ensure an accurate position on the ground. A dance-like motion was implemented so that the tags are seen from a multitude of perspectives.

A hundred samples, containing the joint positions of the robot and the measured coordinates of the tags in pixel space, are downloaded from the robot. Since the actual positions of the tags are known, an error can be calculated for each tag of each measure using the direct kinematics of the robot.

A black box optimization algorithm [7] is used to minimize a function of the errors, the inputs of the optimization being angular offsets in the camera joint (pitch, roll, yaw), in the neck joint (roll, yaw), in the IMU joint (pitch, roll) and a parameter that quantifies the measure noise in pixel space. 25% of the samples are used for cross-validation.

During the 2017 RoboCup, almost every robot was calibrated between matches. Offsets of more than 3 degrees were often found in-between.

6 Development and Workflow

6.1 Build System and Architecture

One of the main issue about software architecture is managing the dependencies with third party libraries, and also with our own different packages. We switched to catkin, which is a build system used by the ROS community [11]. We also developed custom scripts to be able to manage all our many repositories, which helps us to install and update automatically all the dependencies when installing the development environment on a new system.

Since the robots have the same architecture (x86 64 bits) than our computers, we can deploy directly the binaries on it without actually cross-compilation. Some flags are just provided to avoid using extended instructions like SSE4 unavailable on the robots.

In the future, we plan to switch to Intel NUC7i5BNK as main computer, which would provide better overall CPU performances.

6.2 Behavior Viewer

A considerable portion of our strategy is implemented using finite state machine representations. Even if this is a convenient tool to represent the behaviour of the robot, increasing the number of possible states and transitions makes it quadratically harder to validate all possible game scenarios. For instance, free kicks or yellow and red cards that were introduced in the 2017 rules made these states machines more complex.

In order to test our strategy, to reproduce a game situation and check how the robot behaves, a simulated mode has been implemented in our software. From a kinematics simulation (no physics), robot's state and environment are faked. We designed a tool called the *Behavior Viewer* (Fig. 6) which is used to setup and easily reproduce any wanted scenario by mocking the robot position, the ball and also some obstacles. Note that the exact same behaviour code is running in the fake environment and in our robots during actual games, including the referee instructions listening.

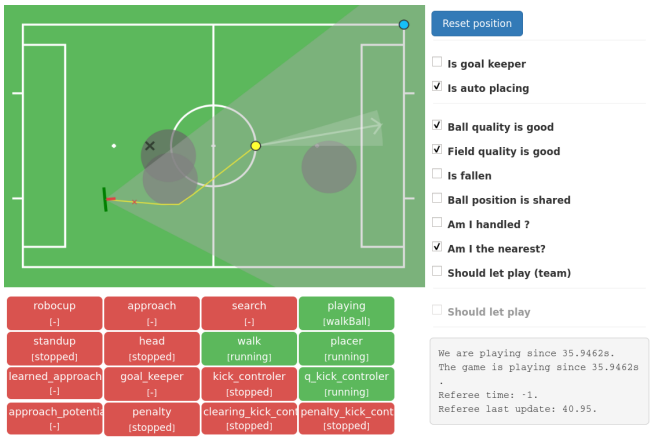


Fig. 6: An example of situation in the Behavior Viewer involving a robot playing with the ball, obstacles, trajectory and vision cone.

6.3 Monitoring and Replay

In addition of previously explained off-line testing process, it is also important to be able to monitor what is happening during the game and to investigate on the logs. Our online monitoring system, which captures the UDP packets that are used by our robots for strategy communication and produces a visualization, now also captures images from a webcam and logs everything so that we can replay a full game step by step. We can also synchronize the robot timestamped log files to have even more accurate data. Fig. 7 is an example of the result.

This system was extensively used for testing games and produces better unbiased opinions about what is needed to be fixed or improved.

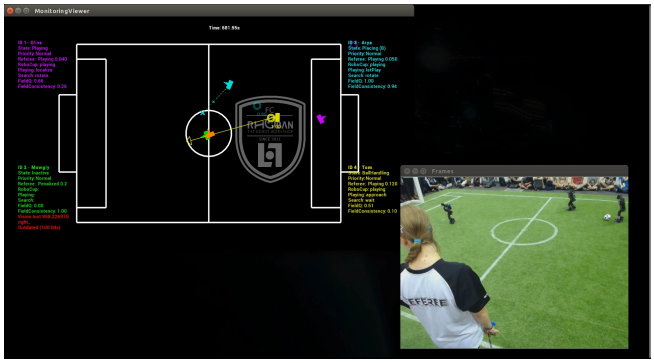


Fig. 7: Monitoring both the state of our robots and the images with full replay ability after the games

7 Conclusion

We now have full monitoring logs from all of the RoboCup games of this year. We plan to use them as an objective tool to define what are the priorities in order to improve the game on our robots.

An early version of other robots detection and obstacle avoidance was implemented. However this detection is still difficult, mainly because opponent robot appearances are unknown and building a training dataset is hard during the competition. Sharing images among teams maybe one of the possible solutions.

Tackling this problem would really improve the quality of the games. First because we would like the robots to be able to avoid from colliding each other in the future, but also use this information to compute more sophisticated strategy. Free kicks are also a good rule-tool for the referees to encourage detecting robots by penalizing the robots colliding each other.

References

1. Allali, J., Deguillaume, L., Fabre, R., Gondry, L., Hofer, L., Ly, O., N'Guyen, S., Passault, G., Pirrone, A., Rouxel, Q.: Rhoban football club: Robocup humanoid kid-size 2016 champion team paper. In: RoboCup 2016: Robot Soccer World Cup XX. Springer (2016)
2. Behnke, S.: Online trajectory generation for omnidirectional biped walking. In: 2006 IEEE International Conference on Robotics and Automation (ICRA 2006). pp. 1597–1603. IEEE (2006)
3. Bellman, R.: A markovian decision process. Journal of Mathematics and Mechanics pp. 679–684 (1957)
4. Broquere, X., Sidobre, D., Herrera-Aguilar, I.: Soft motion trajectory planner for service manipulator robot. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008). pp. 2808–2813. IEEE (2008)
5. Crow, F.C.: Summed-area tables for texture mapping. ACM SIGGRAPH Computer Graphics 18(3), 207–212 (1984)
6. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F.J., Marín-Jiménez, M.J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition 47(6), 2280–2292 (2014)
7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation 9(2), 159–195 (2001)
8. Hofer, L., Rouxel, Q.: An operational method toward efficient walk control policies for humanoid robots. In: Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017. pp. 489–497 (2017)
9. Hugel, V., Jouandeau, N.: Walking patterns for real time path planning simulation of humanoids. In: 2012 IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2012). pp. 424–430. RO-MAN, IEEE (2012)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
11. O'Kane, J.M.: A gentle introduction to ros (2014)
12. Passault, G., Rouxel, Q., Hofer, L., N'Guyen, S., Ly, O.: Low-cost force sensors for small size humanoid robot (video contribution). In: 2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015). pp. 1148–1148. IEEE (2015), `https://youtu.be/_d7Phe0qois`
13. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 779–788 (2016)
14. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
15. Rouxel, Q., Passault, G., Hofer, L., N'Guyen, S., Ly, O.: Rhoban hardware and software open source contributions for robocup humanoids. In: Proceedings of 10th Workshop on Humanoid Soccer Robots at IEEE-RAS Int. Conference on Humanoid Robots (2015)