Rhoban Football Club: RoboCup Humanoid Kid-Size 2016 Champion Team Paper

Julien Allali, Louis Deguillaume, 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. For its fifth participation to the RoboCup Kid-Size League, the Rhoban Football Club reached the first place of the competition in 2016 in Leipzig. This competition aims at opposing teams of small autonomous humanoid robots in real soccer games. Complex mechanics, electronics and software systems are needed to be implemented. In this paper, we summarize and describe some distinctive parts of our architecture. Going from our foot pressure sensors, our open-source alternative Dynamixel firmware, the use of kinematics models, the odometry and camera calibration to our perception system as well as simple but effective team play strategies.

1 Introduction

The Rhoban team is a young small research group in robotics from the computer science lab of the University of Bordeaux, France. It was founded by Olivier Ly in 2010 and is mainly interested in mobile robots, in particular legged and humanoids robots. Our main research activities target motion control and locomotion, including quadruped gaits, kick synthesis, walking control and learning to odometry and planning.

Our interest in the RoboCup Kid-Size is motivated by the very challenging competition along with the great community. The autonomous soccer Humanoid league is a well suited game to address state of the art locomotion and perception questions while allowing for mechatronic innovations. We support the emphasis of the competition on robustness and real world applications of robotics methods.

From our very first participation to the RoboCup in the Kid-Size league in 2011 we learned the importance of mechanical and electronics robustness which led to our second participation in 2013 where we managed to score our first goals. But only after a complete redesign of the walk and the vision system, we were able to reach the quarters in 2014. Then in 2015 – with the new rules including artificial grass – we achieved the third place while beginning to redesign our core software architecture. For our first time, we had the possibility to work on some more "advanced" components such as foot pressure sensors, localization

and odometry estimation from a kinematic model. All these developments eventually led us to the first place in 2016, after we finished the redesign of our core architecture along with numerous improvements of our system.

In the following, we describe specific technical points on our RoboCup 2016 architecture that could be interesting to share with the community.

2 Hardware

2.1 Foot Pressure Sensors

One of our robot hardware specificity is the use of strain gauges based force sensors in our feet [8]. The device takes advantage of the cleats that were added by many teams when switching from flat carpets to 3 cm artificial grass in 2015. Our feet have a rectangular support polygon while getting all the contact forces with the ground through four points. Each of these points are actually mechanical bars with strain gauges glued on it. These are resistor networks which values vary with mechanical deformation. See Fig. 1 of an overview of the device.

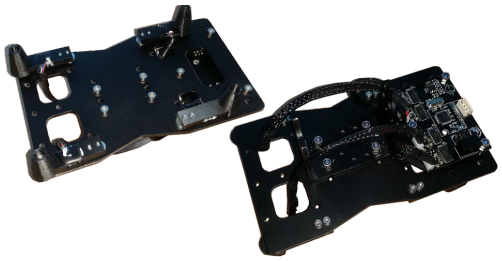

Fig. 1. An overview of our feet devices, below view (on the left) and top view (on the right)

We designed a custom electronic board that features amplifiers and a small microcontroller that is able to communicate directly through the Dynamixel serial bus, allowing daisy chaining with the last ankle motor and thus facilitating hardware integration. The strain gauges are low-cost off-the-shelf components that are easy to find on the market. The mechanical and electronic designs are open-source [10].

The strain gauges measures the normal component of the ground reaction force at each points of the cleats, which allows to compute the center of pressure (CoP) of the robot. The latter is the point P on the ground where the moment of the ground reaction forces vanishes [11]. It is then defined as:

$$P = \frac{\sum_i F_i J_i}{\sum_i Fi}$$

Where F_i and J_i are respectively the force measured and the position of the ith gauge. This is the geometric barycenter of the cleats positions with measured forces as weight.

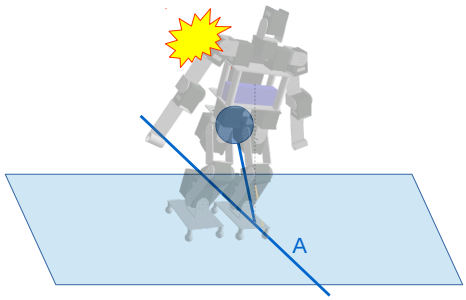

Fig. 2. Perturbations can lead the robot to roll over an edge of the foot, it is then under-actuated (here, it can not apply torque around the A axis)

We use these sensors to enhance the walking stability, tackling the problem of lateral balance. This was already discussed in [4] and [7] which proposed a capture step approach, using the data from the motor encoders and from the inertial measurement unit to estimate the position and speed of the center of mass and then using the inverted pendulum model to predict and adjust the support swap timing and position. Our approach have similarities since we also have a nominal trajectory for the center of pressure that is compared with the one we estimate using the sensor. Especially, a threshold ensure that the mass is indeed transferred to the other foot during support swing, and pauses the move if it does not[1].

To test this method, we used a setup similar to the one used in the push recovery challenge, where a 1 Kg mass was attached to a 1.9 m rope and dropped repeatedly on the robot at 0.5 and 0.6 m (Fig. 3). Here is a table summarizing the experiments results:

	Stabilisation enabled		Stabilisation disabled	
	Fall	No-fall	Fall	No-fall
50 cm	1	19	15	5
60 cm	9	11	14	6

2.2 Cameras and Lenses

In 2016 we decided to switch from standard webcams to small industrial cameras (See3CAM_11CUG from e-con Systems). This change was mainly driven by our

[1] This behavior can be seen in action in the video: `https://youtu.be/avJI_cBuMm0`

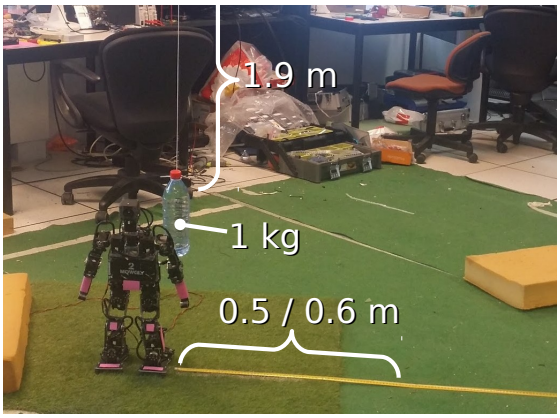

Fig. 3. Benchmark setup for the lateral perturbation rejection tests.

need to minimize the motion blur thanks to a global shutter. Moreover, the extended control over the camera's parameters also allowed for a slightly better color perception. Another strategy that proved quite convenient was the ability to use a relatively wide angle lens (about 100 ° aperture). This wide field of view was oriented vertically, which allowed the robot to see both its feet and the opponent goal in most cases but with the cost of a high image distortion.

Unfortunately during the competition we observed that the camera's USB 3.0 interface provoked interference with the WiFi of our robots. This problem seems to be well established now [1] and for the next year we plan to change again our cameras.

3 Electronics and Firmware

3.1 Custom Electronics Board

While all the high-level logic is executed in the robot embedded computer (Compulab Fitlet), the low-level communication with the motors and sensors is managed by a custom "router" board. This board is driven by a STM32 microcontroller (72 MHz 32 bit Cortex-M3 Arm) and is aimed at optimizing the communication between servomotors (Dynamixel TTL or RS485 bus), sensors (I2C or SPI) and the embedded computer via USB2 (Fig. 4).

This optimization is firstly done by separating the serial servomotor bus in three independent physical buses: one per leg and one for the upper body – the router board dealing with the three buses in parallel. Moreover, a pseudo "SYNC-READ" command is implemented. This command asks for multiple values to be read in one packet. This packet is actually processed by the microcontroller, which issues standard read commands sequentially, parallelizing communication through the three physical buses, and thus increasing the communication speed.

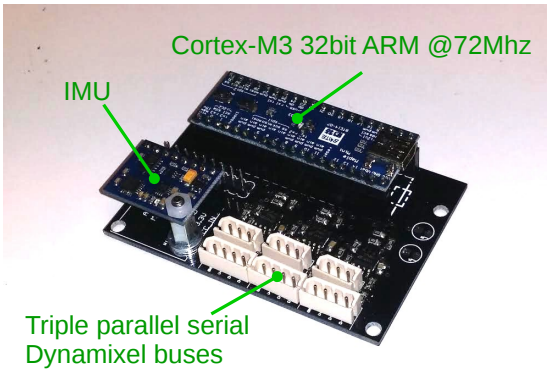

Fig. 4. Our custom board to handle devices communications.

When the robot is walking, a complete cycle, which includes a write and a read on all its devices takes less than 10 ms.

3.2 Dynaban Alternative Firmware for Dynamixel

A currently ongoing open-source project is the Dynaban custom firmware[2] for Dynamixel servo-motors. The aim of this project is to release to the community a working open-source implementation of the Dynamixel firmware in order to increase our control over the actuation. For example, a feed forward control has been implemented to improve the position's tracking accuracy and was originally presented in [2]. This controller makes use of polynomial position and torque trajectories continuously sent ahead of time. This firmware has been successfully tested (only in a standard mode without feed forward) during the whole RoboCup 2016 competition on one of our Sigmaban robot.

4 Software

Our entire code base is implemented in C++11 and currently has the following architecture:

- The low-level[3] thread is running the serial bus communication.
- The motion thread updates the models from low-level data, runs the team play and the game controller services, updates high level states and finally computes the walk and head motion.
- The perception thread sequentially reads a frame from the camera, extracts ball, goal post and field features and then runs the localization particle filter to update ball and absolute position estimation.

[2] Dynaban project: `https://github.com/RhobanProject/Dynaban`
[3] RhAL Rhoban low-level library: `https://github.com/Rhoban/RhAL`

4.1 Robot Kinematics Models

An important component of our architecture is a complete geometric and kinematics model implemented on top of the Rigid Body Dynamics Library (RBDL[4]). The RBDL C++ library is developed by Martin Felis (Heidelberg University) and implements the classical algorithms described in [3]. The model of the robot is directly exported from the Computer-Aided Design (CAD) software to standard URDF[5] file.

Three different instances of the model are used and presented in the following:

- "Goal model": motor's target positions,
- "Current model": state of the robot estimated from current values of sensors,
- "Past model": state of the robot slightly delayed.

The goal model is only considering the 20 joint degrees of freedom (DoFs). The current and past models are also considering the support state (left or right foot) and a 5 DoFs $(x, y, yaw, pitch, roll)$ floating base located at the center of the supporting foot.

"Goal" model: The goal model is used to represent the desired joint state of the robot. Analytical Inverse Kinematics (IK) is implemented for the leg (6 DoFs) and for the head (2 DoFs). The leg IK allows to design a walk and kick motion in Cartesian space by specifying the trunk and flying foot position and orientation. The head IK is used to control the neck yaw and pitch motors in order to target any given point in the Cartesian egocentric frame at the center of the camera's image.

"Current" model: The estimation of the current state of the robot is based on motor encoders, the Inertial Measurement Unit (IMU) and the foot pressure sensors. Firstly, the pressure sensors are measuring the weight on each foot. The foot with the more weight is considered the current support foot fixed on the ground. The IMU has 3 accelerometers and 3 gyroscopes filtered by a AHRS system[6] implemented on the embedded computer. This filter provides pitch and roll Euler angles of the robot's trunk. In addition, a simple integration of the Z gyroscope provides an absolute yaw estimation. This yaw estimation is obviously drifting but has proven to be quite accurate on small time scale. Typically, the drift is about 4 degrees after 30 seconds of robot manipulation. To estimate the robot state, the IMU is considered exact. Given the joint positions, the orientation of the support foot on the ground is set such that the trunk orientation matches the IMU computed roll, pitch and yaw angles. The possible discrepancy between the IMU and motor positions accounts for the soft ground and the mechanical backlash.

[4] C++ Rigid Body Dynamics Library: `http://rbdl.bitbucket.org/`
[5] XML Unified Robot Description Format: `http://wiki.ros.org/urdf`
[6] Open-Source Razor IMU AHRS filtering: `https://github.com/ptrbrtz/razor-9dof-ahrs`

Finally, the estimated state of the robot is also used to evaluate the odometry. The estimation of the robot's self relative motion is a very important ingredient of the localization process. At each step, the relative displacement between the new and the old support foot position is integrated. It comes out that the use of the foot pressure sensors instead of relying only on the feet kinematics is important to achieve accurate results. The accuracy of the odometry is further improved through a calibration process detailed in the 4.2 section.

"Past" model: The past model is used to provide an history of the model state at any point in the past. All the low-level data are stored for a fixed period of time and can be used to rebuild a model of the robot state in the past. In particular, this system is useful for the vision and localization components as images can take a few hundred milliseconds to be processed. It is then necessary to have access to a complete state of the model at the moment the image was taken.

This model is mainly used for the localization of the Cartesian egocentric position of an object on the ground at a given position on the image. And secondly, the prediction of the expected ball radius in pixels at a given position on current image, which is a strong criteria for ball or goal post false positive rejection.

It is to be noted that all these models allow for the external features (ball, goal post...) to be stored in the world absolute reference frame, taking into account the robot's displacement. For instance, even if the localization or ball detection process runs at low frequency, the walk controller always could have access to a fresh and updated ball relative position being patched up by the odometry integration and model kinematics.

4.2 Odometry and Camera Calibration

Instead of relying on classical visual odometry as many other teams do in RoboCup SPL [5], our odometry estimation is based on a simple kinematics integration and a good support foot detection. In addition, the accuracy of the estimation is improved thought a calibration process coming from our previous work [9].

The idea is to account for model errors and sliding ground contacts by learning a corrective model. The original work uses a motion capture setup and a non parametric non linear Locally Weighted Linear Regression (LWPR) method to learn a corrective function of the robot's relative displacement at each step.

This method has been simplified in order to be more conveniently used in the RoboCup context. Instead of the LWPR regression, a classical linear model is fitted. Instead of a complete motion capture setup which would be too challenging to deploy on a field during the competition, the following process is used: the robot is manually driven between two known points on the field several times (6 runs were used).

During each run, all low-level data are recorded and the robot is driven such as all walk directions (forward, backward, lateral steps) are explored. Then, the

robot displacements are replayed off-line and a black-box optimization algorithm (CMA-ES [6][7]) is applied to find the parameters of the model. The optimization try to minimise the error between the simulated robot final position (under current odometry correction) and actual known robot's displacement. With this procedure, the odometry accuracy typically achieves a drift of about 20 cm for a displacement of 2.5 m forward, which was sufficient for our needs.

Another issue requiring a frequent calibration concerns the deformation of mechanical parts. In particular, a deformation of the neck part holding the camera can result in a large distance estimation error of external objects. A discrepancy up to 5 degrees on the kinematics orientation between the trunk and the camera has been detected during the competition.

Here, the calibration is done by aiming the camera at known points on the ground. A correction of the geometry between the camera and the trunk of the robot is then computed by comparing the measured and expected positions of these known points.

It is to be noted that all these calibration procedure needed to be run constantly during the competition.

4.3 Vision Flexible Architecture

The real RoboCup environment being only known at the beginning of the competition, the algorithms used to detect key features often need to be adapted or even modified entirely on site.

In order to allow for a quick prototyping, we represent our vision algorithms as a directed acyclic graphs of independent filters. The topology of the graph and all the parameters of the filters are stored in XML configuration files which can easily be modified. The vision core system is then able to instantiate on the fly, OpenCV filters or custom algorithms based on this file. Basic monitoring and parameters updates are available online, without requiring any interruption of the program. Modifications of the topology of the graph require changes in the configuration file, but they do not require compilation. In order to reduce the computational burden of embedded vision, most of the filters heavily use regions of interest, detecting the areas susceptible to contain useful information in downscaled images.

The consistency and guaranty of continuous improvement of the vision algorithm is insured by a benchmark process which uses manually tagged images and compares them with the results of our algorithm. This allows us to run non-regressions tests to validate our modifications.

This setup allowed us to quickly adapt during the first few days and all along the competition. We were able to experiment several approaches in parallel and to choose the best one based on the benchmark results. Moreover, the quick development time also allowed us to produce different independent algorithms

[7] CMA-ES C++ library: `https://github.com/beniz/libcmaes`

to detect the ball and the goals and able to work on different conditions in order to be aggregated by the particle filter.

4.4 Localization with Particle Filter

In order to estimate the position of the robot on the field, we use a 3-dimensional particle filter in which each particle represents an estimate of the position and the orientation of the robot on the field. This filtering method allowed us to aggregate observations from different sources.

As observations, we used measurement of a magnetic compass, the goal posts, the borders and the corners of the field area. Since the measurements provided by the magnetic compass are particularly noisy, we only use it as a binary information, mainly to help disambiguation the field symmetry. Visual observations are scored according to the angle between the camera to the theoretical position vector and camera to the estimated position vector.

Since it was not possible for us to get rid of false positives, we decided to impose a minimum score on the potential of the particles given visual observations. This value was chosen according to the false positive rate which was provided by our benchmark system.

The mutation of the particles at each step was divided into two parts: controlled mutation based on odometry; and exploration. This system heavily relies on the corrected odometry which allows to strongly reduce the exploration strength.

Due to the limited computational power, we were forced to use a maximum of 1000 particles, which is quite small considering the size of the 3D state space. In order to face this problem, we introduced more a priori knowledge with special particle distributions which are used to reset the filter after specific game events such as kick-off or robot services. Likewise, the cases where the robot falls were handled by adding an uniform noise on all the particles and by using a random orientation.

For development and debugging, we can generate images summarizing the results of the detection and the current state of the particle filter (see Fig. 5).

4.5 User Interface and Configuration Tool

Being able to easily and very quickly debug and tweak some parameters on the robot is an essential element of the RoboCup competition. Our experience led us to the conclusion that a command line interface was far more efficient than a graphical one. So in order to fit our needs we developed the open-source project RhIO[8] (Rhoban Input Output Library).

This is a lightweight client-server library targeted to be integrated into existing code in order to monitor, debug and configure a running process in real time. The main user interface is a bash-like shell with a folder-file architecture. The project is described more deeply in [10]. Note that the network protocol used relies on TCP and can not be used for monitoring during games.

[8] RhIO Project: `https://github.com/rhoban/rhio`

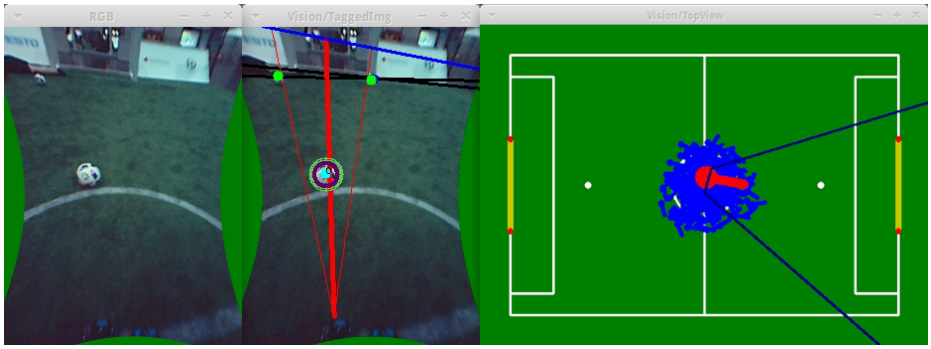

Fig. 5. The result of the vision and localisation processes. The tagged image in the middle is showing the ball estimated position and radius, goal post bases, goal center direction, field borders estimation in black and the horizon line in blue

4.6 Team Play

Compared to the RoboCup Standard Platform League, our team play is still very basic. Nevertheless, some simple and easy-to-implement robot's coordination have proven to be quite effective to improve game quality.

Our robots are continuously listening and broadcasting messages at 3Hz on the WiFi UDP. Note that a high broadcast frequency is needed to encompass the WiFi unstable quality. The packets are containing the:

- robot's unique id,
- ball position in egocentric frame and quality estimation,
- absolute position on the field and estimation quality,
- high level state (for monitoring),
- software errors (for monitoring).

The ball and field quality is an estimation between 0 and 1 of the confidence over the computed position provided by the particle filter.

Based on these information, our first team play strategy has been running since the RoboCup 2014 in João Pessoa. It implements a simple ball *mutex* area. There is a hierarchy among the robots according to their knowledge of the ball position. This system gives the priority to play the ball to the closest robot while the others keep a specific fixed exclusion distance from the ball. Note that the exclusion radius parameter is slightly different for each team member in order to prevent two side attackers to lay on the same circle.

So when the main attacker falls, loses the ball or fails, he either indicates no ball detection (zero quality) or stops broadcasting. Therefore, the next closer side attacker takes the ball lock and tries to recover the ball possession. The last feature of this simple behavior is that side teammates are following the main attacker, resulting in nice grouped progression on the field.

Another team play strategy which was only implemented during the very end of the RoboCup 2016 competition is the ball position sharing. When a robot is unsuccessfully looking for the ball during a fixed period of time and if another team member is localized and knows where the ball is with a sufficiently high quality, these information are shared. For example, this allows lost attackers to come back in defence when the goalie is detecting a nearby approaching ball.

4.7 Monitoring

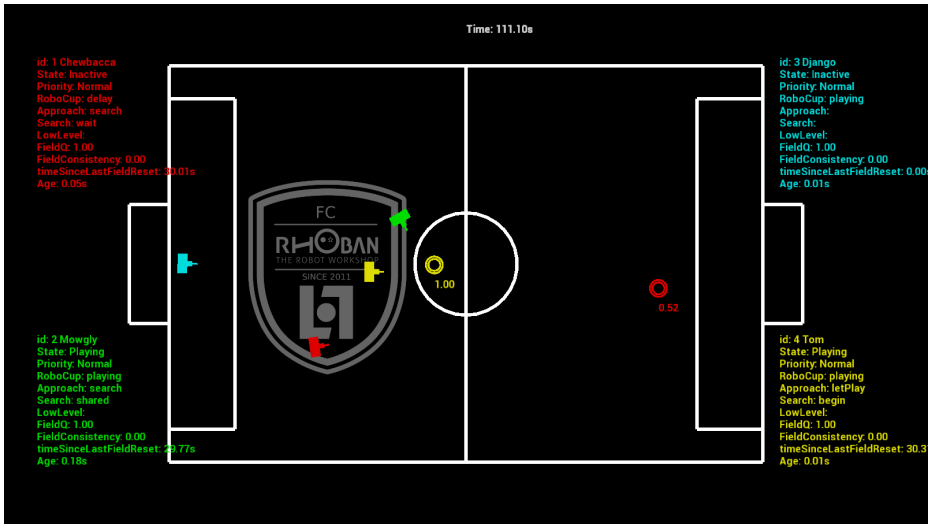

Fig. 6. Monitoring viewer showing in real time the robots state from UDP broadcast

By listening to the UDP broadcasts, an external software is able to monitor the robots internal state during the games. As shown in Fig. 6, we can see robots localization on field, ball estimated position, high level behaviors state as well as software errors. For example, this allow to clearly know that a Dynamixel cable or the USB camera is disconnected after a fall. This feature has proven to be important during games and allowed for a better pick-up or service management of the robots.

5 Conclusion

As always, a lot of points are calling for improvements. We will pursue our inspection of small industrial cameras and comparison of wide versus narrow aperture lenses for RoboCup vision. A better tracking accuracy could be expected if our walk and kick motions were taking advantage of the new Dynaban

feed forward controller. Concerning the software, a more automatic and faster camera calibration procedure would be more convenient to encompass the slow mechanical deformation. The major task for the localization process next year will be to break the field symmetry without the absolute magnetic orientation and still improve its accuracy. Either by recognizing our own goalie or either by detecting external features outside the field with a great caution to avoid moving spectators. Finally, an accurate localization and other robots detection are the last perception requirements to begin to develop real high level strategies similar to the NAOs in Standard Platform League. The Kid-Size league may not be that far from seeing ball passes between robots of the same team.

References

1. USB 3.0* Radio Frequency Interference Impact on 2.4 GHz Wireless Devices. Tech. rep., Intel Corporation (2012)
2. Fabre, R., Rouxel, Q., Passault, G., N'Guyen, S., Ly, O.: Dynaban, an open-source alternative firmware for dynamixel servo-motors. In: Symposium RoboCup 2016: Robot World Cup XX (2016)
3. Featherstone, R.: Rigid body dynamics algorithms. Springer (2014)
4. Graf, C., Röfer, T.: A closed-loop 3d-lipm gait for the robocup standard platform league humanoid. In: Proceedings of the Fifth Workshop on Humanoid Soccer Robots in conjunction with the (2010)
5. Hall, B., Harris, S., Hengst, B., Liu, R., Ng, K., Pagnucco, M., Pearson, L., Sammut, C., Schmidt, P.: Robocup spl 2015 champion team paper. In: Robot Soccer World Cup. pp. 72–82. Springer (2015)
6. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation 9(2), 159–195 (2001)
7. Missura, M., Behnke, S.: Lateral capture steps for bipedal walking. In: Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on. pp. 401–408. IEEE (2011)
8. Passault, G., Rouxel, Q., Hofer, L., N'Guyen, S., Ly, O.: Low-cost force sensors for small size humanoid robot. In: Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on (Video Contribution). pp. 1148–1148. IEEE (2015), `https://youtu.be/_d7Phe0qois`
9. Rouxel, Q., Passault, G., Hofer, L., N'Guyen, S., Ly, O.: Learning the odometry on a small humanoid robot. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE (2016)
10. 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, IEEE-RAS Int. Conference on Humanoid Robots, Seoul, Korea (2015)
11. Sardain, P., Bessonnet, G.: Forces acting on a biped robot. center of pressure-zero moment point. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 34(5), 630–637 (2004)