RoboCup 2016 Humanoid TeenSize Winner NimbRo: Robust Visual Perception and Soccer Behaviors

Hafez Farazi, Philipp Allgeuer, Grzegorz Ficht, André Brandenburger, Dmytro Pavlichenko, Michael Schreiber and Sven Behnke

Autonomous Intelligent Systems, Computer Science, Univ. of Bonn, Germany
`{farazi,pallgeuer}@ais.uni-bonn.de`, `behnke@cs.uni-bonn.de`
`http://ais.uni-bonn.de`

Abstract. The trend in the RoboCup Humanoid League rules over the past few years has been towards a more realistic and challenging game environment. Elementary skills such as visual perception and walking, which had become mature enough for exciting gameplay, are now once again core challenges. The field goals are both white, and the walking surface is artificial grass, which constitutes a much more irregular surface than the carpet used before. In this paper, team NimbRo TeenSize, the winner of the TeenSize class of the RoboCup 2016 Humanoid League, presents its robotic platforms, the adaptations that had to be made to them, and the newest developments in visual perception and soccer behaviour.

1 Introduction

In the RoboCup Humanoid League, there is an ongoing effort to develop humanoid robots capable of playing soccer. There are three size classes—KidSize (40-90cm), TeenSize (80-140cm), and AdultSize (130-180cm). Note that the overlap in size classes is intentional to facilitate teams to move up to higher size classes.

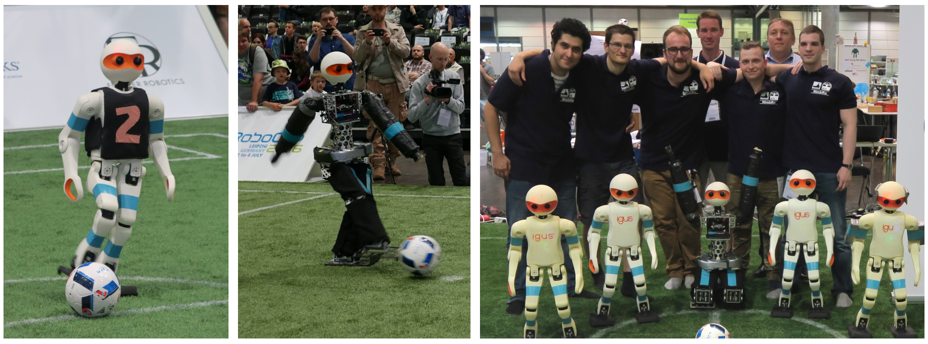


Fig. 1. The igus® Humanoid Open Platform, Dynaped and team NimbRo.

The TeenSize class of robots started playing 2 vs. 2 games in 2010, and a year later moved to a larger soccer field. In 2015, numerous changes were made in the rules that affected mainly visual perception and walking, namely:

- the green field carpet was replaced with artificial grass,
- the field lines are only painted on the artificial grass, so they are highly variable in appearance and no longer a clear white,
- the only specification of the ball is that it is at least 50% white (no longer orange), and
- the goal posts on both sides of the field are white.

Our approach to addressing these rule changes in terms of perception are given in Section 3.1.

In this year's RoboCup, we used our fully open-source 3D printed robots, the igus® Humanoid Open Platform [1], and the associated open-source ROS software. Furthermore, we revived one of our classic robots, Dynaped, with the same ROS software, by upgrading its electronics and PC and developing a new communications scheme for this to work. This was done in such a way that the robot hardware in use was completely transparent to the software, and abstracted away through a hardware interface layer. More details are given in Section 2. Both platforms are shown in Fig. 1, along with the human team members.

2 Robot Platforms

2.1 Igus Humanoid Open Platform

RoboCup 2016 was the first proper debut of the latest addition to the NimbRo robot soccer family—the igus® Humanoid Open Platform, shown on the left in Fig. 1. Although an earlier version of the robot had briefly played in 2015, 2016 was the first year where the platform constituted an integral part of the NimbRo soccer team. Over the last three years, the platform has seen continual development, originating from the initial NimbRo-OP prototype, and seven robots of three generations have been constructed. The igus® Humanoid Open Platform is 92 cm tall and weighs only 6.6 kg thanks to its 3D printed plastic exoskeleton design. The platform incorporates an Intel Core i7-5500U CPU running a full 64-bit Ubuntu OS, and a Robotis CM730 microcontroller board, which electrically interfaces the twelve MX-106R and MX-64R RS485 servos. The CM730 incorporates 3-axis accelerometer, gyroscope and magnetometer sensors, for a total of 9 axes of inertial measurement. For visual perception, the robot is equipped with a Logitech C905 USB camera fitted with a wide-angle lens. The robot software is based on the ROS middleware, and is a continuous evolution of the ROS software that was written for the NimbRo-OP. The igus® Humanoid Open Platform is discussed in greater detail in [1].

2.2 Upgraded Dynaped

Dynaped, shown in the middle in Fig. 1, has been an active player for team NimbRo since RoboCup 2009 in Graz, Austria. Through the years, Dynaped

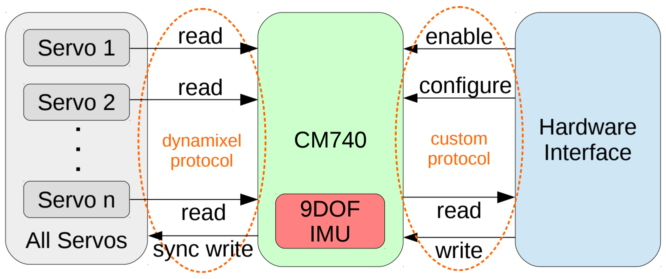


Fig. 2. Dynaped's custom communication scheme

has played both as a goalie and a field player during numerous competitions, contributing to the team's many successes. Dynaped's competition performance and hardware design, including features like the effective use of parallel kinematics and compliant design, contributed to NimbRo winning the Louis Vuitton Best Humanoid Award in both 2011 and 2013.

In 2012, our focus of development shifted towards the development of an open platform—the NimbRo-OP prototype, later followed by the igus® Humanoid Open Platform. Because of platform incompatibilities and a severe electrical hardware failure during RoboCup 2015, we decided to upgrade Dynaped to the newer ROS software, in a way that is transparently compatible with the igus® Humanoid Open Platform. The upgrade included both hardware and software improvements. In terms of hardware, Dynaped has been equipped with a modern PC (Intel Core i7-5500U CPU), a new CM740 controller board from Robotis, and a vision system as in the igus® Humanoid Open Platform, consisting of the same camera unit and 3D-printed head.

To adapt Dynaped to the new ROS software framework, a number of modifications had to be made. Thanks to the modularity of our software, only the low-level modules needed a specific reimplementation, while all of our high-level functionality that contributed to the success during RoboCup 2016 could be used untouched. At first glance, the only fundamental difference seems to be the utilisation of parallel kinematics, leading to the loss of one degree of freedom, but in fact, quite importantly, Dynaped still uses the older Dynamixel actuators. The used Dynamixel EX-106 and RX-64 have very similar physical properties to the MX-106 and MX-64 used in the igus® Humanoid Open Platform, but they lack a bulk read instruction, which is essential for allowing fast communications with multiple actuators with a single instruction. This limitation greatly reduces the control loop frequency, as each actuator needs to be read individually. This increases the latencies with each added actuator to the bus. To reduce these delays, Dynaped utilises a custom firmware for the CM740, which no longer acts merely as a passthrough from the PC to the servos. Instead, it communicates with both sides in parallel (see Fig. 2). On the actuator side, the CM740 queries all registered devices on the Dynamixel bus in a loop. Communications with the PC are performed using an extension of the original Dynamixel protocol, which allows the use of the same, well-developed error handling as our original

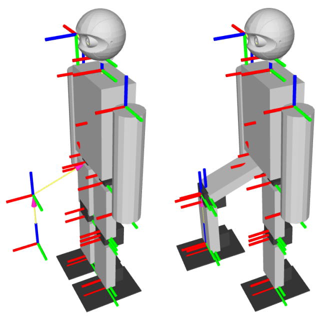


Fig. 3. Serial to parallel kinematic translation. Left: Before translation, virtual chain visible. Right: After translation.

firmware, as well as having the option to still use the CM740 as a passthrough device. The CM740-PC protocol has been extended by four new instructions:

- Configure extended packet communication,
- Enable extended packet communication,
- Send extended packet, and
- Receive extended packet.

To start the custom communication scheme, the hardware interface first sends a configuration packet containing a list of servo ID numbers, along with their respective model types. This informs the CM740 which servo registers it needs to keep reading from and writing to. Typically, these registers correspond to position, torque, and controller gain data. The read packets contain the most recent data from all of the Dynamixel devices, with an indication of how many times it has been read from since the last packet. The write packets include the current position setpoints and compliance values for the servos. In Dynaped's case, the packet transmission frequency is 100Hz, which allows all devices on the Dynamixel bus to be read at least once before a new read packet is sent. This transfer rate would not be achievable on Dynaped's hardware with the traditional request-response transmission paradigm.

Creating a model with parallel kinematics and using it in the hardware interface proved to be another challenge, as ROS does not natively support this. In order to translate between the serial and parallel kinematics, the created model has two sets of leg kinematic chains, a virtual serial one and the true parallel one that is actuated. The virtual kinematic chain receives the commands as-is from the motion modules, which the hardware interface then translates for the parallel chain (see Fig. 3) before sending a command to the actuators. In order to recreate the state of the parallel joints when reading out positions from the actuators, virtual joints have been added in order to offset the next link in the kinematic chain by the same angle that the joint has rotated. With these modifications, the robot can be seen as an igus® Humanoid Open Platform robot by the software, and thanks to our modular design approach, no robot-specific changes had to be done to any motion modules, or other higher level parts of our code.

3 Software Design

3.1 Visual Perception

The primary source of perceptual information for humanoid robots on the soccer field is the camera. Each robot is equipped with one Logitech C905 camera, fitted with a wide-angle lens that has an infrared cut-off filter. The diagonal field of view is approximately 150°. The choice of lens was optimised to maximise the number of usable pixels and minimise the level of distortion, without significantly sacrificing the effective field of view. Our vision system is able to detect the field boundary, line segments, goal posts, QR codes and other robots using texture, shape, brightness and colour information. After identifying each object of interest, by using appropriate intrinsic and extrinsic camera parameters, we project each object into egocentric world coordinates. The intrinsic camera parameters are pre-calibrated, but the extrinsic parameters are calculated online by consideration of the known kinematics and estimated orientation of the robot. Although we have the kinematic model of both robot platforms, some variations still occur on the real hardware, resulting in projection errors, especially for distant objects. To address this, we utilised the Nelder-Mead [2] method to calibrate the position and orientation of the camera frame in the head. This calibration is crucial for good performance of the projection operation from pixel coordinates to egocentric world coordinates, as demonstrated in Fig. 4. As a reference, the raw captured image used to generate the figure is shown in the left side of Fig. 5. More details can be found in [3].

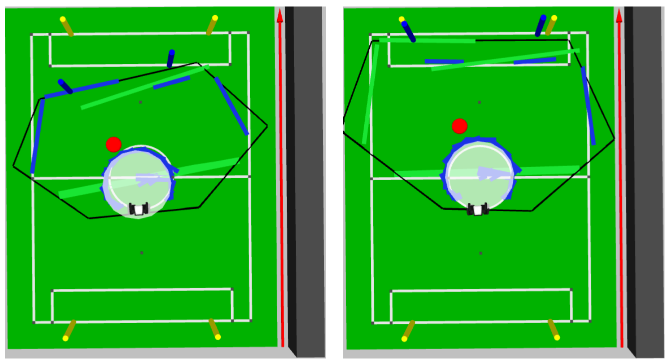


Fig. 4. Projected ball, field line and goal post detections before (left) and after (right) kinematic calibration.

Field Detection: Although it is a common approach for field boundary detection to find the convex hull of all green areas directly in the image [4], more care needs to be taken in our case due to the significant image distortion. The convex hull may include parts of the image that are not the field. To exclude these unwanted areas, vertices of the connected regions are first undistorted before calculating the convex hull. The convex hull points and intermediate points on

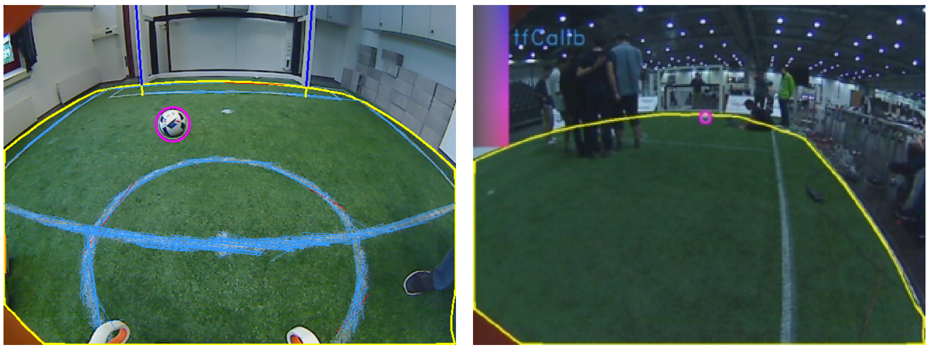


Fig. 5. Left: A captured image with ball (pink circle), field line (light blue lines), field boundary (yellow lines), and goal post (dark blue lines) detections. Right: Distant ball detection on RoboCup 2016 field.

each edge are then distorted back into the raw captured image, and the resulting polygon is taken as the field boundary. An example of the final detected field polygon is shown in Fig. 5.

Ball Detection: In previous years of the RoboCup, most teams used simple colour segmentation and blob detection-based approaches to find the orange ball. Now that the ball has a pattern and is mostly white however, such simple approaches no longer work effectively, especially since the lines and goal posts are also white. We extend [5], our approach is divided into two stages. In the first stage, ball candidates are generated based on colour segmentation, colour histograms, shape and size. White connected components in the image are found, and the Ramer-Douglas-Peucker [6] algorithm is applied to reduce the number of polygon vertices in the resulting regions. This is advantageous for quicker subsequent detection of circle shapes. The detected white regions are searched for at least one third full circle shapes within the expected radius ranges. Colour histograms of the detected circles are calculated for each of the three HSV channels, and compared to expected ball colour histograms using the Bhattacharyya distance. Circles with a suitably similar colour distribution to the expected one are considered to be ball candidates.

In the second stage of processing, a dense histogram of oriented gradients (HOG) descriptor [7] is applied in the form of a cascade classifier, with use of the AdaBoost technique. Using this cascade classifier, we reject those candidates that do not have the required set of HOG features. The aim of using the HOG descriptor is to find a description of the ball that is largely invariant to changes in illumination and lighting conditions. The HOG descriptor is not rotation invariant, however, so to detect the ball from all angles, and to minimise the user's effort in collecting training examples, each positive image is rotated by $\pm 10°$ and $\pm 20°$ and mirrored horizontally, with the resulting images being presented as new positive samples, as shown in Fig. 6. Greater rotations are not considered to allow the cascade classifier to learn the shadow under the ball. The described

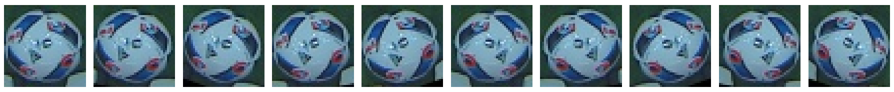


Fig. 6. Ball detection tracking data augmentation extending one positive sample (left-most) to ten, by applying rotations and mirroring operations.

approach can detect balls with very few false positives, even in environments cluttered with white, and under varying lighting conditions. In our experiments, we observed detection a FIFA size 3 ball up to 4.5 m away with a success rate above 80% on a walking robot, and up to 7 m away on a stationary robot, as shown in the right side of Fig. 5. It is interesting to note that our approach can find the ball in undistorted and distorted images with the same classifier.

Field Line and Centre Circle Detection: Due to the introduction of artificial grass in the RoboCup Humanoid League, the field lines are no longer clearly visible. In past years, many teams based their line detection approaches on the segmentation of the colour white [4]. This is no longer a robust approach due to the increased number of white objects on the field, and due to the visual variability of the lines. Our approach is to detect spatial changes in brightness in the image using a Canny edge detector on the V channel of the HSV colour space. The V channel encodes brightness information, and the result of the Canny edge detector is quite robust to changes in lighting conditions.

A probabilistic Hough line detector [8] is used to extract line segments of a certain minimum size from the detected edges. The minimum size criterion helps to reject edges from white objects in the image that are not lines. The output line segments are filtered in the next stage to avoid false positive line detections where possible. We verify that the detected lines cover white pixels in the image, have green pixels on either side, and are close on both sides to edges returned by the edge detector. The last of these checks is motivated by the expectation that white lines, in an ideal scenario, will produce a pair of high responses in the edge detector, one on each side of the line. Ten equally spaced points are chosen on each line segment under review, and two normals to the line are constructed at each of these points, of approximate 5 cm length in each of the two directions. The pixels in the captured image underneath these normals are checked for white colour and green colour, and the output of the canny edge detector is checked for a high response. The number of instances where these three checks succeed are independently totalled, and if all three counts exceed the configured thresholds, the line segment is accepted, otherwise the line segment is rejected.

In the final stage, similar line segments are merged together to produce fewer and bigger lines, as well as to cover those field line segments that might be partially occluded by another robot. The final result is a set of line segments that relate to the field lines and centre circle. Line segments that are under a certain threshold in length undergo a simple circle detection routine, to find the location of the centre circle. In our experiments, we found that this approach can detect circle and line segments up to 4.5 m away.

Localisation on the Soccer Field: Localisation of the robot on the soccer field—the task of estimating the 3D pose (x, y, θ) of the robot—is performed using the field line, centre circle and goal post detections. Each component of the 3D pose is estimated independently. To estimate the θ component, we use the global heading information from the magnetometer, and maintain an internal correction term based on the angular deviation between the expected and detected orientations of the white lines. This approach does not rely on having an accurate magnetometer output, and in experiments was able to correct deviations up to 30° coming from the magnetometer. Using the estimated θ, which is normally quite exact, we can rotate every vision detection to align with the global field coordinate system. The detected line segments can thereby be classified as being either horizontal or vertical field lines. In each cycle of the localisation node, we use the perception information and dead-reckoning walking data to update the previously estimated 2D location. For updating 2D location, we distinguish x and y component using estimated θ. The y component of the localisation is updated based on the y components of the detected centre circle, goal posts and vertical field lines. With the assumption that the robot is always inside the field lines, the vertical sidelines can easily be differentiated and used for updates. The x component of the localisation is analogously updated based on the x components of the detected centre circle, goal posts and horizontal field lines. The horizontal lines belonging to the goal area are discriminated from the centre line by checking for the presence of a consistent goal post detection, centre circle detection, and/or further horizontal line that is close and parallel. This approach can easily deal with common localisation difficulties, such as sensor aliasing and robot kidnapping. In contrast to some other proposed localisation methods for soccer fields, this method is relatively easy to implement and very robust. Our experiments indicate that the mean error of our localisation is better than what was reported in both [4] and [9].

3.2 Bipedal Walking

Motivated by the changed game environment at the RoboCup competition—the chosen application domain for our own use of the igus® Humanoid Open Platform—the gait generation has been adapted to address the new challenge of walking on artificial grass. The use of a soft, deformable and unpredictable walking surface imposes extra requirements on the walking algorithm. Removable rubber cleats have been added at the four corners underneath each foot of the robot to improve the grip on the artificial grass. This also has the effect that the ground reaction forces are concentrated over a smaller surface area, mitigating at least part of the contact variability induced by the grass.

The gait is formulated in three different pose spaces: joint space, abstract space, and inverse space. The *joint space* simply specifies all of the joint angles, while the *inverse space* specifies the Cartesian coordinates and quaternion orientations of each of the limb end effectors relative to the trunk link frame. The *abstract space* is a representation that was specifically developed for humanoid robots in the context of walking and balancing [10]. The abstract space reduces

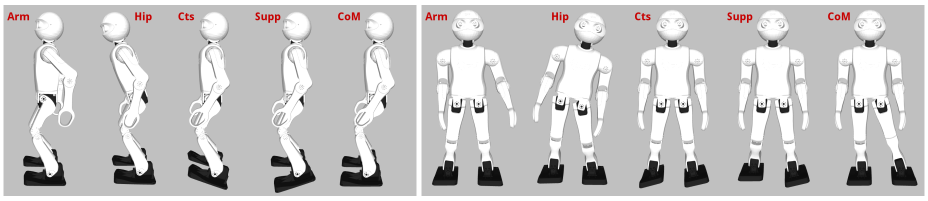


Fig. 7. The implemented corrective actions in both the sagittal (left image) and lateral (right image) planes, from left to right in both cases the arm angle, hip angle, continuous foot angle, support foot angle, and CoM shifting corrective actions. The actions have been exaggerated for clearer illustration.

the expression of the pose of each limb to parameters that define the length of the limb, the orientation of a so-called limb centre line, and the orientation of the end effector. Simple conversions between all three pose spaces exist.

The walking gait in the ROS software is based on an open loop central pattern generated core that is calculated from a gait phase angle that increments at a rate proportional to the desired gait frequency. This open loop gait extends the gait of our previous work [11]. The central pattern generated gait begins with a configured halt pose in the abstract space, then incorporates numerous additive waveforms to the halt pose as functions of the gait phase and commanded gait velocity. These waveforms generate features such as leg lifting, leg swinging, arm swinging, and so on. The resulting abstract pose is converted to the inverse space, where further motion components are added. The resulting inverse pose is converted to the joint space, in which form it is commanded to the robot actuators. A pose blending scheme towards the halt pose is implemented in the final joint space representation to smoothen the transitions to and from walking.

A number of simultaneously operating basic feedback mechanisms have been built around the open loop gait core to stabilise the walking. The PID-like feedback in each of these mechanisms derives from the fused pitch and fused roll [12] state estimates and works by adding extra corrective action components to the central pattern generated waveforms in both the abstract and inverse spaces, namely arm angle, hip angle, continuous foot angle, support foot angle, CoM shifting, and virtual slope. The corrective actions are illustrated in Fig. 7. The step timing is computed using the capture step framework [13], based on the lateral CoM state [14].

Overall, the feedback mechanisms were observed to make a significant difference in the walking ability of the robots, with walking often not even being possible for extended periods of time without them. The feedback mechanisms also imparted the robots with disturbance rejection capabilities that were not present otherwise. Reliable omnidirectional walking speeds of $21\,\mathrm{cm\,s^{-1}}$ were achieved on an artificial grass surface of blade length 32 mm. Over all games played at RoboCup 2016, none of the five robots ever fell while walking[1] in free space. Only strong collisions with other robots caused falls from walking Igus robot s could quickly recover using keyframe get-up motions [15].

[1] Video: `https://www.youtube.com/watch?v=9saVpA3wIbU`

3.3 Soccer Behaviours

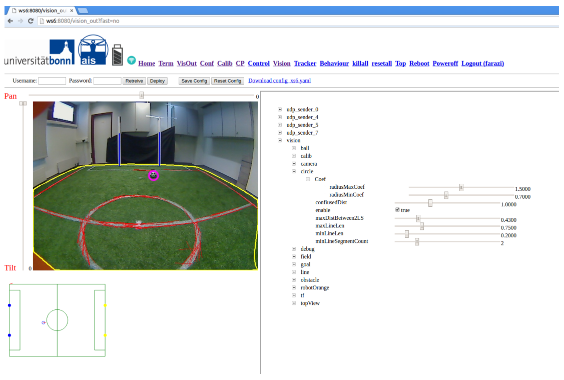
Given the current game state, as perceived by the vision system, the robots must autonomously decide on the higher-level actions to execute in order to try to score a goal. For this we use a two-layered hierarchical finite state machine (FSM), that has a tailor-made custom implementation for RoboCup soccer. The upper layer is referred to as the *game FSM*, and the lower layer is referred to as the *behaviour FSM*. Given the current required playing state of the robot, the former is responsible for deciding on a suitable higher-level action, such as for example "dribble or kick the ball to these specified target coordinates", and based on this higher-level action, the latter is responsible for deciding on the required gait velocity, whether to kick or dive, and so on.

In the order of execution, a ROS interface module first abstracts away the acquisition of data into the behaviours node, before this data is then processed, refined and accumulated into a so-called sensor variables structure. This aggregates all kinds of information from the vision, localisation, RoboCup game controller, team communications and robot control nodes, and precomputes commonly required derived data, such as for example the coordinates of the currently most intrusive obstacle, whether the current ball estimate is stable, and/or how long ago it was last seen. This information is used to decide on the appropriate game FSM state, such as for example *default ball handling*, *positioning*, or *wait for ball in play*, which is then executed and used to compute standardised game variables, such as for example *kick if possible* and *ball target*. These game variables, along with the sensor variables, are then used by the behaviour FSM to decide on a suitable state, such as for example *dribble ball*, *walk to pose* or *go behind ball*. The execution of the required behaviour state then yields the required low-level action of the robot, which is passed to the robot control node via the aforementioned ROS interface module, completing the execution of the soccer behaviours.

3.4 Human-Robot Interfaces

Despite being designed to operate autonomously, our robots still need suitable human-robot interfaces to allow them to be configured and calibrated. For the lowest and most fundamental level of control and operation, each robot can be launched and configured directly on the command line inside SSH sessions directly on the robot PC. This allows the greatest amount of freedom and flexibility in launching ROS nodes and checking their correct operation, but is also a complex and time-consuming task that is prone to errors and requires in-depth knowledge of the robot and software framework.

To overcome these problems, a web application was developed for the robot (see Fig. 8), with the robot PC as the web server, to allow standard web browsers of all devices to connect to the robot for configuration and calibration. This operates at a higher level of abstraction than the command line, and allows users to perform all common tasks that are required when operating the robot. This makes routine tasks significantly quicker and easier than on the command line,


ws6:8080/vision_out
ws6:8080/vision_out?fast=no
Home Term VisOut Conf Calib CP Control Vision Tracker Behaviour killall resetall Top Reboot Poweroff Logout (farazi)
Username: Password: Retreive Deploy Save Config Reset Config Download config_xs6.yaml
Pan
0
Tilt
0

```
udp_sender_0
udp_sender_4
udp_sender_5
udp_sender_7
vision
    ball
    calib
    camera
    circle
        Coef
            radiusMaxCoef          1.5000
            radiusMinCoef          0.7000
        confiusedDist              1.0000
        enable                     true
        maxDistBetween2LS          0.4300
        maxLineLen                 0.7500
        minLineLen                 0.2000
        minLineSegmentCount        2
    debug
    field
    goal
    line
    obstacle
    robotOrange
    tf
    topView
```

Fig. 8. A screenshot of the web application used to help calibrate the robot.

and avoids problems altogether such as hangup signals and resuming command line sessions. By exploiting the client-server architecture of web applications and the highly developed underlying web protocols, the connection is very robust, even over poor quality wireless network connections, and of low computational cost for the robot as most processing is implemented on the client side. The web application, amongst many other things, allows the user to start, stop and monitor ROS nodes, displays all kinds of status information about the robot, allows dynamic updates to the configuration server parameters, shows the processed vision and localisation outputs, allows various calibration and system services to be called, and allows the pose of the head to be controlled manually.

During operation, whether managed over the command line or the web server, the robot can be visualised using the RQT GUI, and dynamically reconfigured using the configuration server. This requires a live network connection to the robot for communication purposes. To configure the robot instantaneously, and without the need for any kind of network connection, a QR code detector has been implemented in the vision module. With this feature, arbitrary reconfiguration tasks can be effectuated due to the great freedom of data that can be robustly encoded in a QR code. QR codes can conveniently be generated on mobile devices, also for example with a dedicated mobile application, and shown to the robot at any time. The robot plays a short tune to acknowledge the QR code, serving as auditory feedback that the QR code was detected and processed.

4 Conclusions

In this paper, we described our platforms and approaches to playing soccer in the Humanoid TeenSize class. During RoboCup 2016, we successfully demonstrated that our robots could robustly perceive the game environment, make decisions, and act on them. Our team NimbRo TeenSize aggregated a total score of 29:0 over five games. We have released our hardware[2] and software[3] to GitHub with the hope that it is beneficial for other teams and research groups.

[2] Hardware: `https://github.com/igusGmbH/HumanoidOpenPlatform`

[3] Software: `https://github.com/AIS-Bonn/humanoid_op_ros`

5 Acknowledgements

We acknowledge the contributions of igus® GmbH to the project, in particular the management of Martin Raak towards the robot design and manufacture. This work was partially funded by grant BE 2556/10 of the German Research Foundation (DFG).

References

1. P. Allgeuer, H. Farazi, M. Schreiber, and S. Behnke, "Child-sized 3D Printed igus Humanoid Open Platform," in *Proceedings of 15th IEEE-RAS Int. Conference on Humanoid Robots (Humanoids)*, (Seoul, Korea), 2015.
2. J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.
3. H. Farazi, P. Allgeuer, and S. Behnke, "A monocular vision system for playing soccer in low color information environments," in *10th Workshop on Humanoid Soccer Robots, IEEE-RAS Int. Conference on Humanoid Robots*, (Korea), 2015.
4. T. Laue, T. J. De Haas, A. Burchardt, C. Graf, T. Röfer, A. Härtl, and A. Rieskamp, "Efficient and reliable sensor models for humanoid soccer robot self-localization," in *Fourth Workshop on Humanoid Soccer Robots*, pp. 22–29, 2009.
5. H. Schulz, H. Strasdat, and S. Behnke, "A ball is not just orange: Using color and luminance to classify regions of interest,"
6. U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, vol. 1, no. 3, 1972.
7. N. Dalal and B. Triggs, "Object detection using histograms of oriented gradients," in *Pascal VOC Workshop, ECCV*, 2006.
8. J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Vision and Image Understanding*, 2000.
9. H. Schulz and S. Behnke, "Utilizing the structure of field lines for efficient soccer robot localization," *Advanced Robotics*, vol. 26, no. 14, pp. 1603–1621, 2012.
10. S. Behnke, "Online trajectory generation for omnidirectional biped walking," in *Proceedings of 2006 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006.
11. M. Missura and S. Behnke, "Self-stable omnidirectional walking with compliant joints," in *8th Workshop on Humanoid Soccer Robots, Humanoids*, 2013.
12. P. Allgeuer and S. Behnke, "Fused Angles: A representation of body orientation for balance," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.
13. M. Missura and S. Behnke, "Balanced walking with capture steps," in *Robot Soccer World Cup*, pp. 3–15, Springer, 2014.
14. P. Allgeuer and S. Behnke, "Omnidirectional bipedal walking with direct fused angle feedback mechanisms," in *Proceedings of 16th IEEE-RAS Int. Conference on Humanoid Robots (Humanoids)*, (Cancún, Mexico), 2016.
15. J. Stückler, J. Schwenk, and S. Behnke, "Getting back on two feet: Reliable standing-up routines for a humanoid robot.," in *IAS*, pp. 676–685, 2006.