ITAndroids Humanoid Team Description Paper for RoboCup 2018

Arthur Azevedo, Davi Herculano, Daniela Vacarini, Gabriel Crestani, Igor Silva, João Filho, José Roberto Junior, Lucas Steuernagel, Marcos Maximo, Miguel Ângelo, Rodrigo Aki, and Samuel Pinto.

Autonomous Computational Systems Lab (LAB-SCA)
Aeronautics Institute of Technology (ITA)
São José dos Campos, São Paulo, Brazil
{arthurazevedo41,herculanodavi,danivacarini,gabrielpcrestani,asilvaigor, joaocdffilho,colombojrj,lucas.tnagel,miguelangelo.dss,krodrigo71008, sacepi}@gmail.com
mmaximo@ita.br
itandroids-humanoide@googlegroups.com
http://www.itandroids.com.br

Abstract. ITAndroids is a robotics competition group associated to the Autonomous Computational Systems Lab (LAB-SCA) at Aeronautics Institute of Technology (ITA). ITAndroids is a strong team in Latin America. In 2016, the group acquired a Robotis OP2 robot and material to build 4 more robots. In 2017, the team built 4 Chape robots and participated in RoboCup Humanoid KidSize for the first time. This paper describes our recent development efforts for RoboCup 2018.

1 Introduction

ITAndroids is a robotics research group at Aeronautics Institute of Technology. The group is multidisciplinary and contains about 45 students from different undergraduate and graduate courses. We are considered a reference team in Brazil and Latin America, where we have won 20 trophies in the last 6 years, including 5 in the Latin American Robotics Competition (LARC) 2017.

Regarding our humanoid team, we have been struggling with low cost robots since 2013, thus making it very hard to attain good results in competitions or even qualify for RoboCup. However in 2016, we have received a Robotis OP2 robot and enough material to build 4 other robots. In 2017, we developed our own robot Chape and competed in RoboCup Humanoid KidSize, which was a great opportunity for learning. In LARC 2017, we placed 1st and 2nd in Humanoid Robot Racing with our two robot designs, and placed 3rd in Humanoid KidSize.

This paper presents our recent efforts in developing a humanoid robot team to compete in RoboCup Humanoid KidSize 2018. The rest of the paper is organized as follows: sections 2 and 3 introduce the robot hardware. Sec. 4 presents our software architecture and tools. Sec. 5 explains our computer vision techniques. Sec. 6 shows our localization approach. Sec. 7 discuss our motion control algorithms. Finally, Sec. 8 concludes and shares our expectations for the future.

2 Mechanics

The mechanic hardware project is based on the DARwIn-OP humanoid robot. Some changes were made in order to meet the team's needs. The robot's CAD was developed using Dassault Systèmes SOLIDWORKS [7].

We designed the robot's torso placing special emphasis on the mechanics and electronics integration. The torso should accommodate PCBs and cables while allowing easy access and assembly of the electronic hardware. For this purpose, we chose a drawer-like configuration. The design leaves room for the airflow and fans were placed on the upper back side, in order to avoid overheating.

The team opted for 3D printed ABS covers. We used vertical ribs in order to increase the cover stiffness and to reduce the deformation caused by the ABS cooling during the print process. The covers also have small vents, in order to allow the airflow, whereas prevent synthetic grass and small particles from entering in the torso and harming the electronics.

The main goal of the head is to hold and protect the camera (Logitech C920). Moreover, it should be easy and quick to manufacture. Fig. 1 shows the CAD model of the head design holding the camera.

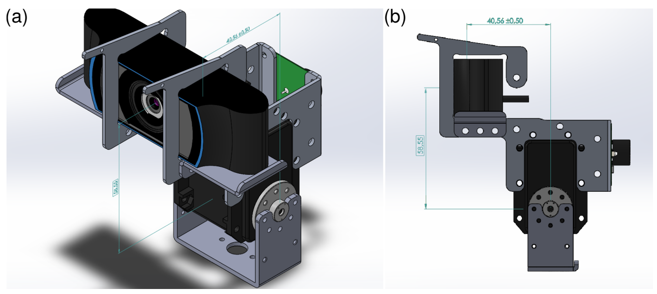


Fig. 1. Head of the Chape robot: a) isometric view; b) profile view.

The whole set encompasses 4 parts, manufactured using 1.5 mm sheets of aluminium 5052-H34. However, the current design showed to be not sufficiently resistant to withstand the falls and deformed. To resolve this, we intend to reinforce the part in the most damaged regions and perform structural analysis.

The lower arms design were upgraded from the model used in RoboCup 2017, since the previous model suffered large displacements under the stand up movement loads. The arms new requirement is that the arm tip displacement shall be lower than 5 mm under the stand up movement loads.

The final design has only 3 parts. The resulting model was simulated in a finite element method (FEM) analysis software with loads similar to those from the stand up movement. Fig. 2 shows this boundary condition: the magnitude of both forces is 30N and the blue surface is considered fixed, which is a more severe condition than those faced by the arms. In the FEM simulation, the arms tip displacement was equal to 4mm, which satisfies the requirements. Moreover, the resulting arm was manufactured and the robot was able to stand up, as desired.

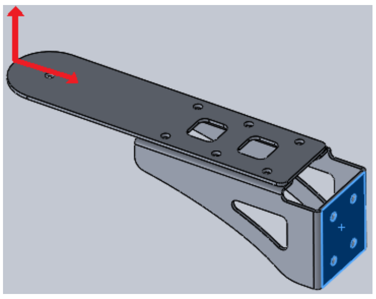


Fig. 2. Applied loads and boundary conditions. The red vectors represent the applied loads and the blue surface represents a fixed boundary condition.

3 Electronics

The electronic system of our Chape robot is in its second version. The architecture follows a modular approach, which allows for fast development and efficient problem isolation. Figure 3 illustrates our electronic modules and their connections.

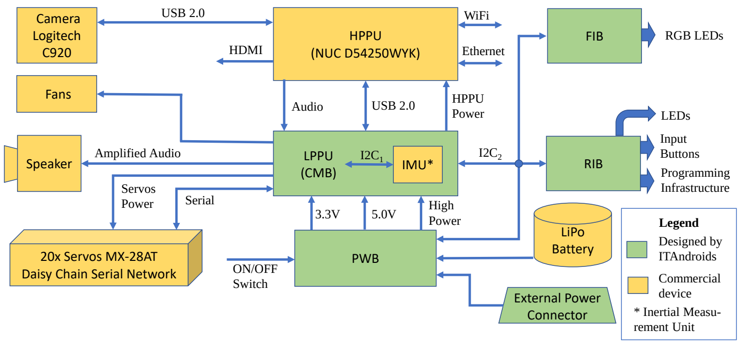


Fig. 3. Chape Electronics Architecture.

The HPPU (High Performance Processing Unit) is an Intel NUC D54250WYK computer. The communication between HPPU and LPPU (Low Performance Processing Unit), implemented by CMB (Control and Monitoring Board), is through USB 2.0. The boards were developed using Altium and have the following functionalities:

– CMB (Control and Monitoring Board): this board deals with low level hardware. It is responsible to communicate with the servos, acquire and interpret IMU data, notify environment messages through the LEDs and buzzer and control the power distribution to the servos and NUC.
– PWB (PoWer Board): the battery and external power can be connected in this board. It measures power consumption and creates the regulated 3.3 V and 5 V power buses.
– RIB (Rear Interface Board): it is an interface with buttons to facilitate command input, with programming infrastructure and with diagnosis LEDs.
– FIB (Front Interface Board): is a board made to be an interface which issues luminous signals that indicate the critical robot status, such as high temperature and low battery.

Due to supplier problems, the LiFe battery needed to be replaced by a 4 cells LiPo battery. With a higher voltage battery, the 5V regulator was replaced to support the battery voltage. Finally, the current system lacks robustness, having occasional problems with electrical contact and communication with the inertial unit, therefore a third iteration is under design for RoboCup 2018.

4 Software Architecture and Tools

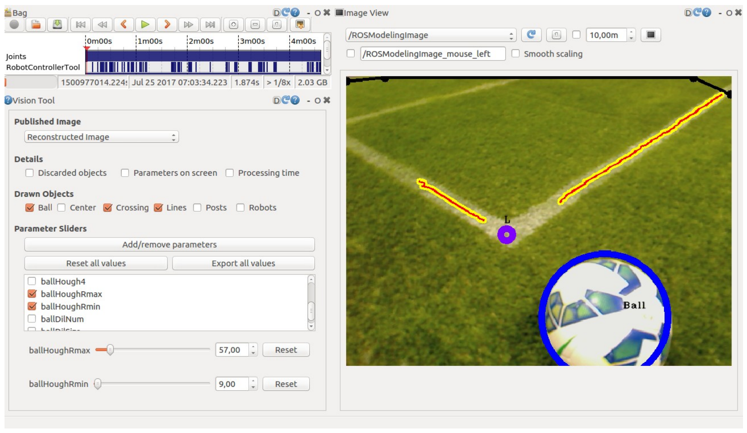
We use a module-based layered approach for code architecture. The main third-party libraries used are OpenCV (computer vision) [4], Eigen (linear algebra) [8], Boost (general purpose) [1], Tensorflow (machine learning) [2] and Keras (neural network) [6]. We heavily rely on the Robot Operating System (ROS) framework [14] and its related tools for testing and debugging purposes. We also use Qt [9] for graphical interface in our debugging tools. We expect to develop a full Humanoid Kid-Size simulation soon using Gazebo [5].

A telemetry system was developed for debugging and calibration purposes. It allows real-time feedback, robot controlling, data logging and data replaying. It uses rqt, the Qt-based framework for GUI development for ROS, and rviz, the 3D visualization tool for ROS, for easy user interfacing. Fig. 4 shows the telemetry tool working. In this figure, a log is being played and processed in the top left plugin, “Bag”. The vision algorithm is tested on the right, “Image View”, according to the commands on the bottom left plugin, “Vision Tool”. This allows computer vision debugging without the need of a real robot.

A world modeling and a robot controller plugins were also developed. The first is used to visualize the output of the localization algorithm using rviz and the second is used to test modeling algorithms without autonomous decision making.

5 Computer Vision

Our vision system uses a base code for the Standard Platform League made publicly available by the Austin Villa team in 2012 [3]. The main differences



Published Image: Reconstructed Image

Details: ☐ Discarded objects ☐ Parameters on screen ☐ Processing time

Drawn Objects: ☑ Ball ☐ Center ☑ Crossing ☑ Lines ☐ Posts ☐ Robots

Parameter Sliders: ☐ ballHough4 ☑ ballHoughRmax ☑ ballHoughRmin ☐ ballDilNum

ballHoughRmax 57,00

ballHoughRmin 9,00

Fig. 4. rqt framework illustrating the vision debugging tool.

between the released code and ours are related to the detection of the white ball, the field border, the white goalposts, and general bug fixes.

The first step in the vision pipeline is the color segmentation, which uses a color table generated by manually classifying pixels in an image and training a Neural Network classifier. Currently we are facing a challenge in order to classify correctly the grass due to changes in illumination throughout the field so we are testing alternatives such as pre-filtering and post-processing the classified image, as well as improving the neural network algorithm for better color classification.

The field border is calculated using the algorithm described in [16]. This algorithm searches the segmented image in a few vertical lines giving a score for each point on the line based on the green and non-green pixels. The maximum score is chosen as the field border on that line. The field border is used to filter "floating" objects, which are above the horizon line. The next step detects the lines and uses this field border to detect the posts.

To detect the white ball, we developed a convolutional neural network based in the model described in YOLO (You Only Look Once) papers [15]. However, we modified the original FastYOLO architecture to reduce computational cost. To do so, we reduced the number of convolution layers and the number of filters in each convolution. As a result, our final convolutional network has a total of eight layers of 2D convolution, as shown in Fig. 5, still following the model described in YOLO papers. The neural network is trained using Tensorflow [2] and Keras [6] libraries, and it uses Tensorflow [2] to run in real-time.

Our algorithm reduces the size of the image from 640x480 to 320x240 to obtain speed, then it creates a grid of 20x15 to return five values for each cell of this grid. The first value is the probability of the center of the ball center's being located inside the cell, the second and third are the coordinates X and Y that represent the center of the ball according to the cell and the last two values are the height and width of the ball according to the whole image. After-

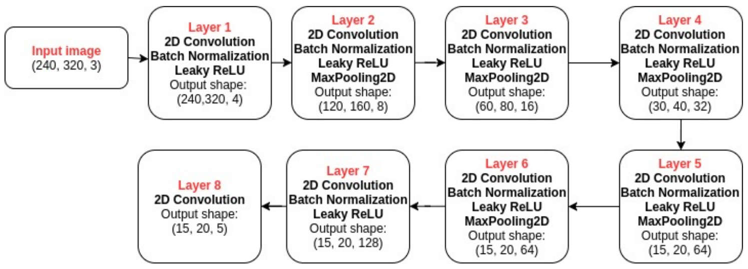


Fig. 5. Flow chart of the convolutional neural network we built. The image show also the shape of the output data for each layer. The names for each function used within layers are defined in Keras documentation [6].

wards, we implemented a simple algorithm to find which cell contains the highest probability of having a ball, so as to know exactly its coordinates.

6 Localization

In order to solve the global localization problem, we use an extended Monte Carlo Localization (MCL) technique, described in [17, 13]. In this MCL adaptation, it is possible to solve the kidnapping problem using sensor reset, which occurs based on the likelihood of the observation in a given moment. Fig. 6 shows the pose estimation based on a recorded log using the DARwIn-OP robot. The first image shows the detected features in a frame, followed by their projections on the 3D space and finally the pose estimation and particles for that instant in time.

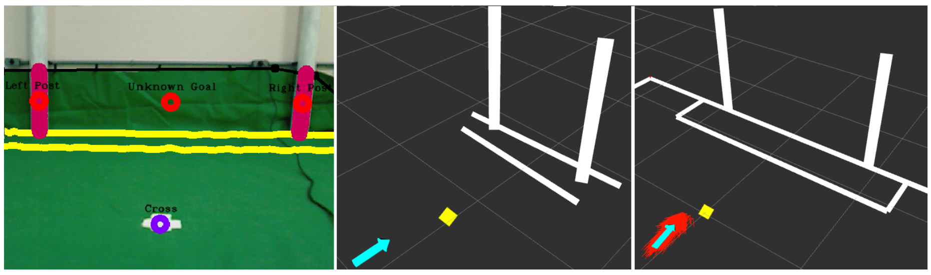


Fig. 6. Pose estimation process based on a recorded log.

To face the landmark ambiguity problem, we adopt an approach based on the maximum observation likelihood, the same strategy we use for field lines observation in Soccer 3D, which is described with more details in [13].

The techniques adopted to solve the kidnapping problem are based in [16], in which multiple features observed at the same time sometimes allow extracting reset poses. However there is still ambiguity involving which quarter of the field the real pose is due to symmetry, so the possible poses are filtered by keeping record of the field quarter the robot is currently located.

7 Motion Control

For walking and kicking, we use the ZMP-based algorithms described in [11]. In 2017, we augmented these algorithms with gravity compensation.

Gravity creates undesired torques at the robot's joints, which create errors in position tracking. By knowing each joint position and each part's center of mass and mass, the expected torque due to gravity at each joint may be computed. Hence, the robot may cancel this gravity effect by applying torques of same magnitude and opposite directions in a feedforward manner. Since the MX-28AT are position-controlled servos, we use a mathematical model of the servo [10] to transform these torques into position commands.

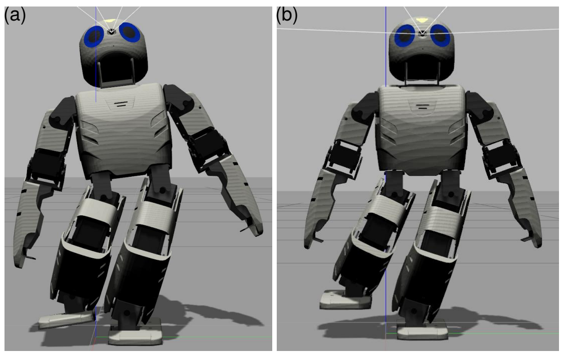


Fig. 7. Snapshots from the Gazebo simulation when the robot is with its right foot off the ground to show the effect of gravity compensation: (a) without gravity compensation; (b) with gravity compensation.

For the getting up movements, we are using the original Robotis OP2 keyframes with slight modifications to make them work on artificial grass. For more information about our keyframe movements framework, please refer to [12].

8 Conclusions and Future Work

This paper presented the recent efforts of ITAndroids group in RoboCup Humanoid KidSize. In 2018, we built 4 new robots based on the DARwIn-OP and participated in the RoboCup Humanoid League. Now we are working on improving the system's robustness and correcting its flaws. Besides, we want to share this project with the community when it is ready, since many teams complain about the difficulty of exactly reproducing the DARwIn-OP robot.

Furthermore, we were able to develop a new ball detection algorithm, as well as significantly improve our motion algorithms. We also made advancements on our localization and debugging systems. Nevertheless, our software still lacks the robustness and performance needed for being strongly competitive in the current level of the competition. Thus, our development efforts will be focused on improving the code behavior in the real robots.

Acknowledgment

We thank our sponsors Altium, ITAEx, Metinjo, Micropress, Poliedro, Poupex, Rapid, and Solidworks. We also acknowledge Mathworks (MATLAB), Atlassian (Bitbucket) and JetBrains (CLion) for providing access to high quality software.

References

1. *BOOST C++ Libraries*. `http://www.boost.org`.
2. Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
3. Samuel Barrett, Katie Genter, Yuchen He, Todd Hester, Piyush Khandelwal, Jacob Menashe, and Peter Stone. The 2012 UT Austin Villa code release. In *RoboCup-2013: Robot Soccer World Cup XVII*. Springer Verlag, 2013.
4. G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.
5. Dr. Philippe Capdepuy. DARwIn-OP Gazebo Simulation Model, 2016.
6. François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.
7. Dassault Systèmes. Solidworks, 2017.
8. Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.
9. The Qt Company Inc. Qt 5, 2017.
10. M. R. O. A. Maximo, C. H. C. Ribeiro, and R. J. M. Afonso. Modeling of a position servo used in robotics applications. In *Proceedings of the 2017 Simpósio Brasileiro de Automação Inteligente (SBAI)*, Porto Alegre, SC, Brazil, 2017. SBA.
11. Marcos R. O. A. Maximo. Omnidirectional ZMP-Based Walking for a Humanoid Robot. Master's thesis, Aeronautics Institute of Technology, 2015.
12. Francisco Muniz, Marcos Maximo, and Carlos Ribeiro. Keyframe Movement Optimization for Simulated Humanoid Robot using a Parallel Optimization Framework. In: Latin American Robotics Symposium. In *Proceedings of the the 2016 Latin American Robotics Symposium (LARS)*, October 2016.
13. Alexandre Muzio, Luis Aguiar, Marcos Maximo, and Samuel Pinto. Monte Carlo Localization with Field Lines Observations for Simulated Humanoid Robotic Soccer. In: Latin American Robotics Symposium. In *Proceedings of the the 2016 Latin American Robotics Symposium (LARS)*, October 2016.
14. Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
15. Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.
16. Thomas Röfer, Tim Laue, Yannick Bülter, Daniel Krause, Jonas Kuball, Andre Mühlenbrock, Bernd Poppinga, Markus Prinzler, Lukas Post, Enno Roehrig, René Schröder, and Felix Thielke. B-human team report and code release 2017. 2017.
17. W. Burgard S. Thrun and D. Fox. *Probabilistic Robotics.* MIT Press, 2005.