ITAndroids Humanoid Team Description Paper for RoboCup 2024

Diogo Bueno, Felipe Farah, Gabriel Padilha, Gustavo Beato, Marcos Levi, Marcos Maximo, Matheus Defilipo, Narayane Medeiros, Samuel Afonso, Thiago Galante, Victor Davi, Vitor Betto and Willian Teleken

Autonomous Computational Systems Lab (LAB-SCA)
Aeronautics Institute of Technology (ITA)
São José dos Campos, São Paulo, Brazil
{dibuenor,farah.affonso,gustavo.beato2015,gplferreira07,narayane.rm,samuel.afonsodesouza,thigalante,victordavisb,vitorbetto}@gmail.com
{willianteleken,m_levipintosousa,matdefilipo}@hotmail.com
mmaximo@ita.br
http://www.itandroids.com.br

Abstract. ITAndroids is a robotics competition group associated with the Autonomous Computational Systems Lab (LAB-SCA) at Aeronautics Institute of Technology (ITA). ITAndroids is a reference team in Latin America, having won more than 75 awards in robotics competitions in the last 12 years. In 2017, the team developed the Chape humanoid robot, and built four units to participate in RoboCup Humanoid KidSize for the first time. Since then, the team has been evolving the robot's hardware and software while participating in many competitions, especially RoboCup and Latin American Robotics Competition (LARC). The team also designed the Chape G2 robot, the second generation of Chape, which is currently under construction and testing. This work describes our recent development efforts for RoboCup 2024.

1 Software Architecture and Tools

We use a module-based layered approach for code architecture[1]. The main third-party libraries used are OpenCV (computer vision) [3], Eigen (linear algebra) [8] and Boost (general purpose) [1]. We heavily rely on the Robot Operating System (ROS1) framework [14] and its related tools for testing and debugging purposes. We also use Qt [10] for graphical interface in our debugging tools. A full Humanoid Kid-Size simulation was also developed using Gazebo [4] and a Chape model for Webots is in the final stages.

A telemetry system was developed for debugging and calibration purposes. It allows real-time feedback and control, as well as data logging and replaying. It uses rqt, the Qt-based framework for GUI development for ROS1, and rviz, the

[1] Available open source code from team: https://gitlab.com/itandroids/open-projects

3D visualization tool for ROS1, for easy user interfacing. This system has been extremely useful for vision and localization debugging. This telemetry system has been updated to allow the reception of debugging information of multiple robots during the game when connected to the game controller network.

The simulation in Gazebo consists of a field with two goals, one ball, and one Chape robot. There are a few significant differences from the real world, but it is accurate enough to test both the localization and the decision making algorithms. Using Webots, we can test out how the robot behaves in a match like situation, including the Game Controller states and RoboCup rules.

2 Motion Control

For walking and kicking, we use the ZMP-based algorithms described in [11]. We augment these algorithms with gravity compensation and a torso stabilization controller [12]. Since these techniques strongly rely on dynamics models, we developed a system identification technique to obtain a better mass distribution model experimentally through foot pressure sensors measurements [6] instead of using data from CAD files. However, we have not executed this process in the Chape robot yet.

Our kicking algorithm is based on splines: the x (forward), z (vertical), and θ (pitch) trajectories of the kicking foot are defined by points that specify the coordinate value at a given time. Then, the points are interpolated using separate cubic splines for each coordinate. The algorithm is only able to execute forward kicks for now, but we expect to generalize for other directions.

3 Computer Vision

Our current approach on computer vision is based on Nimbro team's 2015 paper [7]. This approach uses a convex hull algorithm to detect the field and a hough line detector for the field lines. We do not detect the penalty cross due to the number of false positives encountered. All of the code was made by the team from scratch. The ball and goalposts are detected using a convolutional neural network algorithm.

The first step on the vision pipeline is color segmentation on the image filtered by a Gaussian Blur. The color segmentation uses a color table generated by manually labeling pixels in various images and training a Neural Network classifier. To avoid the same color having two classes, for each labeled pixel the classes are given scores and the class with the highest score is selected as the input of the neural network. Data augmentation is also used to improve performance.

Semantic Segmentation technique classifies the pixels of the images received as input, and is especially useful in robot soccer for identifying elements such as field lines, where bounding boxes are not feasible. Semantic Segmentation Networks operate using an Encoder-Decoder model. They first extract information

from the image, reducing dimensions and increasing channels, and then generate segmentation maps by reversing this process. U-Net and SegNet are two fundamental networks in this context.

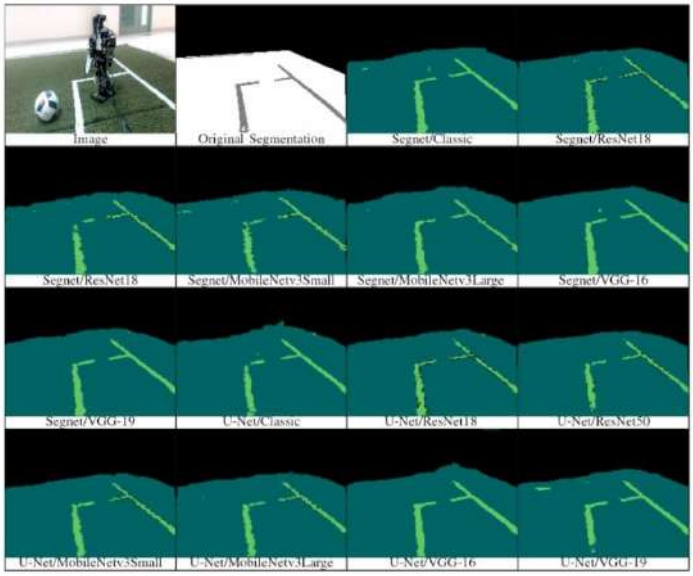

Fig. 1. Examples of segmentations denerated by each ED-CNN [16]

As part of the 2023 Scientific Initiation research [16], 14 Encoder-Decoder neural networks were evaluated to detect the field and the lines. In addition to U-Net and SegNet, other networks were built using the decoders from these networks. Six pre-trained image classification models were used as encoders: ResNet-18, ResNet-50, VGG-16, VGG-19, MobileNetv3Small and MobileNetv3Large. These classification networks are validated alternatives for extracting information from images, a key objective in the encoding process. The evaluation criteria included accuracy on the test dataset, Intersection-over-Union (IoU) for the lines and average inference time, which is important for applications in mobile robotics due to the need for low latency in processing the images observed by the robot. The results showed that the networks created with the U-Net Decoder, combined with MobileNetv3Large or ResNet-18 as the encoder, achieved the best trade-offs between speed and performance.

To detect the white ball and the field's goalposts, we have a convolutional neural network based on the model described in YOLO papers [15]. From the original TinyYOLO architecture, we decreased the number of filters in each layer [20], for our robot has limited processing power. Furthermore, we implemented the Deep Residual Learning for Image Recognition technique [9] to provide more

accurate detections with little increase in computational complexity. As a result, our final convolutional network has a total of nine layers of 2D convolution.

Our algorithm reduces the size of the image from 640x480 to 160x120, then 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 object 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. Afterwards, we implemented a simple algorithm to find which cell contains the highest probability of having a ball, to know exactly its coordinates. For the detection of goalposts, our vision algorithm has the same aforementioned logic, however, instead of looking for the whole goalpost, we only detect the goalpost's vertical supports. Their location is essential to the robot's localization.

We collected a data set to train the neural network and expanded the use of data augmentation to increase the variety of images. Such a technique allowed us to decrease the number of false positives. Currently, we are switching to using the DarkNet's implementation of YOLOv8 for training while inference is done through OpenCV DNN. Moreover, we are using the TORSO21 dataset [2] for training our neural network. We still have to completely integrate this new network into our code.

The transformation from the camera frame to the egocentric world frame proved to be a big challenge due to latencies associated with the system and the mechanical distortions. To solve this problem we consider the camera's timestamp as a reference for the image, and the joint and torso inclination information is synchronized with it. To achieve better results, we also compensate a fixed sensor delay in our torso observer model (2 ms), which was tuned manually.

Fig. 2. Chape robot calibration using an Aruco code carpet.

The mechanical distortions are corrected by associating rotation offsets to the torso and camera, as well as translation offsets to the camera and torso height. These distortions are calibrated using a routine consisting of the robot in stopped

position moving its head detecting Aruco Codes known positions, as shown in Fig. 2. From last year's technique, we improved the result by Aruco codes instead of QR-Codes, because of better detection, and placing codes further away to improve precision at higher distances. Moreover, inspired by other teams[17], we reduced the number of offsets used for optimization.

4 Localization

To solve the global localization problem, we use a standard particle filter (Monte Carlo Localization), as described in [18, 13]. Localization is challenging in Humanoid KidSize due to landmark ambiguity. The lack of unique features makes initializing the filter using a uniform distribution risky, since the filter may converge to the wrong side of the field. Therefore, in the beginning, our algorithm distributes the particles between the 4 possible starting poses in the soccer field, as stated in the rules [5]. Then, resampling is disabled while the head does a 180 degrees scan, accumulating information from the whole scan in the particles' weights before the first resampling. This proved very robust in initializing the filter to the correct robot pose.

Using the Webots simulator to guide our development, we made improvements to our algorithm throughout 2021: parameter tuning, improving the resetting policy, and using the Game Controller states to do so.

5 Decision Making

Our decision making is based on a tree of behaviors. The execution begins at a very high level behavior depending on the robot's role, such as "Attack", and goes to lower level behaviors, such as "Position to Kick", until arriving at the behaviors which will request actions to the control module. Some behaviors are modeled as finite-state machines. Our head policy switches between scanning the field for localization features and tracking the ball (when it has been seen). For navigation, we are using potential fields. Since we have been focusing on developing the low skills of our robots in the last few years, we still lack some basic mechanisms, such as positioning and marking. However, we expect to transfer some of these techniques from our Soccer 2D and Soccer 3D code bases. Our robots lack cooperation for now, but we are working on implementing a simple centralized cooperative decision making. We also are working in refactoring our behaviors into a formal open-source framework we developed [19], in order to maximize the code modularity, organization, and efficiency. Fig. 3 shows the behavior tree we are implementing with this framework.

Acknowledgment

We thank our sponsors: Altium, CENIC, Field Pro, Intel, ITAEx, MathWorks, Micropress, Neofield, Polimold, Rapid, Siatt, Solidworks, ST Microelectronics,

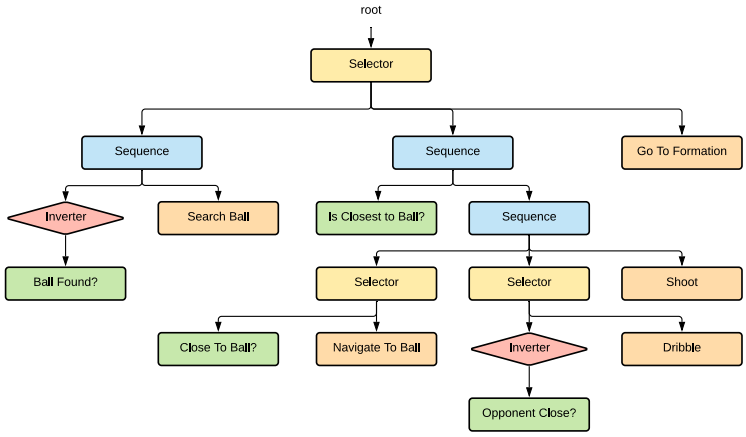

Fig. 3. Planned agent architecture for Chape.

Virtual Pyxis and Visiona. We also acknowledge Mathworks (MATLAB) and JetBrains (CLion) for providing access to high quality software through academic licenses.

References

1. *BOOST C++ Libraries*. `http://www.boost.org`.
2. Marc Bestmann, Timon Engelke, Niklas Fiedler, Jasper Güldenstein, Jan Gutsche, Jonas Hagge, and Florian Vahl. Torso-21 dataset: Typical objects in robocup soccer 2021. 06 2021.
3. G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.
4. Dr. Philippe Capdepuy. DARwIn-OP Gazebo Simulation Model, 2016.
5. RoboCup Technical Commitee. RoboCup Soccer Humanoid League Laws of the Game 2018/2019, 2019.
6. Caroline C. D. da Silva, Daniela V. de Faria, Davi H. V. Barroso, Marcos R. O. A. Maximo, and Luiz C. S. Góes. Three-Dimensional Identification of a Humanoid Robot. In *Proc. of the 2019 ABCM Int. Congress of Mech. Eng. (COBEM)*, October 2019.
7. H. Farazi, P. Allgeuer, and S. Behnke. A Monocular Vision System for Playing Soccer in Low Color Information Environments. In *Proceedings of the 10th Workshop on Humanoid Soccer Robots in conjunction with the 2015 IEEE-RAS International Conference on Humanoid Robots*, 2015.
8. Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.
9. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
10. The Qt Company Inc. Qt 5, 2017.
11. Marcos R. O. A. Maximo. Omnidirectional ZMP-Based Walking for a Humanoid Robot. Master's thesis, Aeronautics Institute of Technology, 2015.
12. Marcos R. O. A. Maximo. *Automatic Walking Step Duration through Model Predictive Control*. PhD thesis, Aeronautics Institute of Technology, 2017.

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. Otávio Ribas and Marcos Máximo. Comparison of encoder-decoder networks for soccer field segmentation. *Latin American Robotics Symposium (LARS)*, 2023.
17. T. Ŕofer. BHuman Code Release.
18. W. Burgard S. Thrun and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.
19. Gustavo L. Silva, Marcos R. O. A. Maximo, and Lourenço A. Pereira. A Minimalist Open Source Behavior Tree Framework in C++. In *2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE)*, pages 306–311, 2021.
20. Lucas Steuernagel, Marcos R. O. A. Maximo, Lucas A. Pereira, and Carlos A. A. Sanches. Convolutional neural network with inception-like module for ball and goalpost detection in a small humanoid soccer robot. In *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, pages 1–6, 2020.