# FBOT BUTIÁBOTS LARC/CBR 2022 RoboCup@Home Team Description Paper

Igor P. Maurell[1] Jardel D. S. Dyonisio[1] João F. S. S. Lemos[1] Pedro L. Corçaque[1] Cristhian L. Froes[1] Marina Z. Rocha[1] Alisson H. Kolling[2] Emilly G. Lamotte[1] Lucas S. Rambo[2] Cedenir B. da Costa[1] Luís F. M. Quadros[1] Victor A. Kich[2] Rodrigo S. Guerra[1] Paulo L. J. Drews Jr.[1]

***Abstract*— This team description paper (TDP) describes the Domestic Robot Intelligent System (DoRIS), created by team FBOT ButiáBots of the Federal University of Rio Grande and Federal University of Santa Maria, Brazil. Giving a special attention to the mostly self-built parts and all the packages developed by our team, such as: people tracking, image to kinect, behavior, world, simulation, quiz, etc. All the developed modules are open source available at: `https://github.com/butia-bots`.**

## I. INTRODUCTION

Domestic robots have been an ambition of engineers for decades. Furthermore, the growing challenges and pressures of modern life lead to a future where the use of robots in a domestic environment will be common. Thanks to current advances in technology, this aspiration is finally viable.

Pursuing that goal, in 2018 a partnership was established between Taura Bots from Federal University of Santa Maria (UFSM) and FBOT from Federal University of Rio Grande (FURG), started the Brazilian United Team for Intelligent Automation - BUTIA. From this partnership was born DoRIS, a domestic robot consisting of a high quality mobile platform, a torso (equipped with CPU and GPU units), a charismatic animatronic face and a sophisticated manipulator.

With DoRIS, the Butia Bots team already has several participations in RoboCup@Home category, being the third best team in CBR/LARC for 3 consecutive years (2018-2019-2020), the current vice champion of the CBR/LARC 2021 and the third best team in RoboCup Thailand 2022.

To introduce our robot, this document is organized as follows: the Sec. III presents a overall description of DoRIS robot architecture, the Sec. IV presents the contributions of our team, the Sec. V presents the conclusion and some future works and a overview finishes the paper on the last page.

## II. TEAM BACKGROUND

In this section, we describe the past achievements of the teams from FURG and UFSM in past robotic events and briefly describe their research interests.

### A. UFSM - Taura Bots

The Taura Bots team was established in 2014 at UFSM, Rio Grande do Sul, Brazil, with the intention of participating in national and international robot soccer tournaments, focusing on humanoid robotics. In 2015 and 2016 they participated in Latin American Robotics Competition (LARC) and also in RoboCup as a joint team with the german team WF Wolves from Ostfalia Univ. of Applied Sciences, placing third both years. During RoboCup 2015, in the humanoid league, they were awarded second place in a technical challenge. Also, recently, they took part in the FIRA AUTCup 2018 in Iran, where they placed first in the archery challenges and third place overall in the Kid Size HuroCup League. Later in 2018 they placed first in the FIRA RoboWorld Cup in Taiwan, in archery, and third in the first Robocar Race in São Paulo - Brazil. Since 2017, the team has been delving more into domestic robotics, which culminated in this partnership with FURGBOT.

### B. FURG - FURGBOT

The FURGBOT team was established in 2002 at FURG, Rio Grande do Sul, Brazil, to compete in the Small Size League. During their history, they competed in the leagues SSL, Mixed Reality League, IEEE SEK and 2D Simulation categories on RoboCup getting six first place awards and more than that in second and third place. Nowadays, the team is part of Intelligent Robotics and Automation Group (NAUTEC) and use NAUTEC's laboratories for development and research. The group has a long experience in developing service and mobile robotic systems for underwater applications. Recently, the team has been expanding their scope towards domestic service and mobile robotics, resulting in this cooperation with Taura Bots.

## III. TECHNICAL ASPECTS

In this section we describe the technical details of our proposed robot architecture, starting from the mechanical structure all the way to the code developed to interact with the world and humans.

### A. Robot Structure

Here, we present the architecture of the DoRIS robot, showing the hardware that constitutes the robot and describing the software used.

### B. Hardware

DoRIS is built in 5 major parts, as shown in Fig. 1, that are: mobile base, torso, manipulator, sensors and head.

[2]Universidade Federal de Santa Maria (UFSM), Santa Maria, RS, Brazil.
[1]Universidade Federal do Rio Grande (FURG), Rio Grande, RS, Brazil.

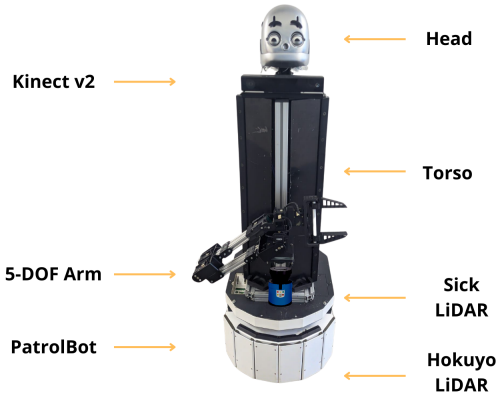



Fig. 1: DoRIS in 2022

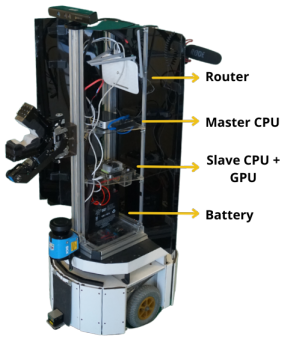



Fig. 2: Internal components

Regarding to the navigation, it is used a customized third generation PatrolBot, which is a differential, programmable and autonomous general purpose mobile robot. For the safety of those around and for the robot's robustness, DoRIS has a torso of triangular cross-section measuring around 80 centimeters with shelves to hold it all together.

### C. Processing units

There are two separate processing units inside the torso, one of them being an Intel NUC, used for ROS master and nodes that not require CUDA faster parallel computing, and a NVIDIA Jetson TX2, dedicated for GPU specific tasks, as the inference of deep neural networks. To connect these two cores we use a 2.4GHZ/5GHZ router that allows teleoperation and visualization.

### D. Sensors

Two LiDAR sensors are employed on the robot. One, on top of the mobile base, used for mapping, due its range, resolution and accuracy, and the other closer to the ground, specifically entrusted to detect small obstacles during navigation. In addition to other sensors presented in the mobile base, the encoders are the most important ones, since it are the main source of data to compute the robot odometry. The others sensors are a RGB-D camera (Microsoft Kinect V2) and the directional microphone (Rode VideoMic Pro).

### E. Manipulator

The manipulator of DoRIS is a custom made 5-DOF arm, which is assembled with 7 Dynamixel MX-106R and 2 Dynamixel MX-64R dividing the arm into six main joints, that are: waist, shoulder, elbow, wrist pitch, wrist roll and gripper. Shoulder and elbow joints take advantage of two motors connected in parallel with synchronous operation, for higher torque and working payload during the manipulation process. All of that attached by steel, aluminium and plastic 3D printed parts for the shoulder-elbow link, elbow-wrist link and gripper frame. Beyond that, the gripper fingers are switchable 3D printed parts with options for many different scenarios, such as fingers made of flexible material, using a fin gripper inspired design to provide adaptability to object shapes, or even smaller and rigid fingers with a longest length contact area, for tighter manipulation maneuver space.

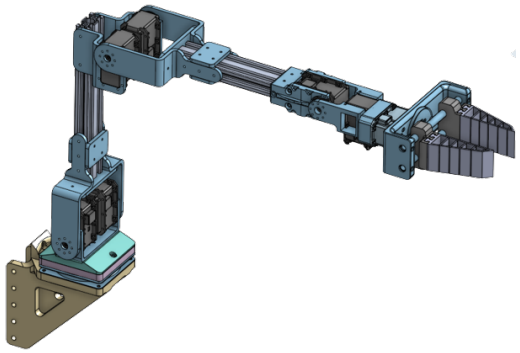


Fig. 3: DoRIS arm project.

### F. Head

We employ a electro-mechanical 3D printed face, Figure 4, manufactured in 3D printed plastic, it counts with 10 hobby servo-motors and 2 Dynamixel MX-64, controlled by a micro-controller Arduino UNO. The DoRIS's face is controlled by the main CPU, which process the ROS nodes referent to the servo-motors and send them through the Arduino UNO. Through an interface it is controlled and monitored the face expressions, currently they are: happy, sad, neutral, angry and scared. Beside these, there is a thread for the constantly blinking movement. In following improvements it is intended to associate the emotions to DoRIS's progress in ongoing tasks, for example, if successful, makes happy face, or if something is uncomprehended change to sad expression.

### G. Software Architecture

The Robot Operating System (ROS) is used as middleware to connect all the packages and to enable them to work together. The core of our system is the Behavior module, that uses ROS Smach [1] to execute each task as a robust state machine. In its construction, many simpler states and useful

Fig. 4: DoRIS's face

machines are implemented to be shared across multiple tasks, which guarantees modularity, robustness and less rework. The data used to perform the tasks can be a current measure, coming directly from the topics or services, or memorized, coming from the World module.

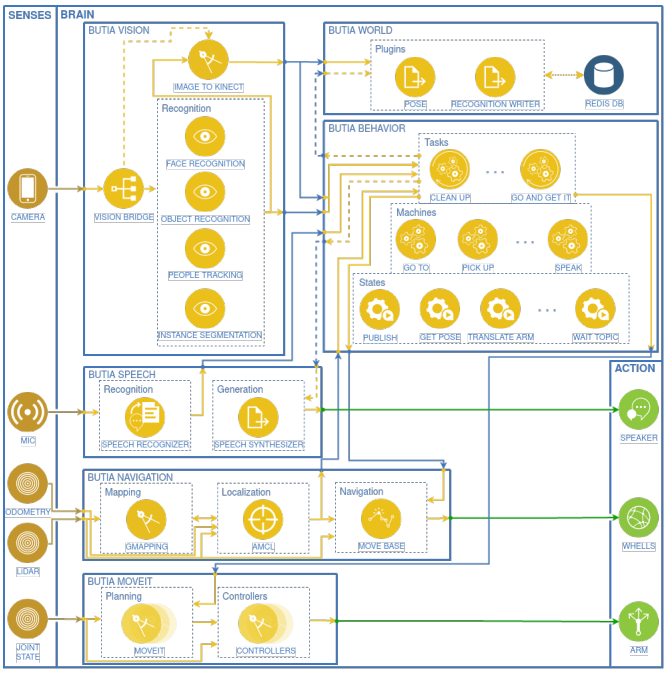



Fig. 5: DoRIS Software Architecture. The system is divided into 3 main parts, that are: senses, brain and action. In the senses, the data is acquired by the robot that perceives the world in many domains. In the brain, the data is processed to be transformed in knowledge about the world, which can be directly used for decision making during tasks execution or can me memorized for future use. The actions are the way for the robot to interact with the world, allowing it to move by the house, talk with people, and manipulate objects.

To do a complete explanation about our software architecture, the other modules are more deeply explained next:

### H. World

The World module has plugins to connect a key-value Redis Database [14] with services, topics or parameters from ROS. Each plugin has its own functions, not only to write and read data from database but to filter and transform it, if needed. Furthermore, the plugins are able to open services or topics to offer to the Behavior module access to memorized data. In short, the database is the robot's long term memory and the plugins are the filtering interface to connect the memory with the rest of the system. At this moment, the only information that the World module deal with is 3D pose, both those statically set and the ones perceived by the Vision module.

### I. Vision

The vision system of our robot is composed of several packages for each of the following tasks: object recognition, people tracking, face recognition and segmentation. We use YOLOV4 [2] for the object recognition. For people tracking, we use DeepSORT [8] to track people detected by the object recognition package. For face recognition we use the pipeline proposed by OpenFace [4], modifying the detection and classification steps. For segmentation, Mask-RCNN [6] is used to segment the image in instances of multiple classes. There are also two other packages, Vision Bridge and Image to Kinect. The first, Vision Bridge, is used to synchronize in time the input RGB, depth, camera information and point clouds from RGB-D camera and pass then along with the same ID. The second, Image to Kinect, is used to estimate the three-dimensional translation of the recognitions. If there is access to a reference point cloud of the recognition, the package also estimates the three-dimensional orientation using RANSAC [10] and FPFH [9] features from PCL [13].

### J. Speech

To perform speech recognition we use the SpeechRecognition library[1] available for Python language. Currently, the speech recognizer node is used for several tasks, such as: follow me, quiz, take orders, among others. The main function of the library in the software is to perform speech-to-text (STT), which transcribes the audio to text, thereon, we are able to use it in many applications in our system. For speech generation, we use the gTTS library (Google Text-to-Speech) and the mixer module from the Pygame library. With gTTS we convert a phrase (text) to audio, and with the mixer module we play back the previously generated audio. This allows us to have a complete conversion between speech-to-text and text-to-speech. We also have experimental support for offline neural speech recognition and speech generation using the methods available in the espnet2[16] library. To speak, Text-To-Speech generates an audio that will be played using the audioop, wave and pyaudio libraries. First, the audio is opened by the wave library, and is discretized into chunks. After that, the audio is played from the pyaudio library. At the same time, while the audio is playing, the audio data is sent to the face to perform the jaw-voice synchronized movement. As the audio has been discretized, the jaw movement is smooth.

### K. Quiz

One implementation that improves human-robot interaction is the ability to answer questions. Currently, we receive

[1]Github Repository: https://github.com/Uberi/speech_recognition

a question by audio, convert it to text, and a NLP algorithm is used to select a filtered sentence and look after the best way to search the question. After this filter, we took the question and use three methods to get the answer. First, we search on Google and, if the answer is not found, we search using the WolframAlpha API [15]. If neither of them found an answer, we search a pre-set question database to answer the question. This implementation is still under development to improve the way that the information is returned, looking forward the best possible interaction. Due to the current development, the method is not yet fully integrated with the software architecture.

### L. Navigation

To be able to navigate and localize itself, our robot employs a Simultaneous Localization and Mapping (SLAM) system, which builds a map of the environment where the robot is. Our SLAM system is based on a grid representation, along with Rao-Blackwellized Particle Filter (RBPF) [12]. This method comes along with the ROS package *gmapping*[2]. Our localization is done by using a Monte Carlo Localization (MCL) method with adaptive sampling of particles, provided by the *amcl* package[3]. With a map and knowing its location, the robot is enabled to navigate throughout the environment. The navigation system works based on cost maps, dividing the environment into a grid and giving each cell a value of occupancy. A global cost map is build upon the environment map provided by the SLAM and a local cost map is generated with the information from the global cost map and from the sensors. Planners are employed to choose the best path through the cost map, with the global planner being a Dijkstra algorithm and the local planner a classic trajectory planning algorithm together with a Dynamic Window Approach (DWA) [11], both available on the ROS package *move_base*[4].

### M. Manipulation

We use MoveIt[1] as motion planning framework. This package supports several motion planning and inverse kinematics backends, such as OMPL[1] for motion planning and Orocos KDL[1] for inverse kinematics, which are the solutions employed by our team for the manipulation component of our software architecture. In order to handle the composition of several simple manipulation tasks into more complex ones, we use the ROS Smach state machine library for the sequential and parallel execution of those tasks.

## IV. CONTRIBUTIONS

### A. Offshore application

In the last few months we have submitted and started working on the autonomous and teleoperated robot for engineering activities in offshore platforms. It consists in a project, financed by PRH-22 (a Brazilian governmental initiative to incentive the oil and gas industry technology advancement), in which the purpose is to implement and adapt DoRIS features looking after an architecture able to serve its coworker. The expected results are to develop and assemble features like: recognize and read sensors, recognize co-workers and verify if they are using PPE correctly, manipulate the required instruments, map the environment to be able to do it all remotely.

### B. Robot Face

Humans communicate with their voice, gestures, body language and with face expressions. Thinking on these terms, the DoRIS' face was designed to minimize discomfort caused by the robot to humans, also improving the human-robot interaction. The face is intended to escape from the Uncanny Valley [7] concept, which is a state where an object's appearance being very similar to a human results in a negative emotional response to it. DoRIS' face has a friendly appearance to encourage humans to engage in a conversation. Besides that, it needs to react socially to the environment, processing and returning an answer by voice, gesture or facial expression.

### C. Dataset Generator

Manual data annotation is typically the most labor-intensive process in any deep learning based computer vision pipeline. Besides, some modalities of annotation cannot be annotated manually with precision in the time interval used in the competition, such as panoptic segmentation and 3-dimensional object bounding box. In order to offer a solution to this issue, we have developed a unity package to do synthetic data generator to the context of RoboCup@Home based on Unity Perception[1], that is a toolkit made around the concept of domain randomization [5] for sim to real transfer. Using our dataset generation tool, it is possible to generate at scale annotated ground truth data for tasks such as object detection and instance segmentation.

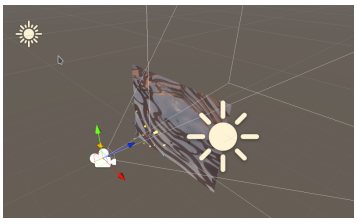
## V. CONCLUSIONS

In this paper, we present the approaches used by the FBOT ButiáBots team for the RoboCup@Home competition. We provide an overview of the robot DoRIS, presenting its hardware composition, showing the platform, sensors, processing units, manipulator and head with animatronic face. The software structure is described with the details of its implementations, demonstrating the use of a World module to provide a detailed portrait of the environment facilitating the decision making. The contributions of our works are introduced, showing how it can be applied outside of the competitions.

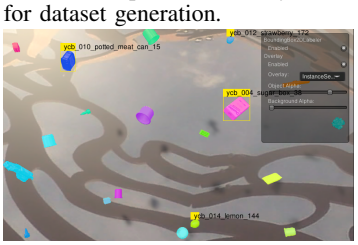
One future area of research is the integration of deep reinforcement learning algorithms, such as D4PG [3], into the manipulation pipeline, possibly with the usage of sim to real transfer techniques, such as domain randomization [5].

[2]ROS Wiki: http://wiki.ros.org/gmapping
[3]ROS Wiki: http://wiki.ros.org/amcl
[4]ROS Wiki: http://wiki.ros.org/move_base
[1]Official Website: https://moveit.ros.org/
[1]Official Website: https://ompl.kavrakilab.org/
[1]Official Website: https://www.orocos.org/kdl.html

[1]Github Repository: https://github.com/Unity-Technologies/com.unity.perception



(a) An example of the Unity scene for dataset generation.




(b) Annotation examples.

Fig. 6: Datasets Generation tool in execution.

## REFERENCES


[1] Jonathan Bohren and Steve Cousins. The smach high-level executive [ros news]. *IEEE Robotics Automation Magazine*, 17(4):18–20, 2010.

[2] Alexey Bochkovskiy et al. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[3] Barth-Maron et al. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[4] Brandon Amos et al. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

[5] Josh Tobin et al. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[6] Kaiming He et al. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[7] Mori et al. The uncanny valley [from the field]. *IEEE Robotics & Automation Magazine*, 19(2):98–100, 2012.

[8] Nicolai Wojke et al. Deep cosine metric learning for person re-identification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 748–756. IEEE, 2018.

[9] Radu Bogdan Rusu et al. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[10] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[12] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.

[13] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. IEEE.

[14] Salvatore Sanfilippo and Pieter Noordhuis. Redis, 2009.

[15] Bruce Walters. Wolfram— alpha, a new kind of science. 2011.

[16] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. ESPnet: End-to-end speech processing toolkit. In *Proceedings of Interspeech*, pages 2207–2211, 2018.

**Title:** FBOT BUTIÁBOTS LARC/CBR 2022 RoboCup@Home Team Description Paper

**Members:** Igor P. Maurell, Jardel D. S. Dyonisio, João F. S. S. Lemos, Pedro L. Corçaque, Cristhian L. Froes, Marina Z. Rocha, Alisson H. Kolling, Emilly G. Lamotte, Lucas S. Rambo, Cedenir B. da Costa, Luís F. M. Quadros, Victor A. Kich, Rodrigo S. Guerra and Paulo L. J. Drews Jr.

## Robot Hardware Description

The robot is built based on performing household tasks. The specifications are described below:

- Base: PatrolBot (MobileRobots Inc) - Differential programmable autonomous general purpose service robot
- Torso: V shaped format, with 3 shelves in the inside, to support the hardware
- Arm: Mounted on torso. 7DOF. Maximum load: 1.5kg
- Neck: 2DOF
- Head: 10DOF, that are: jaw, eyelid, eyebrown and eyes
- Robot dimensions: height: 1.62m (max), width: 0.59m (max)
- Robot weight: 58kg

*Also our robot incorporates the following devices:*

- Hokuyo URG-04LX-UG01 (Ground LiDAR)
- SICK LMS-100 (Body LiDAR)
- Kinect Sensor
- Rode VideoMic
- Intel NUC
- Nvidia Jetson TX2

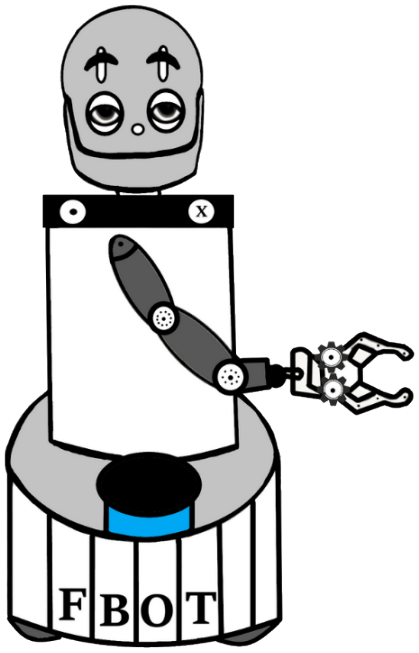



Fig. 7: DoRIS Robot

## Robot's Software Description

*For our robot we are using the following software:*

- Platform: ROS - Robot Operating System
- Navigation: MoveBase
- Localization: AMCL
- Mapping: Gmapping
- Face recognition: OpenFace + OpenCV + Python's face-recognition library
- People Tracking: DeepSORT
- Object Recognition: YOLOV4
- Speech Recognition: SpeechRecognition library for Python
- Arm Control: MoveIt
- Knowledge Storage: Redis
- Task Executor: SMACH