ButiaBots Team Description Paper RoboCup@Home 2022

Paulo L. J. Drews Jr Rodrigo S. Guerra Alisson H. Kolling
Cristhian L. Froes Emilly G. Lamotte Igor P. Maurell
Jardel D. S. Dyonisio João F. S. S. Lemos Marina Z. Rocha
Pedro L. Corçaque Victor A. Kich

January 23, 2023

Abstract. This team description paper (TDP) describes the Domestic Robot Intelligent System (DoRIS), created by team BUTIA 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`.

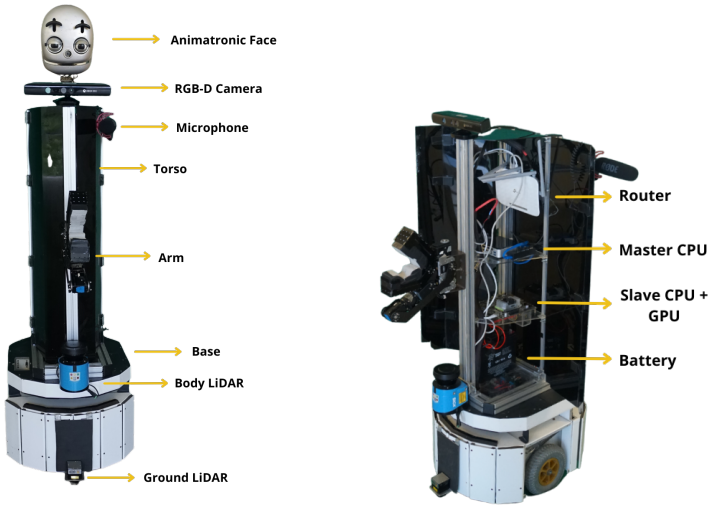
1 Introduction

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) and the current vice champion of the CBR/LARC 2021.

To introduce our robot, this document is organized as follows: the Sec. 2 presents a overall description of DoRIS robot architecture, the Sec. 3 presents the contributions of our team, the Sec. 4 presents the conclusion and some future works and a overview finishes the paper on the last page.



(a) External components (b) Internal components

Fig. 1: DoRIS architecture

2 DoRIS Robot Architecture

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

2.1 Hardware

DoRIS is constructed in 6 major parts, presented in Fig. 1: base, processing units, torso, manipulator, sensors and head, each part which goes along has it's function. Regarding navigation, is used a customized third generation PatrolBot, which is a differential, programmable and autonomous general purpose service robot. And, for robustness and safety of those around, the robot has a triangular shaped torso with around 1 meter height with shelves to hold it all together.

Processing units The processing units are two separated cores, one of them being an Intel NUC, used for general purpose and provided with an Intel Core i5 4250U and 8GB RAM, and a NVIDIA Jetson TX2, dedicated for computing GPU specific tasks. To connect these two cores we use a 2.4GHZ/5GHZ router.

Sensors Two navigational sensors are employed on the robot. One, on top of the mobile base, used for mapping, due its range and resolution, and the other closer to the ground, specifically entrusted to detect small obstacles during navigation. The others sensors are a RGB-D camera (Microsoft Kinect V2) and the directional microphone (Rode VideoMic Pro).

Manipulator For the manipulation, it is used a self-built 7DOF arm, shown in Fig. 2, which takes advantage of the physical capabilities of 3 Dynamixel MX-106T for the shoulder joint and 5 Dynamixel MX-64T for the elbow, wrist and gripper functionalities. All of that attached by 3D printed ABS polymer pieces for the shoulder-elbow link, elbow-wrist and gripper frame.

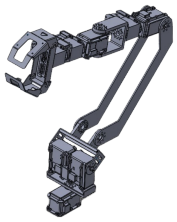


Fig. 2: DoRIS arm project.

Head We employ a electro-mechanical 3D printed face, manufactured in ABS, it counts with 12 hobby servo-motors controlled by a micro-controller Arduino UNO. The robot uses a IMX477 HQ modular camera as visual input inside its face. The DoRIS' face is controlled by a NVIDIA Jetson Nano B01, which process the ROS nodes referent to the servo-motors and send them through the Arduino UNO. The modular camera inside the face is localized in its nose and is independent of the rest of the system. Because of that, the robot face can look at any spot while making a task without mixing up the image-based tasks.

2.2 Software

The diagram presented in Fig. 3 presents a simplified version of DoRIS system. The Robot Operating System(ROS) is used as middleware to connect all the packages and to able then 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 machines are implemented to be shared across the machines of 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.

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

World The World module has plugins to connect a key-value Redis Database [2] 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

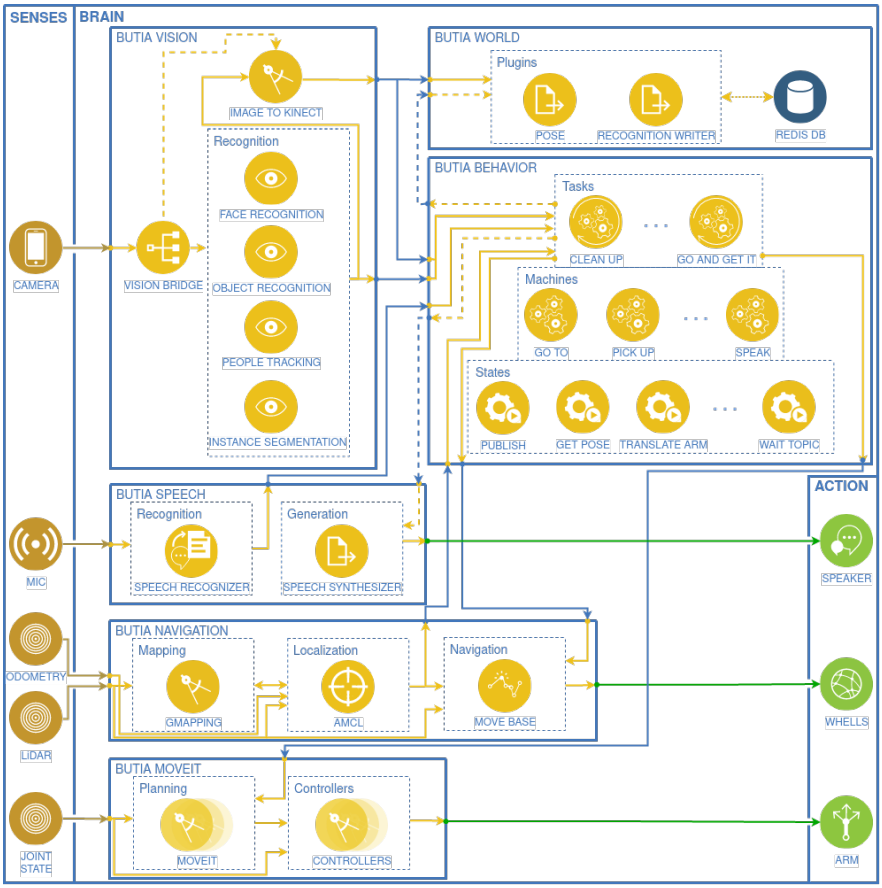


Fig. 3: Diagram of DoRIS software architecture. Communications done by pointed lines are requests and by normal lines are message passing.

that the World module deal with is 3D pose, both those statically set and the ones perceived by the Vision module.

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 [3] for the object recognition. For people tracking, we use DeepSORT [4] to track people detected by the object recognition package. For face recognition we use the pipeline proposed by OpenFace [5], 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 [7] and FPFH [8] features from PCL [9].

Speech To perform speech recognition we use the SpeechRecognition library[1] available for Python language. Currently, the node speech recognizer 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 software. 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.

Quiz One implementation that improves human-robot interaction is the ability to answer questions. Currently, we receive 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 [10]. 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.

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) [11]. 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

[1] Githut Repository: https://github.com/Uberi/speech_recognition
[2] ROS Wiki: http://wiki.ros.org/gmapping
[3] ROS Wiki: http://wiki.ros.org/amcl

planner being a Dijkstra algorithm and the local planner a classic trajectory planning algorithm together with a Dynamic Window Approach (DWA) [12], both available on the ROS package *move_base*[4].

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 complex atomic manipulation tasks into simpler ones, we use the ROS Smach state machine library for the sequential and parallel execution of those atomic tasks.

3 Contributions

3.1 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 it's coworker. The expected results are to develop and assemble features like: recognize and read sensors, recognize co-workers and verify if they are doing the correct usage of the EPI, manipulate the required instruments, map the environment to be able to do it all remotely.

3.2 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 generated by the robot to humans, also improving the human-robot interaction. The face is intended to escape from the Uncanny Valley [13] 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 have 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 expressions. The Fig. 4 shows the some facial expressions of DoRIS.

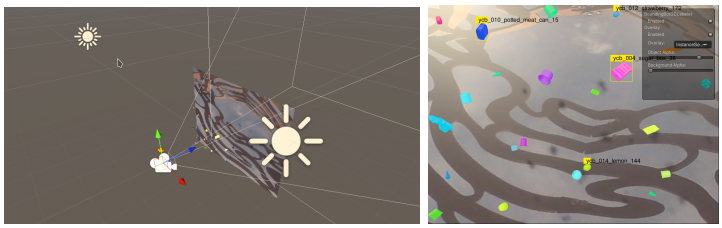
[4] ROS Wiki: http://wiki.ros.org/move_base
[1] Official Website: https://moveit.ros.org/
[1] Official Website: https://ompl.kavrakilab.org/
[1] Official Website: https://www.orocos.org/kdl.html

(a) Idle. (b) Thinking. (c) Anger. (d) Sad.

Fig. 4: DoRIS' expression faces.

3.3 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 [14] 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. An example of the execution of the tool is showed in Fig. 5.



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

Fig. 5: Datasets Generation tool in execution.

4 Conclusions and future work

In this paper, we present the approaches used by the BUTIA 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

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

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 [15], into the manipulation pipeline, possibly with the usage of sim to real transfer techniques, such as domain randomization [14].

References

1. Jonathan Bohren and Steve Cousins. The smach high-level executive [ros news]. *IEEE Robotics Automation Magazine*, 17(4):18–20, 2010.
2. Salvatore Sanfilippo and Pieter Noordhuis. Redis, 2009.
3. Alexey Bochkovskiy et al. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
4. 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.
5. 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.
6. Kaiming He et al. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
7. 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.
8. 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.
9. 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.
10. Bruce Walters. Wolfram— alpha, a new kind of science. 2011.
11. Giorgio Grisetti et al. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.
12. Dieter Fox et al. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.
13. Mori et al. The uncanny valley [from the field]. *IEEE Robotics & Automation Magazine*, 19(2):98–100, 2012.
14. 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.
15. Barth-Maron et al. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

Title: ButiaBots Team Description Paper RoboCup@Home 2022
Members: Paulo L. J. Drews Jr, Rodrigo S. Guerra, Alisson H. Kolling, Cristhian L. Froes, Emilly G. Lamotte, Igor P. Maurell, Jardel D. S. Dyonisio, João F. S. S. Lemos, Marina Z. Rocha, Pedro L. Corçaque, Victor A. Kich

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

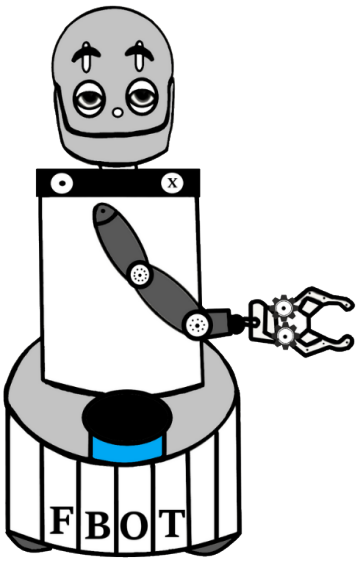


Fig. 6: DoRIS Robot

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