BUTIA LARC/CBR 2019 RoboCup@Home Team Description Paper

Bruno S. Castro[1] Cedenir B. da Costa[2] Cleber D. Werlang[1] David R. Richards[1] Douglas S. Brum[1] Eduardo R. Santana[1] Gabriel N. Niederauer[1] Gustavo H. Nascimento[2] Igor P. Maurell[2] Julio P. C. R. Jardim[1] Khaled H. Guimarães[2] Junior C. Jesus[1] Lucas S. Avila[2] Luciano A. Cardona Junior[1] Luiz F. T. França[2] Paulo L. J. Drews Jr.[2] Renan T. Fonseca[2] Ricardo B. Grando[2] Rodrigo S. Guerra[1] Victor A. Kich[1]

Abstract— **In 2018, a partnership was formed between Universidade Federal de Santa Maria (UFSM) and Universidade Federal do Rio Grande (FURG), with the intention of competing in the LARC/CBR RoboCup@Home league. The team is called *Brazilian United Team for Intelligent Automation - BUTIA*. This paper describes the hardware, software and mechanical aspects of the robot *DoRIS*, and how they aim to solve the tasks and challenges of the domestic environment.**

I. INTRODUCTION

Domestic robots have been a sci-fi ambition for decades. Now, the increasingly aging population intensifies the strain on the infrastructure of current health systems. This, combined with the escalating challenges and pressures of modern life push for a future where the use of robots in a domestic environment is commonplace. Current advances in technology finally make this a feasible aspiration. Pursuing that goal, we present the *Domestic Robotic Intelligent System - DoRIS*, a domestic robot born from a partnership between UFSM and FURG, designed by the recently founded *Brazilian United Team for Intelligent Automation - BUTIA*.

DoRIS is 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.

The remainder of this work is organized as follows: Section II presents the background of the team. Section III describes the technical aspects of the proposed robot architecture, detailing the robot structure (Subsection III-A), the hardware (Subsection III-B), the arm (Subsection III-C), the face (Subsection III-D), the software architecture (Subsection III-E), the slam (Subsection III-F), the localization (Subsection III-G), the navigation (Subsection III-H), the vision system (Subsection III-E) and the speech recognition (Subsection III-J). Finally, Section IV sums up the paper and discusses future directions.

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

Our robot has a mobile base, which is a customized third generation PatrolBot, originally manufactured by Mobile Robots. PatrolBot is a 2-wheel differential-drive, indoor mobile robot, designed and sized to carry payloads of up to 40 kg. The robot's size and drive assembly are designed to work in any wheelchair-accessible environment. Besides that, *DoRIS* has a torso above the mobile base to lay up the hardware and place the input instruments in positions

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

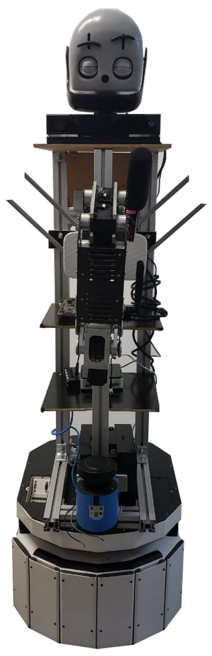

Fig. 1. DoRIS in 2019.

TABLE I
MECHANICAL SPECIFICATIONS OF DORIS

Physical Specifications	
Height	162 cm
Weight	58 kg
Reach (Arm)	80 cm
DOFs	16
Frame Material	Aluminum
Number of servomotors	
MX-106	9
MX-28	5

that facilitate the observation. Our torso is built using structural 40mm x 40mm aluminum profiles, to facilitate the customization, assembly and disassembly of the robot. This simple but generic structure, allow us to put many shelves inside the body of *DoRIS*, adding specific layers for each hardware component, increasing the robot organization.

Table I displays information about the physical specifications of *DoRIS*.

B. Hardware

One Sick LMS-100 (Figure 3) and one Hokuyo URG-04LX-UG01 (Figure 2) are used in our system, the first one on top of the mobile base and the other closer to the ground, specifically to detect small obstacles. These LIDAR systems are adopted for the tasks of mapping and obstacle avoidance.

A Kinect v2 (Figure 4) is mounted on the shoulders of the robot and serves as the major vision component for *DoRIS*. It captures a so-called RGBD image, which means it captures both a red-green-blue color image, used as input of computer vision algorithms, and a depth image, which allows us to measure the distance from the robot to all results obtained in the executed algorithms.

Fig. 2. Hokuyo URG-04LX-UG01 adopted in our robot.

Fig. 3. SICK LMS-100 adopted in our robot.

In order to be able to process both CPU and GPU intensive tasks we are using two computers. One is an Intel NUC, which is used as a general purpose computer, running CPU intensive tasks and main processes, such as ROS core. The other computer is an NVIDIA Jetson TX2, dedicated for running GPU specific tasks, such as the deployment of deep learning models. The main specifications of both computers are presented in Table II. Communication between both computers is handled through ROS over a Ethernet cable connected to a Dual Band router allowing 5GHz Wi-Fi connections with computers out of the robot.

As the tasks of the competition require Speech Recognition capability, we use a Rode VideoMic (Figure 5) directional microphone to improve the quality of the audio data used in the algorithms.

C. Gripper Arm

DoRIS' arm design was adapted from one of the legs of the humanoid robot Dimitri. The structure is composed of a carbon fiber link structure and several aluminum fixture parts.

In order to transform Dimitri's leg into DoRIS' arm, a

TABLE II
CPU AND GPU SPECIFICATIONS

Intel NUC Specifications	
CPU	Intel Core i5 4250U
Memory	DDR3L 1333MHz 8GB
Storage	SSD mSATA 120GB
GPU	HD Graphics 5000
Nvidia Jetson TX2	
CPU1	Denver2 ARMv8 64bit
CPU2	Quad-core ARM Cortex-A57
Memory	8GB 128-bit LPDDR4
Storage	32GB eMMC
GPU	Pascal-Based 256 CUDA Cores

Fig. 4. Kinect sensor adopted in our robot to obtain RGBD images.

Fig. 5. Microphone used in the robot.

few modifications were made. The leg's foot was replaced with a wrist and a gripper built with repurposed parts of a decomissioned DARwIn-OP1 robot, resulting into a 5-DOF system capable of reproducing complex movements. A shoulder was designed to fix the arm in one of the aluminum structures in a way that it's height can be manually adjusted using hammer head bolts and nuts. The shoulder supports two MX-106 Dynamixel Servo-actuators, doubling the force to elevate the arm and hold it in position, with it carrying a payload or not.

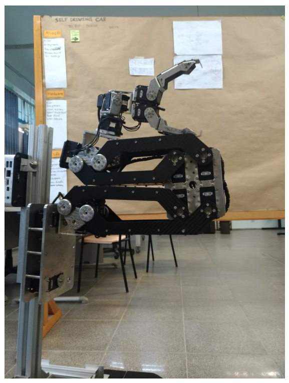

Fig. 6. DoRIS' arm first version

The final configuration of the arm has a total of 12 joints. Since the weight to be lifted by the arm's shoulder pitch joint is too big for just one actuator, we decided to put two in parallel. The figure 6, shows an earlier version of DoRIS' arm without the parallel shoulder joint, parallel actuators were put in the arm's elbow too. Having a total a 6 actives DoFs and 3 passives DoFs in the elbow, wrist and shoulder, DoRIS will be able to handle many tasks in the domestic environment with robustness, precision and efficiency. Although not totally complete for tests in the real world, we tested DoRIS' arm kinematics in our simulation environment (Figure 7) in Gazebo where we are performing manipulation tasks.

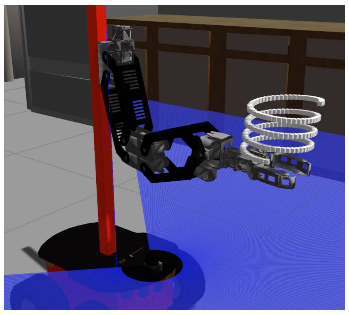

Fig. 7. Testing DoRIS' arm kinematics in our simulation environment in Gazebo

D. Face

The DoRIS' face was designed to minimize discomfort generated by the robot to humans, also improving the human-robot interaction. We designed it looking to escape from the *Uncanny valley* concept, which is a state where an object's appearance being very similar to a human results in a negative emotional response to it.

The face parts were manufactured in a 3D printer using a material that needed to be strong and resistant, but at the same time with a good cost-benefit. Considering that, we chose ABS, which fits under these conditions. The robot's face contains 12 hobby servo-motors, all connected to a microcontroller powered 5V DC. These servos take care of the movements of eyes, eyelids, eyebrows and mouth. The microcontroller is responsible for receiving and interpreting data sent from the NUC which is wired to a RS485 conversor connected to all of the Dynamixel servo-actuators. The packet is decoded and then sent to the micro servos.

DoRIS' face must 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. Figure 8 shows a 3D rendering of the designed face, and Table III show its specifications.

E. Software Architecture

Our software architecture is based on the Robot Operating System (ROS)[1]. ROS is an open-source, meta-operating system for robots. This system provides a high-level abstraction

[1]http://www.ros.org/

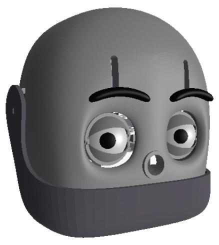

Fig. 8. DoRIS' face 3D model

TABLE III

SPECIFICATIONS OF THE ANIMATRONIC FACE

Face components	
Micro Servo SG90	10
Micro Servo SG92R	2
LMS8UU Linear Bearing	2
Aluminum Bar - 8 mm x 70 mm	2
Arduino Uno	1
LM2596 DC 5v	1
Conversor TTL - RS485	1
Arduino Sensor Shield v5.0	1
Physical Specifications	
Height	20 cm
Width	18 cm
Depth	17 cm
Weight	0.9 kg
Material	ABS

similar to an operating system. The system includes hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management.

A database is used to store information about entities that compose the environment. This database holding the world model is the lead and main part of a knowledge base. The first layer stores the information about unitary entities and objects, information such as localization, shape, pose, etc. Classes of entities are also stored in that layer, so that they can be referenced later.

After that, entities and objects are related in the second layer, storing references between the entities. These references represent directed relations necessary for an easier understanding about the composition of the world, reducing query time by storing information such as that the entity shelf can hold the entity book and a entity table can hold any entity of the type object. The type of object described earlier can be referenced by objects instances in this layer for example with a *HasTypeDefinition* reference.

The database will rely on a nodal structure and this is achieved by manipulating key-value storage. Inside the database, new nodes and references can be created, retrieved, updated and deleted (CRUD operations) according to the information gathered by the robot's sensors. Each sensor, physical or virtual, will require an interface that allows it to transform the world model based on what it is measuring and evaluating. These interfaces will be provided by means of *ROS Services* to manipulate the data stored in the database.

To complete specific tasks, these operations over the knowledge base are grouped inside states, which run in different orders given the context and goal to achieve. These states run through a state-machine engine called *SMACH*. *SMACH* allows programmers to write states in python programming language and connect them in a way that the robot follows an arbitrary path of actions.

Each state in the state machine knows which actions to over the knowledge base to alter it or to retrieve data necessary to complete a task. The knowledge base is considered the best approach due to the fact that states can run in parallel and it's easier to treat things like race conditions, and much easier to research for information in a single world model.

F. Simultaneous Localization and Mapping (SLAM)

Virtually all mapping algorithms considered to be state-of-the-art are probabilistic. Aside from using probabilistic models for the robot and the world representation, they also make use of probabilistic inferences to transform sensor readings on maps. The reason for the popularity of this type of approach lies in the fact that robotic mapping is characterized by sensor noise and uncertainty, and the probabilistic algorithms try to solve this by modeling different sources of noise and it's impact on measures. The decision on which algorithm to use also depends primarily on the computational power available, since embedded systems usually are a little limited on this aspect.

The chosen SLAM method is based on a grid representation, along with Rao-Blackwellized Particle Filter (RBPF) [8] [9]. The RBPF is a Bayesian approximation mapping method, that when combined with active learning strategies, is fast and precise [14] [4]. This approach makes use of a particle filter in which every particle carries a representation of the map and applies adaptive techniques to define the number of particles in any given moment, while also doing selective resampling of the particles. The ROS package that implements this algorithm is gmapping [7].

G. Localization

The Monte Carlo Localization (MCL) method with adaptive sampling of particles is a very efficient way of dealing with the mobile robot localization problem. The adaptability comes from the use of the Kullback–Leibler Divergence (KLD) technique [10], as it allows to calculate the estimate error and, using that information, decide on the number of samples needed. Initially, a high number of particles is required to cover the whole map uniformly, but as the particles converge on a same localization, keeping a high number of samples is a waste of computational resources, that is why the implementation of MCL with KLD sampling increases the efficiency of the algorithm by continuously adapting the number of samples.

The package which implements this solution is the amcl (*Adaptative Monte Carlo Localization*) [6], a localization system for robots with 2D navigation freedom. The choice for this package was made based on it's efficiency and flexibility.

H. Navigation

The navigation system works based on costmaps, which are generated by representing the environment in the form of a grid map. Every cell in this grid carries information about it's occupancy, which is later translated into three states: free, occupied and unknown.

The navigation can be separated into two parts for better comprehension: local and global. In the global aspect of navigation, there is a global planner, responsible for creating the global paths, and the global costmap, which is calculated from the map of the environment – the map can be the one provided by the SLAM algorithm, for example. The global planner generates a trajectory based on this global costmap, which in turn is used by the local planner to decide short term navigation plans. The local planner also uses a local costmap, which is constantly updated by the LIDAR sensor, to generate this short term paths.

Trajectory planning solutions are ideal for achieving a goal position in known static environments, while methods that avoid collision in real time allow for a reactive behaviour in dynamic unknown environments. A hybrid solution, like using a classic trajectory planning algorithm together with a Dynamic Window Approach (DWA) [5], for example, ends up being a great combination.

For the local planner we use the package dwa_local_planner [12], which implements the Dynamic Window Approach. It is specially efficient in avoiding collision when the robot is operating at high speeds, as it works directly in the velocity space and takes in consideration the robot dynamics.

For the global planner we chose to use the Dijkstra algorithm, which can find optimal path even when the costmap is not uniform [18], as it happens with our use case, where the cost value exponentially diminishes when moving away from a obstacle.

We use the package move_base [13] as an interface to configure, execute and interact with the navigation system of the robot.

A diagram illustrating how the navigation system works can be seen in Figure 9. This diagram represents the system after the map have been generated by SLAM. The map is then provided by the node map_server and used by the global costmap. The laser sensor is used to update both the global and local costmaps cell occupancy. The local costmap is created from the laser sensor readings.

The trajectory created by the global planner – using Dijkstra algorithm implemented by the navfn package – is then provided to the local planner, which tries to follow the global trajectory using the DWA algorithm. The local planner controls the robot by sending velocity controls to the mobile base through the RosAria node, responsible for actuating on the motors.

The RosAria node also publishes, among other things, information about odometry, battery level and motor states. This is all done while the localization node (amcl) keeps the robot localized by publishing the transform between the coordinate system of the base and the map. If the planners are unable to generate a viable trajectory, recovery behaviors are executed to try to find if there really is not a valid path or if there was some problem with the costmaps.

I. Vision System

The use of cameras is indispensable to perceive the environment and get informations about the objects and people. This photometric information allows the usage of many computer vision algorithms. Which algorithm to use does not import to the software in general, because as the chosen middleware is ROS, the Vision System needs

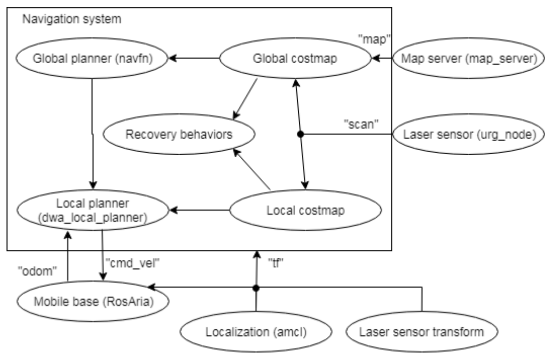

Fig. 9. Diagram of the navigation system.

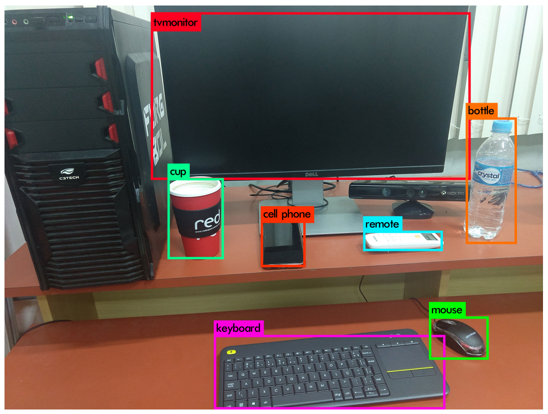

Fig. 10. Object detection using a deep learning network YOLO v3 [16].

to have packages of each algorithm type and it can have nodes to implement this in different ways. The types of computer vision algorithms developed to solve the tasks of the competition are object recognition, face recognition and people tracking. Besides that, the vision system has other five auxiliary packages to help the system to works in the correct way, totaling 8 packages, but in this paper we are going to explain only the main ones.

1) Object Recognition: The object recognition is obtained using the deep learning detector model YOLO v3 [16]. The method detects objects, calculates a bounding box and provides a description label for each object in the image. Furthermore, the 2D position of this box in the image plane is transformed into a 3D position in the world using the depth information of the Kinect. We adopted the package darknet_ros package [3] for real-time object detection. Due to the fact that objects of the competition arena can be much more diverse than the pre-trained ones, a fine-tuning process is performed to expand the detection capacity of the model. The Figure 10 shows an example of object detection. Besides that, this package do an additional, which is classify the detection classes in generic groups that represent two or more classes of the model.

2) Face Recognition: The face recognition is based on the state-of-art approach FaceNet [17]. We used the Openface [2] framework, which provides an execution in pipeline to use the FaceNet. The face recognition process is done in four steps: detection, alignment, embedding and classification.

- Detection: It is the process that find faces in the image. The package has four models that can be used for detection: Haar Cascade OpenCv, SSD OpenCv, HoG dlib and MMOD dlib;
- Alignment: It is the process that adapts the face in a correct format to be processed in the next step. An affine transformation is done to align the face.
- Embedding: The embedding is done by processing the FaceNet with a pre-trained model in a 128 dimension vector.
- Classification: The last step is to classify the face based on faces trained previously. The classifiers are available at the scikit-learn [15] library. Few examples are: linear svm, radial svm and knn.

The package makes the recognition in real time over the images provided by Kinect and publishes the recognitions. Besides, the package also provides two services. The first one makes a training request of a new classifier based on the actual dataset and the second one presents a new person to the robot, increasing the dataset with some images of the person and training a new classifier to recognize the person in real-time.

3) People Tracking: The people detection is also done using YOLO v3 [16]. The object recognizer detects the person, which is compered with the pre-visualized person by the tracker. This comparison is done based on the matching of the SIFT features [11]. At each new correct detection, the tracker adds new features to a map of features, allowing the tracking to continue even if the person change the position. To avoid using the background features in the process, a segmentation request is done by Segmentation before of the SIFT. The tracking is used in many services on the robot. As an example, the Follow Me task, where the robot must follow an operator even if other people get in the field vision.

J. Speech Recognition

The speech recognition of the Doris Robot is carried out in two stages. The first is through the conversion of voice into text using the Google Cloud Speech API for having demonstrated superior performance to other libraries already used. In the second step the natural language processing is done where we will do the data processing using the NLTK library to convert the sentence into actions that the robot must perform. In addition, we use Porcupine [1] Hotword detection with a personalized hotword to start our speech recognition system.

IV. CONCLUSIONS

In this paper we have presented our team BUTIA, as a joint effort between FURG and UFSM. Also, we have described our project to solve the tasks of Robocup@Home category, resulting in the development of the robot *DoRIS*. The project is under development for LARC/CBR 2019, but in this paper we fully described the current state of all the technical aspects of the robot, from the hardware to the software systems, which we believe are of fundamental importance to the domestic environment of the league. We also presented the animatronic 3D-printed face with 12 DoF, allowing the robot a large range of facial expressions. We believe this feature is an important step towards developing more natural human-robot interactions.

REFERENCES

[1] Porcupine hotword detection. `https://github.com/Picovoice/Porcupine`. Accessed: 2019-06-17.

[2] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

[3] Marko Bjelonic. Darknet ros package. `http://wiki.ros.org/darknet_ros`. [Online; accessed 6-July-2018].

[4] Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 176–183. Morgan Kaufmann Publishers Inc., 2000.

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

[6] Brian P Gerkey. Documentação oficial do pacote amcl no ros wiki. `http://wiki.ros.org/amcl`, 2018. Visited on 29/11/2018.

[7] Brian P Gerkey. Documentação oficial do pacote gmapping no ros wiki. `http://wiki.ros.org/gmapping`, 2018. Visited on 29/11/2018.

[8] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2432–2437. IEEE, 2005.

[9] 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.

[10] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[12] Eitan Marder-Eppstein. Documentação oficial do pacote dwa_local_planner no ros wiki. `http://wiki.ros.org/dwa_local_planner`, 2018. Visited on 29/11/2018.

[13] Eitan Marder-Eppstein. Documentação oficial do pacote move_base no ros wiki. `http://wiki.ros.org/move_base`, 2018. Visited on 29/11/2018.

[14] Kevin P Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems*, pages 1015–1021, 2000.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[17] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, June 2015.

[18] Roland Siegwart, Illah Reza Nourbakhsh, Davide Scaramuzza, and Ronald C Arkin. *Introduction to autonomous mobile robots*. MIT press, 2011.