Navigation and obstacle avoidance: a case study using Pepper robot

João R. Silva
Mechanical Engineering - POLO II, University of Coimbra
Coimbra, Portugal
Email: joao.ricardo.silva@uc.pt

Miguel Simão
Mechanical Engineering - POLO II, University of Coimbra
Coimbra, Portugal
Email: miguel.simao@dem.uc.pt

Nuno Mendes
Mechanical Engineering - POLO II, University of Coimbra
Coimbra, Portugal
Email: nuno.mendes@dem.uc.pt

Pedro Neto
Mechanical Engineering - POLO II, University of Coimbra
Coimbra, Portugal
Email: pedro.neto@dem.uc.pt

***Abstract*—In this paper we present a novel strategy and implementation of autonomous navigation for the Pepper robot. The proposed solution is modular and relies on the proper integration of existing Pepper functionalities. The human interacts with the robot using voice and/or gesture commands to setup the robot functionalities. An example use case demonstrates the ability of the robot to navigate in office environment and grasp an object to place in a target location. The robot is also able to avoid obstacles while navigating.**

***Index Terms*—Navigation, Obstacle avoidance, Pepper, Robot**

I. Introduction

Pepper is a companion/social robot designed to allow cognitive and physical interaction with humans. Social robots aim to improve the living conditions of people who interact with them. Pepper is able to recognize faces and basic human emotions like joy, sadness, anger or surprise [1]. These abilities make it ideal to engage with people through conversation.

The motivation to conduct this study comes from the fact that there is a lack of integrated solutions for navigation, obstacle avoidance and grasping of objects for the Pepper robot. In this paper, we study Pepper's capabilities to navigate, to avoid obstacles and to grasp objects. Based on off-the-shelf vision and navigation techniques, our objective is to use Pepper to move an object from a location to another one within an indoor space. In order to encourage the development of other related solutions the running code is provided in GitHub [2]. An example use case demonstrates the ability of the robot to navigate in office environment (Land Mark Detection) and grasp an object to place in a target location. The robot avoids obstacles while navigating and the robot paths are analysed.

Robots have been extensively studied, from autonomous navigation, to grasping to human-robot interaction. Tackling the social aspect, robots have been used in social experiments consisting in spoiling a game task [3]. In this study, the Pepper robot is controlled with the objective of bumping into a table where a human is playing jenga, and making the jenga tower collapse. It is studied how people react to mistakes caused by the robot and if there is any change to the empathy after a failed action when compared to humans. The CARESSES project is a joint EU-japan effort to build culturally competent assistive robots. It was studied the possibility of having culturally aware robots [4], [5]. In [4] it is discussed how to generate robot plans that respect cultural preferences, and how to execute them in a way that is sensitive to those preferences. In [5] the authors discuss how guidelines describing culturally competent assistive behaviours can be encoded in a robot to effectively tune in actions, gestures and words. Related to this topic, in [6] it is proposed an incremental learning model for emotional behaviours which also takes into consideration the cultural traits of the user, identified through long term interaction. The application of social norms in robot behaviour was studied and reported in [7]. Such approach uses the notion of an institution to realize social norms in real robotic systems. It is used a system of leader and follower robots to make a presentation of a room. The authors executed 30 real experiments with user-based evaluation with 40 participants.

In [8] it is created a virtual object dataset that integrates realistic robots (e.g. Pepper) in virtual environments to generate useful interaction sequences in order to boost the implementation of object recognition using deep learning. Object recognition using deep learning algorithms in a mobile service robot is discussed in [9]. To achieve their goals, the authors use two distinguished neural networks. The first one is YOLOv2 that incorporates an existent dataset, COCO, to identify the objects seen by the robot. This method relies on human help in the form of a survey interaction to validate or accurately labelling objects identified by the robot. The answers given by the user are saved and used to create a new dataset from scratch. These data are used by a second neural network called Human Help (HHELP). The two neural networks are used in parallel achieving better results and higher predictive precision.

Kinesthetic learning has been used to to improve the performance of autonomous feeding robots, allowing intuitive teaching by demonstration [10]. However, if the human user is not an expert in robotics, he/she will probably not take into

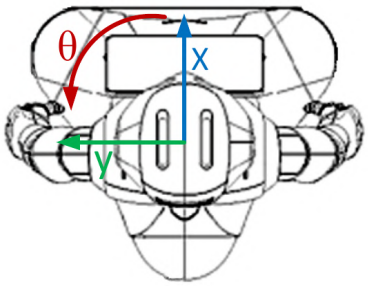

Fig. 1. Pepper coordinate system using *ALMotion* module.

consideration the energy cost of the demonstration activity, nor will be able to make a smooth path for a movement that involves more than 2 joints. To solve this issue, it is presented the Parameterized Similar Path Search (PSPS) algorithm, which allows the robot to improve robot motion learned from kinesthetic learning over a known cost function. Computer vision and deep learning technics have been implemented to teach a robot to copy and predict human movement to facilitate the interactive process [11]. It is proposed the use of a novel motion generative adversarial network (GAN) model that learns to validate the motion prediction generated by the encoder-decoder network through a global discriminator in an adversarial manner. The GAN-based approach is shown to outperform state of the art prediction approaches. The method incorporates images from an RGB camera and the depth sensor to form a point cloud. Each RGB image is processed using OpenPose to get the location of human joints in image coordinates, which in turn will allow determining 3D skeleton points of the human in robot coordinates.

II. Robot Interface

When using Pepper's interaction software (Choregraphe), it is available the option of turning off the autonomous life of the robot. By default, Pepper has autonomous life turned on, meaning that the robot is responsive and seeks outside stimulation to interact with. When disabling the autonomous life, Pepper turns unresponsive but executes programming commands with less regard for security. This aspect, although less safe, makes Pepper easier to maneuver around. The two modes are not quite compatible since if we want to move the robot around, we must renounce from the interaction aspect and vice versa. Our implementation method consists in creating a solution that runs the functionalities of autonomous life, while Pepper's autonomous life is disabled. We implement a python script that takes voice or even gestures from electromyographic (EMG) signals as input to run robot applications. These voice commands together with gestures allow to manage the interactive process with the robot.

A number of studies rely on Robotic Operating System (ROS) to develop solutions for Pepper robots [1]. Although this is true, we aim to study Pepper's capabilities with a python-based interface. It is easier to use and keeps the possibility to integrate external software tools such as artificial intelligence solutions. The python Software Development Kit (SDK) is used on a Pepper robot hardware version 1.6 with software updated to version 2.5. Robot and computer communicate through a TCP/IP connection. Both Pepper robot and computer are connected to the same network, the robot will be able to provide an IP Address that concede the possibility to remotely command it.

The implementation presented in this paper is developed using the python SDK with a *NAOqi* process that includes a public API with modules divided into groups: *NAOqi's* Motion, Audio and Vision. *ALProxy* provides access to every method or module available for robot programing that we need to connect to. There are two types of modules, local modules and remote modules. *NAOqi* provides services where local modules are in the same process so that they can share variables and call each other's methods. In our approach we use the remote modules to establish the robot-computer communication using the network and disabling autonomous life.

III. Modular Implementation

The implementation of the proposed application incorporates different Pepper's specific modules such as:

- *ALMemory* – Provides live data of different parameters such as position, temperature, stiffness. It also allows the storage and retrieving of saved information;
- *ALMotion* – Provides methodologies related to robot motion;
- *ALNavigation* – The navigation functionality allows Pepper to learn previously unknown locations, definition of a map, obstacle avoidance and retrieving points associated to the localization on the map;
- *ALTracker* – This module allows the robot to track different targets, such as landmarks and faces, or even a red ball. They can be used with the aim of establishing a bridge between target detection and robot motion to make the robot keep in view the target in the middle of the camera;
- *ALSpeechRecognition* – Provides the robot the ability to recognize a given predefined word or sentence.

A given robot program can be activated/deactivated using the speech recognition functionality in which we can setup a priori the vocabulary with the word we want Pepper to detect/recognize. Since Pepper has native speech recognition capabilities, it waits until an audio input that corresponds to a specified keyword triggers the associated action. To deal with false positive activation of voice commands, we set a threshold of 0.445. Through trial and error, we found this value to be suitable to keep undesired voice activation to a minimum. We also propose to command the robot by human upper limb gestures. The methodology followed to achieve this robot skill is described in our previous studies [12], [13]. Basically, a gesture is captured by a wearable sensory system (IMUs + EMGs) which sends gesture information to a remote application. This one, which is executed in a remote computer, consists in a machine learning algorithm to process

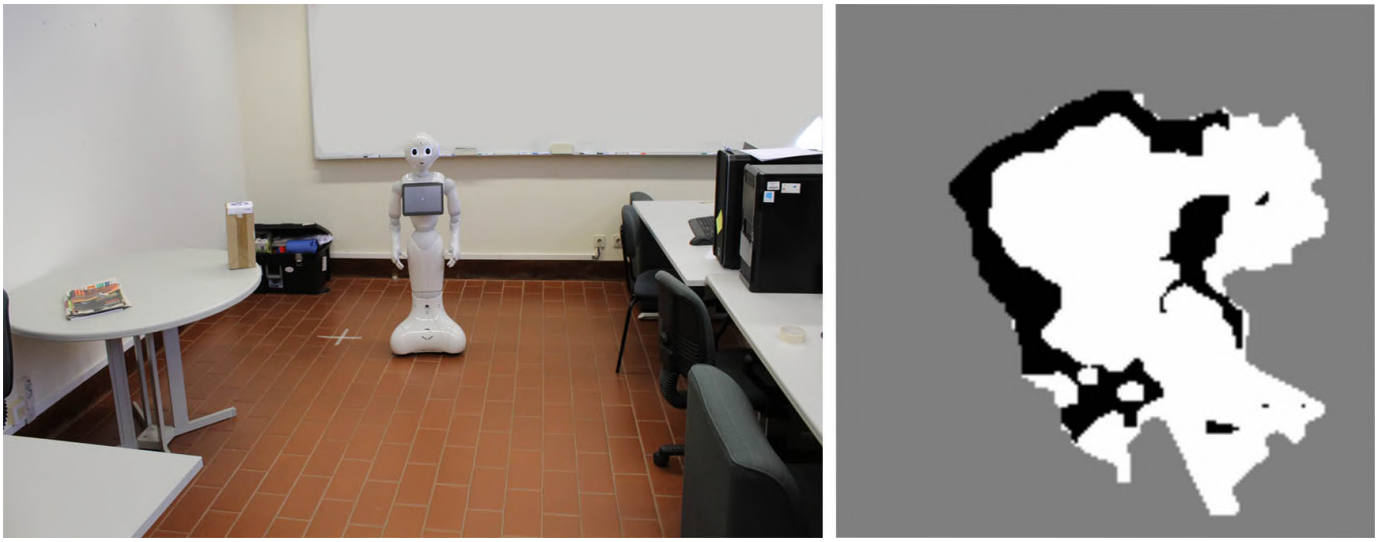

Fig. 2. Real environment (left) and resulting map representation of the explored area (right).

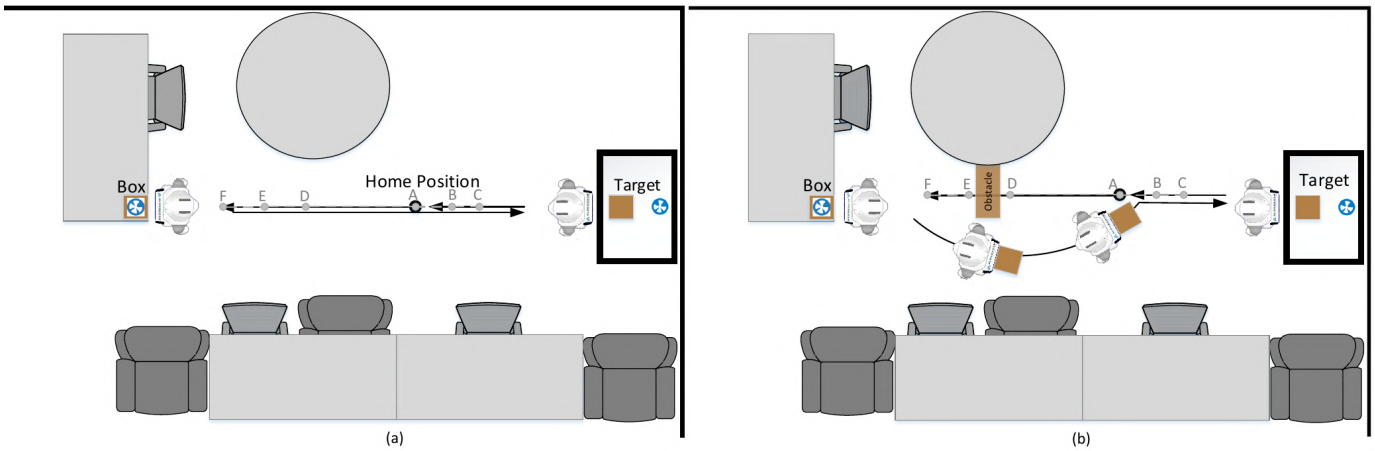

Fig. 3. Top view representation of the explored area. Robot path without obstacle avoidance (a) and with obstacle avoidance (b).

gesture information and identify which gesture a robot's user is performing. Whenever a gesture is recognized, a robot command (which is associated to an individual gesture) is sent to the robot.

A. Navigation

Simultaneous localization and mapping (SLAM) using Pepper sensors can be a challenge [1]. The lasers incorporated on the robot lack the capability to provide positioning points with enough consistency to create a detailed map for SLAM. However, if the exploration is done in the presence of well-defined edges through the physical walls of the desired map, the robot is able to create an understandable map of the environment. Light can be also an issue when creating a map, an intense light source directly on the sensors could result in several errors. In the same way, environments with too much free space near the floor can be problematic because the creation of the map uses sensors below the waist of the robot. Some robot tests were carried out on a real work environment utilizing furniture to better define the edges of the map. The working area and the respective resulting map are shown in Fig. 2 and Fig. 3.

At an initial stage of this study, we considered performing the exploration simultaneously with the landmark tracking in order to save specific locations on the map while its being created. This method revealed to be impracticable. This means that when the method is running no other method can run simultaneously, implicating that the point definition would have to be done separately and after exploration. To deal with this issue, the objective passed by being able to run each module at the time in order to obey the blocking calls and run the experiments smoothly.

While constructing the map, the robot creates a coordinate system relative to the position where Pepper starts the mapping process. When using the ALMotion module Pepper uses a coordinate system that consists in 3 coordinates [x, y, ϑ] being theta the rotation around z-axis in radians Fig. 1. However, when using the Navigation functions, Pepper does not use ϑ,

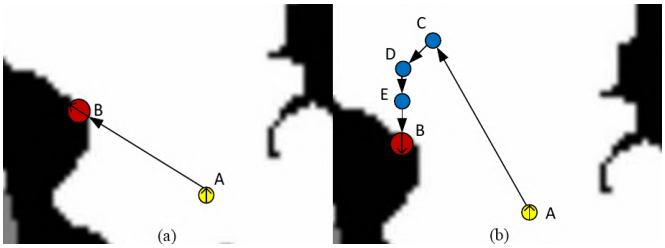

Fig. 4. Navigating with a direct route from point A to point B (a) and our proposed approach (b).

but instead it only uses xy coordinates relative to the map. Pepper can also choose its own path and speed while moving around. Like exploration, navigation also represents a blocking call.

After the mapping process is completed, since the navigation functionality only uses two coordinates and the orientation cannot be controlled, we felt the need of marking various points on the map in order to make good use of the navigation and obstacle avoiding methods. We addressed this issue by saving various points related to the pretended stoppage target, Fig. 4. If Pepper is standing in a position (point A with the orientation according to the arrow inside it) and intends to move to a target position (point B with the respective orientation), moving directly from point A to point B will result on the final orientation being the same as the one taken by the robot during the movement, Fig. 4 (a). This is problematic since we need to define the orientation that the robot should follow to perform the required tasks. To avoid the previously mentioned issue, we propose that after leaving point A, the robot should instead take a longer path to its destiny, moving first to point C, then D and E before arriving to point B. This way the robot is able to adjust its orientation.

B. Object manipulation

For the manipulation of objects, the robot was controlled using the Choregraphe software functionalities to define the motion pretended for each joint of the robot arms. The process started by taking Pepper's joints manually to the desired position, then the timeline box available in Choregraphe made it possible to store the values of the joints in a keyframe, creating a behaviour to the arms that would grab the object over time. Then, we exported the motion to python to be able to integrate it into our script.

Since the robot is in the desired position, the module *ALTracker* becomes active and the robot moves in the direction of the marker respecting a previously defined distance. The activation of the landmark tracker can accurately calculate the distance relative to the target and it will give us the precision we require to perform a given task. Since every landmark provided by Pepper is different and identified with numbers, it is possible to set different goal distances for each one. In the example in Fig. 5, the distance to the landmark positioned on the object was set to 28 cm and 65 cm to the landmark on the target. After Pepper grabbed the object, it is necessary to take care with its positioning during transportation, so that the robot does not block its depth sensor, used to calculate the distance to the landmark where Pepper should place the object. We opted to use an object with high volume and low mass, this way we can have a bigger surface for Pepper to grab the object and preventing possible floatation of the joints positioning that would be hard to compensate with a smaller object. An empty plastic jar, Fig. 7, with 230 mm in height and an average diameter of 120 mm was used in the experiment. We found that although the robot is able to lift more weight, this would compromise the grip of the fingers when moving the object.

IV. Experiments and Results

The main steps leading to the implementation of our experimental setup are represented in the flowchart in Fig. 6. The object and the target are distanced in 3 meters. The robot starts by making the exploration of a well-defined environment to create a map. After retrieving the map, it is used the ALMotion module in order to move pepper to the position correspondent to the points of interest. The ideal path is marked by letters in alphabetic order, Fig. 3. After moving a defined distance, when the robot reaches at one of the marked points, it uses the ALNavigation module to retrieve the coordinates corresponding to the current position on the map. After all the points are covered and all coordinates are retrieved, it once again uses the ALNavigation module to navigate the robot to the home position that corresponds to the point where the robot started both the exploration and the point definition. The actual path taken by the robot while defining points is represented by a black line, Fig. 5.

After initiating the behaviour that will command the robot to grab an object, the robot moves in direction of point F, and when this point is reached, the landmark detection is initiated. After detecting a landmark, Pepper will move in the direction of the landmark calculating its distance to it. Once the defined distance to the landmark is reached, an action follows. In this implementation, when the robot reaches the landmark placed on the jar it will trigger the movement of the arms to pick the object from a table and after completing this action. The robot then moves to the point of the map where the next action is triggered, as soon as it reaches point A landmark detection is once again enabled. This action triggers the robot to move in direction of the target landmark where Pepper will drop the plastic jar.

Dropping the object corresponds to the final task for this particular experiment, after the final task is fulfilled the robot will end the behaviour by returning to the defined home position. Using the `ALNavigation` instead of hardcoding coordinates with ALMotion is preferred since ALNavigation module makes it possible to use the obstacle avoidance method, where pepper can automatically estimate a new path to reach a certain point of the map.

A simple test is done to evaluate the efficiency of the presented method. After the mapping and definition of points, the program is executed 30 times to evaluate if Pepper can

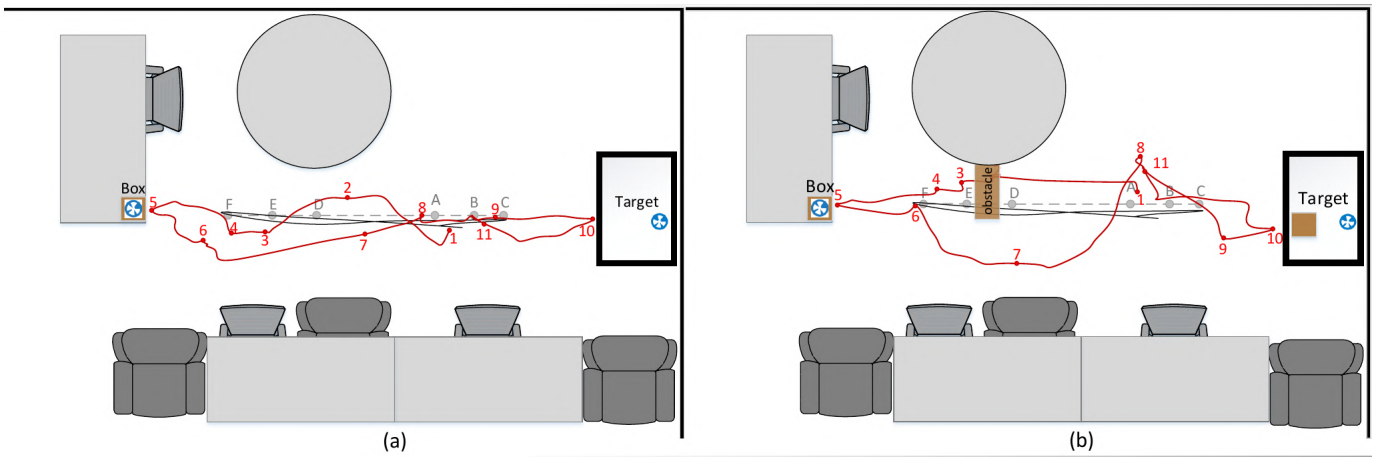

Fig. 5. Robot path without obstacle avoidance (a) and with obstacle avoidance (b).

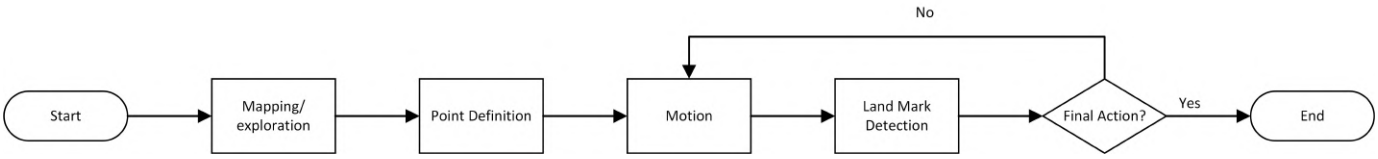

Fig. 6. Flowchart representing the main steps for implementation.

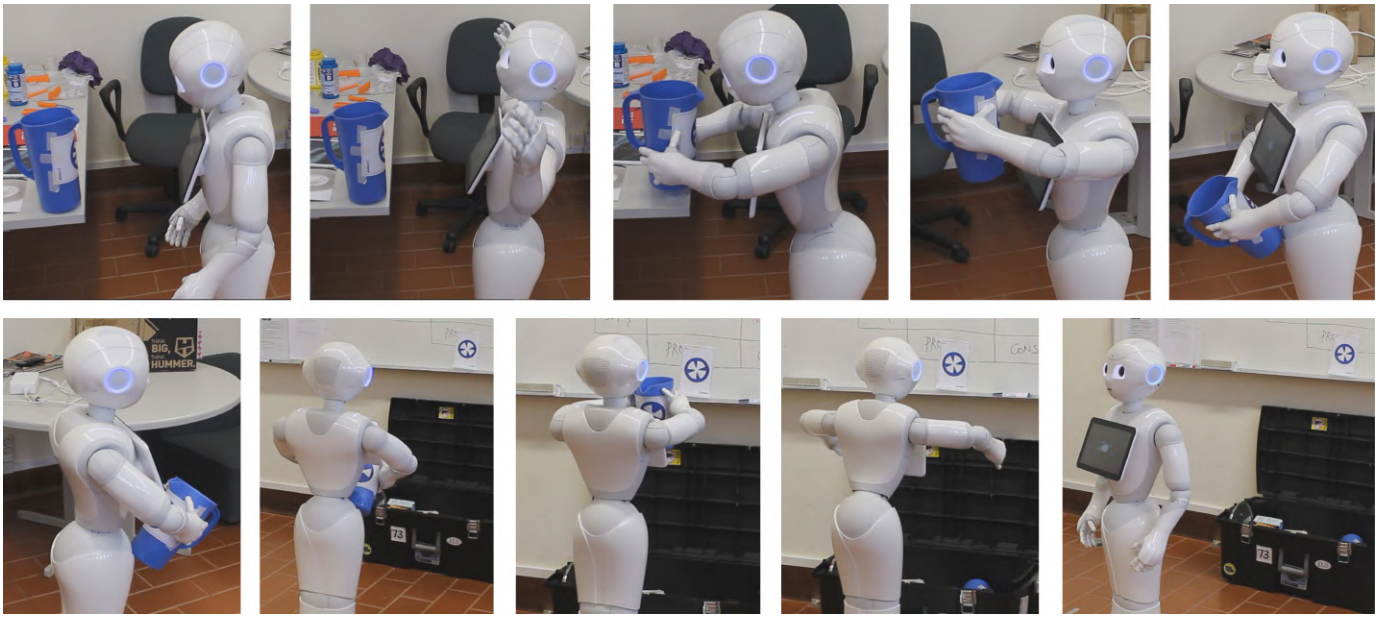

Fig. 7. Example use case sequencing.

TABLE I
RESULTS RELATIVE TO THE IMPLEMENTATION TEST

Collision avoidance scenario	Sucess	Fail
Without obstacle avoidance	13	2
With obstacle avoidance	9	6

pick the jar and take it to the target. The tests are then divided in 6 sessions of 5 tests each in a total of 15 tests being perform with obstacle avoidance and 15 without obstacle avoidance. To test obstacle avoidance performance, we divide the test in two parts. Both tests start the same way, the robot goes straight to the object, from home position to point F, picking it up. After, an obstacle is placed in the path that the robot would take to achieve the final target. The robot proceeds to make the necessary calculations to avoid the obstacle and reach the target in point A. It is important to note that the placement of the obstacle in the path should not occupy too much of the free space available for navigation.

While Fig. 3 represents the ideal path that should be taken by the robot, starting in home position, picking the object, taking it to the target position and then returning home. The actual paths taken by Pepper in the experiment can be visualized in Fig. 5. To create the representation of the paths, we implemented threads on the script that allow running different functions in parallel. While the robot is moving, another function is looping and storing the map coordinates every half second. After the task ends, it compiles the graphs that show the trajectory. The path taken follows the numerical order from Fig. 5. There is a floatation from the ideal pretended path, however, the iterative implementation of points improves the precision of the robot so that the task can be successfully completed. In the end of the experiment the results were promising, showing that the method was successful for the proposed goal. A clearer description of the results can be observed in Table I. The noted failures in the experiment occurred mainly at the start of the test sessions and were in their majority result of human errors. In the case of obstacle avoidance, the major error is due to the obstacle occupying too much free space. Since Pepper will try to keep a safe distance to every detected obstacle, if there is no space available in order to pepper navigate, the robot will struggle to move past the obstacle.

CONCLUSION

The proposed study approaches autonomous navigation and grasping for the Pepper robot. The robot is able to avoid collisions while is navigating, as well as grasp an object and move it to a target location. Also, the human user interfaces with the robot using voice and/or gestures to setup the modular architecture. An example use case demonstrates the ability of the robot to navigate in office environment and grasp an object to place in a target location. The robot is also able to avoid obstacles while navigating.

ACKNOWLEDGMENTS

This work was supported in part by the Portuguese Foundation for Science and Technology (FCT) project CEECIND/01421/2017, project COBOTIS (PTDC/EME-EME/32595/2017), and the Portugal 2020 project DM4Manufacturing POCI-01-0145-FEDER-016418 by UE/FEDER through the program COMPETE2020.

REFERENCES

[1] V. Perera, T. Pereira, J. Connell, and M. M. Veloso, "Setting up pepper for autonomous navigation and personalized interaction with users," *CoRR*, vol. abs/1704.04797, 2017.
[2] https://github.com/JoaoRicardofss/Peppergraspingandnavigation , 2019.
[3] R. Agrigoroaie, A. Cruz-Maya, and A. Tapus, "âoh! i am so sorry!â: Understanding user physiological variation while spoiling a game task," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 313–319, Oct 2018.
[4] A. A. Khaliq, U. KÃ¶ckemann, F. Pecora, A. Saffiotti, B. Bruno, C. T. Recchiuto, A. Sgorbissa, H. Bui, and N. Y. Chong, "Culturally aware planning and execution of robot actions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 326–332, Oct 2018.
[5] A. Sgorbissa, I. Papadopoulos, B. Bruno, C. Koulouglioti, and C. Recchiuto, "Encoding guidelines for a culturally competent robot for elderly care," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1988–1995, Oct 2018.
[6] N. T. Viet Tuyen, S. Jeong, and N. Y. Chong, "Emotional bodily expressions for culturally competent robots through long term human-robot interaction," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2008–2013, Oct 2018.
[7] A. Wasik, S. Tomic, A. Saffiotti, F. Pecora, A. Martinoli, and P. U. Lima, "Towards norm realization in institutions mediating human-robot societies," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 297–304, Oct 2018.
[8] A. Garcia-Garcia, P. Martinez-Gonzalez, S. Oprea, J. A. Castro-Vargas, S. Orts-Escolano, J. G. Rodríguez, and A. Jover-Alvarez, "The robotrix: An extremely photorealistic and very-large-scale indoor dataset of sequences with robot trajectories and interactions," *CoRR*, vol. abs/1901.06514, 2019.
[9] J. Cartucho, R. Ventura, and M. Veloso, "Robust object recognition through symbiotic deep learning in mobile robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2336–2341, Oct 2018.
[10] T. Rhodes and M. Veloso, "Robot-driven trajectory improvement for feeding tasks," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2991–2996, Oct 2018.
[11] L. Gui, K. Zhang, Y. Wang, X. Liang, J. M. F. Moura, and M. Veloso, "Teaching robots to predict human motion," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 562–567, Oct 2018.
[12] P. Neto, M. Simão, N. Mendes, and M. Safeea, "Gesture-based human-robot interaction for human assistance in manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 101, pp. 119–135, Mar 2019.
[13] M. A. Simao, P. Neto, and O. Gibaru, "Unsupervised gesture segmentation by motion detection of a real-time data stream," *IEEE Transactions on Industrial Informatics*, vol. 13, pp. 473–481, April 2017.