EU Horizon 2020 Research & Innovation Program

Advanced Robot Capabilities & Take-Up

ICT-25-2016-2017

Safe Robot Navigation in Dense Crowds

www.crowdbot.eu

Technical Report

D 3.5: Social Navigation

Work Package 3 (WP 3)

Navigation

Task Lead: SoftBank Robotics Europe (SBRE), France

WP Lead: Swiss Federal Institute of Technology (ETHZ), Switzerland

Table of Contents

Table of Figures

Executive Summary

This task brings socially aware navigation strategies on the commercial platform Pepper. Special focus was put on the factors:

- Safety: No physical harm.
- Comfort: Absence of annoyance and stress for humans.
- Naturalness: Similarity between robots and human's behaviour patterns.
- Sociability: Adherence to explicit high-level socio-cultural conventions.

This deliverable reports on the developments made by Softbank Robotics Europe, to meet the objectives of this task between months M1 and M30 of the project. Complementary work by ETHZ on interfacing social navigation behaviours with the CrowdBot local motion planner is described in CrowdBot deliverable D33[1]. Furthermore, complementary work by UCL on investigating similarities and differences between Pepper robot and a powered wheelchair in crowded social situations is presented in CrowdBot deliverable D14[2] section 2.2[3].

1. Overview

In recent years, robotic platforms have been deployed in public environments making evident the need of human-aware navigation capabilities. Tackling this need, numerous researchers have made efforts in order to include social conventions, such as proxemics, to create models for robot navigation. Nevertheless, few efforts have been made concerning the problem of labelling humans as an interactive agent when blocking the robot motion trajectory. Current state of the art navigation planners will either propose an alternative path or freeze the motion until the path is free. In this deliverable, we present our strategy to fulfil this gap in the social navigation capabilities of robots in Human-Robot Interaction. We started the research of this task with an HRI study in order to identify the effects of social interactive behaviours in a navigation task, this study is introduced in section 1.1. Due to the results of our study, we developed a framework prototype of the use of social interactive behaviours communicating with a navigation planner, an overview of this framework is presented in section 1.2. Furthermore, another social behaviour which facilitates the navigation in crowds and extends the capabilities of the robot, is introduced in section 1.3. Figure 1 shows the overview of the social navigation work for Pepper, which is detailed in sections 2-4.

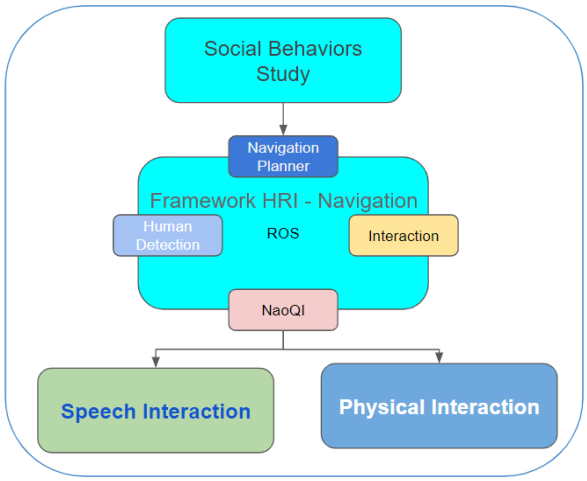


Figure 1. Overview of social navigation work for Pepper.

[1] *D33: Local Interaction Aware Motion Planning*

[2] *D14: First Round Test Evaluation Report*

[3] *2.2 Crowd / wheelchair / robot interaction in a controlled environment (PAMELA Dataset)* *13*

1.1. Investigating the effects of social interactive behaviours of a robot during a navigation task

Identifying the roles and the specific social behaviours that evoke human trust towards robots is key for user acceptance. Specially, while performing tasks in the real world, such as navigation or guidance, the predictability of robot motion and predictions of user intentions facilitate interaction. We present a user study [**1**] in which participants were guided by a humanoid robot around a human populated environment, avoiding collisions while following a socially acceptable trajectory. We investigated which behaviours performed by a humanoid robot during a guidance task exhibited better social acceptance by people, and how robot behaviours influence their trust in a robot to safely complete a guiding task. We concluded that in general, people prefer and trust a robot that exhibits social behaviours such as talking and maintaining an appropriate safe distance from obstacles. The procedure of the experiment as well as the results are presented in section 2.

1.2. Incorporating Human-Robot Interaction in navigation

Based on the conclusions of our user study introduced in section 1.1 and described in section 2, we implemented the first prototype of a framework designed to enhance social competency of robots while navigating in indoor environments. The use of a high-level situation assessment composed by a navigation module and a people perception module is proposed. This framework stops the navigation when a deadlock is generated by people standing in front of the robot and triggers an episodic interaction before re-initiating the navigation if the robot can move. Details of the framework are presented in section 3. In case that the robot is still unable to move, a solution providing physical interaction is introduced in section 1.3.

1.3. Socio-physical interaction module for semi crowded environments

The freezing problem when a robot cannot continue its navigation due to the presence of people in a semi crowded environment, for example in a cocktail party, could be solved by adding social and physical interactions. Then, our approach consists in considering people as interactive agents who can react to social cues of the robot to clear its path.

There are some situations where a physical touch by the robot could be useful, for example when the robot is in a noisy environment and the person does not listen to the petition of the robot to clean the path. In such situations, the robot should be able to detect the human position and also, the positions of the arms or hands of the person. Such information is of vital importance for achieving physical contact while preserving the factors of safety, comfort, naturalness and sociability.

We developed a system to detect human hands to perform a robot arm movement to touch the hand or the arm of a person in front of it, which is useful for asking for permission to pass in a semi crowded environment, and in this way clear its path and continue with its navigation. Details of this system are presented in section 4.

2. Investigating the effects of social interactive behaviours of a robot in a navigation task

Developing robots with the purpose of working in human populated environments highlights the need of integration of strategies that allow robots to move around people in a safe and socially acceptable way. However, such strategies raise new challenges in terms of defining safe robot navigation and natural Human-Robot interactions (HRI) [**32**]. Classical robot navigation approaches to people tend to focus on efficiency during the path planning and execution; variables such as path length, clearance and smoothness are used to quantitatively evaluate performance [**34**], especially in cluttered environments. In contrast, more recent studies have also focused on robot socially-aware navigation, in which a robot can adapt its behaviour by estimating how its position affects its performance on the navigation task while improving social acceptance [**35**]. Several research studies have proposed different strategies for socially acceptable movements for robots in complex environments where human users' presence and activities are unpredictable [**33**].

We designed an experiment to investigate human users' preferences to robot behaviours to enhance their acceptance of a robot in a navigation scenario. In particular, we were interested in investigating people's trust in a robot which is able to safely complete a navigation task. Namely, the robot guiding them in both wide and narrow spaces, such as doors or corridors, in a human-populated environment. We were also interested in collecting people's preferences and perceptions of the social behaviours exhibited by the robot during the navigation task.

This research has been guided by the following Research Questions (R):

– R1: What kind of robot behaviours do people prefer? A robot that behaves more like a tool or as a social entity?

– R2: Which social behaviours should a robot exhibit during a navigation task according to human users' preferences?

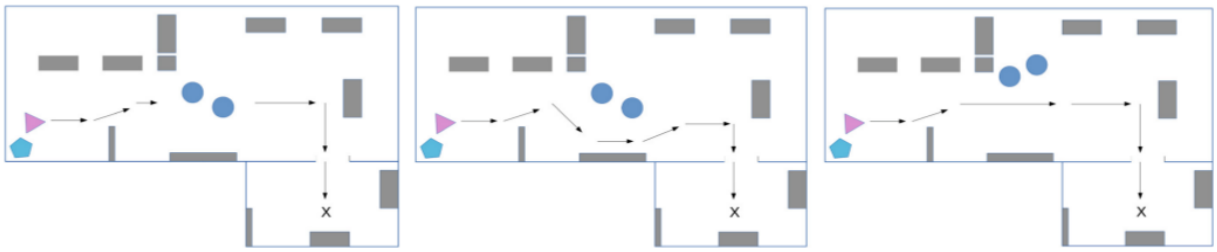
2.1. Experimental design

The experiment consisted of a robot guiding a participant through a passage where two people were standing, then the robot would perform one of three different behaviours, which are described below. The robot was being teleoperated but the participants were not informed about this.

We used a within-subject, counterbalanced, repeated measures experimental design. In order to test our research questions, each experiment was executed under 3 different conditions when the robot encountered people on its path towards the destination:

- condition C1: the robot stops and waits to have a clear path;
- condition C2: the robot performs simple obstacle avoidance while continuing moving;
- condition C3: the robot uses social behaviours to communicate and interact with participants. (speech and right arm movement in the direction of its path)

The participants were each asked to follow the robot and adapt their behaviours to the robot's behaviours. Two actors blocked the robot that guided the participants in the experimental environment (see Fig. 2). In order to analyse the interactions between the human participants and the robot, we asked the participants to complete questionnaires.



a) Condition C1: baseline b) Condition C2: non social c) Condition C3: social

Figure 2. Robot navigation behaviours. Pepper is represented by a triangle, the participant is represented by a pentagon, while actors are represented by circles.

2.2. Experimental Procedure

Participants were asked to imagine that they were in a shopping mall which they were not familiar with. The robot Pepper would help them to find the Information Centre. We told participants that they were free to position themselves next to, or behind the robot, according to their preference for following the robot. Participants completed a pre-experimental questionnaire to collect their experiences with robots and their perceptions of generic robots. Participants were presented with the same navigation task three times. All participants started their interaction with a control condition (C1). Then, they were tested with the social (C3) and non-social (C2) conditions in a randomised order.

During C1, Pepper stopped when it encountered two people talking to each other and blocking the robot's way (see Fig. 2a). When the two people cleared away from the robot's path, it then proceeded with its navigation. Once at the destination, the robot stopped. The experimenter informed the participants they had reached the Information Centre. At the end of the trial, participants completed a second questionnaire to collect their perceptions of Pepper and their self-confidence.

During C2, the robot did not exhibit any social behaviours, but it avoided the actors by moving according to the social norm of passing a person by maintaining position on the right side [**2**] and performing a simple obstacle-avoidance technique (see Fig. 2b). Once at the destination, the robot just stopped. At the end of the trial, participants were asked to answer the same questions as in the second questionnaire. In conditions C1 and C2, the robot moved at a constant velocity of 1 m/s.

In condition C3, the robot faced the participant and invited her to follow it to the Information Centre. It slowly increased and decreased the velocity, depending on whether the navigation was on a straight path, doing a turn or approaching an obstacle. Its velocity varied between 0 m/s to 1 m/s. It slowed down until it stopped when it detected people on its path (see Fig. 2c), while changing the colour of its shoulder LEDS to catch the participant's attention. It gently asked the two actors to let them pass saying "Excuse me, I would like to pass", and it thanked them once it had passed by. The robot continued its guidance task after gesturing with its arm for the participant to proceed. When it arrived at the destination, the robot verbally informed the participant while turning towards her.

At the end of the trial, participants were asked to answer the same questions as in the second questionnaire and another small questionnaire about their preferred social cues. At the end of the three conditions, participants were asked to complete a final questionnaire about their perception of the robot, their emotional state and preferences for Pepper's behaviours.

2.3. Results

We collected data from 12 participants. However, since it was important to test participants' trust of the robot and not of the human operator, at the end of the final questionnaire we asked participants if they believed the robot was behaving autonomously. We decided to exclude two participants who did not believe this. We analysed responses from the remaining participants (5 men and 5 women), aged between 22 to 40 years old [mean age 27.7, std. dev. 5.59]. All participants were PhD students, researchers and administration staff members at SoftBank Robotics Europe in Paris, France. The scenario of the experiment can be seen in Fig.3.

Figure 3. Scenario of the experiment.

2.3.1. Role of the robot

After each trial, we asked the participants to choose a role perceived as suitable for robots from the following: (1) friend; (2) butler; (3) companion; (4) pet; (5) assistant; (6) machine; (7) tool; or (8) other. These robot roles were chosen according to previous studies conducted by Dautenhahn et al. [**3**] and Ljungblad et al. [**4**]. Figure 4 shows that participants' ratings varied based on the different conditions. We can observe that their perception of the robot as assistant and machine drastically changed. People perceived the robot more as an assistant and less as a machine when it exhibited social behaviours. Figure 2 also highlights that there was a small variance in the perception of the robot as tool and companion when comparing these conditions. Participants considered a social robot less as a tool, while they considered a robot that is not social or does use simple obstacle avoidance techniques less as a companion.

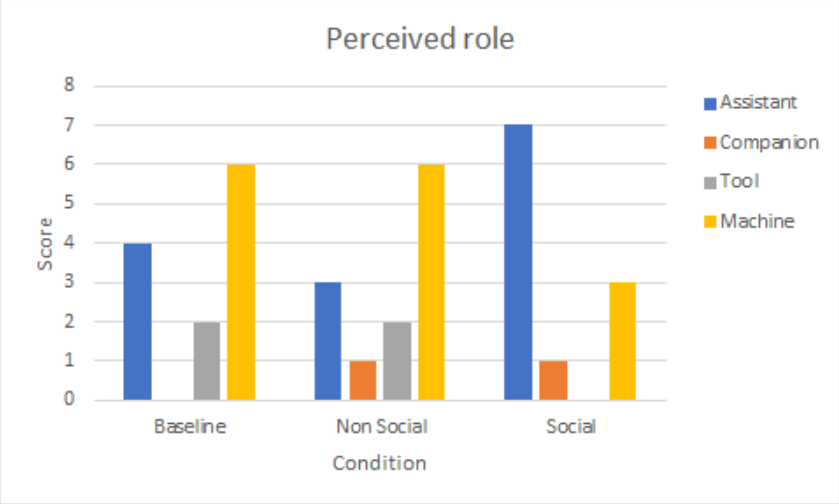


Figure 4. Participants' perception of the robot's role for each condition.

2.3.2. Preferred behaviour by participants

We asked the participants to choose the robot behaviour they preferred or considered most necessary during a guidance task from the following: (1) the robot waited to have a clear path (Baseline) and the robot exhibited social behaviours (e.g. talking, changing velocity, using lights, etc.); (2) the robot waited to have clear path (Baseline) and the robot avoided people without any social behaviour; (3) the robot exhibited social behaviours (e.g. talking, changing velocity, using lights, etc.) and the robot avoided the people without any social behaviour. All participants considered it necessary to have a social robot as a guide compared to both non-social behaviours and the baseline. They did not have specific preferences when asked to choose between a robot without social behaviours (50%) and a robot that waits to have a clear path before continuing its navigation (50%).

2.3.1. Social behaviour preferred by participants

As part of the questionnaire completed by participants after the condition in which the robot showed social behaviours, we asked them to select the specific behaviours they preferred during the guiding task with the robot. These were: speech; coloured lights; maintaining an appropriate distance from obstacles, humans or objects; approaching obstacles at an appropriate velocity, humans or objects, and narrow places; gesture; head and body orientation; or write down any social cues not included in the previous ones. These social cues were chosen according to [**5**], suggesting that naturalistic social interaction in robots can be designed through five main methods of non-facial and non-verbal affective expression: body movement, posture, orientation, colour, and sound. One participant did not answer the question, the remaining participants unanimously preferred a robot that is able to talk. Other preferences were for a robot that is able: to maintain the appropriate distance from obstacles (55.5%), to use gestures to communicate (33.3%), and approach obstacles and narrow places at an appropriate velocity (22.2%).

3. Incorporating Human-Robot Interaction in navigation

With the rise of social robots in our society, modelling interaction with people in real world scenarios is fundamental. Environments where robots are capable of navigating such as hospitals, hotels, restaurants, train stations or airports have been subject of research studies. In this regard, several methods have been proposed for navigating in dynamic and uncertain environments. In summary, they can be classified into two groups: model-based and learning-based methods.

The major exponents of model-based methods rely on social psychology and cognitive sciences to generate human-like paths for robot navigation. One of the most relevant approaches is the Social Force Model (SFM) [**6**], proposed to model pedestrian's behaviour whose motion is influenced by other pedestrians by means of repulsive forces. Many studies have implemented this method but also produced several variations [**7**], later applied to real world environment navigation where robots can avoid or go along with people [**8**]. However, these works show limitations such as the need for parameter calibration in different robots or the need of additional sensors for pedestrian tracking. Also, model-based methods are based on geometric relations, but it is still unclear if pedestrians always follow such models.

In contrast, learning-based methods use policies for defining human-like behaviours, which are usually learnt from human demonstrations by matching feature statistics about pedestrians. These methods apply machine learning techniques such as Inverse Reinforcement Learning (IRL) to model the factors that motivate people's actions instead of the actions themselves. An experimental comparison of features and learning algorithms based on IRL is presented in [**9**], where the authors conclude that it is more effective to invest effort on designing features rather than on learning algorithms. More recent approaches include the use of deep reinforcement learning in order to set restrictions [**10**] (i.e. passing at the left side of the people) instead of learning the features that describe human paths.

Both groups of methods described above focus on the motion of the robot but there are situations where more complex social behaviours are needed. As pointed by [**11**], polite navigation is an important requirement for social acceptance. According to this, an approach for management of deadlock situations at narrow passages is presented, where the robot lets the conflicting person pass and waits in a non-disturbing waiting position. A more recent work [**12**] proposes a navigation using two scenarios: in the first one, the robot asks for collaboration to enter a room. In the second one, the robot asks for permission to navigate between two people who are talking. However, results presented are shown in simulation.

In this section, we focus on a functional implementation and deployment of a scenario of a robot navigating in a corridor and two people blocking the path. These kinds of situations challenge existing planners, yet resulting in a robot freeze while the path is blocked or re-generating a new alternative path, if possible. Instead, the most natural human behaviour is initiating a dialog to ask for permission to pass. In this regard, we propose the use of a high-level situation assessment which is composed by a navigation module and a people perception module.

This framework stops the navigation when a deadlock is generated by the situation previously described and triggers an episodic interaction before re-initiating the navigation once again. As a result, this work incorporates human interaction as part of the navigation flow and deploys it on a robotic platform. The implementation of the framework presented in this section was done using an external CPU (Intel Core i7-3770 CPU 3.40GHz x 8) with Ubuntu 16.04 LTS.

3.1. Navigation

The navigation module is essentially composed of the ROS navigation stack including the packages Adaptive Monte Carlo Localization (`amcl`) and Dynamic Window Approach (`dwa_local_planner`). Due to the low detection range of the Pepper lasers, a similar approach previously presented by [**13**] and [**14**] has been used in order to convert the depth image into virtual laser data, using the ROS package `depthimage_to_laserscan`. Then, the resulting map is generated using `gmapping` (laser-based SLAM) and post-processed for improving its reliability. Similarly, a sketched map of the scenario was also used to improve the performance. The navigation uses a global planner with inflated obstacles and a local cost

map with observations from the virtual laser data. In this way, Pepper will stay away from possible collisions benefiting the motion of the robot.

The planning task is carried out by two main components; the global planner, which is in charge of providing path trajectory from the initial location till the target goal, and the local planner, responsible for obstacle avoidance in a close range while keeping an optimal distance to the global path. In case of failure, two recovery behaviours are implemented allowing the cost map to be cleaned and the robot rotated in place to find a new global path. However, we introduce a preliminary step where we expose the specific time the local planner could not find a valid plan in order to trigger the higher-level situation assessment that allows the identification of deadlock situations generated by humans.

3.2. People perception

Based on our computational restrictions, an accurate approach suitable for CPU processing was required. For this reason, the OpenCV dnn module composed of a `MobileNet-Single Shot Detector` (SSD) [**16**] trained in `Caffe` framework was chosen. This implementation uses an RGB image as input and it is able to detect up to 20 different classes, humans among them, despite occlusions and from different points of view. Once the person is detected, and with the aim to decrease the computational cost, a lightweight correlation tracker implemented on `Dlib` library [**17**] is applied. The module publishes a message every time a bounding box is bigger than an empirically predefined threshold (80 pixels width) in order to filter targets located far away from the field of interaction (see Figure 5).

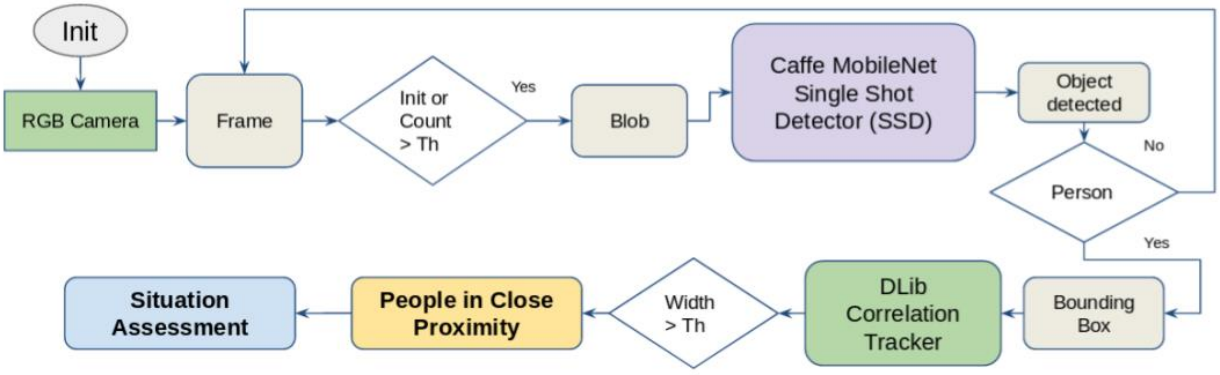


Figure 5. People perception module working principle.

3.3. Situation assessment framework.

The high-level situation assessment framework proposed makes use of the Navigation and People Perception modules to orchestrate the different transitions between them (see Figure 6). As pointed earlier, detecting a failure in the local planner is key to trigger an episodic interaction with the human blocking, but also considering that this event can take place due to several reasons (i.e. a failure due to loss of expected rate in sensor data acquisition). In our case, we are not only interested to know if there is a failure due to an obstacle in front of the robot, but also if the robot has detected a person as a source. At the same time, these changes are used in the global plan in order to know if a newly generated trajectory is better than the original one.

The navigation cycle is initiated by the NAOqi application Navigation App that loads the available destinations from a json file into the Location server. The same application is in charge of interfacing with ROS through the `ActionLib` client ROS package in order to generate the robot motion. As in any complex system, the inputs of the sensors and the outputs from the Navigation and People Perception modules are obtained asynchronously, but sequentially evaluated in order to stop or start the navigation and the user interaction. When the local plan has failed, the data retrieved from the virtual laser is grouped into clusters using

`MeanShift` from the scikit-learn library with a quantile value of *0.20* to estimate the bandwidth; if the mean of any of the clusters is smaller than a specified distance (< 3 meters due to the depth sensor limitation range), the People Perception module engages during *t* seconds in order to detect the human. In addition, if there is an existent plan generated by the global planner, a comparison with the initial trajectory is performed using the difference in path lengths. In consequence, if the newly generated path is shorter or equal than the original one, considering a threshold or level of consideration, the robot still can take it avoiding the interaction. Such measurement becomes a very powerful tool in order to define constraints that respect specific social contexts and groups. For instance, if a physically impaired person is detected or the robot is deployed in a cluttered environment, this would allow to modulate the robot's behaviour or eventually exclude the interaction. Once all the conditions described above are satisfied, the navigation is stopped and the robot starts the verbal interaction. In case the path remains blocked, the robot would wait *5* seconds and ask for permission a second time. Alternatively, the system can search for an alternative path.

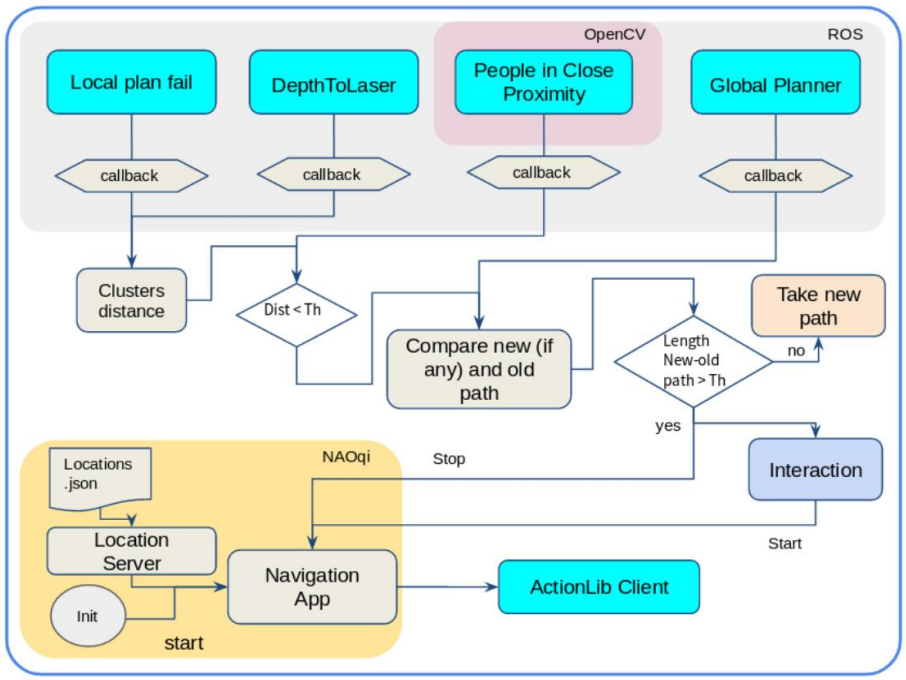
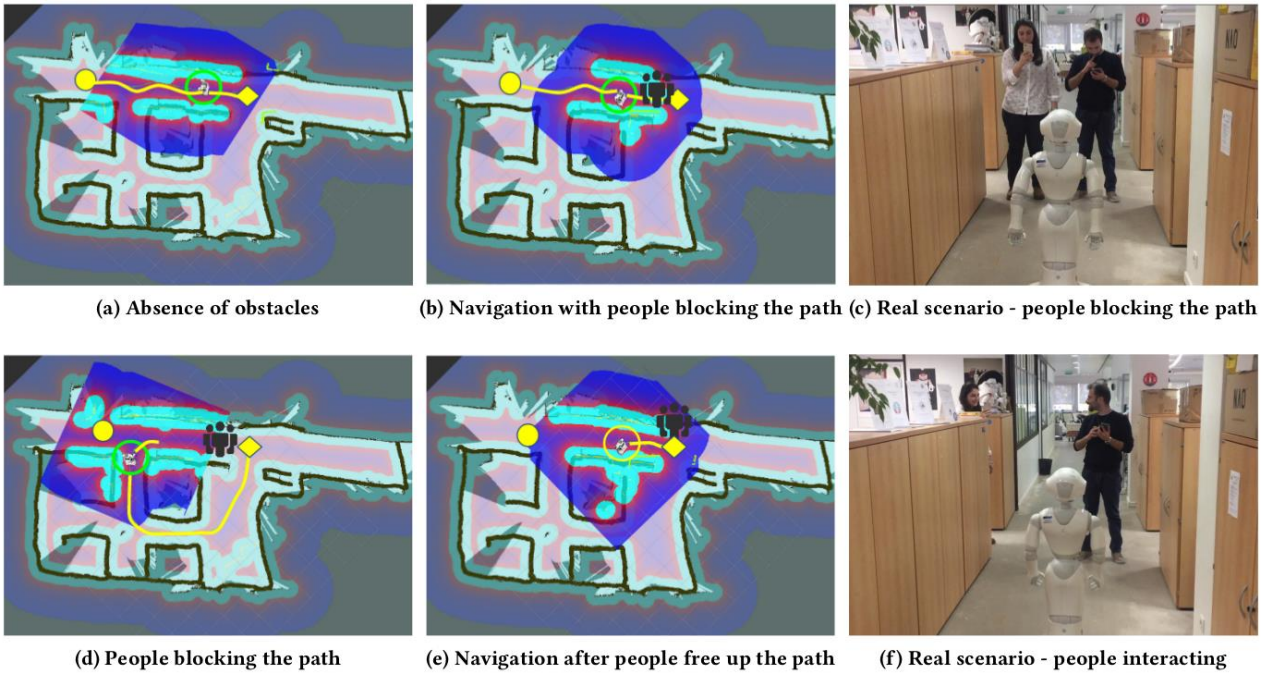


Figure 6. Situation Assessment workflow.

3.4. Results

Three cases where the robot is navigating in a hallway (Figure 7) are presented. The first one shows a free path and the other two people blocking it. Figure 7a shows the robot taking the shortest path due to the absence of any type of obstacles. In contrast, when the path is blocked potentially due to a human (see Figure 5d), the robot changes its initial plan and takes a longer path to reach its goal. In Figure 7b, the robot detects that the path is blocked by humans and interacts with them in order to free it. Finally, it will take a new plan that is no longer than the initial one (Figure 7e).

During our preliminary trials, it is worth mentioning that when the robot asked for permission to pass (figure 7c), one of the people standing moved out while the other remained blocking the potential trajectory. Then, an unexpected human-robot collaboration took place: the person aware of the intention of the robot asked the other one to free the way (see Figure 7f).



(a) Absence of obstacles (b) Navigation with people blocking the path (c) Real scenario - people blocking the path

(d) People blocking the path (e) Navigation after people free up the path (f) Real scenario - people interacting

Figure 7. Generated plans with ROS navigation stack (a) and (b), and with the framework proposed (d) and (e), and view of the real scenario (c) and (f)

Several limitations have been identified during the preliminary tests. First of all, since the computation is not on-board, the segmentation of the network penalizes the performance of the people perception module. Secondly, the situation assessment module working principle is not embedded into the navigation planner; with an integration to the ROS local planner, a consolidated approach could be provided.

4. Hand touching module

The goal of this module is to generate a robot behaviour capable of helping to clear the path of the robot in semi-crowded and loud environments while ensuring that the actions of the robot are socially acceptable. Such behaviour could be a speech for asking permission to pass, as described in the previous section. However, in loud environments a physical contact between the robot and a person in front of it could be more appropriate. Because of the humanoid form of Pepper, it can touch a person with its hand in order to avoid forcing the navigation through people. Nevertheless, the touching action needs to be carefully planned. A study on the meaning of touch [**18**] showed that the hands and shoulders are considered regions that serve to direct the recipient's perceptual focus, and that most of the touches in such study (80.3%) were initiated by hand alone. Touching the shoulder of a person from the back could have been an interesting behaviour, unfortunately the Pepper robot is too small to perform such behaviour. Then, the chosen behaviour was touching the hand of the person blocking the path with the hand of the robot.

The module of hand touching is composed of two submodules, see Fig 8. The first submodule consists of a detector of Human hands and Human arms that outputs a target, hand or arm 3D position, with respect to the robot base. The second submodule takes as input the output of the first submodule and uses a trained Deep Reinforcement Learning (DRL) model to move the robot in order to touch the hand of the person with the hand of the robot (or the arm in case that the hand was not detected), and at the same time directing the gaze of the robot in the direction of the hand.

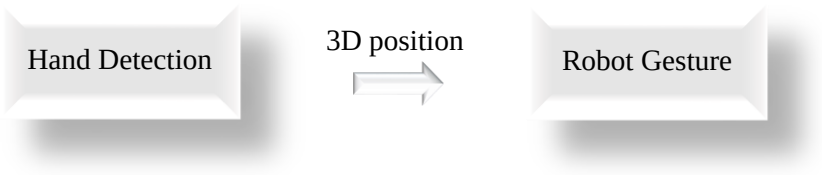


Figure 8. Hand Touching module, composed of Hand detection and Robot gesture Submodules.

4.1 Hand Detection

There are some existing solutions in the state of the art of hand detection that are readily available. One of these systems is OpenPose[**19**], which can not only detect hands, but also the whole human skeleton. Nevertheless, OpenPose was proved to be unsuitable for our needs, this is because of the short distance between the robot and the person in front of it in our target scenario. We need a system capable of detecting hands at a distance between 0.2 m. and 1 m. because of the specifications of the RGB-D sensor and the size of the robot.

Due to the high success of deep neural networks in computer vision tasks, YOLOv3[**20**] was chosen as the object detector to be used, which is one of the state-of-the-art deep learning algorithms that perform real-time object detection, on a Pascal Titan X it processes images at 30 FPS and has a mAP of 57.9% on COCO test-dev.

The approach of YOLO is to apply a single neural network to an image, this network divides the image into regions and predicts bounding boxes and probabilities for each region. The advantages of this approach over classifier-based systems is that its predictions are based on the whole image, giving to the predictions a global context, also it requires only a single network evaluation in contrast to systems like R-CNN[**23**] which requires thousands for a single image. Additionally, the YOLOv3 increases performance compared to previous versions of the system.

The output of the network is a prediction feature map containing a list of bounding boxes, where each bounding box is an array of the following attributes*: [tx, ty, tw, th, po, [p1, p2, …,pc]]* which represent the centre coordinates, the dimensions, the object score and C class confidences respectively.

A simplification diagram, shown in Fig. 9, extracted from the original publication of YOLO presents the idea of the algorithm with images.

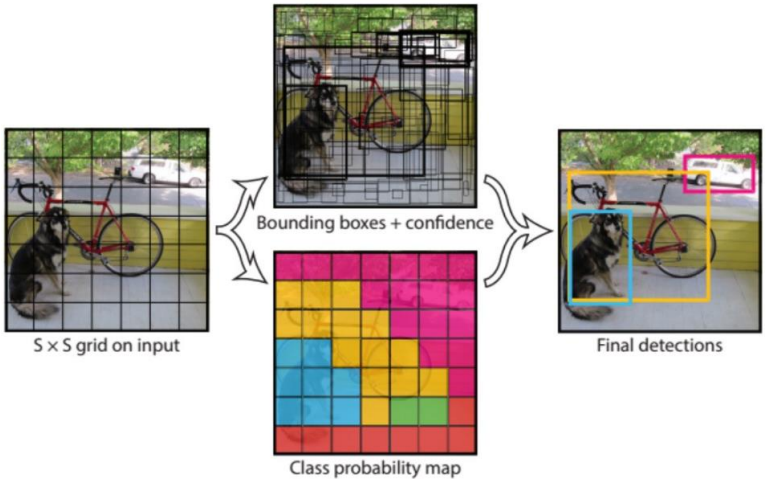


Figure 9. YOLO models detection as a regression problem. It divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities.

.

4.1.1 Dataset

The system uses, as other deep learning methods, a supervised learning approach for which it needs to be trained on a database of annotated images. For this reason, the Open Images Dataset (OID) [**21**] was selected. This database provides a large collection of labelled images and their bounding boxes. The reason behind of the choice of this dataset is that images are very diverse and often contain complex scenes with several objects, allowing the model to learn more complex patterns, see Fig. 10, in contrast to other datasets where the hands were already cropped from the full image [**24**] or they only contain images from first-person point of view [**25**]. Additionally, the OID contains image-level labels annotations, object bounding boxes, object segmentations, visual relationships, localized narratives, among others.

For training the hand detector, two classes were selected: human arm, and human hand. For each class, 10,000 images were downloaded. The arm class allows a secondary target in case the hands of the person are not visible or are not recognized.

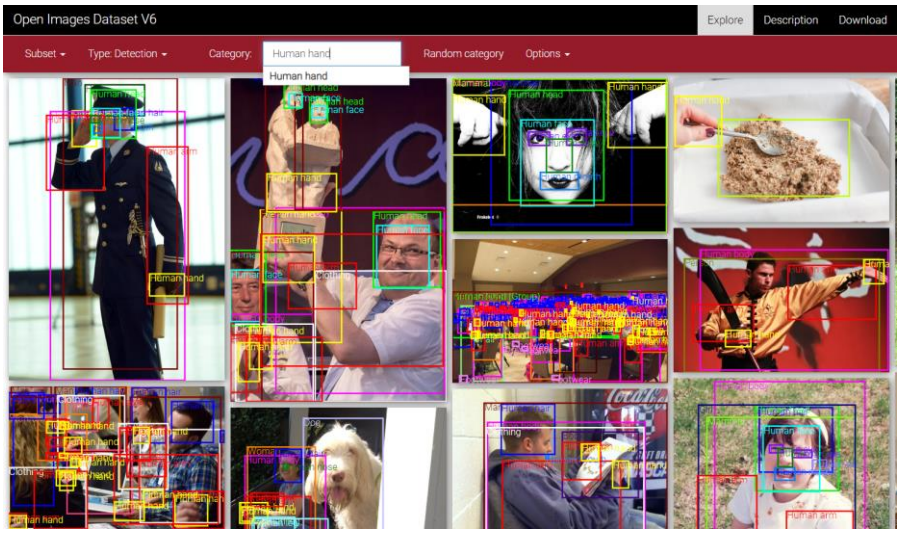


Figure 10. Result of a search for human hand on the online Open Image Dataset.

4.1.2. YOLOv3

The version of YOLO used in this work is the implementation of PyTorch-YOLOv3[4]. The images labels were pre-processed in order to meet the requirements of YOLOv3. The network was trained from scratch with the images downloaded from the Open Images Dataset.

4.1.3. Model Inference

The inference is done using the PyTorch framework inside a ROS package in order to facilitate the communication with the other modules of the CROWDBOT project. In order to retrieve the image, the ROS node in charge of the inference subscribes to the following camera topics:

```
/camera/color/image_raw
/camera/aligned_depth_to_color/image_raw
/camera/depth/camera_info
```

The first topic is used to receive the image which is used as input to the YOLO trained model. The other two topics are used to obtain the 3D position of the bounding box given by the output of the model. The Realsense D435 camera includes a depth sensor that leverages the obtention of the 3D position of the hand (or arm) with respect to the camera.

Depending on the availability of CUDA the implementation of the model uses GPU or CPU to make the inference. Once an image has been obtained through a ROS subscriber, it is converted to an image tensor to be used by the model, which uses the `non_max_suppression` method over the detections with an object confidence threshold of *0.9* and an *IoU* (intersection of union) of *0.4*. The detections are re-scaled to the original size of the image, discarding too small or too big images using empirically obtained thresholds (1/7 and 1/2 ratios of the image size).

In order to speed up the conversion of the depth points in the region of interest to a single 3D position, it is calculated taking the depth information of *10* random pixels from the centre of the bounding box and computing its mean. Afterwards, the ROS `tf2 package` is used to translate the hand position to the base of the robot in order to perform the hand gesture. Finally, the 3D position is published to a ROS topic if it is a new detection or if the previous published point is a range of *0.2* m. The topic of the 3D position of the target (hand or arm) receive a message of type `geometry_msg/PoseStamped` which is named:

```
/hand_detection/target_position
```

[4] *https://github.com/eriklindernoren/PyTorch-YOLOv3#pytorch-yolov3*

4.1.4. Results

After training the model for 400 Epochs of 2400 steps each epoch, the mean average precision (mAP) on the testing dataset was around 0.24% and a loss value of 0.67, while it still has room for improvement it is acceptable for our needs.

The training took about 30 hours on a PC composed of a 1080 Tegra Ti GPU. Further improvement on the mAP would require more and more time and it will increase only a small fraction of the current result. The loss value of the neural network is presented in Fig.11 showing the relative time for each step, as the training was done in separate times each 100 epochs, it can be seen that the log after 100 epochs had descended to 2.82, after 200 epochs it was 1.14, after 300 epochs it was 0.83 and the final value after 400 epochs was 0.67.

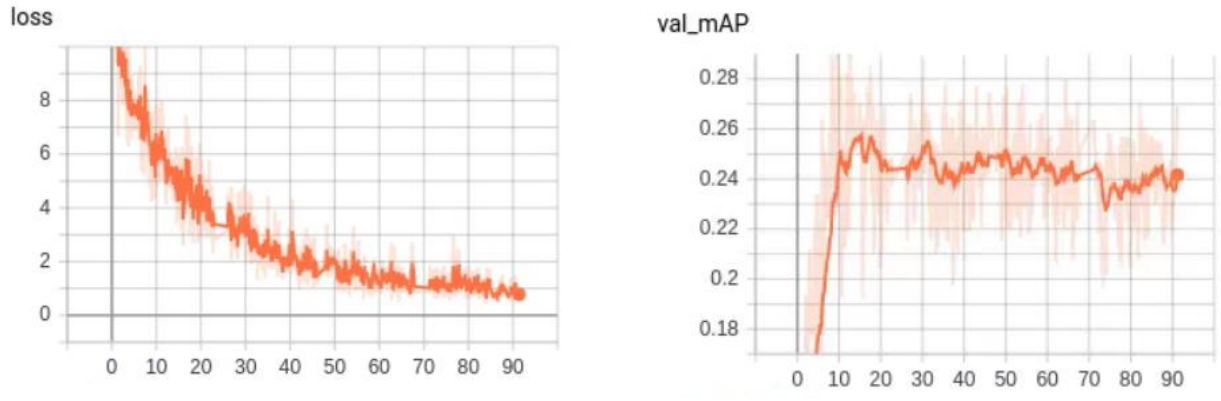


Figure 11. Loss and mAP of the neural network showing the relative time for each step using TensorBoard

Comparing the loss with an example of avg loss taken from the git repository of YOLO[5] (we can see in Fig. 12 how the loss value and the mAP tends to stabilize at the same time, and the value of loss is harder and harder to decrease.

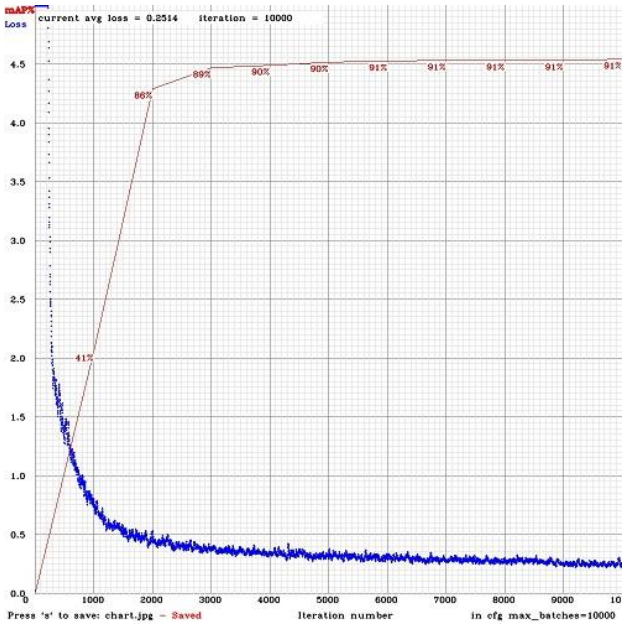


Figure 12. Example of loss and mAP on an object detection training using YOLOv3

In the validation set, the arms present a better mAP (0.33) than the hands (0.27). An example of an output of the trained model can be seen in Fig 13, where the arms are better detected than the hands.

[5] https://github.com/AlexeyAB/darknet

Figure 13. Output of the trained model using YoloV3 for detecting human hands and human arms

The model was also tested with images streamed from the Realsense D435 camera set on the robot in order to verify that it fulfils the needs of this module. An example is shown in Fig. 14, where it can be seen that the model is able to detect human hands at a close distance (<1m.).

Figure 14. Output of the trained model using images streamed from the Realsense D435 camera set on the head of the robot.

4.2 Robot gesture

The goal of this module is to generate a robot movement for allowing it to reach a 3D position target with its hand. The scenario where the robot gesture will be used, which is a Human-Robot Interaction, imposes some restrictions such as the movement of the robot needs to be fast enough to follow changes in the position of the target (hand or arm), also as the size of the robot is small compared to an adult person, using the hip joint to reach a point with the hand would be useful as this increase the action space of the robot. Currently, the NAOqi framework has a module for pointing with the arm to a given 3D position. Nevertheless, it has some limitations like the movement does not include the hip and the head of the robot and it is done in a blocking call, which means that the robot will execute following orders until the movement is done. This is a disadvantage for having a tracking with the head at the same time. Instead, having a head tracking following the target and maintaining it in the field of view of the robot could be of aid for assuring a reactive gesture.

The problem of the movement of a kinematic chain given a desired position of the end effector is a well-known problem in robotics known as Inverse Kinematics (IK) where the most common approach is to use a Jacobian matrix to find a linear approximation. There are many public software libraries for solving IK, including MoveIt, a framework for solving motion planning for robotics. However, IK solvers do not ensure a solution and they could require variable time to find one.

Current approaches in the state of the art for generating robot movements use Deep Reinforcement Learning (DRL) methods [**26**]. Reinforcement Learning (RL) is a machine learning approach for teaching agents how to solve tasks by trial and error by specifying a reward, then Deep RL is the combination of RL with deep learning. Controllers based on DRL can be very computationally efficient at runtime, they provide a control policy learning system, but they do suffer from extremely slow training times. Because of this, the common

way to train DRL in robotics is through simulation, an example of this is the work of [**27**] where the authors teach a simulated robot to move like an animal, then the learned policy is adapted to a real robot. DRL has been used before for teaching the Pepper robot to interact with humans, in [**30**] the authors propose the use of a dual stream convolutional neural network for processing the images inputs of the RGB and depth channels separately. The actions of the robots are hand-coded, they include waving hand, waiting, looking towards the human, and handshake. The robot is positively rewarded when there is a successful handshake and negatively rewarded otherwise.

4.2.1. Proximal Policy Optimization

Policy gradients methods are one of the major pieces in recent improvements in deep learning for control. Nevertheless, they are sensitive to the choice of stepsize for updating the gradients, with a small step the progress can be slow, while a big step could lead to loss of performance, also they are sample inefficient requiring millions of steps to perform simple tasks. PPO was chosen to learn the policy for the robot gesture

The Proximal Policy Optimization (PPO) algorithm [**31**] seeks to improve previous methods by attaining the data efficiency and reliable performance of Trust Region Policy Optimization, while using only first-order optimization. PPO uses an objective with clipped probability ratios, which forms a lower bound of the performance of the policy. To optimize policies, it is alternated between sampling data from the policy and performing several epochs of optimization on the sampled data. A proximal policy optimization (PPO) algorithm that uses fixed-length trajectory segments is shown in Algorithm 1. In this work, PPO was chosen to learn the policy for the robot gesture.

Algorithm 1 PPO, Actor-Critic Style

for iteration = 1,2, … **do**
  for actor = 1,2, … N **do**
    Run policy $\pi\theta_{old}$ in environment for T timesteps
    Compute advantage estimates $A_1, ..., A_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{old} \leftarrow \theta$
end for

The objective function is calculated as follows:

$$L_{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\ ]$$

where

- θ is the policy parameter
- $\hat{E}t$ is the empirical expectation over timesteps
- rt is the ratio of the probability under the new and old policies, respectively
- $\hat{A}t$ is the estimated advantage at time t
- ε is a hyperparameter, usually 0.1 or 0.2

4.2.2. Environment and models

In this work, OpenAI Baselines[6] are used, it is a set of implementations of RL algorithms of the state of the art, more specifically a fork of Baselines called Stable Baselines [**28**] is used, which provides unified structure and documentation[7] for the algorithms.

Training environment

In order to train the model, we used a physical simulator called qiBullet [**29**] which is a Bullet-based simulator for the Pepper and NAO robots. The qiBullet simulator[8] inherits the cross-platform properties of the PyBullet Python module and Bullet physics engine. Also, PyBullet is a fast and easy to use Python module for robotics simulation and machine learning, with a focus on sim-to-real transfer. Examples of simulated environments in qiBullet are shown in Fig. 15.

The simulated environment used in this work involves two robots, one of them is controlled by the PPO algorithm while the other is set at random positions. The goal is to train a policy able to perform a hand gesture that touches the hand of the other robot regardless of its position, while tracking it with the head and the right hand of the robot. The second robot is used because of simplicity. The simulation runs at the default rate of 1/240 seconds using the method `stepSimulation`().

The choice of using qiBullet simulator instead of the one delivered by INRIA for this project, was done mainly because of two reasons: 1) qiBullet offers a high-level python API that facilitates the use of the Pepper robot such as obtaining the positions of all the motors, obtaining the names of the joints, or applying different speeds to the motors with a simple call of a method. 2) qiBullet, as it is based on PyBullet, can be used easily with TensorFlow and OpenAI for applying Reinforcement Learning algorithms. Furthermore, as qiBullet is not intended to be a simulator for navigation or crowds as is the case for the CrowdBot simulator, the latter could be used for testing the generated robot behaviour in crowded scenes.

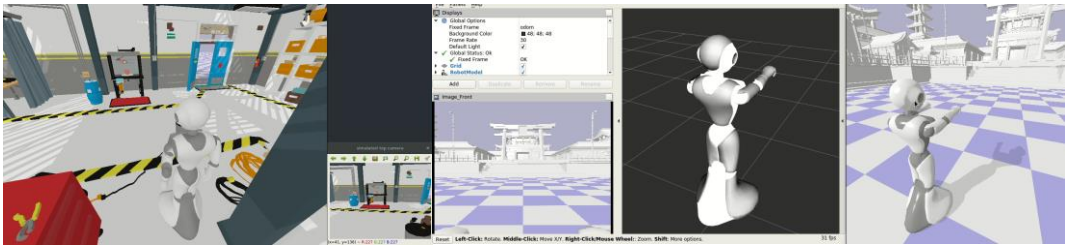


Figure 15. Examples of simulated environments in qiBullet.

Target

The target is given by the 3D position of the left hand of the second robot having the world as reference frame. The position of the robot and its left hand are generated randomly at each episode. The base position is set in a range of *[0.65 – 0.85, -0.3 – 1]* for *x* and *y* coordinates respectively. The left shoulder pitch is set in a range of *[0.4 – 0.8]*. This range gives a target space that contains several reachable targets and some that are not.

The position of the second robot is intentionally set at the right side of the training robot in order to facilitate the learning. Furthermore, an offset of *5cm* in the *y* axe was added to the target, so the robot would not try to reach the joint position inside the hand of the other robot.

[6] *https://github.com/openai/baselines*

[7] *https://stable-baselines.readthedocs.io/en/master/index.html*

[8] *https://github.com/softbankrobotics-research/qibullet*

Body

The model of the simulated robot is given by the qiBullet simulator, it has 11 joints, but only 6 are used for training the movement.

Observations

The observations are composed of the 3D positions of the right hand of the training robot *rh[x,y,z]* and the left hand of the second robot *lh[x,y,z]*, and the angle in radians of the joints of the training robot *J[jθ]*, and the direction vector of the head of the training robot *dv_h[x,y,z],* and the direction vector of the position of the head of the training robot towards the position of the hand of the second robot *dv_lh[x,y,z]*. All observations are normalized between *[-1,1]*.

The joints for training the robot movement are: Hip Pitch, Right Shoulder Pitch, Right Shoulder Roll, Right Shoulder Elbow, Head Yaw, and Head Pitch. Other joints were excluded even if they could improve the action space of the robot, because it would lead to loss of naturalness of the motions.

Actions

The actions are the normalized velocities of the joints. The velocities were also limited to 0.75 for the head joints and to 1.0 for the rest of the joints.

Rewards

We use a combination of two rewards, the first one is related to the movement of the arm of the robot, and the second one is related to the movement of the head.

The arm reward is defined as:

$$a\_r = exp(-1\ ||rh - lh||)$$

Where *rh* is the 3D position of the right hand of the training robot and *lh* is the 3D position of the left hand of the second robot, both having as reference the world coordinates.

The head reward is defined as:

$$h\ r = \exp(-1\ ||dv\_h - dv\_lh||)$$

Where *dv_h* is the direction vector of the head of the training robot and *dv_lh* is the direction vector of the position of the head of the training robot towards the hand of the second robot, both having as reference the world coordinates.

They were combined into a single reward assigning a weight for each one:

$$R = 0.75\ ar + 0.25\ hr$$

4.2.3. Results

Training the robot in simulation, the combined reward increased in the first 10 million steps, after that it decreased and only got similar results after more than 30 million steps. The stop condition to restart the episodes was when the arm of the robot made a collision with itself, and after 5 seconds of the beginning of the episode. The model was saved each 10 million steps.

The training log of the reward is shown in Fig.16, where each colour represents 10 million episodes.

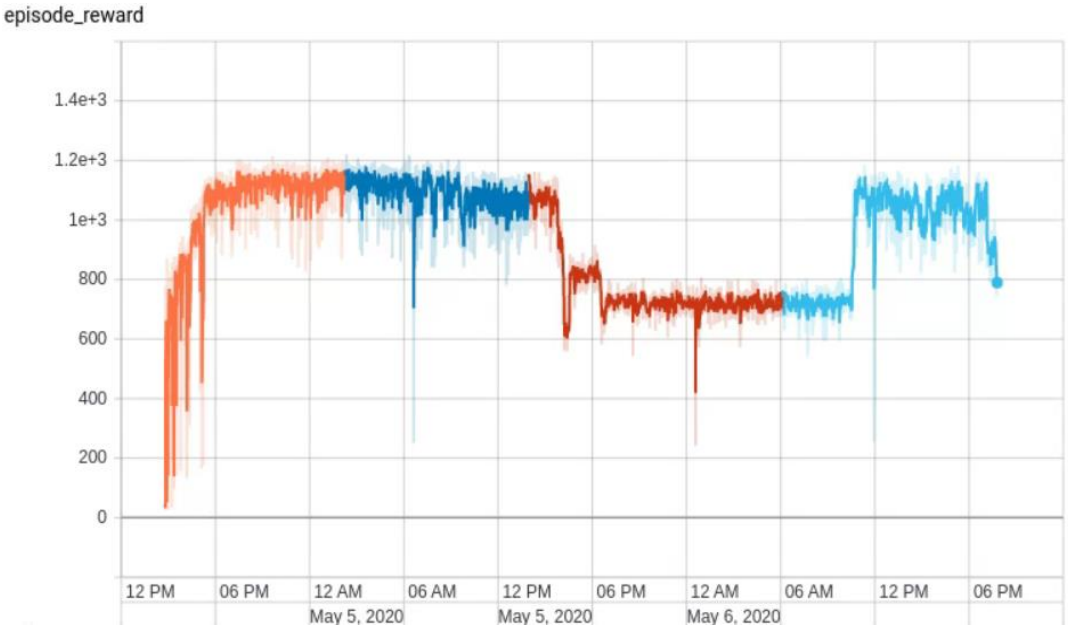
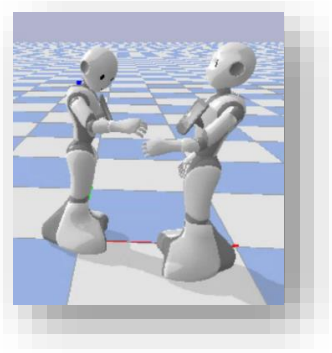


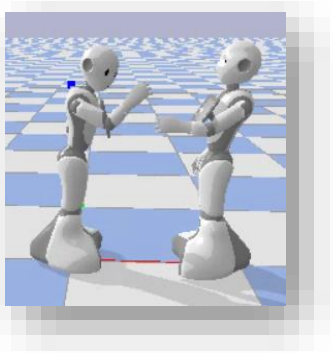
Figure 16. Training without position constraints.

At testing time, the hip pitch of the robot was limited to a range of *[-0.4, 0.2]* in order to avoid undesired motions. The model saved after 10 million steps (see Fig. 17a), shows a motion of the robot including the arm, hip and head that achieves to move towards the hand of the second robot at any position. After 20 million steps (see Fig. 17b) the motion was less accurate, finishing the movement with the hand farther than after 10 million steps. The model saved after 30 million steps was even less accurate (see Fig. 17c). The final model saved after 40 million steps (see Fig. 17d) showed a motion similar to the one after 10 million steps.

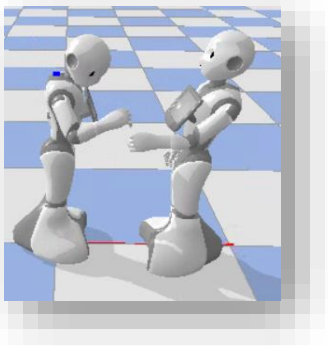
Different motions for different positions of the second robot are shown in Fig. 18, it is worth saying that the robot adapt its movement each time new observations are received.



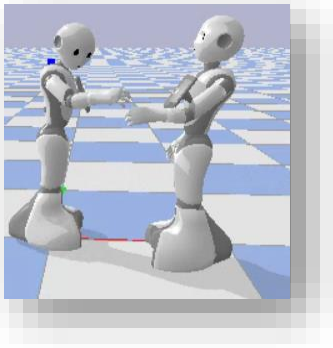
a) 10 million steps



b) 20 million steps

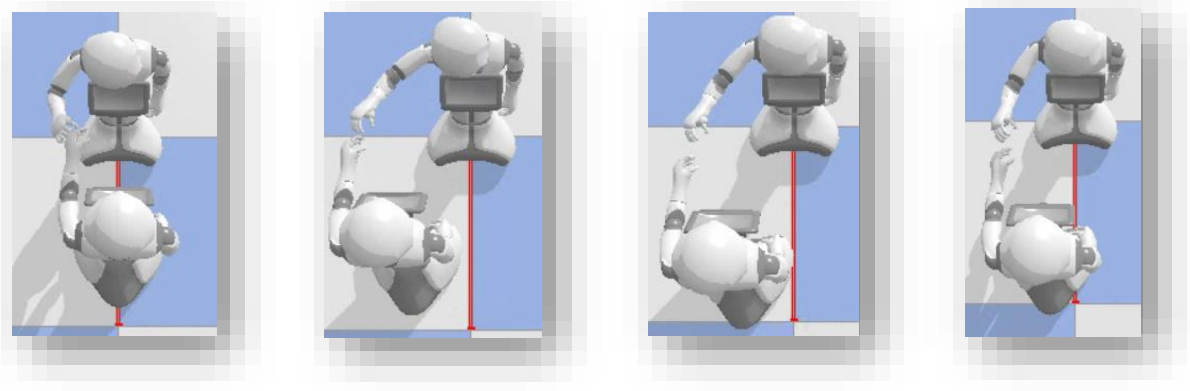


d) 30 million steps



c) 40 million steps

Figure 17. Motion for each saved model with the combined reward.



a) b) c) d)

Figure 18. Different motions for different target positions generated with the model with the combined reward after 40 million steps using qiBullet Simulator.

Other setups were considered for training, showing similar results but with higher variance in the reward for each episode. One of these setups was limiting the angle of the joints at training time in order that they do not reach undesirable positions. The log reward is shown in Fig. 19.

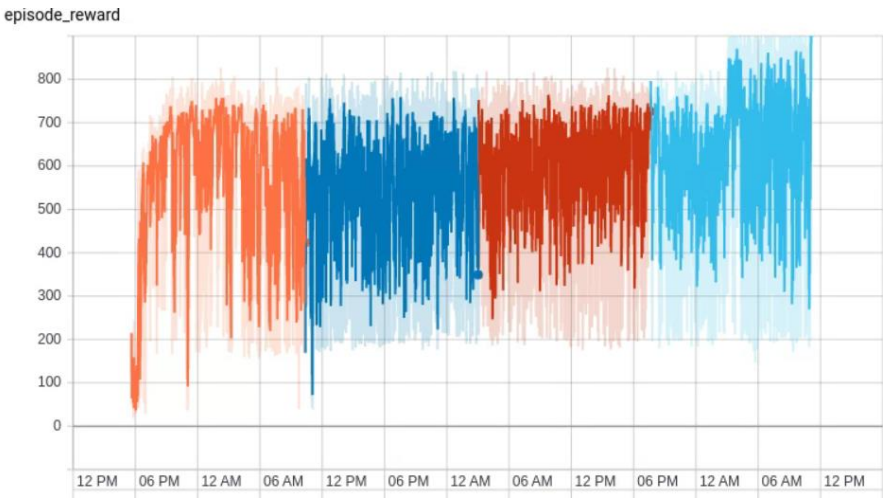


Figure 19. Training the robot with the combined reward with joint angle constraints

Also, the arm motion was trained using only the arm reward, keeping the constraints only on the hip pitch of the robot. Figure 20 shows the training log of the reward, where each colour represents 10 million steps, it can be seen that the reward achieved a better value after almost 40 million steps.

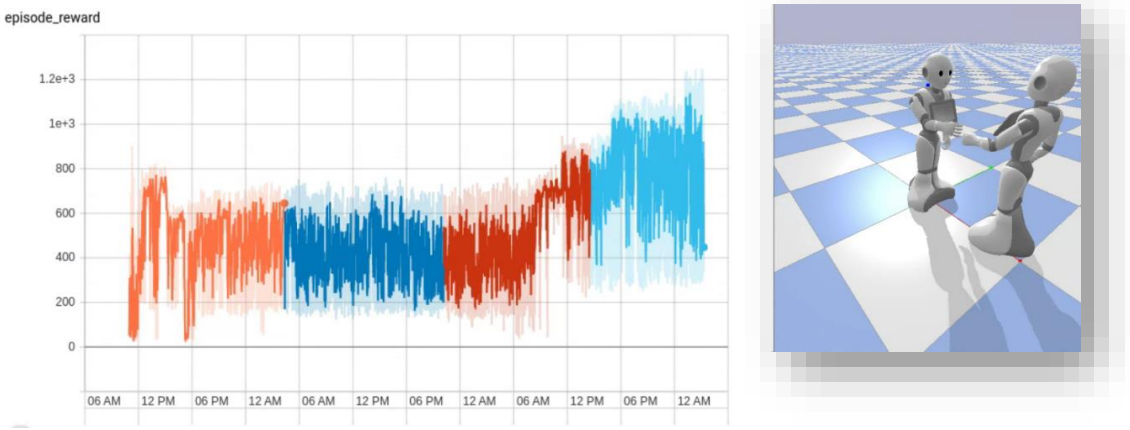


Figure 20. Training the robot with arm reward alone.

The head motion was also trained alone using only the head reward. Figure 21 shows the training log of the reward for 20 million steps, it can be seen that the reward of the head motion alone achieves its maximum value after less than 2 million steps.

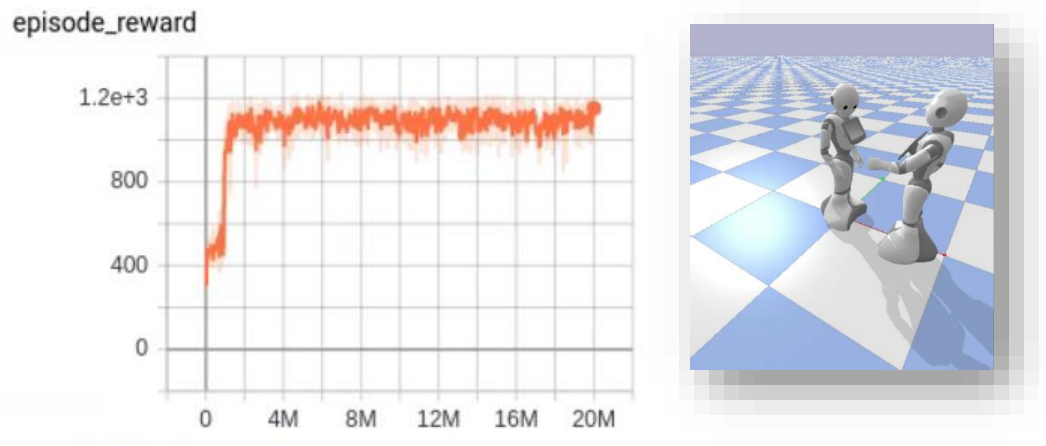


Figure 21. Training the robot with head reward alone.

5. Conclusion

This work package presented a set of social navigation strategies for the robot in order to enable it to move in a more intuitive and socially acceptable manner. The first part, presented in section 2, presented a study on the effect of social interactive behaviour of a robot in a navigation task, results showed that a humanoid robot that uses speech when is confronted with a deadlock situation in a guiding task, is perceived more as an assistant and less machine-like compared to the same humanoid robot that only avoid humans as obstacles or that it stops completely until the path is free. This result, highlights the need of social behaviours in robot navigation when it is confronted when situations that can be solved by using human-like behaviours.

The second part of this work package, presented in section 3, introduced a framework where robot speech can allow to unblock deadlock situations generated by the presence of humans in the robot path. Human detection is an important component of this framework, and the module designed for this purpose in the CROWDBOT project is of vital importance to achieve a robust system for social interaction. Nevertheless, only detecting the position of the people might be not enough for physical interactions, particularly when the robot is performing a motion with its arm, where safety and comfort are crucial.

Finally, in section 4, a hand touching module is presented. The hand touching module is composed of two sub-modules, the first one is in charge of the hand and arm detection at close distances, the second one performs a movement trying to reach the target given by the former, the motions where trained using deep reinforcement learning with only a reduced set of the robot joints in order to increase their naturalness.

References

[1] Rossi, A., Garcia, F., Maya, A. C., Dautenhahn, K., Koay, K. L., Walters, M. L., & Pandey, A. K. (2019, July). Investigating the effects of social interactive behaviours of a robot on people's trust during a navigation task. In *Annual Conference Towards Autonomous Robotic Systems* (pp. 349-361). Springer, Cham.

[2] Kirby, R., Simmons, R., & Forlizzi, J. (2009, September). Companion: A constraint-optimizing method for person-acceptable navigation. In *RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication* (pp. 607-612). IEEE.

[3] Dautenhahn, K., Woods, S., Kaouri, C., Walters, M. L., Koay, K. L., & Werry, I. (2005, August). What is a robot companion-friend, assistant or butler?. In *2005 IEEE/RSJ international conference on intelligent robots and systems* (pp. 1192-1197). IEEE.

[4] Ljungblad, S., Kotrbova, J., Jacobsson, M., Cramer, H., & Niechwiadowicz, K. (2012, February). Hospital robot at work: something alien or an intelligent colleague?. In *Proceedings of the ACM 2012 conference on computer supported cooperative work* (pp. 177-186).

[5] Bethel, C. L., & Murphy, R. R. (2007). Survey of non-facial/non-verbal affective expressions for appearance-constrained robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(1), 83-92.

[6] Helbing, D., & Molnar, P. (1995). Social force model for pedestrian dynamics. *Physical review E*, *51*(5), 4282.

[7] Zanlungo, F., Ikeda, T., & Kanda, T. (2011). Social force model with explicit collision prediction. *EPL (Europhysics Letters)*, *93*(6), 68005.

[8] Ferrer, G., Garrell, A., & Sanfeliu, A. (2013, November). Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1688-1694). IEEE.

[9] Vasquez, D., Okal, B., & Arras, K. O. (2014, September). Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1341-1346). IEEE.

[10] Chen, Y. F., Everett, M., Liu, M., & How, J. P. (2017, September). Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1343-1350). IEEE.

[11] Trinh, T. Q., Schroeter, C., Kessler, J., & Gross, H. M. (2015, October). "Go Ahead, Please": Recognition and Resolution of Conflict Situations in Narrow Passages for Polite Mobile Robot Navigation. In *International Conference on Social Robotics* (pp. 643-653). Springer, Cham.

[12] Vega, A., Manso, L. J., Cintas, R., & Núñez, P. (2018, November). Planning Human-Robot Interaction for Social Navigation in Crowded Environments. In *Workshop of Physical Agents* (pp. 195-208). Springer, Cham.

[13] Perera, V., Pereira, T., Connell, J., & Veloso, M. (2017). Setting up pepper for autonomous navigation and personalized interaction with users. *arXiv preprint arXiv:1704.04797*.

[14] Suddrey, G., Jacobson, A., & Ward, B. (2018). Enabling a pepper robot to provide automated and interactive tours of a robotics laboratory. *arXiv preprint arXiv:1804.03288*.

[15] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[16] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[17] Danelljan, M., Häger, G., Khan, F., & Felsberg, M. (2014). Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press.

[18] Jones, S.E. and Yarbrough, A.E., 1985. A naturalistic study of the meanings of touch. *Communications Monographs*, *52*(1), pp.19-56.

[19] Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., & Sheikh, Y. (2018). OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *arXiv preprint arXiv:1812.08008*.

[20] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[21] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Duerig, T. and Ferrari, V., 2018. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*.

[22] Vittorio, A., 2018. Toolkit to download and visualize single or multiple classes from the huge Open Images v4 dataset. Github, GitHub repository, url: https://github.com/EscVM/OIDv4_ToolKit.

[23] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

[24] Afifi, M. (2019). 11K Hands: gender recognition and biometric identification using a large dataset of hand images. *Multimedia Tools and Applications*, *78*(15), 20835-20854.

[25] Bambach, S., Lee, S., Crandall, D. J., & Yu, C. (2015). Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1949-1957).

[26] Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., & Levine, S. (2018). Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*.

[27] Cully, A., Clune, J., Tarapore, D., & Mouret, J. B. (2015). Robots that can adapt like animals. *Nature*, *521*(7553), 503-507.

[28] Hill, A., Raffin, A., Ernestus, M. Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A.,Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y. Stable Baselines. 2018. GitHub. GitHub repository. url: https://github.com/hill-a/stable-baselines.

[29] Busy, M., & Caniot, M. (2019). qiBullet, a Bullet-based simulator for the Pepper and NAO robots. *arXiv preprint arXiv:1909.00779*.

[30] Qureshi, A. H., Nakamura, Y., Yoshikawa, Y., & Ishiguro, H. (2016, November). Robot gains social intelligence through multimodal deep reinforcement learning. In 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids) (pp. 745-751). IEEE.

[31] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[32] Mitka, E., Gasteratos, A., Kyriakoulis, N., & Mouroutsos, S. G. (2012). Safety certification requirements for domestic robots. *Safety science*, *50*(9), 1888-1897.

[33] Pacchierotti, E., Christensen, H. I., & Jensfelt, P. (2006). Embodied social interaction for service robots in hallway environments. In *Field and service robotics* (pp. 293-304). Springer, Berlin, Heidelberg.

[34] Tzafestas, S. G. (2018). Mobile robot control and navigation: A global overview. *Journal of Intelligent & Robotic Systems*, *91*(1), 35-58.

[35] Koay, K. L., Syrdal, D., Bormann, R., Saunders, J., Walters, M. L., & Dautenhahn, K. (2017, November). Initial design, implementation and technical evaluation of a context-aware proxemics planner for a social robot. In *International Conference on Social Robotics* (pp. 12-22). Springer, Cham.

Annex

Hand Detection Dataset

The images were downloaded using the OIDv4_ToolKit[**21**] with the following command:

```
python3 main.py downloader --classes "Human arm" "Human hand" --type_csv all --multiclasses
1 --limit 10000 -y
```

Hand Detection Training

The training was done following the instructions of the mentioned fork of YOLOv3.

The command to download the weights:

```
wget -c https://pjreddie.com/media/files/yolov3-tiny.weights
```

The configuration file needs to be created in the folder of Pytorch-v3 with the following commands:

```
$ cd config/                                # Navigate to config dir
$ bash create_custom_model.sh <num-classes> # Will create custom model 'yolov3-custom.cfg'
```

Classes:

Add class names to `data/custom/classes.names`. This file should have one row per class name.

Image Folder

Move the images of your dataset to `data/custom/images/`.

Annotation Folder

Move your annotations to `data/custom/labels/`. The data loader expects that the annotation file corresponding to the image `data/custom/images/train.jpg` has the path `data/custom/labels/train.txt`. Each row in the annotation file should define one bounding box, using the syntax `label_idx x_center y_center width height`. The coordinates should be scaled *[0, 1]*, and the `label_idx` should be zero-indexed and correspond to the row number of the class name in `data/custom/classes.names`.

Define Train and Validation Sets

In `data/custom/train.txt` and `data/custom/valid.txt`, add paths to images that will be used as train and validation data respectively.

Train

To train on the custom dataset run:

```
$ python3 train.py --model_def config/yolov3-custom.cfg --data_config
config/custom.data --pretrained_weights weights/yolov3-tiny #to train using the pre-
trained weights.
```