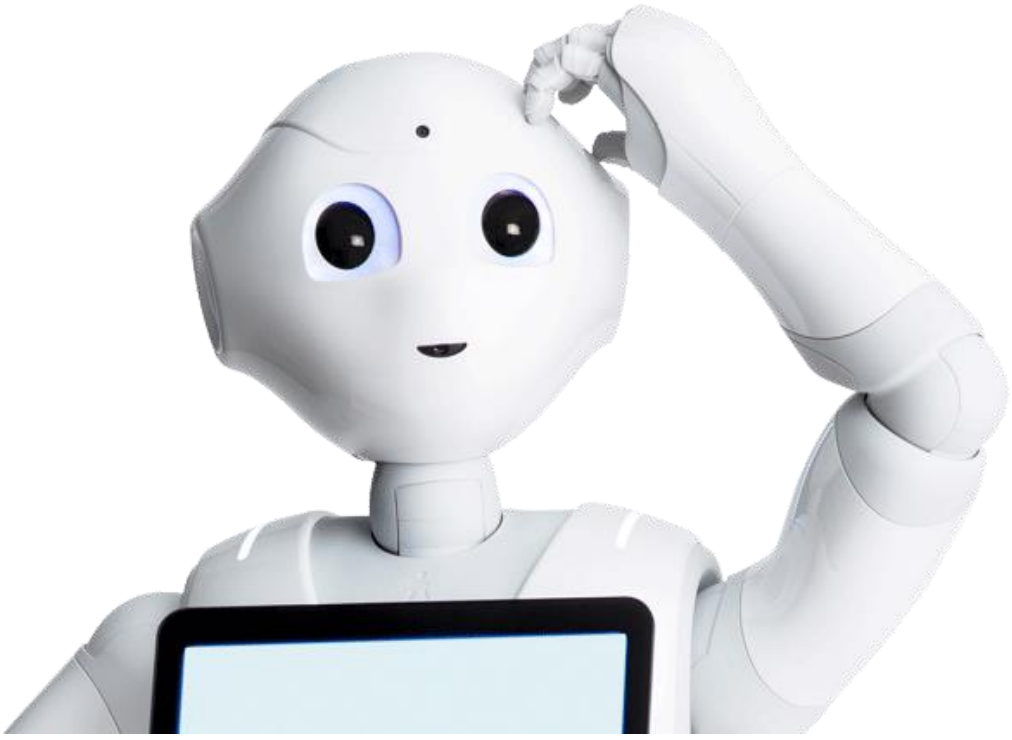
HRI Project:
A privacy enforcement program through the help of Pepper

Matteo Emanuele: 1912588

Pietro Manganelli Conforti: 1754825

Claudia Melis Tonti: 1888489

All the students contributed more or less equally to the project



Abstract

The general purpose of this project is to implement a Human-Robot interaction system which goal is to select a group of objects that the user may want to blur in order to hide them during videocalls. The importance of this task increased during the latter period when, due to the epidemic spreading of COVID-19 the worldwide population has been forced to use videocalls platforms whenever possible.
Through human robot interactions, dialogues and physical interactions, such problem can be tackled.

Introduction and Problem Statement

This project was developed starting from an idea: in the past years this covid situation really changed our way of living. We have spent most of our time in virtual meetings, and the privacy problems that raised from this were plenty. Showing the face of a minor age person to NDA agreement that were broken because of accidental events happened in direct streams or videocall, this problem crippled a majority of the people.
In this project we simulated a privacy enforcement program to remedy to such problem.
The goal of such a program is to gather a list of objects that a user does not want to show while being in a video call, blurring out from the video in real time sensitive objects and items that may be of personal concern.
Of course, this requires quite the gathering of information, tailoring every case to every user. Here is where Pepper comes into play: we decided to develop a Pepper-centric solution, coding the robot to help users participating in this program, guiding them and gathering the information that are needed.
As we have also discussed in class, a key element to Human Robot Interaction is the quality of the humanlike experience: users don't want to be freaked out by uncomfortable interaction, neither they want to lose time trying to make a machine understanding their words or ideas. Such solution we developed aimed to be the most efficient, humanlike and enjoyable possible. We will emphasize this matter later on in the report underlining some key aspect of the implementation as the usertesting we did, as well as the random casualty system we implemented to simulate real life situation in which the robot could find itself on.
We will also discuss the interactions we developed for the user, as well as an overall presentation of the decision tree.

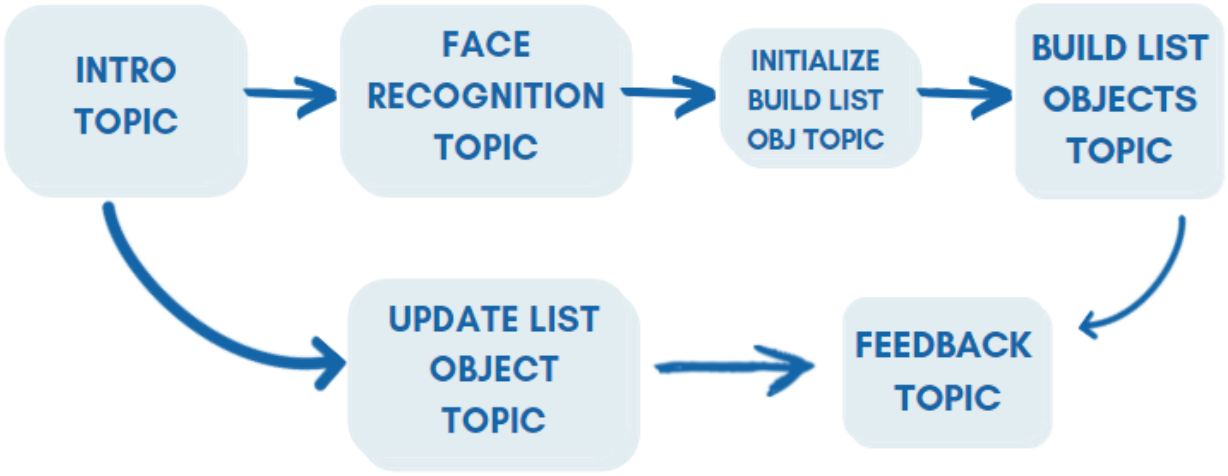
Technical details of the solution we implemented

We implemented the project using the Android Studio idle with the Pepper NAOqi SDK library to apply it to a virtual Pepper emulator. The NAOqi SDK is a tool of Android Studio which when activated allows the use of tools as Chat Topic files and the virtual 3D emulator of Pepper.
Everything was implemented in java and in the NAOqiSDK-unique language that characterize the ".top" files. It has its own syntax as well as its own functions and features. It is mostly used to develop chatbots. It is implemented using "concepts", which are basically list of words and/or entire sentences that are used as flag to be recognized by the robot while speaking with the user.
Generally speaking, the flow of the dialogue is carried over by "proposal" and user answers: a proposal is a sentence that Pepper says while perceiving something to be said to him that fell inside the umbrella of a concept associated to the proposal. The user's answers are instead only represented by the concept, and to each one there is something that pepper needs to say in response. With this logic and infrastructure, it is possible to build a tree like dialogue where each proposal has different answers to which Pepper has other answers to give to each one, and so on and so forth.
Our solution was composed by a total of 5 different topic files and a 2 different java files. These latter two are divided in the one containing the Main activity, taking care of the flow of the dialogue, while the other one contains the tablet implementation we created for one of the interaction that Pepper presents and that will be discussed later on.

Speaking: The decision tree

To implement the dialog, we used a tree-shaped structure of Chat Topics which explores every (or the majority of) possible user's answer.
The dialog starts with an introduction of Pepper to the user. We simulated an event where the user approaches the robot in a limited area around it. Then, Pepper makes a movement which purpose is emulating a face recognition: if the robot recognizes the user, then it welcomes the user back. This scenario is simulating the case in which the user has already been registered to the Privacy enforcement program, and maybe wants to update the information already given. In case the robot doesn't recognize the user's face it presents itself and asks if it can store the face, it just detected to sign it to the system. The simulation of the face recognitition is implemented by the value of an integer, facerecogn (0 if it detects it 1 if not).
After the registration or the login of the user the dialogue proceeds in the InitializeBuildListObj topic, where the robot asks which room the user wants to scan and other information needed to add the objects to the list or to build a new one. Then it finally proceeds to the last topic which is the buildListObj Chat Topic. In this last file the robot explains what it is going to do, asks if it has to repeat itself or to go on with the registration or else to stop the registration. In this

piece of the dialog, it is also managed the case where the robot can't complete the task because of the bad user's instructions.
These last scenarios are implemented using asynchronous functions, for example, to make the robot regain the attention of a distracted user, a timer has been set as limit of the robot waiting for an answer.
The last step of both the branches of the tree ends in the robot asking if the user wants to leave feedback about the experience.
Here is reported a high-level general representation of the dialogue flow:



The tablet

A second type of interaction we implemented is the interaction with Pepper's tablet. I our case the purpose of this interaction is to build or update the list of objects that the user wants to blur.
The tablet application is a simple interface dynamically built by adding items to a Listview. The interface contains also an "ok" button which task is closing the interface and returning an array of selected objects.
The array of available object which will be add dynamically to the list view, emulates the room scansion and the memory storage associated to each user.
In our example the array contains 4 objects and what the click of the "ok" button returns is an array of strings, which are the names of detected objects.
The tablet is turned on when the user decides to build his list of objects. When the user clicks the OK button then the application of the tab list will be turned off and then the dialogue goes on.
When the tablet is turned on it shows a clickable list of objects. The items in the list are highlighted with blue color if the objects are just in the array of selected objects associated to the user. The user can click on the list items to select and deselect the objects.

The use of the tablet is an asynchronous event, but it can't be used in this way, because the dialog should go back to the start once the interface is closed. To solve this issue, the tablet application has been forced to be synchronous making the dialog hold on from where it stopped. At the end of the use of Pepper on the tablet should be shown a QR code which forwards to a form web page to make the user give feedback of his/her experience.

Animations

Another type of interaction that we implemented are the animations. While the dialogue goes by, the robot performs a set of animations implemented by the Pepper SDK library.
Generally speaking, animations on the SDK library can be implemented in three ways: through the usage of two major function called LookAt() and MoveAt(); through the usage of a generic function called Animate(), or with the usage of already built-in animations.
In the first case the results that you can achieve are very limited, and this is because those two functions are used only for moving the head of the robot or the body. They are based on some concepts common to the interactive graphics field like geometric transformations of the cameras of the agent to find the point in the World Reference Frame that needs to be looked at or approached with a movement. Such implementation limiting the purpose of the project so we decided to discard it.
About the second case, we can say without hesitation that it is the most powerful method of all. The not-so-small problem though is that to use the animate function you need to set the joint velocities and positions of each joint of pepper every time, creating key frames to build the whole animation. This is basically how animations are created nowadays, and it is a tremendous effort and also out of the scope of the project. Moreover, the esthetic of such movements was not so pleasant given the fact that create animations is something that it's very technical. Given all these reasons, we decided to discard these options too.

This leads us to our final choice of implementation: using the already built in animation provided by the library. They are easy to use, very nice and already created for different scenarios.
The robot can make gestures as “hello”, “show the tablet”, “take picture” and also some face expressions. Another animation is “walk run”, which simulates the event in which the robot has to follow user’s instructions when he/she describes the trajectory to walk to make Pepper go in the desired room.
The animations are implemented in the project as Qianim files and they require in the java code to be launched asynchronously with an Executor, which is represented by a class and it’s basically something that is associated to the chatbot and when called in the .top file, it executes a piece of code (a)synchronously.

Feedback Experience Form

To conclude, the feedback experience form is the cherry on top we developed to conclude the user experience. It’s based on the anthropomorphism’s Godspeed questionnaire we discussed in class:

GODSPEED I: ANTHROPOMORPHISM

Please rate your impression of the robot on these scales:

以下のスケールに基づいてこのロボットの印象を評価してください。

Fake 偽物のような	1	2	3	4	5	Natural 自然な
Machinelike 機械的	1	2	3	4	5	Humanlike 人間的
Unconscious 意識を持たない	1	2	3	4	5	Conscious 意識を持っている
Artificial 人工的	1	2	3	4	5	Lifelike 生物的
Moving rigidly ぎこちない動き	1	2	3	4	5	Moving elegantly洗練された動き

It is a simple google form that will be opened through a QR code that will be shown on the pepper’s tablet if the user agrees at leaving feedback.
We used the same exact questions above reported, asking the user to leave, for each one, a vote that went from 1 to 5, where the edges of this range expresses two opposing adjectives for the question in consideration.

Feedbacks are extremely important for developing human robot interactions experiences, since the final goal is the user perception of what happened while the interaction was taking place. We simulated this gathering of information to further improve the application given the opinion of the users who tested it.

Here we report the QR code that can be scanned to participate to the questionnaire:

Talking about usertesting, we were capable of refining and further improving the capability of pepper understanding any answer. This was simply achieved asking to people around us to talk with pepper and repeat the experience over and over in order to add and expand the concepts with every possible word or sentence that may be of help for Pepper.
We arranged the answers from the users into an excel file. Here are reported some of the questionnaire results delivered by the users:

4	3	3	5	4
4	2	3	4	2
4	2	2	4	5
2	5	5	5	5
4	2	2	4	4
2	2	3	2	5
5	5	5	5	5
1	2	4	3	5
4	5	3	4	5
5	5	3	3	1
4	2	4	3	5

The rows that present a lower overall grade represents also the firsts feedback we obtained. Every time we tried to change something in order to improve the experience for the users. The numbers seem to show that such refining process actually worked.

Random Casualty system

Among all the implementations we developed and that we have already discussed in this report, one that we wanted to point out more precisely was the so-called Random Casualty System we developed.
The idea that moved the creation of such thing was that we wanted to make the experience as realistic as possible, which means that sometimes it may happen that the user, while talking to pepper, decide to lose the attention and leave. Or maybe, the scanning of the face from Pepper does not go as planned, maybe because the user moved during the process for instance. All these things should be takes into account to achieve a work of a better quality. Sadly, this was not possible to be implemented with a real robot, which means that if we wanted to simulate such scenarios, we needed to handle such cases through logic and code implementation, which is what we decided to do.
We have implemented a system that elaborate 5 different variables:

```
var_stop1.async().setValue("0"); // group event in the beginning
var_stop2.async().setValue("0"); // user loosing interest
var_stop3.async().setValue("0"); // Pepper's path is blocked
var_stop4.async().setValue("0"); // failing face scanning
variable_face.async().setValue("0"); //recognition of the user
```

These variables has the following roles, in order of listing as above:

1. Simulate a group of people approaching Pepper instead of a single user;
2. The user lose interest in talking with pepper and decide to leave in the middle of the speech;
3. While walking towards the point of interest for the scan for building the list of objects, Pepper find himself on a blocked path, without the possibility to proceed;
4. Pepper fails to scan the face of the new user to add into the privacy enforcement program;
5. Variable that we used to handle the event of the user been already registered to the program.

Setting these variables, we were capable of generating branches for the decision tree of the robot, guiding the dialogue accordingly to the situation. This required some fine tuning of the overall flow, but at the end in our opinion was successfully giving a major depth to the experience.
As we can see from the picture, all the variable are initialized deterministically. This is because the function RandomCasualtySystem() took as parameters a string that if equal to “det”, would have set the variables to specific values. This was used for debugging and testing.
While in general, the System handle the assignment of such variables with random threshold that changes at every execution of the code. This generates different scenarios all the time.

Moreover, one last pinpoint is about the var_stop2. Such variable, as we have said, was used to simulate the user losing interest. We implemented such event in different region of the dialogue, taking care of the threshold accordingly checking in which range of values such threshold feel. Doing this we were capable of achieving a total of 8 different casualty scenarios, that if we consider all the possible combination of them, create a various number of possible experiences at every run of the code.

Results

The resulting system is a robot which can properly communicate with the user in at least 4 different ways: language, gestures, the interaction with the tablet and finally a feedback forum made to improve the performance of the application.
The entire application has been structured to make impossible for the user not knowing what to say and what to do. In fact, the robot itself sometimes suggests the statements that the user should use to ask some commanding, if the user is distracted and stops talking the robot tries to regain his/her attention, to incorrect statements it asks to repeat.
These abilities of the robot make possible to have a random causality system which manages every possible choice of the user
The interface of the tablet is very simple to use, so it's very difficult to not be able to use it correctly.

Conclusions

Thanks to the development of this project, we are now able to use all the possible choices to implement a good interaction between a human and a generic software. The great challenge was using the NAOqi SDK, which actually is out of production and which community has been closed. Because of this the use of the only documentation sometimes was not enough. Also, the use of an IDLE as Android Studio and the Java programming language was the only way to make the Pepper emulator work because the Python alternative was no longer supported, not even properly documented.
As mentioned in the paragraph of Related Works, additional functionalities that can be implemented to make the system work better are the implementation of a semantic map's development to make the robot understand the environment and so recognize the rooms and the objects the user may want to consider. Then an application to understand the natural language can be added to improve the dialog with the user. Finally, an application which can recognize user's gestures can make the robot able to add to the list of objects items that it didn't recognize in the semantic map, or also the user can decide to make the robot follow him/her to guide it to the place of interest.

Related works & References

- *Robot semantic mapping through human activity recognition*: A wearable sensing and computing approach: in this paper a method to build a semantic map is explored. The connection with our project is a way use a robot(Pepper in this case) to recognizes the environment and so to make it able to make a scan of the rooms and the objects present in our place of interest. The gathering of information through Human-robot interaction are pushing ahead the field of the data collecting. Such implementation could be an extension of what we have developed for this 3 credit project.
- *Recursive Neural Network Based Semantic Navigation of an Autonomous Mobile Robot through Understanding Human Verbal Instructions*: this paper explore the task of implementing a semantic map by using a natural language processing application to recognize human's commanding.
- *Real-time 3D Hand Gesture Interaction with a Robot for Understanding Directions from Humans*: this paper could be useful to improve the interaction between human and robot by adding the ability to recognize human gestures to make Pepper able to add some pointed objects that it didn't recognize in a previous scan.
- https://developer.softbankrobotics.com/pepper-naoqi-25/naoqi-developer-guide/sdks
- http://doc.aldebaran.com/2-5/home_pepper.html
- https://developer.softbankrobotics.com/pepper-qisdk/api/perceptions/tutorials

Sources

- The link to our project's repository is: https://github.com/Allyjuke96/HRI_project/tree/master

- You can watch our presentation video here: https://www.youtube.com/watch?v=y73mhuGjwt4