Social Interaction with Pepper Robot using Human Emotion Recognition

Author: Francesco Starna. University ID: 1613660.

Abstract

Humanoid robots are becoming more and more a thing close to reality rather than just an image of the future. This is thanks to huge recent technology improvements, especially in the field of Artificial Intelligence (AI). Thanks to AI it is possible to build autonomous systems that are able to interact with people, helping them in different manners. In the scenario of Human-Robot Interaction (HRI) I propose a small simulation project, in which the interaction happens according to emotions.

Introduction

HRI is lately taking place in research, and very soon it won't be strange to meet and interact with some robots in social environments, such as shopping malls, schools, sports (Robocup), hospitals, offices, saloons and so on. The motivations behind this upcoming innovation are mostly related to the aim of working together to accomplish a goal. Indeed, robots can be designed and programmed ad hoc for many different tasks. Moreover, the research field on HRI is growing more and more because these systems integrate a lot of technologies, such as Robotics, AI, Computer Vision, Machine Learning, Natural Language Processing, Design, etc. and so it involves a lot of different experts.

In this regard, for the university course delivered by professor Iocchi, Human-Robot Interaction, I developed a project focusing on social interaction with people. Due to covid19 restrictions unfortunately, it was not possible to access lab resources and robots at university, so I used a virtual environment provided by SoftBank Robotics, which makes it possible to interact with the Pepper Robot.
Pepper is a humanoid robot, purposely designed to allow social interaction. The humanoid morphology is fundamental because during an interaction people expect to find a similar one, who is able to behave according to some social rules and is capable of different interaction modalities [Fong et al. 2003]. In fact Pepper can exploit its several sensors (rbg cameras, depth camera, microphone, speakers, touch sensors, tablet) to reach a high level of "sociality", in the sense of social skills. Thanks to human-like actuators and vision sensors it is also possible to accompany verbal communication to enable social signal processing [Vinciarelli. et al 2008], and further, with the use of touch sensors and the tablet, multi-modal interaction can be designed, handling multiple modalities [Iocchi et al. 2015].

With all these features in mind, I developed my project, especially focusing on emotions; it is very common for people to show emotions and capture them during a conversation, and most of the time this is a key point in order to feel empathy with others. For this reason I implemented within the Pepper Robot an emotion recognition system, trying to give to a conversation more awareness about feelings, and reacting according to them.

Related Works

There are a few works in HRI scenarios related to emotions, not necessarily with humanoid robots. In [Eirini Malliaraki 2018] they focused on human-drone interaction, in a collocated space (indoor or outdoor), in which the aim is to deal with non-anthropomorphism and how to exploit the motion of drones. The video from the front camera of the drone is captured and fed to an emotion recognition system, which provides the most probable emotion of the human who is interacting with the drone. Then according to that, the drone reacts with what they call “drone choreographies”, which are animation scripts based on the drone’s motion. They finally suggest a mapping between 5 emotions (Anger, Happiness, Sadness, Surprise, Fear) and 5 drone’s physical properties (Speed, Rotation, Altitude, Distance, Special movement). Their work is actually built upon a previous one [Cauchard et al. 2016] in which they encoded a range of personality models (the Exhausted, the Anti-Social, the Adventurer Hero, and the Sneaky Spy) that affect the drone’s flight path.

Clearly the motivation behind these research projects was being pioneers in creating a framework, thanks to which it is possible to “install” some personalities to a robot (in this case a drone, for simplicity in programming motion) which fits as much as possible the person or the people who it is interacting with. Thus, under these motivations and insights, I found the inspiration for my project, through these readings and related articles.

Architecture

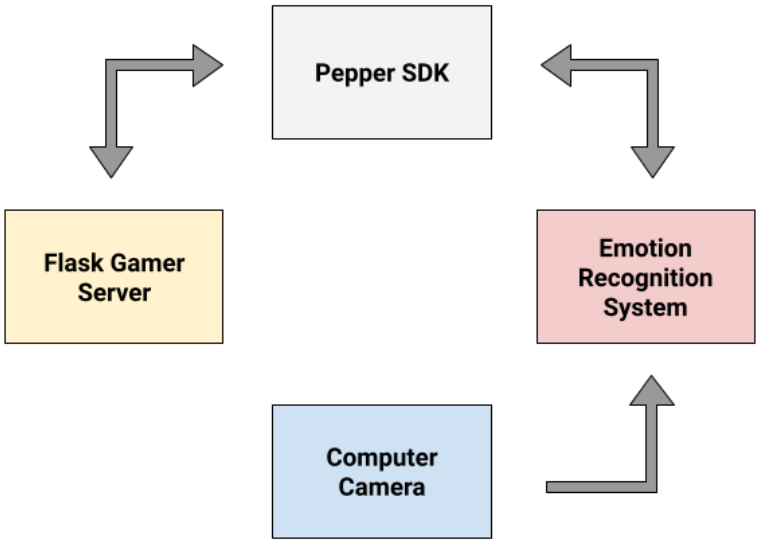

Fig1. Components of the project architecture. The arrows represent input and output of the modules.

The architecture includes different systems, due to the simulation environment which does not provide sensors and devices. I used the camera of the computer, for simulating the robot front camera, a Flask server, for simulating a game on the tablet of the Pepper robot, and a standalone emotion recognition system. A graphical representation can be seen in **fig1**.

The Pepper SDK was provided in a docker image, running in Ubuntu 16, installed in a virtual machine. In order to see the simulation, I also installed the proprietary software Choreograph. Unfortunately, the simulation environment does not provide the use of the NAOqi Vision module, useful to control the Pepper robot's 2d cameras. Therefore, to include vision, I make the camera of the computer working together with the Pepper robot to simulate a front camera, placed on the head. In order to do this and develop the others architecture modules, I have changed the Dockerfile, including some libraries for reading the camera, load a machine learning (ML) model, and write a simple web server, specifically:

```
27:   RUN   apt-get install libcanberra-gtk-module libcanberra-gtk3-module
75:   RUN   pip install --user tensorflow requests flask
```

in which the first ones, serve the docker image in order to create a GUI in which the camera streaming will appear, and the second ones, provide the Google's framework for reading and executing ML models, and web tools for simulating a game on the Pepper robot's tablet.

Pepper SDK

Programming the Pepper robot is done with the NAOqi SDK, which is the proprietary framework of SoftBank Robotics, distributed in python 2.7, which allows high level programming on several robots. NAOqi provides different services intended for interacting with a certain module of the robot, in particular I used:

- `ALMemory`. It is a centralized memory used to store and retrieve all key information related to the hardware configuration of the Pepper robot. It provides information about the current state of actuators and the sensors. Due to the use of a simulator, this service is mainly used to simulate automated speech recognition, saving text from the user side, and retrieving it from the robot side.

- `ALMotion`. It contains methods that allow manipulation of joint stiffness, joint angles, and interpolation time. Walking and approaching it is not possible in the simulation environment, but only joint movements are.

- `ALAnimatedSpeech`. It makes the robot talk in an expressive way. The simulator does not allow sound, and the speech text of the robot appears in a dialog cloud.

- `ALTouch`. It creates a listener that fires a `TouchChanged` event whenever the Pepper robot is touched. Also in this case, touching can be simulated by setting with the `AlMemory` service the right joint value.

The connection to the robot is done by tcp protocol to the local address `127.0.0.1` and port number `9559` (default port of simulator), after which the application is started and all the needed services launched. The class `Pepper.py` abstracts any service methods for connection, speech, motion, touch reaction, memory handling, and exposes simple methods used in the main interaction process. An event handler thread is launched in order to capture human speeches and touches.

Face Emotion Recognition System

The Face Emotion Recognition (FER) system is based on the OpenCV computer vision library, to capture the image from the computer camera, and two pre-trained ML models running on OpenCV and Tensorflow. To make the system work independently from the main code execution, consisting of the interaction between a human and the Pepper robot, I made some changes to the system, enabling the FER system running as a thread. A main loop captures frame by frame from the input camera. At each frame the following operations are performed:

1. Face detection with Haar Cascade Classifier
2. Emotion recognition with pre-trained FER model
3. Display the detected face with the relative emotion

The Haar Cascade Classifier [Viola et. al 2001] uses edges and lines detection with a haar feature, which is a kernel with a darker area (0 values) and a lighter area (1 values). The kernel traverses the whole image from the top left to the bottom right, detecting face features. A set of haar features represents some face's characteristics, such as eyebrows, eyes, nose, mouth, etc. In order to classify a set of haar features as a face, a cascade classification is performed; if a haar feature fails, we can assume that there is no face present in the image frame, otherwise the next haar feature is tested, until all the set has been considered. OpenCV library has a built-in model and provides the weights to set up the classifier.

The FER model is kindly provided by [Serengil 2017]. It consists of a Convolutional Neural Network (CNN) of three convolutional layers and a fully connected layer which applies dropout and ReLU activation function, before applying softmax to obtain the output probabilities. The model was pre-trained in Ferc2013 Dataset, consisting of 48x48 pixel grayscale images of faces, handled during a challenge in Representation Learning ICML 2013 Workshop, whose task was to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). In my implementation the interested emotions are happy, sad and angry, and therefore are the only ones detected. The model was implemented using Tensorflow, together with the pre-trained weights provided by Serengil's Github.

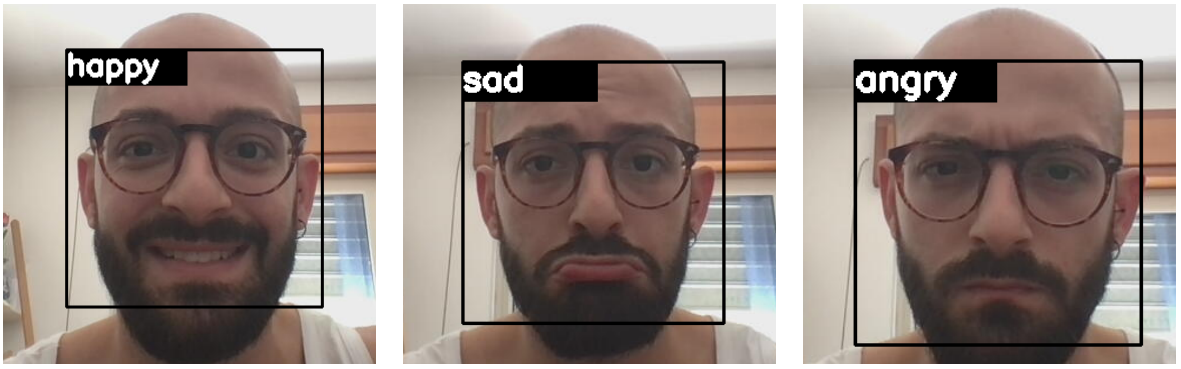

Fig2. Examples of face detection and emotion recognition from the computer camera.

Once the emotion is detected, a rectangle representing the face and a text with the highest probable emotion is shown in the camera frame, with the purpose of simulating what the Pepper robot is “seeing” at each frame. The FER system also implements methods for starting and stopping the detection, waiting the first detection, and displaying the text of human speeches. All these features are included in the class `fer.py`, which in turn is imported and can be used as a module by the class `Pepper.py`. **Fig2**, shows some examples of random frames shown during face detection and emotion recognition phase.

Flask Game Server

To simulate the tablet screen I used a Flask web server, which is written in the `GameServer.py` module, and instantiated into `Pepper.py`, so it directly interacts with the robot’s execution flow. Flask is a license free web micro-framework simple and intuitive to use, suitable for simulating a trivial question answer game. It provides RESTful API and it is based on the Jinja2 templating engine, which allows writing special placeholders and code similar to python.

When instantiating a GameServer, the init method takes some questions, answers (true or false), the player name and a callback function to execute when the user answers a question. It also sets the score value to zero. In order to display the game on the web, I prepared three different templates to be rendered by the Flask server:

- `index.html`. It displays a welcome page to the player and a Start button. When the player pushes the Start button the game starts and the Flask server renders the game template.

- `game.html`. It is a base template for any questions to be displayed. It has two buttons, True and False, to be clicked for answering the question. A javascript function `get_robot` starts to execute, at each interval of 1000 milliseconds, an Ajax GET request which waits for a response by the Pepper robot. After the user has answered true or false the flask server calls the callback function, with input value “true” or “false”. In addition, a POST request is sent to the Flask server which renders the same game template with the next question, until all the questions have been shown and instead the end template is rendered. Every time the player guesses the question, the score value is increased by one.

- `end.html`. When the question game is terminated, it displays the score, which is equivalent to the number of correct answers divided by the number of questions.

Questions and answers are stored in lists at the beginning of the `Pepper.py` module, and refer to simple general robot characteristics and history. The callback function and the Ajax GET request represent the links between the Pepper robot and the Flask server; the callback function coordinates a robot motion behavior, depending on the correctness of the answer, while the request waits for a response from the Pepper robot which informs the Flask game server that the player preferred to answer interacting with it (the robot), and so to render the next question and update the score. This basically introduces redundancy in the communication channel, and therefore it is a small evidence of multi-modal interaction.

Experiments

In this section I explain what the goal of the interaction is, how the main application moves forward from the beginning of an interaction to its end, and the schema followed by the Pepper robot to interact with a person. Some of the interaction details refer to the video `project_demonstration.mp4`, citing the time of the exact portion of it.

Objective

The goal of the interaction can be broken down into three sub-goals:

1. Approaching a person: using the simulator was not so helpful, because even though the motion can be simulated, the environment is abstract and infinite, and so for instance it was not possible to approach a person based on proximity. However, using the FER system, I implemented a function `wait_detection` which takes as input another function to be executed until a first detection happens. In a real scenario the approaching phase could have been done using both proximity and eye contact with a person.

2. Obtaining information about the person and understanding his/her emotional state: when the interaction between Pepper and a person starts, the aim is to empathize with him/her, which is fundamental for the robot in order to react in a proper way. The whole project's aim actually emphasizes the importance of handling an interaction according to feelings, right how humans usually do, and this turns a cold and senseless conversation into a warm and pleasant one. Not surprisingly the terms feelingless and emotionless are used when referring to a person who behaves like a robot. Thus, to cancel this saying, or more likely showing that it is false, Pepper asks and recognizes feelings, before deciding what to do.

3. Successfully accomplishing the interaction with the person: for the purpose of a good interaction, Pepper has to decide a proper reaction according to the emotion recognized and also the information obtained. The accomplishment consists of terminating the interaction with the person and be sure he/she enjoyed it, or helped him/her at that time. Pepper's can't be completely aware about succeeding, a thing that is only possible after many feedbacks by the end of every interaction, but it will definitely propose the right thing according to an emotion-behavior mapping, which can be seen in **Table1**.

Reaching the goal of an interaction, however, does not always depend on the Pepper robot and its architecture. In fact, the following mapping template:

> **IF** emotion is ***x*** **THEN** react according to ***y***

for some x, y belonging to the sets of possible emotions X and reactions Y, it is not appropriate for all the people. Specifically, this depends on the personality of a person, which influences the interaction in several ways. It is certainly possible to extend the mapping also to different personalities, which is not a topic covered in my project.

Emotion	Truthfulness	Robot Behavior	Motivations
Happiness	True	Excited	Sharing happiness and be loved back is what a human usually expects
	False	Playful	Lying and saying the exact opposite of a positive aspect while smiling, usually means trying to be funny. Showing that you enjoyed is rewarding
Sadness	True	Thoughtful	When a person is sad and shares it with you, he/she usually expects you to be nice and cheer him/her up
	False	Distant	When a person is sad but does not share it you why, it usually means that he/she wants to be left in peace and you should not be pushy
Anger	True	Peaceful	Being angry is usually handled with a calm and proactive approach
	False	Distant	When a person is angry but does not share it with you, it usually means that he/she wants to be left in peace and you should not be pushy

Table1. In the first column there is the emotion detected by the FER system, while in the second column we have True or False depending on whether the person who is interacting said the truth or not. For each pair of emotion-truthfulness we have the corresponding robot behavior and the motivation behind the choice.

Interaction

Once the goal of a single interaction is established we can describe the interaction schema which Pepper follows in order to accomplish it. Before going into details let's first describe the interaction with a simple schema. In the following the mentioned functions are described in **Table2**.

INTERACTION SCHEMA

```
# START
1: Pepper detects Person and says Hi
2: Person answers <name>
3: Pepper asks for feeling and detects <emotion>
4: Person answer the <truth> (true or false) or not

#detected emotion and told emotion are equal. Person said the truth
5: IF <truth> is true:
6:    IF <emotion> is 'sad' THEN Pepper says something funny
7:    ELSE IF <emotion> is 'angry' THEN Pepper proposes a game
8:    ELSE IF <emotion> is 'happy' THEN Pepper dances

#detected emotion and told emotion are different. Person lied
9: ELSE IF <truth> is false:
10:   IF <emotion> is 'angry' or 'sad' THEN Pepper asks again

11:        IF Person answers true <emotion> THEN      go back to
                true if else statement

12:        ELSE IF Person answers false <emotion> THEN Pepper says
                 sorry and goodbye

13:   ELSE IF <emotion> is 'happy' THEN Pepper laughs
# END
```

ASYNCHRONOUS EVENTS

★ Whenever the head or hands are touched Pepper reacts

`1`: The application starts with Pepper looking for someone to approach. The function `wait_detection` with input `head_scan` is executed until the camera recognizes a human face . As soon as a face is detected Pepper says hi to the person asking for the name. The function `say` is called every time Pepper “speaks”. [video 0:14-0:18]

`2`: While the person answers through `human_say.py` script, Pepper waits with the function `wait_answer`, and then the function `get_name_fa` finds the name inside the sentence. Pepper stores the name and says it is nice to meet him/her. While `wait_answer` is iterating, it calls `random_say` after each interval of 10 seconds, selecting randomly from a list of similar phrases for expressing misunderstanding. [video 0:18-0:30]

`3-4`: At this point we get into the core of the interaction, in which Pepper asks for feelings. While the person is writing the answer, Pepper starts its FER system and starts detecting the emotion from the face with `start_detection`. When Pepper gets the answer with `get_emotion_fa`, it also stops the FER system with `stop_detection` and gets the emotion detected with `get_emotion`. [video 0:37-0:47]

`5`: Pepper now has enough information about the person to understand if he/she is lying or telling the truth. After comparing the emotion detected with the emotion told, Pepper makes a choice according to the result.

`6`: If the person is telling the truth and the emotion detected is “sad”, Pepper’s behavior becomes thoughtful and it nicely tries to cheer him/her up by telling a funny thing. To this purpose the function `random_say` with argument “funny” is called, which selects randomly from a list of funny statements and jokes about robots. After the person’s reaction, Pepper says goodbye and the interaction ends. [video 0:32-1:03]

`7`: If the person is telling the truth and the emotion detected is “angry”, Pepper’s behavior becomes calm and proactive and it tries to calm him/her down by proposing a game. The person has the choice to accept or reject. If the person rejects, the interaction ends, otherwise the function `game` starts the Flask game server, and Pepper explains the rules of the question game. The person has also the choice of answering by (simulating) touching Pepper’s hands or (web page) tablet. While it’s easy to answer on the tablet by clicking the True or False buttons, Pepper explains that if its right hand is touched, the answer is “true”, if instead its left hand is touched, the answer is “false”.

Every time the person answers a question, Pepper comments positively or negatively depending on the correctness of the answer, selecting randomly a comment from two lists with the function `random_say`. The latter is the callback function given to connect the Flask game server to the Pepper robot. When the person chooses to answer by touching Pepper’s hands, the function `look_hand` is called and a response with the content “true” or “false” is sent to the Flask game server. This connects the Pepper robot with the Flask game server. When the questions are terminated, the game ends, showing the score on the tablet, and Pepper says goodbye to the person. The interaction ends. [video 1:44-3:28]

`8`: If the person is telling the truth and the emotion detected is “happy”, Pepper’s behavior becomes enthusiastic and it shares the happiness with the person inviting him/her to follow the dance, by calling the function `dance`. After the dance is finished, Pepper says goodbye and the interaction ends. [video 3:29-4:03]

`10`: If the person is lying about his/her emotional state, and the detected emotion is “angry” or “sad”, Pepper understands that the person is intentionally hiding his/her real emotion; at the beginning it tries to ask again about the feeling and waits for the person’s answer.
[video 1:27-1:42]
`11`: If the person opens up to speak, and reveals his/her real emotional state, Pepper goes back to the cases when the person is telling the truth.
`12`: If instead the person continues hiding his emotional state, Pepper’s behavior becomes distant and leaves the person alone, apologizing for being invasive, and saying goodbye. The interaction ends.

`13`: If the person is lying about his/her emotional state, and the detected emotion is “happy”, Pepper understands that it is a joke, its behavior becomes playful and he reacts by laughing in order to demonstrate that he understood the joke. Then it says goodbye and the interaction ends. [video 1:04-1:26]

★ As regards asynchronous events, whatever the interaction time, whenever the person touches the hands or the head of Pepper, it react exclaiming:

“Oh you touch my `<joint>`!” where `joint` is in `{Head, R-Hand, L-Hand}`

showing surprise, and running a motion behavior which is `touch_head` or `look_hand` depending on the joint touched. This is done subscribing to the event `TouchChanged` through the `AlMemory` service, and thanks to the `ALTouch` service which provides the method `getStatus` for checking if the joints values have been changed. The touch simulation is done with the `touch_sim.py` script. The touching reactions are not present in the video demonstration, but some significant frames can be seen in **Fig3**.

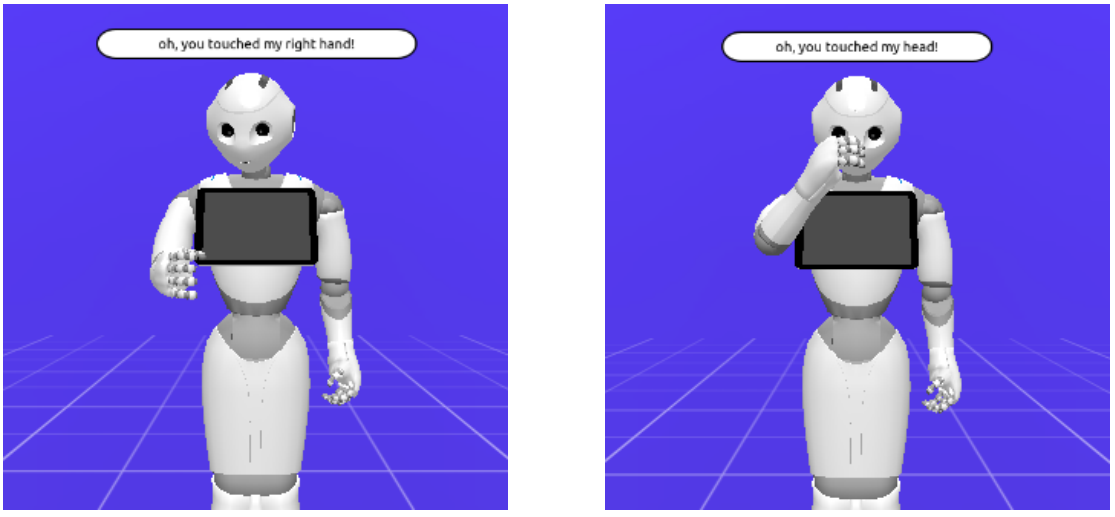

Fig3. Pepper’s motion behaviors when the person simulates touching its joints.

Function	Module	Service	Explanation
say	pepper.py	ALAnimatedSpeech	It takes a <text> argument to be said by the Pepper robot, using a contextual speech configuration
random_say	pepper.py	ALAnimatedSpeech	It takes an argument <context>, which looks for the relative list of sentences and selects one in a random way. The lists are in pepper.py.
fake_asr	pepper.py	ALMemory	Simulates speech recognition by reading on memory at the ASR key
human_say	human_say.py	ALMemory	Simulate speech by writing on memory at the ASR key
wait_answer	pepper.py	ALMemory	It loops until fake_asr returns a valid text. After each <interval> argument it calls random_say
touch_sim	touch_sim.py	ALMemory	It takes a <sensor> argument on which simulate touching , by writing on the memory at the sensor key
get_name_fa	application.py	None	Takes a sentence as input and return the name if it is there
get_emotion_fa	application.py	None	Takes a sentence as input and return the emotion if it is the emotion list
wait_detection	fer.py	cv2	It is a blocking function that calls a <function> argument infinitely until a detection flag is set to True
start_detection	fer.py	cv2	Starts the fer model to predict on each camera frame in order to output the detected emotion. It takes a <frame> argument which is the number of frames on which the model finds the most detected emotion
stop_detection	fer.py	cv2	Stops the face emotion recognition model
get_emotion	fer.py	cv2	Returns the last most detected emotion
head_scan	pepper.py	AlMotion	Pepper makes a 180 degrees look with the head, looking for someone
touch_head	pepper.py	ALTouch, AlMotion	Pepper follows a sequence of motions for touching its head and lowering it a bit
look_hand	pepper.py	ALTouch, AlMotion	Pepper follows a sequence of motions for looking at it right or left hand, depending on the <hand> argument
dance	pepper.py	AlMotion	Pepper follows a complex sequence of motions to simulate a dance
game	pepper.py	AlMotion	Pepper inits the game server for playing
laugh	pepper.py	AlMotion	Pepper follows a sequence of head motions to simulate a laugh

Table2. Description of the main functions developed with the respective service/library used and the module of belonging.

Conclusions

Developing this project was quite challenging, due to the lack of important functionalities not offered by the simulation platform, but also a huge learning experience thanks to the process of putting different systems together to cooperate at the end of human-robot interaction. Moreover, reading papers and articles I now have a clearer idea on the importance and potentiality of HRI in the future.

I believe the proposed project is a valid starting point for the HRI community, hoping that could inspire bigger works on developing personality models to install on humanoid robots which adapt to the human personality at interaction time. However it presents several weaknesses to be strengthened and improvements to be implemented over time.

Weaknesses

- Trivial dialog system: the conversations happen based on a timed function which listens and checks if an answer has been given at each time interval. If for instance the interval is 10 seconds and the person answers immediately to Pepper, he/she has to wait about 9 seconds in order to get feedback from Pepper. Moreover, the pre-decided sentences are common, but not appropriate for every interaction and for every emotional state of the person. It may happen that someone could find the phrases annoying or invasive.

- Total reliability on face emotion recognition system: the interaction flow starts and ends thanks to the FER system. If for any reason the vision sensor (the computer camera in this case) does not work properly, or the person somehow hides his face partially or totally, the system could fail or worse could return a wrong prediction. If the system fails, the interaction could not go on, and remains blocked, while instead if the emotion detected is wrong, Pepper may incur in unpleasant situations in which the person feels uncomfortable due to an unexpected reaction.

Improvements

- Advanced multi-modal interaction: it is the key point for solving the previous weaknesses. In order to improve dialogs one could implement interaction templates, in which conditions and actions are well defined, and can be handled in a more efficient way. Also, using `ALDialog` service provides Pepper with conversational skills by using a list of rules written and categorized in an appropriate way. Another important aspect of multi-modal interaction is robustness to failure; this means for instance that the case of FER failure has to be handled. One may for instance decide to have a “normal” conversation with the person if the emotion is not detected or trust the feelings that the person shares with Pepper, in order to go on with the interaction.

- Multiple persons handling: while the FER system is ready to detect more faces with the relative emotion, Pepper is not. Giving the possibility to interact with more than one person opens to a lot of possibilities and decisions, and it would be a huge improvement. Of course this is a massive work and should be tackled in a conscious way with different experts on human social interaction.

- Richer proposals: since the size of the project, the Pepper's proposals, triggered by the detected emotion, are simple and the only ones available. In a real scenario the proposals should be multiple and let the person choose the one which prefers the most. For instance, regarding the game proposal, when the emotion detected is anger, there could be different types of games, to cover the widest range of genres.

- Acquire more information: knowing the current emotion of a person who is taking part in the interaction is not sufficient to choose the best behavior that Pepper can adopt. Humans naturally change behavior according to the person who is in front of them. Similarly, in order to make a person comfortable during a human-robot interaction, more information such as age, gender, and the surrounding environment are required, so to choose the right language formality, guess preferences, and show even more appropriate reactions.

Thanks

I want to thank the professor Luca Iocchi for delivering this course, and all the people who have participated to the lectures presenting articles, projects, and future development of human-robot interaction, which has been revealed to be an interesting and learning field of artificial intelligence and robotics with several insights, and especially an inspiration for the future professional career.

References

[Fong et al. 2003]

T. Fong, I. Nourbakhsh, K. Dautenhahn (2003)
A Survey of Socially Interactive Robots.
Robotics and Autonomous Systems, 42:143–166.
http://www.cs.cmu.edu/~illah/PAPERS/socialroboticssurvey.pdf

[Vinciarelli et al. 2008]

Vinciarelli, A., Pantic, M., Bourlard, H., and Pentland, A. (2008).
Social signal processing: state-of-the-art and future perspectives of an emerging domain. *In Proceedings of the 16th ACM international conference on Multimedia, 2008.*
http://www.dcs.gla.ac.uk/~vincia/papers/sspsurvey.pdf

[Iocchi et al. 2015]

L. Iocchi, M. T. Lázaro, L. Jeanpierre, A.-I. Mouaddib:
Personalized Short-Term Multi-modal Interaction for Social Robots Assisting Users in Shopping Malls. *ICSR 2015: 264-274*
http://www.diag.uniroma1.it/iocchi/publications/iocchi-icsr15.pdf

[Eirini Malliaraki 2018]

Social Interaction with Drones using Human Emotion Recognition.
HRI '18: Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction.
https://dl.acm.org/doi/10.1145/3173386.3176966

[Cauchard et al. 2016]

Jessica Rebecca Cauchard, Kevin Y. Zhai, Marco Spadafora, James A. Landay
Emotion Encoding in Human-Drone interaction.
HRI '16. In The Eleventh ACM/IEEE International Conference on Human Robot Interaction (pp. 263-270). IEEE Press.
https://dl.acm.org/doi/10.5555/2906831.2906878

[Serengil 2017]

Serengil, Sefik Ilkin
TensorFlow 101: Introduction to Deep Learning for Python Within TensorFlow.
https://github.com/serengil/tensorflow-101/

[Viola et al. 2001]

Paul Viola and Michael Jones
Rapid object detection using a boosted cascade of simple features.
Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.
https://web.iitd.ac.in/~sumeet/viola-cvpr-01.pdf

[project_demonstration.mp4]

https://youtu.be/lU1Jm5o9DDM