ALMA MATER STUDIORUM – BOLOGNA UNIVERSITY

CESENA CAMPUS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COMPUTER SCIENCE AND ENGINEERING DEGREE

Human-Robot interaction with low computational-power humanoids

Discussion in robotic interaction

Supervisor

Prof. Davide Maltoni

Presented by

Igor Lirussi

Co-supervisor

Prof. Alexandre Bernardino

Academic Year 2018-2019

To my uncle, pilot Roland Collino,

flying forever in the skies, you are tracing me the way.

Table of contents

Motivations

Robots have been presented to the society since many decades, but their presence in the every-day life is still a mirage or a luxury for the few. Humanoid robots nowadays especially are confined to the research laboratories or to big companies' factories. All the investigation is carried out on expensive latest-model devices creating expensive platforms, computational costly algorithms and a gap with the society's average financial resources. This leads to the impossibility for the majority of people to afford a humanoid for daily use.

Bulk production for society requires robots built with commonly available technology and simple manufacturing. The software, furthermore, needs to be designed for systems with limited computational power and without expensive sensors/actuators.

Since most of the interaction algorithms are based on powerful platforms, the research aims to fill this disparity creating and testing a suitable software for interaction with social affordable humanoids.

Research Outline

This article investigates the possibilities of human-humanoid interaction with robots whose computational power is limited. The project has been carried during a year of work at the Computer and Robot Vision Laboratory (VisLab), part of the Institute for Systems and Robotics in Lisbon, Portugal. The institution was founded in 1992 and is affiliated to the Instituto Superior Técnico (IST) of the University of Lisbon, ranked amongst the top schools of engineering in Europe. The Computer and Robot Vision Laboratory conducts research on computer and robot vision, focussing on video analysis, surveillance, learning, humanoid robotics and cognition.

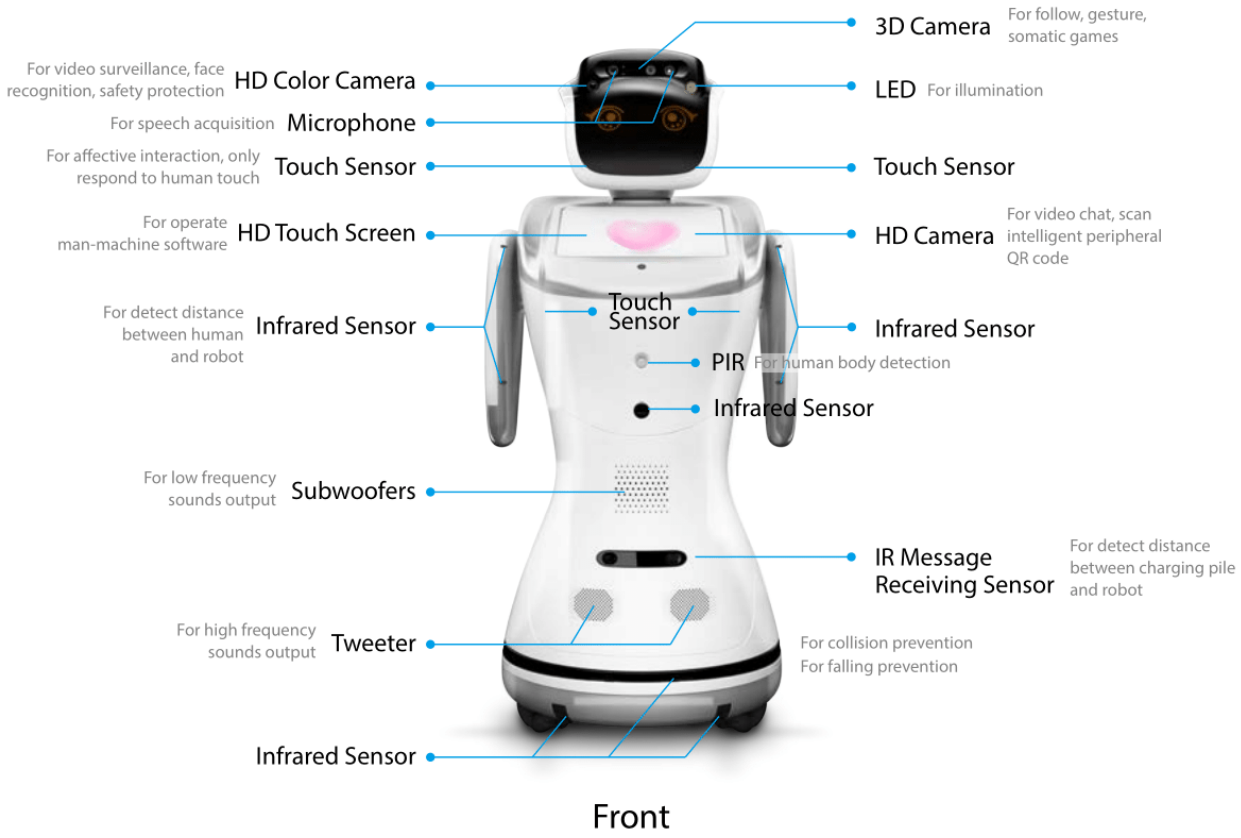
Communication, the basis of interaction, is simultaneously visual, verbal, and gestural. The robot should provide users a natural language communication, being able to catch and understand the person's needs and feelings. The design of the system should, consequently, give it the capability to dialogue with people in a way that makes possible the understanding of their needs. The whole experience, to be natural, should be independent from the GUI, used just as an auxiliary instrument. Furthermore, it has to communicate with gestures, touch and visual perceptions and feedbacks. This creates a totally new type of interaction where the robot is not just a machine to use, but a figure to interact and talk with: a social robot.

The contributions are the investigation of the social capabilities of a cheap humanoid robot, a design for a human-robot interaction pipeline, the Android system in robotics as Finite State Machine and the resource analysis as a controller of a robot. Furthermore, the research investigates the most suitable and lightweight Java algorithms for face detection/recognition, speech recognition/synthesis and dialog processing.

The writing is structured as follows. The first section describes the overview of the hardware used, the robot structure. Section 2 will discuss the general pipeline adopted to define the behaviour of the robot, which forms the basis of the work. Then, the 4 following sections form the core part of the work, where it is deepened into detail every step of the interaction system designed. Finally, the conclusions are given and the future work.

1. Robot overview

The robot chosen for the purpose is a Robot S1-B2 – “Sanbot ELF”, humanoid robot 90 cm tall and of 19 kg of weight [Fig 1.0]. It has a battery that can last for 10 hours of use and a dedicated charging station where he can dock autonomously.



[Fig 1.0 Robot Appearance]

System design

The general architecture of the system is shown in [Fig 1.1]. A tablet with a touch screen interface is located on the chest of the robot. The tablet works as a general controller, running Java applications on a customized Android operating system. The mobile device accesses external devices connected through USB to control the robot actions. The manufacturer provides an SDK to access the other parts of the system.

Two cameras are connected directly to the tablet, one Red Green Blue Depth (RGB-D) camera and one High Definition (HD) color camera. The tablet has another integrated low resolution camera.

Another sensor directly connected to the Android system is a module with 2 microphones, for sound capture. Two USB ports, one Type-A and one micro Type-B, are available for the debug or for the upload of new contents. All the computation relative to the motors and the other sensors is performed by 2 robot micro-controller units (MCU) with ARM32 architecture.

These devices are connected to the tablet both through USB and among each other with a RS232 bus interface.

One MCU is dedicated to the head and computes all the instructions for: (i) the projector present on the back side, (ii) the led screen for the face expressions, (iii) the ZigBee signals and (iv) the array of microphones for sound localization.

Other activities carried out by the MCU of the head are: the control of the LEDs, the touch sensors and the motors above the neck.

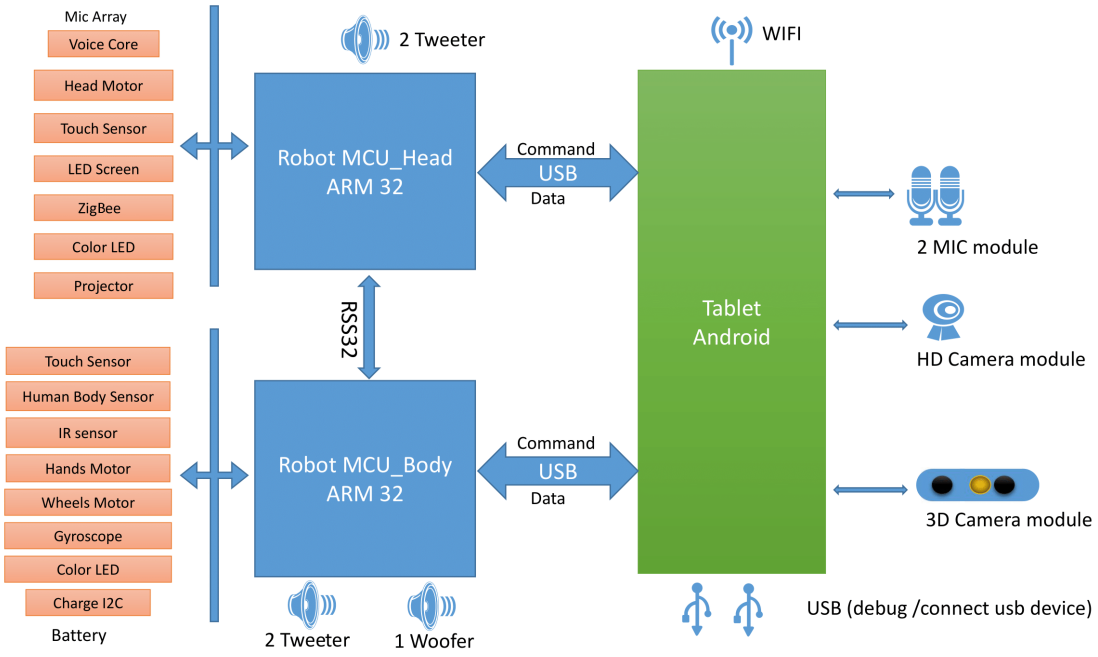
These tasks are also present in the microcontroller for the body of the robot that controls in the same way touch sensors, LEDs and motors.

The actuators controlled by the body MCU are both the wheels' motors and the hand's motors.

Furthermore, also the human body sensor (passive infrared sensor) and the active infrared sensors for measuring distances are managed by this MCU.

Lastly, it manages also the gyroscope for the orientation of the robot and the charge of the battery.

Overall the computation in the whole system is split and balanced between the high-level software, handled by the tablet, and the hardware control, managed by the 2 microcontrollers.



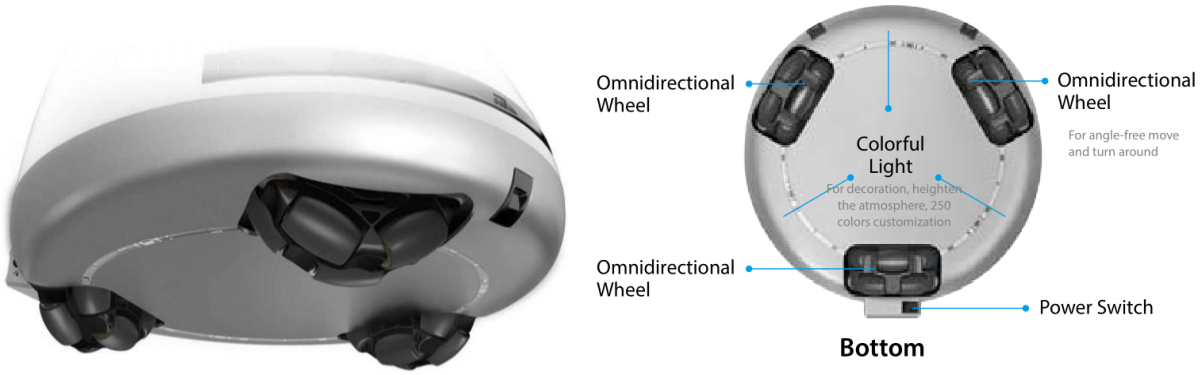
[Fig 1.1 Architecture of the system]

Actuators

The actuators present in the robot are mostly designed for movement and display, rather than physical interaction with the surrounding environment. The humanoid has 2 arms that can move independently for a range of 270 degrees swing back and forth.

In the head there is a joint motor with 2 degrees of freedom (2 x DOF) that permits a movement of 180 degrees at horizontal direction and 30 degrees at vertical direction.

For permitting the robot to move, 3 omnidirectional wheels are present, as shown in [Figure 1.2]. Every wheel has 12 small freely moving wheels for 360-degree spot rotation to navigate paths and surfaces easily, even if complex. The geometrical arrangement of the wheels in an equilateral triangle allows not only back and forth movement, but even horizontal or oblique slides, narrow turns and in-place rotations.



[Fig 1.2 Omnidirectional wheels]

Sensors

The system is equipped with a huge (over 60) variety of sensors.

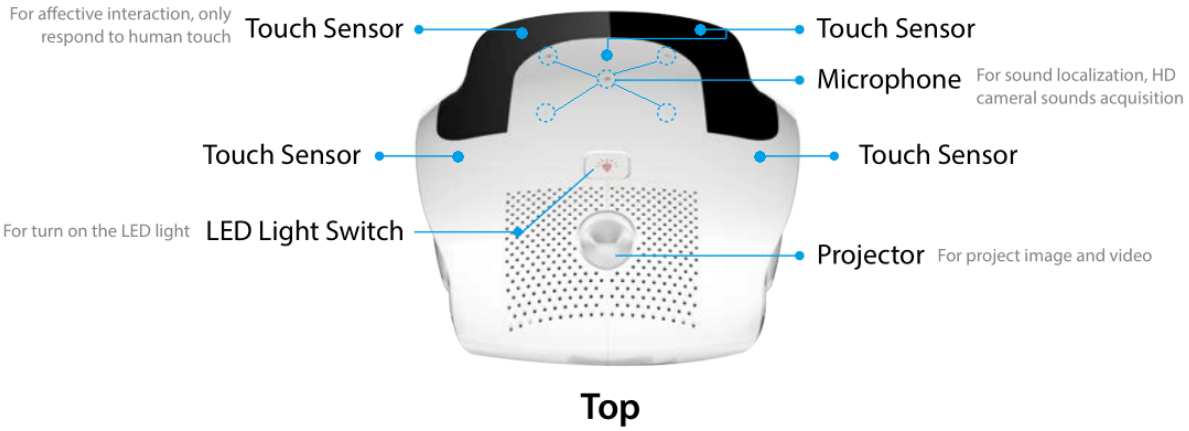
In the head it is installed an RGB-D camera, for capturing the depth of the environment and the 3D structure of the surroundings. Nearby there is another camera, high definition and with a 140 degrees aspheric angle, to have a panoramic view of the front. The aspheric lens helps to reduce or eliminate spherical aberration due to the wide field of view. Below the cameras, to perceive human speech there is a high-quality double microphone, while on the top of the robot an array of 5 small microphones helps detecting the direction of the sounds [Fig 1.3].

Touch sensors are present on top of the head, on the chin of the robot, on the shoulders and on the front of the body. On the hands there are two other touch sensors to detect handshakes.

On the arms, moreover, 4 infrared (IR) sensors are installed to measure distances from the obstacles, like on the bottom where there are 10 of them to avoid collisions.

To detect humans there are 2 passive infrared sensors (PIR) in the front and in the back, these sensors don’t emit infrared but are triggered by the motion of an object emitting infrared radiation (radiant heat) like human body.

The last sensors present are not on the surface, but inside the robot: they are the gyroscope and the electronic compass to retrieve the orientation of the robot.



[Fig 1.3 Touch sensors and microphones]

Media

The tablet is a 10.1 inch (ca. 26 cm) 1080P HD touch display that provides a rich set of interaction possibilities to the user. The position of the screen is in the chest of the robot, it is located at a height easily reachable for both an adult and a child. The display of the tablet can be projected in 60Hz by a laser projector with an HD aspect-ratio of 16:9.

The humanoid has, for audio output, 4 tweeters and a subwoofer for high and low frequency sounds and music.

In the head, in the body and in the arms decorative LEDs are placed to change the appearance of the robot, with the possibility of 250 colours customization. Near the cameras there is another white high power LED to illuminate the view in poor lighting conditions.

The robot, moreover, is capable to show a full range of facial expressions [Fig 1.4] to communicate visually emotions or particular conditions.

[Fig 1.4 Robot facial expressions]

Integrated software SDK

The robot is equipped with a software development kit to aid the developers to control the system and to provide some extra services in the cloud.

The SDK provides a set of “manager” classes in Java that can be called for different purposes. The motion managers control the motors of the moving parts or the wheels. The speech manager handles the synthesis and the recognition, the sound localization angle instead is passed with the hardware manager, which provides also the data of the touch sensors and the gyroscope. Ultimately, the system manager has to be used to control the face expressions and the camera manager the video input.

Some cloud services provided are the speech to text recognition through the Semantic Recognition API and the face recognition with the Face Recognition API.

2. Proposed Pipeline

The proposed pipeline aims to provide the robot with a full independent and interactive behaviour. Thanks to the software developed in this thesis, the robot should be able to work full-time, charging autonomously when needed, moving in its environment safely across an entire floor of a building, and interacting autonomously with the people that it meets, providing them help and services.

To design of the system used a finite state machine [Fig 2.0]. This choice is justified by the fact that these activities are generally mutually exclusive, and also due to the low computational power of the robot.

The default state is **Wandering**, where the robot spends most of its time. In this activity the system lets the robot move with the omnidirectional wheels around the environment. The hands of the robot are oscillating during the advancement to emulate a human walk. Since all the area has to be covered and there is no special destination, the movement is random. This allows it to explore all the places and to meet all the people present in the environment.

To avoid the obstacles, the IR sensors on the bottom are used to block the device before colliding and turn it in another direction. The direction is chosen according to the angle the robot approaches the obstacle. If the obstacle is detected on the right, the robot turns slightly on the left to continue the run. If the sensors detect an obstacle more in the front, the turn is larger.

In this state it is possible to deactivate the walking if the robot has to stand in a pre-determined position, like a greeter or a receptionist. Furthermore, the rotation with the sound localization could be deactivated to immobilize the system completely, in this way, if a sound is detected, the robot will ignore it and not perform any movement; this mode is useful when a special view needs to be covered.

When the battery is low, the system is made to switch to the **Charging** state. This state changes the screen of the robot to inform the users that the normal activity is not running anymore and a battery refill is needed. The robot, while wandering, will search for his charging station and, with an integrated method of the SDK, dock on it. When the battery is sufficiently charged the robot will announce it, move forward about one meter and start back his normal wandering activity.

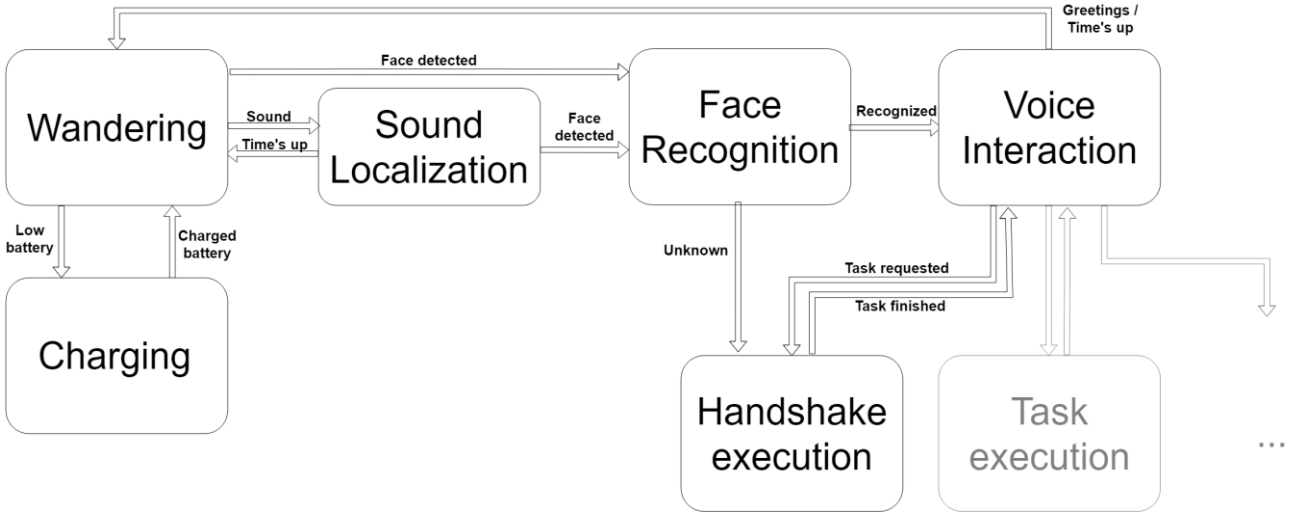
Only with the states **Wandering** and **Charging**, the device is just able to move in the environment and to charge autonomously. To design a robot with a reactive behaviour to humans, the system uses the sensors and the camera to eventually start the interaction. If a loud sound is detected, the robot enters the **Sound Localization** state. The sound source is localized and the software turns the robot heading towards the source of the sound. The device stops wandering and, for a period, listens for other inputs. In this state the face detection is running to find people nearby to interact with. If no person is detected in a specified amount of time, the robot will start **Wandering** again, otherwise the system enters the **Face Recognition** state trying to recognize this person.

The system can commute directly from the **Wandering** state to **Face Recognition**, without the preliminary sound localization. For this reason, the face detector is running also in the **Wandering** state. The presence of a person's face in the camera triggers the system, consequently, to try to recognize it and decide the successive action.

If the person is recognized the robot will move to the **Voice Interaction** state, listening and speaking with the interlocutor. In this state it is possible to ask for extra tasks to execute, like handshake, directions, general info [Chapter Tasks]. If the person is not recognized, the system goes for the Handshake Execution state and,

when this is finished, goes back to **Voice Interaction**. In this sense, face recognition is not a pre-condition to provide the service: the user can be served even if it is not recognized. The **Voice Interaction** is the main developed way to interact, in this activity the user can talk freely with the robot having a chat or ask him to perform a task. A set of tasks is implemented, and can be easily expanded adding knowledge and capabilities to the system. When one of them is finished, the interaction flow goes back to the voice dialog.

Finally, if the interlocutor greets the robot or doesn't answer in the time limit, the system goes back to the normal wandering state, searching another person in the area.



[Fig 2.0 Pipeline of the proposed algorithm]

3. Sound localization

The purpose of this part is to localize a loud sound in the environment and turn the robot attention towards that sound to detect people or relevant events. The sound is localized just with the angle of the source related to the robot in the 2 dimensional horizontal plan.

For the sound localization in the robot the class “hardwareManager” [Fig 3.0], provided by the SDK, has been used. On this class can be set a VoiceLocateListener to execute a function every time a loud sound is detected, the direction of the sound is computed by the SDK.
The direction of the sound is calculated with the help of the array of microphones placed on the robot. The head, indeed, contains a set of microphones directed horizontally in any direction.
When the function is called, the robot’s movement, if it is not busy in other activity, is stopped to permit the rotation. The LEDs of the body flicker to notify that something is detected, by using another function of the class “hardwareManager”. A timer is set to make the robot resume its walk after some seconds if nothing has been identified or a face detected.

```
//voice angle localization
hardWareManager.setOnHareWareListener(new VoiceLocateListener() {
    @Override
    public void voiceLocateResult(int angle) {
        Log.i(TAG, msg: "voice located at : " + angle);
        //if it is idle
        if (!busy && MySettings.isSoundRotationAllowed()) {
            //stop movement
            wanderOffNow();
            //head up
            headMotionManager.doAbsoluteLocateMotion(locateAbsoluteAngleHeadMotion);
            //flicker led
            hardWareManager.setLED(new LED(LED.PART_ALL, LED.MODE_FLICKER_BLUE));
            //rotate at angle
            rotateAtRelativeAngle(wheelMotionManager, angle);

            //wanderOn after a while
            //handler to avoid motor overlapping
            wanderHandler.removeCallbacksAndMessages( token: null);
            wanderHandler.postDelayed(() → {
                    wanderOnNow();
            }, delayMillis: 1000* MySettings.getSeconds_waitingToWanderAfterSoundLocalization());
        }
    }
});
```

[Fig 3.0 Sound rotation callback]

To perform the rotation, a special function “rotateAtRelativeAngle” has been built [Fig 3.1]. The direction and the angle of the movement depends on the orientation of the robot and the shortest direction to take. The function takes as input the manager of the wheel’s movement and the positive angle desired to rotate clockwise, a negative one for the opposite direction.
A transformation has been coded at the beginning to force all the integers to be positive, then the movement is computed depending on the desired angle. If the clockwise relative motion is less than half a turn, the robot simply executes it; if it’s more than 180° it is more efficient to perform it rotating counterclockwise. The same optimization is present for the opposite turns.

A negative or positive result is returned depending on the best direction computed.

```java
/**
 * rotation at a relative angle in the shortest way
 * @param wheelMotionManager the motion manager to rotate the robot
 * @param angle the relative angle clockwise desired to turn,
 *              can be negative to define counter clockwise
 * @return 1 if the rotation is clockwise, -1 otherwise
 */
public static int rotateAtRelativeAngle(WheelMotionManager wheelMotionManager, int angle) {
    //correction negative angles
    while (angle < 0 ) angle = angle + 360;
    //calculation best direction
    if (angle < 180) {
        RelativeAngleWheelMotion relativeAngleWheelMotion = new RelativeAngleWheelMotion(
                RelativeAngleWheelMotion.TURN_RIGHT, speed: 5, angle);
        wheelMotionManager.doRelativeAngleMotion(relativeAngleWheelMotion);
        Log.i( tag: "IGOR-rotation", msg: "turning right " + angle);
        return 1;
    } else {
        RelativeAngleWheelMotion relativeAngleWheelMotion = new RelativeAngleWheelMotion(
                RelativeAngleWheelMotion.TURN_LEFT, speed: 5, (360-angle));
        wheelMotionManager.doRelativeAngleMotion(relativeAngleWheelMotion);
        Log.i( tag: "IGOR-rotation", msg: "turning left " + (360-angle));
        return -1;
    }
}
```

[Fig 3.1 Function for the rotation]

The sound localization mechanism, as the first trigger of the pipeline, allows the robot to be reactive to occurrences in the surrounding environment. This emulates a sort of curiosity, where the system is attracted by loud sounds.
A few different benefits emerge: the alignment with the origin of the sound probably is also an alignment with the person that made it. The attention of the robot can be drawn by calling it, the delay to move again after the sound permits to have time to start the interaction.
Furthermore, in a conversation where two people speak facing each other the machine looks alternatingly to the speakers.
Ultimately, the rotation due to loud sounds randomizes the wandering behaviour even more and leads the social robot to the most frequented spots.
Overall, thanks to the sound localization mechanism implemented in this thesis, the robot is more in contact with the environment (compared to the default behaviour without sound localization), consequently looking smarter, and more responsive if it reacts to the sound stimuli. Reactions to vision stimuli are going to be discussed in the next chapter.

4. Face detection

The pipeline, so far, made the robot capable of moving and charging autonomously, as well as reacting to sound stimuli. To gain the status of social robot the system still has to detect people and interact with them. When the robot walks around in the environment, to be more helpful, it has to eventually start the interaction, not just to be asked for it. This initiative could be also useful if the robot is just standing at the entrance as a receptionist, to give information to people.
Multiple ways exist to detect human presence, from 3D scanning with depth sensors or LIDARs, to vision, from purely passive infrared sensor (PIR sensor) to acoustic sensors.
The detection with sounds has proved to be unreliable to start a conversation, due to the fact that it's difficult to understand if a person is speaking with the machine or with another human.
The PIR integrated in the back of the robot has been used, but just to make it rotate in case a person moves behind it. For these reasons, we decided that the start of an interaction relies on the vision with the RGB camera. The use of a face detector ensures that there is an actual person in front of the robot, avoiding false positives with objects.
Nevertheless, since the camera is oriented forward, where the robot is facing, it allows the interaction to start only when the robot is looking at the person and not when is oriented in another direction. For this reason, it has been chosen, between the 3 cameras, the narrow one that is facing higher, to avoid people too far away and to optimize the capture of the face of a near person that looks down towards the small robot.
Vice versa, the interlocutor has to look directly at the robot to make the face detection possible, ensuring also that the interaction is desired and the person is not doing something else nearby. When a human facing the robot is detected, at this point the interaction can start.

For this task have been analysed four alternatives: the SDK built-in face detector/recognizer by the company, the Android built-in face detector, the Haar cascades and Deep Neural Networks implemented with the OpenCV (Open Computer Vision) library.
According to the application in view, the detector should not give any false positive, detecting humans that are not present, because this would make the robot start the interaction in unexpected conditions. Another requirement is good reliability when a face is in the field of view, even in difficult conditions, like low light, far-away, not frontal or with different expressions. If a face is not detected, the robot will probably ignore the person until a good frame is caught. The last, less restrictive, requirement is the time required for computing a detection. This has been evaluated more softly because, since the system uses the detection just as a trigger, the difference is not relevant in the user experience if the waiting time to be detected is up to 2 seconds. A bigger delay has been judged inappropriate because the person can get bored or can simply not be detected between a frame and the subsequent. This could happen when a frame is processed, the person meanwhile enters in the field of view and after a few seconds goes away, then another frame is taken but no people are detected.
The SDK provides a call-back related to the camera manager when a face is recognized, with some extra information if the person is detected as known, but no parameters can be changed. The Android face detector is based on the Mobile Vision API of Google, the architecture is not editable but it permits enabling tracking and landmarks, and the mode can be set between accurate and fast.

With the OpenCV library for Java it has been possible to develop the Haar cascades and the DNN face detectors. The library has to be inserted in the project, then it is possible to load the files required for the Haar cascades and for the Deep Neural Networks.

Since the primary objective was to be reliable and accurate, at the expense of the reaction time, the Haar cascades has been rejected. The algorithm, even if it was the fastest implementation at 17 FPS, had too many false positives and the inability of detecting the rotated faces. The SDK face recognition didn't detect any inexistent person, but the amount of real people detected was really low in poor light conditions and it took on average 5 seconds to react. Even if rejected, this method has been considered and made compatible as an alternative in case a face recognition is needed. The Android built-in face detector was reliable, but failed in the long distances. The Deep Neural Network of OpenCV revealed to be the most suitable for the pipeline, despite the slow performance of 2 FPS on the robot. The network detected almost all the faces in difficult conditions, with low light and rotated faces, without false positives, triggering the interaction with just a little delay.
To use OpenCV, the library had to be imported in the project and connected to the application, then at the beginning of the desired activity should be loaded. After this process, a Net class is available in the software to be instantiated, but before it's necessary to find a model and a configuration file to load [Fig 4.0]. The files are necessary to create the structure of the network and to populate it with the right weights. The network architecture is defined with its parameters in a configuration file with extension ".prototxt". For the present system, a pre-trained model for face detection has been chosen: the file with extension ".caffemodel".

```
String DNN = "CAFFE";
String modelFile, configFile;
Log.i(TAG, msg: "DNN selected: " + DNN);
modelFile = getPath( file: "res10_300x300_ssd_iter_140000_fp16.caffemodel", context: this);
configFile = getPath( file: "deploy.prototxt", context: this);
net = Dnn.readNetFromCaffe(configFile, modelFile);
```

[Fig 4.0 creation of the network from model and a configuration files]

For every frame grabbed, the image must be converted to a matrix of values; an image processing function is available in OpenCV for this purpose. Then the matrix with the current view of the camera has to be forwarded through the net, which will return in a list of face detections [Fig 4.1].

```
// Get a new frame
Mat frame = inputFrame.rgba();
Imgproc.cvtColor(frame, frame, Imgproc.COLOR_RGBA2RGB);

// Forward image through network.
Mat blob = Dnn.blobFromImage(frame, scalefactor: 1.0,
        new Size(IN_WIDTH, IN_HEIGHT),
        new Scalar(104, 117, 123), swapRB: false, crop: false);
net.setInput(blob);
Mat detections = net.forward();
```

[Fig 4.1 The detections returned by the net]

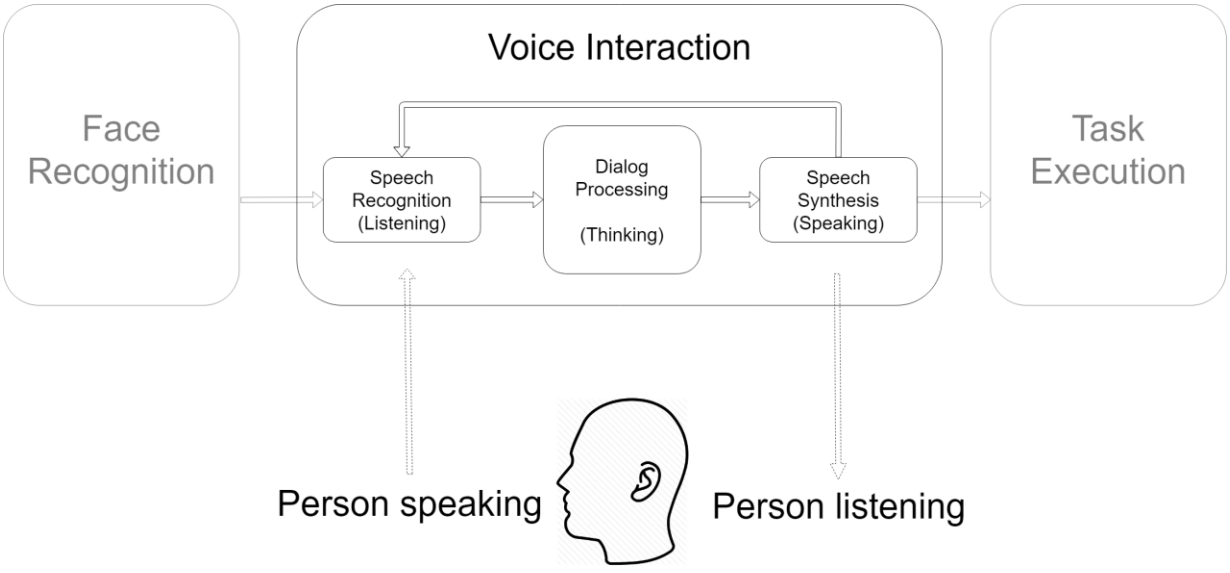
Finally, if there is a detection and the confidence is above the threshold desired, 80 percent, the voice interaction could start.

The resulting user experience is more automatic and pleasant, no actions are required to start the conversation and the objective of making everything independent from the GUI is achieved. A button has been anyway implemented to start the interaction, for any eventuality. The system results more proactive and sharper, making the robot more helpful.
Furthermore, if it has been used a face recognizer the robot could distinguish between people personalizing the speech, as will be discussed in the next chapter.

5. Voice interaction

The voice interaction part is, in the structure of system, the most important step to understand the desire of the user and to give him a positive experience. The objective is simulating the conversation in natural language and interacting with the real person in the most human-like manner possible. Whether the person willingness is just to have a chat or there is a specific demand to satisfy, both these occurrences should be managed by the dialog activity. This part of the pipeline becomes, therefore, a complete independent activity to be performed and, at the same time, a mechanism to choose other tasks. In this last case, it becomes also a turning point after a specific request is accomplished. In other words, it's not just a transient step in the pipeline, but a logic place where a person can engage in a dialogue, or (s)he can perform a choice and come back after that. Here, we have noticed that the user spends most of the time he interacts with the robot. Such as the dialog in daily life for humans, the voice interaction module has a central part on the relationship with the machine. While the trigger with the face-recognition for the approach has a backup way to be executed (with the button pressing), this activity does not have a way to replace it, so it has to work seamlessly, fluidly and pleasantly to provide an overall positive user-experience. A few buttons to permit some choices and to give examples of possible verbal requests are shown on the integrated display. This is not an alternative system but it stimulates curiosity and it facilitates the initial approach of the user that sometimes could be uncertain on how to deal with the robot and what it could do.

To keep the conversation flowing, the process has been divided in 3 sub-steps: recognition, dialog processing and speech synthesis [Fig 5.0]. The mechanism reflects the human way to process an answer in a speech: listening to the request, thinking a solution, tell the answer. The role of every step, discussed in the following paragraphs, is also the opposite and the complement of what the interlocutor is doing in the conversation: when the person is speaking the robot is listening, when the robot is speaking the person is listening. The process is cyclic to keep the conversation going and to create a turn-taking behaviour with the partner.



[Fig 5.0 The conversation flow]

Nevertheless, due to the fact that the speech synthesis is an asynchronous process, the robot will start to listen again before his synthesis has finished. This means that the two last steps are executed

consecutively without any input-wait so the beginning of a new speech recognition is almost immediate, 300ms. The sounds emitted are filtered to avoid recognizing its own synthesis as input. The process has been developed in this way because the robot will listen uninterruptedly, also while it is speaking, and it can even be interrupted humanlike with another request.

a. Speech Recognition

The speech recognition is part of the computational linguistic explained in this chapter. It consists in the translation from the spoken language (computed as audio) to text (a set of characters).
This field, half in computer science, half in linguistics, is not recent and its origins can be found in the late 1950s with rudimental recognition of a single word at once.
Then in the 60s the research continued with IBM's 16-word "Shoebox" machine, but it is in the 1970s that the topic takes off [5.a.1], with the ARPA (now DARPA) Speech Understanding Research program. The S.U.R. program carried ahead the research from 1971 to 1976 for continuous speech recognition [5.a.2], leading to great improvements over the simplistic acoustic modelling with the HARPY System [5.a.3].
During these years came also to light the first commercial speech recognition company, Threshold Technology.
Over the next decade Speech Recognition turns toward prediction with a new statistical method known as the "Hidden Markov Model" [5.a.4]: rather than merely using templates for words and looking for sound patterns, the model took into account the probability of unidentified sounds being words.
Despite this, during this period the technology was still strongly limited by the hardware capabilities and by the discrete dictation with pauses after every single word.
This problem was overcome in the 90s with the rise of processors speed and, finally, the milestone of continuous speech recognition with a large vocabulary was reached.
Software for this task arrived soon in the market, like SPHINX [5.a.5] and Dragon Dictate (then NaturallySpeaking), and speech recognition groups were founded in Microsoft (by the developer of the Sphinx-II system) and Apple.
In the new millennium with the emergence of neural networks as high-quality acoustic models [5.a.6] this technology moved also in the phones becoming within everyone's reach.
Recently, noticeable steps forward have been made thanks to the flexibility of big data combined with machine learning. Since the natural language is really various, the generalization of the deep learning becomes an advantage for the comprehension of the speech.

For our robot the objective was keeping the software on-device while maintaining the recognition fast and accurate. Currently, a few systems are being developed that can perform this task with an accuracy up to 97% [5.a.7], even running twice as fast as real-time on a common phone [5.a.8], with minimum memory footprint [5.a.9], but they are still not available as open source code.
For this reason, we have opted to analyze also server-based solutions, taking into account the use of the "Android integrated speech recognizer" (based on Google Cloud Speech-to-Text), the "Kaldi Speech Recognition Toolkit", the "Mozilla Project DeepSpeech", the

“PocketSphinx” (Sphinx version for Android) and the default speech recognizer, running in the cloud, provided in the Robot SDK.
Unfortunately, due to the impossibility to manually access the audio from robot’s microphone, the choice forcefully fell on this last.

The integrated SDK provides a “speechmanager” class with the possibility to programmatically call the wake up/sleep listening status of the robot in the time when it is deemed appropriate. Furthermore, is possible overwriting these two functions to add custom code executed in these two state changes [Fig 5.1]. Thanks to this, it is possible also to wake up the robot’s recognition system automatically, for a custom duration, after it goes to sleep.

```
//Set wakeup, sleep callback
speechManager.setOnSpeechListener(new WakenListener() {

    @Override
    public void onWakeUp() { Log.i( tag: "IGOR-DIAL",  msg: "WAKE UP callback"); }

    @Override
    public void onSleep() {
        Log.i( tag: "IGOR-DIAL",  msg: "SLEEP callback");
        //infiniteWakeup is a custom variable to control the duration
        if (infiniteWakeup) {
            //recalling wake up to stay awake
            speechManager.doWakeUp();
        } else {
            //change button in the view to notify the sleep status
            imageListen.setVisibility(View.GONE);
            wakeButton.setVisibility(View.VISIBLE);
        }
    }
});
```

[Fig 5.1 Listener onWakeup/onSleep]

It may happen that the person simply stops speaking to the robot without any final greeting. In this case, to avoid listening forever, a timer was implemented to let the robot wait a few seconds for a new sound. When the time is up, if an input is not detected, the recognizer passes to the conversational engine a special code to notify the “no-response”. The robot will give a primary warning and then, after waiting some more time, it will go back to the basic task, wandering around the environment.

With another function is possible to receive directly the string computed when a sentence is recognized [Fig 5.2]. The “speechmanager” provides a multilingual recognition system for EN, IT, FR, DE, PT.
It should be noted that, with this SDK class, the function has to return an answer in less than 500ms, otherwise the operating system will compute a default “I did not understand” response.

```
//Speech recognition callback
speechManager.setOnSpeechListener(new RecognizeListener() {
    @Override
    public void onRecognizeText(@NonNull RecognizeTextBean recognizeTextBean) {

    }

    @Override
    public boolean onRecognizeResult(@NonNull Grammar grammar) {

        //start timer
        //long startTime = System.nanoTime();
        //cast object received to text string lastRecognizedSentence
        try {
            lastRecognizedSentence = Objects.requireNonNull(grammar.getText()).toLowerCase();
        } catch (NullPointerException e) {
            lastRecognizedSentence = "null";
        }
        //recognized part
        //notify update to UI with a separate thread not to freeze the interface
        runOnUiThread(() -> {
                //update ui with text recognized
                tvSpeechRecognizeResult.setText(lastRecognizedSentence);
        });

        //here can start the computation on the text recognized
```

[Fig 5.2 Listener onRecognizeResult]

Having a way to command the system without touching the screen but with natural language, provides the user a better and wider communicative channel.
The interaction is also faster and more intuitive than the manual selection with the screen.
Retrieving the input with this method, even if a huge amount of cases has to be managed, gives the opportunity to think at the robot as a more perceptive system.

From the sentence provided by the recognizer, it is now possible to define a reaction of the robot but, before, a conversational processing core is needed to support the dialog and to choose a task.

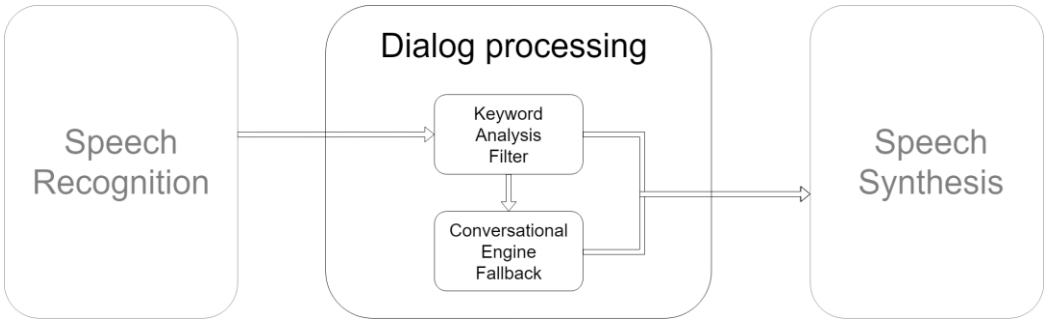
b. Dialog Processing

This chapter will concern the dialog system, or conversational agent, implemented in the robot to find a suitable reply to the user sentence.
People perform this natural activity thousands of times a day, but to succeed at this same task on even a basic level, an accurate design is required.

From a historical point of view, the development of natural language conversational systems has started in 1965, with the development of ELIZA by Joseph Weizenbaum at MIT. ELIZA is a milestone in the field of dialog systems, not just because it was the first but also for his popularity and his early utility. ELIZA, in fact, was built to emulate a Rogerian psychologist: the mechanism was a smart rule-based system that, when it doesn't know the

answer, reflects the patient's affirmations back at him. Asking the motivations of the statement ELIZA is able to draw the patient out in an introspective path. [5.b.1]
In 1971 a similar system, called PARRY, used also a pattern-response structure enriching it with emotions like fear, anger and mistrust to simulate a person with paranoid schizophrenia. When the system was tested using a variation of the Turing Test, psychiatrists could distinguish interviews with real paranoids from interviews with PARRY only 48 percent of the time. [5.b.2]
In the 1970s the field became famous and a lot of other similar implementations were created, some for entertainment purpose, like Jabberwacky, which goal was to develop a fully voice-operated system. During these years, the conversational agents passed also from the close-domain conception, i.e. talking just about medicine, to experimentation on open-domain dialogs. All these systems, like ELIZA and PARRY, were totally based on rules coded manually, making them really difficult to develop and maintain. The necessity of a different architecture, that can create itself the knowledge required, was satisfied by the use of machine learning during the 1980s. The introduction of this new technology allowed the creation of new implementations that can learn from examples. To generate responses, the corpus-based systems mine huge datasets huge datasets of conversations between humans, collected using information retrieval [5.b.3]. Simple Information Retrieval-based systems usually find the most suitable human's response from the conversation's database matching the question asked. The drawback of just mirroring the training data, on the other hand, is that the answer cannot be controlled and it's based on the most common human behaviour, even if wrong.
A decade later, in 1995, A.L.I.C.E. came along, being inspired by ELIZA. ALICE "Artificial Linguistic Internet Computer Entity" developed by Richard Wallace, was one of the earliest systems to integrate both natural language processing and an early form of artificial intelligence [5.b.4]. The conversational engine used AIML, a new mark-up language, to define heuristic pattern matching rules, proving a substantial upgrade on previous software. ALICE was revolutionary, for being open-source and allowing developers to create their own personalized conversational engines. ALICE was, in its turn, an inspiration for many science fiction movies and for Apple's personal assistant "Siri".
In the third millennium, it was developed IBM Watson, specifically designed to compete in Jeopardy, which he won in 2011. It used a big amount of data to find the correct answers to the quizzes. [5.b.5]
In the 2010s the use of dialog systems (typically task oriented) increased thanks to personal assistants like the already cited Siri (2010), Google Now (2012), Cortana (2015), Alexa (2015) and messaging apps used by companies to automatically find the service required by the customers.
In 2016, Microsoft launched on Twitter a self-learning agent called "Tay". The purpose of the research was to emulate human conversations by monitoring and interacting with real people online. Unfortunately, the entity started to misbehave copying also the malevolences from the humans, leading to the experiment's shut-down, after 16 hours. [5.b.6]
Nowadays, applying machine learning technologies for building robust and scalable dialogue systems is still a challenging task [5.b.7]

In this project, the objective of the dialog part was to preserve a reasonably meaningful response to emulate a real human like dialogue, not holding up the interlocutor too much, and to be within the "speechmanager" recognition constraints (less than 500ms). The dialog processing activity had two tasks, to find the requests of the user for a service and to provide natural replies for a generic discussion. The implementation has turned out to be efficient with a double partition mechanism [Fig. 5.3] in which a first part quickly skims the frequent sentences and the task requests, and a fall-back part that manages the remaining interactions.



[Fig 5.3 Dialog processing state]

The primary intent-detection has been developed with the research of key-words related to the most common sentences or task requested. For example, the embodiment of the word "where" indicates an explanation request regarding a place. The inclusion in the same sentence of both "meeting room" and "where" triggers the system to give indications about the location of the meeting room. Synonyms have been taken into account, so the use of "lab" instead of "laboratory" sets off the response, anyway.
This part is responsible for selecting, eventually, the task to execute after the speech synthesis. If a desired option is detected, system will commute from the Voice Interaction state to the state implementing the chosen task.
All this process speeds ups the response-time in such cases to 50ms.
Unfortunately, the keyword based speech analysis system has some drawbacks because it doesn't take into account the variety of the language and all the changes of meaning that the other vocabularies can produce. In example: "Where are the keys of the meeting room?" will trigger the same task to give the indications about the room.
Many other alternative methods are available for understanding the tasks desired in a finite set of choices: the structure of a neural network, for example, is perfect to receive the input words, compressing the information of the sentence, and returning the most likely task in a finite set. This is more flexible than a bare keyword based research, on the other hand the network has to be trained with a big amount of examples. For the variability of the tasks of the new system and the amount of changes done so often, it is too time consuming and unpractical the retraining after every alteration.
It is still possible, scarifying the offline constraint, to use an online service like Wit.ai or Dialog flow to add a new task without the necessity to retrain every time the model. But the wideness of a normal speech, with purposeless parts, makes anyway these systems too tight to sustain a full conversation. The need of an underneath base of knowledge is clear, maintaining the benefits of the first part just for the detection of special tasks.

Clearly the implementation of this coarse filter to improve the performances of the system, as discussed before, is not capable of handling a full conversation, but can pre-filter a lot of intentions. A request not contemplated in the first part still needs to be handled.
First of all, a request to the user for a repetition has been implemented, this pushes the person to talk loudly and scan the words, avoiding misunderstandings for the speech recognizer.
Then a fall-back method is required to handle all the other conversational situations.
A few methods have been analyzed for a conversational core, FireBase Smart Reply, DialogFlow and Wit.ai conversational systems, AIML 2.0 and the default SDK online service.

Task-oriented dialog agents, like Dialog Flow and the similar service provided by the site Wit.ai, have proved to be very useful in intent detection, but they didn't perform well for the conversational part without a specific purpose.
FireBase Smart Reply, is a Google system to suggest, based on a database of real conversations, relevant replies to messages. Smart Reply generally helps users respond to messages quickly, it is not a real conversational system but, since it is based on real dialogs, can be used to find the most common human answer in a general-purpose dialog. The system has been considered because it can be converted not just to suggest, but also to directly reply with the first most popular response, working even offline. However, it is not possible to customize in any part. This is not suitable for the answers of a robot, cause in some cases the most common human answer could not be applied to a machine. The robot's integrated SDK during the tests performed better for the conversation, but it is also not customizable and it needs an Internet connection to work. This, for the objective of the study, has been considered a point against.
The AIML 2.0 system (Artificial Intelligence Mark-up Language), instead, is a fully customizable database built as an XML dialect, that allows to create natural language software agent.
The files can be easily modified with any IDE (Integrated development environment) and all the answers can be personalized to the device (in this case a robot) and his identity (name, owner, purpose). Thanks to this, the name has been changed and, for example, the diet questions lead to a more real "I need only electricity" answer. All these personalized behaviours are not possible with the other analysed options.
The code can be computed with engines for all the platforms, whose job is finding the most suitable reply to a given input. To choose the answer, a "pattern", namely, the structure of the speaker sentence, and the previous conversational history are used to find the proper context of the speech. Being able to deal with contexts turned out to be very important, because analysing just a request out of the context can lead to wrong answers or even to the impossibility to reply (i.e. answering to "and you?"). The code can run offline and with a good amount of situations handled, showing on the tests a better control on the answers and a more reasonable response in topics not covered by the filtering implementation. For this reasons the AIML 2.0 has been chosen as a conversational system and a fall-back for the dialog part.
The ability to deal with contexts is useful also to make inquiries, when the conversation is at a standstill [Fig 5.4]. Making the dialog bidirectional with questions, going from the total human initiative to the mixed initiative in the speech, keeps it interesting for more time.

```
Human: hi
Robot AIML: Hi how are you?
Human: good, you?
Robot AIML: It's all good.
```

[Fig 5.4 Bidirectional conversation]

The mechanism can also save variables to serve as a rudimental memory. In this way it is possible to remember the name and some information collected in the speech or differentiate the answers depending on previously programmed behaviours [Fig 5.5].

```
Human: my name is Igor
Robot AIML: Good to know.
Human: what's my name
Robot AIML: Igor
```

[Fig 5.5 Remembering a name]

The conversational engine has been connected also to the face recognition module to automatically set the name of the person in front. This could be useful to let the robot understand the identity of the interlocutor and personalize the dialogue without asking the name again to a known person. In the case the name is not passed, the conversational engine will call the interlocutor simply “friend” and will ask his name whenever possible to save it in a local variable. The pipeline consequently can be modified swapping, if needed, the face recognition or the face detector without any consequence.
To memorize the name given from the face detector into the conversational engine, a silent presentation from the user is built and passed to the system [Fig 5.6]. The reply of the dialog processing unit is discarded and the robot will continue normally waiting the answer from the first question by the person.

```
Log.i( tag: "IGOR-SPEECH", msg: "-START SILENT PRESENTATION");
//grabbing the name from the previous activity
Intent intent = getIntent();
String name_received = intent.getExtras().getString( key: "name");

//if the name is passed/exists
if(name_received!= null && !name_received.equals("")) {
    //creating a silent presentation with the name received
    String silent_presentaiton = "my name is " + name_received;
    //the silent presentation is given to the conversational engine that memorizes the name
    String response = chat.multisentenceRespond(silent_presentaiton);
    Log.i( tag: "IGOR-SPEECH",  msg: "Human: " + silent_presentaiton);
    Log.i( tag: "IGOR-SPEECH",  msg: "Robot: " + response);
} else {
    Log.i( tag: "IGOR-SPEECH", msg: "-NAME NOT PASSED");
}
Log.i( tag: "IGOR-SPEECH", msg: "-FINISH SILENT PRESENTATION");
```

[Fig 5.6 Silent presentation to memorize a name]

Furthermore, the database can self-expand during the conversation bringing to the interesting capability to learn new notions and correspondences [Fig 5.7]. This is allowed by a module that recognizes the use of the verb “to be” to indicate relations or identities in the language.

```
Human: what is sun?
Robot AIML: I don't know
Human: learn sun is hot
Robot AIML: Ok I will learn that sun is hot .
Human: what is sun?
Robot AIML: hot
Human: what is hot?
Robot AIML: sun
```

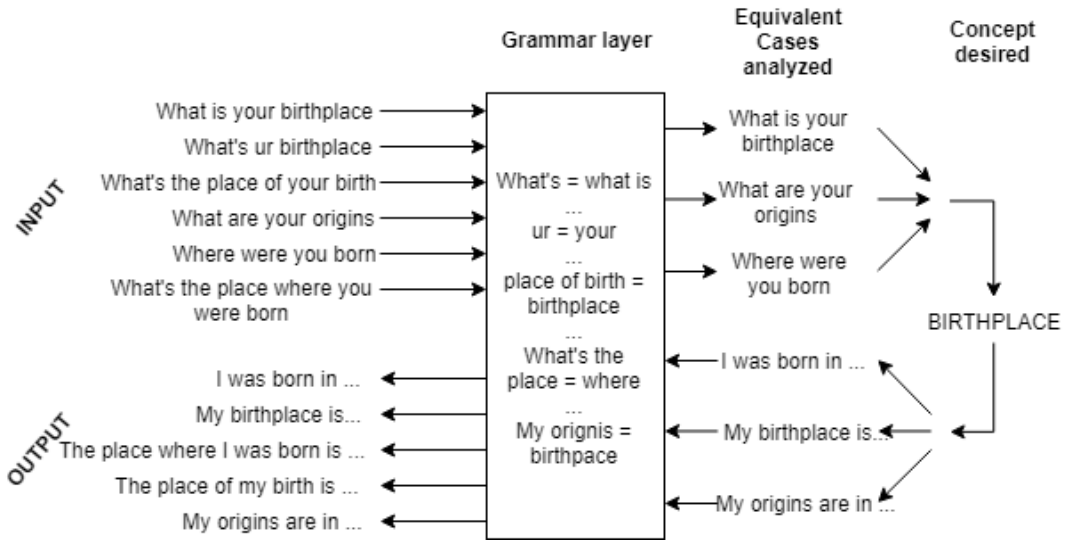
[Fig 5.7 Learning new correspondences]

Is it possible also to define some grammar rules that help to match the sentence spoken with the correct pattern. The spoken language is full of abbreviations, slang or different ways to express the same concept. There is a lot of variety in how people talk, for example there are so many ways people can say hello and they can be interpreted in the same way. In the system designed this process, generally of substitutions, but also of more complex equivalences, is executed at the beginning and permits to exponentially increase the number of sentences understood.
In the [Fig 5.8] a few rules, in AIML defined as <category>, can be seen, they state some grammatical and then logical equivalences. For example, what will match the <pattern> “what’s” and other words after or before (coded with the *) will be substituted with “what is” and the remaining parts of the sentence around. Also, synonyms can be coded, like “place of birth” and “birthplace”, or slang, like “ur” and “your”.
Should be observed that now, for a lot of questions asked, the system works like a cascade to find an answer. For a not explicitly coded question like “what’s ur birthplace” before the most general substitutions are done, “what’s” with “what is” and “ur” with “your”. Then the result will match logically a single specific question: “what is your birthplace” will be interpreted as “where were you born”. Consequently, asking this last question or all the intermediate steps will lead at the same final point, making it easy to answer to a huge variety of equal requests.
Other logical equivalences can be set using also the previous defined grammar, going always to a single concept.

```
<!-- GRAMMATIC abbreviation example -->
<category>
    <pattern>WHAT'S *</pattern>
    <template><srai>WHAT IS <star/> </srai></template>
</category>

<!-- synonyms -->
<category>
    <pattern> * PLACE OF BIRTH *</pattern>
    <template><srai> <star/> BIRTHPLACE <star/> </srai></template>
</category>

<!-- slang -->
<category>
    <pattern> * UR *</pattern>
    <template><srai> <star/> YOUR <star/> </srai></template>
</category>


<!-- LOGIC equivalence -->
<category>
    <pattern>WHAT IS YOUR BIRTHPLACE</pattern>
    <template><srai>WHERE WERE YOU BORN</srai></template>
</category>

```

[Fig 5.8 AIML]

This process of "compression" of the input to a single concept is also reverted in phase of creation of the answer. When the topic is found it is possible to define a list of equivalences for the reply in a way to add variety to the speech. This avoids giving the impression to the user of a mechanical interface that gives always the same answer. Changing the feedback and being more fluid means a more variegated and interesting system. The "sand-glass" design [Fig 5.9] allows, then, to organize better the knowledge of the conversational engine and, most of all, to identify the topic of the speech.



[Fig 5.9 Sandglass design of the knowledge]

The database can be enlarged at will with more files available on Internet or it can be customized building other brand-new modules to add different capabilities. More the database is enlarged, more all the files require time to be charged in the dialog engine. To reduce the waiting time at the loading of the app, 80 seconds in the case of our 10'000 categories, the AIMLs has been converted in CSV before the integration in the robot.

These format creates intermediate files less comprehensible than AIML, but faster to load for the engine, bringing the waiting time at the beginning to 15 seconds.
Collections have been tested and customized for historic events, maths and quiz facts, jokes and world currencies [Fig 5.10]. New behaviours have been investigated also like copying a person when he requests and understanding the gender from the name.

```
<!-- the module tells what happened in history for every day of the year -->
<category><pattern>ON THIS DAY</pattern><template>On this day in history <date format="%B %d"/>:<br/><br/>
<srai><date format="%B %d"/></srai></template></category>


<category>
<pattern>JANUARY 03</pattern>
<template>
1924 - Howard Carter discovered the sarcophagus of Tutankhamun<br/>
1959 - Alaska became the forty-ninth US State
</template>
</category>
```

[Fig 5.10 New behaviour coded]

Now that a meaningful reply has been found by the AIML engine, it is time to communicate it verbally to the user. For this job a speech synthesizer is needed, whom functionality will be discussed in the next chapter.

c. Speech Synthesis

Speech synthesis, also called “read aloud” technology, is the artificial generation of human speech. The intelligible sounds can be produced with different types of technology, but they are lumped together by the necessity of convey a meaning by sonic waves.

Before being software, the first speech-synthesisers were purely mechanical: at the end of the 18th century. Ch. G. Kratzenstein, professor of physiology in Copenhagen, succeeded in producing vowels using resonance tubes connected to organ pipes.
The acoustic resonators were clearly inspired by the human vocal tract: the vibrating tongues were driven by air.
Wolfgang von Kempelen in 1791 modified the machine with models of the lips and the tongue and lips, making it reproduce almost all consonants and vowels.
In the early 1800s, Charles Wheatstone built successfully a customized version of the device, now also some sound combinations and even full words could be produced, and Joseph Faber's built “Euphonia” that produced not only ordinary and whispered speech, but it also sang the anthem "God Save the Queen".
All these devices were still air-based, the first properly called speech synthesizer was VODER [5.c.1] (Voice Operating Demonstrator) presented by Homer Dudley in 1939. The system was inspired by VOCODER (Voice Coder) developed at Bell Laboratories a few years before. After introduction in New York World's Fair of VODER the interest of scientific community in speech synthesis grow more and more. Finally, it was demonstrated that intelligible speech can be produced artificially. The basic idea of VODER is still very similar to present systems which base their model of speech on a source-filter.
The late 1950s gave origin to the first computer-based speech-synthesis systems: the first full speech synthesiser for English was created by Noriko Umeda, in the Electrotehnical Laboratory of Japan, 1968 [5.c.2].

In 1979, thanks to Allen, Hunnicutt, and Klatt who created at M.I.T the MITalk laboratory text-to-speech system [5.c.3], began the commercial use of TTS systems by Telesensory Systems Inc.
In the same years Kurzweil introduced the first reading aid for blind people that, even if expensive, it was useful in libraries and service centres.
Integrated circuits for speech synthesis in the 1980's were produced with increasingly lower costs, like the Votrax chip that was used by Richard Gagnon to build an inexpensive Type-n-Talk system. Texas Instruments used also low-cost linear prediction synthesis chips to build Speak-n-Spell synthesizer based on linear prediction coding (LPC). It gained popularity since was widely used as a reading aid for children.
One of the methods applied in these years that is still used today in speech synthesis is hidden Markov models (HMM). A hidden Markov model is a collection of states connected by transitions with two sets of probabilities in each, this model showed good results in the human speech emulation [5.c.4].
Nowadays the limits of state of the art are not set anymore by technological factors, but just by the methods used. The best speech synthesisers use recorded speech data to train deep neural networks to get closer to the quality, the naturalness and intelligibility of the human voice [5.c.5].

For the robot, the objective was that the system chosen has to be reliable, acoustically pleasant and possibly multilingual. Being the speech synthesis the main feedback from the robot to the user, we considered it as really important for the user-experience. For this reason, a response should be always given, without any jam, till the final greetings and the voice should be more natural as possible. The vocal, lastly, has to reflect the identity of the robot, a female.

Despite the complexity of this kind of systems, this time has been possible to insert in the examined possibilities on-device solutions. Although the recognition could not be performed without a service, making the robot unable to carry on a complete speech, this gives the capability to the robot to give verbal alerts even if it's isolated, without Internet access or connection to external computers.

Between the wide amount of possibilities for a word synthesizers for phones has been chosen for the evaluation: "TextToSpeech engine" integrated in Android, "Google Speech synthesis" and the speech synthesiser integrated in the robot SDK.
All the systems provide a feminine voice with Multilanguage support and behave in a good way. For this reason, it has been preferred the continuity of keeping the same service provided from the robot SDK used in the speech recognition.
Reusing of the provided "speech manager" allowed not only the code to be cleaner, but also to have control on the speed and the intonation of the voice. This possibility has proved to be useful: a slight faster voice gives the sensation of a faster reply and a smarter system and a deeper intonation makes conveys the sensation of a warmer, more human, less toy-like interlocutor [Fig 5.11].

```
//speak settings  language/speed/intonation
private static SpeakOption speakDefaultOption = new SpeakOption();

public static boolean initializeSpeak() {
    speakDefaultOption.setLanguageType(SpeakOption.LAG_ENGLISH_US);
    speakDefaultOption.setSpeed(50); //from 0 to 100 default: 40
    speakDefaultOption.setIntonation(40); //from 0 to 100 default: 30
    return true;
}

public static SpeakOption getSpeakDefaultOption() { return speakDefaultOption; }
```

[Fig 5.11 Settings of the voice synthesiser]

The class provides a simple function call with in input a string of characters to be pronounced and the speak options [Fig 5.12].

```
//say hi
speechManager.startSpeak("Hi" + person_name, MySettings.getSpeakDefaultOption());
concludeSpeak(speechManager);

// 50% say Good morning/afternoon/ecc...
double random_num = Math.random();
Log.i(TAG,  msg: "Random = " + random_num);
if (random_num < 0.5) {
    int hours = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
    if (hours < 6) {
        speechManager.startSpeak("it is so early", MySettings.getSpeakDefaultOption());
    } else if (hours < 12) {
        speechManager.startSpeak("Good morning", MySettings.getSpeakDefaultOption());
    } else if (hours < 18) {
        speechManager.startSpeak("Good afternoon!", MySettings.getSpeakDefaultOption());
    } else if (hours <= 24) {
        speechManager.startSpeak("Good night!", MySettings.getSpeakDefaultOption());
    }
    concludeSpeak(speechManager);
}

//start the dialog activity.
Intent myIntent = new Intent( packageContext: MyBaseActivity.this, MyDialogActivity.class);
//insert the name of the person in the annex
myIntent.putExtra( name: "name", person_name);
MyBaseActivity.this.startActivity(myIntent);
//terminate this activity
finish();
```

[Fig 5.12 SpeechManager.startSpeak]

The audio will be synthesised immediately and his duration depends on the length of the sentence to pronounce. The call, however, is asynchronous: this means that the program will continue executing the following commands when speak is running. This implementation, as said, is useful to keep the robot always listening and responsive to interruptions, but can lead to misbehaviours if there is another sentence to be said that will interrupt and overcome the previous one. For this reason, it is present a listener to execute the code when the speech is finished and a function to check the speaking status. The checking has been used to avoid

closing the activity before the speech finishes, interrupting it, and sometimes to delay the execution of the subsequent code before the synthesis is concluded.
Now that the answer has been delivered to the user this can reply, bringing ahead the conversation or asking for a task to the robot that is again in the listening state.

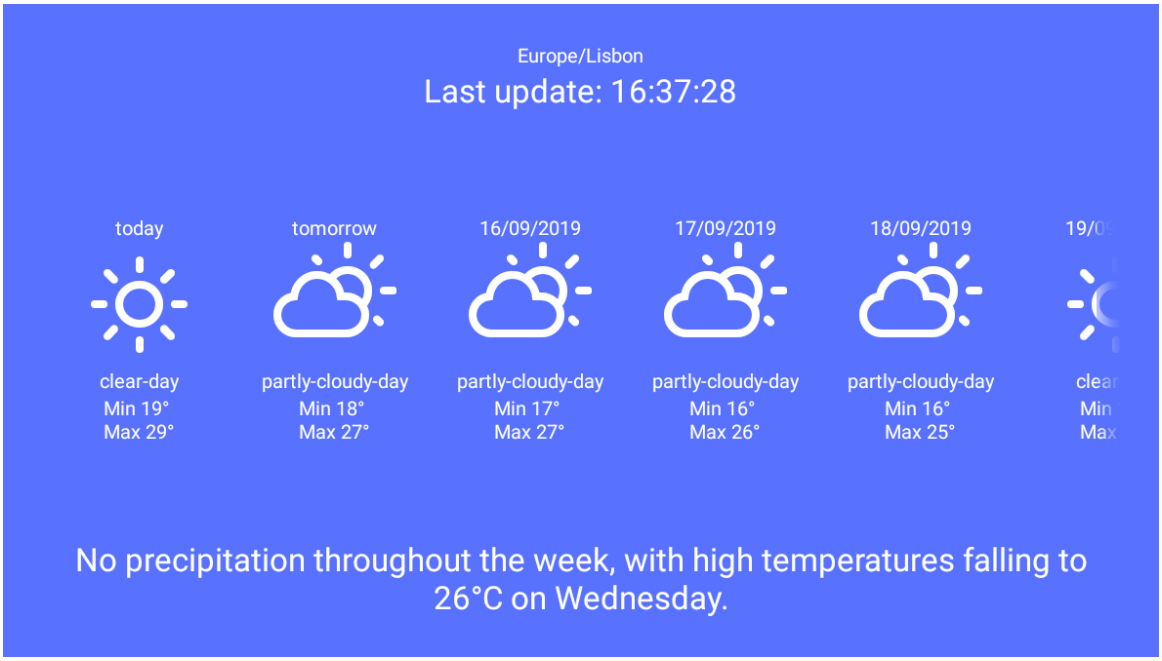
6. Task execution

To demonstrate the capabilities of the system, a set of tasks of different nature has been implemented. Since the robot could be used in multiple scenarios, due to his rich hardware implementation, just a few demonstrative ones have been chosen. Four groups of example tasks are shown below to give an idea of the possible uses.

Public service

The first scenario in which the robot could be useful is public service. Being able with the integrated screen to give info of public utility, makes it perfect for general service in public places or home assistant.
General informational tasks have been implemented, like telling the weather, showing the maps with possibility to change location, or showing the events of an institution's calendar.
For these tasks the Internet connection is of course required to retrieve the updates, the group in fact has been used also to explore the tasks implementable with access to the net.

The weather task was implemented with an API call to an online forecast service, an asynchronous thread handles the download of the JSON without compromising the GUI [Fig 6.0].
When the JSON has been downloaded, with the help of an android JSON interpreter, the forecast of the next 8 days is taken and looped. For every day is shown on the screen the date, the general sky conditions, the icon of the forecast accordingly to the prevision and the minimum and maximum temperature.
The icons for the sky conditions are displayed with a special font, this makes them infinitely scalable and easy to manage.
The interface is implemented manually, with the loading bar that lasts until the download is complete.
Furthermore, a summarizing sentence is shown at the bottom and also synthetized directly to the user, to give him a general idea about the week and answer his question.
The speech recognition is active at this point, this allows exiting not only pressing the button, but also asking it directly to the robot or thanking him. This keeps the conversation flowing and brings back the speech to voice interaction activity.

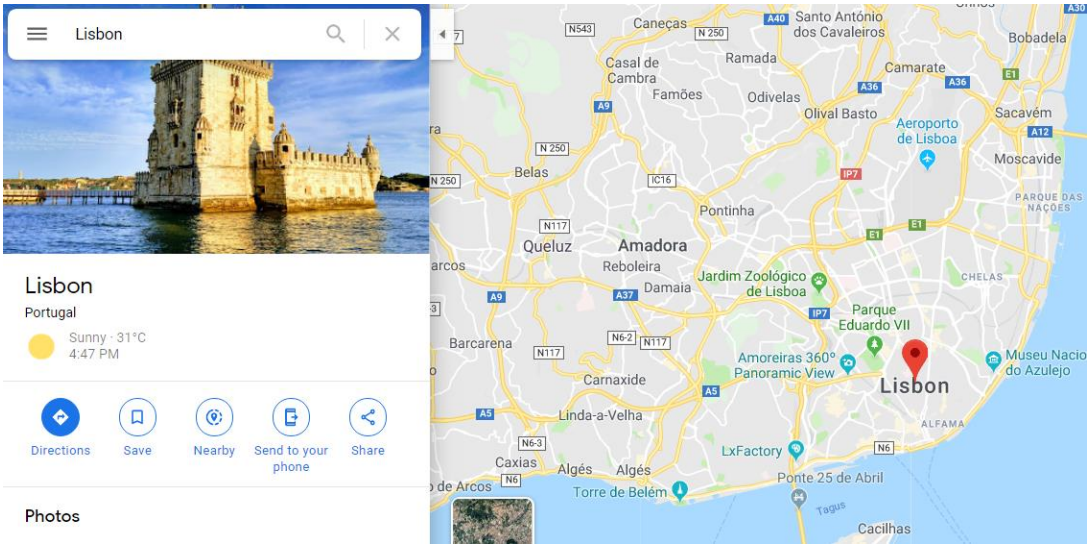


[Fig 6.0 Weather GUI]

Displaying the map of a place asked by the user is another activity implemented to show a public service. The geographic information is shown on the screen charging a map-website, to do this an activity that manages a web-view has been coded. The activity, in addition to handle the web interface, takes from the Dialog system the city desired and computes URL to charge, if no city is specified the current location is displayed as a central point.
Also here the speech recognition is active, not just for the exit but also for changing location.
It is possible to ask the robot, in fact, to show another place by voice. To catch the location desired a list of conjunctions is provided and the sentence spoken is analysed to find the words following one of them [Fig 6.1].

```
boolean newPlace = false;
String[] separators = {"show me ", " of ", " in ", " on ", " to ", };
for (String separator : separators) {
    if (lastRecognizedSentence.contains(separator)) {
        place = StringUtils.substringAfter(lastRecognizedSentence, separator);
        speechManager.startSpeak( text: "OK, let's go to " + place, MySettings.getSpeakDefaultOption());
        newPlace = true;
    }
}
if(newPlace) {
    //Computing URL
    place = place.replace( target: " ",  replacement: "+");
    String url = "https://www.google.com/maps/search/" + place;
    Log.i( tag: "IGOR-WEB", url);
    //loading URL in the web-view
    myWebView.loadUrl(url);
}
```
[Fig 6.1 Computing new location asked]

At this point the URL of the new position is computed and the webpage is refreshed [Fig 6.2]. The user can change location asking again or thank the robot to go back to the normal voice interaction.



[Fig 6.2 Displaying on the robot screen a map requested]

Lastly, the display of the events of an institution and public holydays from the interested calendars [Fig 6.3].

This is done downloading asynchronously the iCal files, available to integrate every online calendar with the personal one. After this, when the download is finished, using the java library “ical4j” an object Calendar is built for each one.

Looping through every calendar is now possible to create a list of events with start time, end time and summary. This list is then adapted and used to fill an open-source calendar view available online.

The user can manually scroll, zoom and listen to the summary of the event when he presses the corresponding box.

With the voice it is possible to control, as usual, the task, going at the present day or exiting.

[Fig 6.3 calendar of an institution displayed]

Retail

The second possible situation analysed is the retail use of the system. For this group has been explored the possibility of extra tasks with an internal execution, without any external connection. The robot in a hypothetical use in a shop can give place directions, and info about products.

To help a customer giving a direction both vocal and gestural methods have been used.
Before the robot explains the position of the place requested in the building, then rotates and indicates the direction to follow to reach it.
To make the robot orient itself and always point the right cardinal angle has been used the integrated compass [Fig 6.4]. If the compass is not initialized the system make the robot look around to calibrate it and obtain the cardinal angle. At this point, having the self-orientation and the point to indicate, the system calculates the angle for the movement, turning in the smartest direction as explained in the chapter of sound localization.
The robot rises his hand and tells the user to take that way, then returns to the previous position waiting again for the voice interaction.

```
private void indicate(int desiredCardinalAngle) {
    //INDICATION
    //if not initiated the gyroscope
    if (currentCardinalAngle == 0) {
        //looking around to initiate the gyro
        int lookingAroundAngle = 50;
        rotateAtRelativeAngle(wheelMotionManager, lookingAroundAngle);
        sleepy( seconds: 2);
        rotateAtRelativeAngle(wheelMotionManager, -lookingAroundAngle);
        sleepy( seconds: 4);
    }
    //rotate at cardinal angle direction desired
    int cardinalrotation = rotateAtCardinalAngle(wheelMotionManager, currentCardinalAngle, desiredCardinalAngle);
    sleepy( seconds: 4);
    //indicate
    AbsoluteAngleHandMotion absoluteAngleHandMotion = new AbsoluteAngleHandMotion(AbsoluteAngleHandMotion.PART_LEFT,
    handMotionManager.doAbsoluteAngleMotion(absoluteAngleHandMotion);
    //speak direction
    speechManager.startSpeak( text: "in that direction!", MySettings.getSpeakDefaultOption());
    concludeSpeak(speechManager);
    //rotate back
    rotateAtRelativeAngle(wheelMotionManager, -cardinalrotation);
    //hand down
    absoluteAngleHandMotion = new AbsoluteAngleHandMotion(AbsoluteAngleHandMotion.PART_LEFT, speed: 5, angle: 180);
    handMotionManager.doAbsoluteAngleMotion(absoluteAngleHandMotion);
    sleepy( seconds: 2);
}
```

[Fig 6.4 The movement to indicate]

To give info about other products has been implemented a presentation activity, working directly offline and using all the motion and media capabilities of the robot.
The presentation is a sequence of actions coded to grab the attention of the spectator and to engage him.
During the performance the robot starts displaying the object on the tablet, changing LEDs colour and face expression. Then moves the head and the rest of the actuators to be more proactive, as a seller who is convincing a customer. At the end reacts with the sensors to the touch on the body or on the tablet to explore the product.

Healthcare

Another scenario for the pipeline is the interaction with people for healthcare. The voice interaction has been already explained in the chapter dedicated, this group of activities completes the investigation of the field with also bodily interaction and the collecting of some suggestions.

An example of interaction with body could be a handshake, a simple gesture that is often used as a primary touch between strangers to introduce each other. This context involves a lot of challenges that will be examined in this paragraph.
To emulate a complete interaction, all the capabilities of the robot have been used: the voice synthesis, the head, hands and body movements, the face emotions and the LEDs.
When the handshake request is detected, the robot, indeed, lift the head, greets the person and simultaneously rises the hand, rotating the body and the head.
The body rotation on the hand side is coded to reach out the partner, showing more willingness. The head rotation in the opposite direction is therefore needed to keep looking at the person.
In this position a timer starts, related to the patience for the answers examined in the chapter of voice interaction.
While waiting the system could incite the opponent to grab firmly the robot hand, to make touching sensors detect better the skin presence. If the timer expires, the system puts the robot in the previous position with a negative reaction, a sad sentence and a disappointed expression.
If the touching is detected the humanoid smiles, lights up and greets vocally.
To shake the hand, the arm motors then are commanded to move repeatedly up and down for a few times before going back to the normal position [Fig 6.5].
This gesture is a peasant way to start interacting with a humanoid, of which often there is mistrust. Helping people being more in contact with a harmless humanoid system, could improve also the relations with others and well-being in the society.

[Fig 6.5 Handshake moment]

Another example could be leaving a comment, a suggestion or an opinion to the robot. This could be whether useful for the developers or to the users. The possibility to give a personal sentence to the robot has been developed with XML. A file inside the system is created the first time the application is used, then every time the person leaves a hint this is saved inside a tag <suggestion>. The whole file will have a list of all the tips gathered, eventually retrievable.

Education

The last scenario examined is education and entertain. The examples developed are project videos and dance.

The first task is quite simple, a projection of a video or the screen of the tablet, but very useful to show media sources to many people. The code implemented uses the settings to open the projection with the right contrast, shape and mode (wall or ceiling). The projector always mirrors the screen of the tablet, so it is possible to show procedures and apps. There is a delay to let the hardware heat, and start to project on the surface desired. The video at this point is retrieved from the memory card of the system and displayed on both screens with a personalized view.
The view makes the video controllable in case a pause or a jump on a particular point is needed.
At the end the system will go back to the voice interaction module, to listen for any other request.

Another request handled by the system is the dance. Clearly the structure of the robot is not able to move fluidly, but the objective is to involve the people surrounding, especially children, to do physical activity and get familiar with rhythm.
To perform the most involving show, the robot plays the music with the integrated speakers and lighting up in every part with colourful animations. The movement coded is rising the hands, turning on the place or doing small slides going back to the previous position.

Conclusions

The research presented a human-robot interaction algorithm that is modular, with optional parts, solid, with always a backup way to proceed, and lightweight. The humanoid showed to be completely autonomous, working seamlessly in and between the periods of interaction, movement and charging.

The design of the voice interaction module with speech synthesis and recognition provides a communication with the robot that is natural, easy and pleasant. The system is able to catch and understand the person's needs and feelings, answering accordingly. Thanks to the fall-back conversational engine, furthermore, the number of topics in the dialog is wide, with answers that are fast and smart.

The limited computational-power, as expected, affected most of all the image processing part, but the decision taken of using the face detection just as a trigger avoided any noticeable consequence. The Internet connection objective in the main pipeline, instead, has been violated only for the speech recognition, but with the soon-to-be availability of open-source on-device implementations, the core part of the system will become connection-independent.

The choice of Android and Java has proved to be useful for a finite state machine and efficient for coding complex behaviours. Furthermore, the use of a mobile device as a controller allowed integrating in the social robot apps and services commonly used on smartphones.

The objective of making the user experience independent of the GUI has been achieved with the navigation between the tasks totally by speech. The whole interaction could be conducted without having to touch the screen, calling the robot, retrieving the info desired and greeting it.

The algorithm is perceptive to external stimuli like vision of faces, sounds, touch, obstacles and human movements. Furthermore, the system makes the humanoid communicative not only with a screen, but also with gestures, expressions and voice.

Future Work

A wide spectrum of future works is possible to expand the system designed and its capabilities.

To make the vocal interaction more natural the voice intonation and speed can be changed dynamically depending on the mood that has to be transmitted.

New task can be added making the robot useful for a huge variety of purposes, for example as a reminder, as physical activity supervisor or for interior 3D reconstruction.

The navigation in the environment can be improved with the RGB-D camera, avoiding obstacles or being able to reach a designed point. The compass data retrieved could also be merged with a localization algorithm for a navigation only in 2D.

Ultimately, the robot could be made modular with different devices. This would be a huge possibility to customize the behaviour of the system just installing the app on the owned mobile device and plugging it as a "brain" module when necessary.

BIBLIOGRAPHY

[5.a.1] Book: “Readings in Speech Recognition, edited by Alexander Waibel” Kai-Fu Lee, 1990

[5.a.2] Klatt, Dennis H, “Review of the ARPA Speech Understanding Project”, in The Journal of the Acoustical Society of America, 1977.

[5.a.3] Lowerre, Bruce T, “The HARPY Speech Recognition System”, in DEFENSE TECHNICAL INFORMATION CENTER, 1976

[5.a.4] Lawrence R. Rabiner. “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”, Proceedings of the IEEE 77, no. 2 (February 1989)

[5.a.5] Kai-Fu Lee, “An overview of the SPHINX speech recognition system”, IEEE Transactions on signal processing, 1990

[5.a.6] Li Deng, Geoffrey Hinton, Brian Kingsbury, “New types of deep neural network learning for speech recognition and related applications: an overview”, 2013

[5.a.7] Zhong Qiu Lin, Audrey G. Chung, Alexander Wong, “EdgeSpeechNets: Highly Efficient Deep Neural Networks for Speech Recognition on the Edge”, eess.AS, nov 2018.

[5.a.8] Yanzhang He, Tara N. Sainath, “Streaming End-to-end Speech Recognition For Mobile Devices”, cs.CL, Nov 2018

[5.a.9] Ian McGraw, Rohit Prabhavalkar, Raziel Alvarez, “Personalized Speech Recognition On Mobile Device”, cs.CL, Mar 2016

[5.b.1] Joseph Weizenbaum. “ELIZA. A Computer Program for the study of natural language communication between man and machine”, Communications of the ACM, Vol. 9, No. 1, 1966.

[5.b.2] Colby, K. M., Hilf, F. D., Weber, S., and Kraemer, H. C. “Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes”. Artificial Intelligence, 1972.

[5.b.3] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang, “A Survey on Dialogue Systems: Recent Advances and New Frontiers”, 2018.

[5.b.4] R. Wallace, “The elements of AIML style,” Alice AI Foundation, 2003

[5.b.5] G. Tesauro, D. C. Gondek, J. Lenchner, J. Fan and J. M. Prager “Analysis of watson's strategies for playing Jeopardy! “, Journal of Artificial Intelligence Research 47, 205-251, AI Access Foundation, 2013.

[5.b.6] Neff, G. and Nagy, P. “Talking to bots: Symbiotic agency and the case of Tay”. International Journal of Communication, 2016.

[5.b.7] Yun-Nung Chen, Asli Celikyilmaz, Dilek Hakkani-Tur, “Deep Learning for Dialogue Systems”, January 2017.

[5.c.1] Homer W. Dudley, "The Carrier Nature of Speech", Bell System Technical Journal, 1940.

[5.c.2] Klatt D. “Review of Text-to-Speech Conversion for English”, Journal of the Acoustical Society of America, JASA vol. 82 (3), 1987.

[5.c.3] Allen J., Hunnicutt S., Klatt D. “From Text to Speech: The MITalk System”. Cambridge University Press, Inc., 1987.

[5.c.4] Lee K. (1989). Hidden Markov Models: Past, Present, and Future. Proceedings of Eurospeech 89

[5.c.5] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, “WaveNet: A Generative Model for Raw Audio”, 2016.

ACKNOWLEDGMENTS

I would like to thank Prof. Davide Maltoni, for being my supervisor and the University of Bologna for making this amazing experience possible.

I would also like to thank my friends of a lifetime from my home-town, the friends of house Bocca4: the old-school ones (YahYeeh), that helped me in difficult moments, and the new ones that gave me a lot of challenges. The friends of Lisbon for the tactical and moral support in my period abroad.

Another thank goes to the people of ISR, in particular to the (more than) colleagues of VisLab.

I express my gratitude to the Prof. Alex Bernardino for welcoming me, supporting and guiding me.
He went well beyond the call of duty.

I am grateful to all the pilots of the 2nd Tactical Transport Squadron of the 46th Air Brigade for the support.

Thanks to Sofia, that accompanies me even in the most difficult periods with patience and wisdom.

Thanks to Beppino, that is always helpful and available for me.

Finally, I would like to thanks my relatives, in particular my mother Claudia: some people give unconditional love, a love without a valid reason, a love that expects nothing in return: it is a form of awareness that knows no limits and deserves gratitude that does not admit boundaries.