HCI Project - MICChinetta as a VUI

Francesco Pegoraro - Luca Angioloni
Università degli studi di Firenze

Abstract

In recent years, all kind of automated devices have emerged as a complement to standard interfaces, and users, conscious or not, interact with these more often than ever.

In this work we make use of various techniques for interfacing the user to a vending machine, aiming to a reliable and effortless interaction.

The user just needs to present himself in front of the machine and have a natural conversation with the system, asking for products to buy.

The user will be prompted with the amount of money due at each step and in the end he will be asked for confirmation of the purchase.

This system is called *MICChinetta* as a tribute to the laboratory in which it was developed: the MICC.

Before the existence of MICChinetta, users at MICC used a mobile application (web app) to interact with the machine. This new method resolves the problem of always having a mobile phone ready and connected to the MICC internal network. The other purpose of this study is to understand if and how a VUI *(Vocal User Interface)* can compete with a traditional GUI.

In this controlled environment, we find that MICChinetta is on average 76% more comfortable and doubtless cooler than the standard GUI.

Code for the project is available on Github.

Introduction

With the rapid advances of machine learning, tasks like face-detection, speech-to-text and natural language processing have reached great results. We explored the possibility to implement this new existing techniques to enhance interactions between humans and automated devices such as MICChinetta.

Our goal was to build an interface designed around this concept and test its usability and performances.

With a web cam, the system should recognize known people. After the recognition, the application must start a conversation-like dialog with the user:

- *MICChinetta*: Good morning Andy, what do you need?
- *Andy*: I would like three beers and a box of chips, thank you.
- *MICChinetta*: Ok, it's three beers and a box of chips for € 3,80. Ok?
- *Andy*: Ok!

All the explanation is proposed in English, but the real system is developed for interactions in Italian.

We first investigated if it was possible to develop such a system in an embedded device like Arduino or Raspberry Pi, but, like expected, it wasn't possible due to low performances of the devices for these complex tasks. So a desktop class computer was used.

We investigated how a person asks for products and how the user expects the transaction with an automated bot to hap-

pen.

We used this knowledge to identify how to design the Bot that will interact with the users.

Related work

In this section we give a brief description of the technologies and techniques used to carry out the various tasks that the system must perform.

Face Recognition

We performed this task using an available Python package [1]. In this paragraph we are going to briefly explain how it works.

Face recognition is composed of a series of related problems to be solved in order:

- *Face-detection*: Analyze picture and find all the faces in it
- *Extract facial features (encoding)*: pick out unique features of the face that you can use to tell it apart from other people.
- *Comparing known faces*: compare the unique features of the face to find a match with an already known face in a list of known faces.

Face Detection

To find faces in an image, the author of the package used **Histogram of Oriented Gradients (HOG)** on the gray scale image taken from the webcam at every single frame. **HOG** is a feature descriptor used in computer vision and image processing for the purpose of object detection and description. The technique creates a local histogram of gradient orientations, resulting in a local descriptor at each image point.

The resulting data is then split into small squares of *16x16* pixels. For each square, the HOG data is aggregated and discretized.

Figure 1: The original image is turned into a HOG representation that captures the major features of the image regardless of image brightness

To find faces in the resulting HOG image, we correlate a known HOG pattern that was extracted from a set of known training images.

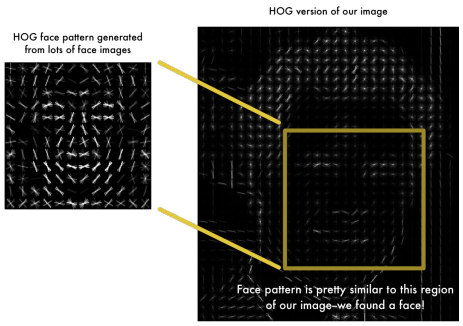

Figure 2: Find a face in a hog image

Using this technique, we can now easily find faces in any image.

Posing and Projecting Faces

To make the recognition invariant to different poses, dimensions, turning directions etc..., the HOG image is then warped to place key-points like eyes and lips in the same position for every face.

To do this, the author used an algorithm called face *landmark estimation*.

This comes up with 68 specific points called landmarks that exist on every face: the top of the chin the outside edge of each eye, the inner edge of each eyebrow etc... The algorithm is machine learning based and was trained to find this points.

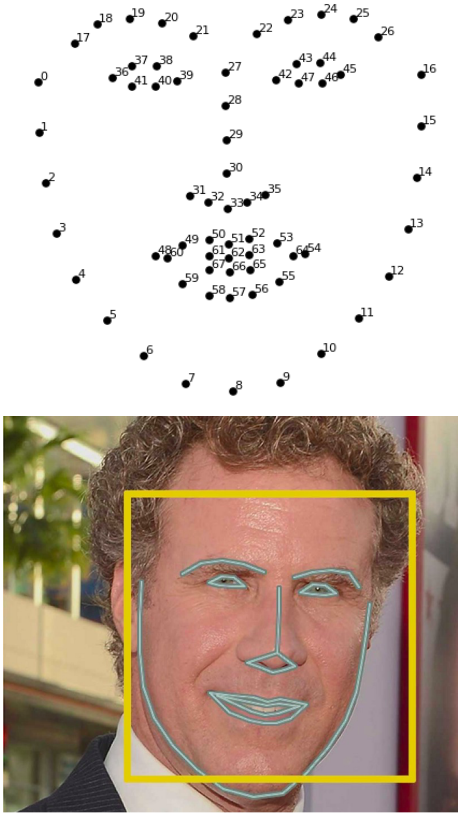

Figure 3: The 68 landmarks we will locate on every face and the result of locating the landmarks on our test image

The image is then transformed and warped accordingly.

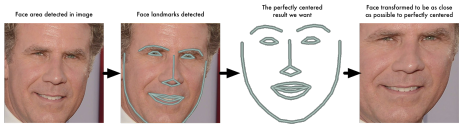

Figure 4: The complete process for centering faces

Encoding Faces

At this point, the position of the face inside of the image and the optimal affine transformations to apply to obtain an aligned and scaled representation of the face are known.

In order to compare different faces and to match images from the same individual, we need a way to describe them; to extract a descriptor of significant features.

This is done using deep learning, specifically using a Convolutional Neural Network[1], trained for an image classification task (face classification), as a feature extractor.

This (headless and pre-trained) CNN generates a 128 dim. vector containing a robust face descriptor for each image fed through the network (the detected and warped face images).

This vector is an embedding of the face in a 128 dimensional space, much lower than the original face image dimensions. This is called a face **encoding** (see Figure 5).

[1]This network is a version of the ResNet-34 network from the paper Deep Residual Learning for Image Recognition by He, Zhang, Ren, and Sun with a few layers removed and the number of filters per layer reduced by half. See: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html

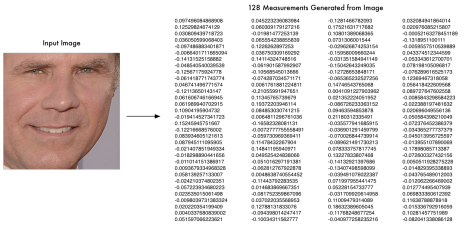

Figure 5: From the detected and warped image face a 128 dim. descriptor is extracted using a CNN

Finding the person's name from the encoding

With the encodings extracted from known faces a database is created. During the recognition phase, the extracted encoding from the face to be identified is compared with the database of known encodings and if under a certain threshold, the closest match is selected to be the face identity.

speech-to-text

To perform speech-to-text recognition we used another Python package[4] that provides a simple API for several speech to text cloud services, including Google Cloud APIs for speech recognition.

The Google service is pretty accurate and can perform speech recognition in several languages. Moreover this package has a useful feature called *dynamic threshold*: it can understand when to stop recording basing on the input signal level (if the user stops talking for example). It can also measure the environment noise, telling apart what it should convert into speech and what to ignore.

Design

For developing the interfaces we decided to use PyQT Framework using Python. The computer running the application will be located near the physical vending machine and the users will find it in *recognition state*, with the interface showing a live view of the scene. Approaching the device, the user will appear on the screen and the recognition will start.

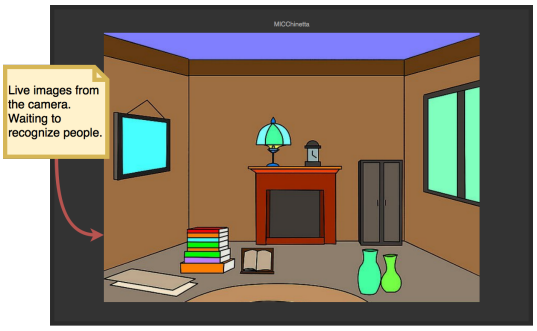

Figure 6: Recognition State: the App is waiting for users to show up

If a person is recognized, a rectangle will appear on the screen around the detected face but, only if the person is known (the identity is saved in the database), a label with the id alias will be shown. If the person is matched with the same identity for multiple consecutive frames, we consider the recognition successful and the system will enter the *conversation state* represented in Figure 8

Figure 7: Person is recognized, after a 10 frame period time green layer as a visual feedback will be shown

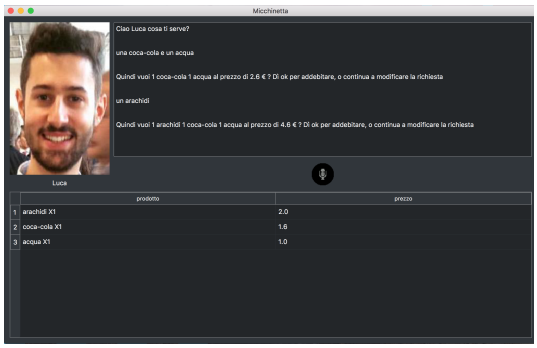

Figure 8: Conversation

As soon as the user is recognized, his reference image from the database will be showed along the matched identity and the conversation shall begin. While speaking, the conversation and the "bill" will be displayed in real time. If the user agrees to the final bill, the machine will commit the transaction and will change its state to idle (recognition state).

Add a new User

Along the main app, an utility app has been developed to help this system administrators to manage the Face Database. Running the script *DatabaseManager.py*, a list representing the current users will be shown; through the *Add user* window, it is possible to both drop an existing new face image, or even take a new photo with the web cam.

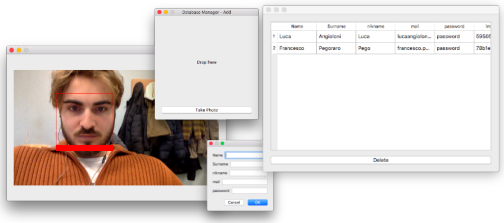

Figure 9: Database manager script

Understanding users needs

The awesome speech-to-text provided by Google API it's not enough to sustain a conversation. In order to speak and really understand what the user needs we developed a *Bot* class.

We used pyttsx package[2] for text-to speech and TreeTagger[3] for natural language processing (word tagging).

The bot starts by greeting the current user and waits for the user to talk. We explored Italian language to understand how users may ask for products at a vending machine and came up with a parsing process

that can see through the request. An example is shown below:

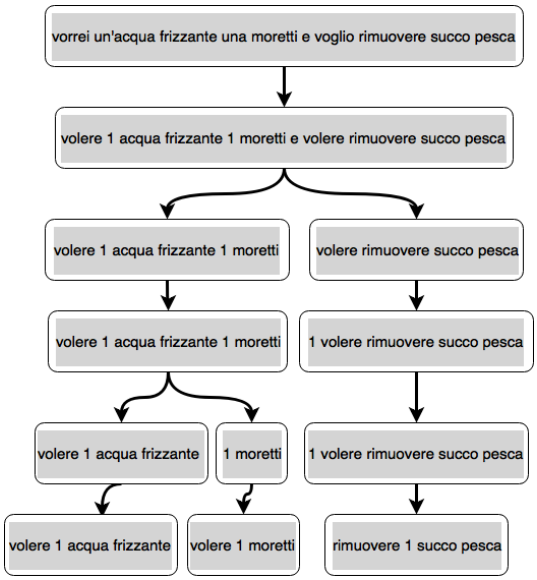

Figure 10: Parsing process (in Italian)

Usability Test

The scenario of these tests is a well known and restricted context: the MICC laboratory. However the usefulness of this system transcends from this and the results obtained are valid also in other similar scenarios.

Environment and tasks

We placed our system right by the vending machine at MICC, providing it an Internet connection (not really reliable that particular day due to works in progress) and installing all necessary software. We gathered all available staff, in couples or alone, and we presented them with a list of tasks, briefly explaining the application purpose and context.

The tasks that users had to perform are[2]:

- *To be recognized*: Present yourself in front of the machine and let it recognize you.
- *Wrong recognition*: Simulate a wrong recognition and explain it to the system.
- *Ask for a product*: You are hungry or thirsty, so you ask for a product.
- *Ask for more products*: You are very hungry or thirsty, so you ask for more than one product at once.
- *Remove a product*: You changed your mind, remove a single product.
- *Remove more products*: You changed your mind, remove many products.
- *Confirm the purchase*: You are finished, agree to the transaction.

After completing the tasks, we asked the users to compile a Google Form structured as a SEQ *(Single Ease Question)* with a 7-point rating scale:

1. I found the reply time too high.
2. I prefer using the mobile application instead of this system.
3. It is hard to use it.
4. The facial recognition works correctly.
5. The Graphic interface is pleasant.
6. I can trust the system.
7. MICChinetta can understand my request.
8. MICChinetta's replies are wrong.

Moreover we provided the users the chance to express some optional comments for each question.

[2]A copy of the tasks list was available to read for the users, while performing them

Results

Error trials: We gathered 13 users, each tried 7 task, so the total amount of trials is 91. Just 12 of these tasks failed, but only 2 resulted in a critical failure due to the lack of connection to Internet, the other 10 where due to fails in speech-recognition and where recovered just by repeating the request.

SEQ:

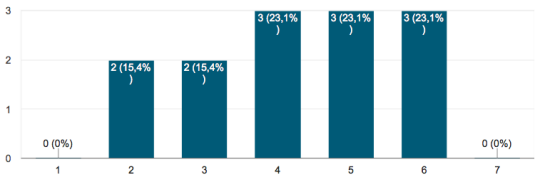

Figure 11: I found the reply time too high:1-low, 7-high

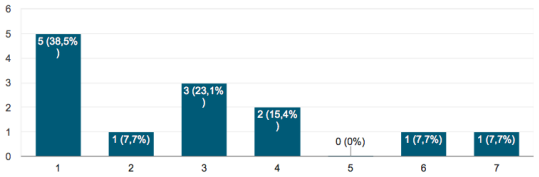

Figure 12: I prefer using the mobile application instead of this system: 1-Definitely no, 7-Definitely yes

Figure 11 shows that user's opinions vary when asked if the latency between MICChinetta's replies are too high. This may be linked to the fact that the reply time varies with respect to Internet connection. Some users however, acknowledge the computational efficiency required for speech recognition and pos-tagging, so their replies are biased.

Figure 12 shows that the majority of users prefers to use the VUI instead of the mobile App.

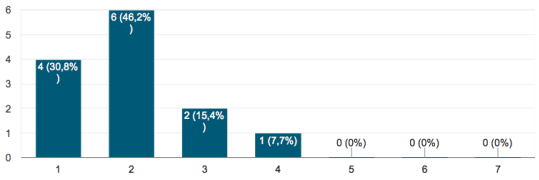

Figure 13: It is hard to use it: 1-Very easy, 7-Very hard.

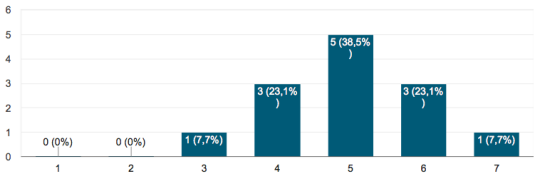

Figure 14: The Graphic interface is pleasant: 1-Very Unpleasant, 7-Very pleasant.

Figure 13 shows that most of the users found the system to be self explanatory, this is indeed crucial using a never-seen application. Moreover users agree about the pleasantness of the Graphical Interface demonstrated in Figure 14.

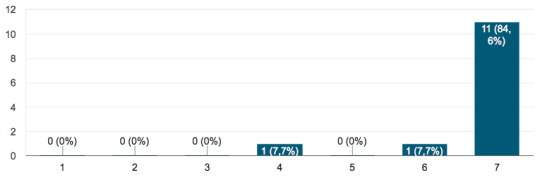

Figure 15: The facial recognition works correctly: 1-Definitely not working, 7-Definitely working.

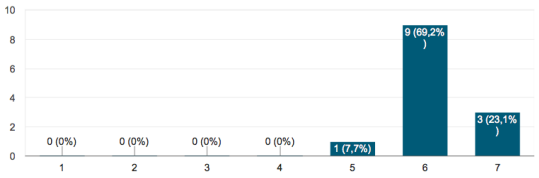

Figure 16: I can trust the system: 1-Completely unreliable, 7-Completely reliable.

Figure 15 confirms that face recognition was definitely working, this is a factor in the

trust placed by the user, reported in Figure 16, and confirmed in the next figures.

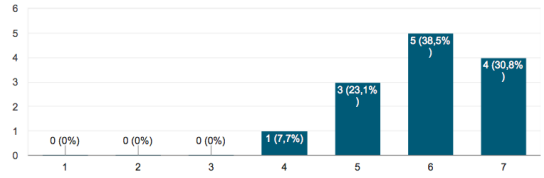

Figure 17: MICChinetta can understand my request: 1-Definitely not, 7-Definitely yes.

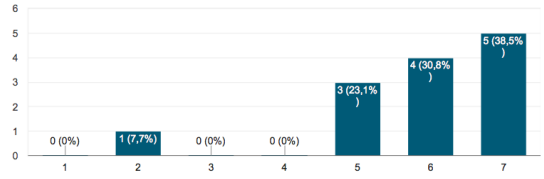

Figure 18: MICChinetta's replies are wrong: 1-Definitely yes, 7-Definitely not.

We discovered that users ask for products in many different ways: someone makes a robot-like list of products, while others use more complex sentences, like in a conversation at the grocery store. In both this cases, MICChinetta behaved greatly. Figure 17 and Figure 18 are a good demonstrations of the conversation skills of the system; it can understand what users ask and answers accordingly.

Conclusion

We presented *MICChinetta*, a VUI-GUI based system that provides assistance in the process of purchasing snacks and drinks at a vending machine. Systems like this are growing fast in popularity for the most various scenarios, like car driving, mobile phones and many more.

We demonstrated the useful functionality of being able to identify users using face recognition and the effectiveness of vocal interactions with an automated system. We discovered the possibilities for augmenting human-computer interactions provided by a prototype like MICChinetta, showing pros and cons of this new approach.

A downside was found in a substantial latency in the responses of the system, but being this a study and a prototype, this problem could and will be solved in future releases, maybe using better machines for carrying out all the computation in local, like speech-to-text, without relying over Internet connection.

Our final study confirmed that MICChinetta, but more in general VUIs, can do almost everything a GUI can do, not reducing tasks times, but giving the users different possibilities and experiences without using arms or hands.

We believe that pushing ourselves in this direction and improving this technologies, they will efficiently substitute standard interfaces where less and less interaction and effort from humans is required.

References

[1] Adam Geitgey. *Python Package for face recognition*. https://goo.gl/tvK4UE.

[2] Peter Parente. *Python package supporting common text-to-speech engines*. https://goo.gl/KLAHTD.

[3] Helmut Schmid. *POS Tagging library*. https://goo.gl/eeXFn3.

[4] Anthony Zhang. *Library for performing speech recognition*. https://goo.gl/eBv9m2.