Animatronics: Gesture Imitation with a Robotic Head

1st Carlos Álvarez Ruiz, 2nd André Cunha Enes Gonçalves , 3rd Guillermo Gil García
Universidad de Sevilla, Sevilla, España

***Abstract*—Animatronics have been a key part of the entertainment industry for more than three decades. In this paper we explore the options of animatronics by moving a robotic head based on the information of facial gesture capture with a camera, testing different integration approaches and finally selecting one that matches our desired behaviour.**

***Index Terms*—Botszy, Mediapipe, Animatronic, gesture capture, Pololu, Robots Can Cry**

I. Introduction

The entertainment industry has been strongly linked for more than forty years to the use of all kinds of animatronic robots for a wide variety of audiovisual works. Movies and television series in recent years have taken advantage of animatronics and motion capture to bring their characters to life. The first animatronic used in the cinema dates from 1964, specifically in the movie *Mary Poppins*, where a mechanical bird was used [1]. Since then, other big blockbusters such as *Jurassic Park* (1993), *The Lord of the Rings* (2001) or the Star Wars various films have largely used animatronics and motion capture to represent their characters, giving rise to a great evolution in the industry behind them.

In our project we integrate animatronics and motion capture, carrying out the complete process from taking images of a face by a camera to imitating its gestures by an animatronic head, including image processing and facial recognition necessary to act as an interface between the two. Both specialized software (MediaPipe, Botszy) and hardware (Pololu Maestro, Arduino) have been used for this purpose.

A. Original idea

This project is based on the previous work from our university colleagues Alejandro Letrado Castellanos and Eduardo Araguete Gamero, whose work was originally based on the work of Irene Luque Martínez. Their project integrated the Botszy platform, Autodesk Maya, and a camera to control a model in Autodesk Maya of the same robotic head that we have in reality, going through Botszy and using Python for image acquisition and OSC message sending to control these movements.

The OSC protocol is a widely used protocol in the entertainment business for musical performance management. Its acronym stands for Open Sound Control and is an alternative to the MIDI standard, since it provides higher resolution without losing the simplicity that characterizes these protocols.

The initial idea that we pursued was being able to manage the robotic head with the use of OSC messages sent from Python to Botszy, but integrating a real model of the head and using a Pololu Mini Maestro board to manage the servos. Although being more deeply explained in its corresponding section, we can note that Botszy is a non-officially released software, and after reaching out the developers the answer was given that the intended functionality was not yet implemented in their program. Therefore, it has been necessary to propose alternatives to carry out this project.

It should be said that the problem of facial features detection is highly resolved and therefore different solutions can be already found ready to be implemented into a project. Nonetheless, a solution with minimum response time has been seeked.

II. Hardware

A. Robots Can Cry Animatronic

The rise in the use of animatronics in the world of cinema has led to the appearance of companies dedicated entirely to this field. Among them, *Robots Can Cry* stands out. It specializes in creating animatronic characters, supervising them on filming sets, character development and puppetering [2].

The robot that promoted our project in the first place is an educational animatronic made by this company of Andalusian origin. This animatronic can be seen in Figure 1. It is a robotic head that has 9 servomotors that allow the implementation of complex movements. It is anchored to a structure created with 3D printing that allows great freedom of movement.

Although early development of the project was carried out with this robotic head, the need for an external power supply for the servos (in the form of a battery and a power distribution system) meant the change to another animatronic head.

B. AnimaTRON 1.0

The AnimaTRON 1.0 robotic head, developed by Waredrone [3], has been used for the final implementation of the project. Thanks to its integrated power system, which only

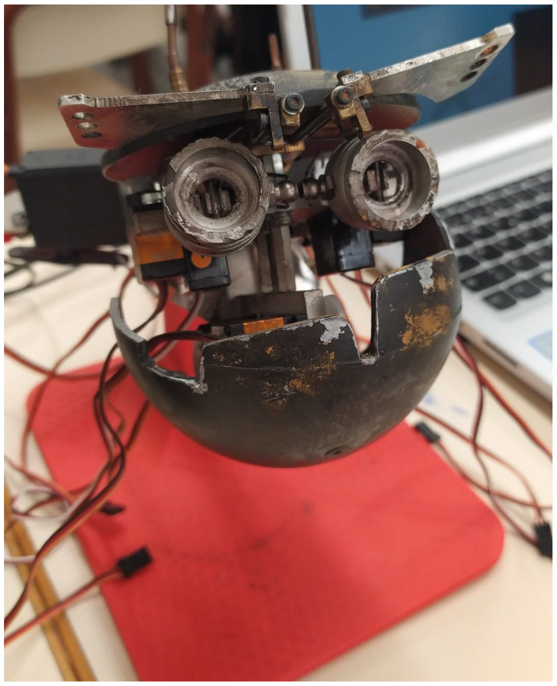


Fig. 1. Educational Animatronic by Robots Can Cry

needs a 5V/3A transformer with USB Type C output, and its pin connection block on the front of the animatronic, this head solved the problems caused by the other one. Figure 2 shows a general view of the device, in addition to the position of the inputs for the signal pins of the servomotors.

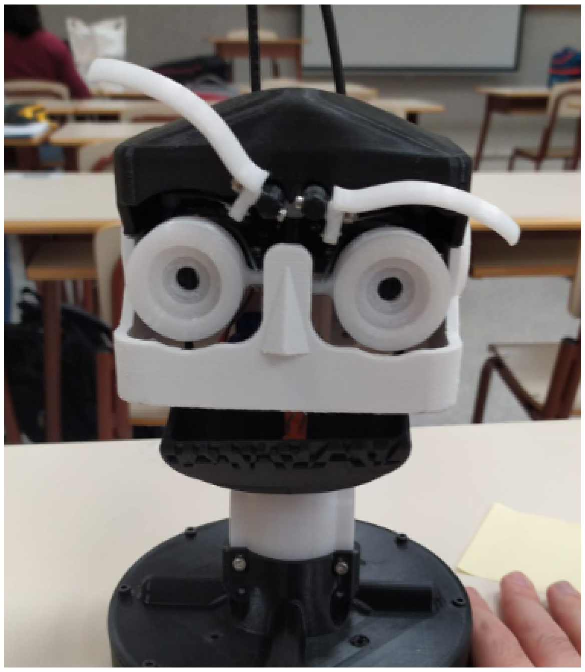


Fig. 2. AnimaTRON 1.0

C. Pololu Mini Maestro 12-Channel USB Servo Controller

The Pololu Mini Maestro 12-Channel USB Servo Controller is a fundamental piece of hardware implementing PWM control of the robotic head's multiple servo motors from one of its three input options: USB for direct connection to a computer, TTL serial for use with embedded systems (this is the option that was used in this project), and internal scripting for self-contained host controller-free applications. It's highly configurable and the 12 channels of this motion control module are enough to meet the needs of the project.

Here are some of the key features of the controller:

- Pulse rate configurable from 1 Hz to 333 Hz
- 250 ns output pulse width resolution (about 0.025º, far beyond what the servos can resolve)
- Individual speed and acceleration control for each channel
- Serial TTL supports 300 bps to 200,000 bps in fixed-baud mode

En la Figura 3 puede verse una imagen del controlador.

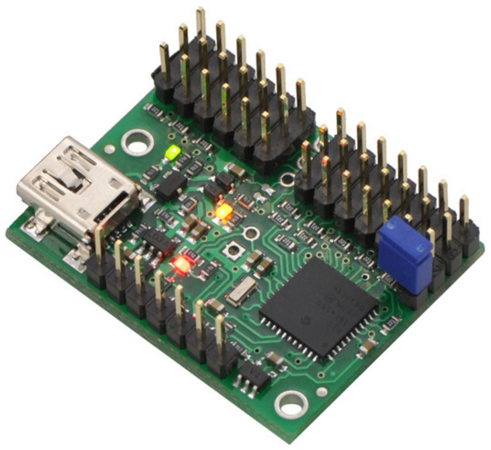


Fig. 3. Mini Maestro 12-Channel USB Servo Controller

D. Arduino UNO R3

The Arduino UNO R3 microcontroller board is widely known for its great versatility. It has 14 digital input/output pins (6 of which can be used as PWM) and 6 analog inputs [4], as well as being easily programmable, making it ideal for applications without complex timing requirements. In this project it has been used as an interface between the computer and the Pololu board, as it constitutes a simple solution when it comes to communicating signals to the servomotor controller board.

III. Software Implementation

A. MediaPipe

MediaPipe is Google's cross-platform framework for creating different data processing pipelines [5]. As an open-source project, the framework has readily available human feature tracking models which can be easily used through the framework's API.

For this project, the MediaPipe *Face Mesh* model was adopted due its ease of deployment when compared to *Dlib* since it can be used solely with *python* without the need of other dependencies. Other advantages over *Dlib*:

- the amount of facial landmarks that it is able to detect and track reliably, 468 *versus* 68 [9]
- *3D* point estimation, which allows to detail how the landmark's perspective shifts over time [8]

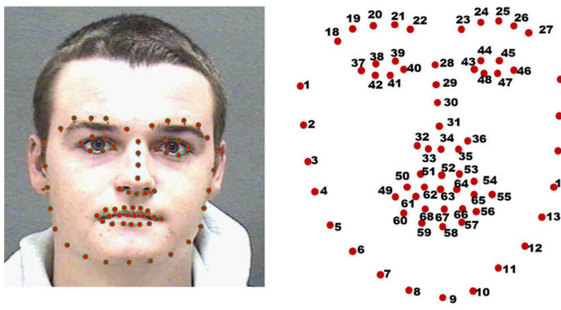


Fig. 4. *Dlib* and face landmark demonstration

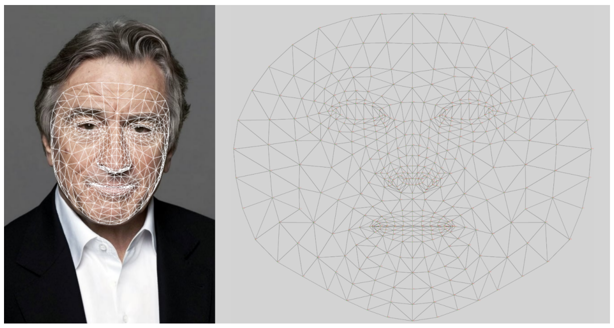


Fig. 5. MediaPipe *Face Mesh* landmarks

By accessing the coordinates of the selected points for the needed features (mouth, eyes and eyebrows), it is possible to deduce the states of these features, *e.g* open or closed [10]. In an ideal scenario, the distance between landmarks would be compared to a length reference, but since it is assumed that there are not known lengths in the video stream, this is not a viable strategy. The used approach consists of using near-stationary landmarks, with very slight movement between facial expressions compared to other points. This enables the use of a reference within the face model, which does not change when moving away from or closer to the camera and the distance between the reference points stays in the same scale as the rest of the face landmark's connections.

The chosen landmarks are part of the nose, since the skin of the nose only experiences very slight stretch and contraction within the full range of facial expressions, which in turn means that the distance between the landmarks that depict the nose also only varies by a small amount. In order to minimize the change in the chosen reference, the average values for its x and y coordinates were used, with a certain number of video frames as the time period for this calculation. Another reason for the use of the average value is that the landmarks' coordinates change from frame to frame, due to the model's uncertainty. A more precise way of applying this method would be to test the model by comparing the variance of the distance between possibly adequate points over a facial expression dataset that ensures the faces are the same distance from the camera and uses the same equipment on every picture, e.g. *JAFFE dataset* [12], *Radboud Faces Database* [11]. This method would provide the best possible reference points for the implementation in this project.

In a future installment, a different method of obtaining a length reference could be achieved by using another model from MediaPipe: *Iris*. This model tracks the subject's iris more precisely than *Face Mesh* and is able to determine the metric distance of a subject to the camera with less than 10% error, without requiring any specialized hardware. The measurement is done by relying on the fact that the horizontal iris diameter of the human eye remains roughly constant at 11.7±0.5 mm across a wide population, along with some simple geometric arguments. [7]

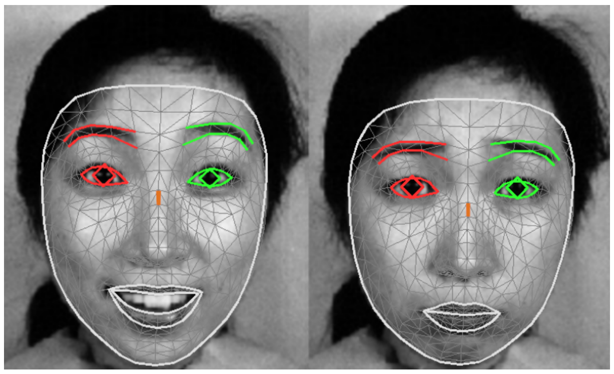
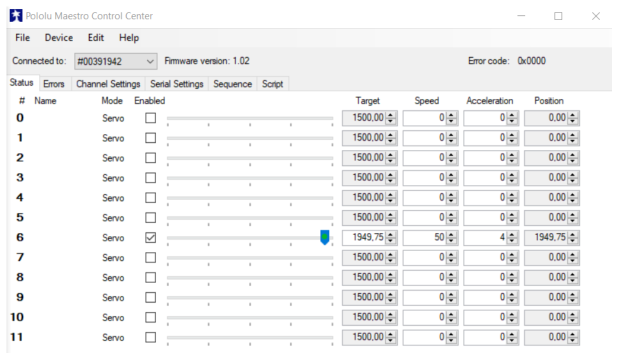


Fig. 6. Distance comparison across possible reference with expression change [12]

B. Pololu Maestro Control Center

This Pololu software allows you to view and configure the different channels, corresponding to each servo, that the connected board has available. It has a double function: on the one hand, it allows you to manipulate the position of each servo (enabling it first) by moving the slider of the corresponding channel, and it also allows you to visualize the position to which we are ordering the servo to go when we command the position by using of functions in our code. A previous serial port configuration is required depending on the communication that we are going to carry out. For the connection through Arduino, UART serial mode must be selected, setting the fixed baud rate at the same rate declared in the Arduino serial initialization. However, if a direct connection with the computer through the USB cable is intended, the USB Dual Port box must be checked instead.

The Polulu Maestrol Control Center interface can be seeing in the Figure 7.



Connected to: #00391942 Firmware version: 1.02 Error code: 0x0000

#	Name	Mode	Enabled	Target	Speed	Acceleration	Position
0		Servo	☐	1500,00	0	0	0,00
1		Servo	☐	1500,00	0	0	0,00
2		Servo	☐	1500,00	0	0	0,00
3		Servo	☐	1500,00	0	0	0,00
4		Servo	☐	1500,00	0	0	0,00
5		Servo	☐	1500,00	0	0	0,00
6		Servo	☑	1949,75	50	4	1949,75
7		Servo	☐	1500,00	0	0	0,00
8		Servo	☐	1500,00	0	0	0,00
9		Servo	☐	1500,00	0	0	0,00
10		Servo	☐	1500,00	0	0	0,00
11		Servo	☐	1500,00	0	0	0,00

Fig. 7. Polulu Maestrol Control Center interface

C. Botszy Solution

Botszy is a physical character rigging software that sets out to revolutionize the audiovisual industry, the animatronics industry and robotics. Its goal is to provide control and management tools for animatronic creatures through specialized software that meets the expectations of professionals in the sector (animatronic specialists and VFX operators).

The program focuses on creating a certain structure that resembles the real animatronic and creating an interface that allows control of the animatronic and monitoring of its status.

Being a program in development, a series of impediments were found when using this program for the development of the project:

- The installation of the program on most used laptops did not work
- On some of the computers where the installation was successful, the program did not launch
- On some occasions, if the program launched, it would not recognize the *Pololu Board*
- The communication function with OSC protocol does not work correctly as it is ”soft-programmed” at the date of completion of the project

For all this, it was decided to dispense with this specialized software and carry out the programming of the control of the robotic head servos manually.

D. Arduino Library for Servo Control

Pololu’s own manufacturers provide a library that allows control of their line of servo controllers from Arduino boards. This library allows us to set the speed, acceleration, and position of each of our servos managed by the Pololu Mini Maestro. In this way, an easy-to-use tool is provided that serves as plug and play for applications in which speed is needed at the time of configuration, i.e. the animatronics business in the audio-visual scene.

E. Arduino-Python integration

In order to connect the Python script, that captures the camera and detects the facial landmarks of interest, with the Arduino script, that moves the servomotors according to the displacement of the defined landmarks, the Python library Serial was used. This library allows data to be sent through the serial port, granting the Python program the status of publisher, pushing the servo positions through the serial port . In parallel, the Arduino scripts reads the serial port.

An extra step must be considered to achieve this integration. The communication between Python and the Arduino board is done through the physical serial port (hardware serial port). Since our board is an Arduino UNO equipped with a single UART, it only has one physical serial port. This implies that, since the communication from the Arduino with the Pololu board is also achieved through the serial port, and the only serial port is already being used, a second serial port must be synthetically added. Arduino provides the SoftwareSerial.h library for this task, allowing the user to add a software serial port using two pins on the board as the new RX and TX.

F. Purely-based Python solution

Although the previous solution easily solves the problem, it comes with a cumbersome assembly to say the least. Using two boards and a computer to control a robotic head suggests that the system can be further compacted.

Since the image acquisition and landmarks recognition script has to be run in Python in order to use the Mediapipe library, an alternative has been sought to dispense with the use of the Arduino board. To achieve this, the Python Maestro class is used, which, in a similar way to how it was done with the official Arduino library, allows the position, speed and acceleration of each servo to be established. Since our program runs on Windows, the port in which the Mini Maestro board is connected to has to be retrieved and specified to the Maestro class.

It should be clarified two connected ports will appear when plugin in the Pololu board, and so is the way the Pololu servo line deals with communication with the computer. A first port for the usage of functions (this will be the port that we will use) and another for communications. If our program were running on Linux, then the local path of the USB port must be specified instead.

Relative to the way the servo movement has been resolved, the following procedure has been followed:

- The distance between two points of interest is measured. For example, to find the aperture of the mouth, the difference between the upper and lower lip is calculated
- Since the calculated distance depends on how far the user stands from the camera, an urge to normalize it appears. To do this, the distance from the nose is chosen, which will not change, and the distance of interest is divided by

the distance from the nose. With this, we already have the normalized distance. The result is saturated so that if it exceeds the maximum value, the result is updated to be equal to this maximum value
- To transform this value to servo movement it has to be divided by the maximum value that this value presents (in the case of the mouth, 2.5), multiplied by the servo travel distance (4000, since the maximum is 8000 and the minimum 4000) and added to the minimum value or subtracted from the maximum value, depending on whether the minimum or maximum position of the servo is considered the resting position. For the case of the mouth, for example, the resting position would be 8000 since this is the position in which the robot has its mouth closed
- Using the Maestro class function, the corresponding servo is set to the required position

G. Movements

Since the movement is located in the face area, the monitored movements will be the following:

- Mouth movement: the movement of the user's mouth will be measured by the distance on the Y axis between the point of the upper lip and the point of the lower lip
- Eyebrow movement: this movement is measured by the distance between the upper point of the eyebrow and its corresponding eye
- Eye movement: since the iris points are not defined, it will be done by closing the eyelid of the eye that you want to move

H. Movements of the real model

The AnimaTRON 1.0 robotic face is actuated with eight servomotors. Each servo matches the following movements in the robot:

- Right eye X axis
- Right eye Y axis
- Left eye X axis
- Left eye Y axis
- Right Eyebrow
- Left Eyebrow
- Open and close mouth
- Forward and back mouth

In order to match the monitored movements, the selected real model movements that will be commanded to the robot will be the following:

- Right eye X axis
- Left eye X axis
- Right Eyebrow
- Left Eyebrow
- Open and close mouth

IV. Results

After finishing this project, we managed to build a system that performs the identification of facial features with the help of Mediapipe and an integrated webcam. This program calculates the action of the servomotors necessary to match these measured movements and by using the Master class it places each servomotor in the corresponding position. The final implementation has a very short delay, and that was one of the main objectives of our whole project.

Therefore, we have a robust and relatively fast system that, with proper prior calibration, could be able to control different systems with servos integrated based on the movement of facial muscles. In this case, the controlled system was the 8-channel robotic head designed first by Robots Can Cry and adapted afterwards by Waredrone.

V. Conclusions and Future Improvements

Our project has achieved the expected result, having minimum delay and being the action time only limited by the speed at which the servomotors are being commanded, with the consequences that this entails to the hardware.

A possible improvement surely is to add the location of the pupils in order to move the robot's eyes more naturally, instead of associating this movement with a one-dimensional pattern as it is done now.

It can be seen with the naked eye that this project is very scalable, so it could perfectly implement the motion detection of a whole body and control a real robotic body with the actions of some model, using the same information flow as our current project.

There is no need to be an expert in the field to believe that Botszy can have a big impact in the animatronics industry, so we would like to try again and integrate this project with Botszy in the future once Botszy has its first official release and fixes the shortcomings and problems that we have found and remarked on this project.

VI. Acknowledgment

We would like to thank all the people who have made this project possible: our professor Jose María Maestre Torreblanca, our colleagues Irene Luque Martínez, Alejandro Letrado Castellanos and Eduardo Araguete Gamero and the people from Robots Can Cry and Botszy for their kind words and instructions.

References

[1] Hisour. *https://www.hisour.com/es/animatronics-43060/*
[2] Robots Can Cry RRC. *https://www.robotscancry.com.*
[3] Waredrone. *https://www.waredrone.com/animatron*
[4] Arduino Docs. Arduino Documentation. *https://docs.arduino.cc/*
[5] Xataka. *https://www.xataka.com/realidad-virtual-aumentada/moviles-capaces-detectar-objetos-3d-tiempo-real-google-hace-posible-a-traves-su-ia*
[6] Botszy *botszy.com*

[7] *https://google.github.io/mediapipe/solutions/iris.html*
[8] *https://arxiv.org/abs/2006.10962*
[9] *https://arxiv.org/abs/1907.06724*
[10] *https://arxiv.org/abs/1907.05047*
[11] *https://rafd.socsci.ru.nl/RaFD2/RaFD?p=main*
[12] JAFFE dataset *https://paperswithcode.com/dataset/jaffe*