IMPERIAL COLLEGE LONDON
Department of Mechanical Engineering

Design of a Hybrid-Expressive Face for a Social Robot

FINAL REPORT

04 June 2020

Abstract

Research conducted in long-term care shows a social robot decreases behavioural symptoms in people with dementia. This project improves upon the current design and software packaging of a social robot's expressive face in preparation for testing on dementia patients in India. The previous face generated several facial expressions by adapting two well-known approaches: a categorical approach and a three-dimensional affect space approach.

Firstly, in preparation for future open-source collaboration, the system code is entirely rewritten and restructured from Max to a simplified Java-based programming language called Processing. The previous seven sub-systems architecture is compressed into one Processing file which carries out all the functions of the previous project, improving ease of use. Secondly, the face design is upgraded to incorporate increased flexibility (degrees of freedom) of the lips to portray asymmetric facial expressions and a global control panel is designed for the end user to toggle any of the key features of the face. Additional parameters are defined to provide a greater degree of user control of the face. Thirdly, a new 'puppeteer' function extracts facial landmark values of a user from a camera feed and renders the robot's face in real time. An open-source convolutional neural network (CNN) model, pre-trained on the well-known FER-2013 dataset, is explored as an initial method of emotion recognition from a camera feed. After adjusting hyperparameters, the model is retrained over the same dataset leading to 66.5% accuracy.

Finally, a recognition experiment (Experiment 1) was conducted with 15 participants who were asked to identify the social robot's expressions. Importantly, expressions - happy, angry, sad and surprised were all recognised perfectly (100% recognition rate) but disgust, fear and stern had poor recognition (56.2%, 31% and 24% recognition respectively). Experiment 2, which tested the recognition of participant's expressions, by the CNN model, revealed that the CNN model often mistakes disgust as fear/surprise (only 38% recognition), whilst the other expressions were recognised effectively (over 85% recognition). Further areas of exploration could include 3D animation software for increased realism features and training the CNN model with different facial expressions datasets.

Acknowledgements

I would like to thank Dr. Ravi Vaidyanathan and co-supervisor Maria Raposo de Lima for their support and guidance throughout the project.

I would also like to thank Danny Bozo for providing access to his previous work on the project.

1 Introduction

Social robots are programmable artificial intelligence systems designed to interact with humans and each other in a socially acceptable fashion, conveying intention in a human-perceptible way and are empowered to resolve goals with fellow agents [1]. The field of social robotics has experienced widespread growth in recent decades, with increased focus on robots that can interactively provide humans physical and cognitive assistance [2]. This project explores modifications to the design and software packaging of an expressive face of a social robot. The original BERT2 robot hybrid face was designed computationally using a visual programming language called Max. While Max makes it easy to interact with different objects linked together, built at the Bristol Robotics Laboratory in 2010.

Background and Literature Review

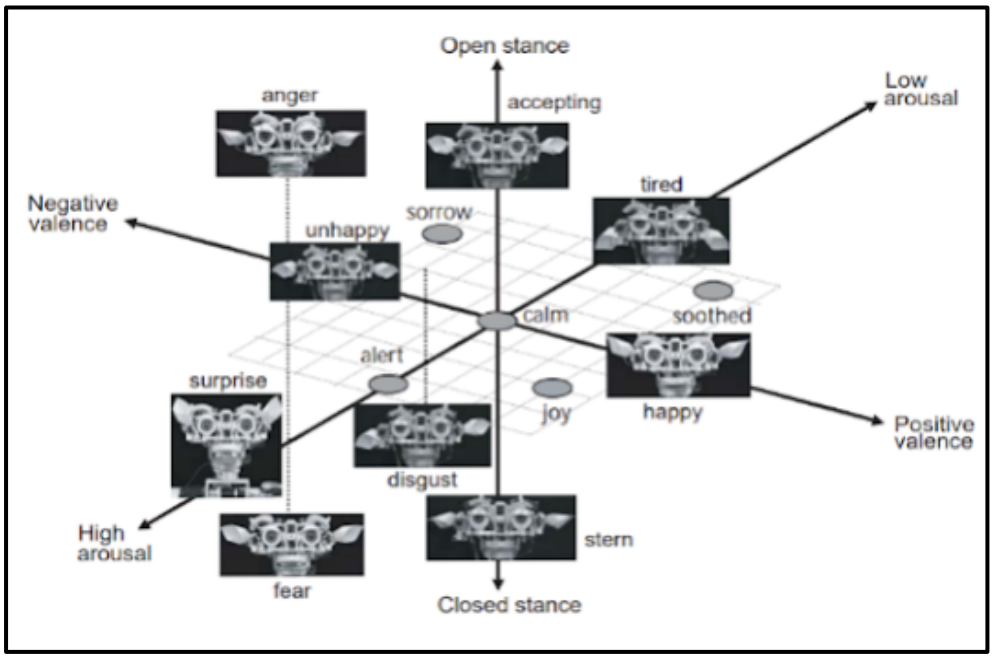
Social robots are programmable artificial intelligence systems designed to interact with humans and each other in a socially acceptable fashion, conveying intention in a human-perceptible way and are empowered to resolve goals with fellow agents [1]. The field of social robotics has experienced widespread growth in recent decades, with increased focus on robots that can interactively provide humans physical and cognitive assistance [2]. Assistive social robots could particularly play an important role with respect to the health and psychological well-being of the elderly [3]. It was noted that 'companion' robots can successfully help patients affected by dementia by helping them communicate their emotions, reduce their anxiety and improve their mood states [4]. Notable researchers in social robotics include Cynthia Breazeal, Hiroshi Ishiguro, Brian Scassellati and Dean Weber. The majority of modern 'affective computing' (computing that relates to, arises from, or influences emotion [5]) is influenced by their previous work.

'*Affective Computing'* is a term coined by Rosalind Picard [5], defined as computing that relates to, arises from, or influences emotions. This was termed with the goal of designing a system that *recognises* and *expresses* affect. Picard considers the role of emotion as essential for enhancing performance of assistive technologies and for enhancing the system's decision making [5]. The ideal behaviour of a social 'affective' robot would therefore be interactive expressions of affect based on accurate affect recognition.

Interactive Expressions

While several methods of communication contribute to spreading emotions/affective information such as vocal tone, body posture etc, Schiano, Ehrlich, Rahardja and Sheridan believe that facial expressions are the *primary* mode of communication of affective information [6] [9]. Therefore, in developing social robots, facial expressions have been a large focus of research. Kismet was an early robot experiment by Dr Cynthia Breazeal, it displayed 8 different emotional states – 'Happiness, Sadness, Surprise, Tired, Anger/Stern, Extreme Anger Disgust, Fear"[7]. Gockley's Valerie and Nourbaksh's Sage are also well known social robots with faces defined as a combination of key basis emotions [20,21] A majority of social robots are built upon the well-reputed definition of seven 'universal facial expressions' of a *categorial nature* by Dr Ekman [8] – happiness, sadness, anger, fear, disgust and surprise. Dr Ekman broke down these basis expressions further into individual components of muscle movement, called action units [AUs]. All the AUs are now defined in the Facial Action Coding System (FACS) [10], which is widely considered as the most refined method for measuring the specified categorical emotions by facial actions. While it's proposed that the specified set of AUs categorise the universal emotions extremely well, subsequent empirical research have proven that FACS works well only for certain emotions. *Galati et al. (1997)* [11] asked fourteen blind and fourteen sighted people to produce six of Dr Ekman's universal set of expressions. The results showed that the theoretically predicted AUs appeared to occur frequently for some emotions such as anger, happiness, surprise, but not for others such as fear, disgust or sadness. *Scherer and Ellgring (2007)* [12] also carried out a similar test with 12 actors in Germany and concluded that several expressions didn't display the AU units predicted in theory. However, given the reference to FACS as the basis for robot expression generation in

several successful research papers such as *Marcos, S et al. (2016)*, *Dr Scassellati et al. (2001)* and *Dr Breazeal (2000) [13,14,15],* FACS's action units can be presumed to be a reliable guide for defining the base expressions for social robots.
While Kismet may appear to be an adaption of Dr Ekman's categorical expressions with 'stern' instead of contempt and 'tired' as an addition to the universal set, Kismet's emotions are represented by an alternative theory of facial expressions – the three dimensional, continuous affect space [7] shown in Figure 1.



This theory is based on an accepted model from previous work by Russell and Fernandez-Dols [21] which posits that most instances of human emotions could be understood within the two general dimensions of *valence* and *arousal.* However, further work since has proven that 2-dimensional models provide biased insights on emotion conception and are unable to clearly disambiguate emotions such as anger, fear or disgust [22,23]. *R.Trnka et al. (2016)* posits that based on their results, there are four dimensional input measures (valence, intensity, controllability and utility) which represent clearly different qualities of discrete emotions and based on this data, a 3D hypercube-projection was produced. [22]. Older work by *Kleinsmith et al. (2007)* also suggests a three-dimensional affect space represented by valence, arousal and *action tendency*.[23]. Kismet successfully used this 3D affect space with a label of 'stance' for the third axis, which is comparable to the action tendency dimension by *Kleinsmith et al.* [23].

3. Software

3.1 Software Selection

In the previous version of this project, Max/MSP was chosen as the best platform for rendering the 3D face due to the design advantages for multimedia purposes. The programming language is mostly used for music programming to build synthesizers. However, it had tools to interface and create basic 3D openGL (Open Graphics Library) graphics objects. Max uses this graphical interface of objects linked by signal paths in combinations known as 'patches' to carry out complex functions. While this is a simple way of producing graphics, the language and interface is relatively new to most users and therefore can be confusing at first. In preparation for collaborating with other universities, it was important to move to a software package based on a more popular programming framework, making it easy to use with minimal training.

My initial research led me to believe that JavaScript was the ideal language for producing graphics for the expressive face. JavaScript is popularly used to animate objects on web platforms and has several library resources/APIs such as 'WebGL' to create realistic 3D animations. However, further research led me towards the use of 3D animation software in creating complex facial meshes capable of being very realistic. There are several powerful animation tools such as Maya and Blender with scripting capabilities which essentially allow users to freely program advanced 3D animations. These software are used in industry for film animations, gaming and increasingly in AR/VR environments. Using advanced animation software with deep learning approaches has led to exciting research into virtual agents/bots with human faces. The chosen initial approach was therefore to learn to animate a custom facial mesh within Blender (which has an in-built IDE for Python scripting) with 'shape keys' corresponding to the categorical and affect space emotions. Shape keys are modifications of a face mesh's vertices which can be animated in a timeline or controlled via the in-built Python scripting. With little experience in Blender, I was able to design an extremely simple 3d face mesh which was then exported and saved. The next step involved looking into useful deep learning approaches such as the use of GANs (Generative Adversarial Networks) to produce realistic speech-driven facial animation[22] or the use of models such as RingNet [21] which can estimate a set of facial expressions from a single image of a person. Speech-driven animation produces realistic mouth movements based on phonemes and automatic expression generation based on several factors such as vocal tone, word choice etc. This would have been an impressive upgrade to the current version of the hybrid-robot face. However, according to the *uncanny valley* theory posited by Mr Masahiro Mori, the more realistic a robot/avatar, the more its imperfections appear strange and monstrous to humans. [24] "When tiny dissimilarities of photorealistic avatars become evident… rather than producing improved empathy…they will tend to inspire mistrust/revulsion" [24]. Thus, a certain threshold of realism should be held back. Therefore, most successful social robots use faces designed in between cartoon-like animation and photorealistic avatars.

Since Blender is an advanced tool that would require significant training in a collaborative setting, and with the focus on past papers [25][26] being on the categorical and affect space interpolation equations, it was decided to revert back to 3D openGL graphics animations to maintain consistency with the previous face.
Using this method is desirable for the eventual implementation of more sophisticated human to robot interactions, as shown in work by Gockley and Nourbakhsh. Much like the previous work, this method relied on writing the code for basic geometric shapes such as circles, ellipses and arcs to build a very simplistic yet friendly and sociable face, and interpolating between expressions using a basis vector matrix with varied weightings. Although JavaScript was a good tool to start with, ideally a library or package that incorporated visual programming in JavaScript or Python was needed for reducing the complexity of the code. After doing some research, I discovered Processing - an open source graphical library with a well-built IDE made for applications in electronic arts, new media art and visual design. The entire reason for moving away from Max was that the code was inherently difficult to understand. While the software package did allow for simplifications in arranging and connecting building blocks of objects within a “patcher” or visual canvas, for more complex code it became increasingly difficult for the user to defragment these building blocks and understand the underlying connections. Processing, on the other hand, uses the well-known Java language with additional simplifications such as additional classes and a graphical interface for simplifying the compilation of this code. Processing’s IDE has a ‘JavaScript’ mode to create web-based animation if needed.

3.2 Software Architecture

The previous system architecture consisted of several sub-systems communicating with each other. Using MAX/MSP, previous students designed seven programs that interact with one another using a UDP (User Diagram Protocol) network interface protocol. The output data from five of the programs would be received by faceController.maxpat where a final thirteen DOF values are outputted for the expression to be rendered.

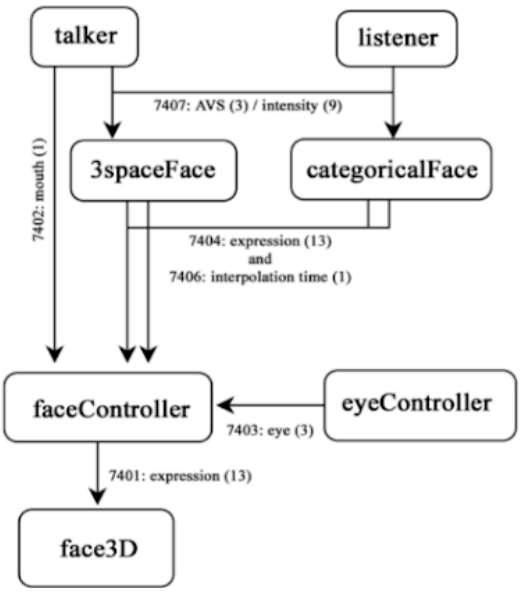


Figure 2. The original 13 degrees of freedom of the face.

While this may be convenient for running each program as a sub-system on different computers and connecting the computers through the same network, in reality it's noted in ***Vaidyanathan, Bozo et al***. that the recommended hardware configuration is a single computer running all the programs. Therefore most of these sub-systems could be compressed into 1 program (a 'control panel'), whilst still having another program to render the face on another device within the network (most like the real scenario – a nurse or student controlling the face on their laptop while the patient can only view the final rendered face on a tablet). In addition to carrying out the functions of the seven programs, this project also explored video based control of the robot's face which in this case requires 2 additional programs (FaceOSC and Emotion_recog.py). The remodelled architecture with the aforementioned changes is displayed in Figure 3 below.

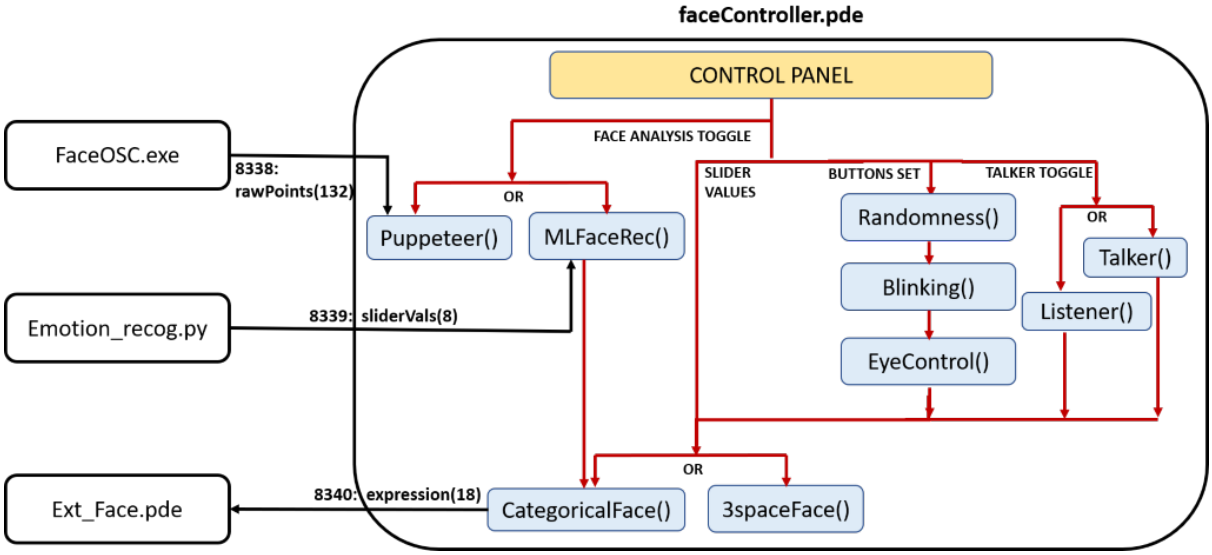


Figure 3. Re-modelled architecture

The faceController.pde file contains the entirety of the 7 programs/patches previously used. The blue boxes represent functions within the program. The graphical control panel consists of sliders, radio buttons, text inputs to allow the user to input the required data into the program. The slider values correspond to the amount of 'happiness', 'sadness'…etc in the categorical case or the amount of 'valence, 'arousal or 'stance in the 3-space affect space case. A set of buttons correspond to animated features such as randomness, blinking and controlling the pitch and yaw of the gaze. The talker toggle button turns the 'talking/listening' feature on and off, but when switched on - there's a button to choose between running the function Talker() – which opens the mouth every time a certain threshold of sound is reached, and Listener() – which runs through a set of pre-programmed facial expressions every time a certain threshold is reached.

The three external programs send and receive information from the faceController file using OSC (Open Sound Control), a protocol for networking computers and multimedia devices, over specified ports for each program. OSC messages are transferred across the Internet or within sub networks using UDP.

4. Design Process

Face Detection

Face detection is the primary step for any FER system. For face detection, Haar cascades were used (Viola & Jones, 2001). The Haar cascades, also known as the Viola Jones detectors, are classifiers which detect an object in an image or video for which they have been trained. They are trained over a set of positive and negative facial images. Haar cascades have proved to be an efficient means of object detection in images and provide high accuracy. Haar features detect three dark regions on the face, for example the eyebrows. The computer is trained to detect two dark regions on the face, and their location is decided using fast pixel calculation. Haar cascades successfully remove the unrequired background data from the image and detect the facial region from the image. The face detection process using the Haar cascade classifiers was implemented in OpenCV. This method was originally proposed by Papageorgiou et al, using rectangular features which are shown in figure 3 (Mohan, Papageorgiou & Poggio, 2001; Papageorgiou, Oren & Poggio, 1998).

Face Rendering – similarities and differences

The primary features of the face are the eyebrows, eyelids, eyeballs and mouth. The original face allowed for a total of 13 degrees of freedom **(Figure 4),** however with adaptations made to the lips to make it more flexible, the new face can allow up to 14 extra degrees of freedom which have been integrated into the basis set of expressions. The additional 14 degrees of freedom allow for asymmetrical expressions of the lip which can be useful in expression states such as disgust, scared or tired. For the purpose of consistency with the original project, the user can decide the level of control complexity – choosing whether to input 13 degrees of freedom values as previously designed or the combined 27 degrees of freedom values for greater control. To reduce dimensionality, the additional 6 vertices are input as 6 location vectors (**figure 4 ii**) rather than directional scalars .

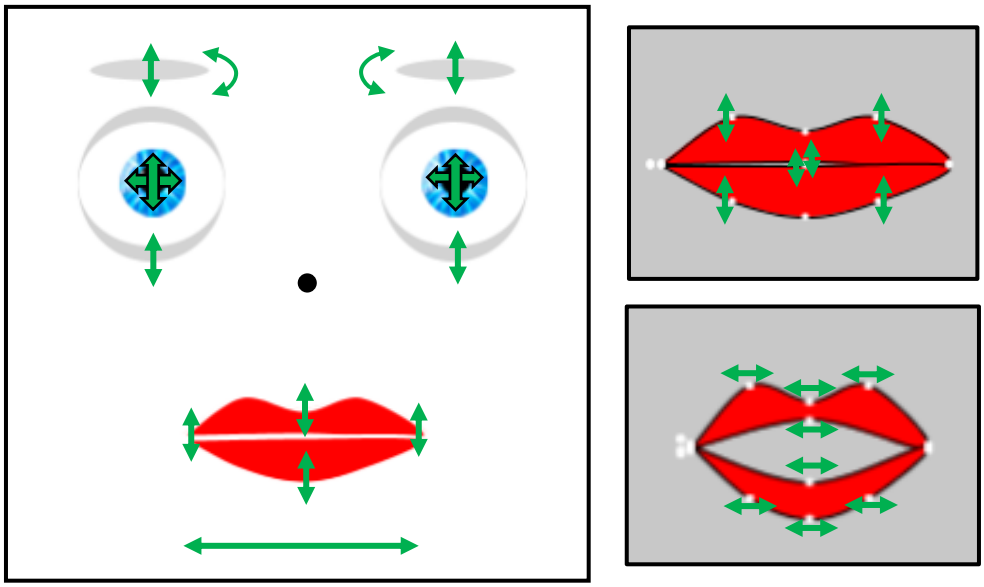


Figure 4. The original 13 degrees of freedom of the face.

In addition to this, I've also added a **translation** variable, $tr_{x,y}$ **(figure 4),** for the user to control the location of the centre of the face on the display screen and a **scaling** variable for each of the facial features **(Table 3)**, which lets the user expand and contract the features without manipulating individual variables. While the translation and scaling variables are set constant across all expressions, combinations of the remaining thirteen variables from **figure 4** and the additional 14 degrees of freedom from **figure 4 ii** characterise the facial expression state $\vec{e}(t)$ at a given time t.

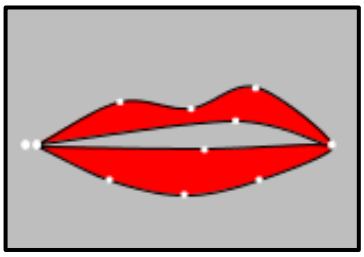


Figure 4 ii . Three additional vertices in both the top and bottom lip result in an additional 14 degrees of freedom being available.

Variable	Input Range	Effect on face (of increasing)	OpenGL Range
B_{ar}	[0,100]	AC rotation of left eyebrow	[π, -π]
B_{hr}	[0,100]	Lower left eyebrow	[-180,-100]
B_{al}	[0,100]	C rotation of right eyebrow	[-π, π]
B_{hl}	[0,100]	Lower right eyebrow	[-180,-100]
L_l	[0,100]	Close left eye lid	[80,0]
L_r	[0,100]	Close right eye lid	[80,0]
E_t	[0,100]	Gaze □ ∞	[0,50]
E_p	[-90,90]	Gaze □ Down	[π/2, -π/2]
E_y	[-90,90]	Gaze □ Left	[π,0]
E_d	[0,100]	Pupil Dilation	[0,30]
M_h	[0,100]	Raise mouth corners	[30,70]
M_w	[0,100]	Widen mouth	[60,190]
M_{ty}	[0,100]	Raise top lip centre	[15,-15]
M_{by}	[0,100]	Lower bottom lip centre	[-15,15]

Neutral Expression	
Input	**OpenGL**
50	0
50	-140
50	0
50	-140
80	64
80	64
100	50
0	0
0	π/2
50	15
50	50
60	140
50	0
50	0

It's important that the values of the variables in the expression state $\vec{e}(t)\vec{e}(t)$ are straightforward, in order to simplify work for future development. The variables in $\vec{e}(t)$ $\vec{e}(t)$ were chosen to have the same ranges as in previous work and linearly scaled to the native units of measurement in openGL (radians for rotations, pixels for distances). **Table 1 i** shows the original input ranges used in the face, their scaled-up openGL ranges and the visual impact of increasing the value of the respective variable. **Table 1 ii** shows the basis values set for a neutral expression, these values sit in the centre of the majority of the input ranges. **Table 2** shows the additional variables defined in $\vec{e}(t)\vec{e}(t)$ for controlling asymmetric lip movements and again linearly scales these to openGL ranges. When combined, the facial expression state is thus a function of 15 constants and 6 vectors defined as:

$$\vec{e}(t) = [B_{ar}\ B_{hr}\ B_{al}\ B_{hl}\ L_l\ L_r\ E_v\ E_p\ E_y\ E_d\ M_h\ M_w\ M_{ty}\ M_{by}\ M_{tx}\ M_{bx}\ \overrightarrow{M_{t1}}\ \overrightarrow{M_{t2}}\ \overrightarrow{M_{t3}}\ \overrightarrow{M_{b1}}\ \overrightarrow{M_{b2}}\ \overrightarrow{M_{b3}}]^T$$

Table 1 i – Values of the 13 DOF inputs to the face mapped via linear scaling to the openGL range,

Variable	Input Range	Effect on lips (of increasing)	OpenGL Range	Neutral	
				Input	OpenGL
M_{tx}	[0,100]	Shift top lip centre left	[45,-45]	50	0
M_{bx}	[0,100]	Shift bottom lip centre left	[45,-45]	50	0
$\overrightarrow{M_{t1}}$	[(0,100), (0,100)]	Shift top vertex 1 up and left	[(30,-30),(25,-25)]	50	(0,0)
$\overrightarrow{M_{t2}}$	[(0,100), (0,100)]	Shift top vertex 2 up and left	[(30,-30),(25,-25)]	50	(0,0)
$\overrightarrow{M_{t3}}$	[(0,100), (0,100)]	Shift top vertex 3 up and left	[(35,-35),(45,-45)]	50	(0,0)
$\overrightarrow{M_{b1}}$	[(0,100), (0,100)]	Shift bottom vertex 1 down and left	[(-30,30),(25,-25)]	50	(0,0)
$\overrightarrow{M_{b2}}$	[(0,100), (0,100)]	Shift bottom vertex 2 down and left	[(-30,30),(25,-25)]	50	(0,0)
$\overrightarrow{M_{b3}}$	[(0,100), (0,100)]	Shift bottom vertex 3 down and left	[(-35,35),(45,-45)]	50	(0,0)

To enlarge the eyebrows (drawn as ellipses), the width and height parameters (undefined in the 13 original variables from previous versions) of the ellipse are each multiplied by the input scale factor. To enlarge the eyeballs (drawn as spheres), the width, height and radius (undefined in the 13 original variables) are each multiplied by the input scale factor. To enlarge the mouth (free drawn based on vertices), the previously defined variables for the mouth vertices (Table 1i, Table2i) are multiplied by the input scale factor.

As mentioned earlier, with Processing, all the functions were written under one program and divided into tabs for ease of access. When the function is run, the standard neutral expression is rendered from the predefined basis vector and the user can alter any of the above 18 variables by typing into a textbox or by adjusting the categorical and affect space slider bars mentioned in the following sections.

Table 3 – Additional translation and scaling parameters (typically fixed for all expressions)

$\lvert tr_{x,y}$	[(0,100),(0,100)]	Shift entire face down and left	[(0, 700),(1000, 0)]
$\lvert B_{sc}$	[0,4]	Enlarge eyebrows	[(0,0), (520,104)]
$\lvert E_{sc}$	[0,4]	Enlarge eyeballs (spherical)	[(0,0,0), (152, 160, 152)]
$\lvert M_{sc}$	[0,4]	Enlarge mouth	[(0,0), 4* $\lvert[M_h\ M_w\ M_{ty}\ M_{by}]^T$ $\lvert[M_h\ M_w\ M_{ty}\ M_{by}]^T$]

Calibration/Hardware

Figure 5

The face is designed to match up to a plastic faceplate designed for the social robot. The previous faceplate in use is pictured in figure 5. As the faceplate's design is subject to change in the future, it's important to make sure that the facial features can be flexibly placed and scaled up.

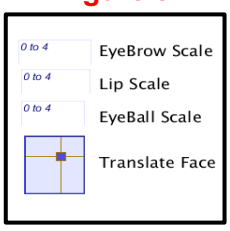


Figure 6

In the previous version of the project, the position of the face was calibrated in a separate text file 'facialPositionCalibration'. To make this process more convenient in this version, the variables defined in Table 3 can be altered in the user friendly control panel with sliders and text fields. Therefore any user can translate and scale up the face **(figure 6)** for future changes made to the robot faceplate.

Facial features design/variable extraction

In Processing, the render modes P2D and P3D make use of openGL to construct 2D/3D sketches. The shapes of the four features were redesigned with the primitives available in Processing.

- **Lips –** The lips were initially rendered using a combination of three semi-circles. The key expression values were stored in an array and interpolation between these values resulted in animated expression changes. A slider's output was rescaled to vary between 0 and 1, corresponding to the interpolation percentage between 2 expressions. The final version of the lips were rendered by creating a custom shape for the top lip and bottom lip each around 7 vertices. Processing's curveVertex() function draws curves on the vertices based on the initial and last vertex (the control points). The lip shapes of the 8 different expressions at their extremes, based on FACS' guide, were made using the curveVertex() tool. These values were then stored in an array and interpolation between these array values resulted in the animation. This process is detailed in the Categorical Expressions Section of this report.

Figure 7 – Initial Lip

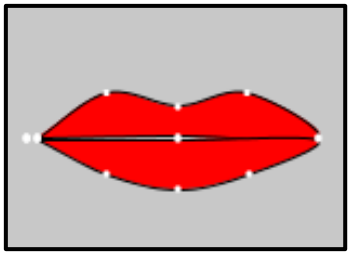


Figure 8 – Vertices

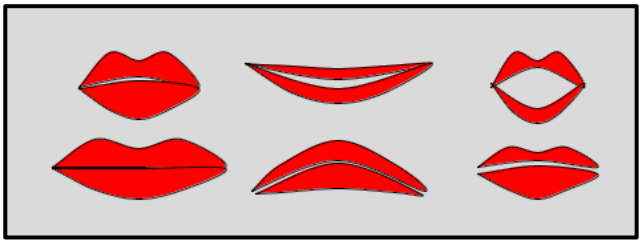
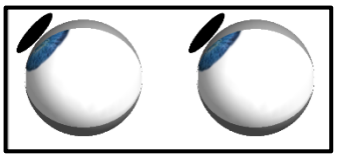


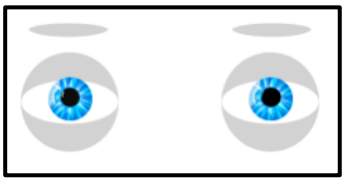
Figure 9 – Expressions Design Process

- **Eyes -** The eyes were rendered in 2D and 3D, leaving the choice to the user. For the 3D case, the eyes were designed as 'ellipsoids' with a defined centre location, rotation about X, Y and Z, width, height and radius. There were noted difficulties producing the 3D spheres/'elllipsoids' and these were consistent with the previous version of the project. Since the pupils are overlayed on top of the sphere, the gap between the pupil and the sphere is notable at the extremes which isn't representative of standard eyes. The 2D version, on the other hand, loses out on realistic eyeball movements as there's no rotation of the eyeball. However, the 2D ellipses are significantly easier to combine without any notable overlay of pupils.



As explained in the previous work, in the 3D case, a simple coordinate transformation is required In order to control the eyes. The local openGL coordinate system is slightly different than the one previously used (specified in Figure 14).

- **Eyelids** – The 2D eye lids are made as an overlap of the eyeballs over larger grey ellipses. Decreasing the height of the eyeball to nearly zero results in a blink. The 3D eye lids are rendered as grey quarter-sphere shells each which can rotate in opposite directions up to ninety degrees. This is exactly the same as the previous version of this project. There isn't a significant variation in eyelid height across expressions, only 'tired' relies on shutting eyelids slightly from FACS. However, happiness and anger could also include some eyelid movement as people tend to squint when extremely happy or extremely angry.



- **Eyebrows** - The two eyebrows were rendered by simply drawing two grey textured ellipses (figure 11) with a defined centre location, rotation angle, width and height. The y-value of the centre location corresponds with the original degree of freedom for the eyebrow height $B_{hr}|B_{hr}/B_{hl}|B_{hl}$, the rotation angle corresponds with the degree of freedom for the eyebrow rotation $B_{ar}/B_{al}|B_{ar}/B_{al}$, and the width and height are the output of the linear scaling by $B_{sc}|B_{sc}$. The width-height ratio was chosen such that the eyebrows would look as realistic as possible. Unlike the previous version of the eye brows, there's no limitation on rotation range, allowing for the user to alter the values as they please between -π and π.

[Equation]

By extracting the relevant input variables needed for openGL shapes, we can split the expression into vectors for the different facial features:
A brow - ellipse() is defined by: angle (given in [Equation]), y-location (given in [Equation]), x-location (fixed), width and height (ratio fixed, values multiplied by [Equation])

[Equation]

By using FACS as a basis, the above expression vector for the brows was formed for each of the 8 categorical
expressions: [Equation], [Equation], [Equation], [Equation], [Equation], [Equation],[Equation], [Equation], [Equation]. A categorical interpolation is simply carried out in this way for each pair of vector expressions:

[Equation]

Randomness

The previous version of the robot face had additional realism features to make the robot more dynamic. The realism feature can be enabled by a user through a radio button on the control panel. This feature adjusts the eyebrow angle, location, pupil location and the eyeball rotation to simulate the subtle micromovements which occur in a realistic human face. Each of the feature values are assigned to a unique variable, each incrementing at different rates. A threshold is assigned for each of these variables and once the threshold is reached the variable is set back to the initial value and therefore we get a repetitive motion unique to each of the degrees of freedom. For example, let Rot1 be the angle of rotation applied to the eyebrows, and let Rot2 be the rotation applied to the eyeballs.

In previous tests, the addition of a realism feature seemed to benefit the user's recognition of certain expressions [26] however failed to enhance the recognition across all expressions. In fact stern, angry, and tired had a decreased recognition rate in comparison to the recognition rates without realism. This hypothesis will be tested again in the experiments section, with an emphasis on whether feature randomness is an aid or a deterrent to the recognition of various expressions.

Blinking

Blinking is an important aspect of human facial expressions. In the previous version of the robot face a blinking factor input by the user would adjust the probability rate for a blink occurring. A random number generator generates a random integer $r \in 0, 1, \ldots, \alpha_{blink} - 1$. α_{blink} is a chosen *blinking factor* initialised at 20, but can be altered by the user's entry into a text field. If r = 0 (an event with a $1/\alpha_{blink}$ probability), then both eyes simultaneously shut and open. As previously explored, a limitation is set on the maximum blink rate by fixing the random number generation rate to 200 ms. This will result in a maximum of five blinks per second at low values of α_{blink}.

3.4 Categorical and Affect Space Implementation

For this program, the same set of expressions were adopted from Breazeal's work in [25]. The basis facial expressions used in her work are happiness, sadness, anger, fear, surprise, tiredness, sternness and disgust. At the root of all these expressions is the neutral expression and the technique used to animate the categorical expressions is interpolating

repeatedly between the neutral expression to each of the individual basis expressions. This technique is generalisable for any number, n, of basis expressions if need be.

For this technique, we think of the neutral expression as being the link between all the different expressions. If we imagine 8 independent axes from neutral to the 8 basis emotions, the emotion $\vec{e}(t)\vec{e}(t)$ at any point in time is defined by the weighted combination of the points along the 8 axes (figure 13). The weighted vector

$\overrightarrow{w_i} = [amth\ amts\ amta\ amtf\ amtsd\ amtd\ amtt\ amtst]^T$

$\overrightarrow{w_i} = [amth\ amts\ amta\ amtf\ amtsd\ amtd\ amtt\ amtst]^T$. This defines the amount by which the basis expression $\overline{b}\,\overline{b_i}$ contributes to the final expression $\boldsymbol{\epsilon}\,\boldsymbol{\epsilon}$.

The set of 8 basis expressions $b_h,\ b_s \ldots\ldots b_n b_h,\ b_s \ldots\ldots b_n$ (defined in the general case as $\bar{e}\bar{e}$ in Section 3.1) are each a vector containing 13 values in the previous version of the project, and 27 values in this version of the project for increased flexibility of the lip (defined with ranges in Section 3.1). The overall expression at any point in time is therefore the sum of the weighted basis difference expressions added to the neutral expression (figure 13).

The application of these equations in Processing is briefly explained below. As explained in section 3.1 – the lips, brows and eyes are rendered using different openGL shapes and functions. Therefore, the values corresponding to each of the features are extracted into separate arrays from the basis expression vectors. The analysis below looks at the interpolation of the lips. The extracted arrays of only upper and bottom lip vertices are fed into the custom 'interpcurve()' function which outputs the interpolated values of the vertices from neutral to the corresponding basis expression.

A key feature in Processing is that the main code block called draw() is continuously repeated at a default rate of 60 frames per second. Therefore all the code within this block repeats 60 times in a

Figure 13. Visual Diagram of Categorical Interpolation

second. The following code for the lips is called within a function called CategoricalFunc(), which is placed in the draw block. Therefore, 60 interpolations are performed and rendered per second – resulting in smooth animations of the face.

Note *ht**x** stores the interpolated upper lip vertices for emotion **x** (eg. ht**s** = values of the upper lip vertices for surprised). Similarly hb**x** stores the interpolated values of the bottom lip vertices for

Face_Controller | Buttons_Class | Global_Variables | Interpolate | LoadPupils | Talkir

```
float[] hts = interpcurve(amts, nt[0], nt[1], st[0], st[1], nt[2], nt[3], st[2], st[3], nt[
float[] hbs = interpcurve(amts, nb[0], nb[1], sb[0], sb[1], nb[2], nb[3], sb[2], sb[3], nb[
float[] htf = interpcurve(amtf, nt[0], nt[1], ft[0], ft[1], nt[2], nt[3], ft[2], ft[3], nt[
float[] hbf = interpcurve(amtf, nb[0], nb[1], fb[0], fb[1], nb[2], nb[3], fb[2], fb[3], nb[
float[] htsd = interpcurve(amtsd, nt[0], nt[1], sdt[0], sdt[1], nt[2], nt[3], sdt[2], sdt[3
float[] hbsd = interpcurve(amtsd, nb[0], nb[1], sdb[0], sdb[1], nb[2], nb[3], sdb[2], sdb[3
float[] htd = interpcurve(amtd, nt[0], nt[1], dt[0], dt[1], nt[2], nt[3], dt[2], dt[3], nt[
float[] hbd = interpcurve(amtd, nb[0], nb[1], db[0], db[1], nb[2], nb[3], db[2], db[3], nb[
```

The 'weighting' terms are updated by the position of the sliders (scaled down to between 0 and 1)

```
void GetSliderValues(){
  //CATEGORICAL SLIDERS
  amth = slider1.getValueF();
  amta = slider2.getValueF();
  amts = slider3.getValueF();
  amtf = slider4.getValueF();
  amtsd = slider5.getValueF();
  amtd = slider6.getValueF();

  //AFFECT SPACE SLIDERS
  amtv = slider7.getValueF(); //VALENCE
  amtar = slider8.getValueF(); //AROUSAL
  amtsta = slider9.getValueF(); //STANCE
}
```

```
  //LIPS - INPUT 18 FLOATS FOR THE STARTING VERTICES, 18 FLOATS FOR THE ENDING VERTICES
float[] interpcurve(float amt,float x1, float y1, float x2, float y2, float x3, float y
  if (amt>=1) {
    amt=1; //reached target
  }
    x = lerp(x1,x2,amt);
    y = lerp(y1, y2, amt);
    a = lerp(x3, x4, amt);
    b = lerp(y3, y4, amt);
    c = lerp(x5, x6, amt);
    d = lerp(y5, y6, amt);
    e = lerp(x7, x8, amt);
    f = lerp(y7,y8,amt);
    g = lerp(x9, x10, amt);
    h = lerp(y9, y10, amt);
    i = lerp(x11, x12, amt);
    j = lerp(y11, y12, amt);
    k = lerp(x13, x14, amt);
    l = lerp(y13, y14, amt);
    m = lerp(x15, x16, amt);
    n = lerp(y15, y16, amt);
    o = lerp(x17, x18, amt);
    p = lerp(y17, y18, amt);
    fill(255,0,0);

    float[] vals = {x,y,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p};
    return vals;
    }
```

Figure 14. Processing code

We saw earlier in Figure 1, that Kismet relied on a different approach known as the 3 dimensional affect space theory. This states that a large number of facial expressions can be created using a 3-dimensional axis of valence, arousal and stance [14,15,18].

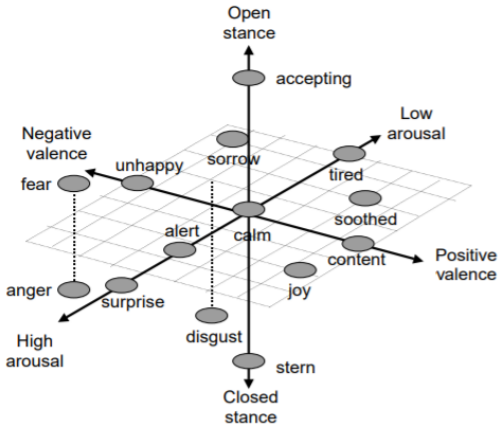


Figure 15.

The notable advantage of this method is that disgust, fear and surprise are supposedly distinguishable [14]. The theory again hinges on the link to the neutral expression, which is at the centre of the 3-dimensional axis. However this time we interpolate across a hypercubic region as shown in figure 16.

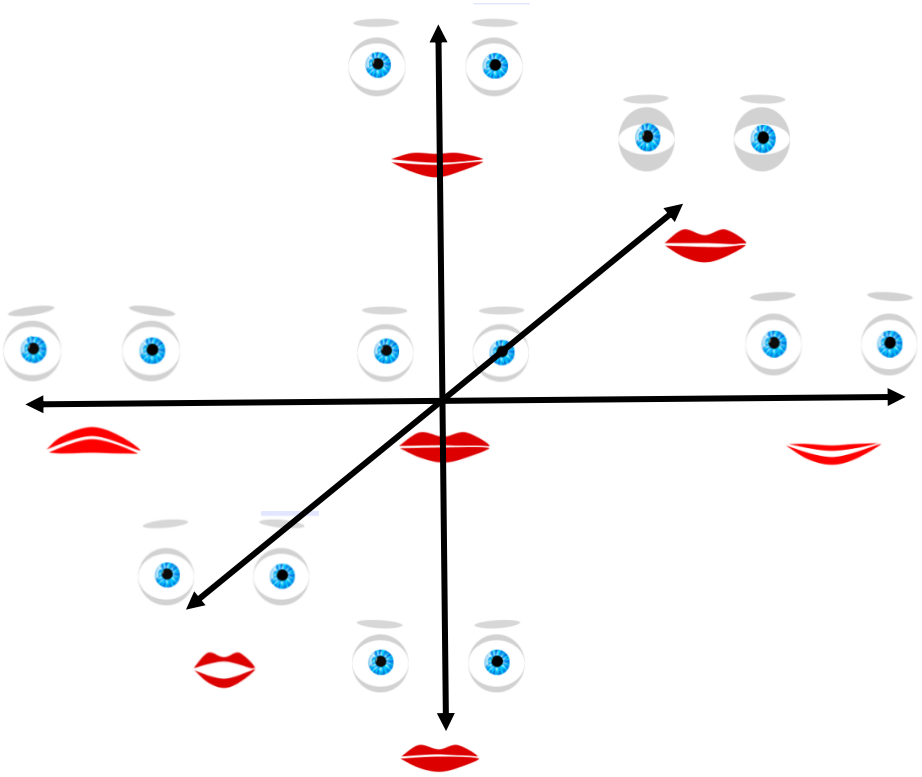


Figure 16. 3D Affect Space with Valence, Arousal and Stance base axes

Each expression can be expressed as a mathematical combination of Valence (V), Arousal (A) and Stance (S). According to Dr Breazeal, Kismet was able to display fear, anger and disgust at the following axes combinations (values approximated from figure 1 hypercube units).

$$Disgust = -0.8 * \boldsymbol{V} - 0.5 * \boldsymbol{A} - \boldsymbol{1} * \boldsymbol{S} \quad \textbf{(1)}$$

$$Fear = -1 * \boldsymbol{V} + 0.5 * \boldsymbol{A} \quad (\boldsymbol{2})$$

$$Anger = -1 * \boldsymbol{V} + 0.2 * \boldsymbol{A} + \boldsymbol{1} * \boldsymbol{S} \quad \textbf{(3)}$$

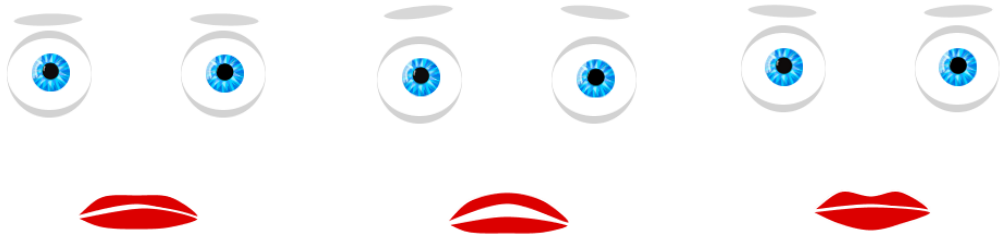


Figure 17. Expression outcomes of the equations above showing Disgust (1), Fear (2) and Anger (3)

For video processing, initially the aim was to build a puppeteer face which would imitate the user based on extraction of facial landmarks. The facial landmark extraction is relatively straight forward with the use of the LBF model in conjunction with the HAAR feature-based cascade classifier for facial detection in open CV, a real time computer vision library. FaceOSC is a program which extracts 132 raw facial landmark values (66 points) using Open CV 2 and sends this data to a port of our choice via OSC (Open Sound Control), a protocol for networking computers and other multimedia devices. This served as a convenient API to use for the facial data extraction into Processing. The 132 raw values were received via OSC as a float array in Processing where further extraction was carried out such that the eyes used only 8 out of the available 16 raw values to get the x and y values of the centre of the left and right eyes (ellipses), and the height and width of the eyes/ellipses, whilst the eyebrows used 20 out of 28 raw values as 5 vertices on each brow were determined to be enough. The remainder 88 values were used for the lips (36 values) and to create a rough outline of the nose (18 values) and jaw (34 values) to add more information on the facial display.

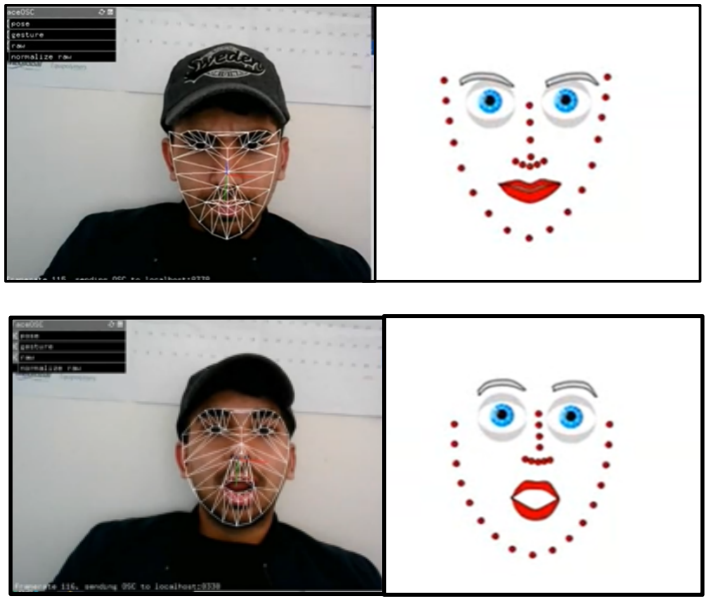


Figure 18. Puppeteer Testing

Emotion Recognition

Threshold classification

After the initial implementation of a puppeteer, the next aim was to develop an emotion recognition algorithm that could be used to detect a percentage of the categorical expressions and send the values across to the original programme which would adjust the slider values according to the incoming real time data. For a first attempt I recorded the changes in 8 key values, which were defined from the set of raw points, for the eight different expressions. I repeated this with six other users, carrying the test out twice each time. From observing this data, I manually extracted the key classifiers to distinguish between different expressions.

Robust ML Framework

While this made a good initial algorithm, it clearly wasn't robust enough to rely on. A machine learning model could potentially produce a more reliable form of classification. For my next attempt, I looked into ML models used in facial expression analysis from which it was clear that I needed to use a convolutional neural network (CNN). I used an open source GitHub repository to start with an initial model. I changed the structure of the model according to the GanVan paper and changed a few of the hyperparameters to minimize error on the given dataset. Below is a graph of the log cost and the accuracy as a function of the number of epochs for the improved version of the model.

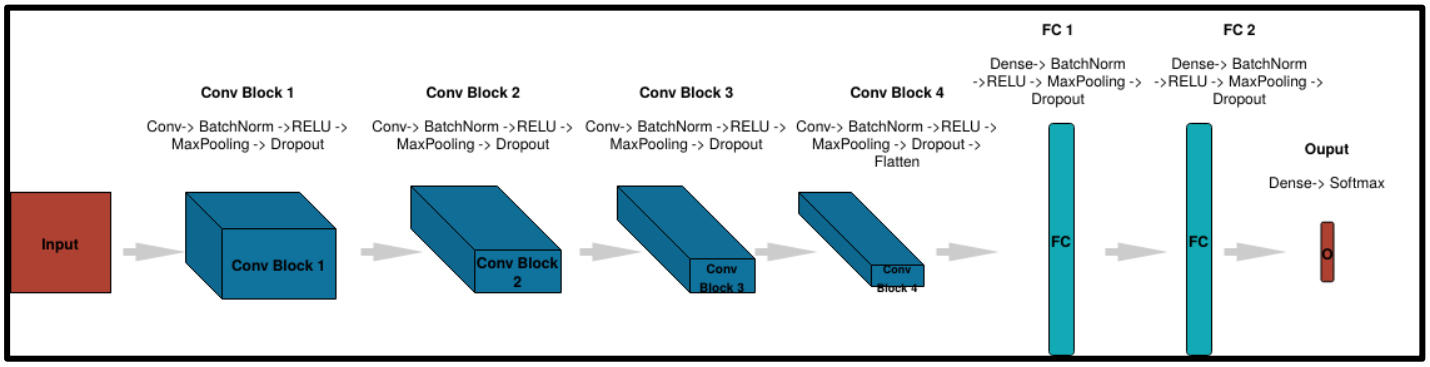


Figure 19 – CNN Model Architecture

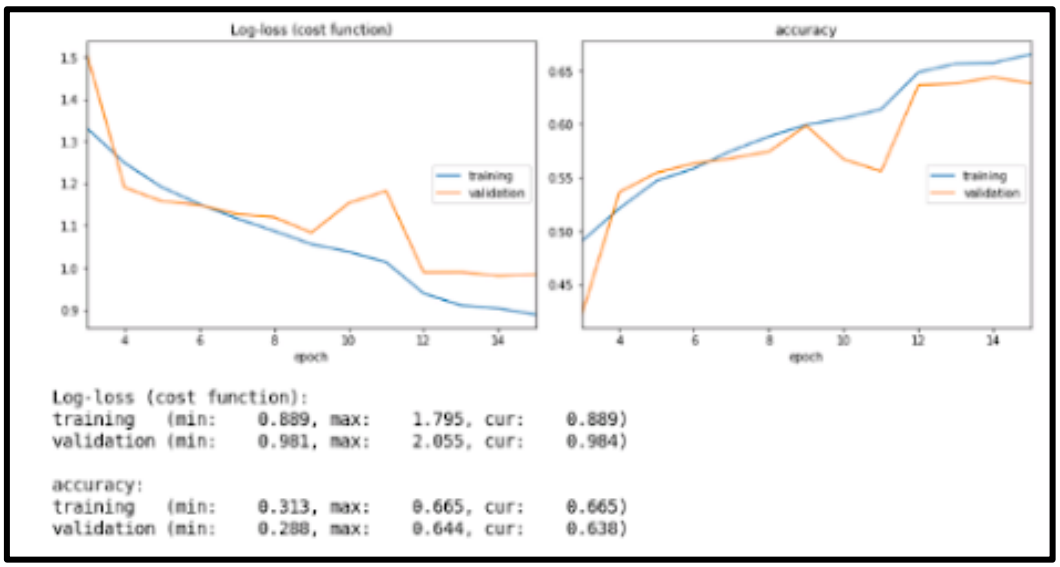


Figure 20 – 64% validation accuracy

References

1. Breazeal, Cynthia & Scassellati, Brian. (2003). How to Build Robots That Make Friends and Influence People. Proceedings of the IEEE International Conference on Intelligent Robots and Systems. 2. 10.1109/IROS.1999.812787.
2. Bishop, L., van Maris, A., Dogramadzi, S., and Zook, N. (2019). Social robots: The influence of human and robot characteristics on acceptance. Paladyn, Journal of Behavioral Robotics 10, 1, 346-358, Available From: De Gruyter<https://doi.org/10.1515/pjbr-2019-0028>
3. Bemelmans, R., Gelderblom, G. J., Jonker, P., & de Witte, L. (2012). Socially assistive robots in elderly care: a systematic review into effects and effectiveness. Journal of the American Medical Directors Association, 13(2), 114–120.e1. https://doi.org/10.1016/j.jamda.2010.10.002
4. Odetti, Luca & Anerdi, Giuseppe & Barbieri, Maria & Mazzei, Debora & Rizza, Elisa & Dario, Paolo & Rodriguez, Guido & Micera, Silvestro. (2007). Preliminary experiments on the acceptability of animaloid companion robots by older people with early dementia. Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference. 2007. 1816-9. 10.1109/IEMBS.2007.4352666.
5. Schiano, D., Ehrlich, S., Rahardja, K., and Sheridan, K., Face to interface: facial affect in (hu) man and machine. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp.193–200.
6. Breazeal, Cynthia. Designing Sociable Robots. The MIT Press, 2002, pg. 112
7. Ekman, P. and Oster, H., 1979. Facial Expressions of Emotion. Annual Review of Psychology, 30(1), pp.527-554.
8. Mehrabian, A. (1971). "Nonverbal communication," in Nebraska Symposium on Motivation 1971, ed. J. K. Cole (Lincoln, NE: University of Nebraska Press), 107–161.
9. Ekman, P., and Friesen, W. V. (1978). Facial Action Coding System. Palo Alto, CA: Consulting Psychologist.
10. D, Galati & S, Sini & S, Schmidt & C, Tinti. (2003). Spontaneous facial expression of emotion: Comparing congenitally blind and sighted children. Journal of visual impairment & blindness. 97. 418-428.
11. Scherer, K. R., & Ellgring, H. (2007). Are facial expressions of emotion produced by categorical affect programs or dynamically driven by appraisal? Emotion, 7(1), 113–130. https://doi.org/10.1037/1528-3542.7.1.113
12. Marcos S., Pinillos R., García-Bermejo J.G., Zalama E. (2016) Design of a Realistic Robotic Head Based on Action Coding System. In: Reis L., Moreira A., Lima P., Montano L., Muñoz-Martinez V. (eds) Robot 2015: Second Iberian Robotics Conference. Advances in Intelligent Systems and Computing, vol 418. Springer, Cham
13. Brian Scassellati, Cynthia Breazeal, Aaron Edsinger, and Paul Fitzpatrick (2001). Active vision for sociable robots. IEEE Transactions on systems, man, and cyberneticspart A: Systems and Humans 31, 5 (2001), 443–4
14. Breazeal, C.: Sociable Machines: Expressive Social Exchange Between Humans and Robots. Ph.D. thesis, MIT (2000)
15. Chamberland, J., Roy-Charland, A., Perron, M. and Dickinson, J., 2016. Distinction between fear and surprise: an interpretation-independent test of the perceptual-attentional limitation hypothesis. Social Neuroscience, 12(1251964), pp.1-18.
16. Ekman, P. (1972). Universals and cultural differences in facial expressions of emotions. In Cole, J. (Ed.), Nebraska Symposium on Motivation, 1971 (pp. 273). Lincoln: University of Nebraska Press.

17. Breazeal, C. (2003) Emotion and Sociable Humanoid Robots. International Journal of Human-Computer Studies. (1-2), 19-22. Available from: http://web.media.mit.edu/~cynthiab/Papers/Breazeal-ijhcs02-final.pdf.
18. Bazo, D., Vaidyanathan, R., Lentz, A. & Melhuish, C. (Oct 2010) Design and testing of a hybrid expressive face for a humanoid robot. IROS. , IEEE. pp.5317-5322Available from: https://ieeexplore.ieee.org/document/5651469 .
19. Kurfess, T., 2015. Robotics And Automation Handbook. Boca Raton: CRC Press, pp.21-12 to 21-15.
20. Sanyal, Soubhik, Timo Bolkart, Haiwen Feng and Michael J. Black. "Learning to Regress 3D Face Shape and Expression From an Image Without 3D Supervision." 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019): 7755-7764.Vougioukas, K., Petridis, S. & Pantic, M. Realistic Speech-Driven Facial Animation with GANs. Int J Comput Vis 128, 1398–1413 (2020). https://doi.org/10.1007/s11263-019-01251-8
21. Mori, Masahiro & MacDorman, Karl & Kageki, Norri. (2012). The Uncanny Valley [From the Field]. IEEE Robotics & Automation Magazine. 19. 98-100. 10.1109/MRA.2012.2192811.