**Jan Ondras**

# Replicating Human Facial Emotions and Head Movements on a Robot Avatar

Computer Science Tripos – Part II

Trinity College

May 18, 2017

# Proforma

| | |
|---|---|
| Name: | **Jan Ondras** |
| College: | **Trinity College** |
| Project Title: | **Replicating Human Facial Emotions and Head Movements on a Robot Avatar** |
| Examination: | **Computer Science Tripos – Part II** |
| Year: | **2017** |
| Word Count: | **11,499**[1] |
| Project Originator: | Dr Oya Celiktutan |
| Supervisors: | Dr Hatice Gunes & Dr Oya Celiktutan |

## Original Aims of the Project

The goal of this project was to develop a novel automatic teleoperation system that can recognise facial expressions of emotions (e.g. happiness) and estimate head movements from a video stream of a human teleoperator, and replicate these emotions and filtered head movements on the humanoid robot Nao in a real-time manner.

## Work Completed

The proposed system was successfully implemented and evaluated. I built on top of the Action Unit and Head Pose Detectors that detect facial action units (e.g. brow raiser) and measure the head pose of the teleoperator. I completed the following tasks:

- Use a neural network to infer emotions from detected action units.
- Smooth and bound the measured head motion to the robot's operating limits by applying constrained-state Kalman filtering.
- Display inferred emotions and filtered head movements on the robot in real-time.

I evaluated individual components and the whole system in user study.
As an extension I examined the replication latency.

[1] This word count was computed by `texcount -inc diss.tex`

## Special Difficulties

At the beginning I spent a considerable amount of time with setting up the legacy environment necessary to run the given detectors.

# Declaration

I, Jan Ondras of Trinity College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed .....

Date .....

# Contents

# Acknowledgements

I would like to thank to my project supervisors Dr Hatice Gunes and Dr Oya Celiktutan for all their advice, help, feedback, and support during this project. I would like to also express my gratitude to all the participants in my user study who helped me to evaluate my project.

# Chapter 1

# Introduction

In this chapter, *Motivation* introduces a robotic telepresence and its applications. Next, *Project overview* describes what the project is about and how it was completed, *Related work* lists the literature relevant to the project, and *Submitted paper* section gives details about my research paper based on this project. Lastly, the structure of the dissertation is laid out.

## 1.1 Motivation

Robotic telepresence offers a convenient substitute for face-to-face communication as it provides psychical embodiment at a remote place (from teleoperator's point of view) and allows the teleoperator to express non-verbal cues such as head movements, hand gestures, facial expressions along with audio cues to another person (user) via robot. An example of such a system is shown in Fig. 1.1.

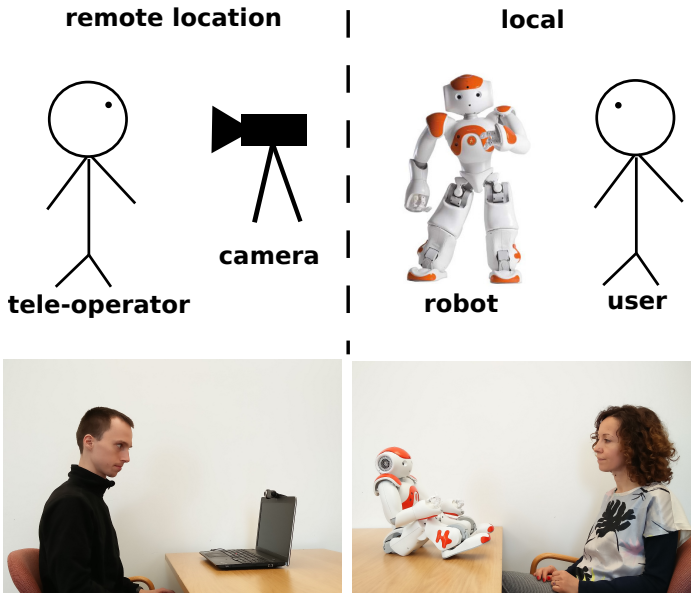



Figure 1.1: Telepresence example with Nao robot.

### 1.1.1 Possible applications

Facial and head cues carry significant information with regard to an individual's social signals including emotions, personality and intentions, which are key in enabling effective communication. For example, remotely located team members are less included in co-operative activities than co-located team members [1], and have fewer conversational turns and speaking time in group conversations [2]. These shortcomings can be mitigated by using robots for telepresence in various settings including remote education [3] or elderly care [4] which are emerging application domains.

## 1.2 Project overview

In this project, I have developed a novel teleoperation system that combines the replication of facial expressions of emotions (e.g. disgust, happiness, neutral) and head movements on the fly on the humanoid robot Nao (see Fig. 1.1). This allows a person at a remote location to be physically "present" via robot at another place to provide a more realistic communication experience.

In particular, given a visual recording of a human face the teleoperation system detects the facial action units (e.g. outer brow raiser) and uses them to infer a target emotion using a pre-trained neural network. Simultaneously, the head movements are measured and consequently smoothed and bounded to the robot's physical operating limits by applying a constrained-state Kalman filter. The recognised emotions and filtered head movements are then concurrently replicated on the robot in real-time. Since the Nao robot has a static face, the LEDs located around its eyes are used to reproduce the teleoperator's expressions of emotions.

I have successfully implemented the above-described system utilising the Action Unit Detector (AUD) and Head Pose Detector (HPD) that were provided by my supervisors. In particular, I trained a neural network for emotion classification, that used the action units detected by AUD to infer emotions; I implemented the constrained-state Kalman filter with a minimum jerk model to smooth and bound the head movements measured by HPD; and I displayed the inferred emotions and filtered head movements on the robot.

Finally, I evaluated the performance of the system and its components both quantitatively and qualitatively by conducting a number of computational experiments and a user study set up (approved by Ethics Committee, see Appendix B) in a real-life scenario. Specifically, I asked 28 participants to use the replication system by displaying facial expressions and head movements while being recorded by a web camera. Subsequently, 18 external observers viewed each of the recorded clips via an online survey and assessed the quality of the robot's replication of the participants' behaviours. The results show that my teleoperation system can successfully communicate emotions and head movements, resulting in a high inter-rater agreement among the external observers evaluated by intr-

aclass correlation coefficients $ICC_E = 0.91$ and $ICC_{HP} = 0.72$ for replication of emotions and head movements respectively.

Thus, I fulfilled the requirements of my core Project Proposal. Moreover, I implemented one proposed extension – measurement of the replication latency. The results show that the replication of emotions is more computationally intensive job than the head pose replication which was expected. Namely, $\tau_E = 49$ ms and $\tau_{HP} = 20$ ms for replication of emotions and head movements respectively.

## 1.3 Related work

Many studies have addressed the automatic recognition of facial expressions and the tracking of head pose in the context of human behaviour analysis and human-computer interaction [5]. However, (i) little work has been done to apply such computational methods to robot teleoperation (most of the teleoperation studies have been concerned with general object manipulation through a robot teleoperation interface [6]); and (ii) only a few studies have focused on the investigation of how teleoperators are perceived from the perspective of their interlocutors [7, 8, 9].

Expression of emotions via robots has been a popular research area. In one prominent work, Chevalier et al. [10] compared two robotic platforms with a virtual agent and a human in animating four emotions (anger, happiness, fear and sadness) through facial and bodily cues. These robotic platforms were Zeno and Nao, however only bodily cues were considered for the Nao robot. They found that facial cues play a more important role in conveying emotions, and sadness is the easiest emotion to recognise. Johnson et al. [11] demonstrated that the Nao robot can also satisfactorily imitate human emotions through facial cues. This was done by altering the colour, intensity, sharpness and orientation in the Nao robot's eyes. Song et al. [12] studied the effects of three interaction modalities (colour, sound, and vibration) on human perception of emotions. They found that the colour modality is the most important channel for communicating affect and using all modalities simultaneously improves the results. Therefore, in my project I used the colour modality for conveying emotions. In comparison to my work, the above-mentioned studies displayed the emotions in various ways and investigated their perception, but they did not deal with the real-time replication of emotions from a human.

Despite the importance of facial cues in interaction, previous teleoperation studies have mostly focused on portraying bodily cues on a robot avatar. For example, Bremner and Leonards [13] demonstrated the utility of iconic gestures using a real-time skeleton algorithm using a Kinect depth sensor. In the TERESA project [14], Shiarlis et al. developed a teleoperation system to allow elderly people to participate in social events remotely. The developed robot was able to semi-autonomously navigate among groups, maintain face-to-face contact during conversations, and display appropriate body poses.

Agarwal et al. [15] developed a real-time system for the imitation of human head movements. They recorded the teleoperator's head motion by a Microsoft Kinect sensor, and

processed these recordings in three steps: (i) low-pass pre-filtering; (ii) neural network-based head pose mapping (model between the Kinect and OptiTrack – ground truths from more accurate sensor); (iii) constrained-state Kalman filtering. This work constitutes a pioneering effort for a head pose replication system.

In my work, I followed a similar approach to the head pose replication of [15], but with the following contributions:

- I used a regular RGB web camera, and an automatic Head Pose Detector (HPD) to track the head movements.
- I extended the system with a novel emotion replication method based on automatic action unit detection performed in a real-time manner.
- I collected an in-house test database for system evaluation.

## 1.4 Submitted paper

This project resulted in a conference paper submitted to the 26$^{\text{th}}$ IEEE International Symposium on Robot and Human Interactive Communication.

> Jan Ondras, Oya Celiktutan, Evangelos Sariyanidi, and Hatice Gunes. Automatic replication of teleoperator head movements and facial expressions on a humanoid robot (under review). In *26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2017.

In the paper I present the teleopration system that I have developed in this project and furthermore, there is a detailed description of Action Unit Detector (utilised by this teleopration system) developed by Evangelos Sariyanidi[1].

## 1.5 Structure of the dissertation

The dissertation is organised as follows:

1. **Introduction**.
2. **Preparation** presents the starting point for the project, underlying theories, and software engineering techniques that were employed.
3. **Implementation** provides the whole system overview and implementation details of individual system components.
4. **Evaluation** describes the methods used to evaluate the individual modules and the entire system.
5. **Conclusion** gives a summary of completed work and results, takeaway messages, and plans for future work.

[1] PhD student at Queen Mary University of London.

# Chapter 2

# Preparation

## 2.1 Overview

This chapter describes the *Starting point* of the project in Section 2.2, then, the topics and areas I had to familiarise with:

- representation of emotions, action units and head pose (Sections 2.3 and 2.4),
- details of the robotic platform Nao (Section 2.5),
- detectors AUD and HPD (Sections 2.6 and 2.7 respectively),
- and necessary theories for emotion classification and head pose filtering (Sections 2.8 and 2.9 respectively).

The chapter concludes with the *Software engineering* techniques used including the high-level system design in Section 2.10.

## 2.2 Starting point

### 2.2.1 Previous knowledge and experience

Prior to the project, I had very little knowledge in Digital Signal Processing and Computer Vision and the courses on these were lectured in Michaelmas and Lent term respectively in this academic year. My knowledge of AI techniques was limited to what was taught in Part IB Artificial Intelligence and I had some basic experience with robotics from my high school. I also had to refresh and broaden my knowledge from Part IB C & C++ course.

Since the vast amount of the implementation was done in Python I had to familiarise with this language and commonly used libraries, as my previous experience was at the elementary level.

It turned out that this project involved lots of data processing and file manipulations for which I have exploited shell scripting. Thanks to the Part IB Unix Tools course I was not starting from scratch.

Lastly, I decided to use LaTeX for typesetting my dissertation. Again, I had very little experience with this tool before.

### 2.2.2 Building on existing code base

In this project I built on top of the following detectors provided by my project supervisors:

- Action Unit Detector (AUD): given a video stream, it detects which facial action units are active for each frame.
- Head Pose Detector (HPD): given a video stream, it estimates the head pose in terms of three rotation angles (pitch, yaw, roll) for each frame.

At the beginning I spent lots of time with setting up the proper environment in order to get the given detectors running. Namely, I had to return to older versions of Ubuntu Linux 14.04, GCC 4.8.4, and OpenCV 2.4.8.

### 2.2.3 Proposal refinement

The following sections describe the deviations from the original plan. The issues mentioned in sections *Detectors* and *Displaying on the robot* were also explained in the Progress report.

#### Detectors

When familiarising with the given detectors, I found out that the AUD and HPD are separate independent programs and not one Action Unit and Head Pose Detector (AUHPD) as described in the Proposal. Initially, I tried to fuse them into one detector, however, it turned out that each of them used a different version of statically linked IntraFace library [16] and neither of the versions was the subset of the other one. Therefore, it was not possible to simply link both of them into one detector and so I used AUD and HPD which brought an advantage of higher modularity of the system.

#### Databases

Due to time constraints I decided not to train the neural network for emotion classification on another dataset (SEMAINE) as originally intended, and I left it as an extension once the more important system-critical tasks are completed.

**Displaying on the robot**

It turned out that the proposed Displaying module based on Robot Operating System (ROS) is not needed since I found that Aldebaran Robotics provides Python and C++ SDKs for communication with Nao robot and so no additional software is necessary.

## 2.3 Emotions and action units (AUs)

Human emotions are known to be expressed by facial muscle movements. Ekman et al. proposed the Facial Action Coding System (FACS) that encodes these facial muscle movements in terms of the activation of a set of predefined action units (AUs) [17]. Fig. 2.1 illustrates such coding system for upper and lower face AUs.

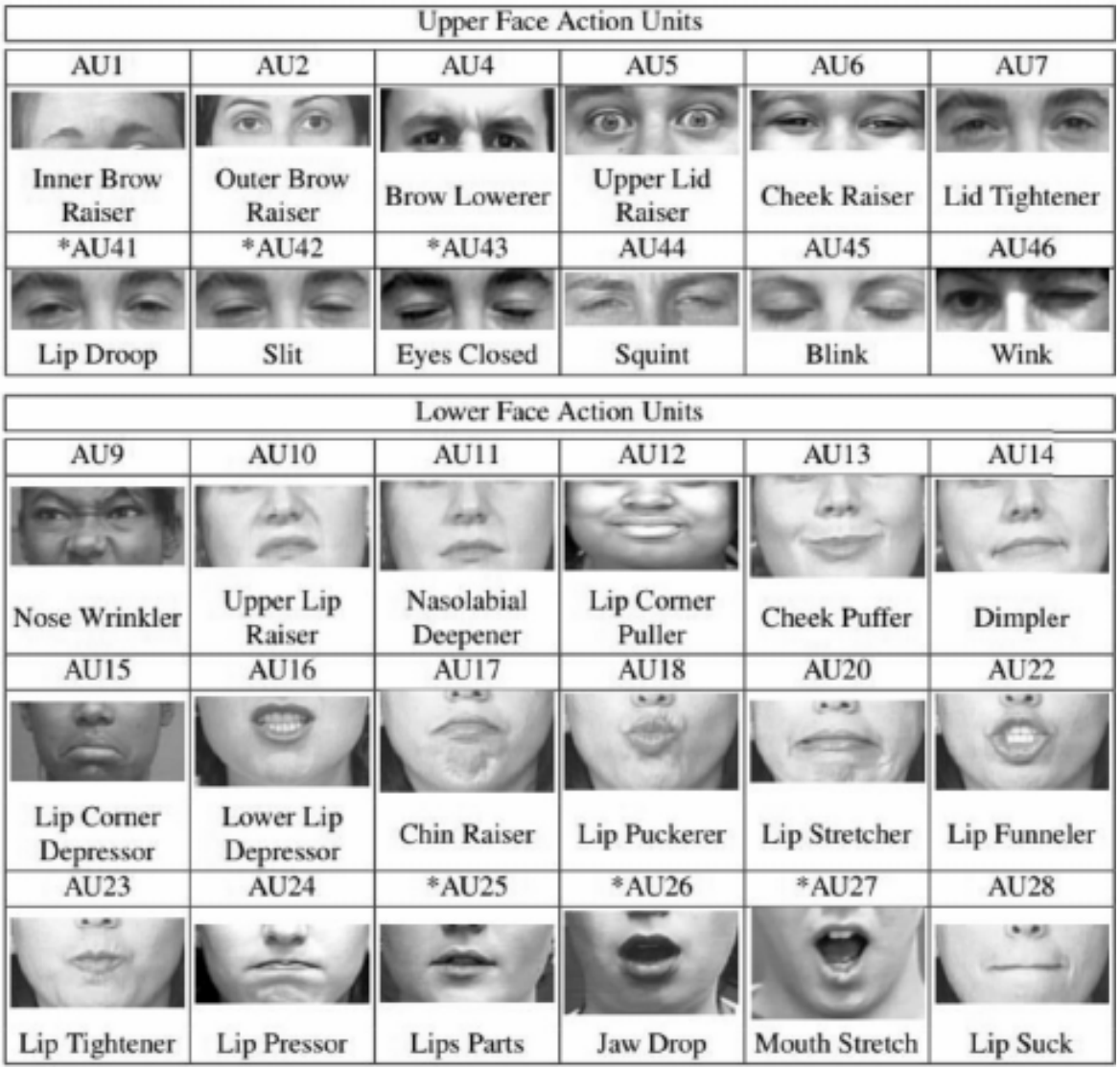



Figure 2.1: Facial action units of upper and lower face. Available from: `https://www.researchgate.net/publication/280298368_IntraFace` (accessed Apr 17, 2017).

This system forms the basis of a significant number of automatic emotion recognition methods. For example, in a simple rule-based method, happiness can be represented as a combination of AU6 (cheek raiser) and AU12 (lip corner puller) [18]. More such mappings for 6 basic emotions (anger, surprise, disgust, sadness, happiness, fear) proposed by Ekman and Friesen [19] are summarised in Tab. 2.1.

Table 2.1: Rule-based mapping between emotions and Action Units (AUs)

| Emotion | Active AUs |
|---|---|
| Happiness | 6, 12 |
| Sadness | 1, 4, 15 |
| Surprise | 1, 2, 5B, 26 |
| Fear | 1, 2, 4, 5, 7, 20, 26 |
| Anger | 4, 5, 7, 23 |
| Disgust | 9, 15, 16 |

## 2.4 Head pose

The head pose can be described by three angles: pitch $\theta_P$, yaw $\theta_Y$, and roll $\theta_R$ that correspond to rotations around x, y, and z axes respectively, as illustrated by Fig. 2.2.

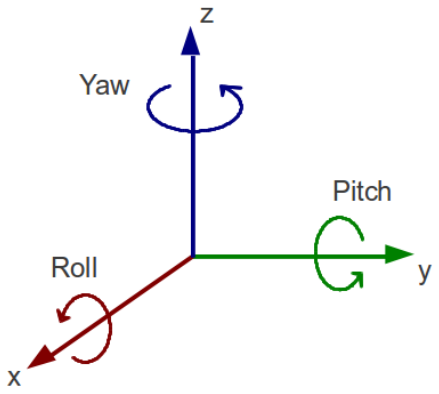



Figure 2.2: Head rotations. Available from: `http://doc.aldebaran.com/2-1/family/nao_h25/joints_h25.html#h25-joints` (accessed Apr 17, 2017).

## 2.5 Robotic platform – Nao

For the robotic platform, I used the humanoid robot Nao developed by Aldebaran Robotics [20] with the technical details of NaoQi version 2.1, head version 4.0 and body version 25. As shown in Fig. 2.3 it is a child-like looking robot with dimensions 57.3 cm × 31.1 cm × 27.5 cm (height, depth, width). It is commercially available and very

widely used in human-robot interaction studies. In all my experiments I sent the required commands to the robot through an Ethernet cable directly from my laptop.

Figure 2.3: Nao robot. Available from: `http://viseindia.in/assets/img/nao.png` (accessed May 4, 2017).

### 2.5.1 Displaying emotions

The Nao robot has a static face, and cannot display facial muscle movements. However, in [11], Johnson et al. demonstrated that the Nao robot can use LED colours and patterns to imitate emotions. Inspired by this study, I mapped each emotion onto a different colour code that was displayed on Nao using its eyes' LEDs. All 16 LEDs in its eyes were lit to the same colour when an emotion was synthesised. The Nao robot's face is shown in Fig. 2.4.

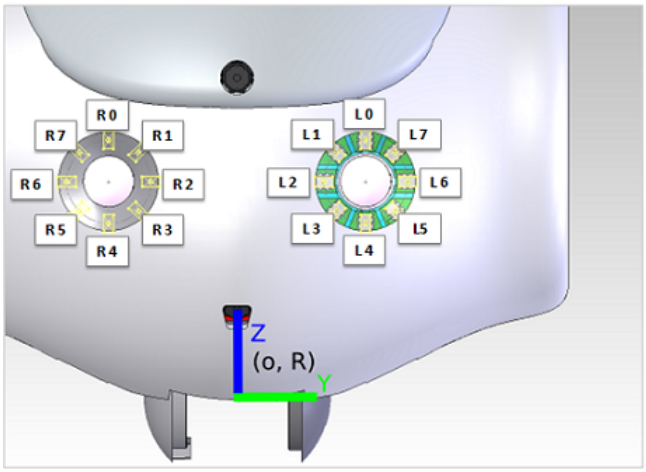



Figure 2.4: Nao robot's face showing the LEDs used to display emotions. Available from Aldebaran Robotics website: `http://doc.aldebaran.com/2-1/family/robots/leds_robot.html` (accessed Apr 17, 2017).

### 2.5.2 Head movements

The Nao robot can rotate its head only around two axis – yaw $\theta_Y$ and pitch $\theta_P$ rotations, as shown in Fig. 2.5. Therefore, we will further consider only these 2 rotations.

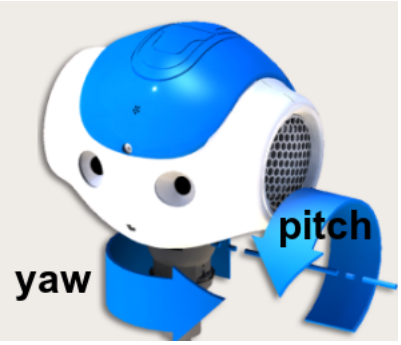



Figure 2.5: Nao robot's head showing the possible head rotations: yaw $\theta_Y$ and pitch $\theta_P$. Roll rotation around z-axis is not possible.

Naturally, the rotation angles $\theta_Y$ and $\theta_P$ have upper and lower bounds. Furthermore, due to potential shell collision at the head level, the pitch motion range is limited according to the yaw value as shown in Tab. 2.2.

Table 2.2: Nao robot's pitch angle range $[\theta_{Pmin}, \theta_{Pmax}]$ limited according to yaw angle $\theta_Y$. From Aldebaran Robotics website: `http://doc.aldebaran.com/2-1/family/nao_h25/joints_h25.html#h25-joints` (accessed Apr 17, 2017).

| Yaw: $\theta_Y$ [rad] | Pitch min: $\theta_{Pmin}$ [rad] | Pitch max: $\theta_{Pmax}$ [rad] |
|---|---|---|
| -2.086017 | -0.449073 | 0.330041 |
| -1.526988 | -0.330041 | 0.200015 |
| -1.089958 | -0.430049 | 0.300022 |
| -0.903033 | -0.479965 | 0.330041 |
| -0.756077 | -0.548033 | 0.370010 |
| -0.486074 | -0.671951 | 0.422021 |
| 0.000000 | -0.671951 | 0.515047 |
| 0.486074 | -0.671951 | 0.422021 |
| 0.756077 | -0.548033 | 0.370010 |
| 0.903033 | -0.479965 | 0.330041 |
| 1.089958 | -0.430049 | 0.300022 |
| 1.526988 | -0.330041 | 0.200015 |
| 2.086017 | -0.449073 | 0.330041 |

Also, the angular velocities $\omega_Y$ and $\omega_P$ corresponding to yaw and pitch rotations are limited to ranges $[-8.26797, 8.26797]$ rad/sec and $[-7.19407, 7.19407]$ rad/sec respectively.

## 2.6 Action Unit Detector (AUD)

This module was provided by my supervisors and it has the following specification:
**Main task:**
Given a video stream with a human face, detect a list of selected AUs (specified by ordered set $S_{AU}$) frame-by-frame in real-time.
**Input:**
a video stream either from a video file (offline mode) or from a web camera (online mode), and an optional calibration video – if not provided then the AUD calibrates according to the first frame of the input video stream.
**Output (per each processed frame):**
a binary vector $\boldsymbol{a}$ indicating which AUs (from $S_{AU}$) are active in that frame. Where $a_i \in \{0, 1\}$ and $\boldsymbol{a}$ has the length $N_{AU}$ – the number of AUs that AUD is set to detect. For example, the output vector

$$\boldsymbol{a} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$$

with $N_{AU} = 7$ and $S_{AU} = \{AU1, AU2, AU4, AU6, AU12, AU25, AU26\}$ indicates that only AU2 and AU26 are active in that frame.

In general, the AUD can detect a maximum of 12 AUs, specified by the set

$$S_{maxAU} = \{AU1, AU2, AU4, AU6, AU12, AU15, AU20, AU24, AU25, AU26, AU27, AU45\}$$

and so we can write $S_{AU} \subseteq S_{maxAU}$.

The AUD was implemented in C++ using OpenCV library and IntraFace [16] which is a free research software for facial image analysis. Fig. 2.6 shows a screenshot of visual output when the detector is running in online mode.

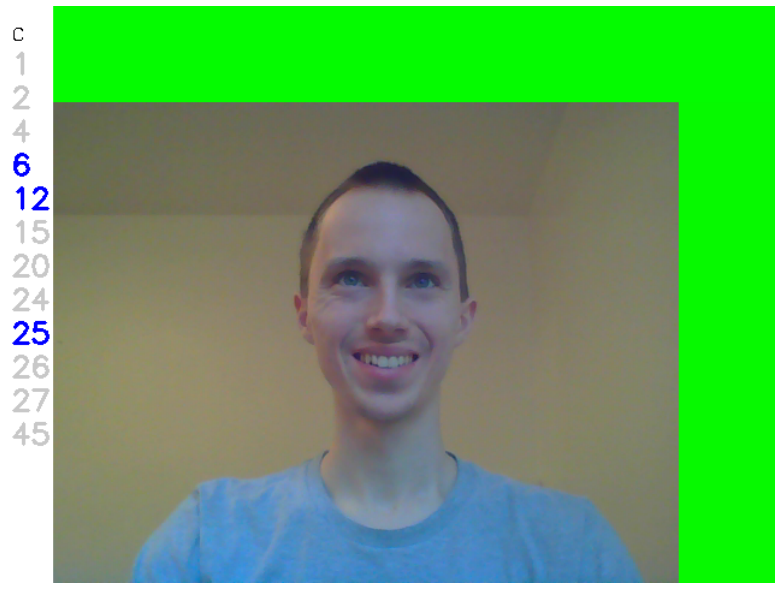



Figure 2.6: The visual output from Action Unit Detector (AUD). Detected action units: cheek raiser (AU6), lip corner puller (AU12), and lips parted (AU25).

### 2.6.1 Details

This AUD was developed by Evangelos Sariyanidi[1] and the detailed description is given in our paper [21]. It uses four types of features, namely, shape, appearance, differential-appearance and differential-shape, following the insights highlighted by Sariyanidi et al. in [5] and [22]:

- Combining shape and appearance features yields a better performance because these feature types carry complementary information.
- Using differential features (i.e. features that describe information with respect to the neutral face) gives higher emphasis to the facial action by reducing person-specific appearance cues.

For each AU four binary SVM classifiers were trained (when the AUD was developed, not in my project), each in conjunction with one of the abovementioned feature types.

The final AU detection decision is given by fusing the outputs of the four individual classifiers. Specifically, the *consensus fusion* approach is adopted, where an AU is detected based on the condition that all four classifiers are in full agreement.

## 2.7 Head Pose Detector (HPD)

This module was also provided by my supervisors and it has the following specification:
**Main task:**
Given a video stream with a human head, estimate the head pose (namely, pitch, yaw, and roll angles) frame-by-frame in real-time.
**Input:**
a video stream either from a video file (offline mode) or from a web camera (online mode).
**Output (per each processed frame):**
pitch, yaw, and roll angles $(\theta_P, \theta_Y, \theta_R)$ of the head pose.

For example, the output triple

$$(\theta_P, \theta_Y, \theta_R) = (0.36 \text{ rad}, -0.51 \text{ rad}, 0.50 \text{ rad})$$

corresponds to the frame shown in Fig. 2.7.

[1] PhD student at Queen Mary University of London.

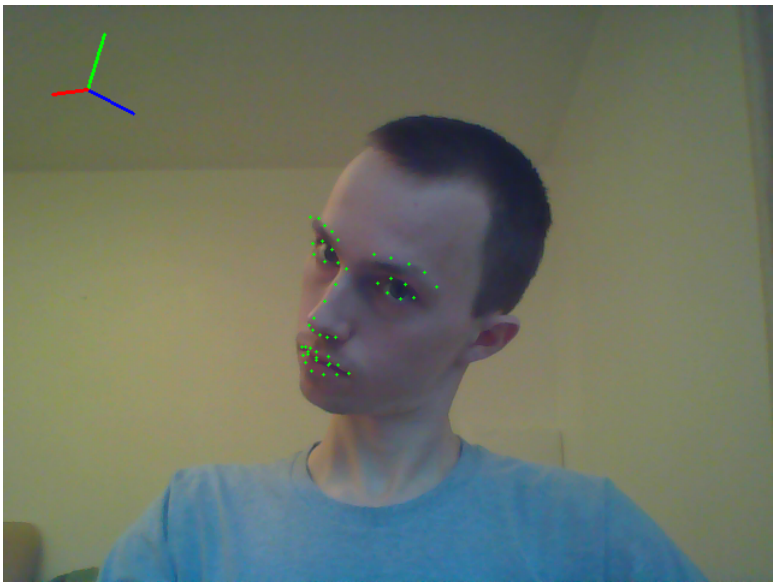


Figure 2.7: The visual output from Head Pose Detector (HPD). Pitch, yaw, and roll angles: 0.36 rad, −0.51 rad, and 0.50 rad respectively.

The HPD is based on the method developed in [16] and it was implemented in C++ using OpenCV library and IntraFace [16] – free research software for facial image analysis.

## 2.8 Multiclass classification

### 2.8.1 Motivation

Based on the facial action units (detected by AUD) the system has to assign an appropriate emotion. Since there are more than two classes of emotions, multiclass classification has to be considered. For this task I used a neural network.

The training and cross-validation of a multilayer perceptron network were well covered by Part IB Artificial Intelligence and Part II Machine Learning and Bayesian Inference courses, however, without an extension to multiclass classification problems. Therefore, in this section I briefly explain how to make predictions using the trained neural network when there are more than two classes.

### 2.8.2 Theoretical details

Let us assume a set of classes $\mathbb{K}$ (emotions) and a sample input row vector $\boldsymbol{x}$ (based on detected AUs) for which we want to make a prediction. If there are more than two classes, instead of the sigmoid logistic function the softmax function $s : \mathbb{R}^{|\mathbb{K}|} \to \mathbb{R}^{|\mathbb{K}|}$ is applied as an activation function for output layer nodes in order to squash the vector $\boldsymbol{z}$ of arbitrary real values to the vector $s(\boldsymbol{z})$ of real values in the range $[0, 1]$ that sum to 1.

$$s(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_{k \in \mathbb{K}} \exp(z_k)}, \qquad i \in \mathbb{K}$$

where $z_i$ and $s(\boldsymbol{z})_i$ are the real values produced by an output layer node corresponding to class $i$ before and after applying the activation function respectively. Furthermore, $s(\boldsymbol{z})$ represents the probability distribution over $\mathbb{K}$ and so we can write $s(\boldsymbol{z})_i = P(\text{class} = i|\boldsymbol{x})$ to express the probability that the input $\boldsymbol{x}$ belongs to class $i$.

Finally, we choose the highest probability class $\tilde{c} \in \mathbb{K}$ as

$$\tilde{c} = \arg\max_{i \in \mathbb{K}} s(\boldsymbol{z})_i$$

For example, for a neural network with one hidden layer with $m$ nodes and input layer of size $n$ we can write

$$\tilde{c} = \arg\max_{i \in \mathbb{K}} s(W_2 f(W_1 \boldsymbol{x} + \boldsymbol{b_1}) + \boldsymbol{b_2})_i$$

where $W_1 \in \mathbb{R}^{m \times n}, W_2 \in \mathbb{R}^{\mathbb{K} \times m}$ are the matrices of trained weights of the input layer and hidden layer respectively; $\boldsymbol{b_1} \in \mathbb{R}^m, \boldsymbol{b_2} \in \mathbb{R}^{\mathbb{K}}$ are the row vectors of trained biases of the input layer and hidden layer respectively; and the function $f : \mathbb{R}^m \to \mathbb{R}^m$ applies the hidden layer activation function to each element of its input argument.

## 2.9 Constrained-state Kalman filter with a minimum jerk model

### 2.9.1 Motivation

The raw head pose measurements (e.g. from HPD, Microsoft Kinect or other sensing devices) typically contain significant noise, which would result in jerky robot head movements when directly replicated on the robot. Furthermore, the direct use of raw head pose measurements could generate commands for the robot that would violate the robot's movement constraints.

Therefore, the constrained-state Kalman filter with a minimum jerk model is applied to raw head pose measurements in order to achieve a smooth head pose trajectory, in terms of angular position, velocity and acceleration, that can be displayed on the robot by taking into account the robot's physical operating limits.

In this section, we will address the head pose filtering following a similar approach to that taken by Agarwal et al. [15]. Please see Section 1.3 for more details about their work and a comparison with my work.

### 2.9.2 Theoretical details

As described in Section 2.4, the head pose is specified by three angles: pitch, yaw, and roll. Without loss of generality, we will further consider only one general rotation angle $\theta$.

The state vector at a time instant $t$ can be defined as

$$x_t = \begin{bmatrix} \theta_t & \omega_t & \alpha_t \end{bmatrix}^T$$

where $\theta_t$ is the angle, $\omega_t$ is the angular velocity, and $\alpha_t$ is the angular acceleration. It has been shown that the voluntary human movements obey a minimised jerk trajectory, which is also the smoothest trajectory [23], [24]. Therefore, we assume the minimum jerk model as a transition model. Let us consider the system given by

$$\begin{aligned} x_{t+1} &= F_t x_t + w_t \\ y_t &= H x_t + v_t \end{aligned}$$

where

$$F_t = \begin{bmatrix} 1 & \Delta T_t & \frac{\Delta T_t^2}{2} \\ 0 & 1 & \Delta T_t \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

is the transition model, $\Delta T_t$ is the time step, $y_t$ is the measurement, $H$ is the measurement model (in our case $H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$, since we measure only angle $\theta_t$), $w_t$ is the Gaussian process noise, $w_t \sim \mathcal{N}(0, Q)$, and $v_t$ is the Gaussian measurement noise, $v_t \sim \mathcal{N}(0, R)$ with covariance matrices $Q$ and $R$ respectively. In the case of the minimum jerk model it is often assumed that the process noise is only at acceleration level which yields

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & q \end{bmatrix}$$

where $q \in \mathbb{R}_{\geq 0}$. Also, we can write

$$R = \begin{bmatrix} r \end{bmatrix}, \qquad r \in \mathbb{R}_{\geq 0}$$

because we measure only angle $\theta_t$. Since the covariance matrices $Q$ and $R$ are uniquely specified by scalars $q$ and $r$ we will further refer to $q$ and $r$ as the process and measurement noises respectively. In practise, it is not easy to estimate them and they often require empirical methods.

The Kalman Filter is described using the following:

1. Time update (prediction)

$$\begin{aligned} P_t^- &= F_t P_{t-1}^+ F_t^T + Q \\ \hat{x}_t^- &= F_t \hat{x}_{t-1}^+ \end{aligned}$$

2. Measurement update (correction)

$$\begin{aligned} K_t &= P_t^- H^T \left(H P_t^- H^T + R\right)^{-1} \\ P_t^+ &= \left(I - K_t H\right) P_t^- \\ \hat{x}_t^+ &= \hat{x}_t^- + K_t \left(y_t - H\hat{x}_t^-\right) \end{aligned} \tag{2.2}$$

where $\hat{x}_t^-$ is the a priori estimate of state $x_t$ given measurements up to time $t-1$ inclusive, $\hat{x}_t^+$ is the a posteriori estimate of $x_t$ given measurements up to time $t$ inclusive, $P_t^-$ is the covariance matrix of the a priori estimation error $(x_t - \hat{x}_t^-)$, $P_t^+$ is the covariance matrix of the a posteriori estimation error $(x_t - \hat{x}_t^+)$, and $K_t$ is the Kalman gain.

The output $\hat{x}_t^+$ of the original filter given by equation (2.2) might be truncated due to the robot's movement constraints, resulting in discontinuities in the robot's head movements. To overcome this problem, we use constrained-state Kalman filter, and project the unconstrained estimate $\hat{x}_t^+$ onto the constraint surface [25]. The constrained state estimate $\tilde{x}_t^+$ can then be formulated as an optimization problem,

$$\tilde{x}_t^+ = \arg\min_x \left(x - \hat{x}_t^+\right)^T U \left(x - \hat{x}_t^+\right) \tag{2.3}$$

subject to inequality constraints $x \geq \begin{bmatrix}\theta_{min} & \omega_{min} & \alpha_{min}\end{bmatrix}^T$ and $x \leq \begin{bmatrix}\theta_{max} & \omega_{max} & \alpha_{max}\end{bmatrix}^T$. Here, $\theta_{min}$, $\omega_{min}$, $\alpha_{min}$, $\theta_{max}$, $\omega_{max}$, $\alpha_{max}$ define the known fixed bounds on state. Setting $U = \left(P_t^+\right)^{-1}$ guarantees the maximum probability state estimates (i.e. minimum variance filter) subject to the state constraints.

## 2.10 Software engineering

The system of a large scale requires thorough planning and design. This was done before implementation phase exploiting the software development skills and techniques that I have learnt in Part IB Software Engineering course.

### 2.10.1 Requirements analysis

I analysed the project requirements at the beginning of the implementation, mainly, by discussion with my supervisors. This helped me to clarify the goals of the project, lay out the structure of the system, identify dependencies between components, and to determine the ordering in which they were implemented. Moreover, the requirements analysis allowed me to assess whether the project satisfied the required functionality.

#### Main system requirement

Given a video stream of a human teleoperator replicate their emotions and head movements on the Nao robot.

### High-level system design

In order to achieve the main requirement I split the problem into separate tasks that led to a modular system architecture shown in Fig. 2.8.

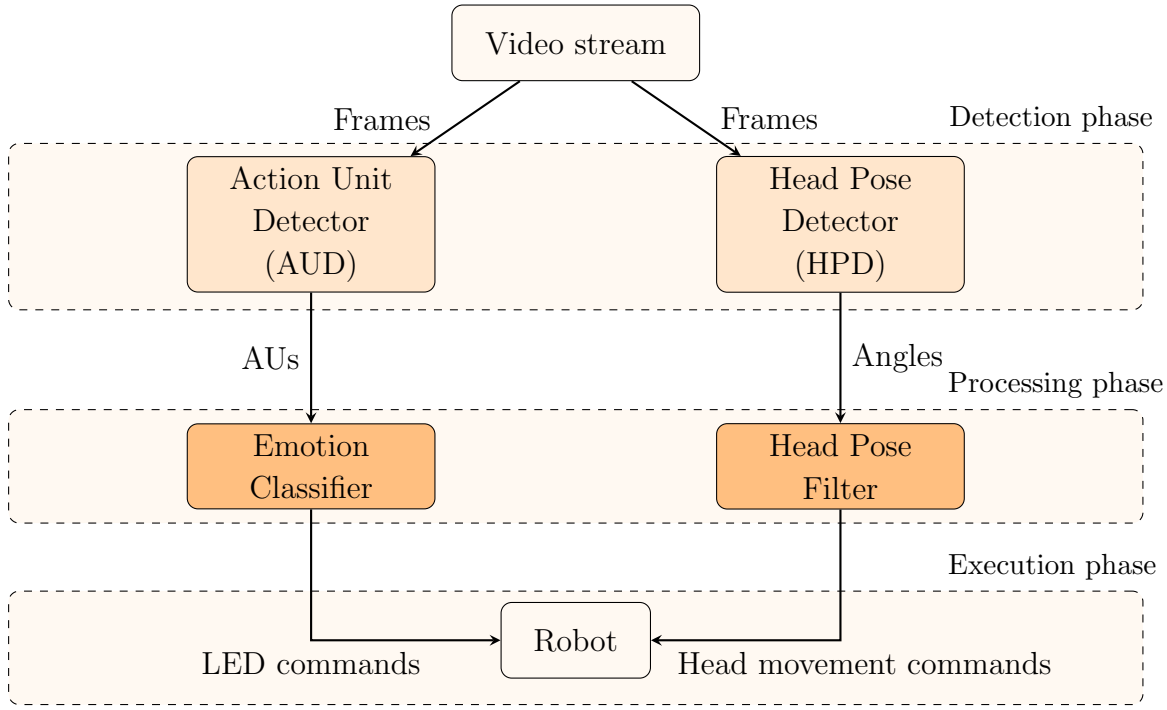



Figure 2.8: High-level system design of the proposed teleoperation system. *Bright orange:* existing modules to be modified. *Dark orange:* new modules to be implemented.

The proposed replication system consists of five main modules: AUD, HPD, Emotion Classifier, Head Pose Filter, and Robot. These components can be grouped horizontally into the following three phases according to the stage of the system pipeline.

1. **Detection phase:** AUD and HPD simultaneously process incoming video stream frame by frame.
2. **Processing phase:** the Emotion Classifier infers the teleoperator's emotion (e.g. happiness) from the detected AUs (e.g. brow lowerer, lip corner puller), and simultaneously the Head Pose Filter mitigates the effect of noisy head pose estimations.
3. **Execution phase:** the robot exhibits the target behaviour according to LED and head movement commands from processing phase components.

Vertically, according to the overall task, the components can be divided into:

1. **Emotion replication:** AUD and Emotion Classifier.
2. **Head pose replication:** HPD and Head Pose Filter.

To summarise the dependencies between modules: the Emotion Classifier requires the implementation of AUD and Head Pose Filter requires the implementation of HPD. How-

ever, the emotion replication components are completely independent of the head pose replication components. So there are two independent flows of data (robot commands) going to the robot.

**Requirements on system components**

1. **Action Unit Detector (AUD)**
   Given a frame from a video stream of the teleoperator, the AUD must detect which AUs were active.
2. **Head Pose Detector (HPD)**
   Given a frame from a video stream of the teleoperator, the HPD needs to measure the head pose specified by angles of rotation.
3. **Emotion Classifier**
   Given the set of active AUs from AUD, the Emotion Classifier must infer an emotion using the neural network trained beforehand. Then the recognised emotion must be encoded as a colour, according to predefined rules, and sent to the robot as LED commands.
4. **Head Pose Filter**
   Given the raw head pose measurements from HPD, the Head Pose Filter needs to filter them using constrained-state Kalman filter with a minimum jerk model. Based on the filtered head pose state estimate, the target position and velocity (to reach this position) need to be sent to the robot as head movement commands.
5. **Robot**
   The robot needs to light up all the LEDs around its eyes to the colour specified in LED commands received from Emotion Classifier. Simultaneously, the robot must move its head according to the target position and velocity specified in head movement commands from Head Pose Filter.

The requirements on the functionality of AUD and HPD were already met at the beginning, see Sections 2.6 and 2.7 respectively, however, they had to be modified in order to interact with Emotion Classifier and Head Pose Filter components in a desired way. Emotion Classifier and Head Pose Filter were completely new components built from scratch using the theories presented in Sections 2.8 and 2.9 respectively. The requirements on the robot were already satisfied at the start of the project.

### 2.10.2 Success criteria

The project will be considered successful if the following criteria are met:

1. Given a visual recording of a human face the Emotion Classifier will recognise correct emotions,
2. the Head Pose Filter will smooth and bound the measured head movements,

3. the Nao robot will display the recognised emotions and head movements concurrently in real-time.

### 2.10.3 Testing and evaluation

In order to assess how well the success criteria were met, I evaluated my project in the following way.

I separately and manually tested the system modules (and modules' units). The whole system was further tested on real Nao robot as well as on simulated robot using Choregraphe application which allowed me to save some time with setting up the real robot each time.

Moreover, three main evaluations were carried out:

- Emotion Classifier (quantitative) evaluation – testing the trained classifier on the in-house database collected in the controlled experiment involving human participants.
- Head Pose Filter (quantitative) evaluation – assessing absolute errors between the head pose estimates produced by the filter and ground truth data from the labelled head pose video database.
- Whole system (qualitative) evaluation – the quality of replication of both emotions and head movements assessed using a web-based survey.

Additionally, as an extension, the replication latency of the system was examined.

For more details see Chapter 4.

### 2.10.4 Challenges for the system

The following list identifies the main challenging conditions for the system.

- Insufficiently or unevenly illuminated teleoperator.
- Occlusion of teleoperator's facial features, e.g. by sun glasses or significant facial hair.
- No reference neutral expression provided to AUD (neither at the beginning of video stream nor with a calibration video).

However, the whole field of emotion recognition faces these challenges.

### 2.10.5 Risk analysis

This section identifies the possible risks associated with the project and proposes some potential solutions to them.

- Detectors
  The provided detectors (AUD, HPD) might not work as expected and it might be cumbersome to use them, e.g. due to insufficient or missing documentation. Such problems would need to be discussed with my project supervisors.
- Laptop failure
  If my laptop suddenly fails I will continue my work from a backup on a MCS machine until it is repaired.
- Robot failure
  In case the robot fails I will continue my work on a similar robotic platform – Pepper robot that is also available at Computer Laboratory.
- Teleoperation system malfunction
  If the developed system misbehaves it will not cause any damage to itself, other objects, or to its operators or other humans, since I use only the robot's LEDs and head movement functions. The other robot-associated risks are covered by its manufacturer. Thus, from my perspective the only consequence of the system malfunction would be the incorrect replication of emotions and head movements.

### 2.10.6 Development methodology

The chosen **modular system architecture** has the following advantages: increased flexibility during the development; easier debugging, testing and extending; and lower risk of not completing the main system requirement.

I used the **iterative software development methodology** within each module and for the system as a whole. This approach fit well with the regular meetings with my project supervisors. Furthermore, it allowed for early debugging, testing, and refinement.

### 2.10.7 Planning and backup

I have made contingency plans to protect myself against hardware and/or software failure. An analysis with potential solutions is discussed in *Risk analysis* in Section 2.10.5.

Moreover, I periodically backed up the project to an external 500GB HDD and USB flash drive. This applied to all project-related documents including the dissertation. I also backed up some crucial parts to my private storage on Google Drive. Since I built upon detectors developed in research I was not allowed to publish everything on GitHub as proposed. Instead, I made more backups locally.

### 2.10.8 Development environment, tools and resources

For my project I needed to use a significant range of software systems to deal with the interacting components that fit together to make the complete system. Not only did some

of these impose severe constraints on my work (e.g. the need to run on older releases of Linux) but I had to create interfaces and other ways to transmit data between them. The following list may give some idea of the scale of complication that I faced.

### Hardware resources

- My laptop: quad-core 2.2GHz Intel i7, 8GB RAM, 1TB HDD, Ubuntu Linux 14.04.
- Nao robot in the Computer Laboratory. The access to it was provided by my project supervisors. For more details about the robot see Section 2.5.

### Programming language

- **C++**[2] – for changes made to AUD and HPD.
- **Python**[3] – for implementation of Emotion Classifier and Head Pose Filter modules.

I chose Python for emotion classification and head pose filtering due to its great support and libraries for data processing, scientific computation, and machine learning tasks. Furthermore, using Python does not require cross-compiling on the robot as compared to C++ which allowed me to do faster testing and development. Another important factor in choosing the programming language for Emotion Classifier and Head Pose Filter was the communication interface with C++ (since AUD and HPD were written in C++). Fortunately, the C++/Python API[4] provides an easy way to call Python functions from C++.

### Tools

The following tools were employed throughout the project development:

- **IPython Notebook**[5] – for development in Python.
- **Choregraphe**[6] – for simulating the virtual robot on my computer.
- **MATLAB**[7] – for evaluating the results gathered in user study (web survey).
- **Shell scripting** – for data processing and file manipulations.
- **LaTeX** – for typesetting the dissertation.

[2] Older GCC 4.8.4 was needed.
[3] Version 2.7.6.
[4] `https://docs.python.org/2/c-api/index.html#c-api-index`
[5] `https://ipython.org/`
[6] `http://doc.aldebaran.com/2-1/software/choregraphe/index.html`
[7] `https://www.mathworks.com/products/matlab.html`

### Libraries

The summary of main libraries used during the project:

- **C++**
  - **OpenCV**[8] – for computer vision tasks.
  - **IntraFace [16]** – for facial image analysis.
- **Python**
  - **NumPy**[9] **and SciPy**[10] – for scientific computing.
  - **scikit-learn**[11] – for machine learning, namely, for neural network.
  - **FilterPy**[12] – for comparison[13] against my own Kalman filter implementation.
  - **CVXOPT**[14] – for convex optimisation problem.
  - **Nao Python SDK**[15] – for communication with Nao robot.
  - **Matplotlib**[16] – for 2D plotting.

### Datasets

I made use of the following labelled datasets for a variety of tasks:

- **CK+ [26]** – for training the Emotion Classifier.
- **Gourier [27]** – for measurement noise estimation of the head movements.
- **UPNA [28]** – for process noise estimation of the head movements & Head Pose Filter evaluation.
- **In-house database (Section 4.2)** – that I collected for Emotion Classifier evaluation & whole system evaluation.

[8] Version 2.4.8. `http://opencv.org/`
[9] Version 1.12.0. `http://www.numpy.org/`
[10] Version 0.18.1. `https://www.scipy.org/`
[11] Version 0.18. `http://scikit-learn.org/stable/`
[12] Version 0.1.5. `http://pythonhosted.org/filterpy/`
[13] Please, see Section 3.5.4 for more details on why I used it for comparison only.
[14] Version 1.1.9. `http://cvxopt.org/`
[15] NaoQi version 2.1. `http://doc.aldebaran.com/2-1/dev/python/index.html`
[16] Version 1.3.1. `http://matplotlib.org/`

# Chapter 3

# Implementation

## 3.1 Overview

This chapter first provides the reader with a detailed system composition described in Section 3.2. Later, it gives the implementation details of individual components of the developed system. Namely, the changes made to detectors AUD and HPD are described in Section 3.3, and the implementation of Emotion Classifier and Head Pose Filter can be found in Sections 3.4 and 3.5 respectively.

## 3.2 From design to implementation

Extending the high-level system architecture proposed in Section 2.10.1, Fig. 3.1 shows the teleoperation system in more detail including components and data flows between them when a new video frame $\Gamma$ is captured.

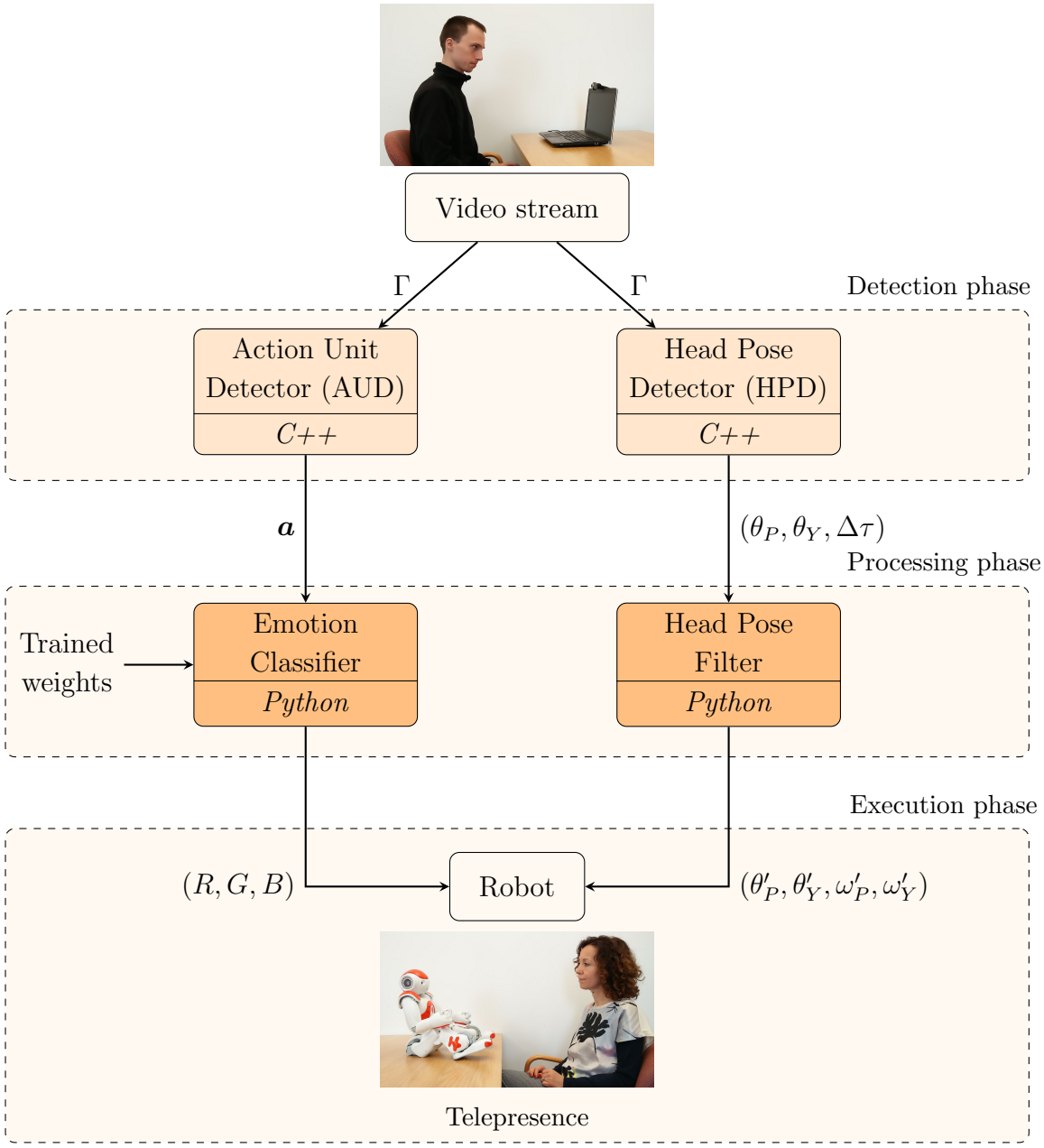



Figure 3.1: Teleoperation system in detail: components, data flows, and layering to phases. *Bright orange:* existing modules modified. *Dark orange:* new modules implemented.

### 3.2.1 Data flows

All the data flows identified in Fig. 3.1 are one-way only and they have the following specification:

- $\Gamma$ – video frame captured.
- $\boldsymbol{a}$ – binary vector indicating which AUs are active in the frame $\Gamma$.

- $(\theta_P, \theta_Y, \Delta\tau)$ – pitch and yaw angles measured for $\Gamma$, and time step between last two frames.
- $(R, G, B)$ – target colour of robot eyes' LEDs corresponding to inferred emotion, i.e. LED commands for $\Gamma$.
- $(\theta'_P, \theta'_Y, \omega'_P, \omega'_Y)$ – target robot head position and velocities to reach this position, i.e. head movement commands for $\Gamma$.

### 3.2.2 Running modes

The developed teleoperation system offers the following running modes.
Based on the source of the system input:

- *online mode* – video stream is taken from a web camera.
- *offline mode* – video stream is taken from a video file.

and depending on the way the system output is displayed:

- *real mode* – output is displayed on actual Nao robot.
- *virtual mode* – output is displayed on simulated robot in Choregraphe application.

### 3.2.3 Order of implementation

Taking into account the dependencies identified during the *Requirements analysis* in Section 2.10.1, I decided on the following order of implementation:

(i) necessary changes to AUD,

(ii) Emotion Classifier implementation,

(iii) changes to HPD,

(iv) Head Pose Filter implementation.

I iterated over this ordering multiple times as proposed in Section 2.10.6, so that on each iteration I improved the robustness of the system based on reflection and assessment of the results from the previous one.

## 3.3 Action Unit Detector and Head Pose Detector

The AUD and HPD originally print their outputs to files. However, reading these files by Python modules while they are being written to would not be efficient for a real-time application. Therefore, I made minor changes to AUD and HPD programs to redirect their outputs. For this I used C++/Python API[1] in the following way:

[1] `https://docs.python.org/2/c-api/index.html`

- AUD calls Python functions of Emotion Classifier when
  - AUD is started – *initRobot$_E$(ipPort)*
  - new frame is analysed by AUD – *updateEmotion$_E$($\boldsymbol{a}$)*
  - AUD is about to close – *finalize$_E$()*
- HPD calls Python functions of Head Pose Filter when
  - HPD is started – *initRobot$_H$(ipPort)*
  - new frame is analysed by HPD – *moveHead$_H$($\theta_P, \theta_Y, \Delta\tau$)*
  - HPD is about to close – *finalize$_H$()*

I also tried other ways such as pipes to achieve the communication between C++ and Python modules, however, they turned out to be more problematic and complicated than the abovementioned C++/Python API.

I made further modifications to AUD and HPD to support various system running modes specified in Section 3.2.2; and to HPD to measure the time interval $\Delta\tau$ between the two most recent frames.

Even though the changes to AUD and HPD were supposed to be straightforward, they took more time than expected since I faced poorly documented code.

## 3.4 Emotion Classifier

### 3.4.1 Overview

In order to implement Emotion Classifier I took the following steps:

1. Data preprocessing – at the beginning I wrote multiple shell and Python scripts to extract and process the data from training dataset (image sequences of people expressing facial emotions).
2. Training the neural network – preliminary results affected further decisions for final training.
3. Real-time emotion classification – using the trained weights.

Next subsections give more details about the above-mentioned steps.

### 3.4.2 Dataset

For emotion classification, I used image sequences from the publicly available CK+ database [26] – widely used benchmark dataset in the field. The CK+ database contains 593 image sequences from 123 subjects. All the sequences start with a neutral face

and end at the peak intensity of a target emotion which enables the calibration of the AUD using the first frame. Each sequence is labelled with one target emotion from the set of eight emotions:

$$Classes_8 = \{neutral, anger, contempt, disgust, fear, happiness, sadness, surprise\}$$

The number of frames per sequence varies between 4–71.

### 3.4.3 Generating training examples

Firstly, I balanced the data so that the number of samples from the most frequent class was cut down to the number of samples from the second most frequent one (since there was one outlier class – neutral). From the remaining image sequences I created .avi videos that were then processed by AUD (running in offline mode) to produce a raw feature vector $\boldsymbol{a}$ (binary sequence of length $N_{AU}$) for each frame in the video stream – resulting in #frames $\times N_{AU}$ matrix $V$ for each video. Where $N_{AU}$ is the number of AUs that AUD was set to detect. Consequently, in order to generate an input feature vector $\boldsymbol{x}$ for the neural network, last $W$ raw feature vectors from $V$ (corresponding to last $W$ frames) were summarised using two different strategies:

- **AVG:** last $W$ frames were averaged per AU, generating a feature vector $\boldsymbol{x}$ of length $N_{AU}$. So in general, the training example is $(\boldsymbol{x}, y)$ where $x_i \in [0, 1]$ and $y \in Classes_8$.
- **CONCAT:** last $W$ frames were concatenated, creating a feature vector $\boldsymbol{x}$ of length $W \times N_{AU}$. Thus, the training example is $(\boldsymbol{x}, y)$ where $x_i \in \{0, 1\}$ and $y \in Classes_8$.

For example, for $N_{AU} = 4$, $W = 2$ and $V = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$, the training examples for class *disgust* would be

$$(\boldsymbol{x}, y)^{(AVG)} = (\begin{bmatrix} 1.0 & 1.0 & 0.5 & 0.0 \end{bmatrix}^T, disgust)$$

and

$$(\boldsymbol{x}, y)^{(CONCAT)} = (\begin{bmatrix} \underbrace{0 \quad 1 \quad 1 \quad 0}_{\text{penultimate frame}} & \underbrace{1 \quad 0 \quad 1 \quad 0}_{\text{last frame}} \end{bmatrix}^T, disgust)$$

when generated by AVG and CONCAT strategies respectively.

### 3.4.4 Training and cross-validation

I implemented the neural network with one hidden layer using Python machine learning library *scikit-learn* [29]. Specifically, I used a multilayer perceptron model with rectified linear unit (ReLU) as an activation function for the hidden layer and softmax for the

output node – as described in Section 2.8. The cross-entropy loss function was minimised by solver L-BFGS, which uses an approximation of inverse Hessian matrix to steer the search. This solver is especially useful in this case due to its fast convergence on small datasets [29] and its robustness to hyperparameter changes. I used the stratified 5-fold cross-validation to tune the following hyperparameters: hidden layer size $hls$, window size $W$, and regularization parameter $\alpha$. I split the cross-validation process into two phases:

- Phase 1: grid search over hidden layer size $hls$ and window size $W$. For each hyperparameter combination I ran 50 random initializations for weights and chose the combination achieving the highest validation accuracy.
- Phase 2: grid search for optimal regularization $\alpha$ given the $hls$ and $W$ from the Phase 1.

I repeated the same procedure for both (input feature vector generation) strategies AVG and CONCAT, and selected the better one. For the best hyperparamaters of the chosen strategy I used the whole dataset (all 5 folds) to train the weights for later testing and real-time classification.

### 3.4.5 Results and implications

The results from data investigation, training and cross-validation yielded important implications that affected the later development procedures.

#### Relevant action units

After processing all of 593 image sequences from the database I found out that only 7 AUs (namely, inner brow raiser (AU1), outer brow raiser (AU2), brow lowerer (AU4), cheek raiser (AU6), lip corner puller (AU12), lips parted (AU25), and jaw drop (AU26)) out of a maximum 12 detectable were ever active. Therefore, from now on we assume

$$N_{AU} = 7$$

since the Emotion Classifier would not be able to learn any useful information from the other 5 always-zero AUs.

#### Reduction of the number of classes

The preliminary results showed that only 4 out of 8 original classes of emotions were reliably distinguished. Specifically, as shown by Fig. 3.2, the classification accuracy on the validation set was much better for the four emotions including neutral (59%), disgust (75%), happiness (81%) and surprise (90%) as compared to the other four emotions: anger (27%), contempt (39%), fear (12%), and sadness (14%). The latter 4 classes were completely confused which was probably caused by AUD not being able to detect enough

distinct AUs to distinguish all these 8 classes of emotions. Therefore, further training was done only with the following set of 4 classes:

$$Classes_4 = \{neutral, disgust, happiness, surprise\}$$

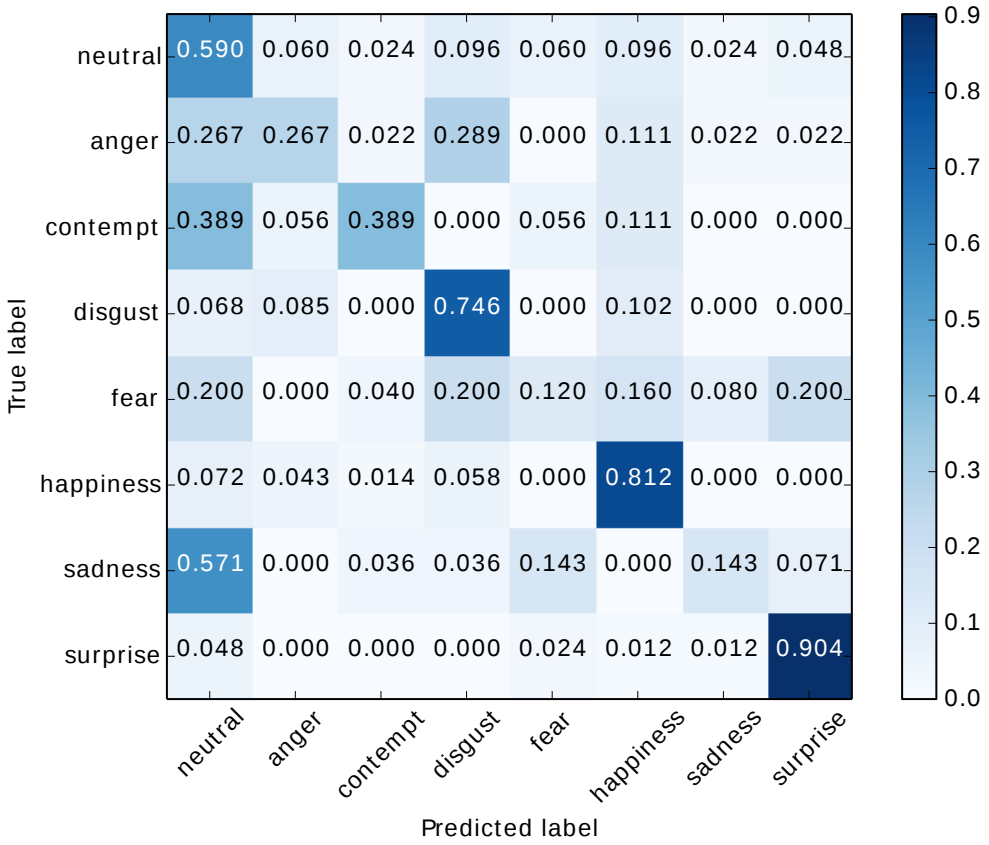



Figure 3.2: Normalised confusion matrix for original 8 classes from CK+ database: summed over 5 validation folds after the optimal hyperparameters were found. Strategy used for generating input feature vectors was AVG. High values on the diagonal indicate correct classification.

### Results for the final dataset of four emotion classes

The refined balanced training set of four classes of emotions $Classes_4$ consisted of 294 examples with the distribution over classes shown in Fig. 3.3. The videos used for this final training set were of length 6–65 frames with mean of 16 frames as described by the distribution in Fig. 3.4.

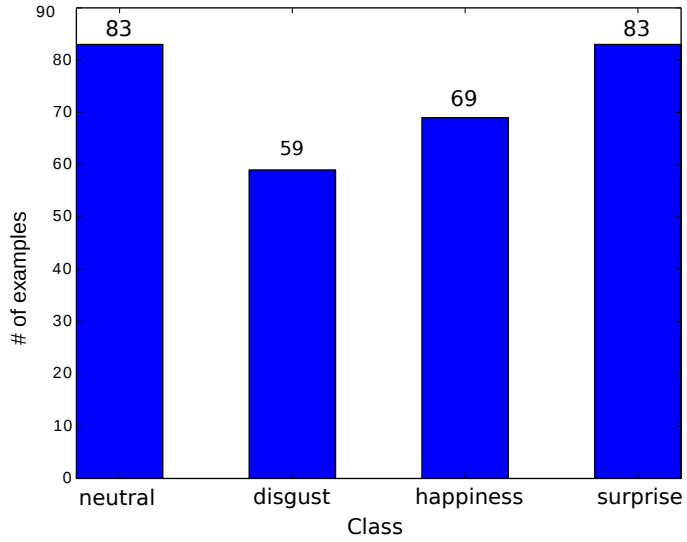



Figure 3.3: Distribution of the refined set of CK+ videos (examples) over final 4 classes used for training the Emotion Classifier. Total count of examples is 294.

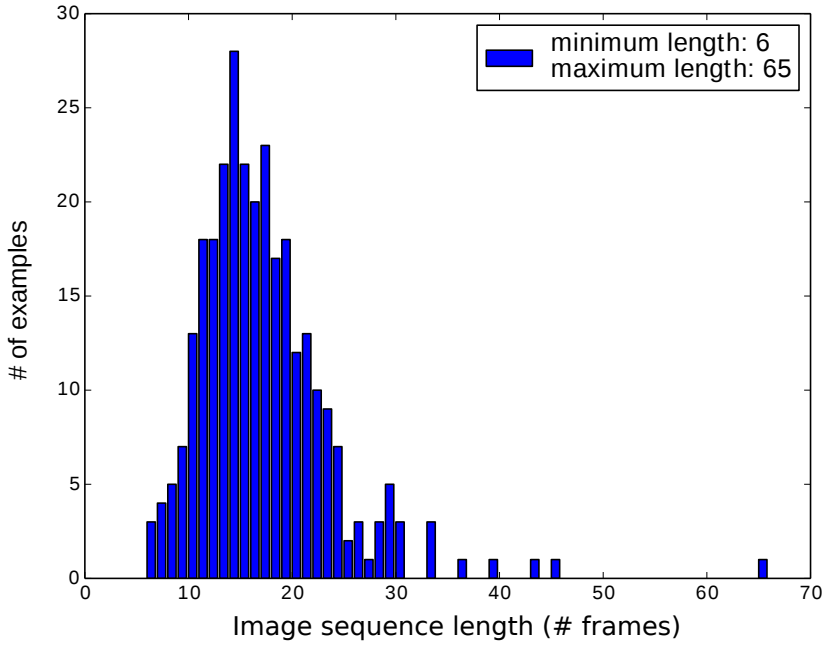



Figure 3.4: Distribution of the refined set of CK+ videos (examples) over image sequence length (frame count). Number of frames ranges from 6 to 65 with mean of 16.

An interesting investigation of the facial action unit activity over time is illustrated in Fig. 3.5. All the videos were right-aligned in time and the number of times the AU was active over all videos was calculated for each frame. The results underline the fact that the image sequences from CK+ dataset end at the peak intensity of the facial expression. Furthermore, the observation that the interesting region consists only from last few frames

approves the choice of the small range of window sizes $W$ used in the cross-validation process (see next page).

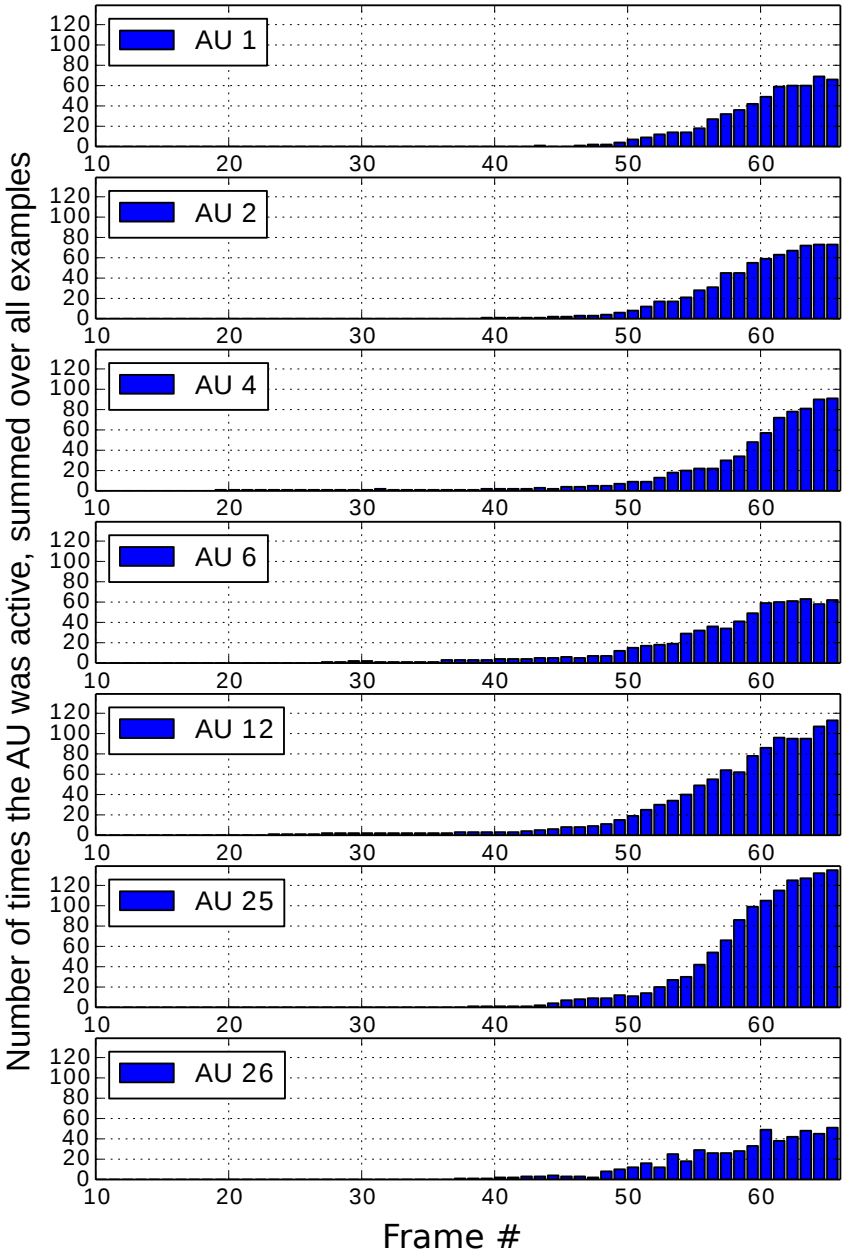



Figure 3.5: Activity of 7 relevant action units in time: counts are summed over all examples from the refined set of CK+ videos. All time series are right aligned in time (to the end of the video recording) since they have different number of frames as shown in Fig. 3.4. The interesting region consists only from last few frames.

The data partitions used for training, validation, and testing are shown in Tab. 3.1.

Table 3.1: The samples used for training, validation, and testing of Emotion Classifier neural network.

| | Training | Validation | Testing |
|---|---|---|---|
| Data source | CK+ | CK+ | In-house database |
| Sample size | 232 | 62 | 224[1] |
| | 80% | 20% | |

In the first phase of the 5-fold cross-validation I searched for optimal hidden layer size $hls \in \{1, \ldots, 29\}$ and window size $W \in \{1, \ldots, 6\}$ – only up to 6 because the shortest examples had only 6 frames, i.e. if higher $W$ was used it would not be possible to create training examples for CONCAT strategy for the shortest videos. Then in the second phase I searched for optimal regularization $\alpha \in \{10^{-6}, 10^{-5}, \ldots, 10^{2}\}$ given the $hls$ and $W$ from Phase 1. The results from hyperparameter tuning for each strategy are summarised in Tab. 3.2. and the corresponding figures from the cross-validation process can be found in Appendix A as follows:

- AVG strategy:
  - Phase 1: Fig. A.1a, Fig. A.2a
  - Phase 2: Fig. A.3a
  - Confusion matrix over all validation folds: Fig. A.4a
- CONCAT strategy:
  - Phase 1: Fig. A.1b, Fig. A.2b
  - Phase 2: Fig. A.3b
  - Confusion matrix over all validation folds: Fig. A.4b

Table 3.2: Comparison of AVG and CONCAT strategies. Mean training and validation accuracies with standard deviations over 5 validation folds for the best hyperparameters $hls$, $W$, $\alpha$.

| | AVG | CONCAT |
|---|---|---|
| Training accuracy | $0.867 \pm 0.016$ | $0.853 \pm 0.014$ |
| Validation accuracy | $0.834 \pm 0.045$ | $0.800 \pm 0.069$ |
| $hls$ | 4 | 3 |
| $W$ | 5 | 2 |
| $\alpha$ | $10^{-3}$ | $6 \times 10^{-2}$ |

[1]Both study groups from the controlled experiment from evaluation phase, see Section 4.2 for more details.

The results show that AVG representation yields higher accuracy on the validation set as compared to CONCAT representation. In other words, it was better to use the low-dimensional feature vectors generated by averaging rather than by concatenating AU information from the last $W$ frames. Therefore, for the real-time classification and also for Emotion Classifier testing I trained the neural network using the AVG strategy (with optimal hyperparameters set during the cross-validation as $hls = 4$, $W = 5$ and $\alpha = 10^{-3}$) and using the whole dataset (all 5 folds), for recognising four emotions.

### 3.4.6 Testing

The final trained model was tested in user study both quantitatively (Section 4.2) using an in-house test database I collected, and qualitatively (Section 4.4) via web survey.

### 3.4.7 Real-time emotion classification and displaying on the robot

Finally, for real-time emotion inference and displaying on the robot, I implemented the three main Python functions as proposed in Section 3.3.

- $initRobot_E(ipPort)$
  Called by AUD at its start-up.
  **Task:** Set up a connection to the robot using the given IP and port.

- $updateEmotion_E(\boldsymbol{a})$
  Called by AUD after a new frame was analysed.
  **Task:** Infer an emotion: using the current $\boldsymbol{a}$ and past $W - 1$ input arguments create the input vector $\boldsymbol{x}$ (using AVG strategy), then apply the trained weights to it as described in an example in Section 2.8. The emotion is then associated with a colour of LEDs in Nao's eyes as explained in Section 2.5.1, for the four emotions in question: neutral emotion with no colour, disgust with red, happiness with blue, and surprise with green. The LED $\sim$ emotion associations used are also shown in Fig. 3.6.


Figure 3.6: The LED $\sim$ emotion associations used.

  Lastly, if the emotion has changed since the last frame, an LED command with the target colour encoded as $(R, G, B)$ is sent to the robot.

- $finalize_E()$
  Called by AUD at its shut-down.
  **Task:** Close the connection with the robot, and optionally show and save the data

processed during the run-time. An example of a visual output is illustrated in Fig. 3.7.

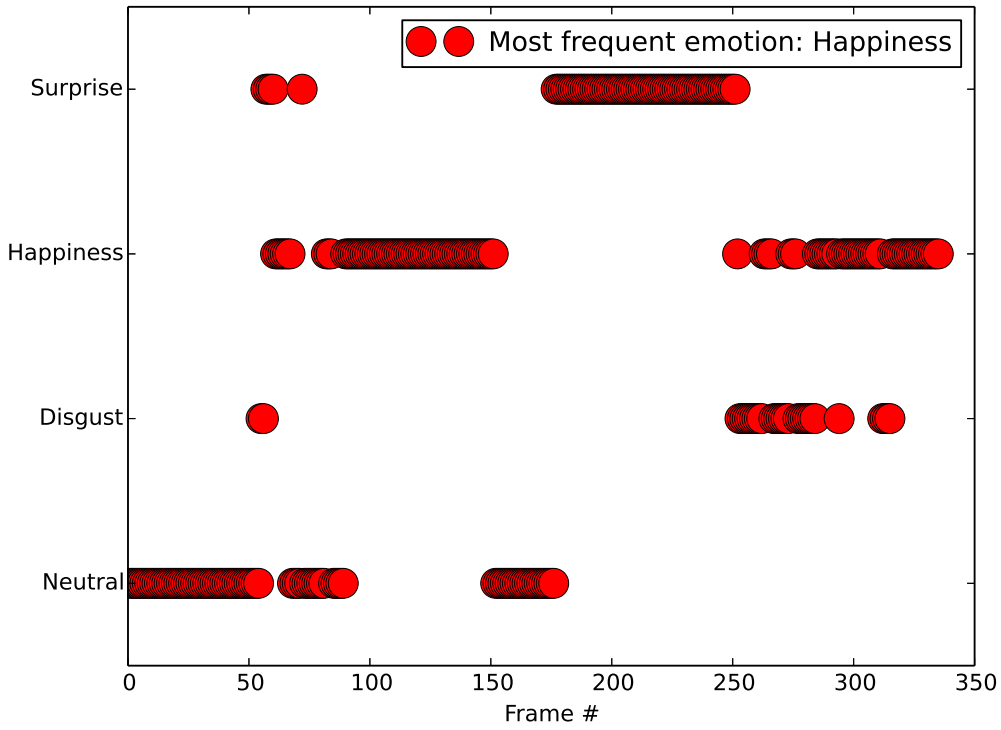



Figure 3.7: Emotion Classifier visual output: emotions as inferred during run-time.

## 3.5 Head Pose Filter

### 3.5.1 Overview

This section presents the concrete instantiation of Kalman filter based on Section 2.9. It further explains how the filter was initialised and how its parameters were determined and set. Then, it briefly compares this implementation with another existing one. It concludes with a description on how the implemented Kalman filter was incorporated and used for real-time head pose filtering.

### 3.5.2 Kalman filter instantiation

I implemented the constrained-state Kalman filter with a minimum jerk model, using the underlying theory from Section 2.9. The main purpose was to avoid jerky head movements and constrain them to the robot's physical operating limits.

I implemented the Kalman filter as a separate Python class. The main method of this class is *update*$(\theta, \Delta\tau)$ which is called each time a new measurement of head pose angle $\theta$ comes in. Since the frame rate is not constant and varies slightly over time, I set the instantaneous filter time step $\Delta T_t$ in equation (2.1) to the measured time $\Delta\tau$ between last two frames which in turn affects the transition matrix $F_t$ used in the filter update step.

For the optimization (quadratic programming) problem given by equation (2.3) I used the CVXOPT library [30].

Since the robot can perform yaw and pitch rotations, we need two separate instances of Kalman filter, one for yaw and one for pitch movement. These two filters operate independently[3] and differ only in initialisation and parameter values – specified in the following section.

### 3.5.3 Initialisation

Both filters were initialised with

$$P_0^+ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \hat{x}_0^+ = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$$

since the initial state (head pose) is expected to be at zero angle with zero angular velocity and acceleration, however, it is not known exactly. Thus, there are non-zero values of variances on $P_0^+$'s diagonal which affect how quickly the filter begins to make sensible estimates. Initially, no correlation (zeros on off-diagonal elements) between position, velocity and acceleration is assumed, however, it is unlikely that they remain uncorrelated as the matrix $P_t^+$ updates.

#### State constraints

As mentioned in Section 2.5.2 the Nao robot's head can operate only within specific bounds. I constrained only the angle and the angular velocity since these are the only parameters I can send to the robot in commands. In reality the pitch angle constraints vary slightly with the yaw angle, as shown in Tab. 2.2. However, in order to simplify and apply the specified Kalman filter, I used the tightest bounds. The state constraints used are summarised in Tab. 3.3.

[3] Assumed for simplification, but in general, they could be correlated (which would be an interesting idea for further investigation and research).

Table 3.3: State constraints for Kalman filter. Robot operates within the angle range $[\theta_{min}; \theta_{max}]$ and angular velocity range $[\omega_{min}; \omega_{max}]$.

| | Yaw | Pitch |
|---|---|---|
| $\theta_{min}$ [rad] | -2.0857 | -0.330041 |
| $\theta_{max}$ [rad] | 2.0857 | 0.200015 |
| $\omega_{min}$ [rad/sec] | -8.26797 | -7.19407 |
| $\omega_{max}$ [rad/sec] | 8.26797 | 7.19407 |

Lastly, the measurement and process noise parameters need to be specified. These were estimated using Python and shell scripts. The details follow.

### Measurement noise estimation

The measurement noise determines how much we can believe our measurements from sensor – in this case the web camera. The higher this noise is the less the Kalman filter relies on the obtained measurements when making predictions.

To estimate the measurement noise parameters $r_Y$ and $r_P$ for yaw and pitch angles respectively, I used the database with labelled head pose images of Gourier et al. [27]. I measured the variances $(r_Y, r_P)$ in head pose measurements $(\theta_Y, \theta_P)$ from HPD using a video stream of a fixed head pose (still image from database projected on another screen) captured by my web camera. I took 1000 measurements for each of 3 subjects at 5 different (ground truth) angle pairs

$$\left(\theta_Y^{true}, \theta_P^{true}\right) \in \{(0°, 0°), (-30°, +30°), (+30°, +30°), (+30°, -30°), (-30°, 30°)\}$$

resulting in $M = 15000$ data points. In general, the measurement noise was then estimated according to

$$r_\Lambda = Var(\theta_\Lambda) = \frac{1}{M}\sum_{i=1}^{M}(\theta_{\Lambda,i} - \theta_\Lambda^{true})^2, \qquad \Lambda \in \{Y, P\}$$

giving the concrete values of

$$r_Y = (8 \pm 5) \times 10^{-5} \text{ rad}^2, \qquad r_P = (6 \pm 2) \times 10^{-5} \text{ rad}^2$$

As a sanity check we can convert these variances to standard deviations in degrees and see that they are reasonably small: $\sqrt{r_Y} = 0.5°, \sqrt{r_P} = 0.4°$.

It is crucial that the same camera was used for the estimation of measurement noises as well as for all head pose replications later on, because the measurement noise is a camera-specific parameter.

### Process noise estimation

The process noise determines how much we can believe our state transition model – in this case the minimum jerk model given by equation (2.1). The higher the noise is the less the Kalman filter relies on the transition model when making predictions.

For the process noise estimation, namely, $q_Y$ and $q_P$ for yaw and pitch angles respectively, I used UPNA database [28] that contains video recordings of head movements of 12 subjects (with 10 videos per subject and 300 frames per video) labelled frame-by-frame. I processed these videos by HPD with Kalman filter connected to its output, and compared the results from the filter with the ground truth values from the database. More specifically, I searched for the process noise $\tilde{q}$ that minimizes the sum of squared errors between outputs from the filter, $f_i(q)$, and ground truth values, $\vartheta_i$, over all frames of all the videos (i.e. over all data points $i$). This was done separately for yaw and pitch angles according to

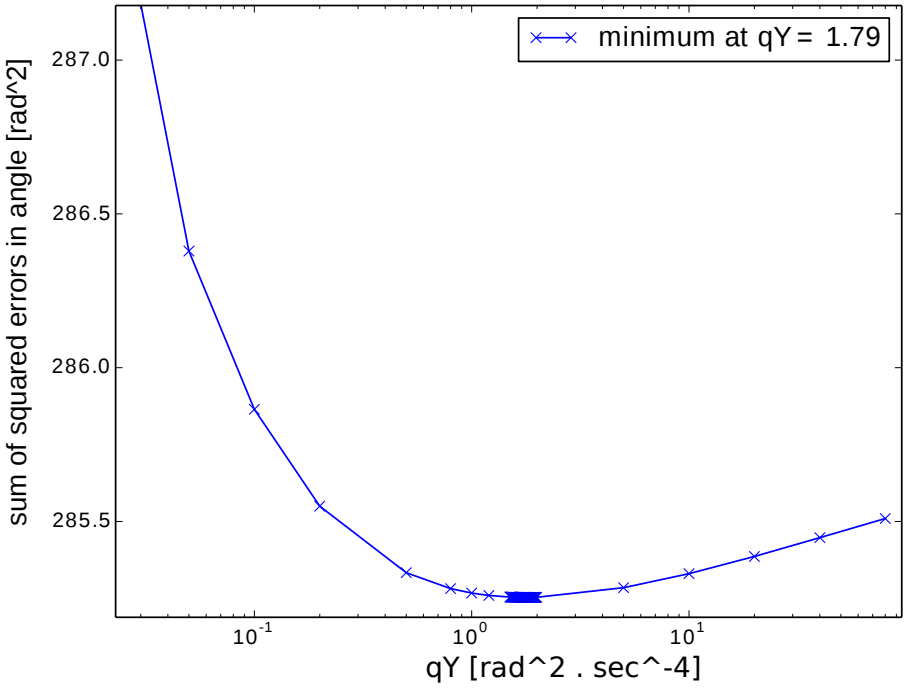
$$\tilde{q}_\Lambda = \arg\min_{q_\Lambda} \sum_i \left(\vartheta_{\Lambda,i} - f_i(q_\Lambda)\right)^2, \qquad \Lambda \in \{Y, P\}$$

For this optimization problem, I used 80% of the database since the Head Pose Filter was later evaluated on the remaining 20% of the data using the optimal parameters learned on the first 80% of data. Finally, for real-time filtering, the whole database was used to estimate the process noises. The optimal parameters for yaw and pitch rotations are shown in Tab. 3.4.

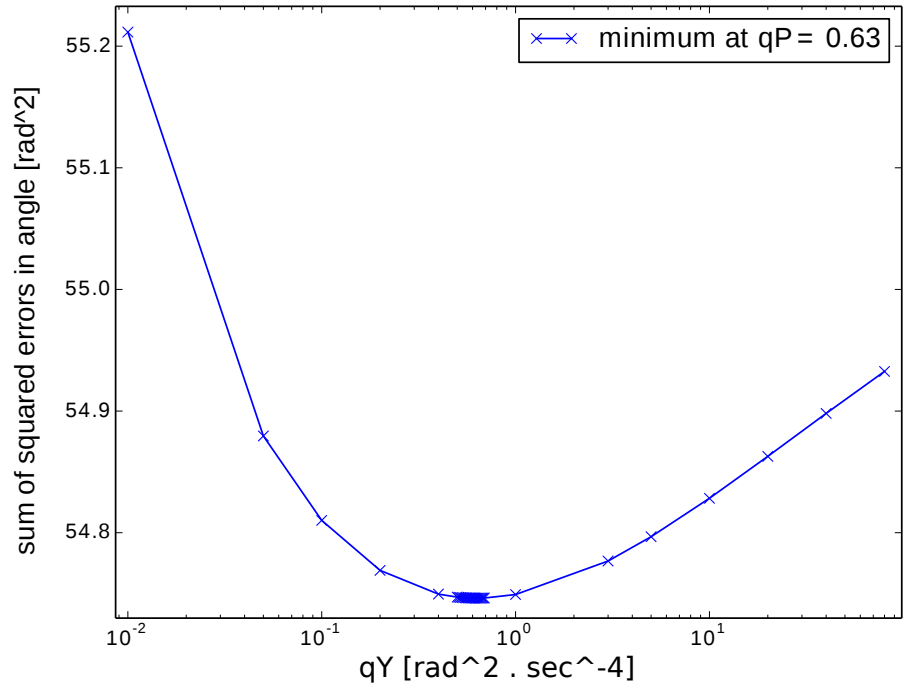
Table 3.4: Process noise estimation. Optimal process noises $q_Y$ and $q_P$ for yaw and pitch rotations respectively.

| # Data points | Percentage | # Subjects | $q_Y$ [rad$^2$.sec$^{-4}$] | $q_P$ [rad$^2$.sec$^{-4}$] |
|---|---|---|---|---|
| 30000 | 80% | 10 | 1.40 | 0.31 |
| 36000 | 100% | 12 | 1.79 | 0.63 |

Figures 3.8a and 3.8b show the optimization process (to estimate the process noises) for yaw and pitch rotations when using the whole database (100% of data).




(a) Yaw




(b) Pitch

Figure 3.8: Finding optimal process noise $q$ (x-axis) that minimizes the sum of squared errors (y-axis) between outputs from filter and ground truths from UPNA database (using 100% of data). Separately for yaw (a) and pitch (b) angles, the minima were found at $\tilde{q}_Y = 1.79$ rad$^2$.sec$^{-4}$ and $\tilde{q}_P = 0.63$ rad$^2$.sec$^{-4}$ respectively. These optimal process noises were then used in real-time replication. Minimum step size: 0.01 rad$^2$.sec$^{-4}$.

### 3.5.4 Comparison and performance analysis

Even though I found an existing Python Kalman filtering library *FilterPy*, I had to make my own simplified implementation for this particular purpose of the project because the *FilterPy* library did not include the constrained-state extension I needed. This had the additional effect that I ended up understanding the inner working of Kalman filter much better.

I could then compare my own implementation with the existing one for both correctness and performance by testing on cases where constraints did not activate. The results show that my implementation is about 2.6 times faster than *FilterPy's* when compared over 2200 data points.

### 3.5.5 Real-time head pose filtering and displaying on the robot

As proposed in Section 3.3, I implemented the three main Python functions for real-time head pose filtering and displaying on the robot. These functions utilise two instances of Kalman filter class.

- *initRobot$_H$(ipPort)*
  Called by HPD at its start-up.
  **Task:** Set up a connection to the robot using the given IP and port.
  Initialise two Kalman filters – for yaw and pitch rotations.

- *moveHead$_H$($\theta_P, \theta_Y, \Delta\tau$)*
  Called by HPD after a new frame was analysed.
  **Task:** Filter the measured head pose $(\theta_P, \theta_Y)$ by calling the *update($\theta, \Delta\tau$)* methods of Kalman filter instances. Hence, the target robot head position and velocities to reach this position are obtained and a head movement command including them as $(\theta'_P, \theta'_Y, \omega'_P, \omega'_Y)$ is sent to the robot.

- *finalize$_H$()*
  Called by HPD at its shut-down.
  **Task:** Close the connection with the robot, and optionally show and save the data processed during the run-time. An example of a visual output is illustrated in Fig. 3.9.

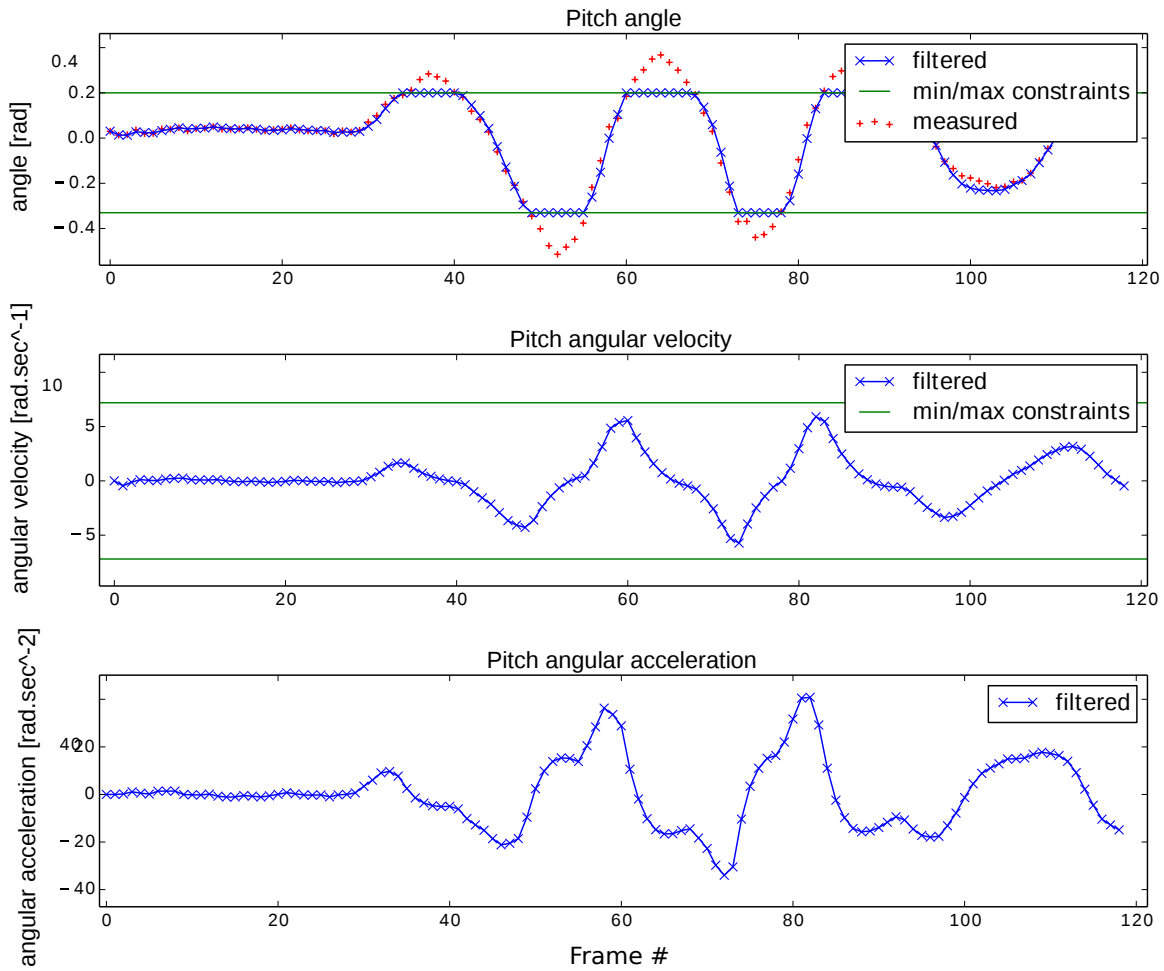



Figure 3.9: Head Pose Filter visual output: measured and filtered state estimates (angle, angular velocity, angular acceleration) for pitch rotation, along with upper and lower bounds on angle and angular velocity. During 120 captured frames.

# Chapter 4

# Evaluation

## 4.1 Overview

The developed teleoperation system was evaluated both quantitatively and qualitatively. The evaluation phase was split into three parts:

1. **Emotion Classifier quantitative evaluation (Section 4.2):** testing of Emotion Classifier on the in-house database I collected in the controlled experiment involving human participants.
2. **Head Pose Filter quantitative evaluation (Section 4.3):** using the labelled head pose video database I assessed the absolute errors between the head pose estimates produced by the filter and ground truth labels from the database.
3. **Whole system qualitative evaluation (Section 4.4):** using a web-based survey respondents rated the replication quality of both emotions and head movements.

Finally, as a proposed extension I measured and evaluated the replication latency in Section 4.5.

## 4.2 Emotion Classifier evaluation

I built an in-house dataset to test the trained model on a set of unseen samples. I recruited a total of 28 participants in two separate groups, and asked them to perform the four emotions (specified by $Classes_4$) twice in front of my web camera. Their participation was approved by Computer Laboratory Ethics Committee (Appendix B) and they were given the instructions shown in Appendix C.

- **Group 1** – 16 participants, starting with a neutral face followed by the display of the target emotion for the rest of the recording.
  ($16 \times 4 \times 2 = 128$ test clips in total)

- **Group 2** – 12 participants, displaying only the target emotion for the full length of the recording (no neutral reference emotion at the beginning – considered as a more challenging group).
  $(12 \times 4 \times 2 = 96$ test clips in total).

Representative examples for each emotion are shown in Fig. 4.1.

Figure 4.1: Representative examples from recordings from the in-house test dataset (from Group 1) for each of the 4 emotions: *neutral*, *disgust*, *happiness*, *surprise* (left to right).

In total, I collected 224 test clips, each had a duration of approximately 15 seconds. The Emotion Classifier scanned each test clip over a sliding window of size $W$, and delivered the recognised emotion at each time step. The final decision for the whole clip was then made by majority voting of the outputs from all temporal windows.

I evaluated the Emotion Classifier using two approaches:

1. AUD was *not* provided with a calibration clip, and used the first frame as a reference for calibration.
2. AUD was provided with a calibration clip, that was the neutral clip of the participant.

The corresponding results are summarised in Tab. 4.1. We can observe that the second approach improved the performance, in particular, the recognition accuracy for Group 2 significantly increased from 32% to 75%. This implies that the AUD relies on a reference neutral face in order to work correctly, and so the classification does not rely on the calibration video if the analysed recording starts with a neutral face.

Table 4.1: Emotion classification accuracy (%) on the test set with and without calibration video.

| Calibration video | Group 1 | Group 2 | Average |
|---|---|---|---|
| *not* provided | 71.9 | 32.3 | 52.1 |
| provided | 73.4 | 75.0 | 74.2 |

I further examined which emotion was the easiest and/or the hardest to recognise. In Fig. 4.2, I present the confusion matrix for Group 1 with calibration video. As expected, the classifier was very successful at recognising neutral. The classifier was successful at recognising surprise (81.2%) and happiness (75.0%), but not disgust (37.5%). Usually the

performance of the trained model is assessed relative to other methods and models tested on the same dataset. However, no one else performed testing on my database and so I could compare my results only with the baseline – chance level[1] – in this case 25%. To conclude, the accuracies for neutral, happiness and surprise emotions were well (more than three times) above the chance level, whereas for disgust the achieved accuracy was just above the baseline. This was probably due to the difficulty of expressing the disgust emotion as reported by many of the participants. The obtained accuracies were satisfactory to proceed to the qualitative evaluation using web survey (Section 4.4).

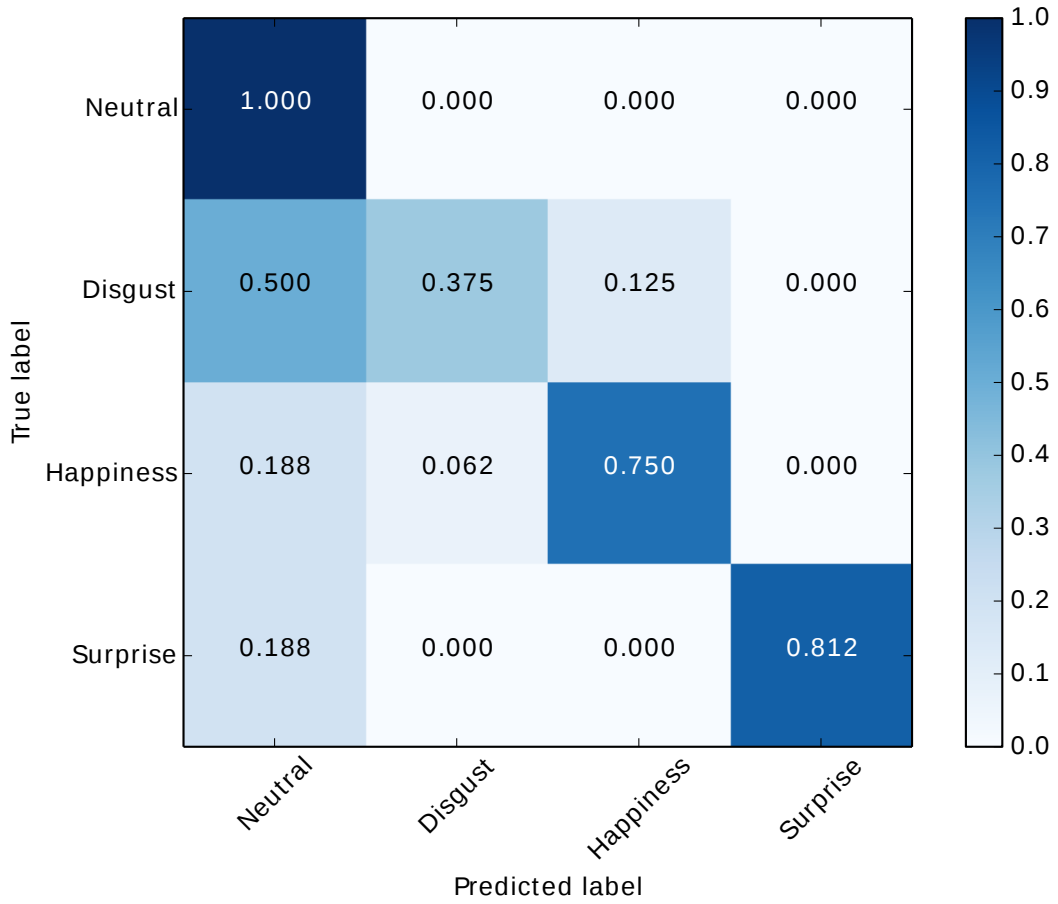



Figure 4.2: Normalised confusion matrix from Emotion Classifier testing on in-house database – participants from Group 1 with calibration video provided. Accuracies for all 4 classes of emotions are above the chance level (25%).

[1]Expected accuracy if the classes were assigned by random guessing.

## 4.3 Head Pose Filter evaluation

The Head Pose Filter was evaluated on the remaining 20% of the data (6000 data points, 2 subjects) from UPNA database using the process noise parameters ($q_Y$, $q_P$) optimal for the first 80% of data. Fig. 4.3 shows the absolute errors (between filtered head pose estimates and ground truths from database) as they fall into different angle ranges. The absolute errors shown come from all the frames of all the videos from the held-out test set. In degrees, the mean absolute error for yaw angle is $4.1° \pm 3.8°$ and $2.5° \pm 2.5°$ for pitch angle.

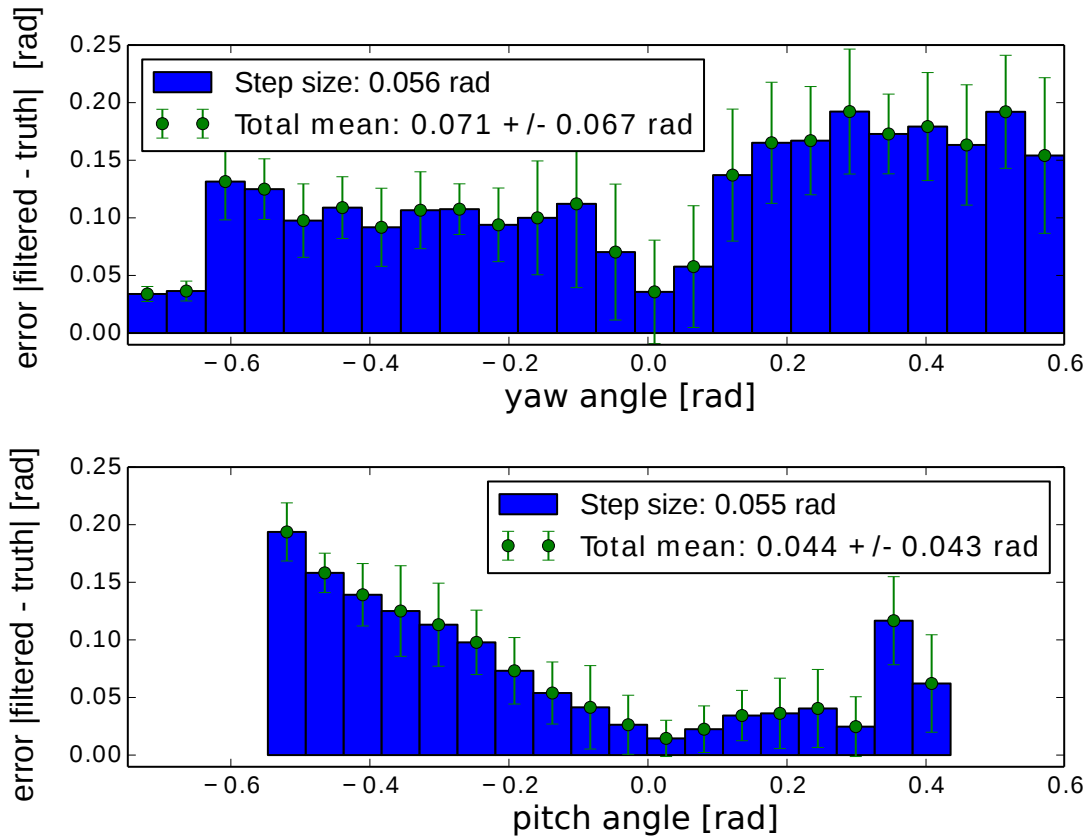



Figure 4.3: Head Pose Filter evaluation on the UPNA database: absolute errors (y-axis) between filtered head pose estimates and ground truth values for different angles (x-axis). Yaw rotation (top), pitch rotation (bottom).

Looking at Fig. 4.3, we can conclude that the mean absolute error over different angle ranges is relatively small, but this error increases as the absolute value of the angle increases.

## 4.4 Whole system evaluation

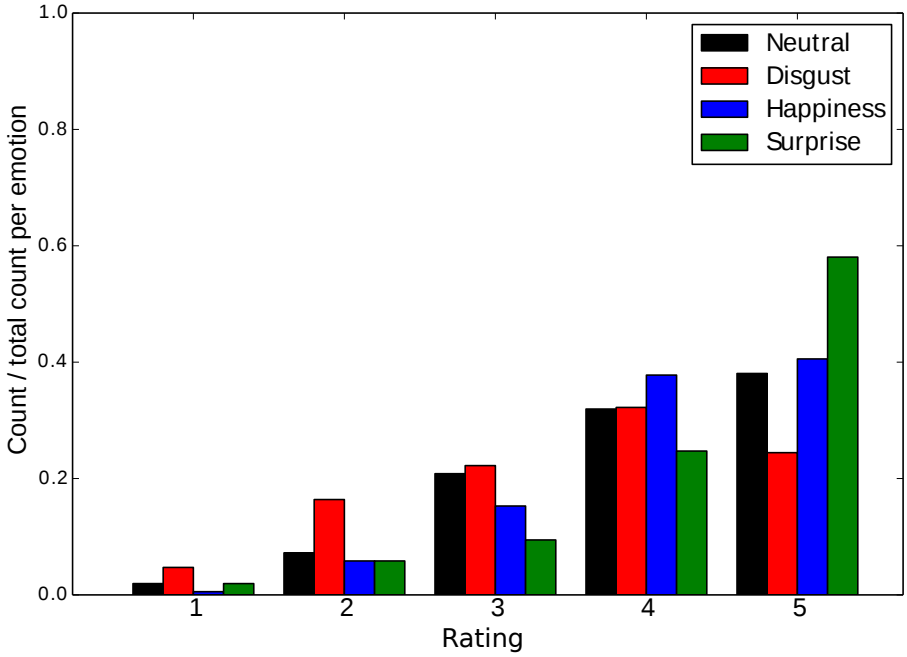
The video recordings of 10 best-performing (highest testing accuracy) participants from Group 1[2], resulting in a total of 80 clips, were replicated on the robot. Then, I created 80 *human-robot videos* (participant's clip along with the clip of the robot imitating their emotions and head movements), as illustrated in Fig. 4.4. An example of an image sequence from the replication process can be found in Appendix D.

Figure 4.4: Representative examples of *human-robot videos* used in the web survey. Emotion *disgust* (left) and *surprise* (right).

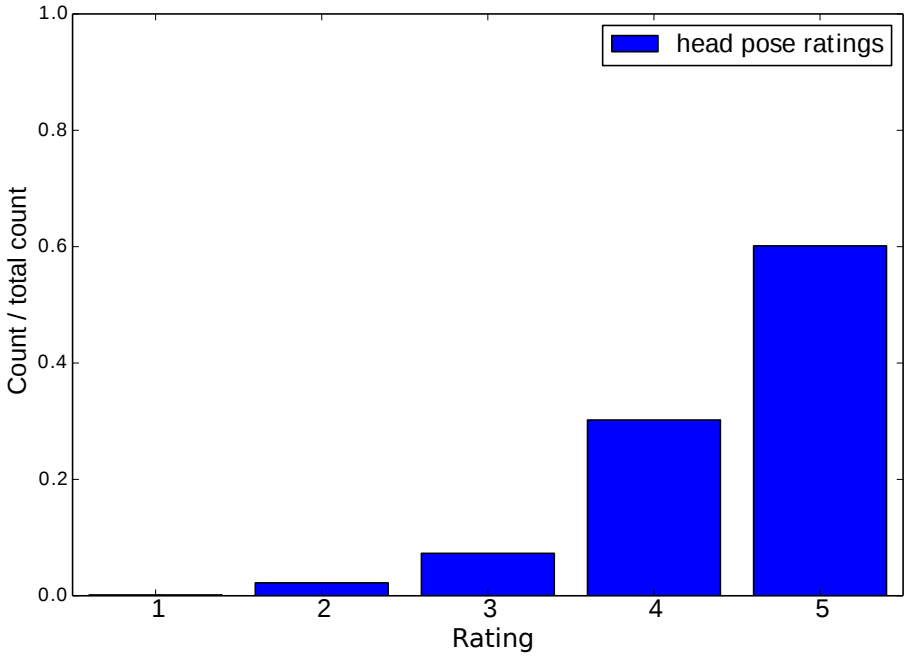
Using Google Forms I prepared an anonymous online survey where I asked 18 external observers to watch each of 80 *human-robot videos*, and rate the replication quality on a 5-point Likert scale. The external observers were also provided with emotion-colour mappings, as I was not concerned with how well a colour represented an emotion. An example of such a web survey question is shown in Appendix E.

Histograms of gathered responses for emotion and head pose replication are shown in Figures 4.5a and 4.5b respectively. We can observe that the replication of surprise was considered most accurate, whereas disgust was not communicated that well. Neutral and happiness were judged similarly, and received better ratings than disgust.

[2]Group 1 was chosen because it was considered as less challenging group, see Section 4.2. Also, the calibration videos were provided to AUD since it was shown that it results in higher testing accuracy, see Tab. 4.1.




(a) Emotions




(b) Head movements

Figure 4.5: Histograms of responses for replication quality of emotions (a) and head movements (b) obtained in web survey. Ratings 1–5 on x-axis correspond to 5-point Likert scale (Very poor–Very good).

In order to assess the observers' agreement, I calculated the intraclass correlation coefficient (ICC) [31] using MATLAB. In this case the ICC is based on the average of $k$ ratings (from $k = 18$ raters) and each target is rated by a different set of raters from a population

of raters. (This corresponds to ICC(1,k) in Shrout and Fleiss convention.) The results are presented in Tab. 4.2. According to the guidelines for interpretation of ICC measures [32], the observers' consensus would be considered *excellent* for all the target sets, except for the head pose which would be interpreted as *good*.

Table 4.2: Intraclass correlation coefficients (ICC) to assess the inter-rater agreement between respondents in the web survey. CI refers to confidence interval.

| Rated target set | ICC | 95% CI |
|---|---|---|
| Head pose | 0.7278 | [0.6328; 0.8074] |
| Neutral emotions | 0.8896 | [0.8047; 0.9489] |
| Disgust emotions | 0.9212 | [0.8606; 0.9635] |
| Happiness emotions | 0.8021 | [0.6498; 0.9083] |
| Surprise emotions | 0.9398 | [0.8934; 0.9721] |
| All emotions | 0.9184 | [0.8899; 0.9423] |
| Head pose and all emotions | 0.9055 | [0.8828; 0.9256] |

To conclude, the results obtained via web survey show that the developed teleoperation system can replicate emotions and head movements with a high inter-rater agreement, namely, $ICC_E = 0.91$ and $ICC_{HP} = 0.72$ for emotions and head movements replication respectively. The ratings for the disgust emotion further suggest that this emotion is the most difficult to replicate on the fly.

## 4.5 Replication latency

The measurement and evaluation of the system's replication latency was one of the proposed extensions I managed to accomplish.

I focused on the processing latency defined as the time from the frame capture to the dispatch of the commands to the robot. This was measured separately for emotions and head movements over 4.5 minutes of replication time. The measured latencies were $\tau_E = 49 \pm 7$ ms and $\tau_{HP} = 20 \pm 3$ ms for emotion replication and head pose replication, respectively. Fig. 4.6 shows the latency measurements.

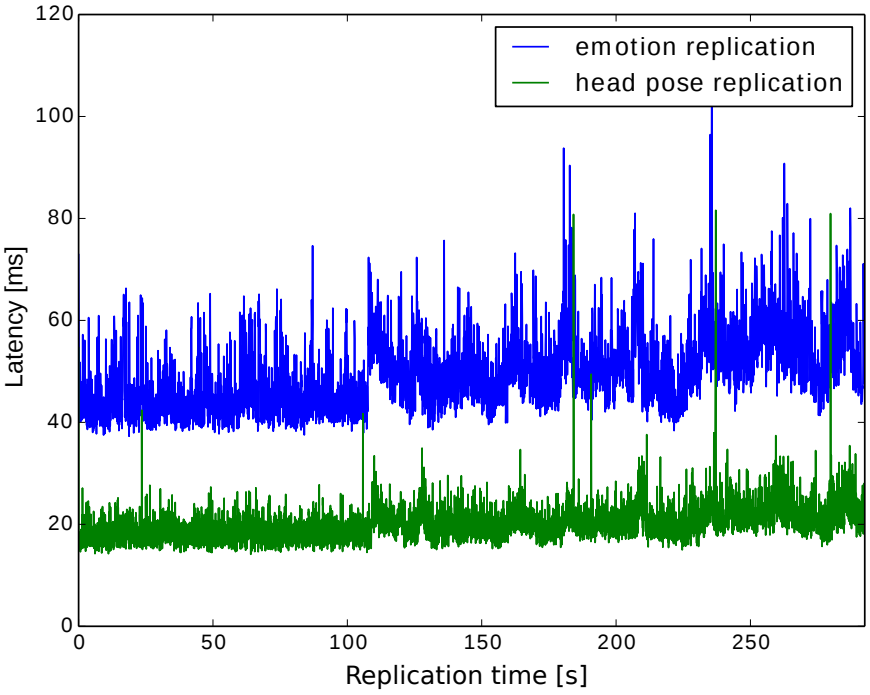



Figure 4.6: Processing latency measurements: emotion replication latency $\tau_E = 49 \pm 7$ ms, head pose replication latency $\tau_{HP} = 20 \pm 3$ ms.

The processing latency measurements show that the emotion replication is more computationally intensive than the head pose replication which was expected ($\tau_E > \tau_{HP}$).

However, these measurements do not include the delay caused by communication over the network[3] which would constitute the major portion of the overall delay in a teleoperation system. Thus, I can compare only the processing delay of my system with the head pose replication system proposed by Agarwal [15] – described in Section 1.3. They measured the overall replication latency including the network communication delay to be 100 to 170 ms and estimated the portion caused by communication over the network to be 34 to 104 ms. Thus, I can conclude that when compared to [15], taking an average as a single measure for comparison, I achieved a lower processing latency (by approximately 70%) for head pose replication. However, making such a one-to-one comparison is always challenging, moreover, they used a different robotic platform.

[3]In all my experiments I sent the required commands to the robot through an Ethernet cable directly from my laptop.

# Chapter 5

# Conclusion

To conclude my project, I am very grateful for this experience since I got exposed to multiple areas of Computer Science and I learned a lot throughout the process. In this chapter, the summary of accomplished work and results are shown in Section 5.1, the take-home messages and outputs from the project are outlined in Section 5.2, and finally, further ideas, possible improvements and extensions are proposed in Section 5.3.

## 5.1 Summary

In this project I successfully met all success criteria and developed a new teleoperation system that replicates human facial expressions of emotions (neutral, disgust, happiness, surprise) and head movements in real-time in a non-invasive manner on the humanoid robot Nao. Using the AUD and HPD detectors provided by my supervisors, I trained a neural network for emotion classification that uses the action units detected by AUD to infer emotions. I implemented the constrained-state Kalman filter with a minimum jerk model to smooth and bound the head movements measured by HPD, and finally, I displayed the inferred emotions and filtered head movements on the robot. Since the Nao robot has a static face, I used the LEDs around its eyes to display emotions.

### 5.1.1 Results

I evaluated the performance of the system and its components both quantitatively and qualitatively by conducting a number of computational experiments and a user study set up in a real-life setting.

- The results from Emotion Classifier evaluation show that the AUD relies on a reference neutral face in order to work correctly. Classification does not rely on calibration video if the analysed recording starts with a neutral face. When comparing AVG and CONCAT strategies, I observed that it was better to use the low-dimensional

feature vectors generated by averaging rather than by concatenating AU information from the last $W$ frames. The testing accuracy (for Group 1 with calibration video) was 73.4%, well above the chance level (25%).

- From the evaluation of the Head Pose Filter, I can conclude that the mean absolute error over different angle ranges of head movements is relatively small, namely, $4.1° \pm 3.8°$ and $2.5° \pm 2.5°$ for yaw and pitch rotations respectively. However, it increases as the angle from neutral position increases.
- User evaluation obtained via web survey shows that the developed teleoperation system can communicate emotions and head movements very well with a high inter-rater agreement, namely, the intraclass correlation coefficients were $ICC_E = 0.91$ and $ICC_{HP} = 0.72$ for replication of emotions and head movements respectively. The ratings provided for the disgust emotion suggest that this emotion is the most difficult to replicate on the fly.

I was able to keep pace with the proposed time schedule and moreover I implemented one proposed extension – measurement of the replication latency. The processing latency measurements show that the emotion replication is more computationally intensive than the head pose replication as it was expected, namely, $\tau_E = 49$ ms and $\tau_{HP} = 20$ ms respectively. When compared to the head pose replication system proposed by [15], I can conclude that on average I achieved a lower processing latency (by approximately 70%) for head movements replication.

## 5.2 Outputs from the work

I really enjoyed this project as I have learned a lot from multiple areas of Computer Science, mainly, Machine Learning, Digital Signal Processing, and Robotics. It also gave me useful experience with researching new topics, planning the work, and conducting user studies.

With the benefit of hindsight, if starting again, I would spent more time with emotion classification in order to experiment with other ways to generate training examples (Section 3.4.3) and thus achieve a better classification performance. Also, if I had more time, training the Emotion Classifier on SEMAINE database could help to achieve better results.

Based on this project and with the help of my supervisors I[1] have submitted a paper titled *Automatic Replication of Teleoperator Head Movements and Facial Expressions on a Humanoid Robot* to the 26$^{\text{th}}$ IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2017. The proposed paper is currently under review. I am very grateful for this experience.

[1]Co-authors: Dr Oya Celiktutan, Evangelos Sariyanidi and Dr Hatice Gunes.

## 5.3 Further Work

Throughout the project I got several ideas that could be expanded on. I believe that the developed teleoperation system can be further improved by the following extensions:

- Use the Recurrent Neural Networks for emotion classification and compare the performance with the simple one hidden layer feed-forward neural network.
- Train the Emotion Classifier on a more challenging database (e.g. SMIC database [33]) that contains more spontaneous emotions and then test under more challenging conditions – e.g. teleoperator is talking.
- Experiment with more classes of emotions – aim for the full set of 8 basic emotions given in CK+ database.
- Investigate the correlations between yaw and pitch head rotations, and use them to design a better Head Pose Filter.
- Incorporate variable (head pose dependent) state constraints into the Head Pose Filter.
- Extend the system to a robotic platform capable of human-like face expressions (e.g. a robot with a silicon-made actuated skin). For example, the androids developed by Hanson Robotics [34] look suitable.

Additionally, I find the following extension as the **most interesting** one:

- Replicate speech as well and investigate temporal correlations between detected action units and audio signals. Consequently, these correlations and audio cues could be used for better emotion classification.

  More specifically, I would use MIR toolbox [35] to extract the features from audio signals. Then, an unsupervised technique would be applied to discover the important dependencies between these two sets of features (detected action units and audio). For feature extraction I would use an audio-visual dataset such as the Belfast Naturalistic Database (BND) [36]. Finally, the discovered correlations would be used in a real-time emotion classification. The new model could then be compared with the non-audio based classifier developed in this Part II Project to see the effect of exploiting audio signals during teleoperation. As a further extension it would be interesting to look at correlations between action units and recognised speech (not only raw audio signals) which would involve incorporating a speech recognition module. This approach could then be compared with the originally proposed one.

# Bibliography


[1] Owen Daly-Jones, Andrew Monk, and Leon Watts. Some advantages of video conferencing over high-quality audio conferencing: fluency and awareness of attentional focus. *Int. Journal of Human-Computer Studies*, 49(1):21–58, 1998.

[2] Brid O'Conaill, Steve Whittaker, and Sylvia Wilbur. Conversations Over Video Conferences: An Evaluation of the Spoken Aspects of Video-Mediated Communication. *Human-Computer Interaction*, 8(4):389–428, December 1993.

[3] F. Tanaka, T. Takahashi, S. Matsuzoe, N. Tazawa, and M. Morita. Telepresence robot helps children in communicating with teachers who speak a different language. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI '14, pages 399–406, 2014.

[4] Y.-S. Chen, J.-M. Lu, and Y.-L. Hsu. Design and evaluation of a telepresence robot for interpersonal communication with older adults. In *Proceedings of the 11th International Conference on Smart Homes and Health Telematics, ICOST 2013, Singapore*, pages 298–303. Springer, 2013.

[5] E. Sariyanidi, H. Gunes, and A. Cavallaro. Automatic analysis of facial affect: A survey of registration, representation, and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(6):1113–1133, June 2015.

[6] David Kent, Carl Saldanha, and Sonia Chernova. A comparison of remote robot teleoperation interfaces for general object manipulation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, pages 371–379, New York, NY, USA, 2017. ACM.

[7] Kaiko Kuwamura, Takashi Minato, Shuichi Nishio, and Hiroshi Ishiguro. Personality distortion in communication through teleoperated robots. In *Proc of IEEE Int. Symp. on Robot and Human Interactive Communication*, pages 49–54. IEEE, September 2012.

[8] Paul Bremner, Oya Çeliktutan, and Hatice Gunes. Personality perception of robot avatar tele-operators. In *The Eleventh ACM/IEEE International Conference on Human Robot Interation, HRI 2016, Christchurch, New Zealand, March 7-10, 2016*, pages 141–148, 2016.

[9] Oya Celiktutan, Paul Bremner, and Hatice Gunes. Personality classification from robot-mediated communication cues. In *25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016.

[10] Pauline Chevalier, Jean-Claude Martin, Brice Isableu, and Adriana Tapus. Impact of personality on the recognition of emotion expressed via human, virtual, and robotic embodiments. In *Robot and Human Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on*, pages 229–234. IEEE, 2015.

[11] David O. Johnson, Raymond H. Cuijpers, and David van der Pol. Imitating human emotions with artificial facial expressions. *International Journal of Social Robotics*, 5(4):503–513, 2013.

[12] Sichao Song and Seiji Yamada. Expressing emotions through color, sound, and vibration with an appearance-constrained social robot. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 2–11. ACM, 2017.

[13] Paul Bremner and Ute Leonards. Efficiency of speech and iconic gesture integration for robotic and human communicators-a direct comparison. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1999–2006. IEEE, 2015.

[14] Kyriacos Shiarlis, Joao Messias, Maarten Someren, Shimon Whiteson, Jaebok Kim, Jered Vroon, Gwenn Englebienne, Khiet Truong, Vanessa Evers, Noé Pérez-Higueras, et al. Teresa: A socially intelligent semi-autonomous telepresence system. 2015.

[15] Priyanshu Agarwal, Samer Al Moubayed, Alexander Alspach, Joohyung Kim, Elizabeth J Carter, Jill Fain Lehman, and Katsu Yamane. Imitating human movement with teleoperated robotic head.

[16] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[17] P. Ekman and W. V. Friesen. Facial action coding system: A technique for the measurement of facial movement. 1978.

[18] M. Pantic. Automatic analysis of facial expressions. *Encyclopedia of Biometrics*, pages 128–134, 2015.

[19] Paul Ekman and Wallace V Friesen. The repertoire of nonverbal behavior: Categories, origins, usage, and coding. *Semiotica*, 1(1):49–98, 1969.

[20] Aldebaran robotics web site. `http://www.aldebaran-robotics.com/`, 2013.

[21] Jan Ondras, Oya Celiktutan, Evangelos Sariyanidi, and Hatice Gunes. Automatic replication of teleoperator head movements and facial expressions on a humanoid robot (under review). In *26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2017.

[22] Evangelos Sariyanidi, Hatice Gunes, Muhittin Gökmen, and Andrea Cavallaro. Local Zernike moment representations for facial affect recognition. In *Proc. British Machine Vision Conf.*, 2013.

[23] Tamar Flash and Neville Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7):1688–1703, 1985.

[24] Reza Shadmehr and Steven P Wise. *The computational neurobiology of reaching and pointing: a foundation for motor learning*. MIT press, 2005.

[25] Dan Simon. Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *IET Control Theory & Applications*, 4(8):1303–1318, 2010.

[26] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 94–101. IEEE, 2010.

[27] Nicolas Gourier, Daniela Hall, and James L Crowley. Estimating face orientation from robust detection of salient facial structures. In *FG Net Workshop on Visual Observation of Deictic Gestures*, volume 6, 2004.

[28] Mikel Ariz, José J Bengoechea, Arantxa Villanueva, and Rafael Cabeza. A novel 2d/3d database with automatic face annotation for head tracking and pose estimation. *Computer Vision and Image Understanding*, 148:201–210, 2016.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[30] Martin S Andersen, Joachim Dahl, and Lieven Vandenberghe. Cvxopt: A python package for convex optimization, version 1.1. 6. *Available at cvxopt. org*, 2013.

[31] Patrick E Shrout and Joseph L Fleiss. Intraclass correlations: uses in assessing rater reliability. *Psychological bulletin*, 86(2):420, 1979.

[32] Domenic V Cicchetti. Guidelines, criteria, and rules of thumb for evaluating normed and standardized assessment instruments in psychology. *Psychological assessment*, 6(4):284, 1994.

[33] Xiaobai Li, Tomas Pfister, Xiaohua Huang, Guoying Zhao, and Matti Pietikäinen. A spontaneous micro-expression database: Inducement, collection and baseline. In *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, pages 1–6. IEEE, 2013.

[34] Hanson Robotics website. `http://www.hansonrobotics.com/robot-gallery/`. Accessed: 2017-04-06.

[35] Olivier Lartillot and Petri Toiviainen. A matlab toolbox for musical feature extraction from audio. In *International Conference on Digital Audio Effects*, pages 237–244, 2007.

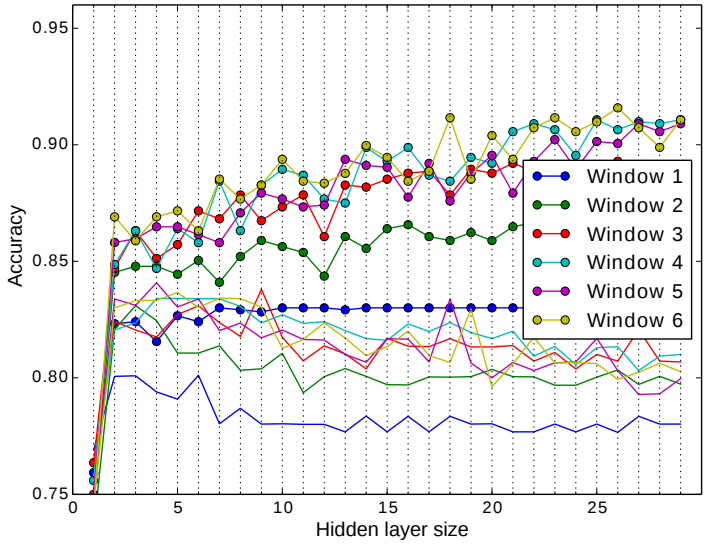
[36] Ellen Douglas-Cowie, Nick Campbell, Roddy Cowie, and Peter Roach. Emotional speech: Towards a new generation of databases. *Speech communication*, 40(1):33–60, 2003.

[37] David O Johnson, Raymond H Cuijpers, and David van der Pol. Imitating human emotions with artificial facial expressions. *International Journal of Social Robotics*, 5(4):503–513, 2013.

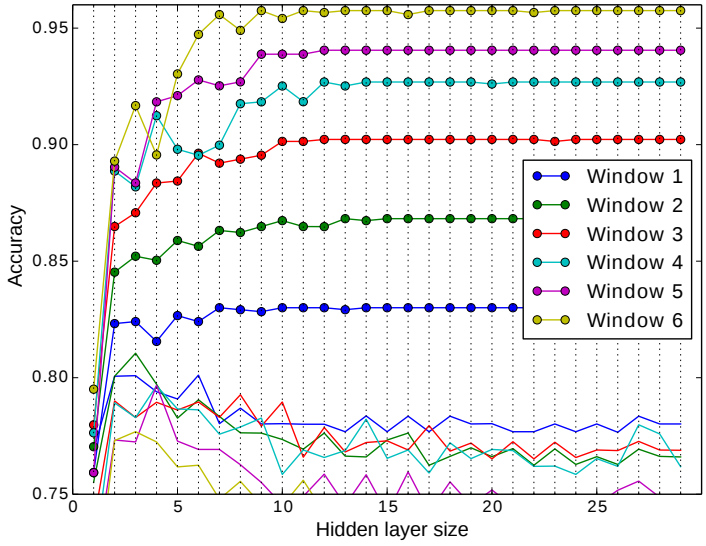
[38] Priyanshu Agarwal, Samer Al Moubayed, Alexander Alspach, Joohyung Kim, Elizabeth J Carter, Jill Fain Lehman, and Katsu Yamane. Imitating human movement with teleoperated robotic head.

# Appendix A

# Cross-validation figures

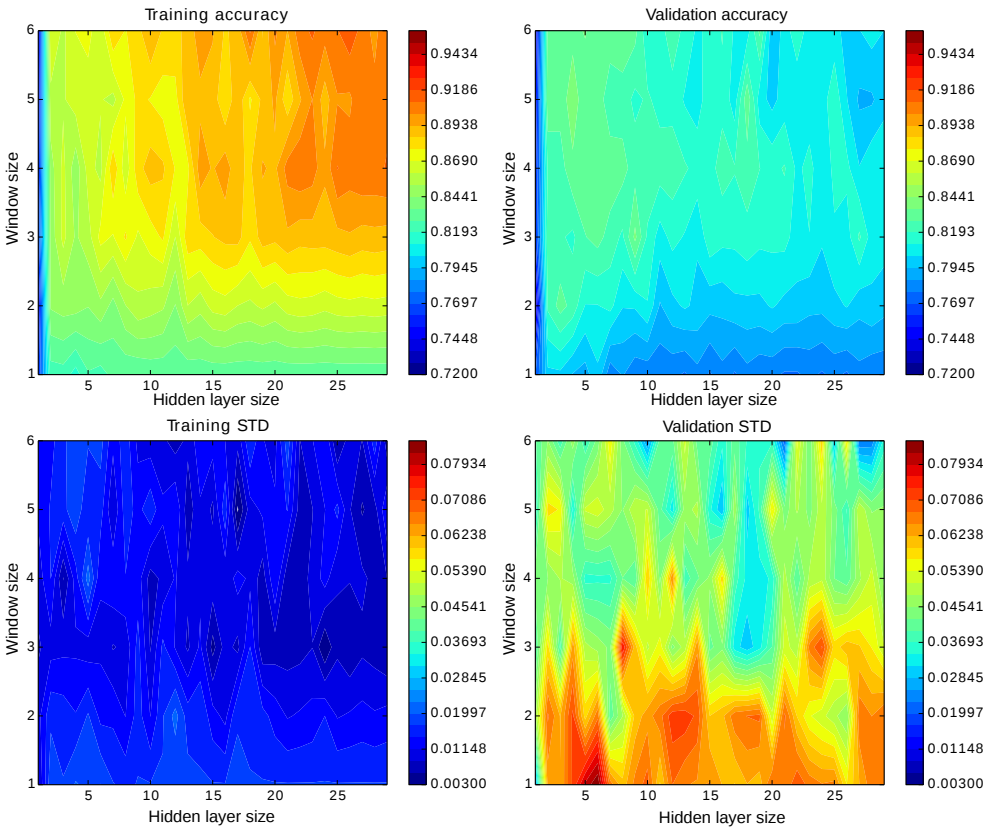



(a) Strategy AVG

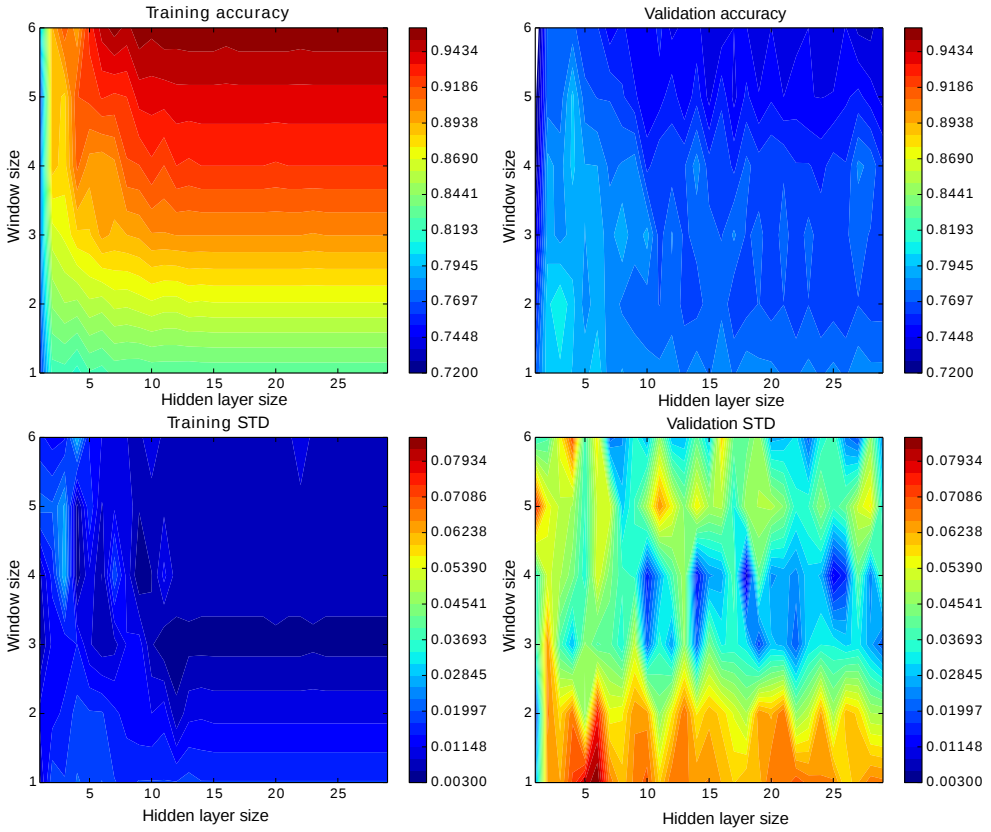



(b) Strategy CONCAT

Figure A.1: First phase of cross-validation on CK+ database: mean training accuracy (with circles) and validation accuracy (without circles) for each combination of hyper-parameters – hidden layer size *hls* and window size $W$. Separately for AVG (a) and CONCAT (b) strategy.

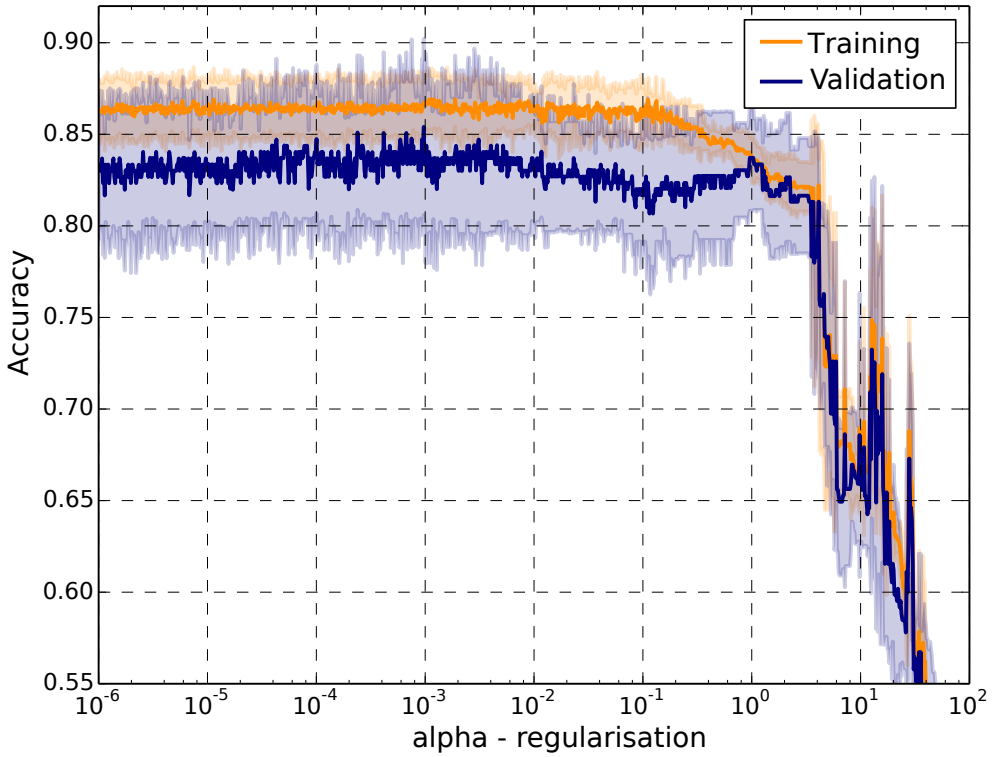



(a) Strategy AVG

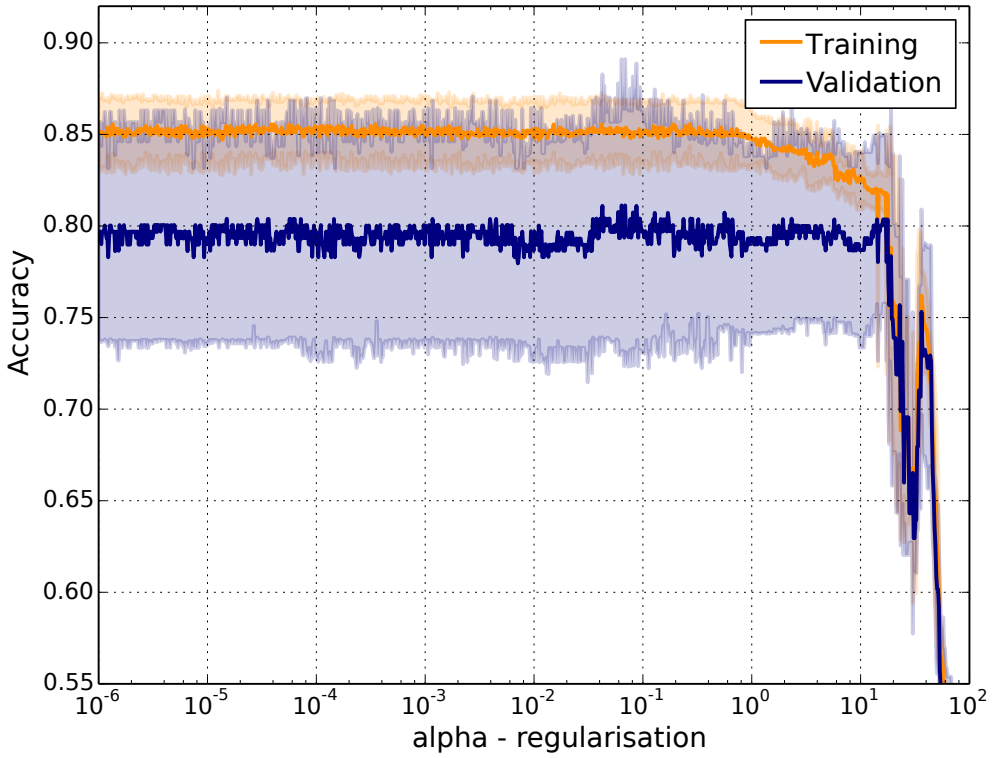



(b) Strategy CONCAT

Figure A.2: First phase of cross-validation on CK+ database: mean training (top left) and validation (top right) accuracies along with standard deviations STD (bottom) for each combination of hyperparameters – hidden layer size $hls$ and window size $W$. Separately for AVG (a) and CONCAT (b) strategy.

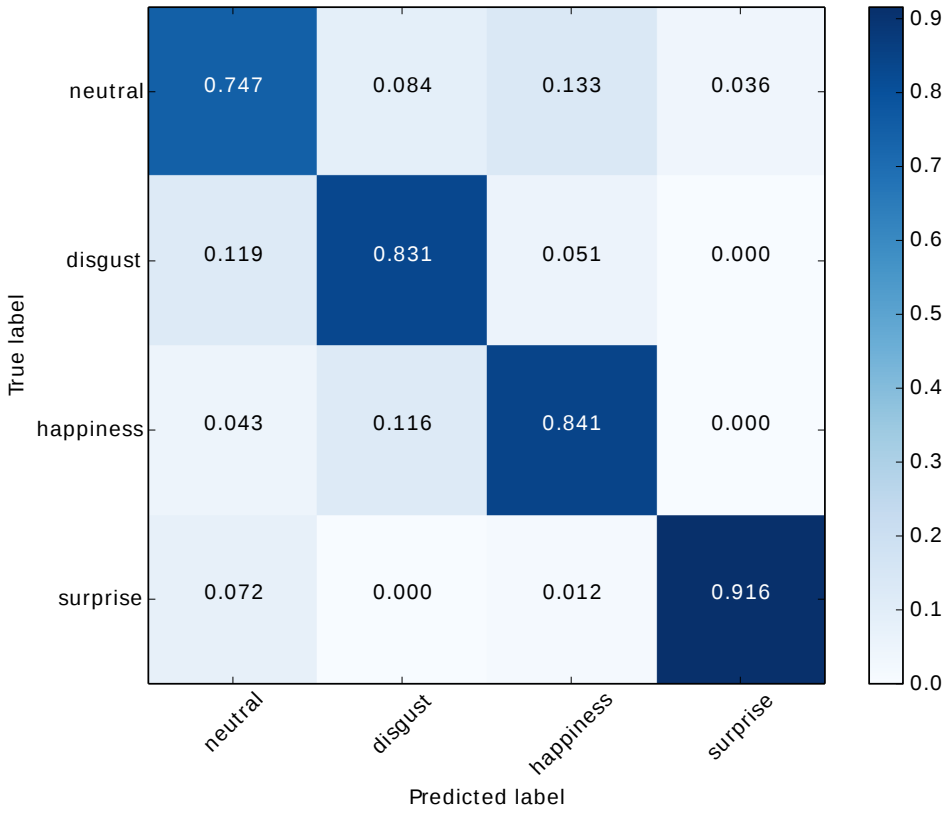



(a) Strategy AVG

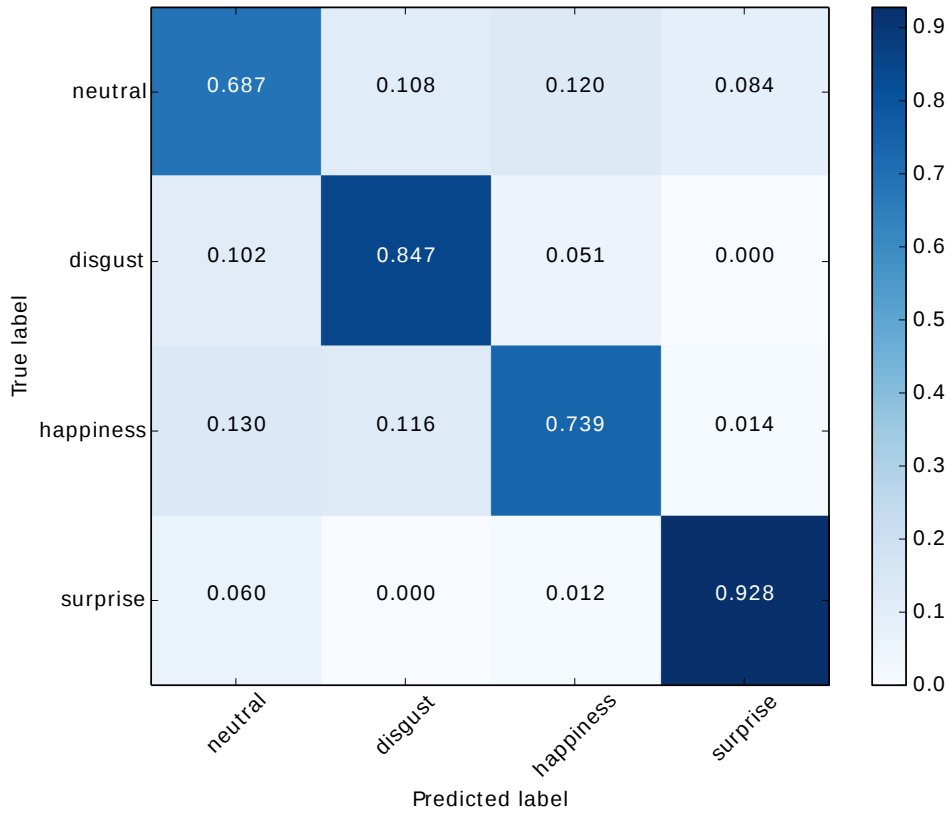



(b) Strategy CONCAT

Figure A.3: Second phase of cross-validation on CK+ database: mean training (orange) and validation (blue) accuracies along with standard deviations for different regularization parameter $\alpha$. Separately for AVG (a) and CONCAT (b) strategy. The accuracies are not much affected by regularization parameter $\alpha$ within the range $10^{-6}$–$10^{-1}$.




(a) Strategy AVG




(b) Strategy CONCAT

Figure A.4: Normalised confusion matrices after cross-validation on CK+ database: summed over 5 validation folds after the optimal hyperparameters were found. Separately for AVG (a) and CONCAT (b) strategy.

# Appendix B

# Ethics Committee approval

**Ethics Review #420**
**TITLE:** Replicating Human Facial Emotions and Head Movements on a Robot Avatar
**APPLICANTS:** Hatice Gunes, Oya Celiktutan Dikici, Jan Ondras
**EMAIL:** Hatice.Gunes@cl.cam.ac.uk, Oya.Celiktutan.Dikici@cl.cam.ac.uk, jo356@cam.ac.uk
**DATES:** 13/02/2017–27/03/2017
**STUDY TYPE:** controlled experiment
**FUNDING BODY:** –

**DESCRIPTION**
The proposed study will take place either in the Computer Laboratory or in the Trinity College. The proposed study consists of two stages, namely, a controlled experiment and an online survey. For the controlled experiment, a total of 30 target-participants will be recruited. Each participant will be asked to display four emotional expressions including neutral, disgust, happiness and surprise, and display four head poses (looking up/down and left/right), and will be recorded using a web camera while performing these gestures for a total duration of 3 minutes. Then the recorded videos will be processed and replicated on the Nao robot that is a commercially available child-like looking humanoid robot, very widely used in human-robot interaction studies, and the robot will also be recorded when replicating the participants' gestures. For the online survey, external observer-participants will be recruited to view the recorded videos of human and robot side by side, and assess the quality of replication using a 5 point Likert scale (i.e. from 1 to 5). Each video pair will be assessed by 10 observer participants. This will result in a total of approximately 1800 annotations for all video clips.

**PRECAUTIONS**
Consent:
Prior to the experiment, target-participants will be given an information sheet that explains the goal of the study and recording procedure, and will be asked to sign a consent form for their participation. The consent form for target-participants will include an explicit description of the fact that recordings of them will be shown to observer-participants via an online survey. Prior to taking the online survey,

observer-participants will be given a description that explains the goal of the study, and will be asked to declare a tick-box with the text of "I understand that I will be viewing video recordings of private individuals. I promise that I will not make copies or screen grabs or otherwise reuse these private recordings." The consent form for target-participants will reproduce the privacy undertaking for observer participants, so that target-participants understand the conditions under which the recordings will be used.

Participant Recruitment:

All the participants will be recruited from adults including undergraduate, graduate students, postdoctoral researchers and other staff of the University of Cambridge. All participation will be on voluntary basis and people can withdraw from the study whenever they want.

# Appendix C

# Instructions for human participants

University of Cambridge

Computer Laboratory

Experiment ID: X1

Researchers: J. Ondras, O. Celiktutan, H. Gunes

## Instructions to Participants

Please ensure that you have read the instructions and completed the consent form.

As a final evaluation of my Part II Project on **Replicating Human Facial Emotions and Head Movements on a Robot Avatar** I would like to carry out a controlled experiment and consequently an anonymous web-based survey to assess the performance of my replication system.

In this experiment you will be asked to display four emotional expressions including neutral, disgust, happiness and surprise, and display four head poses (looking up/down and left/right), and will be recorded using a web camera while performing these gestures for a maximum duration of 3 minutes.

Then the recorded videos will be processed and replicated on the Nao robot that is a commercially available child-like looking humanoid robot, very widely used in human-robot interaction studies, and the robot will also be recorded when replicating the participants' gestures.

Consequently, the videos of you and the robot replicating you will be viewed by other study participants in a web survey to evaluate how good the robot is at replicating the emotions and head movements.

The whole experiment should not take longer than 15 minutes.

# Appendix D

# Human-robot videos

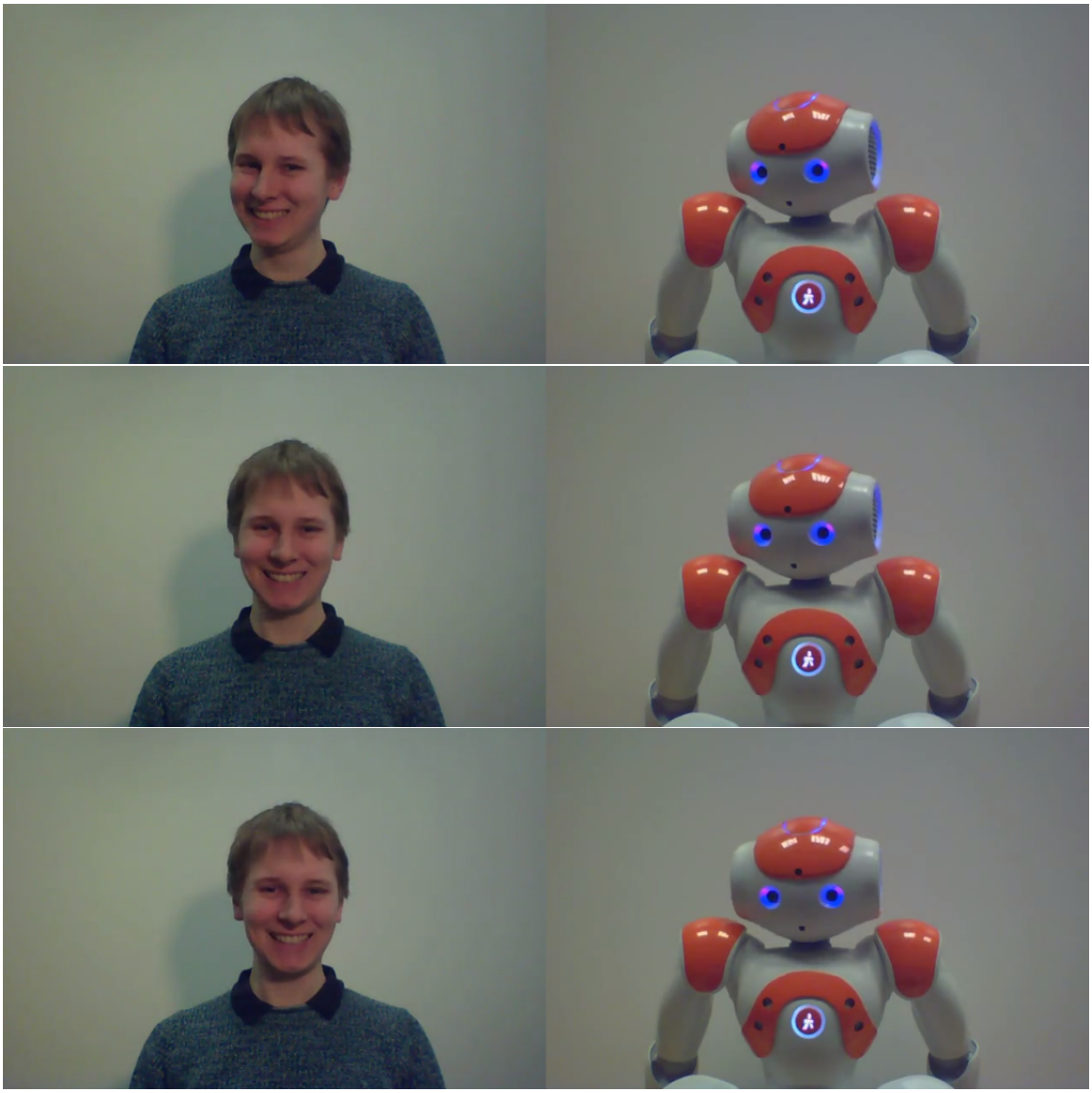


Figure D.1: Image sequence from a *human-robot video* (part one). See Section 4.4 for more details.

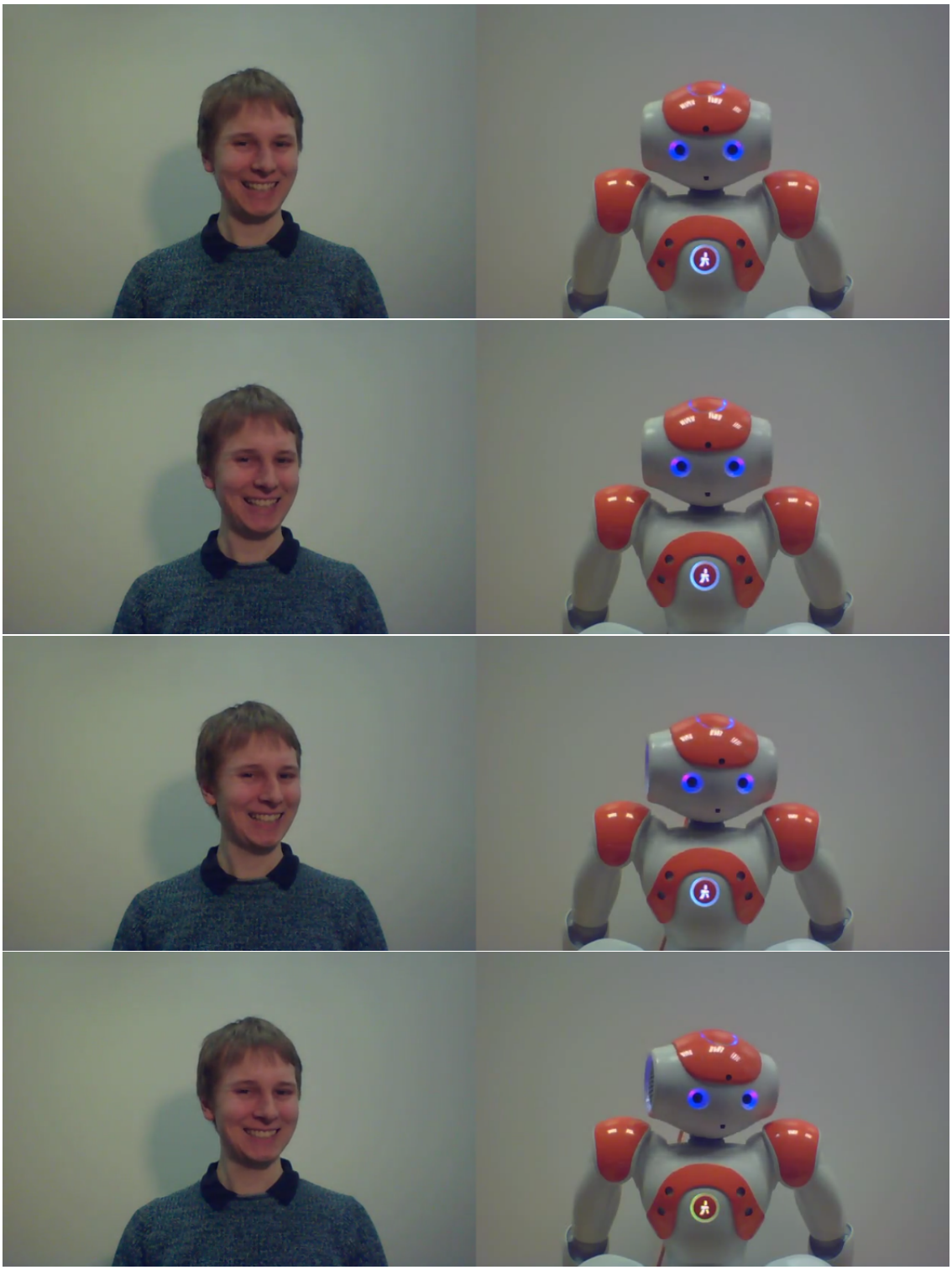


Figure D.2: Image sequence from a *human-robot video* (part two). See Section 4.4 for more details.

# Appendix E

# Web survey example

NEUTRAL | DISGUST | SURPRISE | HAPPINESS

Please, rate the EMOTION replication *

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Very poor | ○ | ○ | ○ | ◉ | ○ | Very good |

Please, rate the HEAD POSE replication *

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Very poor | ○ | ◉ | ○ | ○ | ○ | Very good |

Figure E.1: Example of a web survey question. See Section 4.4.

# Appendix F

# Project Proposal

Computer Science Tripos – Part II – Project Proposal

## Replicating Human Facial Emotions and Head Movements on a Robot Avatar

Jan Ondras, Trinity College

Originator: Dr Oya Celiktutan

21 October 2016

**Project Supervisors:** Dr Hatice Gunes & Dr Oya Celiktutan

**Directors of Studies:** Dr Sean Holden & Dr Frank Stajano

**Project Overseers:** Dr David Greaves & Prof John Daugman

## Introduction

Robotic telepresence offers a convenient substitute for face-to-face communication as it provides psychical embodiment at a remote place (from tele-operator's point of view) and allows the tele-operator to express non-verbal cues such as head movements, hand gestures, facial expressions along with audio cues to another person (user) via robot. An example of such a system is shown in Fig. F.1.

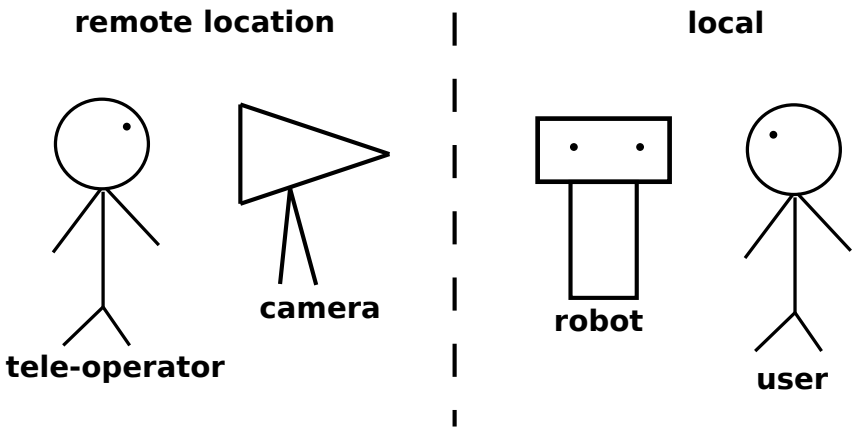



Figure F.1: Telepresence example.

The goal of this project is to automatically recognise emotions and estimate head movements from video sequences of the tele-operator, and develop a system to imitate the recognised emotions and the estimated head movements on a robot avatar in a real-time manner. (The terms emotion and facial expression of an emotion will be further used interchangeably.)

According to Ekman and Friesen [19] people express 6 basic emotions: Anger, Surprise, Disgust, Sadness, Happiness, Fear. These emotions are known to be expressed by many facial muscle movements that correspond to active Action Units (AUs) in recognition systems. E.g. AU2 is the outer brow raiser. Table F.1 shows the mapping to emotions:

Table F.1: Emotions vs AUs

| Emotion | Active AUs |
|---|---|
| Happiness | 6, 12 |
| Sadness | 1, 4, 15 |
| Surprise | 1, 2, 5B, 26 |
| Fear | 1, 2, 4, 5, 7, 20, 26 |
| Anger | 4, 5, 7, 23 |
| Disgust | 9, 15, 16 |

The head pose can be described by three angles (pitch, yaw, roll) that correspond to rotations around x, y, z axes.

## Starting point

Many studies address the recognition and replication of human facial emotions using computer vision techniques, others concentrate on head movements. This project aims to combine these two into one novel system using standard techniques of supervised machine learning and filtering.

I am almost completely new to Digital Signal Processing and Computer Vision, and the courses on these are lectured in Michaelmas and Lent term this year respectively. My knowledge of AI techniques is limited to what was taught in Part 1B Artificial Intelligence and I have some basic experience with robotics from my high school.

The project will be mostly implemented in C++ (since the speed is crucial in a real-time system) building on an existing *Action Units and Head Pose Detector* (later referred to as *AUHPD*). So I will need to refresh and broaden my knowledge from Part 1B C & C++ course. However, for training I consider using Python with SciPy.

I will use GitHub for version control and LaTeX for typesetting my dissertation. I have a very basic experience with these tools.

## Resources required

For this project I will use my own laptop (quad-core 2.2GHz Intel i7, 8GB RAM, 1TB HDD, Windows 10 & Ubuntu Linux). I will backup all my work to GitHub every day and to an external 500GB HDD once per week. If my laptop suddenly fails I will continue my work from a backup on a MCS machine until it is repaired. I accept full responsibility for my machine and I have made contingency plans to protect myself against hardware and/or software failure.

In the later stages I will use Aldebaran's Nao robot in the Computer Laboratory. The access to it will be provided by my project Supervisors and they will be responsible for me whilst I am on the premises. Alternatively, I can test my implementation remotely on a simulated robot using Choregraphe application. In case the robot fails I will continue my work on a similar platform - Pepper robot. For video recordings needed in the Final evaluation stage I will use my laptop webcamera and/or a camera provided by my project Supervisors.

Regarding the software, I will use *AUHPD* provided by my project Supervisors and an open-source software such as Robot Operating System (ROS), OpenCV library, and Qt IDE for C++ development. In order to train *Emotion Classifier* and to develop filtering methods, I will use facial expression databases (CK+ and SEMAINE) and head pose databases (IDIAP and GI4E). These are either publicly available or will be provided by my project Supervisors.

## Work to be done

Firstly, the facial AUs and head pose estimates will be obtained using *AUHPD* given the data from relevant databases. However, this *AUHPD* provides facial AUs and head pose frame-by-frame. In order to deal with the noise further processing is needed and will be different for facial expressions and for head movements. This breaks the project into two

independent tasks (Fig. F.2) that will join in the final stage - displaying the emotions and head poses on the robot.

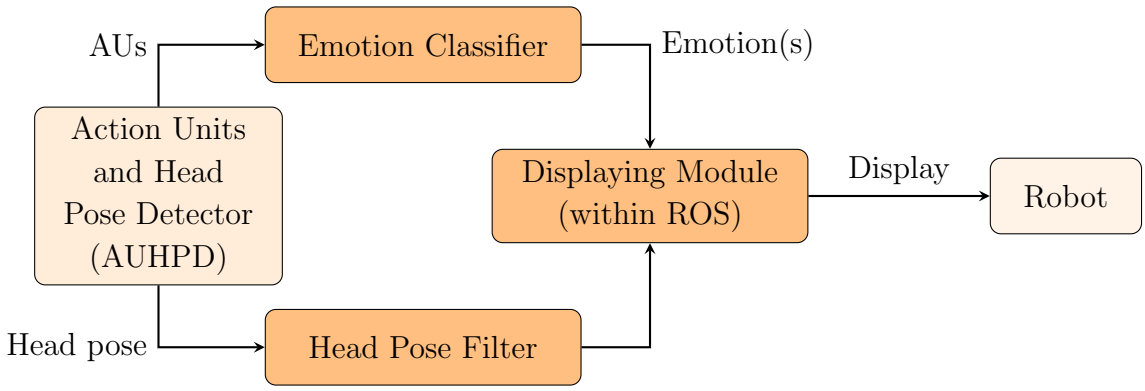



Figure F.2: Project workflow. *Dark orange:* components to be implemented. *Bright orange:* existing components to be used.

The modules to be implemented:

1. **Emotion Classifier:** The mapping from combinations of active facial AUs to corresponding emotions will be learnt from annotated data from databases using neural networks. This is to cope with the noise compared to the hard-coded assignments of emotions to a set of active AUs (as seen in Table 1). Training will be done on two different kinds of data: CK+ database (clean data: posed facial expressions) and SEMAINE database (more noisy data: spontaneous natural facial expressions).
2. **Head Pose Filter:** Raw head pose estimates from *AUHPD* typically contain significant noise which would result in jerky robot movements. Therefore, we need to ensure that the head pose trajectory is smooth and that it has angular position, velocity and acceleration appropriate for displaying on the robot given the robot's movement limitations. This will be addressed by a minimum jerk model of the trajectory and a constrained-state Kalman filter.
3. **Displaying Module:** This involves implementation in ROS: Actually, Nao robot does not have a human like face (nor an actuated skin) but it has many LEDs on its face. Therefore, the recognised emotions will be displayed in terms of LEDs' pattern, colour and brightness according to the mapping discovered by Johnson [37]. Filtered head movements will be directly mapped to the robot.

## Evaluation

I plan to do the evaluation in three phases:

1. **Emotion Classifier evaluation:** I will use standard techniques to evaluate the recognition accuracy of neural network (K-fold and/or one subject out cross-validation).

2. **Head Pose Filter evaluation:** This will be done on datasets (IDIAP, GI4E) of head pose videos continuously labelled in time. The evaluation metric will be the mean (squared or absolute) error between the prediction of the head pose after filtering stage and ground truth label from database.
3. **Final evaluation:** This will involve a controlled experiment - recruiting (e.g. 10) people to come to the Computer Laboratory and they will be asked to make 6 basic emotions while recording the robot replicating them. In this manner 60 annotated videos will be obtained. Consequently, in an anonymous web-based survey respondents will have to assign one of 6 emotions to every video. Then the fraction of respondents that selected the correct emotion will correspond to replication accuracy of the developed system.

   My Supervisors are both very experienced with ethical obligations and requirements regarding the studies involving human participants and they will help me with these.

# Success criteria

The project will be considered successful if the following criteria are met:

1. Given a visual recording of a human head: the *Emotion Classifier* will recognise correct emotion(s),
2. the *Head Pose Filter* will smooth the relevant head movements,
3. the robot will display the recognised emotions and head movements concurrently in a real-time.

# Possible extensions

If I achieve my main result early I shall try the following extensions:

- Measure and evaluate the replication latency of the system.
- Implement the system on another robotic platform (Pepper robot or robotic head with a silicon-made actuated skin).
- Train the *Emotion Classifier* on a more challenging database that contains more subtle muscle movements, e.g. SMIC database.
- Improve *AUHPD*: so far it combines two sources (shape and appearance) into one which is then used for training the *Emotion Classifier*. The possible improvement might be to deal with these two sources separately using two neural networks.
- Use neural networks also for the head pose replication.
- Do the Final evaluation part in uncontrolled conditions (e.g. teleoperator is talking).

# Timetable

1. **Michaelmas weeks 2–4 ($20^{th}$ Oct – $2^{th}$ Nov)**
   - **Deadline:** Project Proposal submission on $21^{th}$ Oct (12 noon)
   - Finish and submit Project Proposal.
   - Study research papers and previous work in this field.
   - Set up version control system and backups.
   - Collect and prepare databases (write appropriate scripts).
   - Familiarize with *AUHPD*, OpenCV and Qt IDE.
2. **Michaelmas weeks 5–6 ($3^{th}$ Nov – $16^{th}$ Nov)**
   - Background reading on neural networks.
   - Decide on design, and training and validation techniques.
   - Implement *Emotion Classifier*.
   - Train *Emotion Classifier*.
3. **Michaelmas weeks 7–9 ($17^{th}$ Nov – $7^{th}$ Dec)**
   - Evaluation of *Emotion Classifier*.
   - Learn about minimum jerk model and constrained-state Kalman filtering.
   - Read paper [38] about similar application.
   - Prototype the *Head Pose Filter* before going on vacation.
4. **Michaelmas vacation ($8^{th}$ Dec – $11^{th}$ Jan)**
   - Finalize, test and evaluate the *Head Pose Filter*.
   - Catch up time for any complications.
5. **Lent weeks 0–2 ($12^{th}$ Jan – $1^{th}$ Feb)**
   - Set up and familiarize with ROS.
   - Compose the whole system implementing the *Displaying Module* in ROS.
   - Write Progress Report and prepare a presentation.
6. **Lent weeks 3–5 ($2^{th}$ Feb – $22^{th}$ Feb)**
   - **Deadline:** Progress Report submission on $3^{th}$ Feb (12 noon)
   - **Deadline:** Progress Report Presentation on $9^{th}$ – $14^{th}$ Feb (2pm)
   - Rehearse the presentation.

- Start Final evaluation (controlled experiment and survey).
- Catch up time for any complications.

7. **Lent weeks 6–8 ($23^{th}$ Feb – $15^{th}$ Mar)**
   - Start drafting the dissertation.
   - Catch up time for any complications.
8. **Easter vacation: ($16^{th}$ Mar – $19^{th}$ Apr)**
   - Finish Final evaluation.
   - Write dissertation main chapters.
   - Possible extensions.
9. **Easter weeks 0–2: ($20^{th}$ Apr – $10^{th}$ May)**
   - Complete the dissertation.
   - Ask for feedback from project Supervisors and Directors of Studies.
   - Make changes based on feedback.
10. **Easter week 3: ($11^{th}$ May – $17^{th}$ May)**
    - **Deadline:** Dissertation submission on $19^{th}$ May (12 noon)
    - **Deadline:** Source code submission on $19^{th}$ May (5pm)
    - Proof reading and final changes.
    - Early submission to leave time for revision.