Multi-Modal Technology for User Interface Analysis including Mental State Detection and Eye Tracking Analysis

Ahmed Husseini Orabi

A Thesis submitted

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Under the auspices of the Ottawa-Carleton Institute for Computer Science

University of Ottawa

Ottawa, Ontario, Canada

Abstract

We present a set of easy-to-use methods and tools to analyze human attention, behaviour, and physiological responses. A potential application of our work is evaluating user interfaces being used in a natural manner. Our approach is designed to be scalable and to work remotely on regular personal computers using expensive and noninvasive equipment.

The data sources our tool processes are nonintrusive, and captured from video; i.e. eye tracking, and facial expressions. For video data retrieval, we use a basic webcam. We investigate combinations of observation modalities to detect and extract affective and mental states.

Our tool provides a pipeline-based approach that 1) collects observational, data 2) incorporates and synchronizes the signal modality mentioned above, 3) detects users' affective and mental state, 4) records user interaction with applications and pinpoints the parts of the screen users are looking at, 5) analyzes and visualizes results.

We describe the design, implementation, and validation of a novel multimodal signal fusion engine, Deep Temporal Credence Network (DTCN). The engine uses Deep Neural Networks to provide 1) a generative and probabilistic inference model, and 2) to handle multimodal data such that its performance does not degrade due to the absence of some modalities. We report on the recognition accuracy of basic emotions for each modality. Then, we evaluate our engine in terms of effectiveness of recognizing basic six emotions and six mental states, which are agreeing, concentrating, disagreeing, interested, thinking, and unsure.

Our principal contributions include the implementation of a 1) multimodal signal fusion engine, 2) real time recognition of affective and primary mental states from nonintrusive and inexpensive modality, 3) novel mental state-based visualization techniques, 3D heatmaps, 3D scanpaths, and widget heatmaps that find parts of the user interface where users are perhaps unsure, annoyed, frustrated, or satisfied.

Acknowledgments

In the first place, many thanks for my supervisor Dr. Timothy C. Lethbridge, who has always tried his best to help me during my PhD. I am extremely thankful for his contributions, efforts, suggestions, and funding during my study. His insightful supervision has always kept me motivated, and helped me improve in the fields related to my work. I am glad to know Tim as a supervisor – my PhD has hopefully just been the start of a long friendship.

I also have to thank the members of my PhD committee for their helpful advice and suggestions.

I wish to deeply thank my parents for their constant support, and unconditional love and care. You have made it clear that the moments of my success have been among the most delightful parts of your life. I will always try my best to keep you always proud of me. My heartfelt thanks to Mahmoud, my brother, who has always been there during my work and study.

My thanks for the University of Ottawa, The Ontario Graduate Scholarship (OGS), ORF, and NSERC for collaborating with and funding my research.

I would like to thank my close friends who have always kept me cheerful during my study. The moments that I have spent with you have made my life much brighter.

Finally, a sincere thank-you to all researchers, around the world, in different fields, such as computer vision, machine learning, and software engineering. Your continuous collaboration in the fields of computing have always inspired me, and provided the necessary substance for my research.

Table of Contents

List of Figures

List of Tables

List of Snippets

Glossary

Active Appearance Model (AAM): An extended approach to ASM (see below), that relies on additional knowledge such as texture and shape information [1]. An AAM model is typically used for facial features alignment and tracking.

Active Blobs Model (ABM): A deformable model (see below) that aims to speed up model alignment processing using active blob, a texture-mapped deformable 2D triangular mesh [2]. ABM is also called an active morphable model.

Active Contour Model (ACM): An early approach to deformable models [3]. It is used to detect objects based on observing shape outline, restrictions, and correlated structures.

Active Shape Model (ASM): A deformable model based on a statistical shape model that improves upon ACM, in order to increase specificity, observation, and detection even for shapes with high variability [4].

Adaptive Probabilistic (AP): A novel fitting optimization technique we introduced to reduce the time complexity and fitting accuracy using adaptive methods that sample the appearance noise to improve accuracy. This is one of our contributions and is described in Chapter 5.

Adaptive Probabilistic Project Out (APPO): An extension to AP; it differs in that it performs an additional step to project out the appearance variations.

Affective computing: The integration of emotions in computing. It focuses on systems or devices that can interpret, recognize, simulate, and process their users' affective states [5].

Affective state: A psycho-physiological state of the user's interaction with a software application from an emotional perspective. It has three main categories: valence, motivational intensity (control), and arousal [5].

Arousal: The activation of the sympathetic nervous system that is stimulated by reactions to events, such as a user's interaction with an application [6]–[8].

Deep Temporal Credence Network (DTCN): A novel multimodal temporal neural network that combines generative and inference as internally independent networks and uses variational inference to extract the joint representation among modalities. This is one of our contributions, and is described in Chapter 6.

Deformable model: A model in computer vision that handles structural variability and matching [9] by 1) using prior knowledge about shapes such as location and size, and 2) examining images at a low-level in order to find the local and global structures of their regions. A region's structure has various shapes and textures used to find the plausible points of interest or objects

Delaunay triangulation: A triangular representation, used in many computer vision algorithms, that links similar points of objects as a mesh of triangles [10].

Eye fixation: The action of fixating eyes onto an object. It is required to start visual attention, or to shift attention [11].

Eye gaze: The action of looking attentively or steadily at a visual location because of reasons such as cognition, surprise, or admiration [12], [13]. Gaze in physiology is used to describe the coordinated motion of the eyes or simply where the eyes are looking.

Facial Action Coding (FAC): A system used to taxonomize human facial movements based on their appearance [14].

Histogram of Oriented Gradient – Patch Active Appearance Model (HOG-PAAM): A type of Active Appearance Model that aims to reduce the texture space using oriented patches. It uses HOG for appearance features extraction. Joining HOG and PAAM is one of our contributions. HOG and PAAM are described in Chapter 5.

In the wild: Describes an image or video captured in unconstrained conditions (typically, real-world conditions that people encounter in their daily life), as opposed to in the laboratory. For example, the recognition of spontaneous mental states of an individual under real-world conditions [15], [16].

K-d tree: A tree of a k-dimensional data structure used to organize a number of points in a space. It is helpful in algorithms related to finding nearest neighbours. In PCA (see below), a k-d tree is used to speed up the optimization process and the extraction of eigenvectors [17].

Landmarks: A group of relative points or pixel locations used to annotate salient regions of an object [18].

Motivational intensity: The extent of the urge to avoid or act towards a stimulus (activity, action, behaviour, perception, etc). Taking an action, such as moving, is derived from a motivational urge [6]–[8].

Multimodal Convolutional Recurrent Variational Autoencoder (MCRVAE): A generative deep neural network, parameterized with an inference model in a DTCN. This is one of our contributions.

Principal Component Analysis: A statistical technique to find the underlying correlations among the variables, and uncover those that are uncorrelated. In computer vision, it is used to reduce the number of features to be analysed.

Temporal Feature Fusion (TFF): A fusion technique introduced to DTCN to extract the temporal joint representation among modalities for a fixed time. It uses variational information and Bayesian inference to extract joint representation among the modalities. It is one of our contributions, and is described in Chapter 6.

Temporal Output Fusion (TOF): A fusion technique that encodes the output of different classifiers into a heatmap. It fuses their outputs and improves their recognition accuracy. This is one of our contributions, described in Chapter 6.

Valence: A positive-to-negative evaluation of an experienced subjective state (in the context of affective computing) [5]. Valence evaluation is determined by the consequences of the experienced subjective state and the eliciting circumstances.

Chapter 1 Introduction

We present a tool that allows for collecting, synchronizing, and analyzing observational data of users from different data sources, while they are interacting naturally with an application. Our key modalities are eye tracking, facial expression, and affective states.

Our tool incorporates a multimodal signal fusion model to improve real time recognition of users' affective and mental states, such that those states can help evaluate usability. We describe our novel visualization techniques that relate the users' affective and mental state to the parts of the screen users are looking at.

Software usability is an important aspect of software quality. Approaches to design and evaluate usability are supported by a large amount of evidence-based HCI research [19]. The ever-increasing use of computers, and issues of productivity and safety are driving a need for increased rigour in the testing and regulation of software usability [20], [21]. However, usability is still often lacking or under-prioritized because it remains difficult to evaluate complex user interfaces. In particular, there is a need for tools that can more precisely capture representations corresponding to end-user mental models without a need for the users to memorize their experiences or to speak their thoughts, feelings, or opinions during sessions. This memorization or speaking disrupts natural use of the system, and generates data that is laborious to analyze.

Evaluating a system depends on the interactions between that system and its users. Some evaluation activities must be undertaken at an early stage so feedback becomes available during requirement and design phases [19], [22].

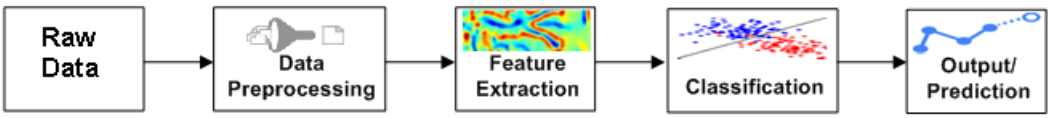
Traditional usability assessments are only in a very limited way able to take into account what the user is thinking. For example, a usability questionnaire focuses on some usability aspects. However, it is not expected to cover all issues that users can face during usability sessions because it is tied with user's exploration of the system [23]–[25]. Additional problems can occur because of miscalculation or human errors. For example, many evaluations are affected by evaluator bias [23].

Affective and mental state detection is not commonly used in traditional usability methods. Such states are used to analyze and infer additional information about users, such as cognitive effort, user performance, and emotion. This information can be retrieved from different modalities such as the brainwaves and heart rate. For example, Freeman et al, found that in electroencephalography (EEG), brain waves at higher levels of beta and lower levels of alpha and theta pattern showed better mental performance [26]. Also, Boucsein indicated that there is a correlation between reduced heart rate and amount of mental effort involved [27].

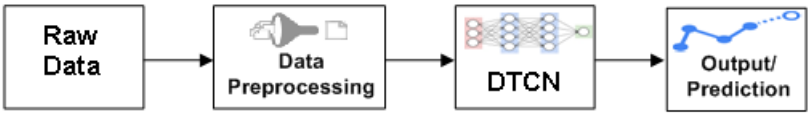
The cognitive effort spent by users, which is a part of affective and mental states, can be measured using techniques such as eye tracking [28]. Yet, such a technique is not particularly well-linked with the affective and mental states of users; this is something we investigate in this thesis.

Using multiple modalities helps collecting more information about affective and mental states [29], [30]. On the other hand, this can lead to an unmanageable amount of data. Especially, common visualization techniques, such as heatmaps and scanpaths, do not link affective and mental states to user interactions.

Deep learning has shown significant performance improvement over earlier techniques in several studies for solving perception problems such as affect recognition [31], [32]. Figure 1-1 shows the key differences between traditional and deep learning models. The features of a deep learning model enable better understanding of visual data, and thereby can efficiently improve the performance of visual recognition [33], [34]. Therefore, we choose to use deep networks as they excel in learning high-level abstract features from large-scale datasets of multiple inputs. Recent research attempts to use deep learning to reduce data dimensionality [35], and to replace hand-crafted features with efficient unsupervised or semi-supervised feature extraction [36]–[38].



(a)



(b)

Figure 1-1. Key differences of traditional (b) and deep learning (a)

In our research, we promote the use of multiple modalities, which means that we need to find a way to handle the issues mentioned above. We use a novel multimodal signal fusion engine, Deep Temporal Credence Network (DTCN) to overcome the issues of multiple modalities in a temporal domain. This will lead researchers to be better able to use recognized affective and mental state of users to help assess user interfaces, and to provide User Emotional Experience (UEX) information to the evaluator [39]. Toward this, we extended traditional visualization techniques, scanpaths and heatmaps to show affective and mental state in a third dimension, such that concave and convex regions that represent negative and positive attitudes respectively. Our novel 3D visualization techniques provide in-depth detail about UI elements, which may not be easily covered in traditional user interface evaluation.

1.1 Challenges, motivations, and aims

Improving User Emotional Experience (UEX) [39], evaluating users' experience while observing their natural interactions with an application, has become significant in HCI research.

However, existing tools address either user experience evaluation, such as Emotient [40] and Affectiva [41], or data observation, such as Noldus Observer XT [42] and IMOTIONS [43]. Additionally, Noldus Observer XT has face reader software used to extract user emotions. None of the existing tools link data captured to user interactions with

an application. Even in the tools that evaluate user emotional experience, such as Affectiva [41], the focus is on analyzing and identifying emotional responses when watching advertisement videos. Such tools leave activities, such as linking interactions with an application, to evaluators or other additional applications. Hence, emotion evaluation is primarily based on the videos' content. Most of the solutions are expensive and require in-lab experience. Noldus Observer XT and IMOTIONS capture several types of physiological inputs, such as ECG, EMG, and EEG, but with additional hardware requirements, which increase the cost. Existing tools do not directly allow for remote testing and extensibility, such that a user is not able to customize the set of devices used for data collection.

The main motivation behind developing our tool is to overcome the above limitations, and to solve the following challenges:

- Use multimodality to collect large quantities of data, which are difficult for an evaluator to manually analyze and interpret.
- Real-time recognition of affective and mental state is challenging when it comes to building an effective and efficient model to fit facial features, and estimate head gestures.
- Common visualization techniques, such as heatmaps and scanpaths, may not be sufficiently helpful, since they do not consider the affective and mental states of users. Such visualization techniques usually rely on eye tracking.
- Remote testing and nonintrusiveness to capture affective and mental states degrade accuracy. We investigate how we can reach acceptable recognition accuracy with nonintrusive techniques.

Based on the above, in this thesis, we aim to meet the following:

- **Affordability:** Our tool by default relies on inexpensive modalities such as web cameras (see remote testing below), which are integrated with modern computers, laptops, and mobile devices. Hence, the tool can be affordable for all computer users. Additionally, the tool can still collect observational data from many other modalities, record user interaction with applications, and relate data observed to UI elements
- **Real time affective and mental state recognition:** We validate the accuracy and efficiency of recognizing the six basic emotions defined by Ekman [44], in addition to six other mental states, agreeing, concentrating, disagreeing, interested, thinking and unsure. These additional mental states are based on the mind reading DVD of Cohen, which we use for evaluation and training [45] (Chapter 9). In this DVD, the emotions have been tagged based on self-identification by the participants.
- **Multimodal signal fusion:** We implement a nonintrusive multimodal model, and validate the accuracy of recognizing affective [29], [30] and mental states from multiple sensors.
- **Remote testing:** We implement head and gaze estimators, and validate their accuracies using web cameras, such that users will not be required use to expensive solutions, such as cameras of high resolution. They can even use other cheap options such as infrared and PS3 motion cameras. We also implement a user

interaction tracker that can record user activities such as keyboard strokes and mouse clicks while they are using an application remotely.

- **Affective and mental state-based visualization:** We enhance the existing heatmap and scanpath visualization techniques, such that they can depict affective and mental states in the third dimension, such that positive emotions are drawn as peaks, and negative emotions are drawn as valleys. We mainly depend on visualization to ease the interpretation of the vast amount of data captured.

1.2 Research Questions

The following are the research questions that have guided the work in this thesis. The top-level research question is, "**How can we use real time affective and mental state recognition to extend the evaluation of user interfaces?**"

User experience evaluations, in general, are conducted in order to give insights to evaluators about how a user thinks (mental state) and feels (affective state) when using an application. Our intent is to explore potential affordable technology that can recognize in real time affective (continuous and discrete) and mental states (discrete) and link them with users' interactions. This technology will track, observe, monitor, and collect users' interactions, which the evaluators otherwise would not able to fully cover or which could not be done without the help of the machine.

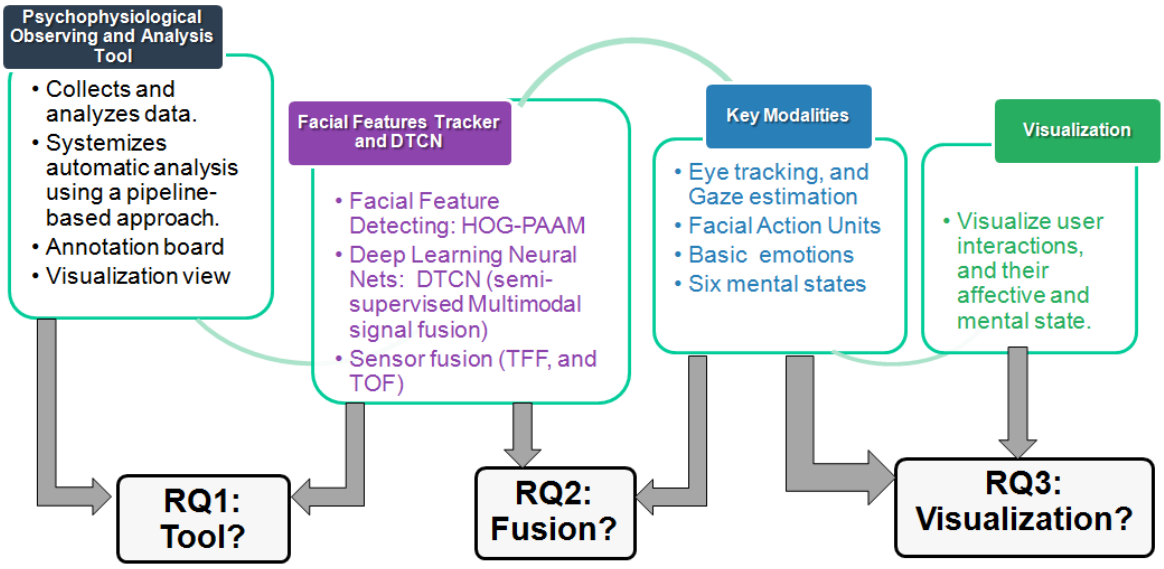


Figure 1-2. An overview of research questions

The following subsections break the top-level research question down into the key questions we have addressed (Figure 1-2).

1.2.1 RQ1: How can a tool be designed to collect and analyze observational data, and meet our goals?

A tool that potentially meets our aims, and resolves the challenges mentioned above, must provide deep and objective analysis of the affective and mental states. Our main intent is to use such a tool to validate our multimodal signal fusion model and address linking users' affective and mental state to UI elements of a user interface.

Toward that direction, we implemented a tool that follows client/server architecture, and applies pipeline-based workflow that provides a high level of abstraction, and can be represented graphically. A pipeline allows for easy arrangement of a chain of components connected together via input and output ports.

We need to investigate the creation of an extension that interprets or classifies data using easy-to-design pipeline diagrams that can be incorporated with other pipeline diagrams, or reused in other places of the tool.

We investigate implementing the following in our tool:

1. **Observational data for users**: This is from different data sources. There are two types of recording:
    a. **Psychophysiological signals:** This mainly depends on the hardware available on the user's side. In this research, we use a basic webcam, as our primary data source, in order to promote remote testing. The data is collected on the client (user's) side, which means a user must enable their webcam for data sharing.
    b. **User interactions with an application (**navigation traces**)**: Include keystrokes or mouse events, in addition to eye tracking data. A trace file is associated with the website being tested, and it is synchronized with other observational data.
2. **Annotation board**: This presents and synchronizes the observed data, and integrates modality developed using pipelines. The synchronization of different modalities is done using a multimodal signal fusion model.
3. **Multimodal signal fusion model:** This is used to show the affective and mental state data in discrete and continuous representations. The outputs of other modalities include facial action units, eye tracking data, and affective states.
4. **Visualization view**: This helps evaluators tackle the amount of the data observed, without being overwhelmed by how to interpret or analyze it. Raw data can only be looked at for further investigation.

1.2.2 RQ2: How can we build a reliable multimodal signal fusion model that provides real time affective and mental state recognition, in a way that can handle the uncertainty of sensor configuration?

The starting point of this research is to address the recent studies that analyze and recognize affective and mental states from different psychophysiological signals, and to investigate the public datasets that can validate our work.

Efficient and effective recognition of affective and mental states is crucial in HCI. Affective states can provide helpful information about negative or positive User Emotional Experience (UEX), while mental states could give insights about how a user thinks.

Affective state recognition differs from mental state recognition. It is based on concurrent and asynchronous psychophysiological signals, such as facial muscle movement, expressions, eye gaze and head gestures. Hence, it is necessary to understand the temporal sequence and the dynamicity of states among psychophysiological signals, in order to have better accuracy.

For a reliable real-time recognition of the affective and mental states, we need to investigate the implementation of a multimodal signal fusion model that is based on our novel Deep Temporal Credence Network (DTCN) (Chapter 6). This will address the following:

- **Reliable detection and tracking of users' facial features:** In our research, we implemented novel techniques that allow for both efficient and effective detection and recognition, despite using Active Appearance Model (AAM) [2]. AAM is known for its accuracy, but it is often criticised for its time complexity. In that direction, we present a HOG-Patch AAM (HOG-PAAM) [46],[47], and two fitting optimization approaches APP and APPO to improve effectiveness without degrading the performance.
- **Dynamics of interchangeable states**: This refers to a sequence of recognized actions such as eyebrow raising or lowering. We group states into 1) *discrete:* a fixed list of certain states, and 2) *continuous:* used to represent data within a range. Both groups of states are used later to show the sequence of actions among different facial expressions.
- **Sensor Fusion:** We implement two novel fusion techniques, Temporal Feature Fusion (TFF), and Temporal Output Fusion (TOF) to enable better joint representation and dynamically determine the interchangeable states among psychophysiological signals. We validate that our approach with DTCN has competitive results when compared to other state-of-the-art work.
- **Multimodality:** We will use different combinations of modalities to validate the accuracy of each modality, as well as the accuracy of the fusion model. The accuracy of each modality consists of two parts, data capturing and emotion detection. Both parts must be evaluated for each modality. We will as well evaluate the fusion model with and without the presence of each modality.

1.2.3 RQ3: How can we link users' affective and mental state to UI elements of a user interface?

One of the first steps in this research is to investigate how to enable eye gaze tracking using basic RGB cameras. We need to investigate the following issue:

- **Head gesture and gaze estimation using a monocular camera:** This requires implementing an efficient fitting technique to estimate the head pose and gaze direction, so as to know which UI elements of a user interface were used during users' interaction. Towards this, we expanded our HOG-PAAM [46],[47] model

into a 3D version, in which head positions are represented in the third dimension, such that the model can help improve the gaze estimation with monocular camera settings such as a webcam.

- **Synchronize affective and mental state recognition with gaze data and screen coordinates**.
- **Visualize user interactions, and their affective and mental state:** We present four novel visualization techniques, 3D heatmaps, 3D scanpaths, attention focus state machines, and widget heatmaps. Each technique can help an evaluator while assessing a user interface.

1.3 Contributions to scientific and engineering knowledge

The key contributions of this thesis are summarized as follows:

- **Psychophysiological observing and analysis tool for user experience:** We designed and developed our tool to integrate and validate the work done in this research.
- **Pipeline-based editor:** We use it to implement psychophysiological sensors and interaction modules.
- **Real-time facial feature detector:** We implemented a HOG-Patch AAM (HOG-PAAM) and two adaptive probabilistic-based formulations, which are Adaptive Probabilistic (AP), and Adaptive Probabilistic Project-Out (APPO).
  - We compare between HOG-AAM and our HOG-PAAM.
  - We validate that our AP and APPO approaches can outperform the state-of-the-art methods in terms of efficiency and effectiveness.
- **Real-time head and gaze estimation:** We use an approach that involves a 3D HOG-PAAM appearance model with DTCN, to enable such detection and estimation. We validate our work on public datasets.
- **Nonintrusive sensors**: We primarily uses nonintrusive techniques to capture data, as opposed to intrusively captured data. We evaluate our work in the case of a nonintrusive affective state analysis from facial features. We further use it to improve emotion recognition accuracy.
- **Multimodal signal fusion model using a novel Deep Temporal Credence Network (DTCN):** We validate the recognition accuracy of affective and mental states captured from multiple modalities using TFF and TOF.
- **Novel visualization techniques:** we implemented novel visualization technique, widget heatmaps, attention focus state machine, 3D heatmaps, and 3D scanpaths.

1.4 Outline

In Chapter 2, we give the necessary background for the reader to understand the thesis, including an overview of emotion representations and the common sensors used to capture signals (Figure 1-3).

In Chapter 3, we introduce the architecture of our tool, and explain how it is designed to work and capture data from different sources.

In Chapter 4, we demonstrate our pipelined framework, an extension to our tool. This framework is designed to ease the implementation of reusable components, such as capturing or interpreting signals from sensors.

In Chapter 5, we show the implementation of our novel adaptive probabilistic Patch Active Appearance Model (PAAM), which we primarily use for estimating (fitting) facial features, and ensure their consistent performance on different facial expressions.

In Chapter 6, a key chapter, we discuss the implementation of our novel fusion techniques, Temporal Feature Fusion (TFF), and Temporal Output Fusion (TOF), as well as our Deep Temporal Credence Network (DTCN).

In Chapter 7, we discuss the implementation of our eye tracking and gaze estimation, which are extensions to our patch active appearance model (Chapter 5).

Chapter 7 is followed by chapters demonstrating implementations of other capturing sources, facial action units (Chapter 8).

In Chapter 9, we show how we can extract basic emotions and mental states to evaluate the affective states of a user. Finally, in Chapter 10, we show the visualization techniques that we use to link among data captured from different sensors (Chapters 7 and 8), emotions (Chapter 9), and gaze information (Chapter 7).

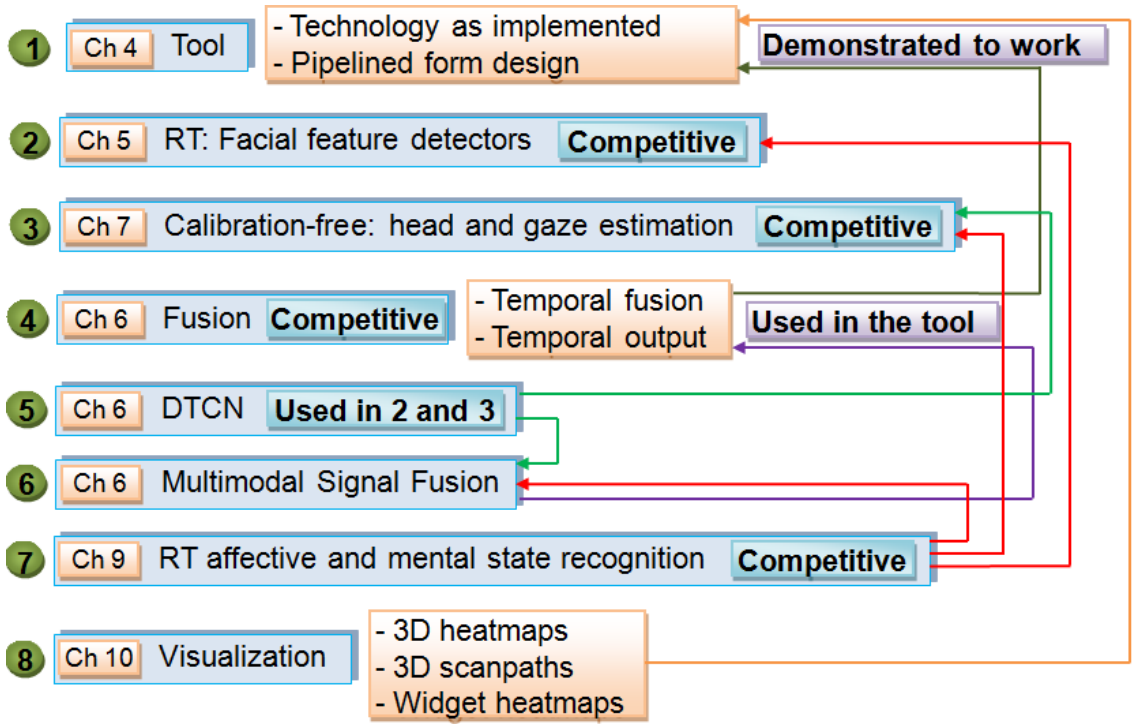


Figure 1-3. An overview of all chapters

Chapter 2 Background

In this chapter, we discuss a list of HCI psychophysiology studies that focus on detecting emotions and cognitive load as a way to understand the affective states of subjects. We compare different techniques based on their level of intrusiveness and invasiveness, accuracy, emotion representation, cost, and configuration effort. Based on our findings from this comparison, we show what factors increase or decrease the accuracy of emotion detection. We aim to highlight the major challenges involved in different techniques, and to show how they can be approached. We show different emotion types and representations related to our research. We discuss the common datasets used for emotion detection and training, and which ones we use in our research.

In this chapter, we assume that readers have a basic knowledge of computing, but are in need of knowledge of affective computing and software usability in order to understand later sections of the thesis. Terms crucial to our research are defined briefly in the Glossary, which appears at the start of the thesis.

2.1 Introduction

In general, psychology focuses on studying the state of mind, while physiology focuses on studying body changes [27], [48]. Psychophysiology focuses on measuring the physiological responses that occur because of changes in psychological state in order to estimate what is in the human mind.

Responses measured are used to construct *indicators*, which are used to estimate cognitive or affective states [5]. Examples of indicators include heart rate, eye movements, body movements, gestures, and blood pressure. An affective state is a psychophysiological construct that ranges based on three main categories: valence (positivity or negativity), motivational intensity (control), and arousal [5]. For example, elevated heart rate can indicate fear, anger, or fatigue.

Our focus is mainly on emotion, and effort and time-based indicators. In terms of effort and time-based indicators, we focus, in particular, on cognitive load indicators. In our comparison, we selected studies that rely on the implementation of any or both of these types of indicators. We categorize sensors listed in these studies based on their intrusiveness and invasiveness.

We use the tem *index* as a scaled composite variable encompassing a number of indicators captured over a range of time to show the peak or valley of some data. For instance, an index can show the number of occurrences of the happiness emotion at a specific time range (e.g. seconds 12 to 22).

We refer to someone conducting a psychophysiological study on a number of subjects or users, as an observer, evaluator, or study conductor.

2.2 Emotion representations

The common emotion representations are discrete and continuous [49]–[52]. We use both representations in our research.

Discrete representations use labels of nouns or adjectives to describe emotions such as anger and happiness [53], and intensity such as anger versus rage. Cultural differences can cause misunderstanding when selecting words, which means that the emotion intensity will not be precisely represented.

To overcome discrete representation limitations, many studies focus on *continuous representation* [49]–[52], in which emotions are shown in a multidimensional scaling or continuous space depicting the emotional distance among variables; e.g. intensity of emotions, such as happiness and sadness.

Research studies describe various emotion spaces, but the most common dimensions that appear are the affective states of *valence* and *arousal* [54]. The valence dimension shows the positivity or negativity of experience such as being pleased versus unpleased. The arousal dimension is used to represent the reaction strength such as alertness and awareness. When an emotion starts due to a motivation, it becomes activated, and when it ends, it becomes deactivated. The strength of reaction depends on activation or deactivation states. The arousal dimension can also be called the sleep-tension or activation dimension [55].

The third dimension is intensity, control affective state (intensity), can be used to distinguish between emotions that occur due to activation or deactivation reactions (Figure 2-1).

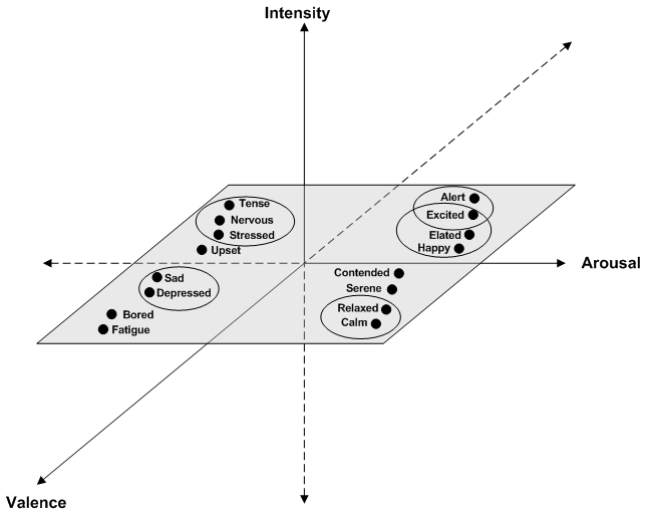


Figure 2-1. Valence arousal view of emotional experience

Many studies focus on 2D representations in order to avoid participants' bias that can occur when using the control affective state [54]. A 2D representation view can represent any emotion just as a coordinate even if it does not have a concise expression or label. Emotion intensity can be depicted simply based on the distance between the emotion point and the centre of the space [54].

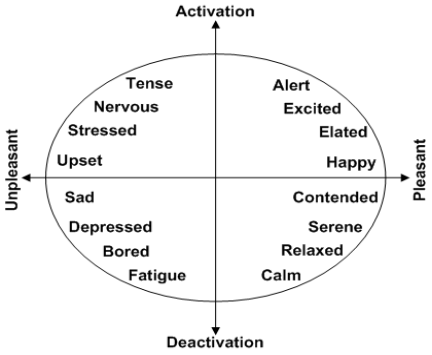


Figure 2-2. A 2D emotional experience view [55]

2.3 Emotion types

The emotion types we focus on in our research are basic, secondary, and compound emotions.

Basic emotions are fundamental and universal across cultures with respect to their cognitive effort, effective adaptation, development during childhood, and their effect on the person's way of thinking and understanding [56]. A basic emotion is primary, which means that it cannot be decomposed into other emotions, and it is evolved as a quick response or preparation towards stimulus. In our research, we focus on the famous six basic emotions of Ekman: anger, fear, surprise, disgust, enjoyment (happiness), and sadness [57].

Secondary emotions usually occur as reactions to primary emotions or existing feelings. Examples include feeling anger upon hurt which was originally caused because of an injury [56], [58]. In contrast to primary emotions, secondary emotions involve a higher level of processing of the stimuli, and can vary in reaction intensity. Such emotions take the form of additional reactions or even chains or reactions.

Basic emotions are used to compose the remaining emotion vocabularies that can sum up to include thousands of *compound emotions* [45]. For example, in some cases, the contempt emotion, which is a mix of disgust and anger, is used with Ekman's basic emotions.

Du et al recognized compound emotions such as happily disgusted and sadly angry [59]. Such emotions may seem contradictory at the beginning but they are possible. For example, if someone is watching a movie and one of the scenes is funny but disgusting at the same time, then this will trigger the emotion of happily disgusted.

Plutchik described three compound emotions primary (love), secondary (guilt), and tertiary (delight). Each compound emotion is a pair of two basic emotions. Love is composed of joy and trust, guilt is composed of joy and fear, and delight is composed of joy and surprise [60].

2.4 Invasiveness and intrusiveness of sensors

A sensor can be either *invasive* or *non-invasive*, and a non-invasive sensor can be either *intrusive* or *non-intrusive* [61].

A sensor would be called *invasive* if it requires perforation of the body, exposure to radiation (such as X-rays) or something similar. A non-invasive sensor might be wearable or placed directly on the body but pose essentially no risk. Obviously, users are more comfortable and accepting of non-invasive sensors. Measuring blood sugar with a needle is an example of an invasive sensor.

A sensor would be called *intrusive* if it requires direct contact with the body or in general if it gets in the way of normal behaviour. A particularly intrusive sensor with a higher level of restrictions can be called *encumbering*; an example would be a head mounted device that prevents the user from moving naturally. The nature of most physiological recording using current technology is intrusive.

A non-intrusive sensor, such as a webcam, does not require direct contact. For example, we can extract facial expressions and eye movements to obtain physiological states in a non-intrusive manner.

Sensors are devices or equipment whose size can widely range; examples include wearable watches a MRI scan machine. We can as well refer to a sensor as a technique in some places. For example, if a web camera is used to capture physiological signals, we refer to it as a technique, since additional software effort is required to translate the recorded video into meaningful psychophysiological data.

In Table 2-1, we show a summarized list of some physiological inputs, all of which are intrusive except for video cameras and audio. The list includes Electromyography (EMG), Electroencephalogram (EEG), Magnetoencephalography (MEG), Electrocorticography (ECoG), Magnetic Resonance Imaging (MRI), fMRI (functional MRI), PET (Positron Emission Tomography), SPECT (Single Photon Emission Computed Tomography), Near InfraRed Spectroscopy (NIS), functional NIRS (fNIRS), Photoplethysmography (PPG), Electrocardiogram (ECG), Electrooculography (EOG), Respiratory Inductance Plethysmography (RIP).

Table 2-1. An overview of common physiological indicators

Physiological measurement	Instruments	Emotion representation
Facial muscles	EMG	Facial expressions
Brain signals	EEG, MEG, ECoG, Chronic electrode implants, fMRI, PET, SPECT, fNIRS, Emotiv headset [62], NeuroSky biosensors [63]	Mental states from different waves
Blood pressure	Sphygmomanometer, stethoscope, palpation, auscultatory, pscillometric, cannula needle (invasive)	Stress, anxiety, problem solving
Blood flow	Plethysmograph, ultrasonic flow meter MAG meter, fMRI	Similar to blood pressure
Blood volume	Plethysmograph	Similar to blood pressure
Blood Volume Pulse (BVP)	PPG	Emotional pattern such as shock, fear, anxiety
Heart rate	ECG, PPG	Fear, happiness, sadness, disgust, anger
Eye blink	EOG, tracking algorithms	Stress, nervousness, fatigue
Eye pupillary responses	EOG, tracking algorithms	Arousal, valence, Stress, nervousness, fatigue
Respiration	Spirometer, RIP, peak flow meter, temperature sensors	Relaxation, stress, valence, arousal,
Speech Recognition (SR)	Sentiment analysis (audio)	polarity, emotions, strength of speaker's opinion
Facial expressions	Video camera	Basic, continuous and compound emotions

2.5 Key challenges of psychophysiology in HCI

Existing HCI tools that depend on capturing psychophysical data are subject to certain limitations and challenges [64].

Dealing with the dynamic nature of behaviour and interaction of users is tricky [65]. For example, some physical activities can happen concurrently, and influence the physiological data being captured. As another example, the current social context can cause the user's behaviour to be different. For instance, some subjects may not behave the same if they asked to perform a task in a lab as opposed to asking them to perform the same task on their own laptops.

The pace of advancement in psychophysiology is not the same as in HCI. The cost of additional equipment or devices used to capture data is the main reason of hindering the development pace. In many cases, studies can be conducted only in the lab due to equipment affordability or portability.

Additional methods and techniques of analysis are usually associated with the introduction of new equipment. More effort is consumed to ensure that captured data from new devices is compatible with other data captured from old devices; otherwise, a data migration technique must be also implemented.

Individuals may feel uncomfortable when using additional sensors [65]. Some sensors can pose additional intrusiveness or restrictions. For instance, a subject may be required to reduce their head movements as much as possible to match a requirement of an eye tracking technique algorithm based on a web camera. Eventually, some users may tend not to adhere to such requirements; this can have a direct impact on accuracy

The collected data is often not absolute, as it can have multiple interpretations. For example, elevated heart rate can be interpreted as either fear or cardiac effort. Many external factors or data interference can affect any psychological study. For example, heart rate results change depending on a person's fitness or stamina. Using simple stimuli can help avoid or at least reduce unexpected circumstances.

When interpreting psychophysiological data in the HCI context, external factors are particularly complicated [65]. For example, heart rate results can have an open-ended number of interpretations such as the user being confused about something in the UI, typing very fast at a particular point in time, or something being shown unexpectedly on the screen.

External factors can also affect the user's emotions. For example, when assessing users' experience while using an application, other factors not necessarily related to this application can cause emotions to change. For instance, according to Lazar et al [66], an average of 42-43% of students reported that their frustration came because of technical problems faced while trying to perform given tasks.

Some factors can co-occur with emotions, and some of which are user-dependent such as personality traits and temper. For example, a user with calm personality can take time to understand how to accomplish tasks, and thus will not necessarily end up with a negative sentiment toward an application. If a user is short-tempered, negative impacts can increase as any small signs of difficulties can irritate them. As a result, a study conductor may mistakenly interpret such experience as an issue with the application rather than realizing that this was because of personality traits.

Some data collected is subjective to noise. Examples include Electromyography (EMG) and Electroencephalogram (EEG). Additional effort is therefore required to reduce such noise.

Environment factors have their direct impacts on physiological signals. For example, mood can change because of bad weather conditions [67]. The accuracy of recorded data can be greatly affected due to the environment such as brightness.

Using multiple modalities can help increase the signal capturing process; however, it can cause additional challenges, such as data inconsistency and very high volume of data.

2.5.1 Variability of physiological responses

Signal variability can cause additional issues such as data inconsistency. This can be alleviated based on contexts and emotion perspectives. Variability can be either *intervariability* or *intravariability*.

Intervariability refers to the variability among different individuals. Differences can occur because of physiological, behavioural, personal, or biological factors. For example, physiological responses are different in athletes as compared to those who do not practice sport [68].

Intravariability on the other hand refers to variability observed in the same individual. Variability can exist over time due to reasons such as mood changes, new stimuli, and new experience. For example, Electroencephalograms (EEG) shows different signals because of mood changes [48]. Caffeine and nicotine intake also cause physiological responses such as blood pressure changes and headache [68], [69].

It is important to reduce unnecessary chances of variability when observing physiological changes. For example, if a study is going to take a long time, the conductor must make sure to eliminate factors that can affect mood such as asking individuals to have their regular cups of coffee during the experiment.

Controlling variability becomes more difficult in cases such as remote or lengthy studies. Alternatively, baseline monitoring can be applied [51]. In baseline monitoring, from day to day, before an emotional assessment, physiological features are extracted in neutral states. An emotional assessment can be later mapped with the baseline extracted in order to reduce intervariability. Creating a baseline however, requires capturing a wide range of data over time, and additional recalibration [51], [61], [70].

Detecting emotions based on physiological responses may be inconsistent, since physiological activation depends on actions or stimuli incoming from different physiological sources. Stemmler et al indicated that context identification could reconcile this inconsistency [71]. A context refers to a similar pattern of conditions that activates certain emotions. Such emotions can later be extracted based on a context.

A context depends on the presence of one or more of the emotion perspectives: Darwinian, Jamesian, social constructivist, and cognitive theory [72]. Tracking all perspectives is not an easy task, since this can increase the possible number of contexts. Concentrating only on specific users or applications, helps reduce the number of contexts. Controlling the number of contexts can improve the ability of emotion elicitation as much as possible [70].

2.5.2 Psychophysiological monitoring issues and threats to validity

There is a possibility to experience delays in receiving physiological signals. As a result, computer responses may show different results depending on the amount of delay. Technical issues may cause delays as well, such as video quality or packet loss in networking communication.

Similarly, some physiological signals may be delayed as they simply come later as a response to other signals. Recording can therefore be real time, but analysis and visualization will be affected by the delay in response.

Physiological signals can have latency with respect to the reason of occurrences. For example, Anttonen and Surakka indicated that they found a six-second difference between stimulus and heart rate signals [73]. As another example, changes of blood pressure take more time before they appear as compared to brain signals.

Most studies focus on monitoring physiological activities under controlled conditions over a short period. This will only highlight certain behaviour or affective states of users. Studying subjects in uncontrolled environments pose more challenges due to the difficulty of monitoring physiological activities. This can become more challenging with multiple modalities, or when nonintrusiveness is required.

Intrusive techniques in an uncontrolled environment may be challenged, as it will be the subjects' responsibility to configure sensors. Configuration effort can be reduced by asking users to use convenient-to-use devices such as watches or wearable bracelets [51]. For capturing signals from sources such as the brain, some easy-to-configure tools can be used such as Emotiv [62] and NeuroSky [63]. However, this will increase cost.

2.6 Previous work on psychophysiological monitoring in HCI

In this section, which is the core of this chapter, we compare an extended list of studies related to emotion detection. First, we will show our comparison criteria. Second, we will compare our selected studies based on such criteria. Finally, we will discuss the findings of these studies.

2.6.1 The criteria of psychophysiological studies

Our list of criteria includes physiological modalities, emotion representation and emotion elicitation, number of subjects, contextual information, time criteria, and assessment methods.

Either subjects or events can elicit emotion. In subject-based elicitation, a subject has to either remember past events or act as though they are experiencing particular feeling or emotion [51], [74]. For example, a subject can be asked to act as if they are experiencing a feeling of anger; this requires the subject to mimic a previous similar experience. Subject-based approaches usually depend on recognizing facial expressions in order to extract emotions.

In event-based elicitation, emotional states of a subject are evoked by showing or letting this subject interact with media files such as video, audio, and pictures. There are several examples of media databases such as the International Digitized Sound System (IADS), Augsburger Database of Biosignals (AuDB), and International Affective Picture System (IAPS) [75], [76].

Complexity of emotion elicitation increases or decreases based on the complexity of the emotions targeted. For example, compound emotions (Section 2.3) — non-basic emotions — such as anxiety are more difficult to detect, compared to the simple emotions of Ekman [57] such as sadness.

Experts are required to evaluate the affective states, which may be challenging and introduce some inaccuracy due to their subjectivity; however, this tends to be more the case for complex and spontaneous emotions in the wild [15], [16]. For a previous experience, subjects still need to report or mimic their feelings or emotions.

In terms of time, emotions can vary based on their length of occurrence. The focus should be on differentiating between emotions that do not last for a long time and those that last for longer periods of time. In that case, emotions should be detected regularly in a way that matches subject's reactions over time. For example, an emotion of relief can occur after a series of surprise emotions.

Temporal analysis requires extraction of statistical features of a stream of data over time. Time-based monitoring is however not an easy task, since captured emotions from different physiological modalities are not easy to synchronize. For example, brain signals get updated quicker than temperature signals [77]. Increasing the number of subjects can increase the accuracy and validity of results; however, it will increase cost and effort accordingly.

A contextual study means that it in particular targets specific users or applications, as a way to decrease variability.

Assessment method types include feature selection, inference, or regression. In feature selection, methods properly select the physiological features that correlate with the affective states of subjects. Examples of feature selection methods include Genetic Algorithm (GA), Sequential Backward Selection (SBS) and Sequential Floating Forward Search (SFFS) [50], [78].

Trying to assess the results from different studies can be challenging because of the differences in feature selection. Usually feature selection takes place after feature extraction in order to remove redundant features, and to maintain consistency. Also, the studies differ in how they prepare their data for training in terms of split ratio between training and testing sets, or different model validation techniques such as cross-validation.

Feature selection is not an easy task because of the possible intervariation and intravariation. An appropriate selection of features can help reduce the variability space before using Machine Learning (ML) methods to infer affective states and emotions.

Examples of ML methods include k-Nearest Neighbours (k-NN), Discriminant function analyses (DFA), Marquardt Backpropagation (MBP) [61], Bayesian Networks (BN), Linear Discriminant Function (LDF), Neural Networks (NN), Autoassociative Neural Networks (AANN), Stepwise Discriminant Analysis (SDA), Support Vector Machines (SVM), Linear SVM, Common Spatial Patterns (CSP), Linear Discriminant Analysis (LDA), Adaptive and Network-based Fuzzy Inference System (ANFIS), Multilayer Perceptron (MLP) [79], Kernel Density Estimation (KDE) [79], and Higher Order Spectra (HOS) [78]. On the other hand, regression methods are important to detect continuous emotions. Examples include Regression trees, Regression Neural Networks, and Dynamic Bayesian Networks (DBN).

2.6.2 Comparison

Table 2-2, Table 2-3, and Table 2-4 show a comparison among different studies based on the criteria mentioned above. These three tables refer to the same comparison. We split the comparison into three to improve readability. Each entry in a table has a label ranging from A-T. In order to locate an entry in the three tables, we look for its label.

Table 2-2. Study overview of emotion extraction (1)

	Study	Elicitation	Modalities	#Subjects	Contextual
A	Nogueira et al [49]	Playing video games	HRV and EMG	22	No
B	Giakoumis et al [80]	Playing the labyrinth game	ECG	19	No
C	Rahnuma et al [79]	IAPS (audio and video)	EEG	8	No
D	Hosseini [78]	Images	EEG	15	Yes
E	Liu et al [81]	IADS (music and sound)	EEG	12	No
F	El Kaliouby [82]	Mental states were trained on the set of videos from the mind reading DVD [45]. After that, the system was tested on each video in the CVPR 2004 corpus, this corpus was developed by the researcher.	Facial expressions and mental states	N/A (based on dataset videos)	Yes
G	Pourzare et al [83]	A combination of EMG and EOC signals influencing EEG signals	EEG	3	No
H	Kim et al [52]	Story narrating, music, and objects such as a toy	HRV (ECG and PPG)	50	No
I	Wagner et al [84]	A dataset from MIT media lab [51].	EMG, ECG, respiration	1	No
J	Haag et al [70]	IAPS (images)	BVP, HRV, EMG, and respiration	1	No
K	Sinha et al [85]	Subject-based	HRV and EMG	27	No
L	Leon et al [86]	IAPS (images)	HRV and BVP	8	No
M	Kim [50]	Music and songs	HRV and BRV	3	Yes
N	Rainville et al [87]	Subject-based	ECG, and respiration	43	No
O	Picard et al [51]	Subject-based	ECG, EMG	1	No
P	Katsis et al [88]	Controlled environment providing realistic driver's conditions	EMG, ECG, respiration	10	Yes
Q	Rani et al [89]	Playing pong game, solving anagrams	EMG and heart sound	15	Yes
R	Lisetti and Nasoz [61]	Movie clips or math problems	HRV	29	No
S	Li and Lu [90]	Images	EEG	10	Yes
T	Takahashi [91]	Commercial films	EEG	12	No

When referring to 4-NN, we refer to 4- Nearest Neighbours. MLP is a feed-forward artificial neural network technique that consists of hidden intermediate layers [84]. MLP with five hidden layers is denoted as MLP5.

The best event-based results were in the research conducted in (Rows R and I); both studies used different stimuli. In (Row R), subjects were asked to report their emotions while watching movie clips that were selected in a pilot study. In (Row I), subjects were asked to select music of interest from AuDB that matches their emotions. The subject-based results of the highest accuracy were in (Row O). The emotion relied on a single subject (who is the experimenter as well) that was able to remember past events that matched emotion elicitation.

In terms of brain signal tracking, (Row T) used EEG to track brain accuracy but the accuracy did not exceed 42%. On other hand, Li and Lu obtained (Row S) high accuracy in their EEG study, which reached 93%. However, the emotion complexity was not high as they only tracked smile and cry statuses. Similarly, (Row G) reached high accuracy using EEG; however, the focus was on tracking facial movements not on extracting emotions. We found that related studies of EEG, heart rate and respiration require further feature extraction and pattern classification analysis.

Several studies that used ECG and EMG showed consistent results. Both ECG and EMG happened to appear together in several results such as (Rows I, J, and K).

Physiological signals in many studies are used independently to extract statistical features. This assumption will cause signals not to be correlated among different modalities as physiological signals occur from different sources from the body as we stated earlier.

We can see that some studies relied on the basic emotions of Ekman [57]; others provided a custom set of emotions that ranges from basic to complex emotions to focus on. Other studies relied on continuous emotion representation such as valence and arousal. Accuracy using valence-arousal representation is lower as compared to using the basic emotions. Valence-arousal is a continuous representation that is not easy even for experts to interpret and can be associated with misinterpretation.

Studies applied on a single subject had highly accurate results. However, further investigations are required to adapt any of these solutions. On the other hand, accuracy never exceeded 85% when the number of test subjects exceeded 15.

Table 2-3 Study overview of psychophysiological signals (2)

	Emotional Representation	Assessment	Results
A	Arousal	Regression Model and Residual Sum of Squares-based (RSS)	85%
	Valence		78%
B	Boredom	LDA	94.17%
C	Happiness, calmness, sadness, and fear	KKDE and MLP	76%
D	Calm-neutral and negatively exited	SVM, LDA, HOS, and GA	82.32%
E	Joy, sadness, anger, and pleasure	SVM	90.72%
F	Agreeing, concentrating, disagreeing, interested, thinking, and unsure	DBN	63.5%
G	Facial movement	k-NN	94%
H	Valence	SVM	78%
	Sadness, anger, and stress		78.4%
I	Anger, joy, sadness, and pleasure	LDF	92.05%
		5-NN	90.91%
		MLP6	88.64%
	Valence	LDF	86.36%
		5-NN	86.36%
		MLP6	88.64%
	Arousal	LDF	96.59%
		5-NN	94.32%
		MLP6	94.32%
J	Arousal	NN	96.6%
	Valance		89.9%
K	Fear and anger	DFA	78%
	Neutral		72%
L	Neutral, positive, and negative	AANN (original set)	71.42%
		AANN (revised set)	80%
M	Arousal	SBS and LDA	98%
	Valence		91%
	Anger, joy, sadness and pleasure		87%
N	Anger, sadness, happiness, fear	Stepwise discriminant analysis	65.3%
O	Neutral, anger, grief, hatred, platonic love, romantic love, reverence, joy	SFFS	81.25%
	Arousal (very high, medium high, high, low, or very low)		84%
	Valence (very high, medium high, high, low, or very low)		87%
P	High stress, low stress, disappointment, and euphoria	SVM	79.3%
		ANFIS	76.7%
Q	Engagement, boredom, anxiety, anger, and frustration	Bayes network	74.03%
		k-NN	75.16%
		Regression tree	83.5%
		SVM	85. 1%
R	Sadness, surprise, amusement, frustration, anger, and fear	k-NN	72.3%
		DFA	75%
		MBP	84.1%
S	Happiness and sadness	CSP	93.5%
		Linear SVM	93%
T	Sadness, happiness, relax, joy, and anger	SVM	41.7%

Table 2-4 Study overview of psychophysiological signals (3)

	Data collection and time aspects
A	Seven subjects were playing at least monthly and the rest were playing intermittently
B	A session ends when a subject finds the labyrinth exit or when the time spent exceeds 10 minutes
C	A subject is exposed to all four emotion stimuli; a minute for each stimuli
D	Each recording lasted for about 3 minutes
E	Two five-session experiments; a session in the first experiment takes 60 seconds and there is a minute of silence between sessions
F	Three videos lasted less than two seconds; the remaining 88 videos lasted approximately 313 seconds
G	A subject was asked to relax for five minutes; there was a 5-second gap between trials
H	Sampling rate of raw ECG is 256 s^{-1} and HRV is 4 s^{-1}
I	25 recording per 25 days, for each emotion

	Data collection and time aspects
K	General screening, interview, training, and two experimental imagery sessions
L	Two different sessions
M	Data is randomly selected from an experimental session
N	Over three month, 90 samples for each emotion from each subject are collected. A sample ranges between 3 to 5 seconds based on the song length.
O	90 seconds for an emotion
P	2000 to 5000 samples per emotion per signal per day, over 20 days
Q	10 seconds window of each signal
R	One-hour sessions on different days; three sessions solving anagrams and three sessions playing pong
S	45 minutes to establish a baseline; movie scenes lasted from 70 to 231 seconds
T	3-second and 1-second trials

2.6.3 Comparison discussion

The accuracy of an assessment or evaluation process depends on the sensors, and implementation of the indicators used. We pointed out that a sensor has intrusiveness and invasiveness levels. Some sensors can have high accuracy but on the other hand can require perforation of the body; certain subjects may not be accepting of this kind of invasiveness. On the other hand, some sensors can be non-intrusive, which are likely to be accepted by almost all subjects; however, accuracy can sometimes be a question.

On the other hand, the implementation of an indicator directly affects accuracy. For instance, an emotion indicator can be simplified such that it only focuses on certain types of emotions such as happiness and sadness. Other emotion indicators can require more implementation effort, but allow assessing compound emotions as well as the basic ones.

It is not intuitive to draw a conclusion to determine which sensors or indicator implementation is better. In our research, we mainly aimed to compare existing psychophysiology studies in the field of HCI, and to determine the trade-offs involved. It was important to analyse many studies in order to ensure that our evaluation is deep enough.

We needed to set clear evaluation criteria that help us to distinguish among different studies, especially because collected data in psychophysiology research is usually quantitative.

Unfortunately, accuracy may not be a sufficient indication that a study is better than another study. The reason is that almost all studies have different emotion representations, captured modalities, number of subjects, and procedures of data collection over time.

We can summarize the main points we collected in our comparison study as below.

- Some studies such as (Row M), showed very high accuracy but the number of subjects was too small; other studies such as (Row H) showed less accuracy but they tested their results against a high number of users.
- Less complexity of emotion representation is usually linked with high accuracy such as shown in (Rows S and M).
- Studies that used multiple modalities such as (Rows J and I) are usually associated with high accuracy.
- Studies such as (Row F) relied on non-invasive approaches to assess complex emotions but showed lower accuracy.
- The only study that assessed compound emotions from our selected studies is (Row F). In this study, accuracy was degraded due to ethnicity differences. Their work was trained against the "Mind reading" DVD [45], in which subjects were mostly British; however, the dataset CVPR 2004 corpus in [82], was recorded later from subjects from different countries and ethnicities. At the same time, the recorded dataset had some issues related to the environment such as recording conditions.
- The only study in our list of comparison that assessed cognitive load represented as stress indication is (Row P). In terms of calculating time and effort in HCI, we found other studies such as [92]. In [92], they were able to detect some usability issues related to users' performance in controlled experiments; the used modalities included eye tracking, mouse, and keyboard. It is important to mention that we did not include some studies in our comparison table such as [92] because it is mainly used to measure user performance, and does not include accuracy for effort and stress indicators.
- Studies that showed the highest accuracy tended to have fewer subjects. Even in (Row S), which has high accuracy with a relatively accepted number of subjects (10) used simplified emotion representation.
- The studies such as (Rows H, N, Q, and R), which were applied against the highest number of users were contextual. We indicated before that a contextual study means that it is subject to certain environment and application restrictions. This can somehow indicate that restrictions are usually associated with increasing the number of subjects. There are still studies such as (Row M), which were contextual and tested against a small number of subjects.
- Some signal sources seemed to be highly affected by the type of emotion representation. For instance, ECG in studies such as (Rows I and J), relied on the valence-arousal (continuous or dimensional) representation and showed higher accuracy than (Row H, K, and T), which relied on basic emotion representation. Even in studies such as (Row O), there was a direct comparison between both representation; the valence-arousal representation showed higher accuracy.
- Studies such as (Rows A, B, and Q), used games for emotion elicitation and showed medium accuracy as compared to other studies. It is important to consider the fact that playing games is usually associated with high cognition [93]. Thus, most likely, subjects in those studies had a high level of cognition.

- The studies in (Rows Q, N, and O), are subject-based but did not show high accuracy as compared to other studies in the comparison. Accuracy seemed to be lower when the time the experiment takes is increased, as in (Row N).

Not all studies listed focused on resolving issues related to intravariability. The main aspect that we are referring to is assessing emotions of a user, in a way that avoids issues such as mood changes. For instance, a user maybe in a bad mood in a morning due to whatever external reason unrelated to the experiment. At night, their mood can return to normal. The only practical solution that we can consider in such a case is to ensure that experiment remains operational through both periods. However, this will only be possible for specific types of devices that can be used all the time. Examples include wearable Electrodermal Activity (EDA) sensors [94].

The above aspects may not be the same for all studies. For instance, study (Row O) showed higher accuracy than (Row T) although the emotion representation was more complicated. At least, the number of participants in (Row O) was so much higher than (Row T). In addition, (Row O) relied on using multiple modalities.

We can conclude that testing against more participants paradoxically can decrease accuracy. As well, obviously, it causes cost and effort to be increased. However, it will remain crucial to test against many users if we need to validate a study.

Using multiple modalities and adopting multimodal fusion technique can be a good way to increase accuracy and reliability of the results. Data such as brain signals can reveal emotions in a different way from what a human can naturally detect. For example, some people may be able to hide their facial expressions [95] meaning that their emotions will not be properly detected; their brain signals however can still possibly reveal their actual emotions.

However, on the other side, multiple modalities increase cost and configuration efforts. Using tools or techniques with low levels of intrusiveness and invasiveness can decrease cost and somehow alleviate these challenges.

Non-invasiveness is challenging as it requires additional programming effort to estimate the missing information and/or data, and at the same time, its accuracy is likely to be lower as compared to invasive approaches.

On the other hand, configuration efforts can be at minimum with non-intrusive and/or non-invasive techniques. For example, a web camera can capture emotions from facial expressions; in such a case, there will be no configuration requirements. As well, the cost will be minimized. However, additional development effort will arise. For instance, experience in computer vision, machine learning, real-time, and algorithm design is required to implement mechanisms to detect and track faces, extract facial expressions, and deduct emotions from the facial expression extracted.

Not all approaches can be non-intrusive and/or non-invasive. For example, extracting brain signals requires some physical contact, hence is intrusive. The configuration of EEG sensors is not easy for beginners; they must be aware of configuration systems such as 10-20 [96].

Chapter 3 Architecture of a Tool to Fuse Psychophysiological Data

In this chapter, we give an overview for the architecture of the tool we implemented to fulfil the requirements of this research, especially in terms of handling psychophysiological responses. Studies related to the analysis of psychophysiological responses have become increasingly relevant in HCI. However, the available tools lack systematic and extensive analyses that cover the complexity and asymmetry of psychophysiological sensors, and appropriately synchronize signals with user interactions. Our tool enables evaluating user experience by linking the psychophysiological responses of users with their interactions. The tool helps collect and observe the psychophysiological signals, navigation traces, and attention focuses of users while they are interacting with applications. We provide a pipeline-based approach that systemizes automatic analysis, data annotation, and data visualization, as well as affective state detection. We pay attention to the necessity to support extensibility as a way to handle the analysis complexity.

In this chapter, we assume that the readers have basic knowledge of web development.

3.1 Introduction

Our tool is designed to provide solutions to evaluate user experience, specifically for web applications. The tool provides an easy way to record user interactions and observe their psychophysiological signals. Data can be observed and collected remotely.

We support lazy rendering of the massive amount of data captured from multiple sensors.

Real-time synchronization helps increase the reliability of analysis and interpretation of the observations. Sensor signals must be captured in a real-time manner, such that data recorded become associated based on time of occurrences. For example, a trace entry needs to be linked to user interactions including their *gaze* data. This helps an evaluator to know what UI elements the user was interacting with at the time each entry was recorded, as well as the other parts of the screen they looked at.

We adopt a *pipeline*-based approach to provide a high level of abstraction, which can also be represented graphically. This approach promotes component-based development, in which a system is modularized as a number of components interacting and transmitting data via connectors and ports. A component is used to analyze and interpret the data observed.

Component-based development allows for extensibility, such that a component is developed in a target language of interest; it can be either simple or bridged to external libraries. For instance, computer-vision-related classifiers can use OpenCV to interpret data observed.

A user of our tool can use components developed by other experts, and may only need to arrange the components and connect them together.

Our main contributions in this chapter can be summarized as follows:

- We provide the architecture of a tool that can collect observational data of users from different data sources, in a way that:
  - Incorporates real-time trackers that allow remote observation of the psychophysiological signals and interactions of users.
  - Enables additional trackers to be integrated.
  - Links between users' affective states and their interaction traces with an application.
- We implement a pipeline-based approach that eases the process of creating user defined analyses and interpretation modules.
- We implement an annotation board, which is used to synchronize the data observed from different sensors based on the user-defined modules developed as pipelines.
- We implement a number of nonintrusive real-time trackers to detect affective states, facial expressions, and eye gazes.

3.2 The top-level view of the architecture

The architecture followed in our tool is client/server. Session recording takes place at the client side, and the storing and analysis of the observed data is done on the server.

Our tool is divided into four main independent components; session management service, management dashboard, pipeline workflow, and visualization (top of Figure 3-1).

A client needs to include a JavaScript file in their main application, and provide contextual details, such as user data. A JavaScript client provides REST services to connect to the communicator services (Figure 3-1), which establish sessions with that client. Once a session starts, traces of users will be recorded.

The fast development of technologies related to capturing psychophysiological signals necessitates supporting extensibility, and the ability to cope with the complex and voluminous data observed. Extensibility implies systematizing the development of new analysis modules that can interpret data observed from new types of sensors in an automatic manner.

The analysis service allows developing modalities that use custom traces, and permit those traces to be effectively visualized and analyzed.

A server administrator is able to use the dashboard service to manage the observed data and review the active sessions being recorded.

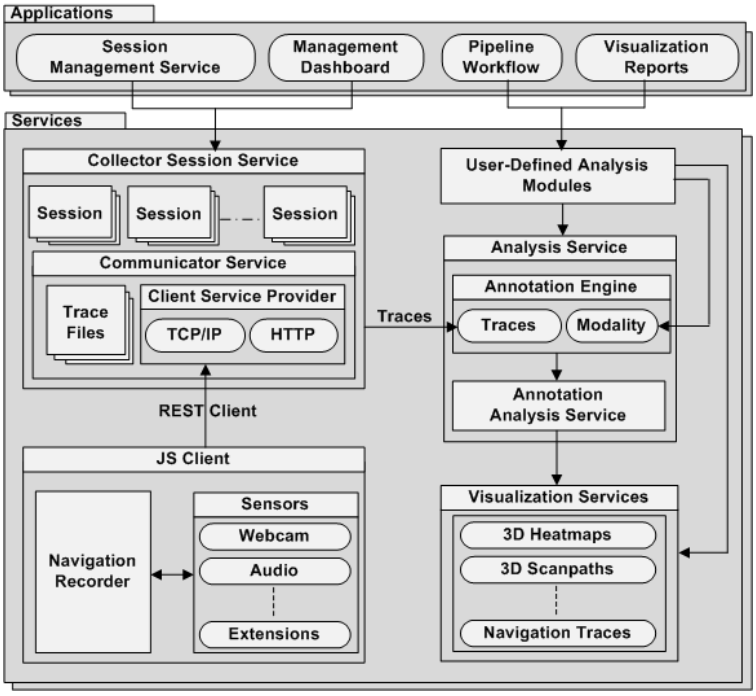


Figure 3-1. Architecture overview

3.3 Data collection

We define two types of data observation; *navigation* and *psychophysiological* traces.

Navigation traces are related to user interactions with an application, mainly keystrokes and mouse clicks as well as gaze data, which shows users' attention or distraction in terms of events; *onGazeMove* and *onGazeIn* for an eye fixation, and *onGazeOut* for the end of a fixation. These events are custom; i.e. not native JavaScript events.

P*sychophysiological traces* typically require sensors to capture psychophysiological signals while a user is interacting with an application. These sensors can be connected with the server through TCP/IP or HTTP. A sensor can be either intrusive or nonintrusive. We promote nonintrusive sensors, primarily based on web cameras, in order to ease remote data observation, reduce cost and make data gathering easier on participants. Nonintrusive data sources can enable real-time recognition of the affective states.

If a user decides to use sensors, other than a simple camera, it will be their responsibility to set them up and to invoke the REST services our tool provides to send the data captured by those sensors.

During a session, we capture screen shots of the pages on which the interactions are taking place. Our capturing is done internally in the browser, such that the entire user screen will not be included. In order to avoid storing a large number of snapshots, we use SURF [97] to calculate the difference between the new image captured and the previous image. If the differences are less than a threshold, we do not store that image, since it will be redundant. In the annotation board, the thumbnail will show the same image, but its time range will increase to span its occurrence and the occurrence of the new image captured, which was omitted (Figure 3-2).

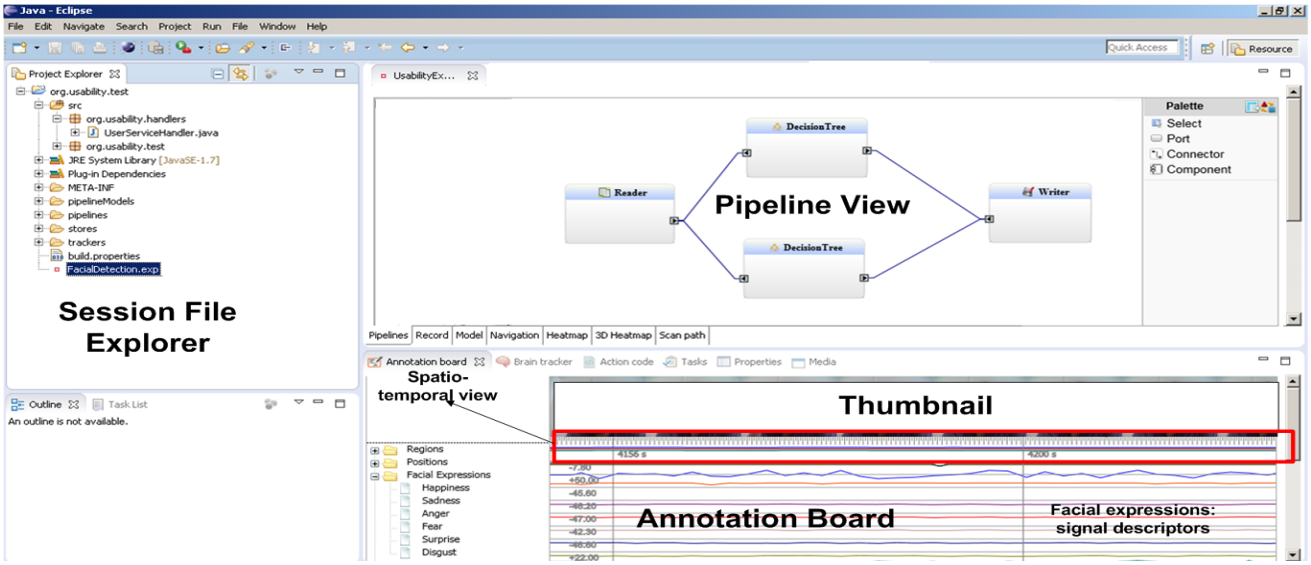


Figure 3-2. Tool Overview

In order to reduce the network traffic, screen capturing and sending to the server only takes place upon changes in the DOM of an application page.

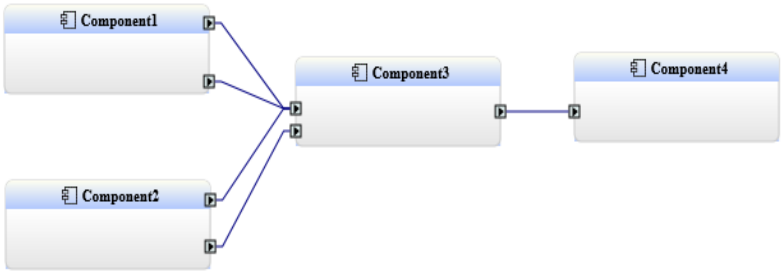


Figure 3-3. A simple pipeline example

3.4 Pipeline

The extension modalities are developed using our pipeline-based approach. The idea is to help users of our tool to make use of predefined components when developing new components. The new components themselves can be used for the development of other components. The information flow among components is defined via ports and connectors (Figure 3-3).

3.5 Annotation board

The traces recorded can be shown in a spatio-temporal representation using the annotation board (bottom right of Figure 3-2).

A spatio-temporal representation means that each annotation entry must have a 1) time span (timestamp start and duration), 2) area of interest region, 3) and annotation description and data.

Each pipeline module can present its interpretation in the annotation board in terms of one entry or more. An annotation entry can be *simple*, or *composite* if it consists of a set of subentries.

The annotation board resolves time into one-second-intervals. Each time range has a start and end in seconds. The end second is determined by the start second and the width of the board ruler (Figure 3-2).

The video-recorded data is displayed as thumbnails in the annotation board, as a way to help link annotated entries and the video data.

In order to deal with the expected massive amount of data captured from different sensors, the annotation board supports lazy loading, such that annotated content will only be rendered upon selecting a range.

An annotation entry renders using a UI descriptor, which includes start and end timestamps. A UI descriptor has width and height. The basic UI descriptors are signal, text, and custom. A user of our tool can program additional UI descriptors.

3.5.1 Signal descriptors

A signal descriptor is used to display data in the form of *discrete* or *continuous* waves. A discrete representation means a data item has one of a fixed list of certain states (Figure 3-4); this can be contrasted with a continuous representation such as [min:-50 to max: 50]. For each curve, there is a separator line indicating the zero point value (a neutral state). Facial expressions can be represented using signal descriptors (bottom right of Figure 3-2).

A compact view shows a list of options for the subentries gathered, such that unselecting an option will remove a subentry from the compact representation. By default, all subentries are checked.

Figure 3-4. An example of a discrete representation

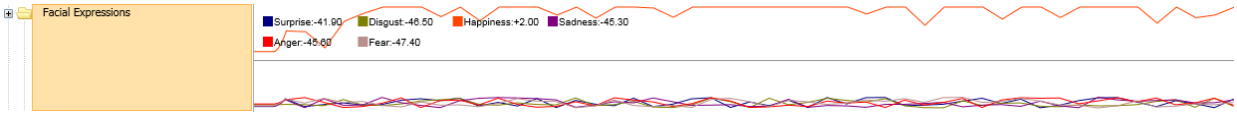


Figure 3-5. An example of a compact entry

3.5.2 Text descriptors

A text descriptor displays the values of an entry as a sequence of text boxes. This sequence is linear: texts on the left-hand side represent textual events that occurred before those to the right of them. Text descriptors are ideal for user interaction events such as mouse and keyboard events, and gaze tracking data, such as eye fixations (Figure 3-6).

Figure 3-6. The rich hover support of a text entry

A text box is supposed to represent events for one second. If some events exist in subsequent seconds, they will be encompassed in one text box. The width of a text box is increased if it encompasses events occurring over several seconds. In such a case, a rich hover can show full information since the size of a text box cannot be adjusted (Figure 3-6).

3.6 Recording module

We implemented a JavaScript recording module to track all user activities. This recording module is a part of the JavaScript file a client adds a reference to in the main of page of the application to be tried on. Alternatively, this recording module can be used internally at the server side.

The recording module follows an event-based paradigm, in which user activities are translated into signals. Signals are encompassed into a compact JSON object, which is sent to the server periodically, by default every 10 seconds (interval); an example is in Figure 3-7.

```
1  {
2          "moseMove": [{
3                  "[11/DIV][1/DIV][1/DIV][2/DIV][1/DIV]": {x: 513, y: 156, timestamp: 1420249686370},
4                  "[11/DIV][1/DIV][1/DIV][2/DIV][1/DIV]": {x: 519, y: 156, timestamp: 1420249686170}
5          }],
6
7          "moseDown": [{
8                  "[11/DIV][1/DIV][1/DIV][2/DIV][1/DIV]": {x: 519, y: 156, timestamp: 1420249686170}
9          }],
10
11         "keyDown": [{
12                 "[11/DIV][1/DIV][1/DIV][2/DIV][1/DIV]": {keyStroke: "A", timestamp: 1420249686170}
13         }],
14
15         "paste": [{
16                 "[11/DIV][1/DIV][1/DIV][2/DIV][1/DIV]": {contents: "Some pasted contents",
17                         timestamp: 1420249686190}
18         }]
19 }
```

Figure 3-7. User interactions as JSON

```
1  <html>
2  <body>
3  <div style="border-color: red; border-style: dotted;">
4          First
5  <div style="border-color: orange; border-style: dotted;">Sub First1</div>
6  <div style="border-color: orange; border-style: dotted;">
7            Sub First2
8  <div style="border-color: blue; border-style: dotted;">Sub sub First2</div>
9  </div>
10 </div>
11 <div style="border-color: green; border-style: dotted;">Second</div>
12 </body>
13 </html>
```

Figure 3-8. An HTML example

For an event, we need a unique identifier for its control (element), which is ideally the id attribute. However, many controls may not have the id attribute defined, since it is optional. In order to overcome this issue, we use the full path of a control as a unique identifier. For example, the path of the div element in Line 5 in Figure 3-8 [1/HTML][1/BODY][2/DIV][1/DIV]. We use the index of a child if the parent has multiple children. For instance, [2/DIV] means that this div is the second in the child list of its parent (Figure 3-9).

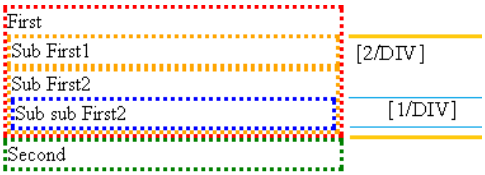


Figure 3-9. The rendering of Figure 3-8

The content of a signal depends on the JS event recorded; the core events include mouse, keyboard, and location change events.

In mouse events, signals include the mouse location, element path, mouse event type, and event time. A mouse event type has several types such as mouse down, mouse move, and mouse up.

In keyboard events, signals include keystroke, element path, and event time. A keystroke captured includes additional information such whether control and alt keys are used.

Capturing paste events requires special handling because retrieving clipboard data is not supported by all browsers. For each control, we create a lazy tracker that only activates when the control is focused for the first time. Upon activation, this tracker will listen to all text changes applied to the control. Thus, it can get the differences before and after a paste event. The differences represent the clipboard data pasted.

3.7 Related work

Existing tools do not directly link user emotional experience and interactions, while collecting observational data. A tool either focuses on evaluating user experience or data observation.

Tools that focus on evaluating user emotion experience usually target videos used for advertisement; these include Emotient [40] and Affectiva [41]. In Emotient, the focus is on facial recognition to extract the six basic emotions [57], while in Affectiva [41], they extract the mental states of users while watching videos.

The tools that focus on data collection include Noldus Observer XT [42] and IMOTIONS [43]. Noldus framework allows observing data via hardware they developed. The solutions that Noldus provide also include face reader that is used to analyze emotions.

Noldus Observer XT and IMOTIONS use many hardware and software elements to collect different types of physiological inputs, such as ECG, EMG, and EEG, as well as eye tracking and facial expressions. IMOTIONS' observer is used by other tools such as Emotient [40] and Affectiva [41].

Chapter 4 Pipeline Architecture

In this chapter, we show how we implemented a pipeline architecture, such that the development of psychophysiological sensors and interaction modules became easier. This is a crucial part of our research, since we need to implement a large number of psychophysiological, classification, and analysis components. The psychophysiological components implemented will be discussed in detail in Chapters 5, 6, and 7.

We select a pipelined approach as pipelines can enforce a high level of abstraction using a number of components interacting independently and asynchronously. The execution of a pipeline depends on its components and attributes. An executable pipeline allows users of our application to reuse components developed by experts from different fields in a way that does not require those users to be familiar with those fields in order to reuse pipeline components.

We use Umple, an open-source model-oriented programming language, for the implementation of components, communication, and validation rules. We participated actively in the development of the aspects of Umple used in this chapter, although that is a body of work separate from this thesis [100].

In this chapter, we assume that the readers have fair knowledge of software architecture, such as pipelines and component-based diagrams.

4.1 Introduction

The concept of pipeline, or scientific workflows, has been around for a long time. Many software frameworks, such as RapidMiner [101], SPSS, and Knime [102], support development using explicit pipelines.

The motivation behind our research is to show how the concept of pipeline diagramming can be used with modelling specifications such as components, attributes, and ports.

We use Umple for the underlying implementation of our framework [100]. We rely on many features Umple provides such as composite structure, state machines, and class diagrams as well the code generation of different target languages such as Java and C++.

Components in the context of component-based modelling in Umple are reusable classes whose instances have their own threads of control and interact via ports and connectors.

In Umple, a class is considered a component once it has at least one port or connector. In this chapter, we will only refer to classes that are also components.

Our main contributions in this chapter are as follows.

- We have developed a framework that aligns pipelines with composite structure, class, and sequence diagrams.

- Our framework dynamically generates forms for the attributes that exist in the pipeline components. The generated classes can have validation rules
- We implemented a communication bridge that allows using libraries of different languages among components.
- We worked on many use cases to validate how our pipelined framework can advance development. We show case studies on this in Chapters 7, 8 and 9.

The diagrams that we will show in this chapter are rendered using our pipeline editor.

4.2 Related work

Scientific workflows have been adopted in various fields such as bioinformatics, economics, mathematics, and data mining [103]. A scientific workflow has a lattice of computational steps where data is passed among components.

Tools such as RapidMiner [101], SPSS [104], and Knime [102] support workflow execution with graphical capabilities. Other tools such as R script [105] and Weka [106] provide primitive workflow support. Additional tools in the fields of data mining and analytics [107], include Knime, SPSS, and SAS JMP.

Some tools are targeted for specific fields, such as R, which is used more in statistics, and Weka, which is used specifically in data mining.

Tools such as R and Weka can be used as extensions with other tools such as Knime.

Node-RED is a browser-based flow tool developed by IBM that is targeted to the Internet of Things (IoT) [98]. It is capable of linking different online services together over the web. However, it does not easily integrate with computer vision and machine learning work.

LabVIEW is a data flow tool targeted to several types of engineers, including real time system developers [99]. But like Node-RED, it is not easily adapted to integrate with computer vision and machine learning tools.

Most of existing tools that support workflow do not provide a script-based approach to add new extensions or capabilities. Even tools such as Knime, which supports extensibility, requires a high level of Java knowledge, and the ability to develop the new UI controls required.

4.3 Pipeline architecture

In our framework, a pipeline consists of *chains* of components. The flow among components forms a Directed Acyclic Graph (DAG), which describes their task execution sequence. A component is a specialized type of class that has its own thread of execution. This means that executing a component does not block other parts of the pipeline, and hence enables concurrent execution.

A pipeline is initiated using a *composite class* encapsulating the pipeline as a whole. To add a component to a pipeline, one adds a class to the *composite structure* of the top-level class.

A component has a state of execution. Table 4-1 summarizes the possible states of a component. In the next section, we will give more details about how a component's state changes

4.3.1 Communication

In accordance with the concepts inherent in composite structure [108], a component can send or receive signals via ports.

A port type in Umple can be *in*, *out*, or both. *In* ports are designed to receive signals from other components, and *out* ports are designed to send signals to other components. In our framework, we have an additional specialized *out* port, which is an error port that is designed to handle exceptions and fault signals.

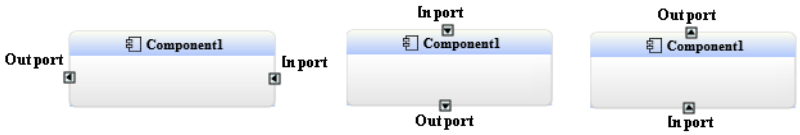


Figure 4-1. The orientation of a port symbol

Diagrammatically, the in and out ports can be differentiated from the direction of their symbols as in Figure 4-1. On the other hand, an error port is shown as an error icon as in Figure 4-5.

Table 4-1. States of a component

States	Description
Idle	The start state; a component is not waiting for any execution request.
Configured	The component is ready for execution and does not have any validation issues. A component becomes configured when required attributes have no validation issues preventing the execution.
Pre-execution	The component is about to execute, but must pre-process the inputs. The connected components will be notified when a component enters this state.
Executing	The component is processing the data received from its inputs.
Interrupted	During the execution of a component, it can be interrupted. When an executing component receives an interruption request, it terminates its execution and enters the interrupted state.
Queued	If the execution of a component relies on the execution of other components, this component itself will be put into a FIFO queue.
Post-execution	It is used as a notifier to inform other components when a component has just finished execution.
Failed	The component has encountered execution errors.

4.3.2 Execution sequence

The execution of a component in a chain depends on the signals received from previous components in that chain. For instance, the execution of Component3 in Figure 4-2 will not start before getting results from Component2. This is the same for Component2, which will not execute before getting results from Component1.

Figure 4-2. A chain of components

A component can receive signals from multiple ports in the same or different components. For instance, Component 4 in Figure 4-3 receives signals from two components Component2 and Component3. The signals received from both components are propagated via multiple ports.

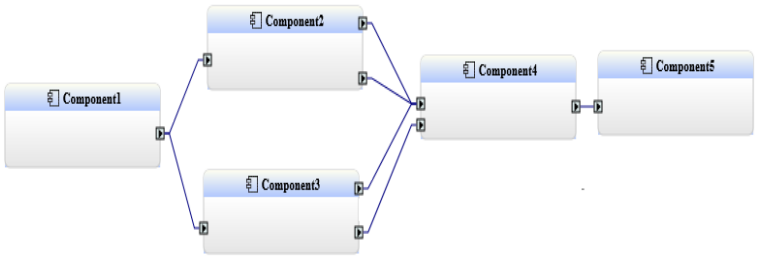


Figure 4-3. Signals propagation from multiple sources

While a component is waiting for the execution of other components in a chain, it will be put into a FIFO queue. The FIFO queue is mainly used to organize signal propagation. For instance, signals from Component1 are propagated to Component2 and Component3. If Component2 was put first in the FIFO queue, the signals will be propagated to it before Component3.

A port or component can additionally have a priority value. When a port or component has a higher priority value than others, it will receive signals before them even if this port or component was not the first in the FIFO queue.

By default, the priority value of a port is set to zero, so all will be treated the same in the FIFO queue. Components of the higher priorities are executed first. Within a component, ports of higher priorities will propagate signals to other components first.

In Figure 4-4, the port priorities are in red and the component priorities are in green. Only the "out ports" have their name shown. Table 4-2 shows the sequence of execution of Component5. For each stage, Table 4-2 shows two columns for the list of the enqueued components and ports; both columns are ordered based on their priorities.

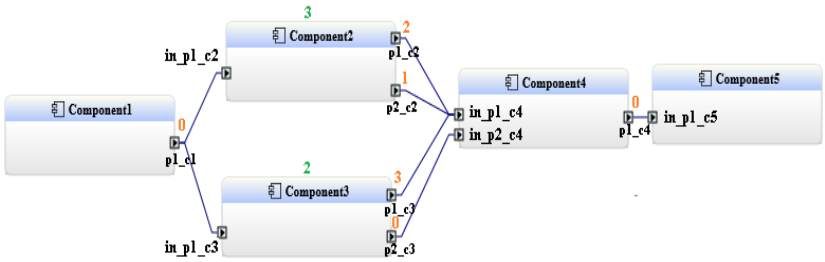


Figure 4-4. Sequence of execution and priority

If the number of stages is K and the current stage is N, the components to be executed will be listed in the stages between 1 and N-1, while the components to be enqueued will be listed in the stages between N+1 and K. For example, at Stage 3, the components to be executed will be Component1 (from Stage 1) as well as Component2 and Component3 (from Stage 2). On the other hand, the components to be enqueued will only include Component5 (at Stage 4). The component to be executed at Stage 3 is Component4.

Table 4-2. An example of sequence of execution based on priority

Stage 1		Stage 2		Stage 3		Stage 4
Components	**Out**	Components	**Out**	Components	**Out**	Components
Component1	p1_c1	Component2	p1_c2	Component4	p1_c4	Component5
			P2_c2			
		Component3	p1_c3			
			p2_c3			

4.3.2.1 Error handling

Upon an execution failure, error signals are propagated via *error* ports.

In Figure 4-5, when the execution of Component2 fails, error signals will be sent to ErrorComponent. ErrorComponent acts as the starting entry of a spinoff chain as it will continue another execution sequence to reach the final component, Component3.

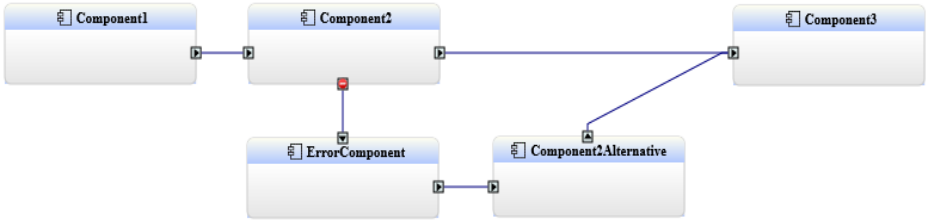


Figure 4-5. A spinoff example of a component pipeline

An error component can send signals back to the component that originally had an issue. For example, in Figure 4-6, ErrorComponent will receive the error signals, normalize the inputs properly, and then send back the normalized signals to Component2.

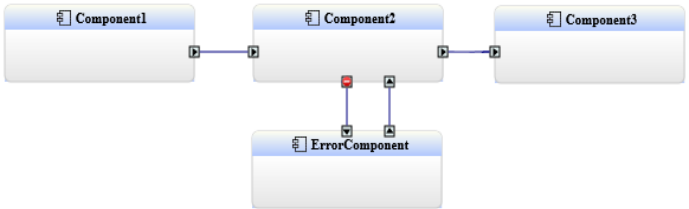


Figure 4-6. An example of an error component normalizing signals

4.3.2.2 Execution behaviour

When executing a component, signals are propagated to successive components until reaching the final components in the chain. The execution of a component may depend on signals from previous components. In such a case, executing this component will require executing previous signals. For instance, executing Component2 in Figure 4-4 will require executing Component1 first.

A component being executed has a progress status that represents the completeness, warnings, and errors of execution as a percentage.

Figure 4-7 shows an example of the execution of Component5. The execution of Component3 has some warnings coloured in yellow; however, this did not prevent propagation of signals to the next component in the chain. Component5 has an error that caused execution to fail. If a previous component in the chain such as Component 2, failed, this would have prevented the execution of Component4 and Component5.

Figure 4-7. Execution with warnings and errors

4.3.2.3 Metacomponents

A metacomponent is a pipeline that can be reused by other pipelines. We will refer to the internal content of a metacomponent as the pipeline of this metacomponent, although technically they are the same.

When adding a metacomponent, its internal content will be treated as a black box. The communication with the internal content of a metacomponent is done by the exterior ports of this component's pipeline. Exterior ports refer to the ports at the boundary of a pipeline. Figure 4-8 highlights the exterior ports of a pipeline. When using this pipeline as a metacomponent, the highlighted ports will be added to that instance as in Figure 4-9.

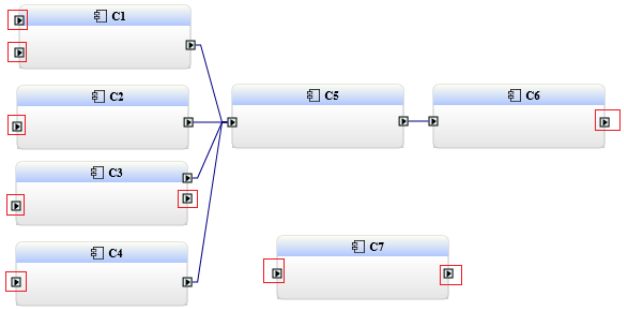


Figure 4-8. A pipeline exterior ports

Figure 4-9. A pipeline as a metacomponent

4.4 Umple features supporting pipelines

In this section, we describe how we use Umple for the development process in our framework.

Each component in a pipeline is written as an Umple model or class in an Umple model. An Umple model can be generated into its equivalent code based on a selected target language. In our research, the selected target languages are primarily Java and C++.

Generated code of an Umple model is designed to execute independently, i.e. without dependencies on external libraries. This helps us develop modules that are independent and easy to integrate with other components with less chances of dependency issues.

The Umple classes utilized by our framework are designed to extend a built-in Umple class, BasePipelineComponent that we implemented to provide common functionality.

BasePipelineComponent is a *lazy* class, which means, in the Umple context, that its attributes are not passed in the constructor; instead, a client will need to set the value each attribute in a single call.

The communication among BasePipelineComponent instances is designed to propagate data in JSON.

The core functions of BasePipelineComponent are "receive" and "send" in addition to several state machine methods that handle the possible states of a component such as idle, executed, and postExecuted.

Each state method makes a call to a function that has empty implementation in BasePipelineComponent. For example, in the case of "execute" state, this function is "onExecute", in the case of "postExecute", this function is "onPostExecute", and so on. We will refer to any of these functions as a client state function or simply a state function.

In the context of our framework, clients are expected to override the state functions, specifically "onExecute", as it will contain their actual component implementation.

There are three parameters in any state function, request, response, and progress. The parameters "request" and "response are of the type "ProxyObject", and the parameter "progress" is of the type "Progress". Both "ProxyObject" and "Progress" are built-in Umple classes.

The main functions of "ProxyObject" are "set" and "get". The "set" method typically receives two parameters, JSON identifier and its value. Alternatively, "set" can receive a single parameter. This is the case if a client wants to

propagate a simple string value directly; we refer to this value as a simple entry. The "get" method is used to retrieve a value of a JSON identifier, which is passed in as its parameter. When "get" is called without parameters, it will retrieve the simple entry of ProxyObject.

On the other hand, "Progress" is used to help clients set the current progress of execution. The main functions of "Progress" are "set," "increment", and "get". For simplicity, we explain different variations of the "Progress" methods as comments in Snippet 4-1.

When a client implements a state function, it should work on assigning the appropriate values to the "response" object. The response object will be propagated to the components next in the chain.

The "request" object holds the content of the "response" object received from a previous component.

In our framework, when a user selects a component, the action code of this component will enable writing the model content of this component in valid Umple syntax. The code snippet assumes that the user is writing the content of an Umple class. Snippet 4-2 shows a simple example, in which two state functions are implemented (Lines 4 and 11).

```
1  //Set the progress to 20%
2  progress.set(20);
3
4   //Increment the progress by 60% so it will be 80%
5   progress.increment(60);
6
7  //Reset progress
8   progress.set(0);
9
10 //Report a warning that allocates 5% of the progress
11 progress.set("warning", "this is a warning message" , 5);
12
13 //Report an error that allocates 5% of the progress
14 //Regardless the other progress, any error causes the
15 //component execution to fail
16 progress.set("error", "this is an error message," , 15);
17
18 //This will void the error reported above, so the execution
19 //will not fail
20 progress.set("error", "0");
21
22 //This will print out the list of warnings reported
23 System.out.println("progress.get("warning"));
24
25 //Complete progress, to notify the next components
26  //in the chain.
27 //A call for progress.set(95) would be the same as there is
28 //5% allowed for the warning set earlier.
29  progress.set(100);
```

Snippet 4-1. An example of using "Progress" methods

Typically, there is no need to write a complete Umple model. Our framework will automatically create an Umple model with a class named after the component selected. After that, this model will be generated and executed. This

complete model, for instance, will force the class written to extend BasePipelineComponent, and import ProxyObject and Progress.

```
1   String first_name= "First";
2   String last_name= "Last";
3
4   void onExecuting(ProxyObject request,
5           ProxyObject response, Progress progress){
6       response.set("A simple entry");
7       response.set("Full Name: ", first+ ""+ last);
8       progress.set(100);
9   }
10
11  void onExecuted(ProxyObject request,
12          ProxyObject response){
13      System.out.println("Executed");
14  }
```

Snippet 4-2. A simple code example

If a client wants to have more options such as creating internal classes or let their class extends other classes, they can alternatively write a complete Umple model instead of a class portion as shown in Snippet 4-2. In such a case, the client must make sure that there is a class named after their component selected; in addition, they have to ensure that they import the required built-in functions such as ProxyObject and BasePipelineComponent. Snippet 4-3 shows an alternative example of Snippet 4-2, in which a complete model is written instead; the assumption is that the component name is "ComponentTest"

```
1   depend ProxyObject;
2   dependBasePipelineComponent;
3   class ComponentTest{
4       isA BasePipelineComponent;
5       String last_name= "Last";
6       void onExecute(ProxyObject request,
7               ProxyObject response){
8           response.set("A simple entry");
9           response.set("Full Name: ", first+ ""+ last);
10          progress.set(100);
11      }
12
13      public void onExecuted(ProxyObject request,
14              ProxyObject response){
15          System.out.println("Executed");
16      }
17  }
```

Snippet 4-3. A simple model example

Transferring objects among components is handled using the composite structure features of Umple [100]. The assumption is that components do not necessarily exist in the same application or execution lifecycle. Therefore, data is transmitted among components as signals. When data to be transmitted is of simple types such as those shown in Snippet 4-2, transmission will be straightforward. For complex data types, Umple marshals data into an

intermediary form that can be transmitted as signals. Once the intermediary object is completely marshalled and received by a component, it will be unmarshalled back into its original form.

A pipeline in our framework is stored as a JSON file. This JSON file has the list of Umple classes and models the pipeline uses.

4.4.1 Form representation and validation rules

In our framework, a form is generated dynamically for each component based on the public attributes defined in an Umple class. For instance, Boolean attributes are represented as checkboxes, string attributes are represented as textboxes, and enumerations are represented as selections. If an attribute has a value, this value it will be considered the default when opening the form.

In terms of validating attribute values, Umple helps users set validation rules on their attributes.

A simple form example is shown in Figure 4-10; the Umple code portion is shown in Snippet 4-4. There is a constraint set on the attribute "age" in Line 3.

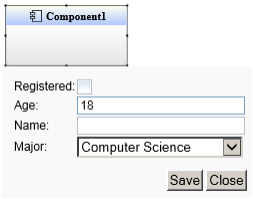


Figure 4-10. A form example for a component

```
1  Boolean registered= false;
2  [age>=18]
3  Integer age= 18;
4  String name;
5  Major {computerScience, engineering};
6  Major major= Major.computerScience;
```

Snippet 4-4. An example of attributes

4.5 Execution mechanism and language bridge

Umple allows developers to use the capabilities of the target language selected such as C++ and Java. In many cases, a developer still may need to use libraries implemented in the target language selected. For that, we allow developers to configure Jar or DLL files. After that, a component can be created to allow using the APIs existing in these. The communication among components is socket-based.

The essence of this approach is to allow using libraries of different languages; e.g. a language bridge. For instance, a component can use Weka library, and another component can use Open CV library.

4.6 Use case

In this section, we show a Weka wrapper use case, to demonstrate how our pipeline framework can help developing classifiers that use external libraries. This can be helpful in domains, in which classification and analysis of data captured form different modality is essential.

Weka is one of the most popular machine learning and data mining software tools [106]. Therefore, reusing the features provided by Weka can be beneficial in many ways. Using our language bridge (Section 4.5), we created an interface to access APIs of the Jar file of Weka. Components can use this interface in order to use Weka APIs such as decision trees, clustering, and prediction

Figure 4-11 shows the pipeline of a decision tree. The DecisionTree component makes direct calls to the decision tree API of Weka. An overview of the code body of DecisionTree is shown in Snippet 4-5.

```
1  public void onExecuting(ProxyObject request,
2              ProxyObject response, Progress progress){
3     //This is a call to a JNI interface function
4    decisionTree(progress, respone,
5           classifyColumn,  numberRecords,
6           reducedErrorPruning, pruningMethod,
7          qualityMeasure,  minimumNodes,
8          splitAverage, binaryNominalSplits,
9          maxNominal, numberOfThreads);
10
11    //Notify other components that execution is complete
12    //The results of execution is already set to the /"response" object after the call to "decisionTree"
13    //If there is an error, it will be reported in "progress", which will cause execution to fail
14    progress.set(100);
15 }
```

Snippet 4-5. A decision tree example

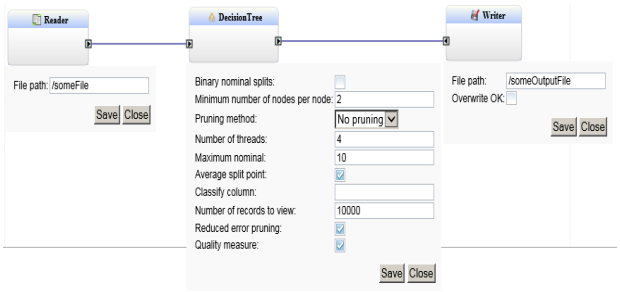


Figure 4-11. A Weka example

Chapter 5 Adaptive Probabilistic Patch Active Appearance Model

In this chapter, we show how we implemented efficient and effective fitting techniques based on Patch Active Appearance Model (PAAM), an optimization of AAM. This allows for emotion extraction from facial expressions, which is a central part of this research. Our implementation depends on Active Appearance Model (AAM) [4], which we will use later for other key parts of this research, such as gaze tracking.

Shape and texture statistical models such as AAM are commonly used for image alignment and fitting. AAM is highly effective, but it is often criticized for being less efficient than alternatives. Numerous algorithms have been proposed to improve the efficiency of AAM fitting, in particular, Inverse-Compositional Image Alignment (ICIA) [2], [120]. The ICIA techniques however can have degraded performance in the presence of objects that have high appearance variation such as faces. As a way to overcome this limitation, we propose PAAM with HOG (HOG-PAAM), which employs Histogram of Oriented Gradients (HOG) features.

We introduce two fitting optimization techniques, Adaptive Probabilistic (AP), and Adaptive Probabilistic Project-Out (APPO). Both techniques estimate Gaussian noise to precompute a fixed estimate for appearance-optimal parameters. They differ in that APPO applies an additional step to project out appearance variations. We show that our approach, although using AAM, is effective, time-efficient, and space-efficient for facial alignment and fitting, as compared to techniques such as simultaneous, alternate and Bayesian AAM algorithms.

We compare our approach to existing AAM techniques such as AIC [5], POIC [2], and SIC [2]. We compare all techniques in terms of recognizing affective states, to validate the performance of fitting facial expressions.

In this chapter, we assume that readers have good knowledge of machine learning, as well as aspects of computer vision such as shape, texture, and appearance models. The core terms and techniques referred to in this chapter are defined briefly in Glossary.

5.1 Introduction

Many computer vision studies focus on the construction of deformable models because of their importance to object detection, recognition, and tracking. Examples of deformable models include Active Contour Model (ACM) [3], Active Shape Model (ASM) [4], and Active Appearance Model (AAM) [4].

For many years, Active Appearance Model (AAM) has been one of the most commonly used techniques for constructing deformable models [109], [110]. Coots et al, originally introduced AAM as an extension of ASM and ACM [4].

AAM is a generative model, meaning that it consists of statistical models used to capture and learn from variations of the shape and appearance of an object. Then, it generates instances of this object based on the learnt model parameters.

A statistical model of a shape is constructed from a fixed number of points or landmarks that annotate an object of interest. It uses statistical techniques such as Generalized Procrustes Analysis (GPA) to remove global similarity transforms, and limit shape point variability to a non-rigid deformable shape.

A statistical model of an appearance is built from shape-free texture images. Each of these is a normalized image of an object with respect to its associated annotated shape points. It uses a canonical *reference frame* such as the mean shape, and *motion* or *wrapped parameters* $W(p)$, such as Piece-Wise Affine (PWA) or Thin Plate Spline (TPS).

An appearance model helps to capture texture space variation, but there is a trade-off in terms of efficiency, model size, and performance with respect to the number of eigentextures.

Reducing the number of eigentextures can lead to performance degradation, but it improves the speed and size of the model (see results in Section 5.6). Patch Active Appearance Model (PAAM) was introduced to optimize fitting by reducing texture space, and only retaining regions of textures around shape model points [111]. It can be done by applying a local transformation around these shape points such as patching.

Landmark localization or alignment is crucial to solve computer vision problems such as facial feature detection. Fitting is used to estimate shape parameters, which are used to locate landmarks of a specific object in unseen images. It is an iterative process, which means that shape model parameters are initially estimated, and then iteratively updated. Updates are typically based on an error function such as the least squares method that calculates the differences between an image and the reference frame of the model.

Fitting is a non-linear optimization method, therefore, it has two main drawbacks 1) approximation could be impractical, and 2) Jacobian and Hessian computation can be expensive. There are two main different approaches to optimize the fitting: *regression*-based and *NLS*-based (*non-linear least squares).*

A *regression*-based technique directly learns to map appearance features to incremental approximations, which update the shape model parameters. Hence, it does not require learning from shape and appearance models. Recent related work to train non-linear regressors is based on discriminative methods or boosting techniques [112], [113].

Non-linear Least Squares (NLS)-based techniques are represented as Lukas-Kanade (LK) problems, which depend on Gauss-Newton gradient descent optimization [114] to iteratively solve model parameters. However, as compared to regression-based methods, gradient descent methods can be inefficient when used with AAM [2]. When AAM applies a gradient descent method, the fitting become sensitive to appearance variation increases such as illumination, pose, and occlusion.

LK has different settings for how to minimize the NLS error between a wrapped input image and a model instance with respect to shape and texture parameters. LK was proposed with a forward compositional setting, but later it was changed to Inverse Compositional (IC) [2] to improve efficiency; an example is shown in Figure 5-1.

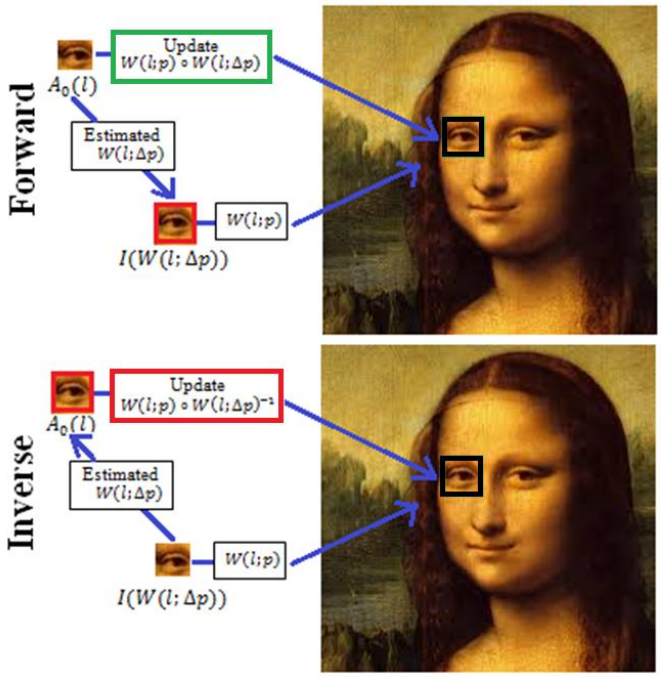


Figure 5-1. Forward versus inverse compositional. Adapted from [2]

The green-bordered box is for the forward settings, and the red-bordered box is for the inverse settings

Recent *NLS*-based techniques include probabilistic and adaptive, as well as IC versions of these. Such techniques work on improving the efficiency by reducing the cost of solving appearance optimal parameters, and the computation of Jacobian and Hessian matrices.

Probabilistic approaches use a Bayesian formulation to reduce the texture space and appearance variation sensitivity [115]. Adaptive-based approaches inexpensively precompute mean texture template and estimated update parameters to retain accuracy and improve efficiency.

IC is commonly used for fitting in AAM [116], [117], since it places wrap images, and estimate wrap updates on the model side, which can be pre-computed before starting fitting iterations. This is especially true with Project-Out IC (POIC) algorithms, which precompute and project-out appearance variation before fitting [2]. Projecting out refers to reducing the parameter matrix size after removing image or texture variation.

As a result, a POIC algorithm can be as efficient as regression-based methods, but accuracy is degraded, as it solves wrap image appearance parameters, and computes Jacobian and Hessian on the model side on its way to improve efficiency.

Descriptive texture features such as Histograms of Oriented Gradient (HOG) [46] and Scale-Invariant Feature Transform (SIFT) [118] have shown an improved accuracy for building unsupervised and robust AAM.

AAM has several disadvantages. For instance, in terms of handling variations, AAM can perform effectively if it is trained against certain subjects; however, this will be a problem if subjects are unknown. When variation increases, regression becomes more challenging due to the difficulty to differentiate between shape and texture parameters. Some factors such as occlusion and pose are not intuitive to handle. Reaching both effective and efficient solutions is hard to achieve due to the high complexity of the cost functions.

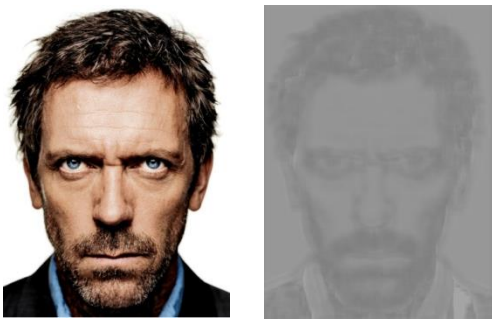
In this chapter, we investigate how to obtain both effective and efficient fitting even when using AAM.

Our approach and key contributions in this chapter can be summarized as below.

1) We use HOG as a feature extractor with PAAM to estimate facial features from a given image. Toward this, we employ densely-sampled HOG in training to increase effectiveness. To the best of our knowledge, this is the first time that PAAM is developed using HOG features. Results that are more efficient can be reached with PAAM due to two main reasons. First, in PAAM, holistic texture features are not considered. Second, PAAM focuses on certain features of objects.
2) In this chapter, we will show a comparison between HOG-AAM and HOG-PAAM using state-of-the-art methods in facial features detection.
3) As a way to reduce the complexity of the AAM appearance cost functions, we implement novel adaptive probabilistic formulations, in which a Gauss-Newton estimator is employed in order to precompute a fixed and inexpensive wrap image texture template to solve appearance optimal parameters, and remove appearance variation from texture space.
4) We introduce two adaptive probabilistic-based formulations, which are Adaptive Probabilistic (AP), and Adaptive Probabilistic Project-Out (APPO). In our discussion, we will show the differences between AP and APPO, especially in terms of effectiveness and efficiency.
5) We show that our AP and APPO approaches can outperform state-of-the-art methods in facial alignment such as Alternating IC (AIC) [116], Project-Out IC (PIC) [2] [4], and the Simultaneous IC [109].

5.2 Histograms Of Oriented Gradients (HOG)

We employ histograms of local edge directions to describe the distinctive image features for shape and texture models. They are also characterized by local geometric and photometric invariance, which makes HOG suitable for deformable models. An HOG descriptor is an extraction function that aims to localize sub-windows, and cluster gradient orientations of an input image [46],[47].



(a) original (b) Dense HOG

Figure 5-2. Illustration of a dense HOG features

Channels are averaged and brightness is changed for better features visualization.

A given image of a size ($H \times W$) is vectorized and denoted as i, so i is a vector form of a length (L_I). Then, the feature extraction function (hg) of HOG starts with the computation of image gradients. In such a case, if the image is RGB, we have a channel for each colour gradient. The channel gradient that has the largest norm is kept. At each pixel of an input image (i), the HOG function ($\mathcal{H}$) extracts descriptors D.

For a vector-based image (i) of a length (L_I), a descriptor ($\mathcal{H}$) is represented as

$$hg = \mathcal{H}(i) \tag{5-1}$$

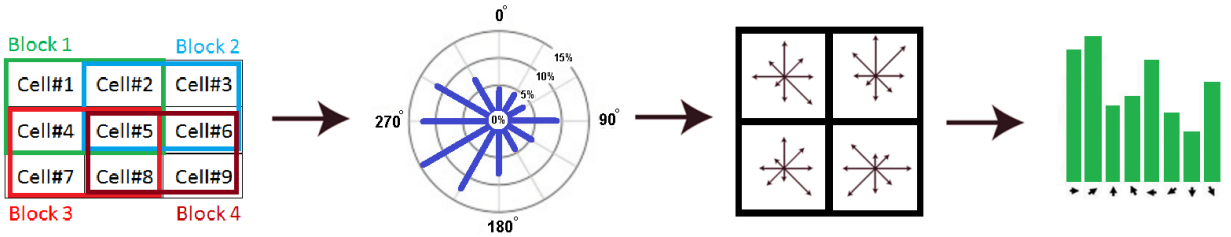
$$\mathcal{H}: \mathbb{R}^{L_I} \rightarrow \mathbb{R}^{L_I D} \tag{5-2}$$

HOG applies spatial neighbourhoods and orientation binning for each pixel of an image. Neighbourhoods (N) are two dimensional representations, and can be cells and blocks Figure 5-3 (a). A cell (N_{cell}) is a subwindow of pixels, on which HOG is created. A block (N_{block}) represents a number of cells and contains ($N_{block} \times N_{block}$) cells. Cells can be overlapped with their neighboured blocks. A local contrast normalization is applied on each block, to make image features invariant to the intense differences of illumination, and other variations.

A cell is used to compute the magnitude of gradient for each pixel. While, the orientation binning (N_{bin}), Figure 5-3 (b), quantizes the gradient into a number of bins (0-180) and applies a trilinear interpolation between (N_{bin}), which represents theirnonlinear aspects. From there, we create a histogram of oriented gradient, Figure 5-3 (d), which has N_{bin}. After that, we build feature vectors of the histogram of cells and their blocks. A descriptor (D), Figure 5-3 (c), has a length (L_D) as:

$$L_D = N_{bin} N_{block}^2. \tag{5-3}$$

In this chapter, we work on different parameters, cell sizes$N_{cell} \in \{4, 8\}$, cell blocks $N_{block} \in \{1, 2\}$,and $N_{bin} = 9$.



(a) Cell and blocks (b) Orientation Binning (c) Descriptor (d) Histogram

Figure 5-3. HOG illustration

5.3 Probabilistic PCA (PPCA) and ZCA whitening

AAM mostly relies on PCA to build statistical models of shape and texture. PCA can be either deterministic or probabilistic; in this section, we will outline the probabilistic version, with respect to Zero-Phase Component Analysis (ZCA). One of our contributions is to use a probabilistic version with ZCA transformation functions such as whitening [119].

PCA processing is time-consuming [115], [120]. In PCA, a k-d tree [17] is used to speed up the optimization process and the extraction of the eigenvectors. PCA analyzes texture to find orthonormal bases (U) that represent the latent space (C) projection of the mean-texture training set (S). The number of shapes and pixels are denoted by n and m respectively. Both "n" and "m" can be written in upper case in order to represent the complete sets. For example, M represents the set of all the eigenvalues, while m represents the *number* of eigenvalues; e.g. $m = |M|$.

Texture data features are represented in a $l \times hg$ matrix, where hg represents feature vector and l landmarks. Then, we obtain a covariance matrix (C) that has eigenvectors (E) and eigenvalues on the diagonal (B).

$$C = EBE^T \tag{5-4}$$

PPCA defines a generative model where W_z and m are learnt using an Expectation Maximization (EM) algorithm. It samples the observed data to compute the mean vector m and sample noise error ε (an isotropic Gaussian approximation). On the other hand, W is the principal space, which uses the learnt latent variables z to generate an observed sample y.

$$y = W_z + m + \varepsilon \tag{5-5}$$

$$z \cong N(0, Identity) \tag{5-6}$$

$$\varepsilon \cong N(0, \sigma^2 \cdot Identity) \tag{5-7}$$

Zero-phase component analysis (ZCA) was first used in Independent Component Analysis (ICA) as a way to minimize data transformation as close as possible to the original basis functions [119]. Whitened data remains whitened upon rotations. In the least squares sense, the results of whitened data are very close to the original data.

The PCA whitening transformation (W_{PCA}) is calculated by adding a small value (ε) to the eigenvalues:

$$W_{PCA} = \sqrt{B^{-1} + \varepsilon E^T} \tag{5-8}$$

As opposed to PCA, whitened ZCA images have resemblance to original images. In ZCA whitening (W_{ZCA}), eigenvectors (E) of the covariance matrix (C) are used as an orthogonal matrix. The value of (ε) is estimated by the sampled noise variance (σ^2) obtained during training, which strengthens and makes edges more visible (Figure 5-4):

$$W_{ZCA} = E\sqrt{B^{-1} + \sigma^2 E^T} \tag{5-9}$$

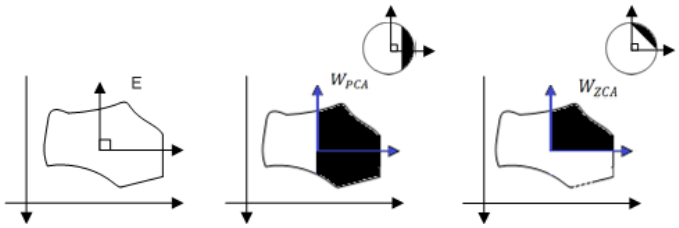


Figure 5-4. PCA versus ZCA whitening transformation

5.4 Patch Active Appearance Model (PAAM)

In this section, we start to present our new contributions. PAAM defines shape and motion models as well as texture and appearance patches of deformable objects of an image.

A fitting strategy of PAAM aims to reduce texture space information, and optimize wrapping functions without affecting accuracy. This is using patches that have oriented and fixed sizes.

Figure 5-5. An example of PAAM Texture Mean

5.4.1 Training

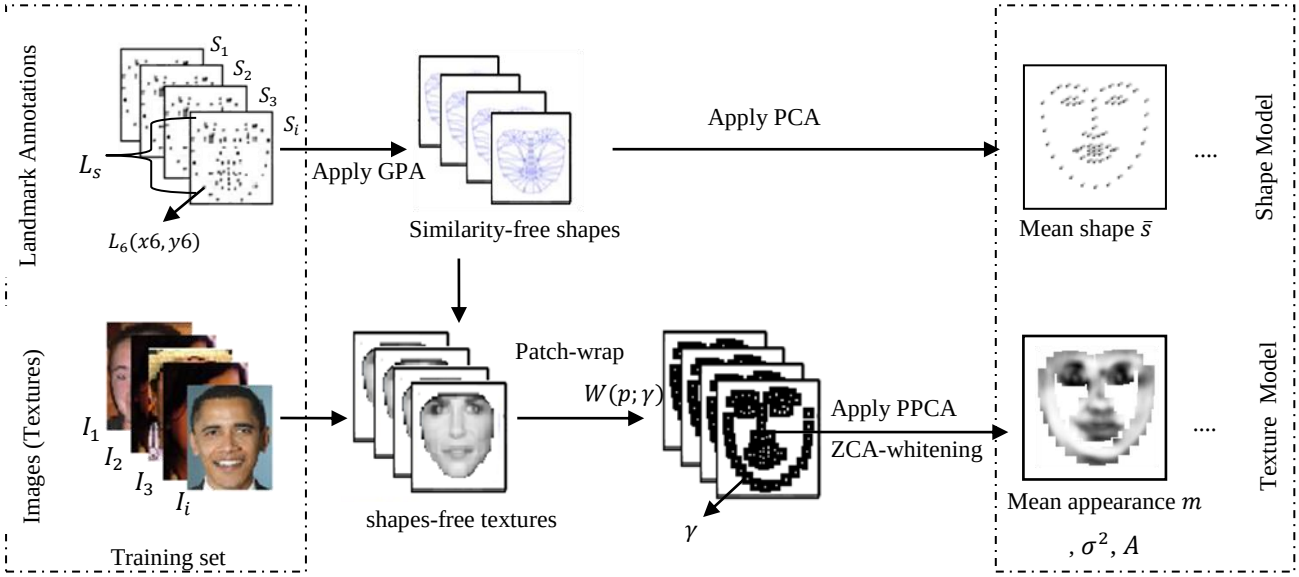


Figure 5-6. PAAM training illustration

Our training process (Figure 5-6) uses a set of manually annotated training shape instances $\{S_i\}$. In a training image (such as faces), $I_i(x, y) \in R^N$, a shape instance (s) has a set of consistent landmarks L_s, such that $s = [x_1, y_1, \dots, xL_s, yL_s]^T$. Ashape instance can be represented as$S_p = \bar{s} + U_{sP}$; p represents the number of parameters.

The motivations behind the training process in PAAM include 1) building shape, texture, and motion models that eliminate shape variations and provide similarity textures, and, 2) providing every test image an initial shape and number of texture vectors.

A shape model is defined on a structure by obtaining a mean shape $(\bar{s})$ and orthonormal basis of shape eigenvectorsN_s, such that $U_S \in R^{2L_S \times N_S}$.

Our texture model is defined by obtaining the mean appearance $(\bar{a})$, appearance eigenvectors $\{A_0, A \in R^{\{N,m\}}\}$, and sampled variance of appearance/texture noise (σ^2).

A motion model is defined as a patch-wrap function $W(p;\gamma)$, such that γ refers to patch vertices. This function (such as Piecewise) applies an affine transformation to map texture into the mean shape. This helps to provide shape-free textures and weight vectors of a test image.

Towards defining shape, texture, and motion models, we first use Generalized Procrustes Analysis (GPA) to remove the similarity transformation parameters (i.e. rotation, scaling and, translation parameters) from the training shape instances $\{S_i\}$. GPA aligns shape instances and obtains similarity-free shapes (D).

Second, we apply PCA on the similarity-free shapes to obtain a mean shape $(\bar{s})$ and shape eigenvectors N_s.

Third, we use a *motion model* (w) to create shape-free textures for a test image (I) by wrapping each (I_i) to the mean shape (s_0) frame. A model can be represented as$\bar{I} = A_0 + A_c$, such that A_0 refers to the initial appearance, and A_c refers to the update parameter where $c = A^T(I - A_0)$.

After that, PPCA is applied on these shapes to obtain mean appearance ($\bar{a}$), appearance eigenvectors (A), and inactive components variance.

We employ Thin-Plate Spline (TPS) transformation to compute the mapping [121]. This is based on the mass centre among triangles extracted using Delaunay triangulation of the source and target shapes. This minimizes the bending energy of TPS interpolation by $f(x, y)$ as below and makes it work as expected with Patch Active Appearance Model. More details are in [122].

5.4.2 Texture Features

Texture features describe distinctive characteristics of an image. Such features are extracted using a descriptive function, which calculates a feature vector for each pixel location $l(x, y)$.

For instance, a test image $\boldsymbol{I}$ of a size $H \times W$ is transformed using a feature extraction function $F(I)$ into a multi-channel matrix of the same size; $F(I) = R^{H \times W} \rightarrow R^{H \times W \times D}$, where ($D$) represents future vectors channels of a descriptor length (L_D); see Equation (5-3).

In our work, we use HOG and patch-based texture features. A patch texture $T_L(l)$ represents test images as several texture-patch vectors and shapes (S).

A patch has a reference landmark (L), and Square Size (SZ) that has fixed width and height. A reference landmark (L) of a location $l$represents a pixel coordinate$(x, y)^T$ of a texture patchT_L.

The size of a patch can be represented as a square patch with a width and height of SZ pixels. For instance, $SZ = 5$ defines a number of 5×5 pixels, surrounding the reference landmark.

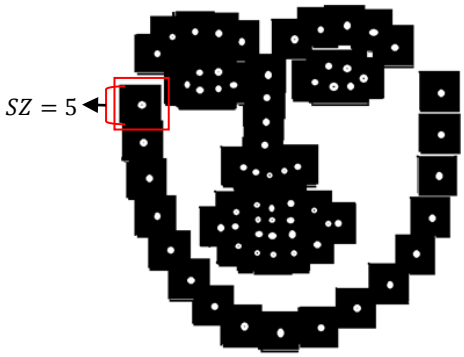


Figure 5-7. Patch size of 5

The selection of SZ is essential, as it influences the accuracy of PAAM and the amount of texture space reduced. Table 5-2 shows the performance of SZ with respect to normalized mean RMS and standard deviation. We selected $SZ = 10$ for the remainder of our work.

5.4.3 Fitting Optimization in PAAM

Fitting in PAAM aims to minimize the summation of squared errors among shape-free HOG textures (H) (Figure 5-8). A Hog-Patch appearance model is defined as below. hgextracts the dense HOG features, and$W(p;\gamma)$ is a patch-wrap function.

$$\text{argmin}_{p,\lambda}\|hg(W(p;\gamma)) - (\bar{a} + U_A)\|_{\lambda}^2 \tag{5-10}$$

Below, we discuss compositional Gauss-Newton algorithms with regard to HOG-PAAM, and fitting performance, in comparison with our proposed formulations AP and APPO.

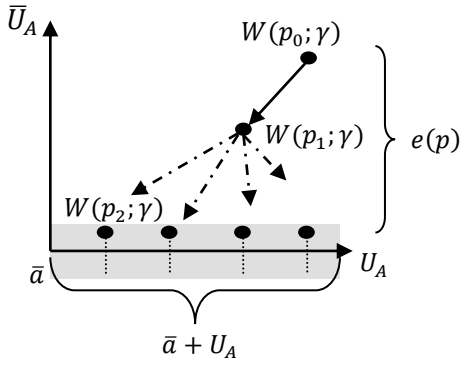


Figure 5-8. Illustration of fitting optimization in PAAM

Generally, a compositional Guess-Newton algorithm is a non-linear optimization method used to solve the problem defined by Equation (5-10) iteratively.

The basic steps involved in PAAM to optimize fitting include 1) estimating the current parameter (p) and defining an incremental wrap parameter (Δp), 2) modifying wrap functions to include patch parameters, 3) applying first order approximation using Taylor series on the error term of patch parameters, 4) solving an incremental parameter Δp, and determining its placement on image (forward) or model (inverse) side, and 5) computing optimal shape parameters iteratively by finding the displacement from incremental wrap $W(\Delta p;\gamma)$ from the current wrap estimate W(p; γ).

A first order Taylor expansion can solve Equation (5-10) with respect to (ΔP). For that, the current patch wrap is composed of incremental patch wraps per iteration. Equation (5-11) shows a forward formula, and Equation (5-12) shows an inverse formula.

$$f: W(p;\gamma) \leftarrow W(p;\gamma) \circ W(\Delta p;\gamma) \tag{5-11}$$

$$f: W(p;\gamma) \leftarrow W(p;\gamma) \circ W(\Delta p;\gamma)^{-1} \tag{5-12}$$

The placement of an incremental patch wrap on an image depends on whether the selected formula is forward or inverse. For both formulas, an incremental patch wrap is linearized around an identity wrap (Figure 5-8). Equation (5-13) shows how to calculate forward linearization, and Equation (5-14) shows how to calculate inverse linearization. An image Jacobian J_W and a model Jacobian ($J_{\bar{a}}$) are computed when ΔP is equal to zero. In both equations, an error image $e(p)$ is calculated as in Equation (5-15).

$$\min_{\Delta p,\lambda} \left\| hg\big(W(p \circ \Delta p;\gamma)\big) - (\bar{a} + U_A) \right\|_{\lambda}^{2} \approx e(p) + J_W \Delta p \tag{5-13}$$

$$\min_{\Delta p,\lambda} \left\| hg\big(W(p;\gamma)\big) - \Big(\bar{a}\big(W(\Delta p;\gamma)\big) + U_A\big(W(\Delta p;\gamma)\big)\Big) \right\|_{\lambda}^{2} \approx e(p) + J_{\bar{a}} \Delta p \tag{5-14}$$

$$e(p) = W(p;\gamma) - (\bar{a} + U_A) \tag{5-15}$$

Using the chain rule formula, an image Jacobian J_W can be expanded as ($\nabla W \times \frac{\partial W}{\partial P}$), while a model Jacobian $J_{\bar{a}}$ can be expanded as ($\nabla W(\bar{a} + U_A)\frac{\partial W}{\partial P}$).

In the next subsection, we compare Alternating Inverse Composition (AIC) and Project Out Inverse Composition (POIC) algorithms.

5.4.4 Fitting Optimization in AIC

AIC algorithms [5] solve minimization problems for the optimal parameters of both shapes and appearances. This is done in an alternative formula to Equation (5-10), as shown in cost Equations (5-16), and (5-17).

$$\text{argmin}_{\Delta P} \| hg(W(p;\gamma)) - \text{a}_{\lambda}(\text{W}(\Delta p;\gamma)) \|_{I - U_A U_A^T}^{2} \tag{5-16}$$

$$\text{argmin}_{\Delta\lambda} \left\| hg\big(W(p;\gamma)\big) - \text{a}_{\lambda} + \Delta\lambda\big(\text{W}(\Delta p;\gamma)\big) \right\|^{2} \tag{5-17}$$

$$\Delta P = H^{-1} J_{AIC}^{T} \left[hg\left(W(p;\gamma)\right) - \bar{a} - \mathrm{U_A}\lambda \right] \tag{5-18}$$

$$\Delta\lambda = U_A^{T} \left[hg\left(W(p;\gamma)\right) - \bar{a}\left(\mathrm{W}(\Delta p;\gamma)\right) - \mathrm{U_A}(\mathrm{W}(\Delta p;\gamma))\lambda \right] \tag{5-19}$$

$$J_{AIC} = \widehat{U}_A \left[J_{\bar{a}}|_{P=0} + \sum_{i=1}^{N_A} \lambda_i J_{u_i}|_{P=0} \right] \tag{5-20}$$

$$H^{-1} = J_{AIC}^{T} J_{AIC} \tag{5-21}$$

$$\lambda \longleftarrow \lambda + \Delta\lambda \tag{5-22}$$

The current estimate $(\Delta P, \Delta\lambda)$ is calculated iteratively for λ and P, respectively with a fixed estimate.

The increment of shape parameters$\Delta\lambda$, as shown in Equation (5-19), works on the complement of the orthogonal appearance subspace$\widehat{U}_A = I - U_A U_A^{T}$.

After that, we need to calculate the current Jacobian estimate (J_{AIC}), Hessian Matrix, and appearance parameters (λ) as in Equations (5-20), (5-21), and (5-22). Those equations are calculated and updated additively; this is to solve the optimal appearance parameters.

The iterative update of parameters causes the complexity cost for each iteration to be $O(m^2 n^2 + (m+n)N + n^3)$.

5.4.5 Project-Out

POIC algorithms separate shapes from appearances by solving Equation (5-10). POIC does not have an additional step to optimize appearance parameters. Equation (5-23) shows the updated cost function used for POIC.

$$\mathrm{argmin}_{\Delta p} \left\| hg(W(p;\gamma)) - \bar{a}\left(W(\Delta p;\gamma)\right) \right\|^2_{I - \mathrm{U_A U_A^T}} \tag{5-23}$$

POIC is computed using the orthogonal complement of the appearance subspace. The equations below show how to compute the first Taylor expansion, shape parameters increment (ΔP), Hessian matrix (H), and Jacobian estimate (J_{POIC}) in POIC.

$$\bar{a}\left(\mathrm{W}(\Delta P;\gamma)\right) \approx \bar{a} + \mathrm{J}_{\bar{a}}|_{\mathrm{P}=0} \Delta P \tag{5-24}$$

$$\Delta P = H^{-1} J_{POIC}^{T} \left[hg\left(W(p;\gamma)\right) - \bar{a} \right] \tag{5-25}$$

$$H^{-1} = J_{POIC}^{T} J_{POIC} \tag{5-26}$$

$$J_{POIC} = \widehat{U}_A J_{\bar{a}}|_{P=0} \tag{5-27}$$

In order to reconstruct appearance vectors, appearance parameters (λ) must be updated for each iteration as follows.

$$\lambda = U_A^{T} \left[h\left(W(p)\right) - \bar{a} \right] \tag{5-28}$$

POIC algorithms are fast, since Jacobian and Hessian matrices can be precomputed. Hence the complexity cost per iteration is $O(nN + n^2)$.

5.5 Adaptive probabilistic optimization

An adaptive probabilistic-based optimization defines a Maximum Likelihood formulation. This formulation obtains optimal shape parameters using a probabilistic generative model such as Probabilistic PCA where sampled noise variance is already trained from the training phase. This model solves the minimization problem of shape parameters that are required in Equation (5-10).

Our cost function for a particular test image I depends on 1) linearizing I to solve appearance parameters $\mathrm{a_p}$; in inverse settings, $\mathrm{a_p}$ is replaced with $\bar{\mathrm{a}}$, and 2) computing Euclidean distance of ZCA Mahalanobis transformation (whitening) within texture space $I - \mathrm{U_A}\sqrt{B^{-1} + \sigma^2}\mathrm{U_A^T}$, where B is the orthonormal bases on the diagonal matrix and σ^2 is the sampled noise variance.

$$\mathrm{argmin_{\Delta p}}\left\|\mathrm{hg}\big(\mathrm{W(p;\gamma)}\big) - \mathrm{a_p}(\mathrm{W}(\Delta\mathrm{p};\gamma))\right\|^2_{I-\mathrm{U_A}\sqrt{\boldsymbol{B^{-1}+\sigma^2}}\mathrm{U_A^T}} \tag{5-29}$$

For iteration, we use a precomputed appearance parameter $\mathrm{a_p}$ to estimate the current $\Delta\mathrm{p}$. A precomputed value is calculated by linearizing the test image as in the following cost function.

$$\mathrm{a_p} = \mathrm{U_A^T}\big[\mathrm{hg}\big(\mathrm{W(p;\gamma)}\big) - \mathrm{a_\lambda} + \Delta\lambda\big(\mathrm{W}(\Delta\mathrm{p};\gamma)\big)\big]_{\lambda=\mathrm{I},\mathrm{p}=0} \tag{5-30}$$

The update of $\Delta\mathrm{P}$ is computed as follows.

$$\Delta\mathrm{P} = \mathrm{H^{-1}J_{PO}^T}\big[\mathrm{hg}\big(\mathrm{W(p;\gamma)}\big) - \mathrm{a_p}\big] \tag{5-31}$$

$$J_{PA} = J_I a_p \tag{5-32}$$

$$H^{-1} = J_{PA}^T J_{PA} \tag{5-33}$$

$$\lambda = U_A^T[h(W(p)\,) - a_p] \tag{5-34}$$

PPO projects out appearance variations $\widehat{\mathrm{U}}\mathrm{n_{PPO}} = I - \mathrm{U_A}\sqrt{B^{-1} + \sigma^2}\mathrm{U_A^T}$ based on the orthogonal complement of the appearance subspace, where Jacobian ($\mathrm{J_{PPO}} = \widehat{\mathrm{U}}\mathrm{n_A J_{\bar{a}}}|_{\mathrm{P}=0} + \mathrm{J_I a_p}$), and Hessian ($\mathrm{H^{-1}} = \mathrm{J_{PPO}^T J_{PPO}}$) are projected-out respectively.

Adaptive Probabilistic algorithms linearize and solve appearance parameters as a template for pre-computing Jacobian and Hessian matrices. Hence the complexity cost per iteration is $O(nN + n^2)$.

Algorithm 1 shows the steps involved implementing our approach. When setting the 'projectOut 'parameter to true (Line 3), the algorithm applied will be APPO, otherwise, it will be AP.

Algorithm 1 | **Complexity** $O(nN + n^2)$

```
// states: {Setup, Fit}
AdaptiveProbabilisticFitter: {
 voidSetup(bool projectOut = false){// project-out is disabled by default
delta_parameter = computeJacobianWrap();
      motionTransforfer = ThinPlateSplinesFunction;
      this.projectOut = projectOut;
}

void Fit(int max_iterations = 10) {
      //Compute wrapped image
      wImage_parameters = computeWrapImageParams(textureModelMean);
      // compute projected texture weights
      textureWeights = projectTextureVector(wImage_parameters);
      // update texture mean template
      textureModelMean.updateTextureMean(textureWeights);
      textureComponents = textureModelMean.orthonormalizedComponents;
      epsilon =textureModelMean.noiseVariance;
      appearance - = textureModelMean.getZCAWhitenedComponents(epsilon).T();
      jacobian = computeSteepestDescentImage(textureModelMean, delta_parameter, appearance);
      if(projectOut) {
            jacobian =jacobian.T();
            jacobian =( jacobian -(jacobian.dot(textureComponents.T())).dot(textureComponents)).T();
      }
      hessian= computeHessian(jacobian );

     // 10 is the maximum iterations number
     for (int iterations = 0 ; error<= convergenceError && iterations  <max_iterations; iterations++ ){
            wImage_parameters = computeWrapImageParams(textureModelMean);
            sd_delta_update = computeSteepestDescent(jacobian,wImage_parameters,  textureModelMean);
            delta_parameter.update(sd_delta_update);
            delta_parameter.compose(delta_parameter.iverse());
            error = delta_parameter.calculateError();
      }
   }
}
```

5.6 Experiment results

Our experiments are applied on in-the-wild datasets, Helen [123] and AFW [18], which consist of images that do not have special constraints or conditions, and hence have a variety of variations such as resolution, illumination, pose, and occlusion.

In our training, a HOG-PAAM model is trained against 811 images of LFPW [124] while selecting a patch size of 10 (SZ) (Section 5.6.2), 12 eigenshapes N_S and 50 eigentextures N_A (Section 5.6.3). We determined these parameters based on performance evaluation on different parameters with the aim to select the minimum value to reduce the model space and improve efficiency.

We also obtained the Faces In-the-Wild Challenge (300-W) [125], in which annotation is based on 68 landmarks.

A fitting process starts with computing the boundary box of faces using the face detector of Max-Margin Object Detection (MMOD) [126], and adjusting the mean shape appropriately within the boundary of the detected boxes using global similarity transform.

The fitting accuracy is evaluated using point-to-point RMS error between estimated facial feature landmarks (x_i^p, y_i^p) and ground truth landmarks (x_i^g, y_i^g), which are normalized by the face size. The face size is computed from ground truth landmarks (FS), and error (NRMSE) are denoted as:

$$FS = \frac{\left(\max x_i^g - \min x_i^g + \max y_i^g - min y_i^g\right)}{2} \tag{5-35}$$

$$NRMSE = \frac{\sum_{i=1}^{L} \sqrt{\left(x_i^p - x_i^g\right)^2 + \left(y_i^p - y_i^g\right)^2}}{\mathrm{FS}} \tag{5-36}$$

Normalized RMSE is tested using a repeated-measures (RM) analysis of variance (ANOVA) for multiple comparisons followed by a Bonferroni correction post hoc test, and a significance threshold of p = 0.05. Mauchly's test is used to validate the assumption of sphericity, where the variances of the differences between all combinations are equal, the Greenhouse-Geisser correction is applied if not. Then, we sort groups based on the values of the Mean μ, of the normalized RMSE and when we display the results we show Standard Deviation σ (SD), such that$\mu \pm \sigma$. In tables such as Table 5-2, we sort by groups of *accuracy bands* based on their statistical significance, such that each group will have a statistical significance greater than the next ones in the order. The shaded cells in each group highlight the significant ones.

5.6.1 A comparison of different HOG Parameters

In this experiment, we need to show the performance of HOG-AAM on a combination of different parameters, specifically, cell sizes ($N_{cell} \in \{4, 8\}$) and cell blocks ($N_{block} \in \{1, 2\}$), which result in HOG images of a number of channels ($L_D \in \{9, 36\}$). We compare our HOG-PAAM and HOG-AAM. The experiment is applied on 224 images of LFPW [124]. The results of this experiment are represented in a form of Cumulative Error Distribution (CED) as shown in Figure 5-9 and Table 5-1.

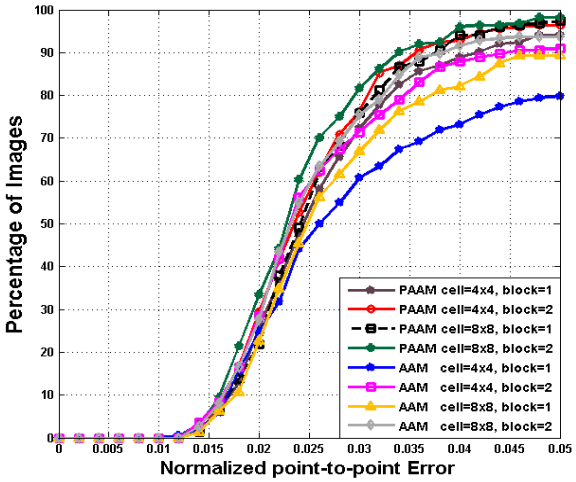


Figure 5-9. A comparison of different HOG Parameters

An RM ANOVA using Bonferroni correction is performed on PAAM with different HOG parameters; it is found to be significant when compared with AAM except when cell block is 2 ($p < 0.0001$). Clearly, block normalization can greatly improve the accuracy of fitting, as opposed to reducing cell sizes N_{cell}, which slightly degrades accuracy.

The experiment shows that HOG-PAAM noticeably outperforms HOG-AAM and with lower model space size (cells and blocks). In particular Hog parameters ($N_{cell} = 8$, $N_{block} = 2$) has the least mean square error (0.024 ± 0088), and PAAM with Hog parameters ($N_{cell} = 4, N_{block} = 1$) still slightly outperforms AAM-HOG ($N_{cell} = 8, N_{block} = 2$) with 33.9% lower model size.

Table 5-1. A comparison of NRMSE of different HOG Parameters, with accuracy bands shaded to show differences of statistical significance

HOG			$\mu \pm \sigma$	Accuracy band			F
Type	C	B		A	B	C	29.002
PAAM	8	2	0.024 ± 0.0088				
PAAM	4	2	0.025 ± 0.0137				
PAAM	8	1	0.025 ± 0.01				
PAAM	4	1	0.028 ± 0.0156				
AAM	8	2	0.030 ± 0.0285				
AAM	4	2	0.032 ± 0.0296				
AAM	8	1	0.035 ± 0.0342				
AAM	4	1	0.046 ± 0.0481				

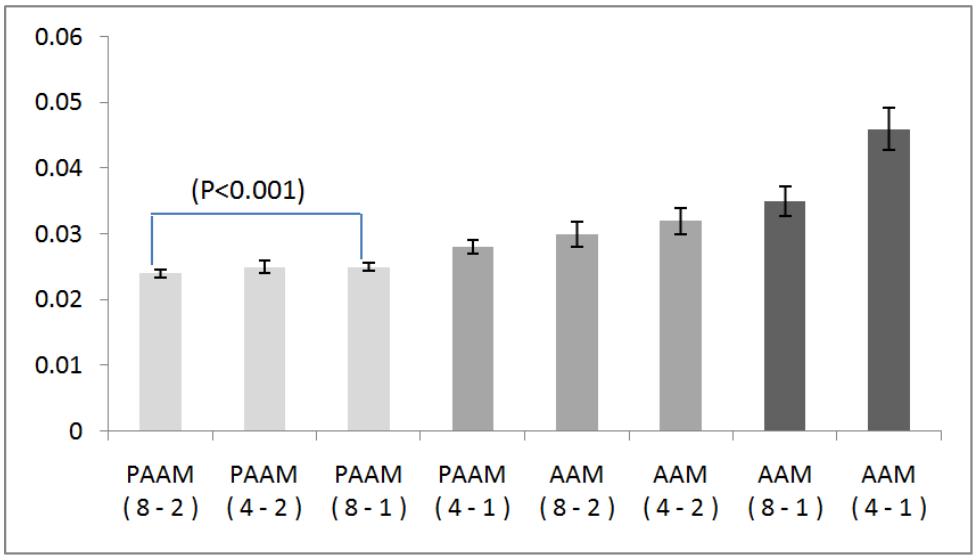


Figure 5-10. Histogram of NRMSE of different HOG Parameters (C-B). Error-bars are standard error

Lighter colour indicates a greater statistical significance and different accuracy band

5.6.2 A comparison of different PAAM patch sizes

In this section, we compare the fitting performance of different path sizes, and show how the model size can be further compressed. which reduces memory footprint and improves speed. The results of this experiment are represented in the form of a CED chart as shown in Figure 5-11, as well as in Table 5-2.

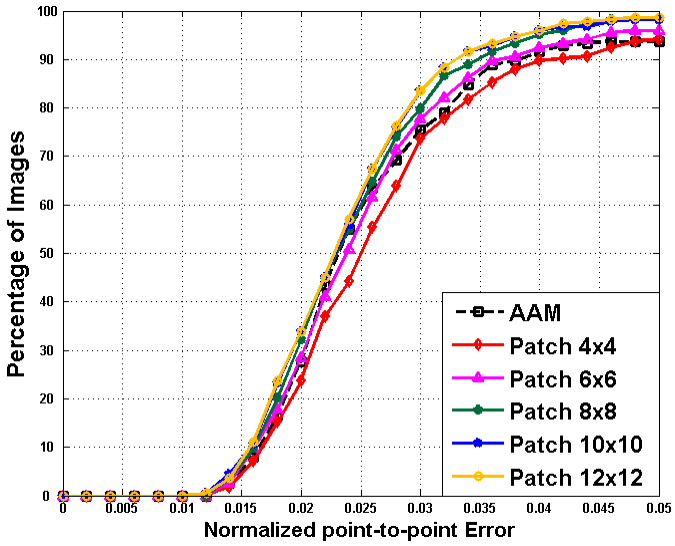


Figure 5-11. A comparison of different PAAM patch sizes

An RM ANOVA using Bonferroni correction is performed and patches with size from 10 to 14 ($p < 0.005$) are found to be significant at $p < 0.005$. A pairwise comparison showed significance difference when patch size is less than 6 ($p < 0.0001$).

The experiment shows that HOG-PAAM with patch size 10 clearly outperforms AAM and has the least space size and least mean square error (0.023±0.008).

Table 5-2. A comparison of NRMSE of AAM with different PAAM patch sizes

	$\mu \pm \sigma$	Accuracy band	*P*	Space	F
		A	*(<0.005)*	Reduced	
AAM	0.030±0.0285		-	-	10.589
10x10	0.023±0.0079		0.0016	14.42%	
12x12	0.023±0.0078		0.0017	-4.38%	
14x14	0.024±0.0082		0.0028	-15.48%	
8x8	0.024±0.0085		0.0031	51.16%	
6x6	0.027±0.021		0.230	30.04%	
4x4	0.028±0.032		0.655	61.66%	

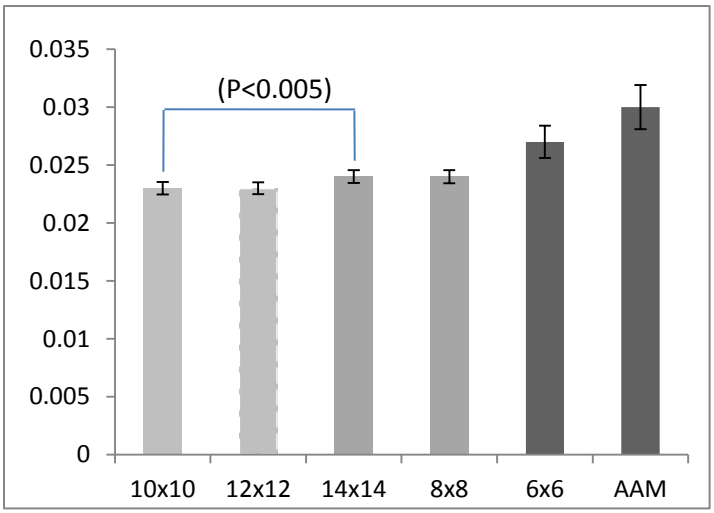


Figure 5-12. Histogram of NRMSE of different PAAM patch sizes. Error-bars are standard error

Lighter colour indicates a greater statistical significance and different accuracy band

5.6.3 A comparison of different eigentexture sizes

In this experiment, we show the performance of eigentextures numbers with respect to accuracy and model size. The results of this experiment are represented as a CED chart shown in Figure 5-13.

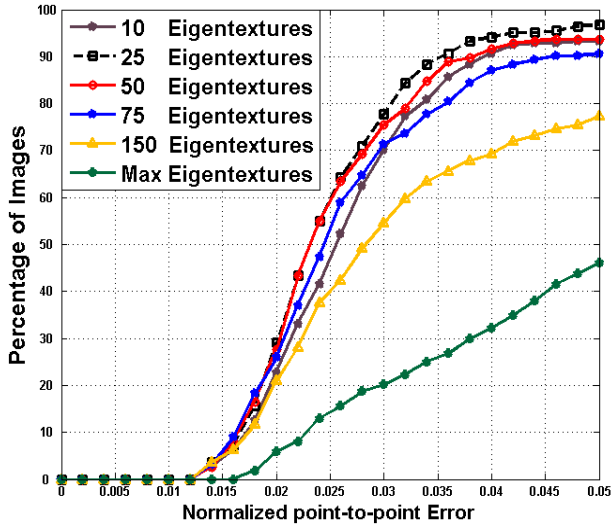


Figure 5-13. A comparison of different eigentexture numbers

Table 5-3. A comparison NRMSE of different eigentexture numbers

EigenTextures #	$\mu \pm \sigma$	Accuracy band		Reduced Space
		A	B	
MAX	0.064±0.0439			-
50	0.023±0.0079			93.39%
25	0.024±0.0082			96.79%
75	0.025±0.0166			90.37%
10	0.028±0.0101			98.51%
150	0.034±0.0297			80.74%

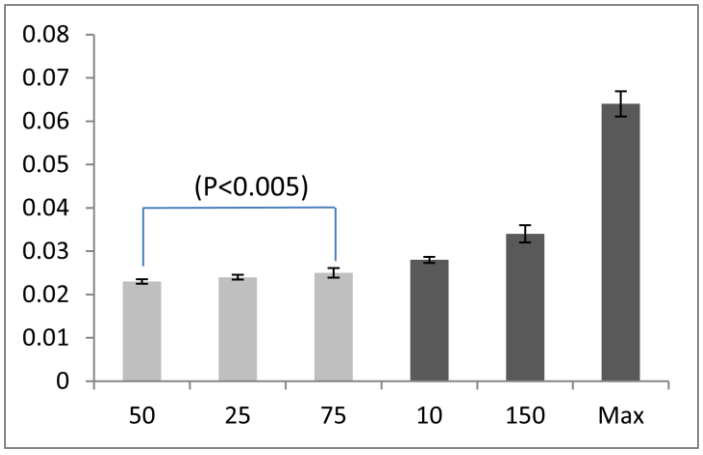


Figure 5-14. Histogram of NRMSE of different eigentexture numbers. Error-bars are standard error

Lighter colour for lower mean and different accuracy band

An RM ANOVA using Bonferroni correction is performed and eigentexture numbers from 10 to 100 are found to be significant at $p < 0.0001$. A pairwise comparison found no significance difference between numbers from 10 to 100 at $p < 0.005$.

This experiment shows the eigentexture value of 50 has the least mean square error (0.023 ± 0.0079), with 93.39% reduced model size.

5.6.4 A comparison based on the affective states

In this section, we show an experiment (Table 5-4), in which we compare the performance of affective states on fitting with respect to accuracy. The dataset used for the comparison is KDEF [127]. We had to annotate all images manually in KDEF with the 68 landmarks to be consistent with our training dataset.

Table 5-4. NMSRE of the affective states, as computed using several state-of-the-art methods

	Angry	Disgust	Fear	Happy	Neutral	Surprise	Sad
	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$
AP	0.018 ± 0.004	0.019 ± 0.0042	0.015 ± 0.0027	0.014 ± 0.0022	0.014 ± 0.0021	0.016 ± 0.0049	0.017 ± 0.0033
AIC	0.018 ± 0.0047	0.019 ± 0.0043	0.016 ± 0.0032	0.014 ± 0.0024	0.013 ± 0.0022	0.016 ± 0.0050	0.017 ± 0.0049
APPO	0.018 ± 0.0042	0.019 ± 0.004	0.016 ± 0.003	0.014 ± 0.0022	0.013 ± 0.0021	0.016 ± 0.0053	0.017 ± 0.0037
SIC	0.023 ± 0.0072	0.023 ± 0.0057	0.020 ± 0.0052	0.017 ± 0.0045	0.016 ± 0.0031	0.033 ± 0.0216	0.023 ± 0.0061
POIC	0.020 ± 0.0076	0.020 ± 0.0055	0.017 ± 0.0032	0.015 ± 0.003	0.015 ± 0.0021	0.018 ± 0.0063	0.019 ± 0.005

An RM ANOVA using Bonferroni correction is performed and the affective-states of neutral and happy are found to be significant at $p < 0.0001$. A pairwise comparison found a significant difference between disgust, anger, surprise, sadness and fear at ($p < 0.0001$), as shown in Table 5-5.

The results indicate that the training datasets did not cover enough variations of facial expressions, since most subjects tended to be neutral most of the time, or happy in many other occasions. Such limitations cause neutral and happy emotions to have higher statistical significance than other affective states. This means that using the estimated facial features only to evaluate the other affective states will potentially have lower accuracy. Hence, there is still a need to use the appearance features along with the estimated points of facial features to provide reliable affective states detection. This will be less subject to the limitations mentioned above.

Table 5-5. A pairwise comparison of affective states

Method	Contrast	P-value
Neutral	Disgust, Angry, Surprise, Sad and Fear	< 0.0001
	Happy	0.027
Happy	Disgust, Angry, Surprise, Sad, and Fear	< 0.0001
Fear	Disgust and Angry	< 0.0001
	Surprise	0.003
	Sad	0.563
Sad	Disgust and Angry	< 0.0001
	Surprise	0.011
Surprise	Disgust	< 0.0001
	Angry	0.007
Angry	Disgust	0.020

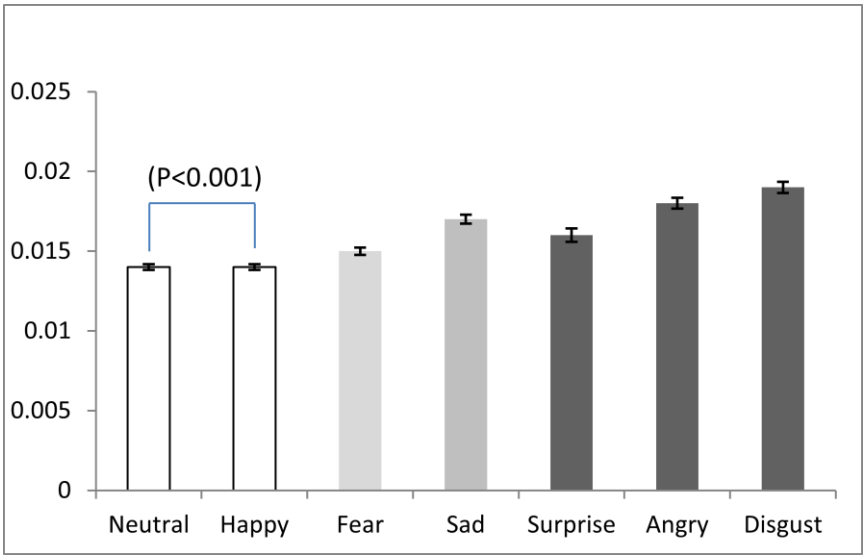


Figure 5-15. NRMSE histogram of pairwise comparison of affective state. Error-bars are standard error Lighter colour indicates a greater statistical significance and different accuracy band

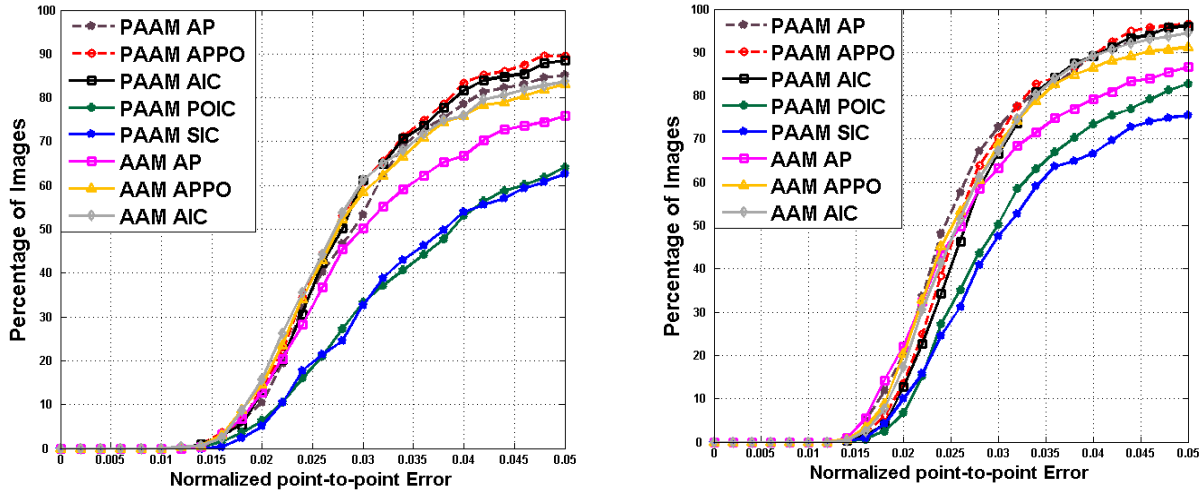
5.6.5 A comparison of state-of-the-art methods

In this experiment, we compare the performance of our introduced fitting formulations against AIC, SIC and POIC. We used the best-obtained parameters from previous experiments; patch SZ = 10, eigentextures = 50, and HOG parameters ($N_{cell} = 8, N_{block} = 2$). Then, as mentioned, we tested the performance on 68 landmarks against datasets, Helen [123] and AFW [18]. The results of this experiment are represented as a CED chart (Figure 5-16 and Table 5-6).

Table 5-6. A comparison of state of art methods and our methods, AP and APPO

Our methods are marked with *, and are shown to be comparable with AIC, even though the latter is much less efficient. Values are NRMSE.

Method	Contrast	Helen			AFW			
		P-value	$\mu \pm \sigma$	Accuracy band	P-value	$\mu \pm \sigma$	Accuracy band	
				A			A	B
AP*	POIC	< 0.0001	0.030±0.019		< 0.0001	0.038±0.033		
	SIC	< 0.0001			< 0.0001			
	AIC	0.563			0.472			
	APPO	0.851			0.494			
AIC	POIC	< 0.0001	0.031±0.016		< 0.0001	0.037±0.032		
	SIC	< 0.0001			< 0.0001			
APPO*	POIC	< 0.0001	0.30±0.015		< 0.0001	0.037±0.032		
	SIC	< 0.0001			< 0.0001			
	AIC	0.649			0.975			
SIC	POIC	0.398	0.43±0.031		0.141	0.053±0.037		
POIC			0.46±0.057			0.060±0.061		



(a) AFW (b) Helen

Figure 5-16. A comparison of state-of-the-art methods

An RM ANOVA using Bonferroni correction is performed and AP, APPO and AIC are found to be significant at p< 0.0001. Pairwise comparison found no significant difference among AP, APPO and AIC at p< 0.0001.

This experiment, using the Helen dataset, shows APPO has the least mean square error (0.030±0.015), and on the AFW dataset, APPO has the least mean square error (0.037±0.032). AP and APPO have a relatively close error rate but with better efficiency (Figure5-17).

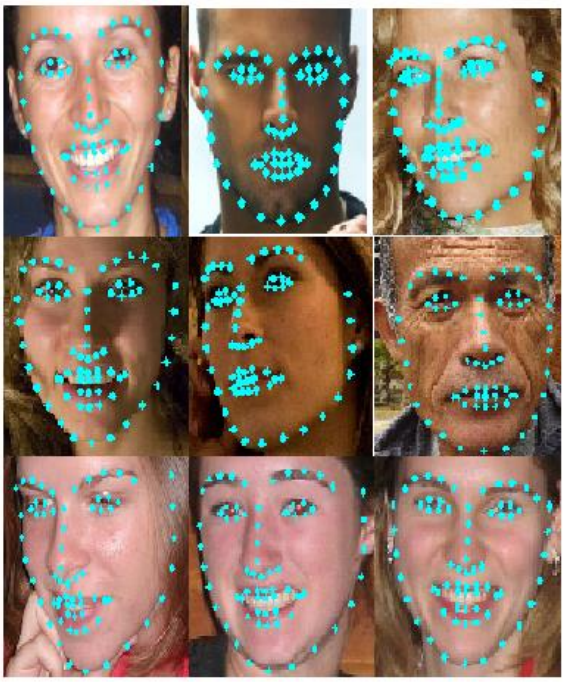


Figure5-17. Selected fitting results from the APPO method on the three datasets. The first row is LFPW [124]**, second row is Helen** [123]**, and third row is AFW** [18]**.**

5.7 Summary

In this chapter, we presented two novel fitting techniques, Adaptive Probabilistic (AP) and Adaptive Probabilistic Project-Out (APPO), which are both based on Patch Active Appearance Model (PAAM) and use dense HOG features. We call our approach HOG-PAAM.

We evaluated the fitting accuracy of facial features based on affective state recognition. The motivation of this comparison was to investigate whether fitting accuracy can change among different affective states. We found that certain affective states such as happy and neutral have significant differences from other states. We conclude that this is mainly because the training dataset does not consider different variations of affective states in the training process. This as well indicates the need of using appearance features along with the estimated facial features to produce a reliable affective state detector.

Finally, we evaluated our algorithms against common techniques, AIC, SIC and POIC. The datasets used in the evaluation are Helen [123] and AFW [18]. Our algorithms sometimes but not always showed significant differences in terms of accuracy, as compared to other techniques. They did always show significant differences in terms of efficiency and space. The results from experimenting with the Helen dataset indicate that APPO has the least mean square error (0.030 ± 0.015). On the other hand, the results of AFW indicated that APPO has the least mean square error (0.037 ± 0.032). Both AP and APPO have a similar error rate as compared to other techniques but with better efficiency.

Chapter 6 Multimodal Fusion

In this chapter, we show how we can fuse inputs captured from various sequential modalities. This is a crucial part of our research, since sensors of multiple modalities are used to collect data, which can 1) be of different kinds and unsynchronized, 2) have different temporal representations, 3) be noisy, and 4) have an excessive number of features to make a straightforward analytical decision.

Deep networks, which excel in learning features from visual data, can be used to overcome such limitations. Nonetheless, deep supervised models, such as convolutional neural networks, are expensive, since they require a large amount of labelled data.

We propose a novel approach, Deep Temporal Credence Network (DTCN), which is a multimodal sequential, and semi-supervised deep network. DTCN has a sequential multi-input structure that parameterizes a recurrent deconvolutional generative model with a recurrent convolutional inference model. Hence, it is composed of two independent generative and inference models.

Multimodal Convolutional Recurrent Variational Autoencoder (MCRVAE),is a generative model of data, captured from multiple modalities. It extracts a joint representation synchronizing and fusing multiple modalities via recurrent joint-variational layers. A joint-variational layer is trained using Bayesian inference to locally learn the latent representation of different modalities of data inputs, and be able to infer missing modalities. This allows for learning compact and abstract high-feature representation, as well as subtle motion patterns. After that, DTCN uses these latent representations to predict multi-output regression or classification. We build our models on two encoding-decoding structures. The first is used to handle normal image input types and the other is used to handle continuous data representation, time-series data.

We aim to solve two main issues, multiple modalities and temporal dynamics. For that, our framework contributes two novel fusion mechanisms, Temporal Feature Fusion (TFF) and Temporal Output Fusion (TOF). TFF depends on MCRVAE to capture the compressed joint feature representations of input modalities used to build temporal network structure, and to solve the problem of complex temporal dynamics. TOF takes place after multi-output regression or classification in order to fuse outputs. TOF enables encoding temporal dynamic changes of outputs over fixed duration, which is helpful to build a consensus function that alleviates the time-domain-dependent accuracy problem. TOF encodes outputs as a heatmap image with a standardized range, and then incorporates temporal changes in order to resolve the issues of time domains and constraints.

Results show that our semi-supervised learning method is promising, in that it is able to infer missing modalities, and still maintain good performance In the next chapters, we evaluate our framework against the state-of-the-art related to different high-level tasks such as recognising affective states, and validating the performance and robustness of fusion strategies.

In this chapter, we assume that readers have good knowledge of deep learning.

6.1 Introduction

Feature representation [33], [34] is linked to efficient understanding of image data, and to performing better visual recognition. Traditionally [128], [129], it is created by manually determining the features that can provide a better abstraction of visual data, which improves the performance of recognition such as HAAR or HOGS [46]. Then, features extracted are used to train classifiers such as image classification.

Recently, deep learning models [130], [131] have become a leading way to learn feature representation automatically by creating deep hierarchical structures. For example, stacking shallow models in an appropriate hierarchical structure has helped to uncover and automatically learn more abstract features such as edges [34].

Deep models are known to have good performance due to their deep hierarchical structure (Figure 6-1). Deep hierarchal models consist of many layers, which examine the latent representation of the input data. In a way, that can transform data into a better representation.

A typical example of deep hierarchical networks is Convolutional Neural Network (CNN). CNN is constructed by stacking multiple distinct layers types that include convolution, pooling, and local response normalization (LRN). A convolution structure [131] preserves spatial information, which shares weights and relative location features through convoluting between an input image and filter.

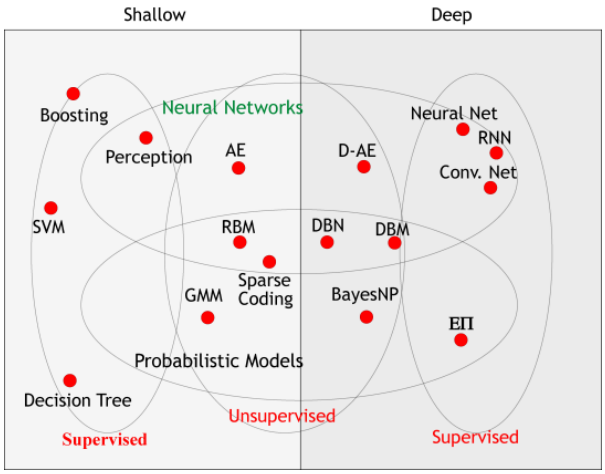


Figure 6-1. Deep learning method overview. Adapted from [132]

The expected results of convolution operations depend on the number of kernels *k* to extract convolved features. Hence, there is a need to reduce these features to avoid overfitting, and to increase the variance, which is pooling. LRN is used to normalize the local input regions, with neurons that have unbounded activation such as Rectified Linear Units (ReLU). CNN is challenged due to requiring a massive number of training parameters and labelled data, which consequently consume a plethora of computing resources.

Unsupervised techniques are commonly used in many applications to understand latent and shared representation of unlabelled data, which can be used to have a discriminative representation, and to eliminate redundancies. In unsupervised deep neural networks, a deep hierarchy is created by stacking layers on the top of each other where each layer feeds the next and so on [34].

Semi-supervised deep networks are initially trained in an unsupervised way to learn a better representation of the input data, and then are fine-tuned to become a classifier or regressor. Semi-supervised techniques utilize the unsupervised nature to stabilize network performance.

An autoencoder is a common example of unsupervised learning, which is typically used for dimensionality reduction [35]. It is a non-recurrent feed forward network, which is trained using the *encoder-decoder* paradigm to reconstruct their input and end up with lower-dimensional representation.

There are many variations of autoencoders, which differ based on their approximation learning of the identity function and how the constraints are placed. A Stacked Autoencoder (SAE) [133] is a stack of shallow autoencoder models that first encodes input data in order to learn features, and then reconstructs that input data. A Stacked Sparse Autoencoder (SSAE) introduces sparsity on hidden units during network training to enable sparse representations of inputs [134]. A Denoising Autoencoder (DAE) takes a different direction by corrupting the input with random noise during network training to improve the performance stability with noisy or different variations of inputs. A Variational Autoencoder (VAE) uses variational Bayesian methods to learn the probabilistic distribution of latent variables [135]. A VAE differs from a DAE on how they corrupt input data during training, as the first performs it in the stochastic hidden layer, while the other at the input level. However, the previous variations have a main limitation as their structures work by vectorizing high-dimensional input and discarding potential latent information from their hierarchical structure [133].

Convolution structures are used to solve such a limitation by maintaining the information of the hierarchical network structure. A Convolutional Autoencoder (CAE) convolves the input to train an autoencoder [136]. The convolved features of each layer enable more abstract feature representation, which maintains the structural information.

There are different types of unsupervised deep networks that rely on convolution structure rather than autoencoders; these include Convolutional Restricted Boltzmann Machines (CRBM) [137] and Convolutional Deep Belief Network (CDBN) [138]. They mainly differ based on being generative models that learn from the probability distribution of their inputs. DBN can be alternatively described as a composition of simple unsupervised learning modules such as autoencoders or RBMs [139]. However, DAE and VAE inherit the autoencoder architecture, and can work as generative models [140]. Vincent showed that DAE can be competitive to RBM [141].

Understanding temporal dynamics relies on motion estimation and prediction. It is inspired from the fact that infants develop skills for handling occlusion, and covering from motion understating [142]. With appropriate understanding of sequences, temporal dynamics can solve complex problems such as emotion recognition.

However, the spatiotemporal sequence results in a high dimensionality of features, making it nontrivial to handle. Therefore, using an unsupervised technique would be beneficial in such a situation. Recently, recurrent neural networks using Long-Short-Term-Memory (LSTM) [143] has been used to tackle such issues, and to perform sequence learning. Most of these approaches perform in a supervised manner. Combining LSTM with an autoencoder architecture is shown to be able to predict future frames in an unsupervised manner [144]. However, this approach takes a sequence of linear frames as input. As we focus on estimating optical flow, there might be a need for variational smoothness.

Therefore, our work incorporates convolution structure with LSTM modules (Convolutional LSTM) in VAE architecture. This is to satisfy the issue of unsynchronized data of different temporal representations.

Multimodal deep neural networks have gained attention for their capabilities to find correlations between multiple sources, and to outperform networks that only learn *unimodal* features [145]–[149]. Layer-wise training is an efficient approach whereby training takes place in many stages, instead of one single optimization for all stacked layers. A multimodal deep autoencoder with greedy layer-wise training enables a means of learning low-level features and cross-modality correlations [145]. We use a deep multimodal neural network, to satisfy better denoising of data and reduction of an excessive number of features resulting from multiple modalities.

We introduce two novel fusion strategies Temporal Feature Fusion (TFF) and Temporal Output Fusion (TOF). TFF is used to find joint representation, and fuse multiple modalities, and then 1) reconstruct and synchronize the multimodal input given, 2) reduce their dimensionality and 3) predict future frames of their input.

TOF takes place at a later stage, where we have multi-output result data, such that a deep understanding of their temporal changes for a fixed time is needed. We introduce a novel way to encode the temporal changes of multi-output signals into a normalized heatmap image, which can be used with the DTCN architecture. We compare these approaches in terms of their performance in recognizing basic and complex emotion (Section 6.6).

We provide a generative deep multimodal semi-supervised approach using our DTCN, as well as MCRVAE, which applies variational inference training to fuse multiple modalities. Then, we use our semi-supervised trained inference model for classification [150]. Multimodal features show better results than unimodal features networks (Chapter 2) [145].

The contributions related to this chapter can be listed as follows:

1) We demonstrate our Multimodal Convolutional Recurrent Variational Autoencoder (MCRVAE) architecture that applies unsupervised training to fuse multiple modalities,
2) We explain our novel Deep Temporal Credence Network (DTCN), a semi-supervised deep network on the top of the MCRVAE architecture.
3) We introduce a novel way to encode temporal changes into a representation compatible with convolution structure.
4) We provide two encoder-decoder structure types used to train multi-output regressors and classifiers.

5) We introduce two novel fusion approaches, Temporal Feature Fusion (TFF) and Temporal Output Fusion (TOF).
6) We show a case study that demonstrates the performance of inferring missing modalities and semi-supervised learning methods, which are both used for mental state detection in our research.

6.2 Background

6.2.1 Autoencoders

An autoencoder is a non-recurrent feed forward network that consists of input, hidden, and output layers, such that hidden layers form *encoder-decoder* structure, while the units of both input and output layers are constrained to be the same length (Figure 6-2). An autoencoder can be alternatively defined as an adaptive multilayer network performing as a nonlinear generalization of PCA, in a way that encodes high-dimension input into low-dimensional representation, and decodes and recovers the encoded data to reconstruct the input [151]. Therefore, when the number of hidden layers is the same as input and output units, an autoencoder will perform as a linear PCA [115], [120].

An autoencoder receives an original input ($x \in R^d$), and encodes (f_θ) it using a nonlinear mapping of latent y into a hidden representation $y \in R^{d'}$, where $y = f_\theta(x) = \sigma(W_x + b)$, and $\theta = \{W, b\}$. W, b, and σ correspondingly represent the weight of a learning matrix, bias vector activating neural units, and nonlinear function such as sigmoid.

The reconstruction of inputs takes place while decoding where a latent representation y is back-mapped ($f_{\theta'}$) to a reconstructed vector z, where $z = f_{\theta'}(y) = \sigma(W'y + b')$ and $\theta = \{W', b'\}$. Both W and W' parameters are trained to use a distance function such as Euclidian distance (D) to minimize the mean error of reconstruction between x and z, such that $D(x, z) = \|x - z\|^2$ with approximation $x \approx z$. Therefore, if we attain a good reconstruction of the input, it means the input reconstructed retains much of the given input information.

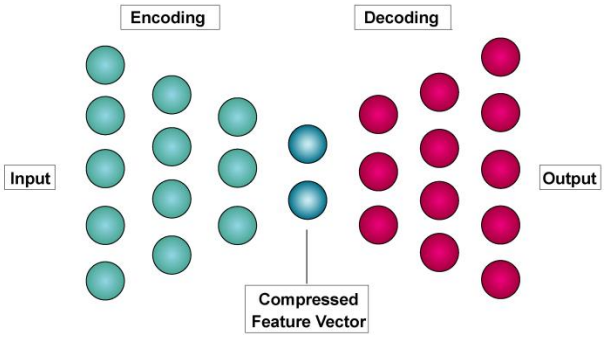


Figure 6-2. Deep autoencoder [152]

6.2.1.1 Sparse autoencoder

Using Gaussian distribution to initialize random weights of deep neural network is typically linked to poor performance and very slow backpropagation. Alternatively, pre-training each layer, in an unsupervised manner with sparsified representation, leads to avoiding local minima and increasing the network's performance stability [151].

Sparsity enforcement aims to constrain hidden units that are activated, so it leads to interesting discovery of data with larger number of hidden units. An active or firing neuron is a neuron with an output close to 1. As the idea of sparse autoencoder is to enforce a constraint on reducing the number of active neurons, most of their hidden unit neurons are inactive [153]. As a result, a sparse autoencoder has only a few active hidden units.

A *k*-Sparse autoencoder is a common example where the sparsity enforcement works by finding the strongest *k* hidden units activated while zeroing out the rest [154]. The error minimization is back-propagated through the active k units to reconstruct their input. However, stacking multiple layers of sparse autoencoder is found to be infeasible to train, as weight optimization is challenging. This is because large initial weights affect the selection of appropriate local minima, while small initial weights result in tiny gradients in early layers, which compromise the training process [151].

6.2.1.2 Denoising Autoencoder (DAE)

A conventional autoencoder can be described as an approach to learn interesting information through retaining information of their input. Therefore, constraints are applied to assist them to find interesting information and avoid perfecting the reconstruction of their inputs.

Toward that direction, a Denoising Autoencoder (DAE) extends the autoencoder structure to enforce a corruption constraint to learn inherent representation of their input to output a clean reconstruction of their inputs from partially noisy ones [133]. A DAE extends the autoencoder architecture to enable working as a generative model, which has competitive performance in comparison to a probabilistic model such as RBM [155]. Hence, a DAE differs from a sparse autoencoder in imposing a denoising criterion on inputs [133], instead of constraining the number of active neurons in their hidden units.

The idea of a DAE revolves around training autoencoders to clean or denoise their input to find latent representation. Training addsv, a variable that distributes an amount of noise to their inputx, based on the features of their input. Then, reconstructing inputs from such a corrupted version, such that encoders can learn features that are more robust in terms of input perturbations [156].

Examples of noise patterns include binomial noise, for black images, and uncorrelated Gaussian noise for coloured images. As another example, in [157], contractive autoencoders use sensitivity penalization, measured using Frobenius norm of Jacobian, such that trained models are trained to be less sensitive to small image variations.

DAE training can be described as corrupting an original input x to be x′, such that x′~qD(x′|x). After that, a corrupted input x' is mapped to a hidden representation, such that $y = f_\theta(x') = \sigma(Wx' + b)$, which is used to reconstruct $z = f'_\theta(y) = \sigma(W'y + b')$.

6.2.1.3 Variational Autoencoder (VAE)

A variational Autoencoder (VAE) is an unsupervised generative neural network that uses a deep directed graphical model $p(x|z)$ to generate data points x randomly from latent variable z [135]. The hierarchy of a VAE model consists of two layers, and computes $p(x|z)$ to learn the encoder-decoder $q(z|x)$ posterior approximation [158]. The encoder is used to map input into continuous inherent variables $q(z|x)$, and then a decoder maps these variables to reconstruct their input $p(x|z)$.

Data point generation takes place in several steps. First, a value $\hat{z}$ is generated with respect to prior coding distribution $p(z)$. Second, a value $\hat{x}$ is generated using conditional likelihood distribution $p(z|x)$. After that, a probabilistic encoder $q(z|x)$ can be used to approximate a posterior $p(z|x)$, such that this can maximize and then relax the likelihood of a data point above lower-bound estimates, where $\log p(x) \geq -D_{KL}(q(z|x)|p(z) + \sum_z q(z|x) \log p(x|z)$. Divergence $-D_{KL}$ is calculated from prior encoding distribution $q(z|x)$ to a prior $p(z)$, and $\sum_z q(z|x) \log p(x|z)$ represents the reconstruction error expected from $z \sim q(z|x)$. Lower-bound estimates are differential, so backpropagation can be used to compute gradient parameters of both encoder and decoder.

The encoder-decoder enables parametric distribution fitting of their data, where both encoding distribution $q(z|x)$ and decoding distribution $p(x|z)$ are parameterized using functions such as multivariate Gaussian. These parameters are trained by a stochastic gradient descent method, with an estimator such as unbiased SGVB [159]. The divergence D_{KL} extends the cost to the reconstruction process of the autoencoder to converge in an expected probability distribution. However, the reconstruction expected in practice is not easy, so differentiable transformation with additional random noise variable ϵ[135] is introduced to re-parameterize random variable z to enable practical estimation.

Data examples are fed to the encoder in order to get the distribution parameters of latent variables, which can be used to obtain data samples. Afterward, data samples are fed to the decoder network, which uses a function such as Bernoulli Cross Entropy (BCE) or squared error [155] to minimize the reconstruction error.

6.2.1.4 Dropout

Dropout is a regularization technique to average the network model and prevent overfitting on fully connected layers [160]. Dropout refers to temporarily dropping out random units for each layer with a certain probability (q). It also includes incoming and outgoing links of the units omitted (Figure 6-3). Therefore, it prevents complex co-adaptation [161] of their input data by training hidden/visible units to avoid dependency on other units to update their states.

(a) Standard neural net (b) After dropout appliance

Figure 6-3. Dropout technique

6.2.1.5 Convolutional Autoencoder (CAE)

Autoencoder variations discard possible spatial information from 2D image structures (width, height and depth/colour channels) [133] and introduce redundancy in their input. Convolution layers arrange neurons in 3 dimensions width, height, and depth (colour channels) (Figure 6-4).

A Convolutional Autoencoder (CAE) [137], [138], [162], [163], introduces additional layers to preserve spatial locality and enable weight sharing among all regions of their input. Therefore, input reconstruction is alternatively a linear combination of abstract input regions. CAE conventionally includes three main layers to be stacked or composed as needed. They are convolution, pooling and Local Response Normalization (LRN). Convolution mainly differs in taking the spatiality and local connectivity structure of input into consideration. Therefore, it 1) enables learning good representation filter of small regions of inputs and weight sharing, and 2) reduces the number of free parameters. Convolutional networks show better results on data that is noisy or corrupted, as compared to conventional stacked autoencoders.

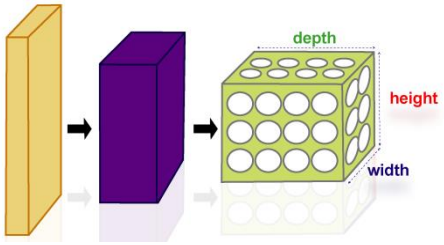


Figure 6-4. CNN Layers arranged in 3D

Convolution can be described as a kernel filter, sliding window, performed on a 2D image to convolve spatial and abstract features. The features convolved maintain 2D image structural information. A kernel filter has a size $n \times n$ and sums element-wise multiplication of its values with original input, while sliding the filter over the whole 2D input matrix. Thus, convolution hyperparameters enable different choices to improve the stability and performance of the network (Figure 6-5). These include convolution size narrow or wide, stride size, depth, and zero padding.

CAE expands its input x into a latent map representation of kernel convolved c features, $y^c = \sigma(\mathrm{W}_\mathrm{x}^c + b^c)$. Y is a set of convolved feature maps, where each c-th map use a single bias B to signify shared features of their whole

input. The reconstruction differs on weight sharing$z = \sigma(\sum_{c \in Y} y^c * W'^c + B)$, where 2D convolution '*' occurs over weights dimensions.

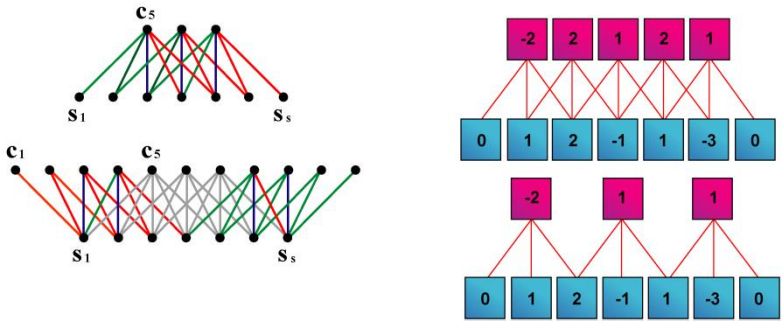


Figure 6-5. CNN hyperparameters

Convolved features can result in overfitting. Hence, a pooling layer is used as a non-linear down-sampling technique in order to reduce output dimensionality to a fixed-size output, and to enable translation-invariance with the most salient representation. It splits input images into a group of non-overlapping sub-regions, where each region results in a selected value such as maximum value. Commonly, it is inserted in-between convolution layers. Common pooling techniques are max-pooling and stochastic-pooling [164], which are used to regularize deep neural networks (Figure 6-6); e.g. eliminate the need for weight-decay regularization (such as L1 and L2 [164]) over hidden units or/and weights.

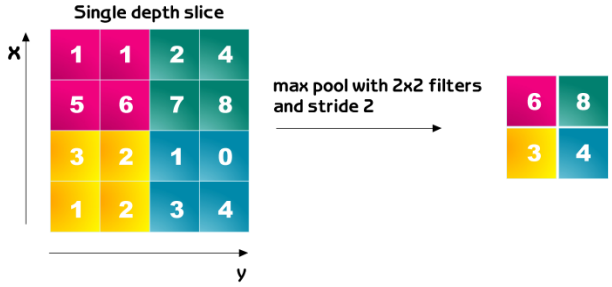


Figure 6-6. Max-Pooling

6.2.1.6 Stacked Denoising Autoencoder (SDA) and Stacked CAE (SCAE)

Autoencoder stacking is a composition technique involving stacking multiple layers [136], [155] of a specific autoencoder in a greedy layer-wise manner, where each layer connects to the input of the successive layer (Figure 6-7). Stacked autoencoder has k-th autoencoders and n layers. Encoding takes a place in a forward order where each$k$-th encoding layer acts as the input data of the next k-th encoding layer and the same for decoding layers, $z^l = \sigma^l(W^l y^l + b^l)$. Decoding runs k-th layers in a reverse order $z^{n+l+1} = \sigma^{n+1}(W^{n-1} y^{n-1} + b^{n-1})$. Thus, the last decoding layer is the final reconstruction of their inputs, and results in a high-order feature representation. Examples include Stacked Denoising Autoencoder (SDA) and Stacked CAE (SCAE).

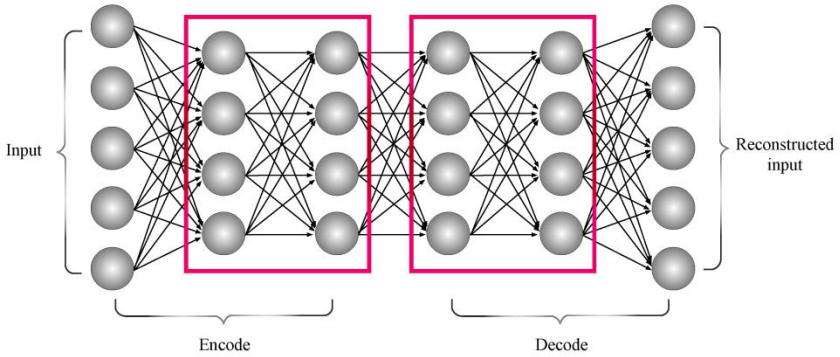


Figure 6-7. Two stacked autoencoder structures

6.2.2 LSTM Autoencoder

Long Short-Term Memory (LSTM) [143] is a type of recurrent neural network, in which hidden units are replaced by LSTM modules (Figure 6-8) to preserve information for long periods, and hence have shown successful results in several tasks such as machine translation [165], acoustic modelling [166] and sequence learning. LSTM can be applied to multi-dimensional spaces, in which data is treated as spatial sequences [167]. It differs from conventional Recurrent Neural Network (RNN) in that their memory modules sum their activities over time.

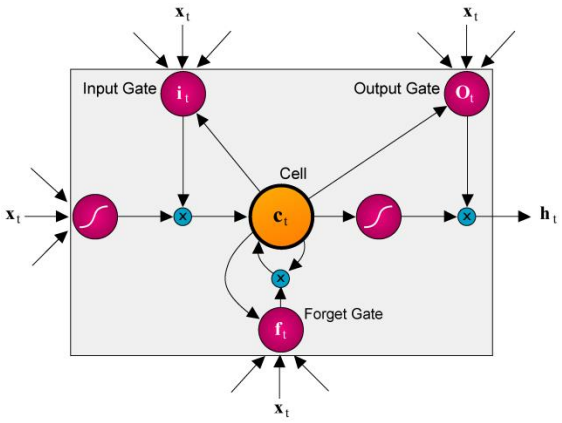


Figure 6-8. LSTM Module

An LSTM module uses gates to signify which interesting information to remember or to forget, and when it should make an output. Each module has a memory cell, which preserves a state c_t at time t. The memory cell is accessed through input i_t, forget f_t, output o_t gates, and $\odot$ is Hadamard entry-wise product. A memory cell has *peephole* linking between their gates and each gate has internal access to the state c_{t-1} of their owning memory cell. LSTM modules work as follows. It takes a sequence of linear inputs, and iteratively takes input x_t at time t with the previous hidden states h_{t-1} of all LSTM cells to compute each gate and cell activations. First, it performs a biased-linear addition of their inputs i_t, followed with a hyperbolic tangent non-linearity to activate the sigmoidal gates. Then, the memory cell state c_t adds it to the multiplication of its state c_{t-1} by the forget gate's activation f_t. Finally, it performs a hyperbolic tangent non-linearity to the memory cell state and multiplies it by the output gate's activation o_t to compute the final LSTM result.

$$i_t = \sigma(x_t w_{xi} + h_{t-1} w_{hi} + w_{i_b}) \quad (6\text{-}1)$$

$$f_t = \sigma(x_t w_{xf} + h_{t-1} w_{hf} + w_{f_b}) \quad (6\text{-}2)$$

$$c_t = i_t \odot \tanh(x_t w_{xc} + h_{t-1} w_{hc} + w_{\tilde{c}_b}) + c_{t-1} \odot f_t \quad (6\text{-}3)$$

$$o_t = \sigma(x_t w_{xo} + h_{t-1} w_{ho} + w_{o_b}) \quad (6\text{-}4)$$

$$h_t = o_t \odot \tanh(c_t) \quad (6\text{-}5)$$

Classic LSTM mostly works in a supervised manner, therefore extending an autoencoder with LSTM modules enables it to work in an unsupervised manner [144] to reconstruct their input or predict future input. LSTM-Autoencoder variations differ based on how they receive their input in terms of linearized, convolutions, or sequence input, and how many layers are they composed of, such as two LSTM layers.

6.3 Architecture

A Deep Temporal Credence Network (DTCN) is a semi-supervised generative directed graphical model composed of an unsupervised Multimodal Convolutional Variational Recurrent Autoencoder (MCRVAE) network (Figure 6-9). MCRVAE is trained in an unsupervised manner to learn probabilistic reconstruction of its input and to act as a feature detector. We use Glorot [168] random weight initialization schemes with batch normalization to resolve the problem of model weight initialization and provide better internal distributed representation to highlight abstract features. Then, the network model is trained in a semi-supervised manner, which means using MCRVAE unlabelled data features with a small amount of labelled data to do a classification task, .

In our architecture, a separate encoder-decoder structure is featured for each modality (Figure 6-9); e.g. video, and audio. Hence, features are learnt separately at each modality, before learning a joint representation correlating all modalities [145]. For efficient approximate inference and continuous latent representation, we use Stochastic Gradient Variational Bayes (SGVB) [169].

Fully connected autoencoders, including denoising autoencoders, ignore the structure of 2D images, which may cause redundant parameters or force spanning the entire set of global features, even for small inputs. Hence, we use a convolutional encoder-decoder structure [136]. In a typical convolutional structure, weights are shared among all locations of an input, which helps preserve the spatial structure and locality of images. The reconstruction depends on patches based on latent code.

The output layers have the same dimensions as input images in order to ease calculating the reconstruction error between inputs and outputs per modality. In order to train networks, we use backpropagation on images that capture the state space. Typically, backpropagation is applied respectively to some parameters to compute an error function gradient, such that weights and parameters are then fine-tuned using Stochastic Gradient Descent with Backpropagation Through time (BPTT). Congruently, a cost function is used to minimize the Mean Squared Error (MSE). Afterwards, top-level activations can be used with other classifiers such Softmax.

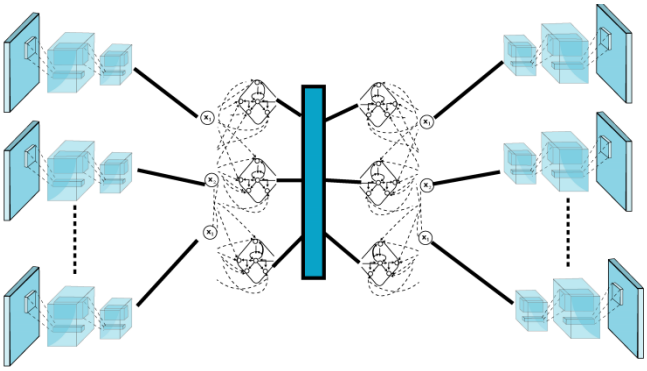


Figure 6-9. Multimodal architecture

6.3.1 MCRVAE

Our approach, MCRVAE combines recurrent network and variational Bayesian methods to extend an autoencoder in order to 1) learn from joint representation of different modalities and high-dimensional sequences and 2) consider the temporal structure of their input sequence.

MCRVAE enables handling multiple modalities by building a joint-variational layer containing a high-level abstraction among all input modalities. We differ from traditional VAE or conditional VAE in that each modality is independently conditioned on a joint-variational layer.

MCRVAE consists of three stages an encoder, recurrent joint-variational, and decoder. The *encoder* compresses the given input, and it contains several layers of convolutions followed by pooling layers. The *decoder* reconstructs the given input, and it consists of several unpooling layers followed by deconvolutions. The decoder maps the features predicted into pixels.

Recurrent-Joint-Variational consists of three layers LSTM encoder, joint stochastic layer, and LSTM decoder. *Recurrent-Joint-Variational* receives the compressed representation of their input modalities and passes it through an LSTM encoder to estimate the conditional probability of their input modalities sequence, and output at subsequent timestamps. Therefore, there is a joint-variational or stochastic layer at every time-step based on LSTM hidden state variable h_{t-1}. The joint-stochastic layer produces $q_T(z_T|x)$ posterior approximation, where z and T represents latent variables and timestamp. Then, the computed prior biases $p(x|z_T)$ are passed as an input to LSTM decoder.

The divergence D_{KL} with random noise variable ϵ [135] is used in the reconstruction process of the autoencoder to converge in an expected probability distribution and enable practical estimation.

6.3.1.1 Multimodal inputs

We consider multiple modality input with different kinds of dimensions and structure. In this research, we work with different types of input such as image and continuous data. A formal description of an input is as follows. We have a dataset that contains M modalities and sequence of datapoints N.

Dataset $(\mathrm{x}^1, \mathrm{x}^2, .., \mathrm{x}^\mathrm{M}) = \{(\mathrm{x}_1^1, \mathrm{x}_1^2, .., \mathrm{x}_1^\mathrm{M}), .., (\mathrm{x}_\mathrm{N}^1, \mathrm{x}_\mathrm{N}^2, .., \mathrm{x}_\mathrm{N}^\mathrm{M})\}$ represents two modalities we want to generate. We avoid any preprocessing except for continuous data representation (Section 2.2), which is encoded into an image.

6.3.1.1.1 Preprocessing

We encode continuous data into two linear heatmap images, to enable the use of convolution operations and to learn abstract features among different time-series data.

Given M modalities continuous data $(X^1, X^2, .., X^M) = \{(x_1^1, x_2^1, .., x_n^1), (x_1^2, x_2^2, .., x_n^2)\}$ where a modality (m) $m \in [1, M]$with an n size and consistent time synchronization, for the first image, we rescale M data into interval [0, 1] by

$XS^{1"} = \frac{XS^1 - \min(XS^1)}{\max(XS^1) - \min(XS^1)}$.

Each time-series data is represented as a colour-mapped horizontal line over a fixed time. However, if some of the continuous data were correlated, the normalization would bias the result as usually datasets have outliers, and that leads to scaling normal data into lowest colour-mapped range. Therefore, the other image depicts the correlation among different data. Examples of continuous data include EEG channels, which we assume will work in our multimodal framework; however, we have left testing of this for future work (Section 11.3).

We standardize correlated M modalities data group to ensure that feature values have a zero-meanμ and unit-variance $XS^1 = \frac{X^1 - \mu}{\sigma}$. After that, we rescale these data into interval [-1, 1], where 0 is the mean, 1 is above average, and -1 is below average. In one particular case, all continuous data included in a modality are independent, such that this image would be unnecessary, and should be eliminated.

6.3.1.1.2 Data augmentation

We need to augment the dataset when it is relatively small; e.g. not enough subjects or variations as compared to other larger datasets used. We do data augmentation based on 4 aspects; rotation, brightness or contrast, scaling, and stretching. In *Rotation*, we specify rotation constraints such as starting at -60 and ending at 60 with 15-degree increments. In b*rightness* or *contrast*, we specify a min/max factor and increment such as [0.2 - 1.5, 0.3], which is used to train the network in different light conditions other than a controlled environment. In s*caling*, we specify shrink and stretch constraints with increment [-10% to +10, 5%]. *Stretching* is followed with cropping to ensure the dimension of an image has not changed.

6.3.1.2 Modality encoding layers

We use a straightforward approach, in which each modality is encoded and decoded separately, such that it is used to build a joint-representation layer. Encoding uses a Convolutional Neural Network (CNN) structure that is simply a CNN feeding an LSTM layer. The LSTM layer is used to model temporal dynamics based on the recurrent layer residing on the top of high-level features extracted by convolution layers. Therefore, it eliminates biased-linear multiplication [144] (fully connected) and reduces the number of parameters with spatial local convolutions.

$$i_t = \sigma(\text{CNN}(x_t)w_{xi} + h_{t-1}w_{hi} + w_{i_b}) \quad (6\text{-}6)$$

$$f_t = \sigma(CNN(x_t)w_{xf} + h_{t-1}w_{hf} + w_{f_b}) \quad (6\text{-}7)$$

$$c_t = i_t \odot \tanh(CNN(x_t)w_{xc} + h_{t-1}w_{hc} + w_{\tilde{c}_b}) + c_{t-1} \odot f_t \quad (6\text{-}8)$$

$$o_t = \sigma(CNN(x_t)w_{xo} + h_{t-1}w_{ho} + w_{o_b}) \quad (6\text{-}9)$$

$$h_t = o_t \odot \tanh(c_t) \quad (6\text{-}10)$$

$$(x^1, x^2, .., x^M) = \{(h_t^{e_1}, h_t^{e_2}, .., h_t^{e_M})\} \quad (6\text{-}11)$$

Each modality can have different structures. CNN is mainly used to preserve spatial locality, eliminate redundancies, and reduce the number of free parameters and memory footprint. At the end, encoding layers are then used to feed an LSTM layer.

The LSTM module is extended to expand recurrent modules over a number of temporal steps. It projects inputs received at each step of time (t) to compute subsequent memory activations. Therefore, it enables 1) training constraints-based generic memory modules to improve memory decoding such as variational smoothness, and 2) preserving layout and locality. At time (t) for instance, activating CNN cells is computed, such that the parameters involved are the feature maps of a modality input after convolution CNN (x_t).

6.3.1.3 Max-pooling layers

In hierarchical networks, a max-pooling layer is used to achieve translation-invariant representation [170], by downsampling latent representation using a constant, typically the maximum value of nonoverlapping subregions. Activating a neuron in a latent representation depends on matching inputs and features in a region of interest, which helps improve filtering.

We use max-pooling layers to allow sparsity in hidden layers. Typically, max-pooling layers only serve in a supervised forward architecture. To circumvent this limitation, we eliminate values that are not maximal in nonoverlapping regions, in order to force expanding feature detectors.

In a reconstruction phase, a sparse latent representation helps generalising and decreasing the number of filters used for decoding per pixel. Hence, there will not be direct necessity for L1 and L2 regularization [164] over hidden units or weights, which aim to resolve overfitting issues and minimize reconstruction errors.

6.3.1.4 LSTM encoding/decoding

We follow the encoder/decoder architecture of spatial autoencoders (Figure 6-10). The encoder (h^e) represents a convolutional recurrent layer followed by hyperbolic linearity ($tanh$) and spatial max-pooling subsampling layer. The decoder (h^d) applies the opposite operations, spatial upsampling, using the nearest neighbours, in order to transform the output to the original input size.

The target is to learn a function f: $x_{1:t}, s_t \rightarrow x_{t+1}$, such that x and s respectively represent the frame and synchronized inputs at time t, or a range of time; from 1: t for instance.

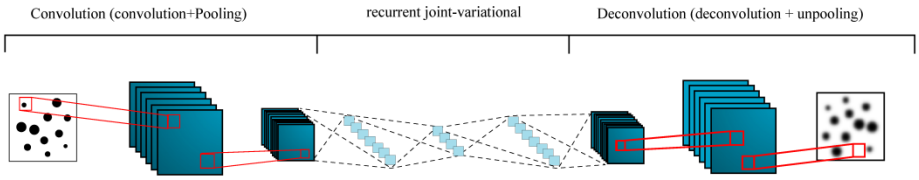


Figure 6-10. Encoder-decoder structure

We use our tool to ensure those different modalities' rates are properly synchronized. For example, the video FPS is 30, which means each frame represents 33 ms of other modalities input. As explained, the main layers in our architecture include encoding, joint-variational, and decoder layers. We have two main encoder/decoder structures based on input modality type.

- *Image/video input*: We use Encoder-Decoder (Type 1) architecture (Figure 6-11). There are 8 weighted layers on the encoder/decoder side, where the first five layers are convolutional, and the rest are fully connected layers that then connect to joint-variational. The first and second convolutional layers are followed by the response-normalization layers, which are followed by maxpooling layers. A kernel from the second to the fifth convolutional layers is only connected to kernels in the previous layer. Kernels of the second and third layers are all connected. Neurons in a fully-connected layer are connected to all neurons in the previous layer.

  Only the first and second convolutional layers intervene in maxpooling and response-normalization layers. The rest of the convolutional layers are still connected to the maxpooling and response-normalization layers. The first convolutional layer filters input image spatial dimensions using stride, such that a stride represents the distance among neighbour neurons in a kernel map. The second convolutional layer receives inputs filtered from the previous convolutional layer, and so on. On the other side, the decoder follows the same structure but instead uses deconvolutional and unpooling layers.

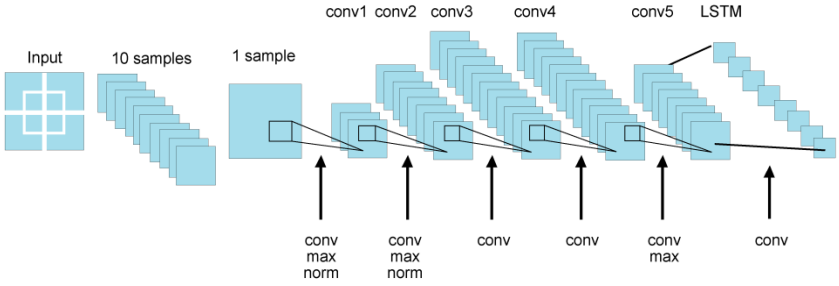


Figure 6-11. Encoder-Decoder Structure Type 1

- *Continuous data input*. We use an encoder-decoder (Type 2) architecture (Figure 6-12). There are 4 weighted layers on the encoder-decoder side, where the first two layers are convolutional, and the rest are fully-connected layers that then connect to joint-variational. First and second convolutional layers are followed by the response-normalization layers, which are followed by maxpooling layers. On the other

side, the decoder follows the same structure but instead it uses deconvolutional and unpooling layers.

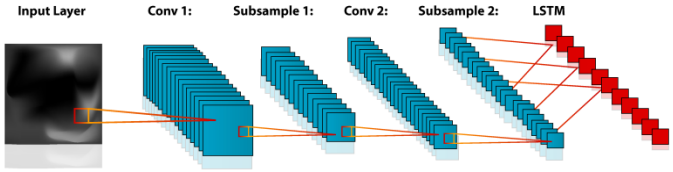


Figure 6-12. Encoder-Decoder Structure Type 2

- *Temporal data input*. We use an encoder-decoder (Type 3) architecture (Figure 6-13), It follows the same structure of encoder-decoder type 2 but it differs in that the sliding window segments the time series, where the height represents the number of input attributes.

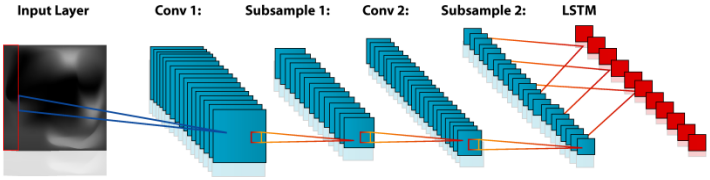


Figure 6-13. Encoder-Decoder Structure Type 3

6.3.1.5 Recurrent-joint-variational layer

Our objective is to be able to fuse and generate input modalities over different time steps, capture temporal changes among them, and enable semi-supervised learning for robust feature extraction. We assume we have two modalities x^1 and x^2 as in Equation (6-12). First, we extract features of each modality using convolutional network to compute the state of LSTM encoder $h_t^{e_1}$ and $h_t^{e_2}$ as in Equation (6-13) .

Recurrence captures motion change and allows predicting subsequent changes. A dense flow transformation (Figure 6-14) map $\mathcal{T}$of the same dimension of their recurrent memory states can be estimated by estimating displacement vector fields or motion trajectories v_t between two consecutive time steps t and $t+1$. $v_t(x,y)$ represents the displacement vector at points x and y at time step t and the corresponding point at time step $t+1$. Tp_t is a trajectory point that starts at location (x,y) in the time step t. Therefore, it can be used to estimate the next motion of each pixel in a static image. To do that, we need the last convolutional layers to be with relatively large kernels ($14x14$ or $15x15$) depending on the modality of the encoder-decoder structure. We use the memory space to estimate the next T consecutive time steps, and then estimate their motion. We apply smoothness constraints using a bilateral filter on consecutive time steps to reduce noise.$\mathcal{T}^m$ is an arbitrary frame that captures motion changes of each modality.$\mathcal{T}^m$ represents motion changes variable, which possesses useful information that can be used to signify the important features of images.

Figure 6-14. Motion estimation mapping

Each one of resulting variables $h_t^{e_1}, h_t^{\mathcal{T}e_1}, h_t^{e_2}$, and $h_t^{\mathcal{T}e_2}$ is independent conditioned on the same joint-representation z as in Equation (6-16). Therefore, it is possible to infer missing modality at different steps of time. However, if the missing modalities are complex, the generation might fail.

$$(x^m, x^{m+1}, .., x^M) = \left\{\left(h_{t_1}^{e_m}, h_{t_2}^{e_m}, .., h_{t_T}^{e_m}\right), \left(h_{t_1}^{e_{m+1}}, h_{t_2}^{e_{m+1}}, .., h_{t_T}^{e_{m+1}}\right), .., x^M\right\} where\ t \in [1,T], m \in 1, M \tag{6-12}$$

$$\left\{\left(h_t^{e_1}, h_t^{e_2}\right)\right\} = \left\{LSTM\left(CNN(x^1), \epsilon, h_t^{e_1}, h_t^{d_1}\right), LSTM\left(CNN(x^2), \epsilon, h_t^{e_2}, h_t^{d_2}\right)\right\} \tag{6-13}$$

$$\mathcal{T}^m(x, y, 2t-1) = v_{h_{t-1}^m}^x(Tp_t),\ \mathcal{T}^m(x, y, 2t) = v_{h_{t-1}^m}^y(Tp_t) where\ x \in [1,W], y \in [1,Y], t \in [1,T]$$

$$Tp = (x, y),\ Tp_t = Tp_{t-1} + v_{h_{t-2}^m}(Tp_{t-1})\ where\ t > 1 \tag{6-14}$$

$$\left\{\left(h_t^{\mathcal{T}e_1}, h_t^{\mathcal{T}e_2}\right)\right\} = \left\{LSTM\left(h_t^{e_1}, \epsilon, h_t^{\mathcal{T}e_1}, h_t^{d_1}\right), LSTM\left(h_t^{e_2}, \epsilon, h_t^{\mathcal{T}e_2}, h_t^{d_2}\right)\right\} \tag{6-15}$$

$$z_t = q_t = q_t(z_t|h_t^{e_1}, h_t^{\mathcal{T}e_1}, h_t^{e_2}, h_t^{\mathcal{T}e_2}) \tag{6-16}$$

$$p_t = p_t(z_m|h_t^{d_1}, h_t^{\mathcal{T}d_1}, h_t^{d_2}, h_t^{\mathcal{T}d_2}) \tag{6-17}$$

$$Aux_t = q_t(x^c|h_t^{d_1}, h_t^{\mathcal{T}d_1}, h_t^{d_2}, h_t^{\mathcal{T}d_2}, aux) \text{ where } c \in [1,C] \tag{6-18}$$

$$\left\{\left(h_t^{\mathcal{T}d_1}, h_t^{\mathcal{T}d_2}\right)\right\} = \left\{LSTM\left(z_t, h_t^{\mathcal{T}d_1}\right), LSTM\left(z_t, h_t^{\mathcal{T}d_2}\right)\right\} \tag{6-19}$$

$$\left\{\left(h_t^{d_1}, h_t^{d_2}\right)\right\} = \left\{LSTM\left(z_t, h_t^{d_1}, h_t^{\mathcal{T}d_1}, y^1\right), LSTM\left(z_t, h_t^{d_2}, h_t^{\mathcal{T}d_2}, y^2\right)\right\} \tag{6-20}$$

$$\{(y^1, y^2)\} = \left\{\left(CNN(y^1, h_t^{d_1}), CNN(y^2, h_t^{d_2})\right)\right\} \tag{6-21}$$

$$l_t(x^1, x^2) = -D_{KL}(q_t(z_t|h_t^{e_1}, h_t^{\mathcal{T}e_1}, h_t^{e_2}, h_t^{\mathcal{T}e_2})||p_t(z_t)) + \sum_{(z_m|m=1)}^{M} q_t\left(z_t\middle|h_t^{e_m}\right) \log \mathrm{p_t}(h_t^{d_m}|z_t) + q_t\left(z_t\middle|h_t^{\mathcal{T}e_m}\right) \log \mathrm{p_t}(h_t^{\mathcal{T}d_m}|z_t). \tag{6-22}$$

$$l_t^{VI}(x^1, x^2) = l_t(x^1, x^2) - \sum_{(z_t|m=1)}^{M} D_{KL}(q_t(z_t|h_t^{e_1}, h_t^{\mathcal{T}e_1}, h_t^{e_2}, h_t^{\mathcal{T}e_2})||q_t\left(z_t|h_t^{e_m}\right)) + D_{KL}(q_t(z_t|h_t^{e_1}, h_t^{\mathcal{T}e_1}, h_t^{e_2}, h_t^{\mathcal{T}e_2})||q_t\left(z_t|h_t^{\mathcal{T}e_m}\right)). \tag{6-23}$$

Variational inference can be used for semi-supervised learning [171], where it can handle labelled (x, y) and unlabelled data (x). It typically consists of encoder $q(z|x, y)$, decoder $p(z|x, y)$ and a discriminative

classifier $q(y|x)$. Encoding f_{enc} is used to encode labelled data to be parameterized using multinomial distribution $Cat(.)$ into latent variable z, while decoding f_{dec} is used to decode a parameterized distribution of image and then estimate the probability to generate input from latent variable z and class label y. However, the results obtained from this way report high classification error [172].

The Auxiliary Deep Generative Model (ADGM)[173] reported better results by extending it to include auxiliary variables to embed more information about class-specific latent distribution $q(y|x, Aux)$. Therefore, we use the same approach but we differ in that encoder-decoder are replaced by LSTM hidden states and joint-representation that have an additional auxiliary variable for each target class c to describe class specific latent distribution between their hidden states and labels as in Equation (6-18). We use a re-parametrization trick [135] and Monte Carlo (MC) to update the latent variables through backpropagation. As reported in [173], approximating expectation using Monte Carlo improves the results.

Recently, efforts have been made to use minimum variation of information learning (VI) to provide robust joint-representation for multiple modalities [174], [175]. As shown in Equation (6-22), we use VI to estimate information distance among each modality and increase what each one knows about others. Then, equation (6-23) follows the lower-bound principle to obtain a lower marginal likelihood bound of the model to make the approximation close to the posterior distribution. Therefore, we would be able to infer the amount of information of other modalities, and to enable synchronization among modalities.y^m is the final reconstructed passing over steps on time T. Then, we use Gaussian distribution to compute the total cost L. SGVB is applied to optimize feature representation for each modality.

It is expected to have lower prediction accuracy with long-term subsequent frames, so a way to circumvent such an issue and stabilize training, is to use a curriculum learning-based multi-step prediction on training data [176]. Hence, we fine-tune the model to have longer-term frames. Figure 6-15 shows the structure flow of the recurrent-joint-variational layer.

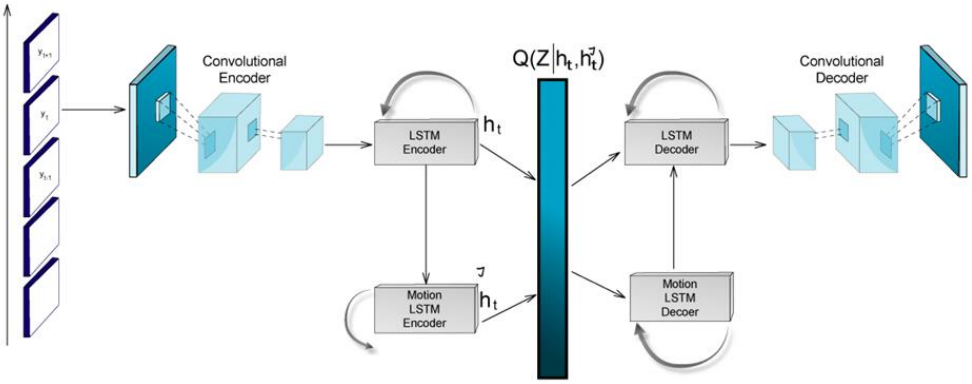


Figure 6-15. Recurrent-joint-variational flow

6.3.1.6 Decoding layers

Convolutional Neural Networks (CNN) have shown effective image upsampling [177]. We use deconvolution, the inverse process of convolution, in which deconvolution kernels map a spatial region of inputs from 1×1 to $d \times d$. Upsampling a spatial region $s \times s$ can be achieved without explicit upsampling of the entire feature map when using a stride s; this requires a smaller number of convolutions making it more efficient.

6.4 Semi-Supervised Training

We initialize DTCN, and use variational inference [171] to train with MCRVAE posterior distribution $p(z|x)$ approximation over its latent variables and $p(z|x,y)$generated by a class y. Hence, we depend on joint representation to infer classification labels. Therefore, DTCN has a classification layer at the end; i.e. Softmax [150]. Transductive SVM (TSVM) can be used with MCRVAE, in order to support semi-supervised learning (Figure 6-17). We use a self-training scheme, in which a model is augmented using labelled data that are confidently predicted from unlabelled data using a self-learnt classifier. This classifier is initially trained against the given labelled data [178]. Such as a process repeats until satisfying a termination condition (Figure 6-16).

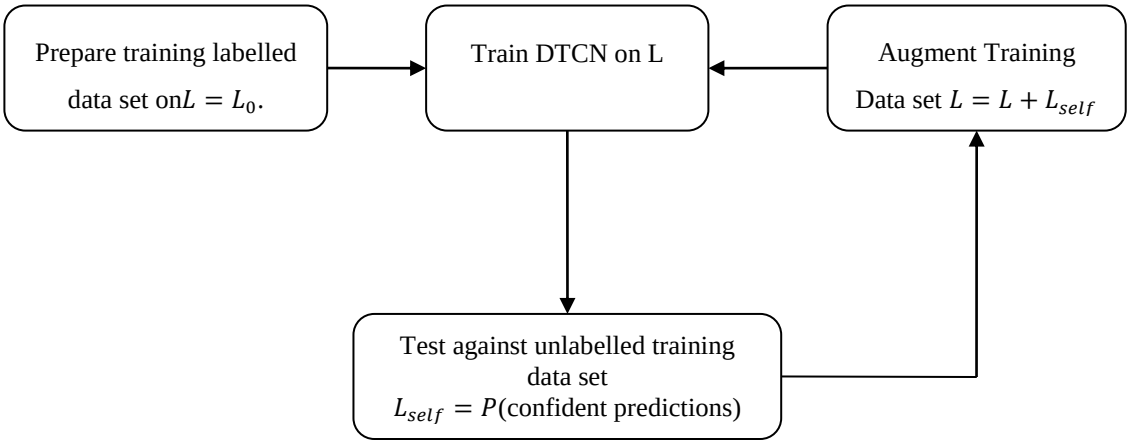
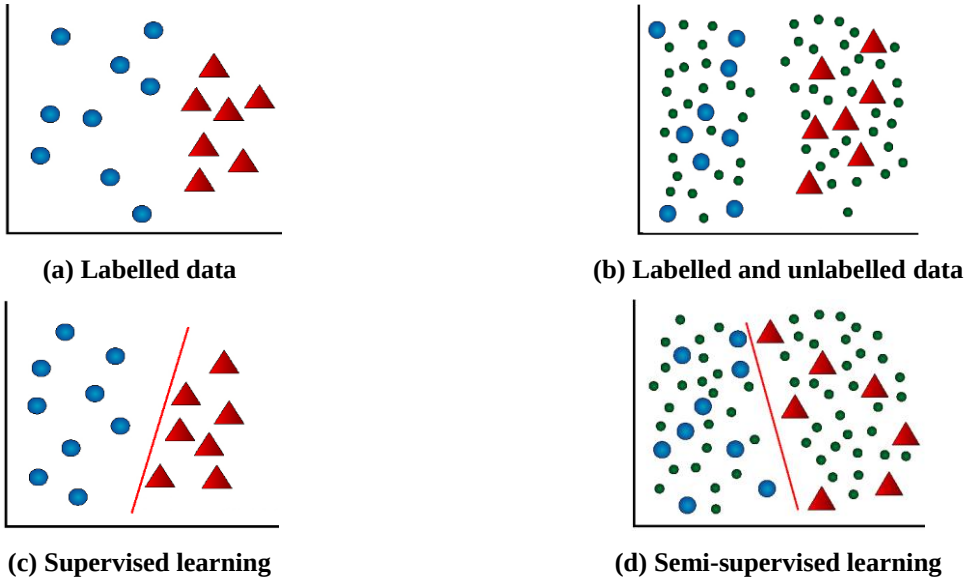


Figure 6-16. Illustration of a self-training scheme

In case of using TSVM, we use regressor chain multi-output regression or classification. Regressor Chain (RC) is a transformation method based on single-target-chaining methods and multiple-label chain classifiers [179]. RC training requires selecting a random full chain (C) of a number of target variables ($V_1, V_2, \dots V_n$), and then building a regression model for each target variable based on the order of the chain selected, such that the first model focuses on predicating V_1, and subsequent models focus on predicting subsequent targets, $V_2, V_3, \dots, V_n$ [180].

A subsequent model is trained on transformed datasets$D_i^* = \{(x_i^{*(1)}, y_i^{*(1)}), (x_i^{*(2)}, y_i^{*(2)}), \dots, (x_i^{*(n)}, y_i^{*(n)})\}$, such that $(x_i^{*(n)}, y_i^{*(n)})$ is a transformed input vector augmented by actual previous values in the chain [180].



(a) Labelled data **(b) Labelled and unlabelled data**

(c) Supervised learning **(d) Semi-supervised learning**

Figure 6-17. Visualization of different learning methods

Classification planes are in red in both c and d

RC is affected by the order of the chain selected. Therefore, updated ensemble versions, Ensemble RC (ERC) [181] are used to overcome such an issue by creating a number of regression chain models of different orders. If the number of distinct chains is less than 10, models are created for all distinct label chains; otherwise, a random chain selection is applied.

6.5 Fusion techniques

There have been many methods for data fusion. Common methods are based on concatenating fully-connected layers among different modalities, which is reported to improve classification accuracy [182]. Concatenation can be applied directly on input or after doing additional operations such as convolution or stacking layers, and then on hidden layers. In other work, instead of concatenation, element-wise production is used to enable multiplicative fusion of different modalities [183].

We implemented two novel fusion techniques, Temporal Feature Fusion (TFF), and Temporal Output Fusion (TOF) discussed as below.

Temporal Feature Fusion (TFF) uses MCRVAE explained earlier, to fuse and compress different modality features along with their motion changes into a joint representation. Such a representation captures abstract features among modalities, as well as their subtle motion changes. However, this still fails to capture long-range temporal changes. Hence, we implemented an additional structure to enable temporal structure modelling.

First, we split a video or sequence of images into a set of clips of fixed time t based on the network structure, where the clip length n is the number of seconds$t \times$ frames per second. We process frames using MCRVAE, and then extract their joint representation features to be a compressed input. Second, we construct a clip set consisting of n modality, where each modality refers to frame features.

Then, the clip set is given to temporal network structure based on each t duration, which we refer to as a time slice (Figure 6-18). In order to avoid excessive computation cost with longer t duration, which becomes even impractical when frames exceed 100, we determine α as a time slice constant, 3 seconds for instance. When frames exceed α, we iteratively construct nested temporal structures that fuse time slices to cover longer temporal structures. Finally, we use the fused temporal structure of fixed-time duration sequences to train a Deep Temporal Credence Network (DTCN).

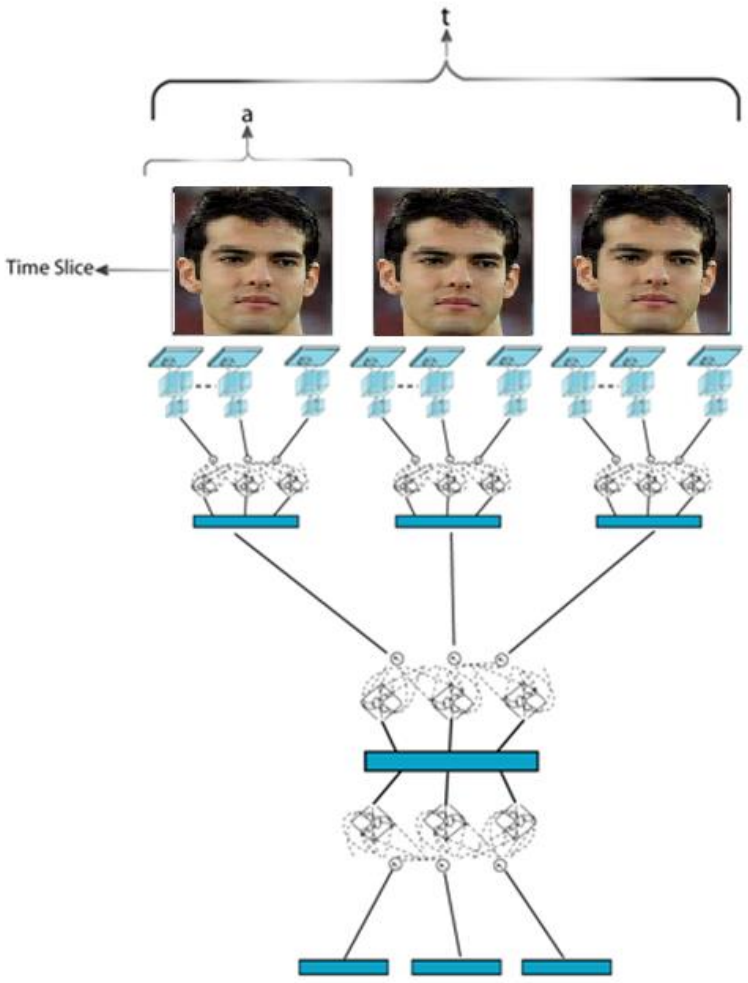


Figure 6-18. Temporal Feature Fusion (TFF) structure

On the other hand, Temporal Output Fusion (TOF) takes place at later stages, where we need to improve the classification for fixed duration. First, multi-output regression or classification is computed. Then, their outputs are encoded as a heatmap image with standardized range for correlated data as explained in Section 6.3.1.1.1. We refer to a signal in this section as the result of either multi-output regression or classification. Multi-output is treated as a correlated group of data. Each preprocessed signal is received as a 2D matrix of dimensions, such that each row represents a signal unit ranging from zero to one, for both discrete and continuous data (Figure 6-19). The height of a 2D matrix is equal to the number of signals, K. The width of a 2D matrix is equal to $the\ number\ of\ seconds \times \#frames\ per\ second$. The number of seconds is equal to the temporal duration we specified for videos. The sampling rate $\mathcal{S}$ refers to the spatial convolved features of a segment of time, while the time step t is a sliding window movement time. Then, we use these image sequences of fixed duration as inputs to train our Deep Temporal Credence Network (DTCN).

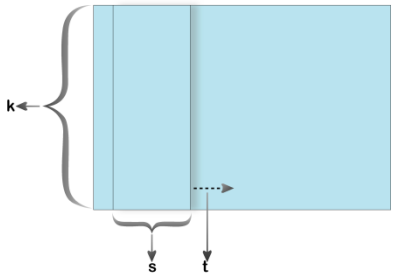


Figure 6-19. Temporal fusion sliding window

6.6 Experiment

In the section, we do experiments to 1) evaluate the missing modalities inference and the model performance. 2) evaluate semi-supervised performance, and 3) inspect the images generated qualitatively. The next chapters will test different aspects of our multimodal fusion engine such as multimodal fusion, and representation visualization. For training, we use Adam optimizer [184], [185] with parameters of learning rate = 3e-5, momentum = 0.9, gradient momentum = 0.95, and min squared gradient = 0.01.

6.6.1 Dataset

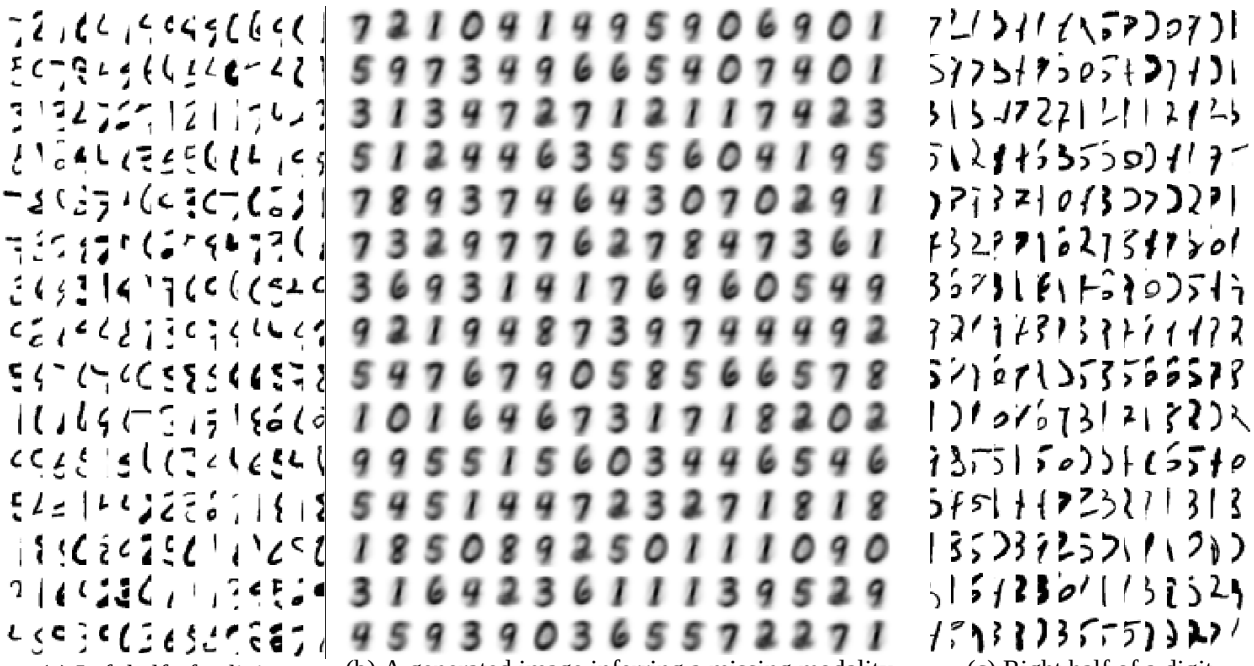
We evaluated our work on MNIST [186] datasets. MNIST is a large dataset of handwritten digits, and it consists of 50k, 10k and 10k of training, validation and testing images respectively.

6.6.2 Evaluation

We evaluate the performance of semi-supervised learning on the MNIST digit classification benchmark. Firstly, we combine the validation and training set into one set. Then, we do a class balanced split of 50000 training points between labelled and unlabelled sets of a size varied from 100 to 3000, such that each class will have the same number of labelled points in order to ensure class balance. We use randomised sampling to create a number of datasets in order to have confidence the bounds of mean performance under repeated draws of datasets.

6.6.2.1 Multimodal evaluation

In this experiment, we augmented the handwritten digit MNIST dataset to have two inputs, in order to test the capability of inferring the missing modalities. Towards that, we split a digit into two halves, left and right, such that the size of each half will be (28×14).



(a) Left half of a digit (b) A generated image inferring a missing modality (c) Right half of a digit

Figure 6-20. The test of generating images from one half

We evaluate our work on different combinations, the left and right halves present, the left half only present, and the right half only present. We report the performance against these combinations in terms of the accuracy of predicting the correct digit and generating the missing half; i.e. modality. We first perform a qualitative inspection in order to evaluate the conditional generation ability to generate a missing modality. Figure 6-21 shows the digit generation that properly infers the missing half of a digit (Figure 6-21- b).

Then, we evaluate the performance of the model with different combination to inspect the accuracy when a modality is missing. Figure 6-21 reports the recognition accuracy when generating missing modalities.

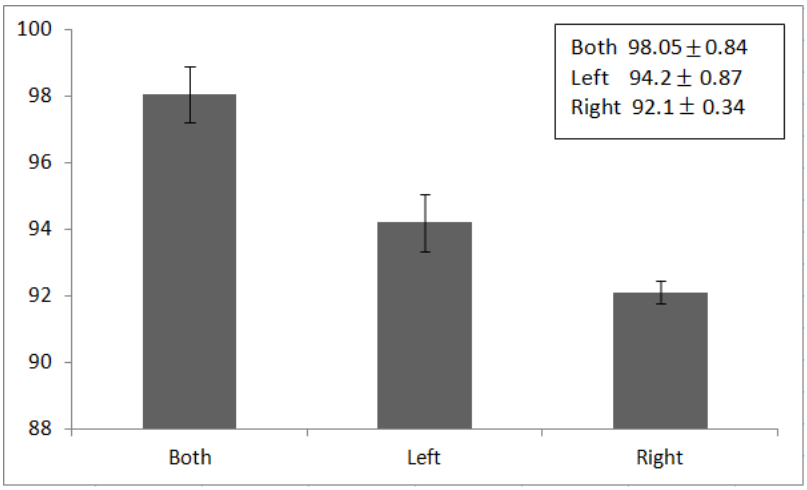
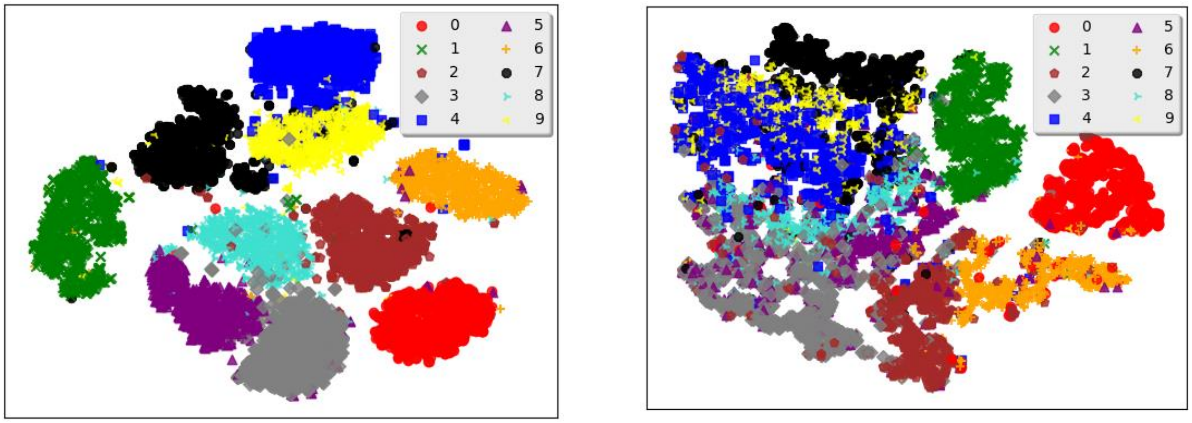


Figure 6-21. Histogram of missing modality recognition accuracy. Error-bars are standard deviation

6.6.2.2 Semi-supervised qualitative evaluation

First, we evaluate MCRVAE obtained joint representation including information about multiple modalities. Figure 6-22 shows an image generation sample. Figure 6-22 visualizes that MCRVAE discriminates more latent representation than VAE and CVAE. It was also tested with labelled data (digit labels) as an additional input modality, which encourages using MCRVAE to perform classification tasks.



(a) MCRVAE **(b) VAE**

Figure 6-22. 2D latent representation

Then, we generate the image corresponding to a conditioned modality with digit labels. Figure 6-23 shows that learnt joint representation works properly and we are able to generate images with digit label attribute.

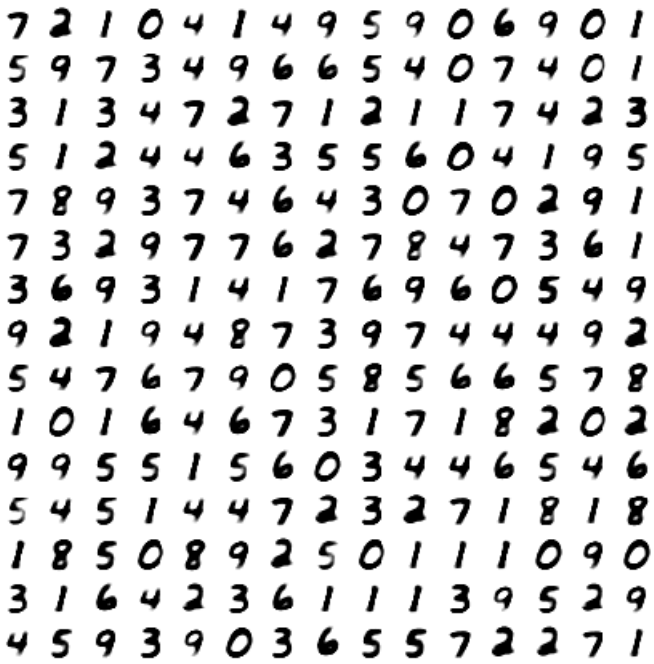


Figure 6-23. The test of log-likelihood by generating images

6.6.2.3 Semi-supervised quantitative evaluation

We test the performance of semi-supervised learning using DTCN. The evaluation is based on a standard digit classification benchmark, MNIST.

The dataset used for semi-supervised learning splits 60,000 (training and validation set) training points between labelled and unlabelled data, such that the size varies between 100 and 3000, and each class has the same number of points labelled in order to ensure balance. We use randomized sampling to create datasets and confidence bounds in order to obtain mean performance for repeated draws of datasets.

Table 6-1. Benchmark for 100 labelled data among different studies

Our approaches are highlighted in grey

Study	**Name**	**Evaluation (SD)**
Kingma et al [171]	Deep Generative Model (M1+TSVM)	11.82 (±0.25)
Pitelis et al [187]	AtlasRBF	8.10% (±0.95)
Kingma et al [172]	Deep Generative Model (M1+M2)	3.33% (±0.14)
Miyato et al [188]	Virtual Adversarial	2.12%
Rasmus et al [189]	Ladder	1.06% (±0.37)
Maaløe et al [173]	Auxiliary Deep Generative Model (10 MC)	0.96% (±0.02)
DTCN		3.2% (±0.10)

Table 6-1 indicates that DTCN shows competitive accuracy in comparison to all of the previously proposed models in the MNIST dataset with 100 labels (Figure 6-24). Furthermore, DTCN is able to maintain temporal information.

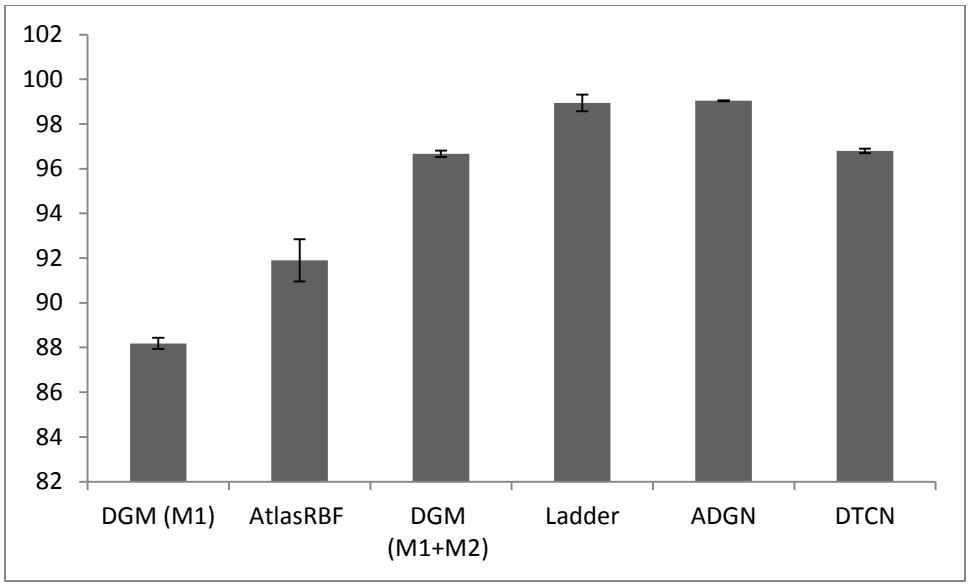


Figure 6-24. Histogram of semi-supervised recognition accuracy. Error-bars are standard deviation

We report the performance after increasing the labelled data size (Figure 6-24). Table 6-2 indicates that increasing the labelled data size leads to a better performance that is relatively close to that with only 100 labelled data items (Figure 6-25).

Table 6-2. Benchmark for 100-3000 labelled data

Labels number	DTCN
100	3.2% (±0.10)
600	2.89% (±0.06)
1000	1.76% (±0.05)
3000	1.12% (±0.02)

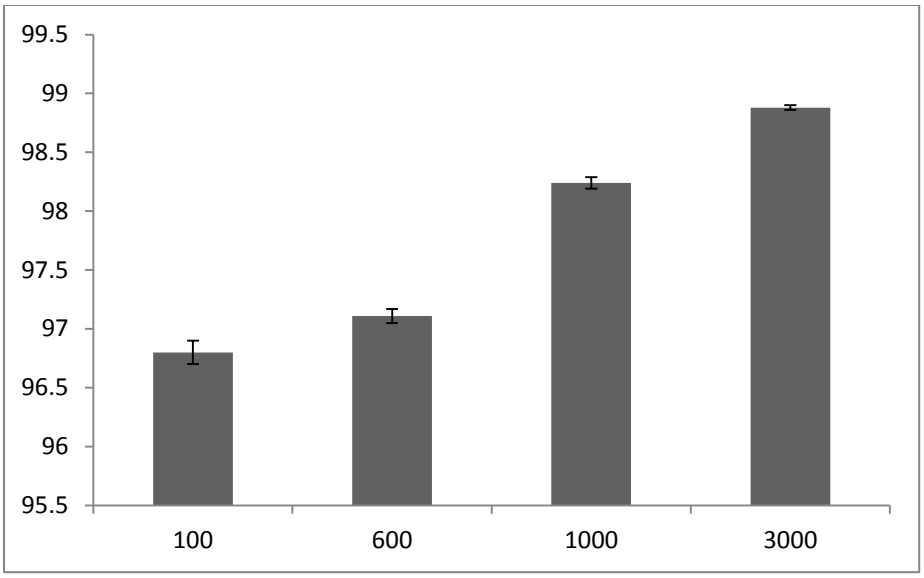


Figure 6-25. Histogram of different labelled data size. Error-bars are standard deviation

6.7 Conclusion

We presented our framework, DTCN, which eases fusion of multiple modalities while capturing temporal information through learning the recurrent joint representation shared among them. The semi-supervised nature of this framework enables training over large unlabeled videos, which can do a supervised task, and is capable of generating missing modalities. However, supporting both multiple modalities and outputs with a semi-supervised variational framework causes models to be computationally expensive. In particular, the need to re-evaluate the likelihood generation for each class during training,

In the next chapters, we will use our framework to implement key modalities, which will be a practical way to test multiple modalities and outputs. As well, we will use of our fusion techniques, TOF and TFF.

Chapter 7 Eye Tracking and Gaze Estimation

Eye tracking and gaze estimation are a crucial part of our research, since we need to link users' affective states and their gaze information. We present a novel method for gaze estimation in the wild, which enables using both of appearance and shape models to improve the performance of machine learning regression techniques. We apply a 3-D HOG and Patch Active Appearance Model (PAAM), in which head positions are represented in the third dimension, and this later can be used to provide calibration-free gaze estimation. We use Deep Temporal Credence Network (DTCN), the deep learning implementation of ours, to enable multi-output regression (Chapter 6) using a semi-supervised training technique. The results, evaluated against multiple datasets, demonstrate that our method shows competitive results to the state-of-the-art methods.

In this chapter, we assume that readers have good knowledge of deep learning. Key terms and techniques referred to in this chapter are defined briefly in the Glossary.

7.1 Introduction

Gaze tracking is a commonly used technique in studies that focus on analyzing the behaviour of users, specifically the focus of attention of user interactions with an application [12]. Gaze estimation is used to determine and track the eye gaze point, as well its direction. The key challenges of calibration-free approaches include 1) estimating an accurate head pose, 2) being person-independent and 3) handling unconstrained conditions; i.e. different lighting. Head movements change the gaze directions and points of interest, so they are required to be directly associated, in order to have an accurate gaze estimation.

The reliance on computer vision does not require physical contact. The regions of interest detected can be used for further processing such usability evaluation.

Gaze estimation, in learning-based techniques, is trained independently away from variations of head pose data [190], such that estimation will not work for specific users or devices. Most of the existing training datasets are collected under controlled conditions [191]–[193], which do not handle variability such as eye appearance and head pose differences.

Relying on monocular cameras for gaze estimation is beneficial, since such cameras are available on all modern devices such as laptops and mobile phones.

Most of the existing gaze estimation techniques are appearance-based, and handle the eye image data as a high-dimensional input. Hence, it is believed to show good results with low-resolution cameras. Such techniques are not often trained against multiple datasets, which make them subject to bias issues related to salient object detection and recognition [194].

In our approach, we use a HOG Patch Active Appearance Model (HOG-PAAM) (Chapter 5) [110] trained against the LFPW dataset [124], in order to predict face shape and appearance model in the wild. PAAM is used to optimize fitting by reducing textures that are not around shape feature points [195]. PAAM efficiency is increased using feature descriptors such as Histograms of Oriented Gradient (HOG) [46]. After that, we apply additional processing steps using HOG-PAAM to normalize the data and then it is used to train DTCN to do multi-output regression, for gaze estimation.

The training process consists of three main stages; 1) head pose model in terms of transformation matrix contains translation, scale, roll, pitch, and yaw; 2) eye model, which includes eye centre localizations along with external and internal corner, and 3) gaze directions.

We train and evaluate our methods against two datasets, Pointing' 04 [196], and Columbia gaze dataset [193].

The contribution of this work is threefold

- We expand the 2D-HOG-PAAM to be 3D-HOG-PAAM, such that head positions will be presented in the third dimension, so they become a part of the model. We use this to validate the accuracy of a calibration-free gaze estimation
- We use Deep Temporal Credence Network (DTCN) to enable multiple outputs and improve accuracy.
- We show an evaluation of state-of-the-art head-pose and gaze-estimation methods on the two datasets mentioned above. The evaluation consists of two parts; head pose, and gaze estimation. We aim to show that the fitting results of our method outperform state-of-the-art methods in terms of accuracy.

7.2 Related work

There are different approaches for eye gaze estimation, such as distinctive-feature-based, model-based, and photometric appearance-based [13].

Distinctive-feature-based methods, as the name suggests, are used to estimate gaze directions from facial distinctive features. For instance, Ferhat et al [197] used iris features to estimate gaze. Similarly, Wang and Sung [198] used images of irises for gaze estimation. On the other hand, Jianfeng and Shigang [199] used pupil centres, and Karakaya et al used the elliptical features of irises [200]. Yoo et al [201] used corneal reflections for eye gaze tracking. Many other studies used corneal reflections for feature extraction [202]–[204].

Model-based methods use a hybrid of feature-based methods. Such methods typically focus on extracting low-dimensional features such as pupil centre and iris contour. Then, they align these features on defined geometric features of shapes such as an eyeball model. However, such methods typically require high-resolution images, or additional hardware such as infrared camera.

For instance, Wood and Bulling [205] used a one-circle model-based algorithm to track gaze iris-contour. Studies that adopt model-based approaches usually use morphable models. For instance, Koutras and Maragos [206] used Active Appearance Model (AAM) for gaze estimation.

Photometric appearance-based methods analyze variations based on the photometric appearance of an entire object being viewed. Lu et al [207] used Adaptive Linear Regression (ALR) for appearance-based gaze estimation. Sugano et al [208] used visual saliency for calibration-free gaze estimation. Yoshimura et al [209] used neural networks for gaze estimation with the consideration of head pose, such that head angles and eye areas are used as learning features.

Appearance-based methods still face challenges in uncontrolled environments, which require calibration to avoid performance issues. Calibration takes place before gaze detection. and most applications do one-time calibration per person [210]. Therefore, calibration-free methods [211] need to resolve person dependency and head motion in uncontrolled environments. For example, subjects are required to hold still without moving their bodies or heads.

Early studies assumed fixed head pose when estimating gaze as opposed to recent research, many of which focused on 3D-head-pose estimation [209], [212] that can be used for in-the-wild estimation. Other approaches include template-based methods such as SVM [213] and geometric-based methods, which use the local features of regions such as eyes and mouth [214].

7.3 Method

2D HOG-PAAM is used to detect and align facial features (Chapter 5), and then extract appearance and shape information; our shape models are built based on 68 landmarks, and trained against the LFPW dataset [124]. We estimate the translation (x, y, z), as well as the rotation vectors between the fit model and mean shape obtained from the HOG-PAAM model. A normalized similarity-free shape (s_0) is constructed from a shape instance (S), as in the equation below, while ignoring parameters such as rotation. After expanding 2D-HOG-PAAM, which consists of 68 facial landmarks, to 3D representation, a raw vector size will be $68 \times 3 = 204$.

$$s = s_0 + \sum_{i=1}^{m} p_i s_i \tag{7-1}$$

In Figure 7-1, first, we use RGB images of head appearances with a fixed 100 x 100 (width x height) size to estimate their pose in 3D space. We use shape information to extract, crop, and wrap eyes image patches, where these patches are rotated to be at the midpoint of an eye corner with a fixed focal length. Each eye's patch is resized to a fixed 60 x 40 (width x height). We encode HOG-PAAM normalized shape model information into binary image masks, which connect and draw facial landmark shapes. Then, we perform histogram equalization to reduce different light condition effects.

Head image, each eyes and encoded shape image ($X^{head}, X^{rightEye}, X^{leftEye}, X^{shape}$) are the modalities that are as inputs to DTCN to predict head pose, and eye gaze directions. We then apply a cross-dataset performance evaluation.

7.4 3D-HOG-PAAM

3D-HOG-PAAM is an extended 2D-HOG-PAAM with head pose parameters. HOG is used as an appearance feature extraction function for facial features of an input image [47]. PAAM defines texture, appearance, and motion models as patches of deformable objects of an input image [195]. 3D-HOG-PAAM estimates a 3D shape model of facial features, as well as their HOG appearance features from a given image.

With 2D-AAM, Gaussian regression can be used to estimate the gaze and pose directions as well as the relations between such directions and AAM parameters [215]. However, their training method requires a sufficient number of images of different variations, which can be impractical to handle for all possible face and gaze variations.

Therefore, in 3D-PAAM, a facial shape overcomes this issue, as head poses are expressed using geometrical transformation, and hence, training data will be manageable. The head pose transformation matrix is predicted in the process, which contains translation, scale, roll, pitch, and yaw. The training and evaluation are discussed in Section 7.7.

A 2D-pose parameter (p) defines a posture change, in which, scale (sz) indicates the model size, and $roll$ indicates the rotation to the model plane, such that both x and y coordinates are translated using a translation function ($trans$).

$$p = [roll, sz, trans(x), trans(y)] \tag{7-2}$$

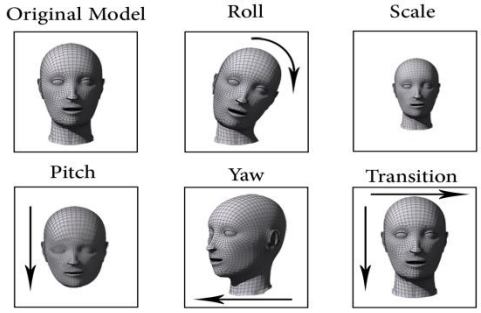


Figure 7-1. An image of different fields of view

In order to expand a 2D-pose parameter s(p) to 3D representation, we add additional arguments, yaw and $pitch$ (Figure 7-1).

$$p = [yaw, pitch, roll, sz, trans(x), trans(y)] \tag{7-3}$$

3D expansion requires depth information to be added into the third dimension (z) using a stereo matching algorithm [216].

$$s = (x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_n, y_n, z_n)^T \tag{7-4}$$

3D-pose parameters Equation (7-5) are used to expand 2D-AAM to these dimensions of different directions, positions, and angles. The pose parameter (p') used for such transformation is defined below, such that p_t represents the coordinates before transformation, sc is a scale defined by the model size (sz), eye position (t) refers to the midpoint of eye corners, and (r) is a rotation function applied on coordinates, x, y, and z.

$$p' = t.sz.r(x).r(y).r(z).p_t \tag{7-5}$$

$$t = \begin{pmatrix} 1 & 0 & 0 & trans(x) \\ 0 & 1 & 0 & trans(y) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7-6}$$

$$sc = \begin{pmatrix} sz & 0 & 0 & 0 \\ 0 & sz & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7-7}$$

A 3D rotation (r) uses Euler angles to define a rotation from the head (x) to camera coordinate systems. The parameters α, β, and γ refer to rotation around z, x, and y axes respectively.

$$r(x) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha.\frac{\pi}{180}) & -\sin(\alpha.\frac{\pi}{180}) & 0 \\ 0 & \sin(\alpha.\frac{\pi}{180}) & \cos(\alpha.\frac{\pi}{180}) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7-8}$$

$$r(y) = \begin{pmatrix} \cos(\beta.\frac{\pi}{180}) & 0 & \sin(\beta.\frac{\pi}{180}) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta.\frac{\pi}{180}) & 0 & \cos(\beta.\frac{\pi}{180}) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7-9}$$

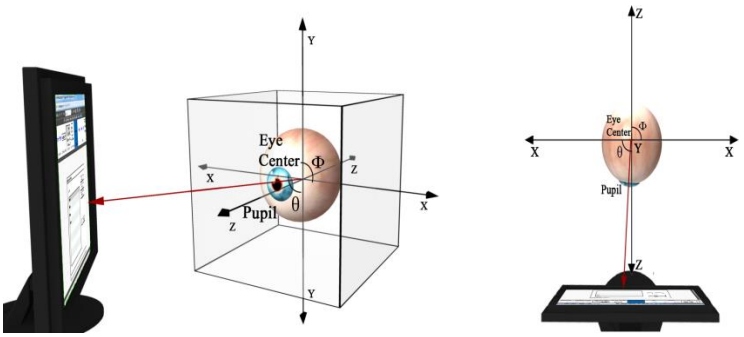
$$r(z) = \begin{pmatrix} \cos(\gamma.\frac{\pi}{180}) & -\sin(\gamma.\frac{\pi}{180}) & 0 & 0 \\ \sin(\gamma.\frac{\pi}{180}) & \cos(\gamma.\frac{\pi}{180}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7-10}$$

An input image still has 2D-repsentation, which means that it is required to use a projection function (G) to project the 3D-PAAM space into 2D space when calculating the error between models.

7.5 Head pose and gaze estimation

We initially use 2D-HOG-PAAM shape information to compute yaw, pitch, and roll head pose parameters using the EPnP algorithm [217] and then expand it into 3D-HOG-PAAM. Then, we do a further refinement on head pose parameters from predicted head pose information. A 3D-HOG-PAAM is used as previously mentioned to construct modality input represented in the head image, both eye image patches, and the encoded facial landmark image.

Gaze direction can be estimated using predicted head and iris positions with respect to the radius and location of eyeballs. Also, eyeball radius can be calculated directly using the fit HOG-PAAM eye model, by getting the difference between the landmark on the eyeball centre and the one on the far left or far right of the eye. Finally, we calculate the anglesθ and ϕ, which indicate the rotation between iris and eyeball centers in the horizontal and vertical directions respectively (Figure 7-2). Both angles define the orientation of the gaze direction.



(a) Front view (b) Top view

Figure 7-2. Gaze direction orientations

In a similar manner, an eye model is able to transform the model to all directions, angles, and positions (Equation 7-5), such that θ and ϕ are substituted for yaw (Equation 7-6) and pitch (Equation 7-10) respectively.

7.6 Feature vectors

We have four input modalities RGB image of the person's head with fixed size (100x100), two eye RGB images with fixed size (60x40), encoded facial landmark image with fixed size (100x100), and initial head pose parameters (T_x, T_y, T_z, roll, yaw and pitch). Roll, yaw, and pitch are then refined with the predicted values. The final output contains head translation and rotation vectors, as well as both eye gaze direction in terms of vertical and horizontal polar coordinates.

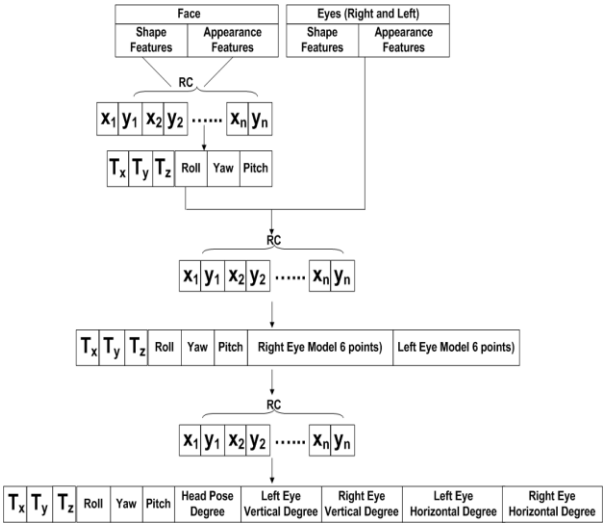


Figure 7-3. Feature Vectors

7.6.1 Hand-crafted features

In this section, we discuss using HOG-PAAM features (Chapter 5) as a baseline of hand-crafted features. The settings used are 8×8 cell size, a block of size 2 and 9 bins, such that we use 36-dimenstional features. We choose the first setting, as it showed better results during our tests. Then, we perform PCA on these feature vectors to reduce the dimensionality and extract abstract features. We refer to these features as planar representation [129] of HOG.

7.6.2 Deep Temporal Credence Network (DTCN)

A DTCN is used to map from the initial head pose parameters, head appearance image, eye appearance image, and encoded normalized shape parameters to an accurate head pose estimation, and eye gaze angles (g) in normalized space. The output is the head pose vector and two eye's 2D gaze angle vector ($\hat{g}$) that consists of two gaze angles, yaw ($\hat{g}_\theta$) and pitch ($\hat{g}_\phi$). All Input modalities are pretrained using encoder-decoder structure type 1 (Section 6.3.1.4), which have the same structure of LeNet network [162]. We pre-process eye appearance images to be grey-scaled. We followed the LeNet structure to derive the parameters of CNN for each image input. The kernel size of convolutional layers are $5x5$ with 1 pixel stride and for max-pool layers are $2x2$. The numbers of features in these convolutional layers for head and encoded shape image are 50, 100, 150 and 100 respectively, and for eye appearance image are 20, 25, 50 and 25 respectively. Then, they connect to their LSTM encoding layer and joint-variational layer. Then, we do supervised multi-output regressor chain training to predict the output.

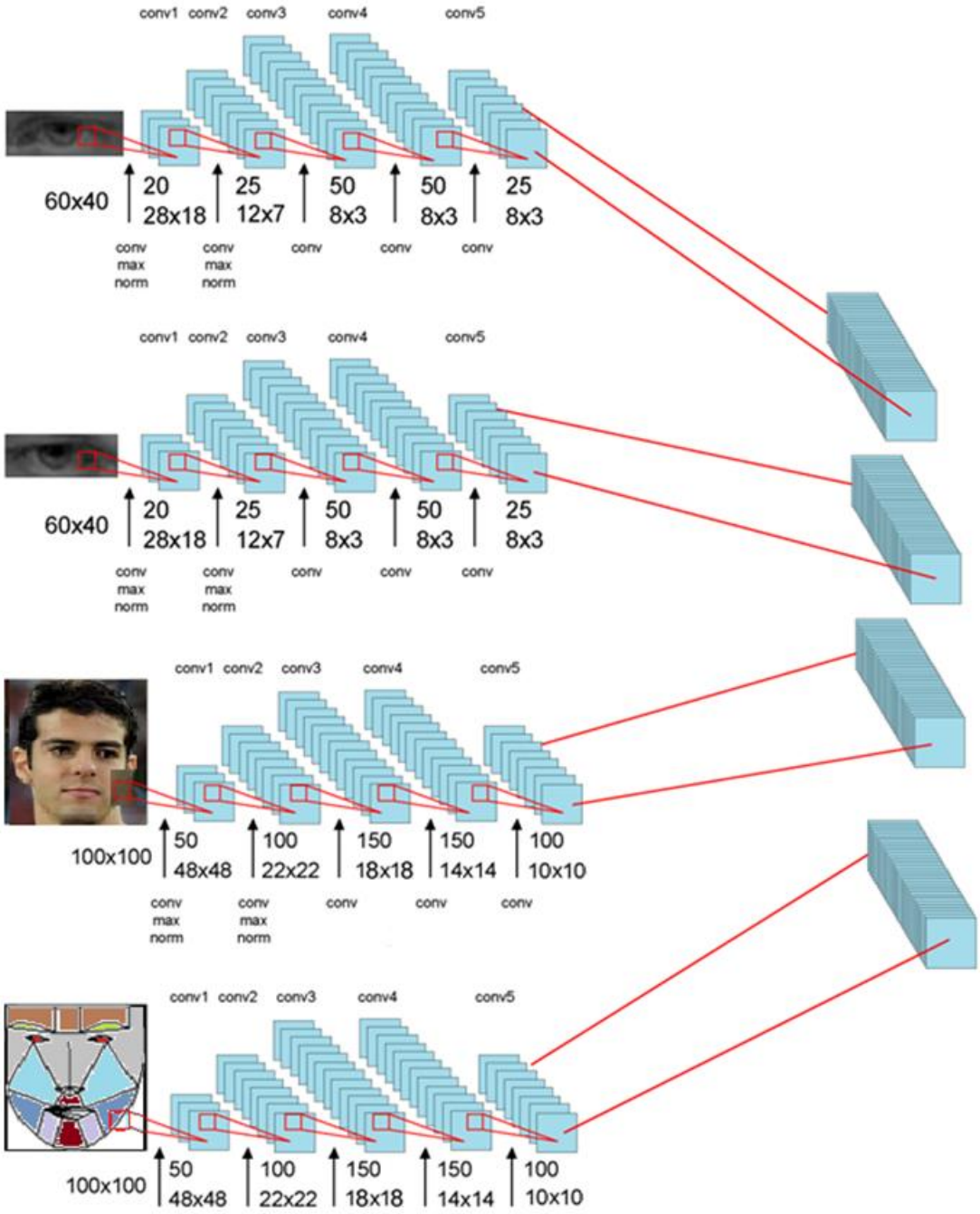


Figure 7-4. Encoder-decoder structure for head pose and gaze estimation

7.7 Experiments

We conduct within-dataset experiments to validate the effectiveness of our approach. We use handcrafted HOG-PAAM with Support Vector Regression (SVR) [218] as baseline. We compare our method with baseline and other state-of-the-art methods on head pose and gaze estimation datasets.

7.7.1 Datasets

For head pose estimation, we use Pointing's 04 dataset [196], which is commonly used for head pose estimation evaluation for the multi-output regression problem. Pointing's '04 dataset consists of 2790 face images for 15 subjects, such that each subject has two 93-image sets of different head poses, yaw and pitch, annotated by bounding boxes describing head and eye regions.

For the evaluation of gaze direction, we use the Columbia gaze dataset [193], which consists of 5880 images of high-resolution for 56 subjects looking at 7×3 gaze points. The dataset represents gaze directions as yaw and pitch angles. Each subject has 5 head poses and 21 gaze directions for each head pose. Subjects are of different ethnicities and have different variations; for instance, 21 subjects wore glasses.

7.7.2 Evaluation methods

We have two validation protocols for each dataset to compare with recent state-of-the-art methods. For Pointing's 04 dataset, [219], [220], we crop the annotated images and resize them into 100×100 pixels, and do 5-fold cross validation on the training dataset. On the other dataset, we apply 8-fold person-independent cross-validation. This means each fold has 49 subjects for training and 9 subjects for testing. It is necessary to make folds person-independent to validate the elimination of calibration and generalization testing.

We use Mean Absolute Error (MAE) to measure error in degrees to perform quantitative evaluation. We use it to estimate yaw and pitch angles degree error. Then, we can calculate the joint angular error, which is the norm of yaw and pitch errors. The joint angular error is used mainly with gaze estimation while we show the average of degree errors for head pose instead.

Table 7-1. Pointing's 04 dataset head pose estimation results (5-fold cross-validation)

Our approaches are highlighted in grey. We use our PAAM as a baseline

Study	Name	Yaw	Pitch	Average
Zhen [221]	Supervised Descriptor Learning (SDL)	4.12	2.09	3.11
Hara and Chellappa [219]	k-clusters Regression Forest (KRF)	5.29	2.51	3.90
Geng and Xia [222]	Multivariate Label Distribution (MLD)	4.24	2.69	3.47
PAAM HOG+SVR (baseline)		5.56	3.76	4.66
DTCN		3.84	1.89	2.87

The results indicate that our DTCN has high estimation accuracy for both yaw and pitch, and shows competitive performance as compared to state-of-the-art algorithms such as SDL [221] with a notably reduced MAE (Table 7-1). The DTCN results show that employing an unsupervised deep learning method to reduce dimensionality and extract features performs competitively when compared to handcrafted descriptors. Note that standard deviations were not published for the compared studies, so we were not able to compute p-values.

Table 7-2. Columbia dataset gaze direction joint-angular results

	Joint-Error
SDSE mHOG+LBP (kNN)[190]	2.69
PAAM HOG+SVR (baseline)	3.72
DTCN	2.11

The results of gaze direction estimation are shown in Table 7-2. Our approach shows competitive results as compared to the handcrafted methods presented in our baseline, as well as [190], which used different combinations of handcrafted features and dimensionality reduction approaches.

7.8 Conclusion

We presented a novel approach that is calibration-free for gaze estimation in the wild. Our approach depends on deep learning, for which we implemented Deep Temporal Credence Network (DTCN) (Chapter 6) in order to enable multi-output regression and semi-supervised training. For calibration-free gaze estimation, we applied a 3D HOG and Patch Active Appearance Model (PAAM), in which head positions are represented in the third dimension.

The evaluation of our gaze estimation shows a substantial performance improvement as compared to person-independent validation protocols, proving the benefit of our calibration-free approach. We use the gaze information to visualise users' affective states on a user interface and enable a detailed visualization methods (Chapter 10).

Chapter 8 Facial Action Units

Automating Facial Action Coding (FAC) units detection is challenging, since existing approaches do not provide solutions that can work spontaneously and generically in wild. In this chapter, we use our framework to overcome FACS detection challenges. The network receives facial images as inputs, and then classifies them into Facial Action Coding (FAC) units. We show an evaluation conducted on public facial action units datasets, DISFA and CK+. We show that features extracted using our framework improve recognition accuracy when used as an independent module. In our evaluation, we demonstrate our method on existing public FACS datasets. in terms of accuracy and F1 and AUC.

8.1 Introduction

A considerable amount of research has recently focused on automating Facial Expression Recognition (FER), since it applies in several fields such as HCI and psychology [223], [224].

Typically, FER relies on Facial Action Coding System (FACS) (Figure 8-1), which is used to recognize facial expressions, caused by combinations of facial muscle movements, as Action Units (AU) [14]. Implementing an efficient AU algorithm that can recognize facial expressions is not an easy task due to the variation of visual appearance and facial attributes, in addition to the high dimensionality of input images, and the lack for sufficient training data. An important factor is that only small parts of a face correspond to an AU.

AU algorithms typically use features, which can be handcrafted, such as HOG and LBP, or geometrical, such as facial landmarks. Such features are not designed for specific tasks, which possibly leads to limitations related to the performance of the classifiers learnt.

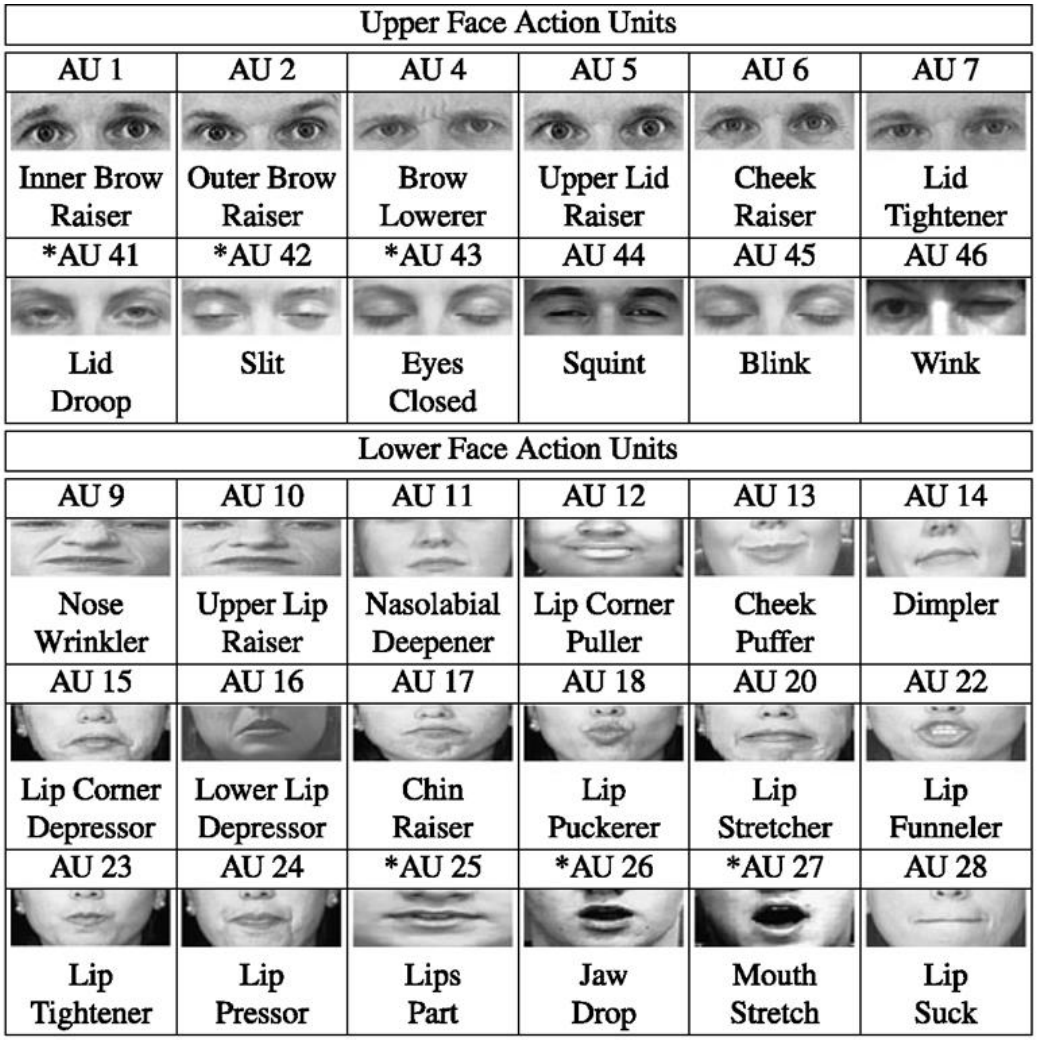


Figure 8-1. Facial Action Units [225]

Deep learning techniques, which generally have better computation capabilities with the availability of GPUs for handling larger images and sequence of images ,can make parallel processing faster. The multistage behaviour in a deep network allows for direct feature learning from pixels. Hence, an algorithm can be directly trained from pixels to labels, which enables features to be tailored for specific tasks. Deep learning however is so far not used to any substantial extent in FER and AU recognition research.

An AU algorithm must handle the necessary facial features to model AU detectors. We evaluate our approach on four facial features, *shape*, *appearance*, *muscle areas* and *dynamic* features can model an AU detector of high accuracy. The performance of a model depends on how to appropriately model and fuse features, so they can be used jointly.

We use our DTCN approach (Chapter 6) to handle the task of FER. We use different input modalities, which are person's face RGB image and encoded binary face shape image from our HOG-PAAM facial landmarks (Chapter 5).

Hence, we can learn relevant information regarding facial shape features, rather than using handcrafted or geometrical features.

We can list our contributions as follow:

1) We use our semi-supervised framework, DTCN to fuse different modalities for improving AU detection.

2) We encode facial shape information along with muscle regions into a binary image to signify the geometric representation of faces. Such features are used as an input image modality.

3) Our evaluation results, conducted against CK+[226] and DISFA [227] datasets, compare our method with other state-of-the-art techniques, as well as those using deep learning methods. The classification of AU is a multioutput and multilabel problem, for which we assign a label for positive and negative intensity.

8.2 Datasets

The CK+ dataset consists of 592 image sequences including both spontaneous and nonspontaneous facial expressions from 123 subjects. An emotion label represents one of seven basic emotions; i.e. the big six of Ekman [44], in addition to contempt. A value, ranging from zero to seven, is assigned to an AU in a frame to represent emotion intensity, such that zero means the absence of the action unit label and seven refers to the maximum intensity. The cited literature does not state exactly how the subjects were able to identify their emotions, but we understand that the recorded emotions were not spontaneous.

DISFA contains 54 (right and left camera) videos of facial expressions captured from 27 participants, while each was spontaneously watching videos stimulating emotions. For each subject, a total of 4845 frames were recorded. AU intensity in a video frame ranges from zero, which means the absence of emotion, to five, which refers to the maximum intensity.

The AUs included in CK+ and DISFA are 30 and 12 respectively (Table 8-1). An action unit type can be either discrete (D) or continuous (C).

Table 8-1. List of AUs in DISFA and CK+

D and C refer to discrete and continuous respectively. Compiled from [226]–[228]

AU Code	Name	Support	Type	AU Code	Name	Support	Type
1	Inner Brow Raiser	Both	C	18	Lip Puckerer	CK+	C
2	Outer Brow Raiser	Both	C	20	Lip Stretcher	Both	C
4	Brow Lowerer	Both	C	21	Neck Tightener	CK+	D
5	Upper Lip Raiser	Both	C	23	Lip Tightener	CK+	C
6	Cheek Raiser	Both	C	24	Lip Pressor	CK+	C
7	Lid Tightener	CK+	D	25	Lips Part	Both	C
9	Nose Wrinkler	Both	C	26	Jaw Drop	Both	C
10	Upper Lip Raiser	CK+	C	27	Mouth Stretch	CK+	C
11	Nasolabial Deepener	CK+	C	28	Lip Suck	CK+	D
12	Lip Corner Puller	Both	C	29	Jaw Thrust	CK+	D
13	Cheek Puller	CK+	D	31	Jaw Clencher	CK+	D
14	Dimpler	CK+	C	34	Cheek Puff	CK+	D
15	Lip Corner Depressor	Both	C	38	Nostril Dilator	CK+	D
16	Lower Lip Depressor	CK+	C	39	Nostril Compressor	CK+	D
17	Chin Raiser	Both	C	43	Eyes Closed	CK+	D

8.3 Method

For each image or frame received as an input, there is a corresponding colour-coded image as an auxiliary input modality, which contains the shape, muscle regions, and appearance features of that input. Our method is divided into 3 steps as follows

1) Detection and alignment of facial points: We use HOG-PAAM to detect faces and align their facial landmarks.
2) Determine facial area of interests such as facial muscles.
3) Encode a colour-coded shape face image.

8.3.1 Binarized shape face Images

We use our PAAM-APPO approach (Chapter 5) to pre-process images and align facial landmarks. Then, we locate and mark fixed facial muscle regions. Studies have found that activities of the corrugator muscle, which controls frowns, vary inversely with the emotional valence of presented stimuli and mood states [229]. Activities of the zygomatic major muscle, which controls smiling, are said to be positively associated with positive emotional stimuli and mood states. The corrugator muscle can also be involved in positive experiences.

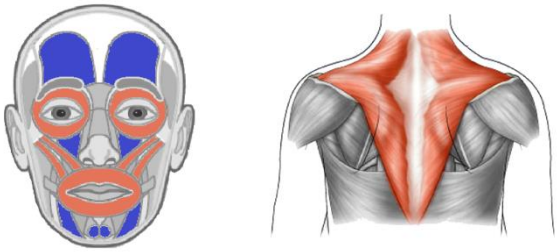


Figure 8-2. Relevant facial and shoulder muscles selected for emotion and facial action units

Table 8-2 and Figure 8-2 show the muscles we selected [230], [231]. A muscle can be associated with multiple facial action unit and emotion indices. For example, if someone has to smile while taking a photo, zygomaticus major will be contracted, and the emotion index will be a social smile. If this person is smiling naturally because of happiness, the emotion index will be smile.

Table 8-2. Facial muscles selected

Muscles	Location	Appearance	Emotion index
Corrugator Supercilii	Between nose and forehead	Eyebrow moves slightly downward and vertical wrinkles are created between the eyebrows	Negative
Zygomaticus Major	At both ends of the lips	• Pulls the lips from both sides • Non-Duchenne smile: gathers skin around eyes. Only zygomaticus major muscle is contracted	Neutral (social smile) Positive (smile) Negative (disgust)
Orbicularis Oculi	Exactly below the lower eye lids	• Eye blink • Duchenne smile: gathers skin around eyes. Both zygomaticus major and orbicularis oculi muscle groups are contracted	Positive (smile)
Frontalis	Forehead	Raises eyebrows and wrinkles	Neutral (social smile) Positive (smile) Negative (disgust)

Then, we construct a binary mask image that draw facial landmarks and connections corresponding to facial muscles such as eyes and mouth (Figure 8-3). Then, we use it to build a colour-coded image for facial regions.

Figure 8-3. Binarized shape face images

8.3.2 Deep Temporal Credence Network (DTCN)

DTCN is used to semi-supervise train multi-label classifier to predict facial action units from shared data features obtained from person's face RGB image and a mono-channel binarized shape information image. The network settings is specified to receive two 100×100 pixel input images. The network settings are specified to receive 100×100 pixel input images. We use encoder-decoder structure type 1 (Section 6.3.1.4).

We use person's facial RGB appearance image and encoded binary face shape image to be the input modalities (Figure 8-4). There are six weighted layers. The first five layers are convolutional, while the last one is an LSTM encoding layer.

The first and second convolutional layers are followed by the response-normalization layers, which are followed by maxpooling layers. Convolutional layers have a kernel of the size $5x5$ with 1 pixel stride and the max-pool layers are $2x2$. The numbers of features in the convolutional layers for person's face and encoded binary face shape image are 50, 100, 150, 150 and 100 respectively. As well, layers are connected to the LSTM encoding layer and joint-variational layers.

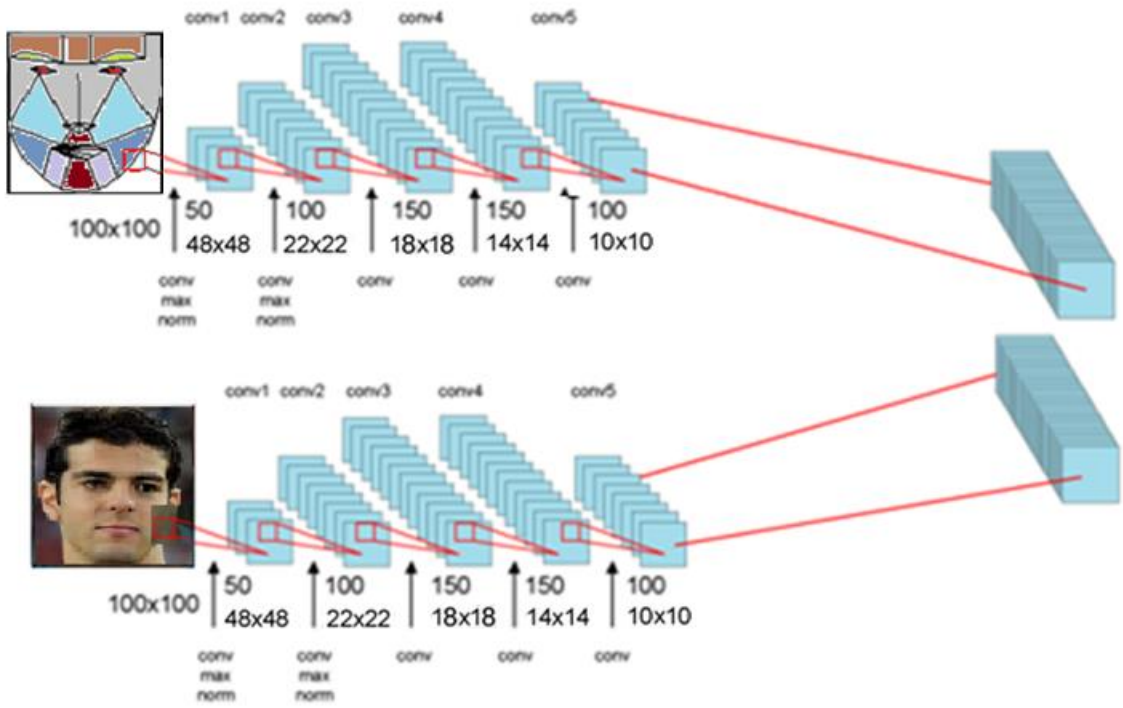


Figure 8-4. Encoder-decoder structure for AU detection

8.4 Experiments

We evaluated our work against two datasets, 1) the spontaneous dataset DISFA [227] and 2) CK+ [226]. We mainly use the DISFA datasets for data preparation and training, while CK+ [226] is used for testing, as CK+ is

considerably smaller than DISFA. For the DISFA dataset, we followed a 3-fold subject-based partition into training, validation and testing sets. Training and validation represents 75% of the dataset, and 25% is used for testing. CK+ is used entirely for generalization testing and is evaluated with the best model trained from the DISFA dataset. We evaluated our work on the presence of facial action units and did not evaluate intensity regression.

8.4.1 Preparation

We process videos or image sequences to create 4-dimensional vectors (time step, image height, image width, and channels), which are used for training and testing our work. For the evaluation, we determined the fixed number of time steps in each segment to be 20, each with 6 seconds duration, hence our frame-rate was 3.33fps. This contrast with 30fps for the DISFA dataset, which has 180 frames in each 6s.

8.4.2 Datasets processing

In DISFA, data is annotated for each second, meaning that we can create a sequence of frames within one second, and hence are able to compare with other state-of-the-art methods. DISFA consists of around 130,815 FACS annotated frames. However, our face detector can detect around 75,000 valid face frames from stereo videos — right and left cameras; 5434 4-dimensions vectors after downsampling for each camera position. DISFA provides 12 AU annotations with their intensities, For AU detection, we mapped every intensity higher than 2 as present.

The CK+ dataset contains around 10,708 valid face images. However, the image sequences vary from 4 to 15 frames. After processing the dataset, this resulted in 327 4-dimension vectors. We only tested CK+ on 10 AU annotations, while AU presence is determined by every intensity higher than 2.

8.4.3 Evaluation metrics

We perform a subject-independent validation and testing protocol, which uses a leave-one-out subject-out testing scheme. We measure the performance of our network on AU detection tasks using accuracy score metrics, F1 and AUC [232]. F1 represents the harmonic means of precision and recall, while AUC quantifies the relations between true and false positives and evaluates the internal quality of the classifier such as skewed sample distribution.

Accuracy metrics are insensitive to the skew amounts in testing datasets [232]. We compare the accuracy, F1 and AUC of the 12 AUs available in both DIFSA and CK+, against other related research that is compatible with the comparison, in terms of the detectable AU and datasets applied (Table 8-3). We compute the average metric over all AUs

8.4.4 Results

In this section, we present the results of our experiments using two paradigms: dataset-specific (Table 8-3) and cross-dataset (Table 8-4) evaluations.

Table 8-3 and Table 8-4 show the results of 12 AUs for DISFA and 10 AUs for CK, respectively. The accuracy average of dataset-specific evaluation (Table 8-3) is close to 90%, and performs better in many AUs in the Facial

Action Coding System (FACS). This is because we fuse features obtained from landmarks for regions that are related to FACS, such as jaw, nose, lips, and so on, Hence the accuracy and detection of such regions is improved since they are visible in the landmarks. Note that we were not able to compute p-values due to lack of information in the cited studies.

Table 8-3. Classification performance of AU detection

A dash means that an AU is not compatible for classification due to dataset structure.

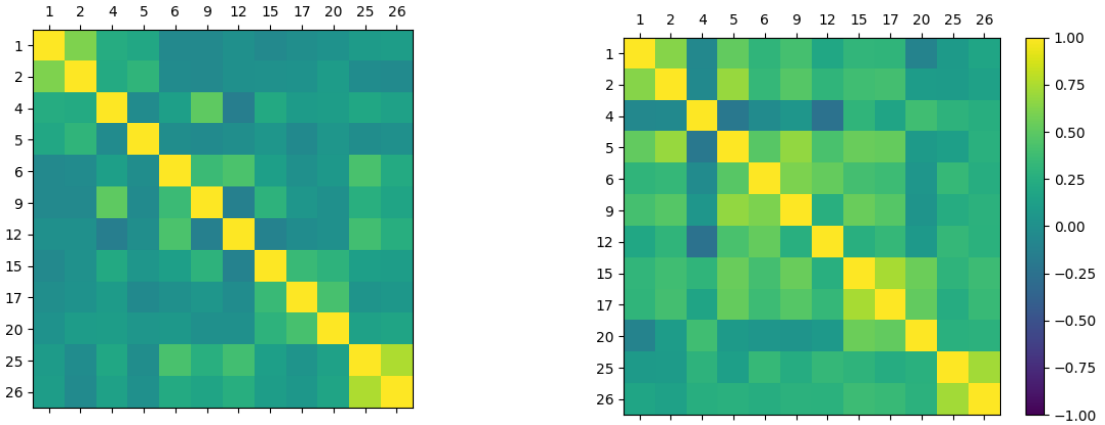
	AU	1	2	4	5	6	9	12	15	17	20	25	26	Average
DISFA (25%)	Accuracy	89.67	91	76.67	93.33	92.67	90.67	91	87	91.67	93.67	-	-	89.74
	F1	88.64	87.32	75.18	89.01	89.52	81.08	86.11	56.18	88.69	69.84	-	-	81.16
	AUC	91.34	93.48	81.96	95.43	94.66	94.17	92.75	92.91	93.81	96.58	-	-	92.71
DIFSA to CK+	Accuracy	90.2	91.6	87.1	90.4	91.1	90.1	88.8	85.2	85.8	90.2	86.3	84.1	88.41
	F1	68.59	66.4	70.75	33.33	78.45	55.2	80.76	58.66	58.96	61.42	87.3	83.89	66.98
	AUC	94.51	95.42	92.36	95.08	94.69	94.73	92.68	91.73	92.09	94.69	87.05	86.43	92.63

For the cross-dataset evaluation, we perform generalization tests against CK+. The overall accuracy is 88.41%. We compared against some of the available state-of-the-art, high performance research (Table 8-4).

Table 8-4. Classification performance of other state-of-the-art of AU detection

Method	Metrics/AU	1	2	4	5	6	9	12	15	17	20	Average
MC-LVM [233]	F1	72.22	85.85	75.05	-	59.94	-	63.45	54.81	73.35	-	69.24
	AUC	92.51	96.60	90.51	-	84.24	-	95.02	87.21	90.82	-	90.99
BGCS [233], [234]	F1	61.11	71.90	67.84	-	65.05	-	80.46	54.23	69.98	-	67.23
	AUC	84.44	91.21	88.21	-	84.91	-	94.54	84.12	84.97	-	87.49
HRBM [233], [235]	F1	66.81	64.52	60.12	-	54.11	-	65.60	60.47	66.67	-	62.62
	AUC	88.88	92.26	81.47	-	88.23	-	94.19	87.91	91.61	-	89.23
QDA-CNN [236]	Accuracy	85.21	86.88	75.58	93.73	85.64	93.42	91.07	85.13	88.14	94.59	87.94
Our method	Accuracy	90.2	91.6	87.1	90.4	91.1	90.1	88.8	85.2	85.8	90.2	**88.41**
	F1	68.59	66.4	70.75	33.33	78.45	55.2	80.76	58.66	58.96	61.42	**66.98**
	AUC	94.51	95.42	92.36	95.08	94.69	94.73	92.68	91.73	92.09	94.69	**92.63**

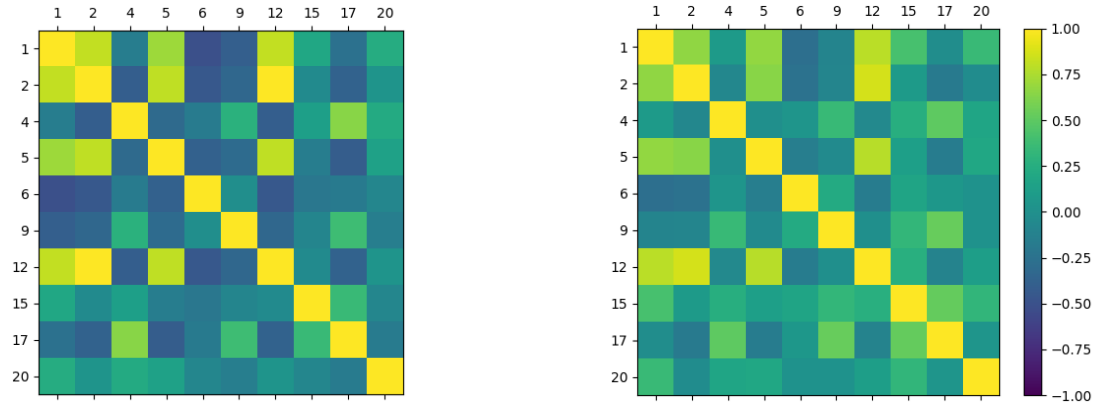
Figure 8-5 and Figure 8-6 show heatmaps of pairwise correlations, among 12 AUs, from ground truth labels (a), as well as prediction labels generated by our approach (b) for DIFSA and CK+ respectively. The development partition of DISFA is visualized using a 75%-25% split. We apply data augmentation on neutral facial expressions, the 13th label, before training and validating a network. We use Phi-coefficient to measure AU correlation.



(a) Ground truth labels (b) Prediction labels generated by our approach

Figure 8-5. DISFA 25% Pairwise Correlations measured among AUs

The heatmaps (Figure 8-5 a and b, and Figure 8-6) are very similar, which indicates that our CNN is able to approximate ground truth correlations — for instance AU1 and AU2 (Table 8-1) are highly correlated in Figure 8-5-b. The neutral label has negative correlation with other action units (Figure 8-5-b).



(a) Ground truth labels (b) Prediction labels generated by our approach

Figure 8-6. CK+ pairwise Correlations among AUs

8.5 Conclusion

We showed how we used our DTCN framework to learn shape, appearance, and dynamic features of regions required for AU detection. Shape and appearance features are learnt from the local image regions and binary shape image associated with input frames. We showed how we use our HOG-PAAM to encode face shape models and compute binary image masks. We also visualized the results to depict that our DTCN obtained better performance than competitive approaches, under different conditions of training and testing.

Chapter 9 Affective and Mental State Evaluation

In this chapter, we aim to verify the accuracy of our framework in terms of automating Facial Expression Recognition (FER) and emotion recognition. We mainly target video for evaluating a temporal FER and emotion recognition from a sequence of images. We also evaluate the fusion of other signals captured using our framework, which include head and eye movements, and affective states.

In previous chapters, we mainly used Temporal Feature Fusion (TFF) to create a robust joint stochastic representation and improve accuracy. In this chapter, we evaluate the other fusion technique: Temporal Output Fusion (TOF), which takes place after receiving fused classification outputs. TOF encodes outputs as a heatmap, and then incorporates temporal changes in order to fuse different classifiers and improve emotion recognition.

We train and evaluate our emotion recognition using several datasets, CK+, and EmotiW 2016 for basic emotions, and the mind reading DVD of Cohen for complex emotions. We test our approach against basic and complex emotion representations.

In this chapter, we assume that readers have a basic knowledge of computing, but may not have knowledge of affective computing. Some terms referred to in this chapter are defined briefly in the Glossary.

9.1 Introduction

A facial expression can be associated with a muscle action, represented as Action Unit (AU), and/or affective state, such as the six basic emotions [57]. Emotion recognition from facial expression depends on decoding a combination of facial expressions into emotions, which is well-described using Facial Action Coding System (FACS) [237].

The recognition of facial expressions and emotions is more effective and natural when captured from a sequence of images as opposed to a static image [238], since facial expressions change over time. The intensity of an emotion or facial expression is modelled dynamically (Table 9-3) determined by temporal transitions, neutral, onset, apex, and offset. The time span of an emotion or facial expression varies among AUs. Capturing temporal transitions is crucial for emotion recognition from videos, for which research uses approaches such as Conditional Random Fields (CRF)[239], Dynamic Bayesian Networks (DBN), and Hidden Markov Models (HMM).

In this chapter, we explore emotion recognition from facial expressions, as well as other nonintrusively-captured signals, head and eye movements, and affective states, which we discussed in previous chapters.

We evaluate Temporal Output Fusion (TOF), which fuses multiple classifiers to improve emotion recognition. We distinguish between TFF, which takes place at the classification stage, and other traditional sensor fusion methods, which take place at early stages.

The contributions of the work in this chapter can be summarized as follows:

1) We contribute a novel fusion technique, in which signals are encoded as a heatmap (Section 6.3.1.1.1), with temporal derivatives determined with 6-second temporal duration.
2) We compare between TFF and TOF fusion, to evaluate the improvement gained in emotion recognition.
3) We evaluate our fusion technique on emotion recognition methods using the abovementioned datasets. Table 9-1 summarizes available public datasets.

9.2 Datasets

We needed to use several datasets in order to allow for recognition of basic and complex emotions (Table 9-1). We also use in-wild dataset such as EmotiW 2016 [15], [16], to evaluate temporal facial expressions from videos [240].

Table 9-1. List of datasets used for emotion recognition evaluation

Type	**Datasets**
Basic emotions	EmotiW 2016 [15], [16], CK+ [226], and KDEF [127]
Continuous emotions	MAHNOB-HCI [241] and DEAP
Complex emotions	Mind reading DVD of Cohen [45]
Temporal sequence	SFEW [242], EmotiW 2016 [15], [16], and FEEDTUM [240]
Static	CK+ [226] and KDEF [127]

9.3 Features

Our signal representation consists of facial expressions, and head and eye movements (Table 9-2). In order to be consistent, we use the same notation as Facial Action Coding System (FACS) to describe signal units. We use the same FACS prefix, AU. The prefixes used for head movements and eye movements HD and EY. The prefixes are used to distinguish signal codes based on signal types. For instance, in FACS, AU51 to AU58 are used to describe simple head-related actions, while our head signal units, HR1 to HR3 are used to describe compound head movements, yaw, pitch, and roll (Table 9-3). The same applies to eye movements EY1 to EY3.

A discrete signal is binary, meaning that a signal unit is either active or inactive, while a continuous signal has a value ranging from zero to one showing the intensity of that signal, which is determined by the muscle. Some approaches calculate the intensity by analyzing the facial animation unit [227], such as measuring the distance between the eyebrows and eyes, or the amount an eyebrow has risen. In our approach, we avoid getting into such complicated calculations, especially as discretization may be imprecise; we only rely on capturing optical movements of muscles associated with action units.

Table 9-2. Signal representations of different modalities used.

A signal type can be either discrete (D) or continuous (C)

Signal Number	Name	Muscle	Type	Signal Number	Name	Muscle	Type
AU1	Inner Brow Raiser	Frontalis	D	AU12	Lip Corner Puller	Zygomaticus major	D
AU2	Outer Brow Raiser	Frontalis	D	AU45	Blink	Orbicularis oculi	D
AU4	Brow Lowerer	Corrugator Supercilii	D	AU46	Wink	Orbicularis oculi	D
AU6	Cheek Raiser	Orbicularis oculi	D	EY62	Eyes Turn Right	Eyeball	C
AU7	Lid Tightener	Orbicularis oculi	C	EY63	Eyes Up	Eyeball	C
HD1	Head yaw	Head	C	EY64	Eyes Down	Eyeball	C
HD2	Head pitch	Head	C	EY65	Walleye	Eyeball	C
HD3	Head roll	Head	C	EY66	Cross-eye	Eyeball	C
EY61	Eyes Turn Left	Eyeball	C	EY68	Eyes Positioned	Eyeball	C
AU11	Nasolabial Deepener	Zygomaticus minor	D				

In Table 9-3, we show some examples of how action units change over time. When action units must appear together or as a sequence, we use the + operator, and when only one of them is enough to appear, we use OR. If an action unit or a number of action units appear repeatedly, we put them within square brackets. We use the → operator to refer to transitioning; i.e. onset, apex, and offset transitions. When a transition → leads to the same action unit(s), this refers to the change of intensity from lowest to highest or vice versa

Table 9-3. Examples representation dynamics using our action units

Action	Action Unit (AU) code	Dynamic
Head yaw (HD1)	AU51 OR AU52	Periodic
Head pitch (HD2)	AU53 OR AU54	
Head roll (HD3)	AU55 OR AU 56	
Persistent head yaw	[HD1]	Episodic
Persistent head pitch	[HD2]	
Persistent head roll	[HD3]	
Lip bites	[AU32]	
Lid tightener	AU7 → AU7 → AU7	
Lip corner puller/depressor	AU12 → AU15 → AU12+AU25	
Lip pucker	AU18 → AU18 → AU18	
Brow raiser	AU1 → AU1 → AU1 OR AU2 → AU2 → AU2	
Teeth showing	(AU12+ AU125) → (AU12+ AU125) → (AU12+ AU125)	
Mouth opening	(AU26 OR AU27) → (AU26 OR AU27) → (AU26 OR AU27)	
AU1 (inner brow raiser), AU2 (outer brow raiser), AU7 (lid tightener), AU12 (lip corner puller), AU15 (lip corner depressor), AU18 (lip pucker), AU25 (lips part), AU26 (jaw drop), AU27 (mouth stretch), AU32 (lip bite), AU51 (head turn left), AU52 (head turn right), AU53 (head up), AU54 (head down), AU55 (head tilt left), AU56 (head tilt right)		

Comprising all data captured from different modalities into signals helps ease and synchronize them together (Figure 9-1).

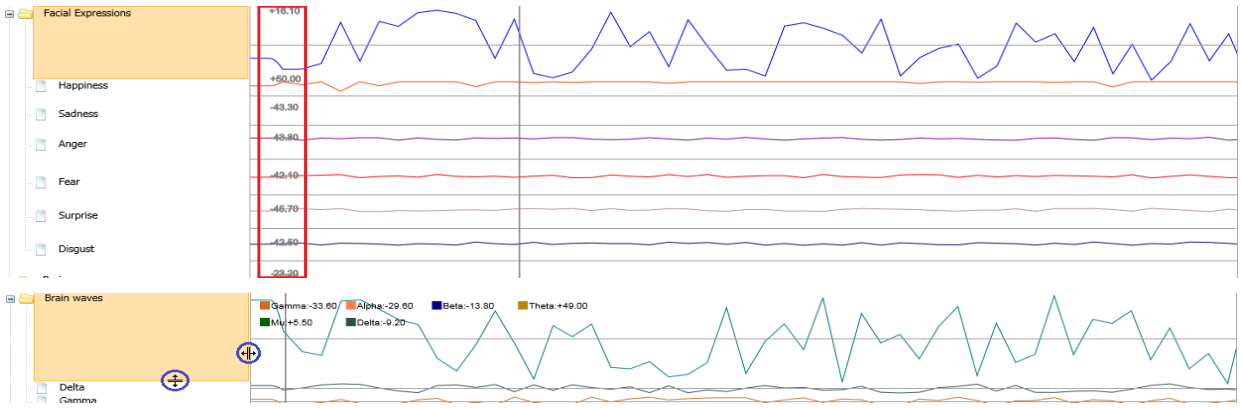


Figure 9-1. An example of signal representation of different modalities and outputs using our tool

9.4 Methods

Extracting dynamics is complicated as they are associated with multiple transitions, action units, and muscles. For instance, opening mouth or showing teeth can be linked with surprise and happiness; these may occur together or individually. Without full understanding of dynamics, an emotion may be misclassified as the other one. Temporal intervals depend on the number and size of each temporal node. For our experiment, we set the temporal duration to 6 seconds and the time steps number to 20.

Active AUs are represented as ordinal latent states corresponding to emotion intensity, while inactive AUs are represented as nominal latent states, since they only refer to neutral emotions or idle targets.

Temporal Output Fusion (TOF), encodes classifier output with their changes onto an image. This image contains discrete and continuous signal data, and can be used with a Convolutional Neural Network (CNN) (6.3.1.4).

9.5 Experiment results

In our experiment, we focus on discrete emotions, either basic or complex. We tried different approaches on different datasets (Table 9-1). The datasets used are CK+ [226] (static images), EmotiW (2016) [15], [16] (for basic emotions), and the mind reading DVD [45] (for complex emotions).

9.5.1 Preparation

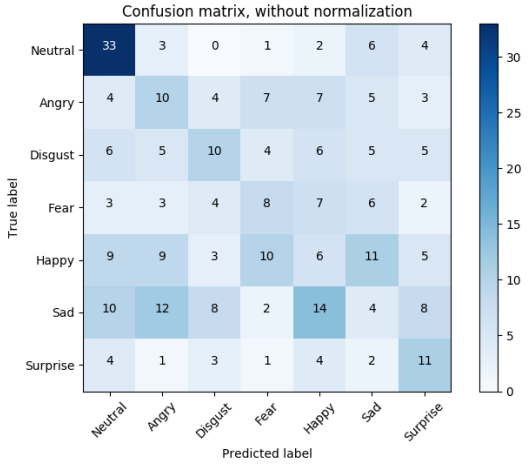
We process videos or image sequences to create 4-dimensional tensors (Section 8.4.1), which are used for training and testing our work. CK+ consists of 327 static image sequences (Section 8.4.2). Therefore, we needed to apply data augmentation (Section 6.3.1.1.2) on CK+ for increasing the training dataset instances, which helps to improve the detection accuracy of CK+ to 97.2%.

Cohen's mind reading DVD consists of 156 short clips that can be used for testing complex emotions, such that the number of videos we used for training, testing, and validation were only 91, 41, and 24 respectively. However, this dataset did not contain spontaneous emotions, since it was captured from actors; in addition, the datasets were quite

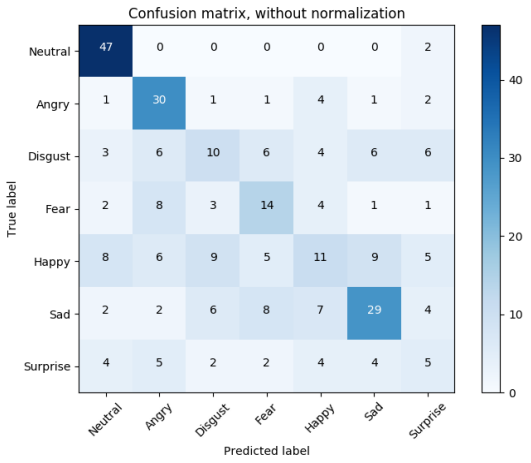
small. We applied data augmentation to transform coloured-images into greyscale, mirroring, and random zoom and rotation. We managed to augment the datasets and increased our test samples into 114 to test different conditions such as lighting.

9.5.2 Evaluation metrics

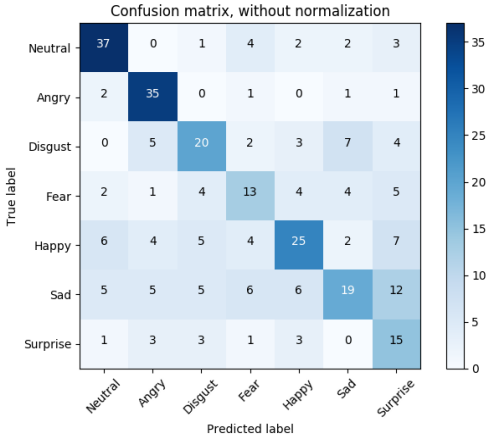
For EmotiW 2016, we have two available training and validation datasets. We train the network on the training set only with 0.3 split ratio, and then we use the other validation dataset as a testing dataset, to evaluate our work. For the other dataset, we followed a 3-fold partition into training, validation and testing sets. Training uses 55% of the dataset, and 45% is used for validation and testing (split equally). We report the performance using F1 and accuracy [232]. We intend to submit our result to be evaluated on EmotiW workshop which properly validate our work in comparison with others.



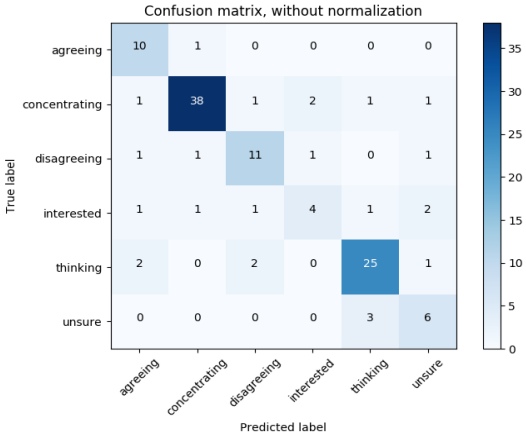
(a) DTCN - Confusion matrix on validation set



(b) TFF - Confusion matrix on validation set



(c) TOF - Confusion matrix on validation set



(d) TOF - Confusion matrix on testing set

Figure 9-2. Confusion matrices results

9.5.3 Results

We do the evaluation on three types of model to compare between our fusion approaches. The first model uses the same encoder-decoder structure for FACS (Figure 8-4) without fine-tuning or pre-training with models, such as AlexNet, VGG [243], and VGG 16 [244]. However, we reported a lower accuracy (27.3% - Table 9-4), as it is trained against a fairly small dataset. Figure 9-2-a shows the DTCN confusion matrix on validation sets.

Then, we applied Temporal Feature Fusion (TFS), which fuses the joint layers of different trained models such as FACS and gaze information. This managed to increase the results to 48.67%. The other trained models such the action unit models were trained on DISFA, which has a relatively large number of images as compared to those in EmotiW and explains the results difference. Figure 9-2-b shows the DTCN confusion matrix on validation sets.

Table 9-4. The evaluation of different models we implemented for emotion recognition

		EmotiW [15], [16]	**Mind Reading** [45]
DTCN	**Accuracy**	27.3	-
	F1	26.6	-
TFF	**Accuracy**	48.67	-
	F1	44.26	-
TOF	**Accuracy**	54.67	78.99
	F1	53.3	71.67

Finally, the Temporal Output Fusion (TOF) reported the best results (54.67% Figure 9-2-c), since it relates the output of different models, and there is a direct relation between facial action units and emotions. However, there is still a room for development and to have better results by having more pre-training data and a larger number of images.

In terms of complex emotions, we evaluated the results on TOF and the accuracy was 78.99% (Figure 9-2-d), which was negatively impacted because of the size of Cohen's data (as mentioned above).

9.6 Summary

In this chapter, we compared our two novel fusion approaches (TFF and TOF) that we introduced in Chapter 6. We reported a high accuracy when evaluated using a controlled-accuracy dataset (CK+). We also tested our results against a publicly available in-the-wild training and validation dataset: We reported competitive results. For usability experiments, users and not put in completely unconstrained conditions, so our results with in-the-wild datasets actually exceed what we needed to achieve for this thesis. We also validated complex mental state emotions against videos in a DVD and obtained competitive results. However, for the latter, we weren't able to directly compare against other state-of-the art works; also the dataset in the DVD does not show truly spontaneous emotions the recorded users were actors.

Chapter 10 Visualization Techniques

The affective states associated with user interactions, fixations, and psychophysiological signals could be difficult to understand, since data are unsynchronized and captured from different modalities, as we showed in the previous chapters. As a part of our research, it is important to overcome such an issue by implementing appropriate visualization techniques, since we use a large number of modalities making it almost impossible to understand signals captured.

There are common visualization techniques such as heatmaps, navigation models, and scanpaths. However, those techniques do not link the affective states with visualization.

In this chapter, we present novel visualization techniques, 3D heatmap, 3D scanpath, and widget heatmap. In a 3D heatmap, the third dimension represents the polarity of User Emotional Experience (UEX) in terms of satisfaction or dissatisfaction. The convexity or the concavity of a 3D heatmap is determined, such that regions with positive emotions will have peaks, and regions with negative emotions will have valleys

We illustrate a way to show a sequence of heatmaps for user interactions as opposed to the traditional ways that show a static image. In this approach, while a user is interacting with an application, we capture a heatmap for each distinct state of the user interface, such that we end up with a sequence of heatmaps, where each heatmap represents temporal user interactions.

The same idea of 3D heatmap applies for 3D scanpaths. On the other hand, in a widget heatmap, interactions and fixations are displayed on an entire widget as opposed to a single pixel.

The annotation boards we showed in Chapter 3 are also an example of the visualization techniques we present in this thesis.

10.1 Introduction

A heatmap is a 2D graphical representation, in which values are represented as colours, which makes it easier to understand and spot certain parts of an image. Heatmaps are a known way to represent user interactions based on activities such as mouse clicks and movements. The colours are used to represent the intensity such as the frequency of activities; for instance, red and green can indicate high and medium frequencies respectively.

Heatmaps are however limited in terms of representing the true nature behind user interactions, such as satisfaction level. For instance, a heatmap may represent the places in UI that a user interacted with the most, but it cannot determine which places led to a negative or positive user experience.

Existing heatmap tools usually assume that there is a single static heatmap representing a certain UI state of user interactions.

Motivated by the above limitations, we introduce an advanced heatmap technique, 3D heatmap, in which the efforts and actions of users are represented on the first two dimensions, and emotions are represented in the third dimension (Figure 10-1).

For each UI state, a new 3D heatmap is created. A UI state refers to a distinct representation of UI controls, for example being in a certain dialog. User interactions are tracked and assigned to the heatmap corresponding to the given UI state.

The heatmaps of a user's session can be used for tasks such as usability evaluation. For instance, an evaluator becomes able to find out the places in the UI that have notable negative user emotions, which means that they should be investigated for potential usability issues.

Cognitive effort can be estimated from user interactions or eye movements. The latter requires specialized software and use of a camera. Heatmaps are often generated from such eye movements, specifically eye fixations and saccades. Fixations refer to the eyes focusing on specific locations on the screen, while saccades refer to moving the eyes around between fixations.

We will refer to eye movements, or activities such as mouse and keyboard events as actions. Such actions suggest effort spent over time. For instance, more actions likely suggests more effort, and vice versa.

The third dimension requires capturing additional inputs. In this chapter, we rely on facial expressions to assess emotions, but we will not discuss their implementation, as this is not central to the 3D heatmap concepts.

Heatmaps are useful to determine the frequencies of actions on certain widgets (also known as controls) of the UI. To maximize accuracy, we introduce a new heatmap, the widget heatmap in which frequencies are applied on UI widgets as opposed to pixels.

Our visualization techniques require clients to allow video recording when starting a session, as this is crucial for nonintrusive tracking. Without video recording enabled, the recorded session will only show user interactions but without emotions. In such a case, navigation traces can still be used, but traditional heatmaps and scanpaths will be used instead of the 3D versions.

Users are not restricted to using certain types of camera, such that they can use web cameras available in normal personal computers and laptops. The accuracy is expected to be better when using cameras of higher quality.

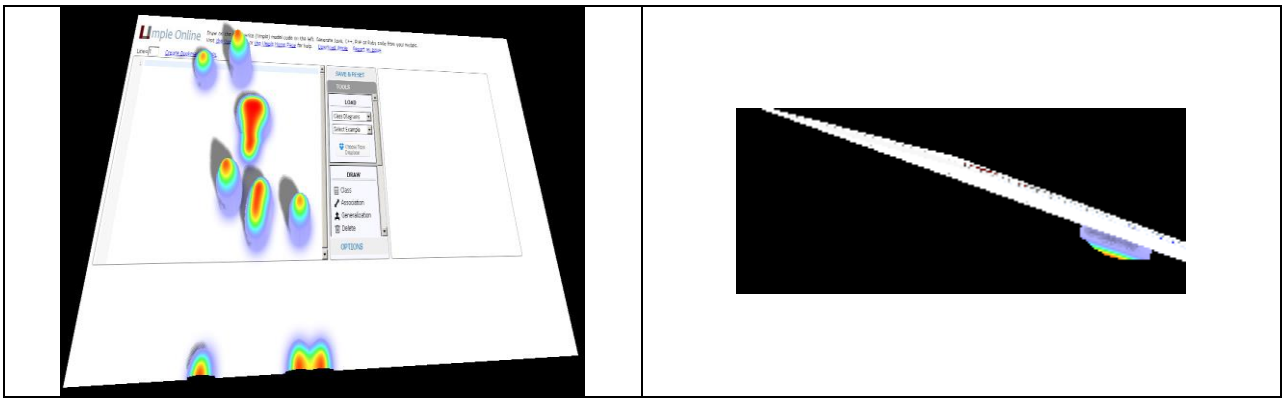
Our contributions can be summarized as follows

- We demonstrate unique 3D heatmap and scanpath visualization techniques that show emotions as well as user interactions and eye movements.
- We demonstrate widget-based visualization. To the best of our knowledge, there is no study discussing heatmaps on widgets, rather than specific attention points.

– We introduce dynamic heatmap generation while at the same time, reducing the number of heatmaps generated by combining similar ones.

This chapter is organized as follows. In Section 10.2, we discuss the basic heatmap concepts and implementation details. In Section 10.3, we will show how we handle the creation of multiple heatmaps. In Section 10.4, we will introduce our widget heatmap approach. In Section 10.5, we will discuss the emotion model we use. In Section 10.6, we will discuss our new 3D heatmap approach. In Section 10.8, we discuss the related work.

It is important to mention that the visualization technique we are describing here is used to highlight a user's interaction with a certain UI. Hence, when we see a visualized image (Figure 10-1), it reflects a view of a UI with overlays based on the techniques applied. An overlay is used to link emotions with a certain location, such that valleys (b) are used to represent users' negative emotions, and peaks are used to represent users' positive emotions (a). The shadows used in Figure 10-1 are for aesthetic reasons only.



(a) a front view of a heatmap example (b) a back view showing one of the valleys

Figure 10-1. A 3D heatmap example

10.2 Heatmaps

A heatmap represents actions at a certain UI state captured as a background image (B) of $h \times w$ size.

An action (A_i) appears at coordinates (x, y), and it is visualized as a circle-like shape that has a radius, which increases based on the action sensitivity (S). An action index i is used to determine that action's order among other actions; i.e. A_{i-1} comes before A_i, and A_{i+1}comes after A_i An index i is also used with the action intensity (I_i) and transparency (T_i).

An action can refer to an eye fixation or saccade, or can refer to a user action such as mouse click or movement. In all cases, an action starts with a location, with which the users interacted; i.e. a location a user fixated their eyes on, a widget a user clicked with the mouse, and so on. An action ends when the user’s attention changes to a different location, such as fixating eyes on a different place or clicking on a different location.

An action has duration A_D, which makes more sense for eye fixations. Eye saccades and mouse movements have minimum durations (M), since they occur quickly. The minimum duration (M) is the same as the value set for heatmap brightness. Heatmap brightness must have a valid nonnegative integer value.

A 2D heatmap is sufficient to represent the locations and durations of actions, but not sufficient enough to represent information such as user's emotions (see 3D heatmaps in Section 10.7).

We need to define additional rules to set durations for some actions. For instance, mouse clicks appear rapidly but they have high impacts similar to long eye fixations.

We calculate intensity (I_i) of a pixel (x, y) at a (D) distance away from an action (A_i) as below.

$$D = \sqrt{x^2 + y^2} \tag{10-1}$$

$$I_i = e^{\frac{D^2}{2S}} \tag{10-2}$$

We as well calculate the transparency T_i as shown below. This is done accumulatively, since T_i depends on the value of (T_{i-1}). The initial value T_0 is equal to the opaqueness value (P). If P is 100, the background image (B) will be completely hidden, and if P is zero, B will be completely visible.

$$T_i = \mathrm{T_{i-1}} + \begin{cases} \left(\frac{\mathrm{S - D}}{\mathrm{S}}\right) . \left[\frac{\mathrm{A_i - (P.M)}}{\mathrm{M}}\right] . \mathrm{I_i} & \mathrm{D \leq S} \\ 0 & \mathrm{D > S} \end{cases} \tag{10-3}$$

The transparency of a pixel (x, y) of an action A_i follows the equation above, but with the assumption that D is equal to zero.

$$T_i = \mathrm{T_{i-1}} + \left[\frac{\mathrm{A_i - (P.M)}}{\mathrm{M}}\right] . \mathrm{I_i} \tag{10-4}$$

Intensity calculation depends on the distribution function used such as linear or Gaussian functions. The sensitivity in a Gaussian distribution is equal to (σ^2), which refers to the distribution variances.

$$T_i = \mathrm{T_{i-1}} + \left[\frac{\mathrm{A_i - (P.M)}}{\mathrm{M}}\right] . e^{\frac{D^2}{2\sigma^2}} \tag{10-5}$$

Typically, intensity is used with Gaussian functions to control smoothness as shown in the equations above. In linear functions, when smoothness is not applied, intensity is equal to one, such that T_i will be calculated as below.

$$T_i = \mathrm{T_{i-1}} + \left[\frac{\mathrm{A_i - (P.M)}}{\mathrm{M}}\right] \tag{10-6}$$

Figure 10-2. A colouring range example

T_i has a value that ranges from zero to one, and it is assigned a colour from a colouring scheme (CS) based on that value. A colouring scheme (CS) refers to some settings that we use to distinguish intensities based on colours. For example, if a $CS = \{Red, Blue, White\}$, a pixel colour will be in a range of the colours shown in Figure 10-2.

A frequency map is constructed for actions in a heatmap image (B). In order to decrease the size of a frequency map, an image can be represented as patches of pixels.

Actions occur over time at different UI states, such that a new heatmap is constructed to represent a certain UI state.

After that, an action map is created from the frequency map and Gaussian filter (Figure 10-3 and Figure 10-4), by applying the equations mentioned above. Complexity is $O(Nhw)$, where N represents the number of actions. If the value of $h = w = n$, complexity will be $O(Nn^2)$.

The action map is applied to a heatmap image (B) accordingly.

```
1  Input: actions;
2  foreach (UI_state in UI states){
3      heatmap= startHeatmap(UI_state)
4      frequencyMap=
5          constructFrequencyMap(heatmap, actions)
6      actionMap= constructActionMap(frequencyMap)
7      appslyActions(actionMap, heatmap)
8  }
```

Figure 10-3. Heatmap algorithm

An output example is in Figure 10-4

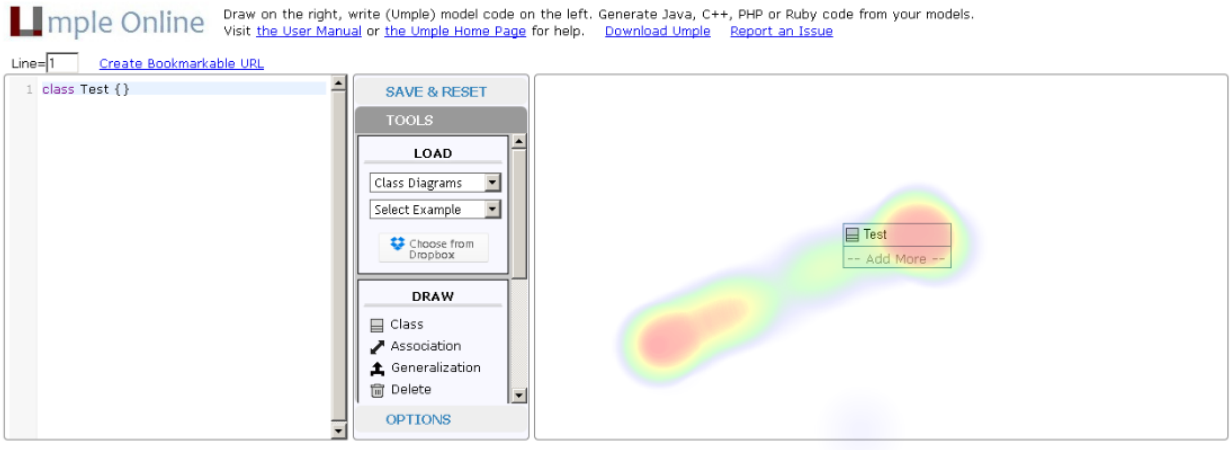
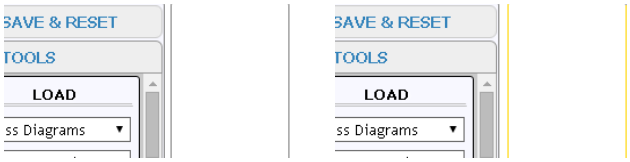


Figure 10-4. A 2D heatmap example

10.3 Sequence of heatmaps

A heatmap typically represents a single UI state. Switching among different states occurs quite often, even for the tiniest UI changes. As a result, numerous heatmaps will be created. Many of the heatmaps created may look similar, and thus should be combined. For instance, Figure 10-5-a has a black border and Figure 10-5-b has a yellow border; such a difference is so small, such that we can combine the actions of both figures, and only use one figure, a or b.



(a) (b)

Figure 10-5. Images with high similarities

To overcome this issue, when entering a new UI state, we compare the image of this state and the previous one. If the difference in this comparison exceeds a threshold (Line 5), we start a new heatmap; otherwise, we keep recording actions on the current heatmap (Figure 10-6). We can change the threshold value in order to reduce or increase the number of heatmap images generated. We selected the value of 5, which is relatively small, in order to only ignore images that are too highly similar. There are many algorithms to compare the degree of similarity between images based on salient points such as SURF [97], SIFT [118], and MSER. We selected SURF in our work, since SURF is one of the commonly used algorithms to detect image similarities [97].

```
1  Input: currentUI_state, newUI_state
2  Number threshold= 5
3  Image image1= toImage(currentUI_state)
4  Image image2= toImage(newUI_state)
6
7  if(compare(image1, image2) > threshold){
8      return image2;
9  }
10 return image1;
```

Figure 10-6. Image comparison

10.4 Widget heatmaps

In a widget heatmap, each widget (i.e. UI control) has a frequency that starts as zero. When a new action occurs at a widget, the frequency of this widget will increase based on the weight of this action, such as eye fixations versus eye saccades. An overlay image will be placed on any widget of nonzero frequency (Figure 10-7).

As opposed to normal heatmaps, a distribution function is not applied.

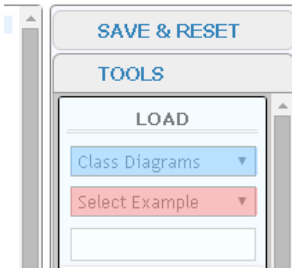


Figure 10-7. A widget heatmap example

The colour (c_i) of an overlay image will represent the frequency (F_i) of its widget (i) when compared to the total frequency (F), such that.

$$c_i = F_i/F \quad (10\text{-}7)$$

The colour of the intersection of overlay images is the average colour of those overlay images.

10.5 Scanpaths

Scanpaths are useful to visualize the eye tracking data captured using our framework (Chapter 7).

Noton and Stark introduced the term scanpath for the first time in 1971 as a way recognize the patterns of eye movements that occur during saccades [29]. A scanpath describes a visual pattern or the sequence in which eye movements are represented.

In our scope, a scanpath shows a path of eye fixations, as well as mouse interactions, such that each focal area has a labelled circle (Figure 10-8). The bigger the diameter of a circle is, the longer the length of the eye fixations or events are. For example, in Figure 10-8, the user dragged and dropped a diagram node, which is indicated in the circles 1 and 2; both circles have the biggest diameter in the scanpath. After that, they fixated their eyes for a few moments at places labelled as 3, 4, and 5. Finally, they fixated their eyes for a longer moment onto another place on the screen labelled as 6.

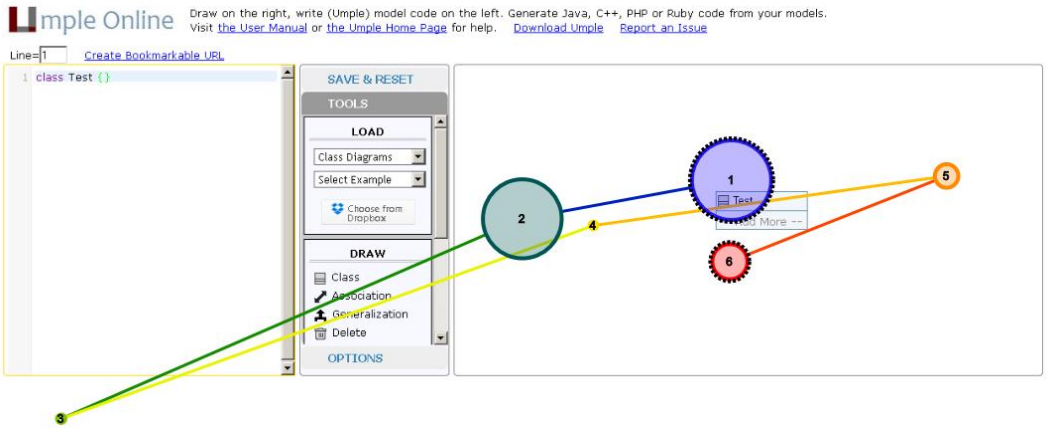


Figure 10-8. Another scanpath example

10.6 Emotion models

An emotion model focuses on linking between user emotion experience (E) and the effort (F) spent when using an application. This can help us determine if the effort was increased due to negative emotions, meaning potential usability issues, or due to neutral or positive emotions, meaning otherwise.

Model information is captured from eye movements and facial expressions. Total effort (F) consists of mental effort (F_m) and physical effort (F_P).

$$F = F_m + F_p \quad (10\text{-}8)$$

Mental effort (F_m) is calculated as an aggregate function, in which information is captured from different apparatus (A) such as eye trackers. Physical effort (F_p) is calculated in a similar manner, and it depends on activities such as head movements. The calculated effort varies based on the apparatus used. The equation below is similar for both mental effort (F_m) and physical effort (F_p). However, both types of effort can still have different values, since each one has different apparatus (A) and actions (f_s), such as fixations, that occur at a time segment (S), which cause effort to increase over time (t)

$$F_m(A) = F_p(A) = \int_{t0}^{t} \sum_{s=0}^{A} f_s(t)\, dt \tag{10-9}$$

The emotion (E) engagement model applies a formula based on the emotions extracted, facial expressions in our case. This depends on categorizing emotions into negative and positive emotions. An emotion engagement (E) ranges from -1 to 1, to represent emotion intensity, such that negative experience has negative values, neutral engagements are set to zero, and positive experience has positive values (Figure 10-9).

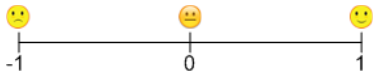


Figure 10-9. An emotion engagement range

10.7 3D heatmaps

The basic idea of 3D heatmaps is to apply a surface construction on the 2D heatmaps captured. This is with the assumption that a distribution function is already applied, which is Gaussian in our case.

10.7.1 Optimization

The optimization is applied on the frequency maps before starting surface construction.

We apply hierarchical clustering on data that has similar or very close values, such that the actions of coordinates that are so close will be grouped and assigned an average value. For instance, in Figure 10-10, a number of mouse clicks and eye fixations were involved at very close coordinates when trying to rename a class. Hence, the final output looks as though if these actions all occurred at the same coordinate.

Similarly, emotions can be clustered based on the polarity being negative or positive.

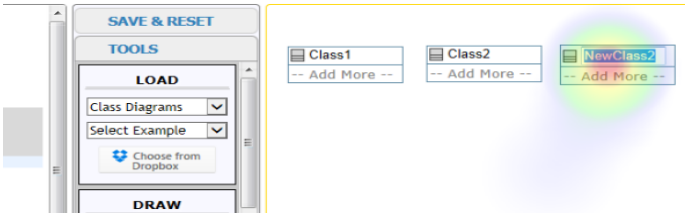


Figure 10-10. Close actions in a heatmap

10.7.2 Surface construction

Surface construction is a signed-distance function (Figure 10-11) defined by a continuous scalar function within a 3D space [245]. A construction function (f) represents *point clouds* matching a condition, in which numbers must be zero for each coordinate, x, y, and z.

$$f(v) = 0 \qquad v \in \{x, y, z\} \tag{10-10}$$

A construction function is positive when points are above the surface, and negative when points are under the surface. In such a case, a vector distance (d) is defined between the points and surface as below.

$$f(v) = d \qquad v \in \{x, y, z\} \tag{10-11}$$

In Figure 10-11, on-curve points are constraints of zero values, and the rest of the points are defined as vector normal constraints for a vector distance d.

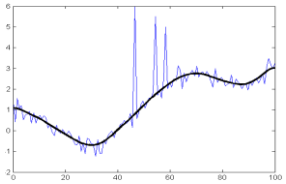


Figure 10-11. A signed-distance function example

Vector distances (d) for all positions in space can be obtained using vector interpolation and dimensional transformation.

We use a Self-Organizing Map (SOM) for the surface construction, since SOM is known for its high speed [246]. The construction process (Figure 10-12) can be summarized as below. The major steps are visualized in Figure 10-13.

- **Data normalization:** actions in the frequency maps must be normalized, such that an action will have coordinates (x, y, z), in which z represents the user emotion experience (E) occurred at the coordinates (x, y).
- **Network construction:** initialize a SOM network that follows a competitive learning approach, as explained below.
- **Error rate:** in competitive learning approach, a winning neuron is selected [246], error rate is calculated, and then correction is applied on the weight vector.
- **Termination:** the process above applies iteratively until the error rate is negligent. SOM will end up with the boundary of point cloud, which will be followed with Delaunay triangulation [247].
- **Smoothing:** apply a kernel method to smooth the surface by using 3D structure interpolation. This helps reduce noise.

- **Surface construction:** a constructed surface takes the features of the terrain surface such as the colour. Construction consists of the steps below
  - Construct face and normal vectors, and eliminate free forms.
  - Apply a distance-based function, such as Euclidian distance, to estimate the kernel. A distance-based function is used to calculate the difference between a fixed point in a direction, and another point.
  - Construct a distance-based kernel matrix based on the differences calculated from the step above.
  - Construct an iso surface using the distance-based matrix.

```
1   Input: heatmaps, frequencyMap;
2   prepareFrequencyMapFor3D(frequencyMap);
3   applyHierarchicalclustering(frequencyMap);
4   networkInstance= constructSOMNetwork(frequencyMap);
6   errorRate=∞;
7   weightVector= networkInstance. weightVector();
8   while(errorRate>threshold){
9       errorRate = networkInstance.findWiningNeuron();
10      weightVector= networkInstance.weightVector();
11  }
12  _3DFrequencyPoints = applySmooth(networkInstance);
13  vector= constructVectors(_3DFrequencyPoints);
14  kernelMatrix= constructKernelMatrix( _3DFrequencyPoints, vectors);
15  return createIsosurface(kernelMatrix);
```

Figure 10-12. Image 3D construction

A constructed surface will appear as a square or rectangular terrain that can be 3D rotated or transitioned, and has 3D camera options.

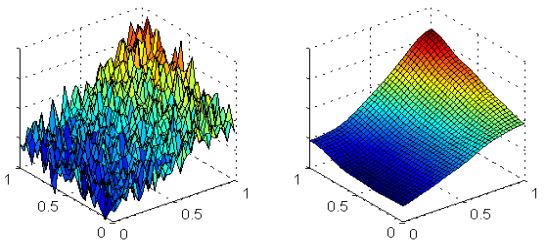


Figure 10-13. Before smoothing, after smoothing, after surface construction

10.8 Related work

The popularity of heatmaps to render interactions and/or fixations has been increasing recently. This can be observed by the introduction of much software and many websites offering heatmaps services. Examples include Inspectlet (Inspectlet.com), Mouseflow (Mouseflow.com), and Clicktale (Clicktale.com).

Studies related to detecting eye movements using techniques such as height maps, have been around for long time [248]. In 1992, Nodine et el introduced a way to trace eye positions as opposed to eye movements. This was the start

of the use of fixations in research, which Wooding found that it was still overlooked before the introduction of the fixation heatmaps [249].

Several usability studies rely on eye tracking to supplement tests [250], where several metrics are involved, to analyze usability issues, such as fixation, saccade, and gaze data.

The idea of 3D visualization was discussed in a few studies. The focus was more on constructing 3D models, rather than focusing on depicting user emotions as a part of the visualization techniques. Stellmach et al introduced 3D attentional maps to aggregate gaze data [251]. The study did not consider user interactions as a part of the 3D model, focusing only on the eye tracking data on 3D-rendered models as opposed to testing against applications or videos. The development of 3D models in virtual environment was challenging and still in early stages.

Maurus et al presented a 3D gaze visualization using real-time-generated heatmaps [252]. The study did not discuss whether techniques introduced will be used for detection of usability issues or at least, to be linked with user interactions. The focus was on showing the rendering performance of the heatmaps generated.

In terms of 3D implicit surface reconstruction, there are several techniques such RBF [253], Self Organized Map (SOM)[246], and Fuzzy C Means [254]. An algorithm such as RBF is highly effective, but it is criticised for its slow performance [246]. Other techniques such as SOM have less accuracy, but they are highly efficient.

10.9 Conclusion

We presented novel visualization techniques, 3D heatmaps and widget heatmaps. 3D heatmaps extend traditional heatmaps to include user emotional experience in the third dimension. A widget heatmap shows the user interactions on certain UI elements rather than certain areas.

These novel visualization techniques aim to ease the usability evaluation by linking users' emotion experience with their attention focus on UI elements.

Chapter 11 Conclusions and Future Work

In this thesis, we presented a tool, and its various underlying technologies, that eases the process of data collection, analysis, and linking from multiple sources. We demonstrated a multimodal signal fusion model that fuses data and extracts meaningful information to be used in domains such as HCI. Fusion is mainly used to improve emotion recognition accuracy – affective and mental states. We explored and provided real-time recognition of affective and primary mental states. The six primary mental states we considered are agreeing, concentrating, disagreeing, interested, thinking, and unsure.

The multimodal fusion engine can use the fused data to build classifiers and multioutput regressors, which can be used for other subtasks or classification.

We provided novel methods for visualizing, analyzing, and interpreting data as a way to improve User Emotional Experience (UEX).

11.1 Answers to research questions

In this section, we provide answers to the research questions listed in Chapter 1 based on the discussions in this thesis.

RQ1: How can a tool be designed to collect and analyze observational data, and meet our goals?

We addressed this in Chapters 3, 4, 6, and 10. In particular, Chapter 3 discussed the tool architecture that we implemented to collect and fuse data, while in Chapter 4,we explained the pipeline-based approach we used to systemize communication among different components, as well as data analysis and annotation. Chapter 6 presented the implementation details of multimodal signal fusion models, which are used for interpretation. Chapter 10 addressed the visualization methods we implemented to depict affective and mental states, and how they can be used to understand user experience.

RQ2: How can we build a reliable multimodal signal fusion model that provides real time affective and mental state recognition, in a way that can handle the uncertainty of sensor configuration?

We addressed this question in Chapters 5, 7, 8, 9, and 10. We implemented two novel fusion techniques, Temporal Feature Fusion (TFF), and Temporal Output Fusion (TOF) (Chapter 6).

We also demonstrated the performance of data fusion, which improves the accuracy of affective and mental state recognition and prediction in a real-time manner — User Emotional Experience (UEX).

TFF is applied at early stages, and used by all of our implemented modalities (Chapters 6, 7, 8, and 9), since we needed to fuse each of these with other modalities. TSF is applied at the extracted joint representations. On the other

hand, TOF, which applies at late stages, was only used with the emotion modality (Chapter 9). This was a part of the in-wild recognition that required using outputs of other modalities in order to enhance results.

We showed in Chapter 2 the different emotion representations as well as the sources commonly used to capture affective and mental states in a real-time manner. The main goal of having real time recognition is to capture primary mental states.

We used our tool (Chapter 4) to test and fuse data from different sensors, each of which can be used alone as a source for emotion recognition or can be fused with other sensors — gaze estimation (Chapter 7), and facial action units (Chapter 8).

RQ3: How can we link users' affective and mental state to UI elements of a user interface?

In Chapter 5, we showed how head gesture and gaze estimation can be used to track eye gaze and screen coordinates. We demonstrated how both inputs can be used as a means to collect UI elements, at which a user looks, such that these elements can be synchronized with other predicted data such as the current emotional state of that user. Such data can be used with other visualization techniques to give additional insights.

Towards that, in Chapter 10 we introduced three novel visualization methods, 3D heatmaps, 3D scanpaths, and widget heatmaps, which are all used concurrently to link user emotional states with a user interface; i.e. they can help an evaluator to have better assessment while evaluating a user interface.

11.2 Contributions

We listed the related contributions at the beginning of each chapter. In this section, we summarize the top-level contributions of our research.

A psychophysiological observing and analysis tool, which follows a client/server paradigm, in which session recording takes a place at the client side, and the storing, processing, and analysis of the observed data is done on the server (Chapter 3). We applied a pipeline-based approach to manage and chain components, as well as to analyse, annotate, and visualize data (Chapter 4).

Real-time facial feature detectors that are competitive with those in the published literature**.** We implemented several detectors (Chapter 5 and Chapter 8).

Calibration-free real-time head and gaze estimation, also better than that in the published literature according to certain metrics. These are used with eye tracking (Chapter 7).

Two novel fusion techniques, Temporal Feature Fusion (TFF), and Temporal Output Fusion (TOF).(Section 6.5)

A novel Deep Temporal Credence Network (DTCN). (Chapter 6)

Multimodal signal fusion mode using our DTCN. This provides fast queuing, interpretation, and comprehension of the large amount of data received from different sensors, and produces meaningful fused data.

Real-time affective and primary mental state recognition. Our approach recognizes the basic six emotions of Ekman, as well as six primary mental states, agreeing, concentrating, disagreeing, interested, thinking, and unsure (Chapters 2 and 11). We were able to test the basic emotion recognition against in-wild datasets.

Novel visualization techniques: We produce 3D heatmaps, 3D scanpaths, and widget heatmaps (Chapter 10), which give quick hints about data associated with the affective states of users.

11.3 Future work

The work and results presented in this thesis could be further developed in the following future directions:

Empirical Evaluations: Apply empirical evaluation methods to test the usability of the various aspects our tool.

Extensibility: Incorporate different extensions and tools into our pipeline framework, such that an evaluator can develop sophisticated analysis and visualization modules.

Semi-automated UEX metrics: Investigate the effectiveness of emotion-based indicators for outlining usability issues. For instance, the users' interactions with a website can be assessed with less supervision by the evaluator. Examples of indicators might include effort, emotion response, and cognitive effort.

Spontaneous and micro-expression dataset: Build a multimodal dataset that includes spontaneous recording of micro facial expressions, comprises different sensors, and considers the primary six mental states as well as basic emotions.

Intrusive and nonintrusive sensors: To date, nonintrusive sensors have theoretically demonstrated many benefits and show competitive performance. But there are few studies that empirically test their efficiency and effectiveness related to HCI. The main reasons for not doing this yet are the lack of comprehensive dataset,

EEG evaluation: At the beginning of this research, we planned to validate the emotion recognition of our inexpensive and nonintrusive capturing against intrusively captured brainwave signals. However, we have not found any datasets available that sufficiently test the basic and complex emotions. Hence, we decided to leave it out. In particular, the emotion recognition results, which we obtained from our nonintrusive capturing met our expectations. These are intended to be used as insights for further semi-automated usability evaluation.

References

[1] C. J. T. T. F. Cootes, G. J. Edwards, "Active appearance models," *Springer Berlin Heidelb.*, pp. 484–498, Dec. 1998.

[2] I. Matthews, I. Matthews, S. Baker, and S. Baker, "Active appearance models revisited," *Int. J. Comput. Vis.*, vol. 60, pp. 135–164, 2004.

[3] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, pp. 321–331, 1988.

[4] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active Shape Models-Their Training and Application," *Comput. Vis. Image Underst.*, vol. 61, pp. 38–59, 1995.

[5] R. W. Picard, "Affective Computing," in *M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 321*, 1995.

[6] E. Harmon-Jones, P. A. Gable, and T. F. Price, "Does Negative Affect Always Narrow and Positive Affect Always Broaden the Mind? Considering the Influence of Motivational Intensity on Cognitive Scope," *Curr. Dir. Psychol. Sci.*, vol. 22, no. 4, pp. 301–307, Aug. 2013.

[7] E. Harmon-Jones, C. Harmon-Jones, D. M. Amodio, and P. A. Gable, "Attitudes toward emotions.," *J. Pers. Soc. Psychol.*, vol. 101, no. 6, pp. 1332–50, Dec. 2011.

[8] P. A. Gable and E. Harmon-Jones, "Does arousal per se account for the influence of appetitive stimuli on attentional scope and the late positive potential?," *Psychophysiology*, vol. 50, no. 4, pp. 344–50, Apr. 2013.

[9] T. McInerney and D. Terzopoulos, "Deformable models in medical image analysis: a survey," *Med. Image Anal.*, vol. 1, no. 2, pp. 91–108, Jun. 1996.

[10] J.-F. Dufourd and Y. Bertot, "Formal study of plane Delaunay triangulation," *Interact. Theorem Proving*, vol. 6172, pp. 211–226, 2010.

[11] D. D. Salvucci and J. H. Goldberg, "Identifying fixations and saccades in eye-tracking protocols," *Proc. Symp. Eye Track. Res. Appl. ETRA 00*, vol. 469, no. 1, pp. 71–78, 2000.

[12] C. H. Morimoto and M. R. M. Mimica, "Eye gaze tracking techniques for interactive applications," *Comput. Vis. Image Underst.*, vol. 98, no. 1, pp. 4–24, Apr. 2005.

[13] D. W. Hansen and Qiang Ji, "In the Eye of the Beholder: A Survey of Models for Eyes and Gaze," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 478–500, Mar. 2010.

[14] P. Ekman, *Handbook of Cognition and Emotion*. Chichester, UK: John Wiley & Sons, Ltd, 1999.

[15] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon, "Collecting Large, Richly Annotated Facial-Expression

Databases from Movies," *IEEE Multimed.*, vol. 19, no. 3, pp. 34–41, 2012.

[16] A. Dhall, R. Goecke, J. Joshi, and T. Gedeon, "The Fourth Emotion Recognition in the Wild Challenge 2015: Baseline, Data and Protocols," in *18th ACM International Conference on Multimodal Interaction, Tokyo, Japan*, 2016.

[17] Y. Jia, J. Wang, G. Zeng, H. Zha, and X.-S. Hua, "Optimizing kd-trees for scalable visual descriptor indexing," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3392–3399.

[18] Xiangxin Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2879–2886.

[19] J. Nielsen, *Usability engineering*. Morgan Kaufmann; 1 edition, 1993.

[20] A. Holzinger, "Usability engineering methods for software developers," *Commun. ACM*, vol. 48, no. 1, pp. 71–74, Jan. 2005.

[21] A. S. Alain Abran, Adel Khelifi, Witold Suryn, "Usability Meanings and Interpretations in ISO Standards," *Softw. Qual. Control*, vol. 11, no. 4, pp. 325–338, 2003.

[22] G. D. Fiora T. W. Au, Simon Baker, Ian Warren, "Automated usability testing framework," in *Proceeding AUIC '08 Proceedings of the ninth conference on Australasian user interface*, 2008, pp. 55–64.

[23] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI '90*, 1990, pp. 249–256.

[24] J. Nielsen, "Ten Usability Heuristics," *useitcom*, 2005. [Online]. Available: http://www.useit.com/papers/heuristic/heuristic_list.html.

[25] J. Nielsen, "The usability engineering life cycle," *Computer (Long. Beach. Calif).*, vol. 25, no. 3, pp. 12–22, Mar. 1992.

[26] L. S. Frederick G. Freeman, Peter J. Mikulka, Mark W. Scerbo, "An evaluation of an adaptive automation system using a cognitive vigilance task," *Biol. Psychol.*, vol. 67, no. 3, pp. 283–297, 2004.

[27] W. Boucsein, "The Use of Psychophysiology for Evaluating Stress-Strain Processes in Human-Computer Interaction," in *Engineering Psychophysiology: Issues and Applications*, 2000, pp. 289–309.

[28] A. Poole and L. J. Ball, "Eye Tracking in Human-Computer Interaction and Usability Research : Current Status and Future Prospects," *Psychology*, vol. 10, pp. 211–219, 2005.

[29] D. Noton and L. Stark, "Scanpaths in saccadic eye movements while viewing and recognizing patterns.," *Vision Res.*, vol. 11, no. 9, pp. 929–42, Sep. 1971.

[30] J. Nielsen and Kara Pernice, *Eyetracking Web Usability*. New Riders, 2009.

[31] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance

in Face Verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[32] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[33] X. Tian, L. Yang, Y. Lu, Q. Tian, and D. Tao, "Image Search Reranking With Hierarchical Topic Awareness," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2177–2189, Oct. 2015.

[34] H. Li, Y. Wei, L. Li, and C. L. P. Chen, "Hierarchical Feature Extraction With Local Neural Response for Image Recognition.," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 412–24, Apr. 2013.

[35] Y. Bengio, "Learning Deep Architectures for AI," *Found. Trends® Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[36] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[37] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised Deep Feature Extraction for Remote Sensing Image Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1349–1362, Mar. 2016.

[38] A. Coates, H. Lee, and A. Y. Ng, "An Analysis of Single-Layer Networks in Unsupervised Feature Learning," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 215– 223.

[39] A. Liapis, N. Karousos, C. Katsanos, and M. Xenos, "Evaluating User's Emotional Experience in HCI: The PhysiOBS Approach," in *16th International Conference, HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings, Part II*, 2014, pp. 758–767.

[40] M. S. Bartlett, G. C. Littlewort, M. G. Frank, and K. Lee, "Automatic Decoding of Facial Movements Reveals Deceptive Pain Expressions," *Curr. Biol.*, vol. 24, no. 7, pp. 738–743, Mar. 2014.

[41] D. McDuff, R. el Kaliouby, T. Senechal, M. Amr, J. F. Cohn, and R. Picard, "Affectiva-MIT Facial Expression Dataset (AM-FED): Naturalistic and Spontaneous Facial Expressions Collected 'In-the-Wild,'" in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 881–888.

[42] P. H. Zimmerman, J. E. Bolhuis, A. Willemsen, E. S. Meyer, and L. P. J. J. Noldus, "The Observer XT: A tool for the integration and synchronization of multimodal signals," *Behav. Res. Methods*, vol. 41, no. 3, pp. 731–735, Aug. 2009.

[43] N. Bosch *et al.*, "Automatic Detection of Learning-Centered Affective States in the Wild," in *Proceedings of the 20th International Conference on Intelligent User Interfaces - IUI '15*, 2015, pp. 379–388.

[44] P. Ekman and H. Oster, "Facial Expressions of Emotion," *Annu. Rev. Psychol.*, vol. 30, no. 1, pp. 527–554, Jan. 1979.

[45] S. Baron-Cohen, *Mind Reading: The Interactive Guide to Emotions, Version 1.3*. Jessica Kingsley Pub; Cdr

edition, 2004.

[46] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, 2005, vol. I, pp. 886–893.

[47] E. Antonakos, J. Alabort-i-medina, G. Tzimiropoulos, and S. Zafeiriou, “HOG active appearance models,” in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 224–228.

[48] R. J. Davidson, “Affective neuroscience and psychophysiology: toward a synthesis.,” *Psychophysiology*, vol. 40, no. 5, pp. 655–65, Sep. 2003.

[49] P. A. Nogueira, R. Rodrigues, and E. Oliveira, “Real-Time Psychophysiological Emotional State Estimation in Digital Gameplay Scenarios,” in *Communications in Computer and Information Science*, 2013, vol. 383 CCIS, no. PART 1, pp. 243–252.

[50] J. Kim and E. André, “Emotion recognition based on physiological changes in music listening.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 12, pp. 2067–83, Dec. 2008.

[51] R. W. Picard, E. Vyzas, and J. Healey, “Toward machine emotional intelligence: analysis of affective physiological state,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 10, pp. 1175–1191, 2001.

[52] K. H. Kim, S. W. Bang, and S. R. Kim, “Emotion recognition system using short-term monitoring of physiological signals,” *Med. Biol. Eng. Comput.*, vol. 42, no. 3, pp. 419–427, May 2004.

[53] P. Shaver, J. Schwartz, D. Kirson, and C. O’Connor, “Emotion knowledge: further exploration of a prototype approach.,” *J. Pers. Soc. Psychol.*, vol. 52, no. 6, pp. 1061–86, Jun. 1987.

[54] J. A. Russell, “A circumplex model of affect.,” *J. Pers. Soc. Psychol.*, vol. 39, no. 6, pp. 1161–1178, 1980.

[55] D. M. W. Daniel L. Schacter, Daniel T. Gilbert, *Psychology*. Worth Publishers; Second Edition edition, 2010.

[56] A. Damasio, *Descartes’ Error: Emotion, Reason, and the Human Brain*. Penguin Books, 1994.

[57] P. Ekman, “Facial expression and emotion.,” *Am. Psychol.*, vol. 48, no. 4, pp. 384–392, 1993.

[58] S. Brave and Clifford Nass, “Emotion in human-computer interaction,” in *The human-computer interaction handbook*, 2002, pp. 81–96.

[59] S. Du, Y. Tao, and A. M. Martinez, “Compound facial expressions of emotion.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 111, no. 15, pp. E1454-62, Apr. 2014.

[60] R. Plutchik, “Circumplex models of personality and emotions.,” in *Theory of Emotion*, R. Plutchik and H. R. Conte, Eds. Washington, DC, US: American Psychological Association, 1980.

[61] C. L. Lisetti and F. Nasoz, “Using Noninvasive Wearable Computers to Recognize Human Emotions from Physiological Signals,” *EURASIP J. Adv. Signal Process.*, vol. 2004, no. 11, pp. 1672–1687, 2004.

[62] M. Duvinage, T. Castermans, M. Petieau, T. Hoellinger, G. Cheron, and T. Dutoit, “Performance of the Emotiv Epoc headset for P300-based applications.,” *Biomed. Eng. Online*, vol. 12, p. 56, 2013.

[63] J. Katona, I. Farkas, T. Ujbanyi, P. Dukan, and A. Kovari, “Evaluation of the NeuroSky MindFlex EEG headset brain waves data,” in *SAMI 2014 - IEEE 12th International Symposium on Applied Machine Intelligence and Informatics, Proceedings*, 2014, pp. 91–94.

[64] T. Lin, M. Omata, W. Hu, and A. Imamiya, “Do physiological data relate to traditional usability indexes?,” in *Proceeding OZCHI ’05 Proceedings of the 17th Australia conference on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future*, 2005, pp. 1–10.

[65] A. Duchowski, “Eye Tracking Techniques,” in *Eye Tracking Methodology*, London: Springer London, 2007, pp. 51–59.

[66] J. Lazar, A. Jones, and B. Shneiderman, “Workplace user frustration with computers: an exploratory investigation of the causes and severity,” *Behaviour & Information Technology*, vol. 25, no. 3. pp. 239–251, 2006.

[67] J. J. A. Denissen, L. Butalid, L. Penke, and M. A. G. van Aken, “The effects of weather on daily mood: a multilevel approach.,” *Emotion*, vol. 8, no. 5, pp. 662–7, Oct. 2008.

[68] B. J. Maron and A. Pelliccia, “The heart of trained athletes: cardiac remodeling and the risks of sports, including sudden death.,” *Circulation*, vol. 114, no. 15, pp. 1633–44, Oct. 2006.

[69] P. Omvik, “How smoking affects blood pressure.,” *Blood Press.*, vol. 5, no. 2, pp. 71–7, Mar. 1996.

[70] A. Haag, S. Goronzy, P. Schaich, and J. Williams, “Emotion Recognition Using Bio-sensors: First Steps towards an Automatic System,” in *Affective Dialogue Systems*, vol. 3068, E. André, L. Dybkjær, W. Minker, and P. Heisterkamp, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 36–48.

[71] G. Stemmler, M. Heldmann, C. A. Pauls, and T. Scherer, “Constraints for emotion specificity in fear and anger: the context counts.,” *Psychophysiology*, vol. 38, no. 2, pp. 275–91, Mar. 2001.

[72] R. R. Cornelius, “Theoretical approaches to emotion,” in *International Speech Communication Association (ISCA) Workshop on Speech and Emotion, Belfast, Ireland*, 2000.

[73] J. Anttonen and V. Surakka, “Emotions and heart rate while sitting on a chair,” in *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI ’05*, 2005, p. 491.

[74] P. Ekman, R. W. Levenson, and W. V Friesen, “Autonomic nervous system activity distinguishes among emotions.,” *Science*, vol. 221, no. 4616, pp. 1208–10, Sep. 1983.

[75] B. N. & C. P J Lang, M M Bradley, “International affective picture system (IAPS): Affective ratings of pictures and instruction manual.” University of Florida, Gainesville, FL, 2008.

[76] S. Jerritta, M. Murugappan, K. Wan, and S. Yaacob, “Emotion recognition from facial EMG signals using higher order statistics and principal component analysis,” *J. Chinese Inst. Eng.*, vol. 37, no. 3, pp. 385–394,

Apr. 2014.

[77] R. Cowie *et al.*, “Emotion recognition in human-computer interaction,” *IEEE Signal Process. Mag.*, vol. 18, no. 1, pp. 32–80, 2001.

[78] S. A. Hosseini, “Classification of Brain Activity in Emotional States Using HOS Analysis,” *International Journal of Image, Graphics and Signal Processing*, vol. 4, no. 1. pp. 21–27, 2012.

[79] K. S. Rahnuma, A. Wahab, H. A. Majid, and B. Crüts, “Analyzing brain activity in understanding cultural and language interaction for depression and anxiety,” in *Procedia - Social and Behavioral Sciences*, 2011, vol. 27, pp. 299–305.

[80] D. Giakoumis, D. Tzovaras, K. Moustakas, and G. Hassapis, “Automatic recognition of boredom in video games using novel biosignal moment-based features,” *IEEE Trans. Affect. Comput.*, vol. 2, no. 3, pp. 119–133, 2011.

[81] Y. Liu, O. Sourina, and M. K. Nguyen, “Real-time EEG-based emotion recognition and its applications,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6670 LNCS, pp. 256–277.

[82] R. El Kaliouby, “Mind-Reading Machines: Automated Inference of Complex Mental States (PhD Thesis),” University of Cambridge, 2005.

[83] S. Pourzare, O. Aydemir, and T. Kayikcioglu, “Classification of various facial movement artifacts in EEG signals,” in *2012 35th International Conference on Telecommunications and Signal Processing, TSP 2012 - Proceedings*, 2012, pp. 529–533.

[84] J. Wagner and E. Andre, “From Physiological Signals to Emotions: Implementing and Comparing Selected Methods for Feature Extraction and Classification,” in *2005 IEEE International Conference on Multimedia and Expo*, 2005, pp. 940–943.

[85] R. Sinha, “Multivariate Response Patterning of Fear and Anger,” *Cogn. Emot.*, vol. 10, no. 2, pp. 173–198, Mar. 1996.

[86] E. Leon, G. Clarke, V. Callaghan, and F. Sepulveda, “A user-independent real-time emotion recognition system for software agents in domestic environments,” *Eng. Appl. Artif. Intell.*, vol. 20, no. 3, pp. 337–345, Apr. 2007.

[87] P. Rainville, A. Bechara, N. Naqvi, and A. R. Damasio, “Basic emotions are associated with distinct patterns of cardiorespiratory activity.,” *Int. J. Psychophysiol.*, vol. 61, no. 1, pp. 5–18, Jul. 2006.

[88] C. D. Katsis, N. Katertsidis, G. Ganiatsas, and D. I. Fotiadis, “Toward Emotion Recognition in Car-Racing Drivers: A Biosignal Processing Approach,” *IEEE Trans. Syst. Man, Cybern. - Part A Syst. Humans*, vol. 38, no. 3, pp. 502–512, May 2008.

[89] P. Rani, C. Liu, N. Sarkar, and E. Vanman, “An empirical study of machine learning techniques for affect

recognition in human-robot interaction," *Pattern Anal. Appl.*, vol. 9, no. 1, pp. 58–69, 2006.

[90] M. Li and B.-L. Lu, "Emotion classification based on gamma-band EEG.," *Conf. Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Conf.*, vol. 2009, pp. 1323–6, Jan. 2009.

[91] K. Takahashi, "Remarks on emotion recognition from multi-modal bio-potential signals," in *IEEE International Conference on Industrial Technology, 2004. IEEE ICIT '04.*, 2004, vol. 3, pp. 1138–1143.

[92] D. Tamir, O. V. Komogortsev, and C. J. Mueller, "An effort and time based measure of usability," in *Proceedings of the 6th international workshop on Systems development in SOA environments - WoSQ '08*, 2008, p. 47.

[93] A. C. Oei and M. D. Patterson, "Enhancing Cognition with Video Games: A Multiple Game Training Study," *PLoS One*, vol. 8, no. 3, 2013.

[94] M. Z. Poh, N. C. Swenson, and R. W. Picard, "A wearable sensor for unobtrusive, long-term assessment of electrodermal activity," *IEEE Trans. Biomed. Eng.*, vol. 57, no. 5, pp. 1243–1252, 2010.

[95] K. L. Schmidt and J. F. Cohn, "Human facial expressions as adaptations: Evolutionary questions in facial expression research.," *Am. J. Phys. Anthropol.*, vol. Suppl 33, pp. 3–24, 2001.

[96] R. Oostenveld and P. Praamstra, "The five percent electrode system for high-resolution EEG and ERP measurements.," *Clin. Neurophysiol.*, vol. 112, no. 4, pp. 713–9, Apr. 2001.

[97] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Comput. Vis. Image Underst.*, vol. 110, pp. 346–359, 2008.

[98] A. Javed, "Complex Flows: Node-RED," in *Building Arduino Projects for the Internet of Things*, Berkeley, CA: Apress, 2016, pp. 51–73.

[99] W. Tłaczała, "Modeling with LabVIEW™," in *Handbook of Measuring System Design*, Chichester, UK: John Wiley & Sons, Ltd, 2005.

[100] M. Husseini Orabi, A. Husseini Orabi, and T. Lethbridge, "Umple as a component-based language for the development of real-time and embedded applications," in *IEEE International Conference on Model-Driven Engineering and Software Development*, 2016.

[101] N. Kitcharoen, S. Kamolsantisuk, R. Angsomboon, and T. Achalakul, "RapidMiner framework for manufacturing data analysis on the cloud," in *The 2013 10th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2013, pp. 149–154.

[102] S. Beisken, T. Meinl, B. Wiswedel, L. F. de Figueiredo, M. Berthold, and C. Steinbeck, "KNIME-CDK: Workflow-driven cheminformatics," *BMC Bioinformatics*, vol. 14, no. 1, p. 257, 2013.

[103] A. Bolt, M. de Leoni, and W. M. P. van der Aalst, "Scientific workflows for process mining: building blocks, scenarios, and implementation," *Int. J. Softw. Tools Technol. Transf.*, Sep. 2015.

[104] U. B. Baizyldayeva, R. K. Uskenbayeva, and S. T. Amanzholova, "Decision Making Procedure: Applications of IBM SPSS Cluster Analysis and Decision Tree," *World Appl. Sci. J.*, vol. 21, no. 8, pp. 1207–1212, 2013.

[105] R. Ihaka and R. Gentleman, "R: A Language for Data Analysis and Graphics," *J. Comput. Graph. Stat.*, vol. 5, no. 3, p. 299, Sep. 1996.

[106] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explor. Newsl.*, vol. 11, no. 1, p. 10, Nov. 2009.

[107] A. Sarica, G. Di Fatta, and M. Cannataro, "K-Surfer: A KNIME Extension for the Management and Analysis of Human Brain MRI FreeSurfer/FSL Data," in *International Conference, BIH 2014, Warsaw, Poland, August 11-14, 2014, Proceedings*, 2014, pp. 481–492.

[108] OMG, "UML 2.4.1," 2011. [Online]. Available: http://www.omg.org/spec/UML/2.4.1/. [Accessed: 01-Jan-2015].

[109] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *Int. J. Comput. Vis.*, vol. 56, pp. 221–255, 2004.

[110] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 681–685, Jun. 2001.

[111] G. Tzimiropoulos and P. Maja, "Gauss-Newton Deformable Part Models for Face Alignment In-the-Wild," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference*, 2014.

[112] H. Wu, X. Liu, and G. Doretto, "Face alignment via boosted ranking model," in *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.

[113] X. L. X. Liu, "Generic Face Alignment using Boosted Appearance Model," *2007 IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007.

[114] E. Antonakos, J. Alabort-i-Medina, G. Tzimiropoulos, and S. P. Zafeiriou, "Feature-based Lucas-Kanade and active appearance models.," *IEEE Trans. Image Process.*, vol. 24, no. 9, pp. 2617–32, 2015.

[115] J. Alabort-i-medina and Stefanos Zafeiriou, "Bayesian Active Appearance Models," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014, pp. 1–7.

[116] G. Papandreou and P. Maragos, "Adaptive and constrained algorithms for inverse compositional active appearance model fitting," in *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.

[117] D. Cristinacce and T. Cootes, "Feature Detection and Tracking with Constrained Local Models," in *17^{th} British Machine Vision Conference, Edinburgh, UK*, 2006, pp. 929–938.

[118] C. Liu, J. Yuen, and A. Torralba, "SIFT flow: Dense correspondence across scenes and its applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 978–994, 2011.

[119] A. J. Bell and T. J. Sejnowski, “The ‘independent components’ of natural scenes are edge filters,” *Vision Res.*, vol. 37, no. 23, pp. 3327–3338, Dec. 1997.

[120] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *J. R. Stat. Soc. Ser. B (Statistical Methodol.*, vol. 61, no. 3, pp. 611–622, 1999.

[121] Jongwoo Lim and Ming-Hsuan Yang, “A Direct Method for Modeling Non-Rigid Motion with Thin Plate Spline,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, pp. 1196–1202.

[122] S. Belongie, J. Malik, and J. Puzicha, “Shape matching and object recognition using shape contexts,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.

[123] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, “Interactive Facial Feature Localization,” in *ECCV’12 Proceedings of the 12th European conference on Computer Vision - Volume Part III*, 2012, pp. 679–692.

[124] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, “Localizing parts of faces using a consensus of exemplars,” in *CVPR 2011*, 2011, pp. 545–552.

[125] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, “300 Faces in-the-Wild Challenge: The First Facial Landmark Localization Challenge,” in *2013 IEEE International Conference on Computer Vision Workshops*, 2013, pp. 397–403.

[126] D. E. King, “Max-Margin Object Detection,” in *arXiv:1502.00046*, 2015.

[127] D. Lundqvist, A. Flykt, and A. Öhman, *The Karolinska Directed Emotional Faces - KDEF, CD ROM from Department of Clinical Neuroscience, Psychology section, Karolinska Institutet*. 1998.

[128] P. Dollar, R. Appel, S. Belongie, and P. Perona, “Fast Feature Pyramids for Object Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1532–1545, Aug. 2014.

[129] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–45, Sep. 2010.

[130] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. Lecun, “Pedestrian Detection with Unsupervised Multi-stage Feature Learning,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3626–3633.

[131] K. Fukushima, “Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.,” *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, 1980.

[132] Y. LeCun, “Deep Learning Tutorial,” in *ICML, Atlanta, 2013, Center for Data Science & Courant Institute, NYU*, 2013.

[133] P.-A. M. Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion,” *J.*

Mach. Learn. Res., vol. 11, no. 3/1/2010, pp. 3371–3408, 2010.

[134] J. Xu *et al.*, "Stacked Sparse Autoencoder (SSAE) for Nuclei Detection on Breast Cancer Histopathology Images.," *IEEE Trans. Med. Imaging*, vol. 35, no. 1, pp. 119–30, Jan. 2015.

[135] D. P. Kingma and W. Max, "Auto-Encoding Variational Bayes," in *The International Conference on Learning Representations (ICLR), Banff*, 2014.

[136] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," in *21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I*, 2011, pp. 52–59.

[137] M. Norouzi, M. Ranjbar, and G. Mori, "Stacks of convolutional Restricted Boltzmann Machines for shift-invariant feature learning," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2735–2742.

[138] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, 2009, pp. 1–8.

[139] G. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[140] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized Denoising Auto-Encoders as Generative Models," May 2013.

[141] P. Vincent, "A Connection Between Score Matching and Denoising Autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, Jul. 2011.

[142] R. Baillargeon, "Infants' reasoning about hidden objects: evidence for event-general and event-specific expectations," *Dev. Sci.*, vol. 7, no. 4, pp. 391–414, Sep. 2004.

[143] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[144] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised Learning of Video Representations using LSTMs," Feb. 2015.

[145] J. Ngiam, A. Khosla, M. Kim, J. Nam, Lee, Honglak, and A. Y. Ng, "Multimodal Deep Learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 689--696.

[146] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with Deep Boltzmann Machines," in *In Neural Information Processing Systems (NIPS)*, 2012.

[147] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[148] J. Huang and B. Kingsbury, "Audio-visual deep learning for noise robust speech recognition," in *2013 IEEE*

International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 7596–7599.

[149] D. Rao, A. Bender, S. B. Williams, and O. Pizarro, "Multimodal information-theoretic measures for autonomous exploration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4230–4237.

[150] T. Joachims, "Transductive inference for text classification using support vector machines," *Int. Conf. Mach. Learn.*, vol. 99, pp. 200–209, 1999.

[151] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–7, Jul. 2006.

[152] Deeplearning4j, "Deep Autoencoders," 2016. [Online]. Available: https://deeplearning4j.org/deepautoencoder. [Accessed: 01-Jan-2017].

[153] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, and C. Suen, "UFLDL Tutorial," 2016. [Online]. Available: http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial. [Accessed: 01-Dec-2016].

[154] A. Makhzani and B. Frey, "k-Sparse Autoencoders," Dec. 2013.

[155] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008, pp. 1096–1103.

[156] F. Chen, Y. Wu, G. Zhao, J. Zhang, M. Zhu, and J. Bai, "Contractive De-noising Auto-Encoder," in *Intelligent Computing Theory*, 2014, pp. 776–781.

[157] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contracting autoencoders: Explicit invariance during feature extraction," in *the Twenty-eight International Conference on Machine Learning (ICML)*, 2011.

[158] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *JMLR Workshop and Conference Proceedings*, 2014.

[159] P. M. Williams, "Matrix logarithm parametrizations for neural network covariance models.," *Neural Netw.*, vol. 12, no. 2, pp. 299–308, Mar. 1999.

[160] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[161] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," Jul. 2012.

[162] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012.

[163] M. D. Zeiler and R. Fergus, "Stochastic Pooling for Regularization of Deep Convolutional Neural

Networks," Jan. 2013.

[164] A. Y. Ng, "Feature selection, L 1 vs. L 2 regularization, and rotational invariance," in *Twenty-first international conference on Machine learning - ICML '04*, 2004, p. 78.

[165] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *Adv. Neural Inf. Process. Syst.*, vol. 27, no. NIPS 2014, Sep. 2014.

[166] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," in *Neural and Evolutionary Computing*, 2014.

[167] A. Graves, S. Fernández, and J. Schmidhuber, "Multi-Dimensional Recurrent Neural Networks," in *ICANN'07 Proceedings of the 17th international conference on Artificial neural networks*, 2007, pp. 549–558.

[168] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[169] A. Tjandra, S. Sakti, S. Nakamura, and M. Adriani, "Stochastic Gradient Variational Bayes for deep learning-based ASR," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 175–180.

[170] D. Scherer, A. Müller, and S. Behnke, "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition," in *Artificial Neural Networks – ICANN 2010*, 2010, pp. 92–101.

[171] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," in *NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, Canada*, 2014, pp. 3581–3589.

[172] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-Supervised Learning with Deep Generative Models," Jun. 2014.

[173] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, "Auxiliary Deep Generative Models," Feb. 2016.

[174] K. Sohn, H. Lee, and X. Yan, "Learning Structured Output Representation using Deep Conditional Generative Models," in *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.

[175] M. Suzuki, K. Nakayama, and Y. Matsuo, "Joint Multimodal Learning with Deep Generative Models," Nov. 2016.

[176] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, 2009, pp. 1–8.

[177] A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox, "Learning to Generate Chairs, Tables and Cars with Convolutional Networks," Nov. 2014.

[178] C. Rosenberg, M. Hebert, and H. Schneiderman, “Semi-Supervised Self-Training of Object Detection Models,” in *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION’05) - Volume 1*, 2005, pp. 29–36.

[179] J. Read, B. Pfahringer, G. Holmes, and E. Frank, “Classifier chains for multi-label classification,” *Mach. Learn.*, vol. 85, no. 3, pp. 333–359, Dec. 2011.

[180] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, “A survey on multi-output regression,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 5, no. 5, pp. 216–233, Sep. 2015.

[181] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, and I. Vlahavas, “Multi-target regression via input space expansion: treating targets as inputs,” Feb. 2016.

[182] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang, “Exploiting Feature and Class Relationships in Video Categorization with Regularized Deep Neural Networks,” Feb. 2015.

[183] E. Park, X. Han, T. L. Berg, and A. C. Berg, “Combining multiple sources of knowledge in deep CNNs for action recognition,” in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016, pp. 1–8.

[184] I. Goodfellow, Y. Bengio, and A. Courvile, *Deep Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, 2016.

[185] Y. N. Dauphin, H. de Vries, and Y. Bengio, “Equilibrated adaptive learning rates for non-convex optimization,” Feb. 2015.

[186] Li Deng, “The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web],” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[187] N. Pitelis, C. Russell, and L. Agapito, “Semi-supervised Learning Using an Unsupervised Atlas,” in *European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II*, 2014, pp. 565–580.

[188] T. Miyato, S. Maeda, M. Koyama, K. Nakae, and S. Ishii, “Distributional Smoothing with Virtual Adversarial Training,” Jul. 2015.

[189] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, “Semi-Supervised Learning with Ladder Networks,” in *NIPS’15 Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015, pp. 3546–3554.

[190] T. Schneider, B. Schauerte, and R. Stiefelhagen, “Manifold Alignment for Person Independent Appearance-Based Gaze Estimation,” in *2014 22nd International Conference on Pattern Recognition*, 2014, pp. 1167–1172.

[191] C. D. McMurrough, V. Metsis, J. Rich, and F. Makedon, “An eye tracking dataset for point of gaze detection,” in *Proceedings of the Symposium on Eye Tracking Research and Applications - ETRA ’12*, 2012,

p. 305.

[192] U. Weidenbacher, G. Layher, P.-M. Strauss, and H. Neumann, “A comprehensive head pose and gaze database,” in *3rd IET International Conference on Intelligent Environments (IE 07)*, 2007, vol. 2007, pp. 455–458.

[193] B. A. Smith, Q. Yin, S. K. Feiner, and S. K. Nayar, “Gaze locking: passive eye contact detection for human-object interaction,” in *Proceedings of the 26th annual ACM symposium on User interface software and technology - UIST ’13*, 2013, pp. 271–280.

[194] A. Torralba and A. A. Efros, “Unbiased look at dataset bias,” in *CVPR 2011*, 2011, pp. 1521–1528.

[195] S. K. Zhou, J. Shao, B. Georgescu, and D. Comaniciu, “Pairwise Active Appearance Model and Its Application to Echocardiography Tracking,” in *9th International Conference, Copenhagen, Denmark, October 1-6, 2006. Proceedings, Part I*, 2006, pp. 736–743.

[196] N. Gourier, D. Hall, and J. L. Crowle, “Estimating Face orientation from Robust Detection of Salient Facial Structures,” in *ICPR, International Workshop on Visual Observation of Deictic Gestures*, 2004, pp. 1–9.

[197] O. Ferhat, A. Llanza, and F. Vilariño, “A Feature-Based Gaze Estimation Algorithm for Natural Light Scenarios,” in *Pattern Recognition and Image Analysis*, 2015, pp. 569–576.

[198] J.-G. Wang and E. Sung, “Gaze determination via images of irises,” *Image Vis. Comput.*, vol. 19, no. 12, pp. 891–911, Oct. 2001.

[199] L. Jianfeng and L. Shigang, “Eye-Model-Based Gaze Estimation by RGB-D Camera,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 606–610.

[200] M. Karakaya, D. Barstow, H. Santos-Villalobos, J. Thompson, D. Bolme, and C. Boehnen, “Gaze estimation for off-angle iris recognition based on the biometric eye model,” in *Biometric and Surveillance Technology for Human and Activity Identification X Ioannis Kakadiaris; Walter J. Scheirer; Laurence G. Hassebrook Baltimore, Maryland, USA | April 29, 2013*, 2013, p. 87120F.

[201] Dong Hyun Yoo, Jae Heon Kim, Bang Rae Lee, and Myoung Jin Chung, “Non-contact eye gaze tracking system by mapping of corneal reflections,” in *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, 2002, pp. 101–106.

[202] C. H. Morimoto, A. Amir, and M. Flickner, “Detecting eye position and gaze from a single camera and 2 light sources,” in *Object recognition supported by user interaction for service robots*, 2002, vol. 4, pp. 314–317.

[203] C. Hennessey, B. Noureddin, and P. Lawrence, “A single camera eye-gaze tracking system with free head motion,” in *Proceedings of the 2006 symposium on Eye tracking research & applications - ETRA ’06*, 2006, p. 87.

[204] T. Nagamatsu, Y. Iwamoto, R. Sugano, J. Kamahara, N. Tanaka, and M. Yamamoto, “Gaze Estimation

Method Involving Corneal Reflection-Based Modeling of the Eye as a General Surface of Revolution about the Optical Axis of the Eye," *Ieice Trans. Inf. Syst.*, vol. E95.D, no. 6, pp. 1656–1667, 2012.

[205] E. Wood and A. Bulling, "EyeTab: model-based gaze estimation on unmodified tablet computers," in *Proceedings of the Symposium on Eye Tracking Research and Applications - ETRA '14*, 2014, pp. 207–210.

[206] P. Koutras and P. Maragos, "Estimation of eye gaze direction angles based on active appearance models," in *2015 IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 2424–2428.

[207] F. Lu, Y. Sugano, T. Okabe, and Y. Sato, "Adaptive Linear Regression for Appearance-Based Gaze Estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 10, pp. 2033–2046, Oct. 2014.

[208] Y. Sugano, Y. Matsushita, and Y. Sato, "Appearance-Based Gaze Estimation Using Visual Saliency," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 2, pp. 329–341, Feb. 2013.

[209] H. Yoshimura, M. Hori, T. Shimizu, and Y. Iwai, "Appearance-Based Gaze Estimation for Digital Signage Considering Head Pose," *Int. J. Mach. Learn. Comput.*, vol. 5, no. 6, 2015.

[210] Lim Choon Kiat and S. Ranganath, "One-time calibration eye gaze detection system," in *2004 International Conference on Image Processing, 2004. ICIP '04.*, 2004, vol. 2, pp. 873–876.

[211] F. Alnajar, T. Gevers, R. Valenti, and S. Ghebreab, "Calibration-Free Gaze Estimation Using Human Gaze Patterns," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 137–144.

[212] Y. Lu, Feng Sugano, Yusuke Okabe, Takahiro Sato, "Head Pose-Free Appearance-Based Gaze Sensing via Eye Image Synthesis," in *21st International Conference on Pattern Recognition (ICPR 2012), Tsukuba, Japan*, 2012, pp. 1008–1011.

[213] J. Ng and S. Gong, "Composite support vector machines for detection of faces across views and pose estimation," *Image Vis. Comput.*, vol. 20, no. 5–6, pp. 359–368, Apr. 2002.

[214] J.-G. Wang and E. Sung, "EM enhancement of 3D head pose estimated by point at infinity," *Image Vis. Comput.*, vol. 25, no. 12, pp. 1864–1874, Dec. 2007.

[215] M. Takatani, Y. Ariki, and T. Takiguchi, "Gaze Estimation Using Regression Analysis and AAMs Parameters Selected Based on Information Criterion," in *ACCV 2010 International Workshops, Queenstown, New Zealand, November 8-9, 2010, Revised Selected Papers, Part I*, 2011, pp. 400–409.

[216] Y. Matsumoto and A. Zelinsky, "An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement," in *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, 2000, pp. 499–504.

[217] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An Accurate O(n) Solution to the PnP Problem," *Int. J. Comput. Vis.*, vol. 81, no. 2, pp. 155–166, Feb. 2009.

[218] Pao-Tsun Lin, Shun-Feng Su, and Tsu-Tian Lee, "Support vector regression performance analysis and systematic parameter selection," in *Proceedings. 2005 IEEE International Joint Conference on Neural*

Networks, 2005., 2005, vol. 2, pp. 877–882.

[219] K. Hara and R. Chellappa, "Growing Regression Forests by Classification: Applications to Object Pose Estimation," in *13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II*, 2014, pp. 552–567.

[220] M. Fenzi, L. Leal-Taixe, B. Rosenhahn, and J. Ostermann, "Class Generative Models Based on Feature Regression for Pose Estimation of Object Categories," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 755–762.

[221] X. Zhen, Z. Wang, M. Yu, and S. Li, "Supervised descriptor learning for multi-output regression," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1211–1218.

[222] X. Geng and Y. Xia, "Head Pose Estimation Based on Multivariate Label Distribution," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1837–1842.

[223] G. Sandbach, S. Zafeiriou, M. Pantic, and L. Yin, "Static and dynamic 3D facial expression recognition: A comprehensive survey," *Image Vis. Comput.*, vol. 30, no. 10, pp. 683–697, Oct. 2012.

[224] M. F. Valstar *et al.*, "FERA 2015 - second Facial Expression Recognition and Analysis challenge," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2015, pp. 1–8.

[225] Y.-I. Tian, T. Kanade, and J. F. Cohn, "Recognizing action units for facial expression analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 97–115, 2001.

[226] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended Cohn-Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*, 2010, pp. 94–101.

[227] S. M. Mavadati, M. H. Mahoor, K. Bartlett, P. Trinh, and J. F. Cohn, "DISFA: A spontaneous facial action intensity database," *IEEE Trans. Affect. Comput.*, vol. 4, no. 2, pp. 151–160, 2013.

[228] S. M. Mavadati, M. H. Mahoor, K. Bartlett, and P. Trinh, "Automatic detection of non-posed facial action units," in *Proceedings - International Conference on Image Processing, ICIP*, 2012, pp. 1817–1820.

[229] J. T. Larsen, C. J. Norris, and J. T. Cacioppo, "Effects of positive and negative affect on electromyographic activity over zygomaticus major and corrugator supercilii.," *Psychophysiology*, vol. 40, no. 5, pp. 776–85, Sep. 2003.

[230] R. Stern, W. Ray, and K. Quigley, *Psychophysiological Recording*. Oxford University Press; Second Edition edition, 2000.

[231] P. Ekman, R. J. Davidson, and W. V. Friesen, "The Duchenne smile: Emotional expression and brain physiology: II.," *J. Pers. Soc. Psychol.*, vol. 58, no. 2, pp. 342–353, 1990.

[232] L. A. Jeni, J. F. Cohn, and F. De La Torre, “Facing Imbalanced Data Recommendations for the Use of Performance Metrics,” in *Int Conf Affect Comput Intell Interact Workshops*, 2013, pp. 245–251.

[233] S. Eleftheriadis, O. Rudovic, and M. Pantic, “Joint Facial Action Unit Detection and Feature Fusion: A Multi-Conditional Learning Approach,” *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5727–5742, 2016.

[234] Y. Song, D. McDuff, D. Vasisht, and A. Kapoor, “Exploiting sparsity and co-occurrence structure for action unit recognition,” in *Eleventh IEEE International Conference on Automatic Face and Gesture Recognition (FG 2015)*, 2015.

[235] Z. Wang, Y. Li, S. Wang, and Q. Ji, “Capturing Global Semantic Relationships for Facial Action Unit Recognition,” in *ICCV (International Conference on Computer Vision), Sydney, NSW, Australia*, 2015, pp. 3304–3311.

[236] S. Ghosh, E. Laksana, S. Scherer, and L.-P. Morency, “A Multi-label Convolutional Neural Network Approach to Cross-Domain Action Unit Detection,” in *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference*, 2015.

[237] and J. C. H. Paul Ekman, Wallace V. Friesen, *Facial Action Coding System: The Manual on CD ROM*. A Human Face, Salt Lake City, 2002.

[238] M. Liu, S. Shan, R. Wang, and X. Chen, “Learning Expressionlets on Spatio-temporal Manifold for Dynamic Facial Expression Recognition,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1749–1756.

[239] C. Sutton, “An Introduction to Conditional Random Fields,” *Found. Trends® Mach. Learn.*, vol. 4, no. 4, pp. 267–373, 2012.

[240] F. Wallhoff, B. Schuller, M. Hawellek, and G. Rigoll, “Efficient recognition of authentic dynamic facial expressions on the feedtum database,” in *2006 IEEE International Conference on Multimedia and Expo, ICME 2006 - Proceedings*, 2006, vol. 2006, pp. 493–496.

[241] M. Soleymani, J. Lichtenauer, T. Pun, and M. Pantic, “A Multimodal Database for Affect Recognition and Implicit Tagging,” *IEEE Trans. Affect. Comput.*, vol. 3, no. 1, pp. 42–55, Jan. 2012.

[242] A. Dhall, O. V. R. Murthy, R. Goecke, J. Joshi, and T. Gedeon, “Video and Image based Emotion Recognition Challenges in the Wild: EmotiW 2015,” in *ICMI ’15 Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, 2015, pp. 423–426.

[243] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep Face Recognition,” in *26th British Machine Vision Conference, BMVC*, 2015, pp. 1–12.

[244] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in *Computer Vision and Pattern Recognition, arXiv:1409.1556*, 2014.

[245] G. Turk and J. F. O’brien, “Modelling with implicit surfaces that interpolate,” *ACM Trans. Graph.*, vol. 21,

no. 4, pp. 855–873, Oct. 2002.

[246] C.-S. Wang, T.-R. Chang, and M.-C. Lin, “Reconstruction and Representation for 3D Implicit Surfaces,” in *International Conference, CSE 2011, Qingdao, China, July 9-10, 2011. Proceedings, Part II*, 2011, pp. 364–372.

[247] D. Attali, J.-D. Boissonnat, and A. Lieutier, “Complexity of the delaunay triangulation of points on surfaces the smooth case,” *Proc. Ninet. Conf. Comput. Geom. SCG 03*, p. 201, 2003.

[248] C. F. Nodine, H. L. Kundel, L. C. Toto, and E. A. Krupinski, “Recording and analyzing eye-position data using a microcomputer workstation,” *Behav. Res. Methods, Instruments, Comput.*, vol. 24, no. 3, pp. 475–485, Sep. 1992.

[249] D. S. Wooding, “Fixation maps,” in *Proceedings of the symposium on Eye tracking research & applications - ETRA ’02*, 2002, p. 31.

[250] C. Ehmke and Stephanie Wilson, “Identifying web usability problems from eye-tracking data,” in *Proceeding BCS-HCI ’07 Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it*, 2007, pp. 119–128.

[251] S. Stellmach, L. Nacke, and R. Dachselt, “3D attentional maps,” in *Proceedings of the International Conference on Advanced Visual Interfaces - AVI ’10*, 2010, p. 345.

[252] M. Maurus, J. H. Hammer, and J. Beyerer, “Realistic heatmap visualization for interactive analysis of 3D gaze data,” in *Proceedings of the Symposium on Eye Tracking Research and Applications - ETRA ’14*, 2014, pp. 295–298.

[253] Y. Ohtake, A. Belyaev, and H.-P. Seidel, “3D scattered data approximation with adaptive compactly supported radial basis functions,” in *Proceedings Shape Modeling Applications, 2004.*, 2004, pp. 31–39.

[254] L. Wang, B. Yuan, and Z. Miao, “Ellipsoid Criterion and Fuzzy C Means Algorithm for 3D Point Cloud Data Denoising,” in *2008 Congress on Image and Signal Processing*, 2008, pp. 361–365.

Appendix A

In this appendix, we show screenshots highlighting the core features of our tool, with brief descriptions. We show both thick and web clients. An administrator of our system is referred to as an evaluator, who typically uses the thick client. Users typically use the web client.

The thick (server) client gives more options, such as designing an experiment, and recording and visualizing user interactions. The thin client still provides some administrative options, but they are less than the thick client's options.

We will start first by showing screenshots of the Eclipse-based thick client. Figure 3-2 previously showed an overview this. An evaluator is able to create a new experiment from the Eclipse file menu; the created experiments are shown in the session file explorer (Figure 3-2).

Figure A-1 shows the experiment wizard. We set a name for the experiment, as well as the project it belongs to. For an experiment, we have a number of users, whose have profiles showing details such as names, ages, descriptions, fields of interests, and experience.

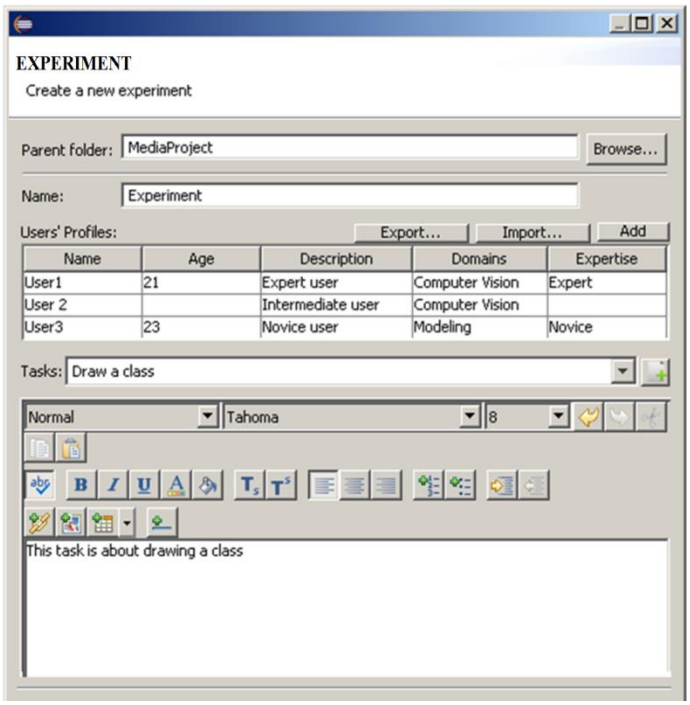


Figure A-1. Adding a new experiment using our thick client

An evaluator is still able to manage users' profiles and experiments from their properties (Figure A-2).

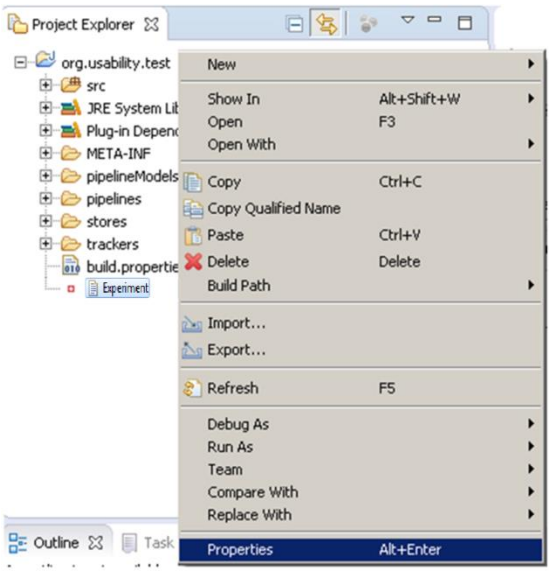


Figure A-2. Properties view of experiments

An evaluator is given a number of tabbed views when opening an experiment (Figure A-3), such as pipelines, record, model, navigation, heatmaps, 3D heatmaps, navigation traces, and scanpaths; all are discussed below.

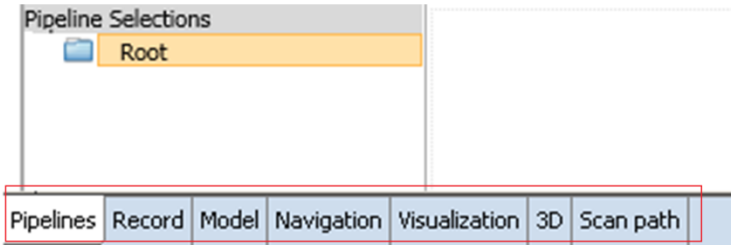


Figure A-3. Tabbed views of an experiment on the thick client

The *pipeline* view, as the name suggests, is used to handle pipeline creation. It has its own internal explorer (Figure A-4), which gives a virtual hierarchy of the pipelines defined in an experiment project.

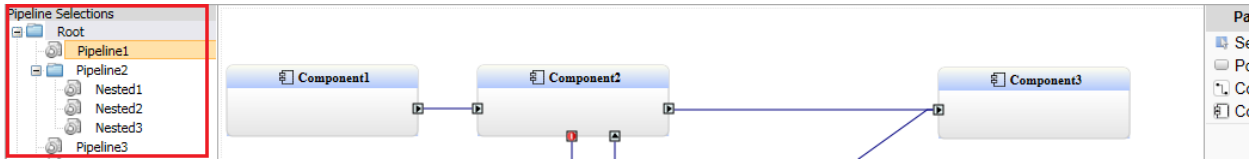


Figure A-4. Pipeline virtual file explorer

The pipeline view provides the required diagramming features. The basic items of a pipeline are components, ports, and connectors, for which we use the palette menu (Figure A-5) to draw on the drawing area (Figure A-6).

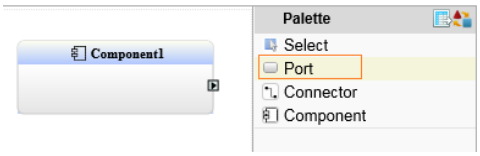


Figure A-5. An example of a palette menu item

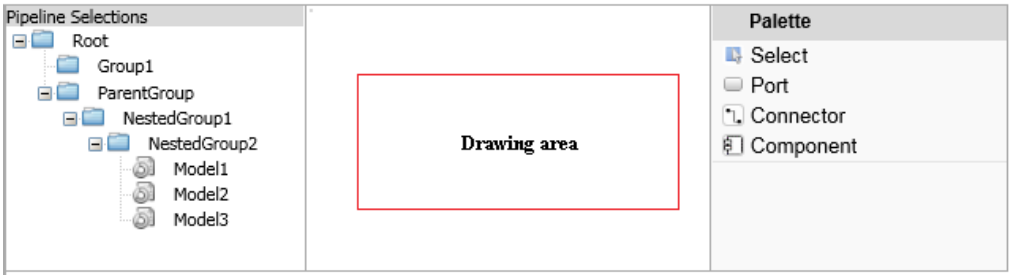


Figure A-6. Pipeline drawing area

Each element of a pipeline has its own property view. For instance, when selecting a port, its properties view will show fields such as its name and type (Figure A-7).

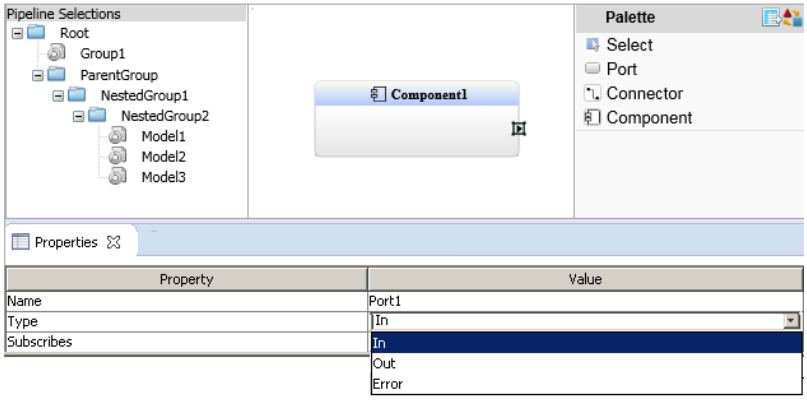


Figure A-7. The properties view of a port

A component has its own action code view, which contains its execution handling (Figure A-8). We previously showed some examples in Chapter 4, such as Snippet 4-2.

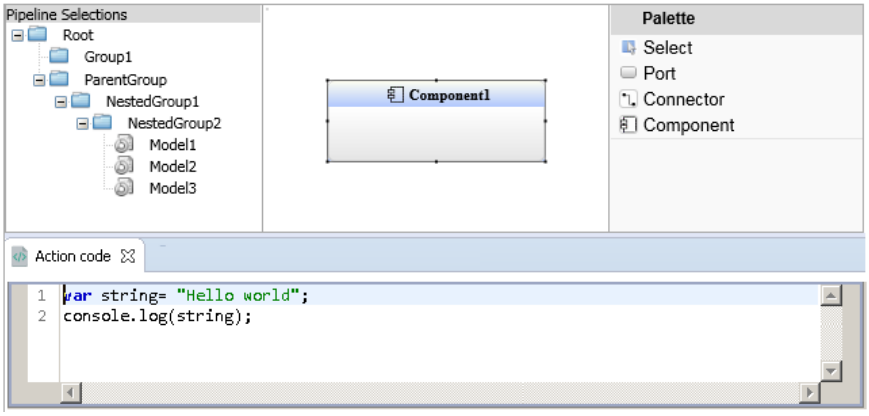


Figure A-8. An example of action code view

A pipeline or component can be used as an item (metacomponent) in other pipelines; the concepts were discussed in Section 4.3.2.3 For this, we use the import action (Figure A-9) to browse the available pipelines or components defined in an evaluation project (Figure A-10).

Figure A-9. The import metacomponent action

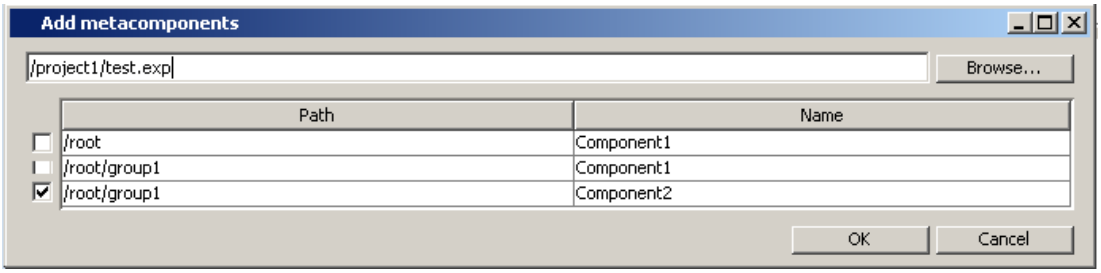


Figure A-10. Add metacomponent dialog

When importing a metacomponent, it is added to the palette menu (Figure A-11).

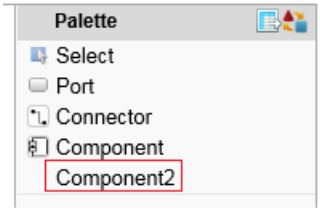


Figure A-11. New metacomponents in the palette menu

We can set an appropriate icon for a metacomponent from its property view (Figure A-12).

Property	Value
Name	Component2
Show State	
Priority	0
Default warnings severity	0
Default error severity	0
Custom Properties	Manage ...
Icon url	/somePath/icon.png

Figure A-12. Update a component icon URL

Hence, we become able to distinguish components visually from their icons (Figure A-13).

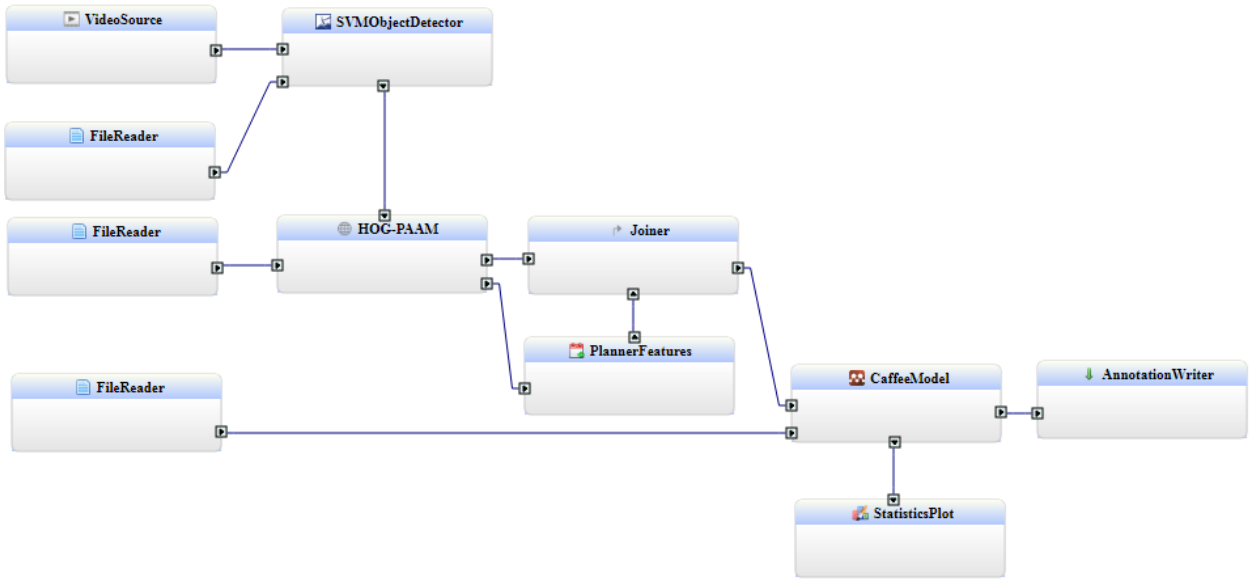


Figure A-13. An example of metacomponents of different icons

The *record* view is used to set up a recording session with a user (Figure A-14). A user is given a website to access and perform tasks against (Figure A-15); based on which their interactions will be recorded. Interactions can be in different forms such as key strokes, mouse clicks, and eye gaze.

Figure A-14. An overview of recording view

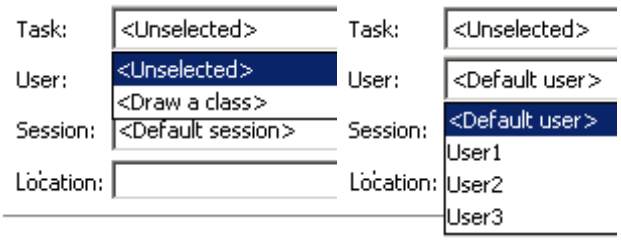


Figure A-15. User and task selections

An evaluator is able to create a session, which has number of tasks to perform (Figure A-16).

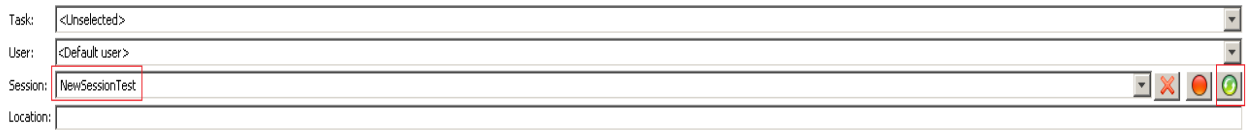


Figure A-16. The creation of a new session

When a user accesses a recording link, a widget will show options to start recording actions, and to select the tasks they are working on.

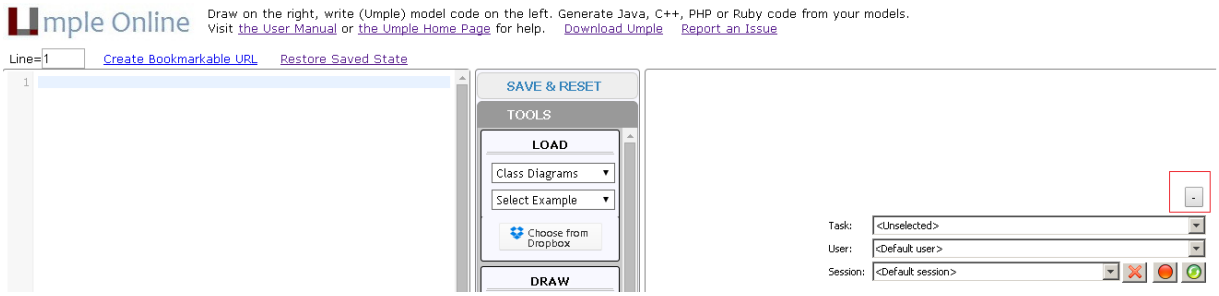


Figure A-17. The tracking perspective operating on a system to be evaluated

While a user is using the website, their mouse and keyboard interactions will be recorded. Additionally, if a user's camera is enabled, their gaze information as well as affective states will be recorded. All information recorded is sent to the server, such that it can be used in other visualization views (see tabbed views in Figure A-3), such as heatmaps, 3D heatmaps, and scanpaths (Chapter 10).

The information can also be displayed in the annotation board view (Figure 3-2), which gives a spatio-temporal representation; discussed previously in Section 3.5.

In the next few lines, we show screenshots of the web client. An overview of our web client is shown in Figure A-18.

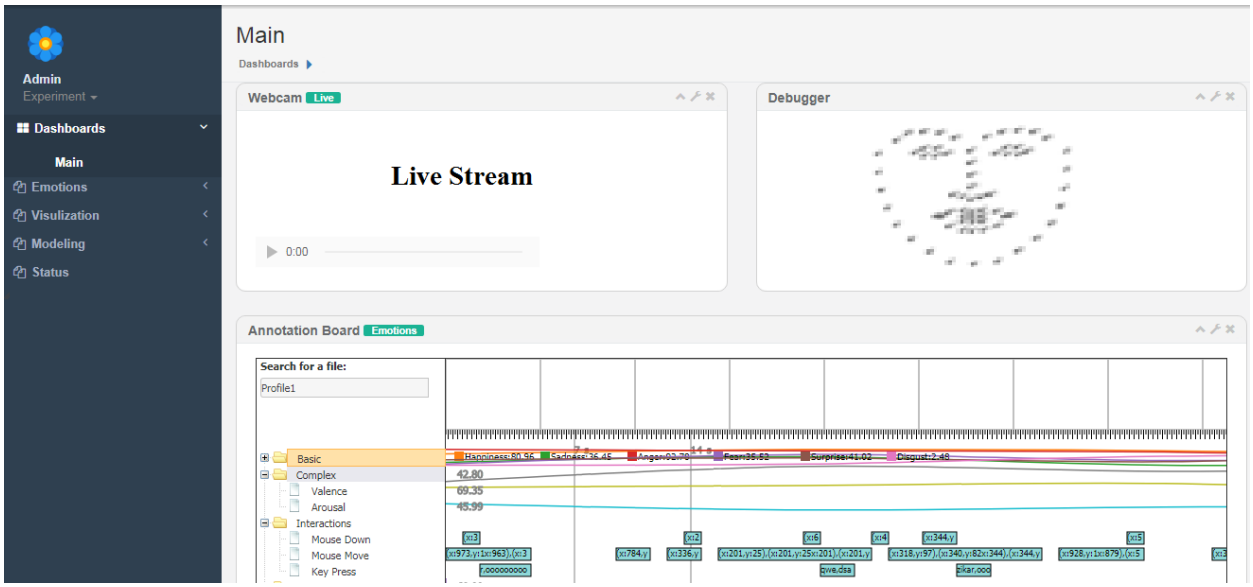


Figure A-18. An overview of the web client

In the web client, users are also able to use visualization techniques, as well as the annotation board (Figure A-19). We have a dashboard page, which gives options for users to customize their gadgets, such as web cameras, action units, eye gaze estimation, and annotation board.

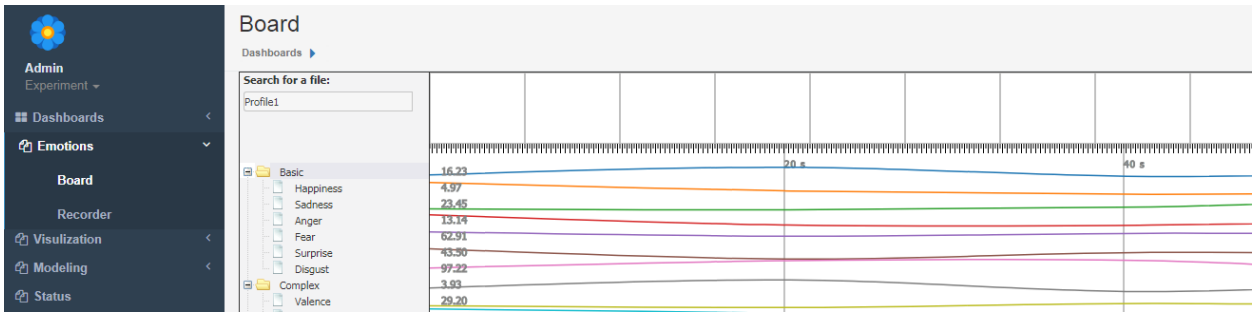


Figure A-19. The web-based annotation board

As well, we have a web version of pipelines (Figure A-20). However, the web client does not provide options such as creating metacomponents or importing other pipelines as the thick client does. For now, we use the web client to view pipelines, or apply minor edits such as updating layouts, or adding simple components, ports, and connectors.

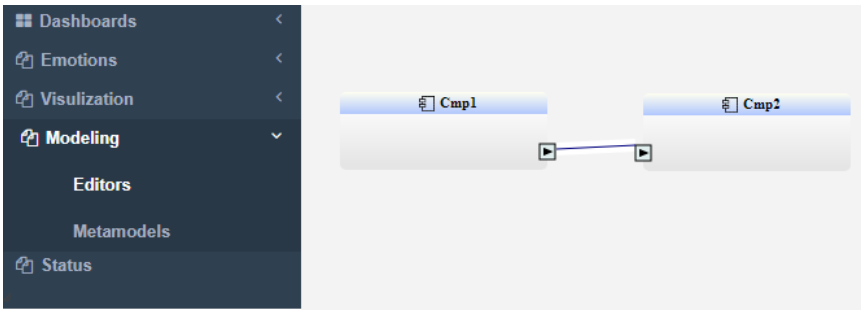


Figure A-20. An example of web-based pipeline