Engagement Detection Framework for Hand Gesture and Posture Recognition

by

Ghassem Tofighi

Bachelor of Science, Sharif University of Technology, 2006

Masters of Science, University of Isfahan, 2011

A dissertation

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2017

Engagement Detection Framework for Hand Gesture and Posture Recognition

Doctor of Philosophy 2017

Ghassem Tofighi

Electrical and Computer Engineering

Ryerson University

Abstract

Hand gesture and posture recognition play an important role in Human-Computer Interaction (HCI) applications. They are main attributes in object or environment manipulations using vision-based interfaces. However, before interpreting these gestures and postures as operational activities, a meaningful involvement with the target object should be detected. This meaningful involvement is called engagement. Upper-body posture gives significant information about user engagement.

In this research, for our first contribution, a novel multi-modal model for engagement detection, called Disengagement, Attention, Intention, Action (DAIA) framework is presented. *Disengagement* happens when the user is disengaged from the target object. *Attention* occurs when user pays attention to the target, but doesnt have the intention to take any actions. In *Intention* state, the user intends to perform an action, but still does not. *Action* state is when the user is performing an action with hand. Using DAIA, the spectrum of mental status for performing a manipulative action is quantized in a finite number of engagement states. The second contribution of this research is in designing multiple binary classifiers based on upper-body postures for state detection. 3D skeleton data is extracted from depth image and is used to extract body posture information. Combining the output of all binary classifiers in an order makes engagement feature vector. Moreover, This feature vector could be extended using other channels of biometric information such as voice or gaze. However the engagemnet classifiers recognize the state change with acceptable accuracy, minor changes in body postures or false detection of joint locations for some milliseconds may result in transition to another states. For removing this unwanted noise and increasing the accuracy of the system, an Finite State Machine (FSM) is designed based on the properties of human activities. The design of *Engagement FSM* is our third major contribution. Finally, rotation matrix is used to increase the number of samples for training the deep learning classifier for hand posture recognition.

Acknowledgements

During past years, I had amazing experiences that could not happen without support of many people, who they helped me, and supported me. I would like to thank my final PhD supervisor Prof. Kaamran Raahemifar who allowed me to follow my own industrial and research interests, and his advices helped me to complete this dissertation.

I also could not forget Prof. Anastasios Venetsanopoulos who advised and supported me by all means during the first three years of my PhD studies. He was a guiding light for generations of students; may his gentle soul rest in peace.

At last but not least, I offer my special thanks to my parents who never stopped supporting and encouraging me all the time.

Dedication

Dedicated to my Parents

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Hand gesture and posture recognition play an important role in Human-Computer Interaction (HCI) applications. Hand gestures and postures are main attributes in object or environment manipulations using vision-based interfaces. However, before interpreting these gestures and postures as operational activities, a meaningful involvement with the target object should be detected. This meaningful involvement is called engagement[1].

In this research, a novel multi-modal model for engagement detection, DAIA, is presented. DAIA consists of four different distinguishable mental states, namely, *Disengagement*, *Attention*, *Intention*, and *Action*. *Disengagement* happens when user is disengaged from target object. *Attention* occurs when the user has attention to the the target, but doesnt have intention for taking any actions. In *Intention* state, user intends to perform an action, but still not doing it. Finally, in Action state, user is performing an action with hand. Using DAIA, the spectrum of mental status for performing a manipulative action is quantized in a finite number of engagement states.

Based on important body postures factors regarding engagement, multiple binary classifiers are desinged for upper body postures for state detection. 3D skeleton data is extracted from depth image and is used to extract body posture information. By combining the output of all binary classifiers an engagement feature vector is created. This feature vector could be extended using other channels of biometric information such as voice or gaze. Using the engagement feature vector, an SVM classifier is trained to detect the most important transition in mental state which is *Intention to Act*, and indicates the transition from *Attention* to *Intention* or *Action*.

While these classifiers recognize the state change with acceptable accuracy, minor changes in body postures for a few milliseconds may result in transition to other states. To remove this unwanted noise and increase the accuracy of the system, an FSM is designed based on the properties of human activities. The design of *Engagement FSM* is the third major contribution of this research. Different hand postures in Intention state provide useful information for the system. For recognizing different hand postures in this state, a novel algorithm for hand-shape modeling is proposed. This algorithm is the forth major contribution of this research.

Another contribution of this research is in introducing a new hand posture dataset, HandReader dataset, which consists of 724 RGB images from 10 different hand postures of 74 idividuals. These postures are 10 American Sign Language (ASL) alphabets; namely $A, B, C, D, G, H, L, I, V$, and Y. Upper-body posture gives significant information about the user's engagement.

For vision-based hand posture and gesture recognition, beside common RGB cameras, development of a range of imaging technologies using Time-of-flight (ToF) cameras and RGB-D sensors such as Kinect, and miniaturized camera systems such as wearable cameras or smart-phones have paved the way for completely new types of techniques for hand posture and gesture recognition applications. These applications include desktop computer interaction, smart meeting rooms, sign language recognition, virtual object manipulations, and health-care intelligent systems.

In this chapter, we first introduce engagement metrics and definitions, and afterwards, review hand posture and gesture recognition technologies.

Communication between human and computer play an important role in our daily life. We are always looking for more convenient and faster ways of interacting with machines. Interaction between human and computer occurs at the user interface, which could be both software and hardware based. Characters or objects displayed by software on a computers monitor, or the input received from users via hardware peripherals such as keyboards and mouses, and other user interactions with large-scale computerized systems such as aircraft and power plants are some examples of human-computer interaction. Human-computer interaction involves studying, planning, and designing of the interaction between people (users) and computers. Human-computer interaction involves different human body positions and actions. There are many challenges to proposing a useful model for communication and interaction. However, the easiest way to interact with a machine is similar to the method used for the daily communication between two human beings.

Gestures are a form of nonverbal communication in which visible bodily actions are used to communicate important messages, either in place of or together with speech and in parallel with spoken words. Gestures include movement of the feet, hands, face, or other parts of the body.

Gestures are a form of nonverbal communication in which visible bodily actions are used to communicate important messages, either in place of or in parallel with spoken words. Gestures include movement of the feet, hands, face, or other parts of the body.

To achieve a natural human-computer interaction for virtual environment applications, the human hand could be considered as an input device. Hand gesture is frequently used in everyday life. It is also an important component of body language. Hand gestures are a powerful human- to-human communication modality. Compared with the traditional Human-Computer Interaction devices, hand gestures are less intrusive and more convenient in exploring the 3D virtual worlds [2]. They also can transfer information between human and machine even faster. The human hand is a complex articulated object consisting of many parts. Considering the global hand pose and each finger joint, the human hand motion has roughly 27 Degrees of Freedom(DOF) [2]. Therefore, hand gesture/posture recognition is a challenging issue in computer vision and pattern recognition. Furthermore, hand gestures in sign language play an important role in the daily life of deaf people. Sign language is a language which uses manual communication and body language to convey meanings. Sign language involves combining postures, orientation and movement of the hands, arms or body, along with facial expressions to express a speakers thoughts. This paper examines the use of computer vision and pattern recognition for interpreting 10 different non-motion-based signs from American Sign Language.

1.1 Motivation

Gesture recognition technologies, especially hand gesture recognition, have variety of applications in different fields. Everyday, new devices and consumer electronics based on these technologies come to the market. For instance, some years ago, Microsoft designed a motion detection device called Kinect, which is based on the gesture recognition technologies. Kinect is a motion sensing input introduced for Xbox 360 video game console and also Windows PCs. Kinect uses an infrared camera to create a depth map of the scene in front of the camera. It enables users to control and interact with Xbox 360 through gestures and spoken commands and without the need for a game controller. Selling a total of 8 million units in its first 60 days, Kinect holds a Guinness World Record for being the fastest selling consumer electronics device. Following this recognition, Kinect was deployed in many applications as an interactive input device[3, 4].

The application of hand gesture and posture recognition technology is not limited to game controllers. Gesture recognition is useful for processing information which is not conveyed through speech or type. Sign language recognition [5, 6, 7, 8], socially assistive robotics[9, 10, 11], directional indication through pointing[12, 13], virtual controllers, and remote controllers[14, 15] are a few examples of technologies employing hand gesture and posture recognition. The main focus of this research is on pixel-based posture recognition using a single camera. This approach is useful in both RGB and depth-based hand posture classification.

1.1.1 Market Trend

The market is changing rapidly due to evolving technology and increasingly more original equipment manufacturer are adopting gesture recognition technology. Markets and Markets report that from 2013 to 2018 the gesture recognition market is estimated to grow at a healthy Compound Annual Growth Rate(CAGR), and to exceed \$15.02 billion by the end of 2018. Analysts predict that the Global Gesture Recognition market grow at a CAGR of 29.2% over the period from 2013 to 2018. Currently consumer electronics application contributes to more than 99% of the global gesture recognition market. As per the report published, the healthcare application is expected to emerge as a significant market for gesture recognition technologies over the next five years. The automotive application of gesture recognition is expected to be commercialized in 2015 [16]. Gesture recognition is one of the fastest growing technologies, and in the near future lots of new applications will be developed based on this novel type of human-computer interaction.

1.2 Challenging Issues

1.2.1 Engagement Detection Challenges

User mental status detection has variety of applications in human activity recognition systems. Without a proper algorithm for detecting human intention to interact, the vision-based system is always on, therefore any kind of activity may interpreted as an interaction. This problem is known as "Midas Touch" problem. In the most of real applications such as gesture-based game controllers, baseline approach is defining some *focus gestures* such as *waving hand* to let the controller understand the user is going to start a meaningful interaction. Furthermore, sequences of gestures should be segmented for gesture recognition; this is known as *Gesture Segmentation*.

Designing a framework capable of recognizing user's intention to interact without any focus gesture is highly desirable. Furthermore, without a proper gesture segmentation technique, gesture recognition systems could not work accurately.

A variety of studies strives for a multi-modal approach using some features of facial expression, body motion, voice, or seat pressure to elucidate on mental states for performing an action. Body posture gives important information about engagement. Various approaches have been investigated based on body language analysis to improve human-computer interaction. Intention to engage with an agent e.g. a robot [17, 18], or interactive display [19], are some of these studies. Measuring the engagement intent is used in service robots to identify relevant gestures from irrelevant gestures[17, 20, 21].In addition, intention to engage with a display for improving user identification is addressed in Schwarz et al. [19]. The role of body pose and motion in users interest detection using body tracking systems such as Kinect has been addressed in several research [22, 23, 24, 25]. In our study, after investigating all related literature, we propose a novel framework for engagement and ”Intention to Interact” detection which is an essential tool for hand posture and gesture recognition.

Moreover, for hand posture recognition, we study several shape descriptors to find the best compromise between accuracy and computation of hand posture recognition which will be appropriate in a real-time application. We study different algorithms used for feature extraction from static hand gestures, as well as the classification techniques employed for recognition. Multiple feature extraction methodologies and classification techniques have been compared in recent comparative studies [26, 27]. Bourennane and Fossati[26] examined two families of contour-based Fourier descriptors and two sets of region-based moments, all of them invariant to translation, rotation and scale changes of hands. These approaches are also independent from the cameras view point. The experiments were performed on the Triesch benchmark database and on their large internal and private dataset. Their results show that the best recognition rates with Euclidean distance are given by Fourier Descriptors. Bourennane and Fossati[26] also proposed their evaluation with different classifiers with Fourier Descriptor to examine whether the results were better than the Euclidean distance. They employed Bayesian classifier, support vector machine, and k-NN as their classification techniques. The recognition rates also increased with respect to Euclidean distance. The authors concluded the results were similar for all three types of classifiers, with k-NN being slightly better.

Trigueiros et al. [27] also presented a comparative study of seven different algorithms used for feature extraction for hand posture modeling. These algorithms are Radial Signature, Histogram of Gradients (HOG), Centroid Distance Signature, Local Binary Patterns(LBP), Fourier Descriptors, Centroid Distance Fourier Descriptors, and the Shi-Tomasi Corner Detector. Trigueiros and colleagues analyzed these algorithms with RapidMiner in order to find the best learner. Their results showed that when Radial Signature and Centroid Distance are used separately, they give better results in terms of computational complexity.

1.3 Applications

The applications of this study are as follows:

- To create a framework which assists in hand posture and gesture recognition
- To create a smart meeting room that detects the level of people's engagement
- To create a real-time system which allows users to control the virtual objects with hand gestures and postures
- To automatically find the engaged person with the object or screen in mulit-user platform
- To control a mobile robot using hand postures and gestures
- To help deaf people interact with computers using American Sign Language (ASL) or other sign languages that have large classes of postures
- To create a real-time system that can learn from a limited number of posture samples
- To create a real-time system that allows users to train their own hand posture classes (for example for controlling objects in a computer game), and is optimized automatically based on the set of postures defined by the user.
- To control a mobile robot using hand postures.
- To allow directional indication through pointing (for example turning lights off or on by pointing to the source, or changing the TV channel using different postures); a strategy which is useful in smart homes.
- To create virtual controllers in Virtual Reality (VR) systems.

1.4 Objectives and Contributions

We can briefly summarize the objectives of this research as follows:

- To find a solution for "Midas Touch" Problem
- To quantize mental states of engagement into distinguishable states
- To find out a meaningful relationship between Upper-body posture and engagement states
- To discover transition rules among engagement states
- To recognize hand postures

The following contributions addresses the objectives of this study:

- **Engagement States**: Spectrum of mental states of engagement is quantized in four distinguishable states (Disengagement, Attention, Intention, Action). However some of these states are mentioned in previous research with similar applications of engagement detection (Michalowski et al.[28], Bianchi et al.[22] ,Schwarz et al.[19], Vaufreydaz et al.[29] Leite et al.[30], but to the best knowledge of author, this is the first time that engagement detection is quantized to these four distinguishable mental states which is very useful for gesture segmentation.
- **Engagement Feature Vector**: A new mapping between *Human Body Joint 3D Positions* ($P_{3\times N\times T}$) and *Engagement Feature Vector* ($E_{C\times T}$) is offered as follows:

$$f : P_{3\times N\times T} \longmapsto E_{C\times T} \tag{1.1}$$

In equation 1.1, C is the number of binary classifiers. Similar to Schwarz et al. (2014)[19], we have used multiple weak binary classifiers to determine whether a basic feature exists in human body posture. For instance if the location of hand is above head or not. By the way, we have extended binary classifiers to more meaningful features and designed 37 classifiers instead of 5 presented in [19]. However in this research we have just mentioned to body joint locations, our mapping function could be extended as multi-modal data to Engagement Feature Vector. For example, we can easily add another basic feature such as gaze binary classifier (1 if not gazing, 0 if otherwise) or voice binary classifier (1 if user voice exists, 0 if otherwise). So, our generalized mapping function is as follows which maps a combination of all multi-modal data (M_T) to Engagement Feature Vector ($E_{C\times T}$) as follows:

$$f : M_T \longmapsto E_{C\times T} \tag{1.2}$$

- **Engagement Finite State Machine (FSM)**: The main goal for designing Engagement FSM is classifying each of the four states not only based on Engagement Feature Vector, but also based on the previous state. The Engagement FSM brings memory for the classifier. Without FSM, Engagement Classifiers should only decide based on the current frame and features extracted from human, e.g. 3D joint locations. By introducing a Finite State Machine and designing proper *Transition* and *Guard* conditions, the system can classify more precisely.

- **Training Hand Posture Classifiers with a small number of samples**: We have designed several classifiers for hand posture recognition. We created a dataset of 740 samples for 10 different hand postures from 74 different individuals. We have shown how to increase the number of samples for training the deep learning classifier by rotation matrix for small deviations. Our approach helped in creating hand posture classifiers which are sensitive to direction of hand postures, but invariant to small deviation, scale, and transformation.

In the following section, a more detailed outline of our approach is presented.

1.5 Our Approach

In all feature extraction and classification steps, we consider using and developing real-time algorithms to achieve better results in terms of computational complexity. Furthermore, the current widely used RGB-D sensors, such as Kinect, do not give us a very high depth resolution for objects which are very close together (i.e., the different parts of hand in each posture). Therefore, we can't count on the internal features of the depth image of a hand for creating a 3D model. In addition, even if we were able to create a 3D model, it would have been computationally expensive and not useful for a real-time application. Considering these requirements and limitations, we chose 2D appearance-based approach and will develop algorithms based on this model.

After studying all recent descriptor for modeling the shape of a hand, our goal is to present a new shape descriptor, or optimize the best available one, for static hand postures that provides better results with minimum computational complexity. The computation complexity should be suitable for usage in real-time applications. Furthermore, all of the above mentioned shape descriptors are reported as invariant to translation, rotation and scale changes of hands. This set of properties are essential for an efficient shape descriptor[31]. However, almost similar hand postures with different orientation may be used for not the same signs. For instance, I and J have similar static posture shapes in ASL, but the orientation of these two postures is different. A rotation invariant shape descriptor puts both these postures into the same class. In this research, we try to develop a shape descriptor which could be rotation invariant or rotation sensitive, based on our decision. Even if our descriptor is rotation invariant, we are still able to report the orientation angle of the posture with respect to the trained model after classification.

For data analysis, feature selection and extraction, dataset preparation and data transformation are the key steps. To construct a reliable model of hand postures, it is necessary to have the hand shapes of a large number of people. Our study addresses this issue by introducing a public RGB and binary static hand posture dataset. This dataset might be used as a benchmark dataset for future research.

Currently, researchers use their own internal datasets to evaluate the performance of feature descriptors. For instance, Bourennane and Fossati[26] used an internal dataset that was not published publicly, and therefore, cannot be compared with other studies. In addition, although there are publicly available datasets, such as NTU-Microsoft-Kinect- HandGesture dataset[32], that contain 1000 cases, these datasets have been obtained from only 10 different persons. Researchers have also used Trieschs static hand posture dataset[33] as a benchmark. Trieschs dataset consists of 10 hand postures (A, B, C, D, G, H, I , L, V, Y) from 24 people with 3 different backgrounds (light, dark, complex). Most real time approaches, after some preprocessing steps convert a hand shape to a 2D binary image called silhouettes, and extract features from these black and white binary images. In addition, as mentioned earlier, after introducing RGB-D sensors, the complex background will not be one of the main challenges for extracting hand from the background. Hence, among all these images, those with dark or light background carry distinct information about the hand shape and posture for different people and are useful for the purpose of hand shape modeling. We chose the images with dark background for extracting the hand shapes. We also extended Trieschs static hand posture dataset by creating HandReader static hand posture dataset. Our dataset consists of 500 images from 10 different hand postures showing 10 different alphabets in American Sign Language. The dataset was created by capturing images from 50 people, both males and females, performing the 10 postures in front of a camera. After some preprocessing steps, we extracted binary images from Trieschs static hand posture dataset with dark background and our own dataset to create a new static hand posture dataset, called HandReader2.

HandReader2 is a new dataset of hand-shape silhouettes which contains 128×128 binary square images with the actual shape of hands exactly centered in the squares. This dataset consists of 740 square silhouettes from 10 different hand postures performed by 74 different people. From these 740 images, 500 are extracted from our HandReader dataset and 240 from Trieschs static hand posture dataset. These postures are 10 American Sign Language alphabets (A, B, C, D, G, H, L, I , V, and

Y). Creating this dataset allows us to make a more realistic hand shape model based on considerably different hand shapes. HandReader2 is publicly available and can be used by anyone who is interested in hand posture modeling or other related areas. 6.4 shows a set of these 10 silhouettes from HandReader2 dataset.

1.6 Research Publications

List of all publications and their full-text related to this research could be accessed from Google Scholar page[34] or author's homepage[35, 36]. There is also a continuous research for Alzheimer's Disease detection [37, 38, 39, 40, 41] using Deep Learning approach that author is involved.

1.7 Summary

Hand gesture and posture recognition play important role in HCI applications. They are main attributes in object or environment manipulations using vision-based interfaces. However, before interpreting these gestures and postures as operational activities, a meaningful involvement with the target object should be detected. This meaningful involvement is called engagement.

In this chapter, the outline of our research for engagement detection is presented as DAIA framework. DAIA consists of four different distinguishable mental states which are *Disengagement*, *Attention*, *Intention*, and *Action*. *Disengagement* happens when user is disengaged from target object.

Chapter 2

Literature Survey

2.1 Engagement Theories and Modeling

The understanding and analysis of engagement has a long history. It originated in psychological[42], educational[43], and organizational fields[44] to investigate and evaluate the focus, and thereby outcome, of learning and work. It has been a qualitative, observational evaluation of an individuals state, that focused on long term engagement rather than situational participation.

As more artificial agents are developed the HCI community started focusing on the detection of engagement of the human user on an immediate level to improve computer responses for more natural interfaces. Qualitative and automated methods of engagement detection, qualitative and automated, enables us to provide new insights into automated work engagement detection in the work environment.

One of the advantages of engagement detection is gesture segmentation. The gesture segmentation problem is the first step towards visual gesture recognition, i.e., detection, analysis and recognition of gestures from sequences of frames[45].

For gesture recognition, gesture segmentation involves finding the frame in which a gesture starts, and the last frame related to that specific gesture. Therefore, gesture segmentation is necessary for every gesture recognition system. For instance, Bulzacki [46] created a dataset of gestures and afterwards using Polynomial Motion Approximation and Principal Component Analysis (PCA) techniques achieved gesture classification. However, Bulzacki segmented the gestures manually and then fed them to the gesture segmentation system for classification. He assumed that he already had a sequence of meaningful frames and tried to develop an algorithm to classify those different frame sequences to different gestures.

To automatically segment those frames for hand postures and gestures, we have offered DAIA to manually segment a continuous video clip to one of four different meaningful engagement states. Using our approach, the system can automatically segment those frames related to gestures and feed them to any gesture recognition system such as the one provided by Bulzacki. On the other hand, in this research, we did not offer a new gesture recognition system, rather we proposed a system which could be used for gesture segmentation.

This section aims to capture the theoretical work and practical implementations around engagement state detection in group interaction and HCI. Theoretical definition of engagement is presented and the description of an engagement evaluation scale is discussed. In addition, the research methodology and the implementation of the approach is investigated.

2.1.1 Engagement Definition

Engagement has been investigated in various fields such as education, organizational behavior, work, or media. Several definitions have been suggested for engagement:

- the value that a participant in an interaction attributes to the goal of being together with the other participant(s) and continuing interaction [47, 48]
- The process by which two (or more) participants establish, maintain, and end their perceived connection; Engagement is directly related to attention [1, 17, 48, 49]
- Effort without distress[50]
- A meaningful involvement [43]
- Enabled through vigor, dedication, and absorption [44, 51]

2.1.2 Engagement Modeling

In our previous research[52, 53], we defined engagement as an attentive state of listening, observing, and giving feedback, leading into protagonist action in group interaction. The focus of our previous research was engagement in meeting rooms[52]. For the present research, we define engagement as a meaningful involvement with virtual and real objects for manipulation. This manipulation occurs using hand gestures and postures.

Intention to Act

One aspect of engagement highlighted in this research is a a crucial moment in interaction that user has intention to perform an action. This crucial moment is called *"Intention"* or *"Intention to Act"*. This moment is crucial because it indicates a strong change in mental user state. Intention to Act is understood as:

- Instructions that people give to themselves to behave in certain ways[54]
- Most important predictor of a persons behavior[55]
- Mental states of individuals in which intentions are seen as internal commitments to perform an action while in a certain mental state [56]
- a function of both attitudes toward a behavior and subjective norms toward that behavior, which has been found to predict actual behavior [57]
- An act or instance of determining mentally upon some action or result (Dictionary.com)
- Purpose or attitude toward the effect of one's actions or conduct a determination to act in a certain way (Merriam Webster dictionary)
- is a mental state that represents a commitment to carrying out an action or actions in the future (Wikipedia)

In our research, we use the following intention definition: *Intention is a mental state defined by an internal commitment to perform a specific action in the immediate future.* It is of relevance for this research because it defines a specific form of engagement that is relevant for human-computer interaction specifically. It helps create a comprehensive understanding and interpretation of human-to-human and human-computer interaction for seamless meeting flows[53].

2.1.3 Engagement Features Extraction

A variety of studies strives for a multi-modal approach using some features of facial expression, body motion, voice, or seat pressure to elucidate on mental states.

Body posture gives important information about engagement. Various approaches have been investigated based on body language analysis to improve human-computer interaction. Intention to engage with an agent e.g. a robot [17, 18], or interactive display [19], are some of these studies. Measuring the engagement intent is used in service robots to identify relevant gestures from irrelevant gestures which is known as Midas Touch Problem [17, 20]. Another research interest is related to learner engagement with robotic companions or interfaces[21]. In addition, Intention to engage with a display for improving user identification is addressed in Schwarz et al. [19]. The role of body pose and motion in users interest detection using body tracking systems such as Kinect has been addressed in several research [22, 23, 24, 25].

Benkaouar et al. [58] is discussing gaze and upper body posture for engagement detection. Schwarz et. al [19] used combination of gaze, upper body and arm position for the purpose of intention detection in engagement. Vaufreydaz et. al [17] used gaze and proxemics and Salam et. al [48] employed human state observation for engagement detection. Engell et al. used gaze and facial expression[59] and Balaban et al. [60] employed weight, head, and upper body motion; Scherer et al. [] and Dael et. al [51] discuss voice, face, posture. Using Finite State Machine (FSM) for multi-modal system modeling is addressed in multiple research[61],[62],[63],[64]. A study by Frank and Fruchter (2015[65]) uses cognitive flexibility as an internal evaluation of engagement during meetings. However, this approach as well as self-reports require active involvement of meeting participants. Research in this field focuses on the human body and its features to generate indicators of mental states. Non-verbal behavior and body language expresses emotions (Schindler et al. 2008, Mead 2011[20]); it conveys information about the speaker's internal state, and attitude toward the addressees (Gutwin & Penner 2002[66]). Both body posture and gesture (Ekman 1999[67], Ekman and Friesen 1972[68]) have been identified as important social signals that indicate mental states and thereby also engagement (Figure 2.1). We use the preexisting qualitative research to develop an Engagement Framework offering quantitative measurable indicators.

Various approaches have been investigated to use body language to improve interactions with digital agents such as assistant robots and displays (Vaufrydaz 2015[29], Schwarz 2014[19]). These studies use varying aspects of non-verbal language, such as upper body pose (Mead 2011[20], Schwarz 2014[19]), proxemics (Vaufreydaz 2015[29]), facial expression and head posture (Vaufreydaz 2015[29]), or the absence of movement (Witchel 2014 [69]) and sometimes the combination of a small set of features from facial expression and posture in multi-modal approaches.

Weight and weight distribution on a seat has been used to measure engagement (Mota & Picard 2003[70], DeMello et al. 2007[71], Balaban 2004 [60]). Weight is an indicator of body posture that can be measured without affecting the participant.

The PBL Lab researchers[72] developed a cloud service and application called eRing (engagement Ring) to detect and provide real-time feedback of learners degree of engagement during a virtual interaction. eRing collects body motion data using Microsoft Kinect sensor, analyzes and interprets a small set of the body motions and body positions including head, shoulder, and hand joints. eRing provides real-time feedback of the degree of engagement based on three states disengaged, neutral listen, and engaged. The body motion analysis is performed for two units of analysis individual learner and team. eRing runs on a Microsoft Azure cloud platform. eRing was used in the architecture, engineering, construction (AEC) Global Teamwork course testbed in 2014 and 2015.

All of the studies mentioned in this section discuss a very small set of potential classifiers. They also do not make full usage of the amount of qualitative research on non-verbal body language and its indication of mental states. In this research, we address these features for engagement modeling and detection.

2.1.4 Observable Engagement

In this research we have focused on observable indicators of engagement that allow a non-intrusive evaluation of the user without disrupting the activity. Research in this field has focused on the human body and its features to generate indicators of mental states.

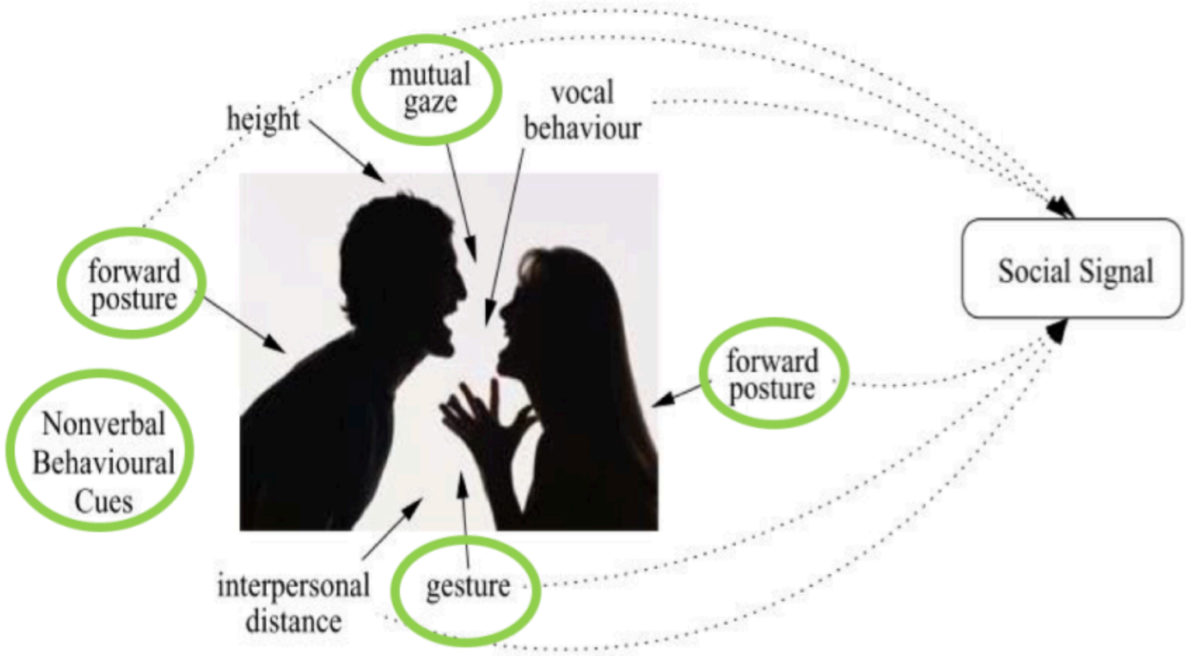
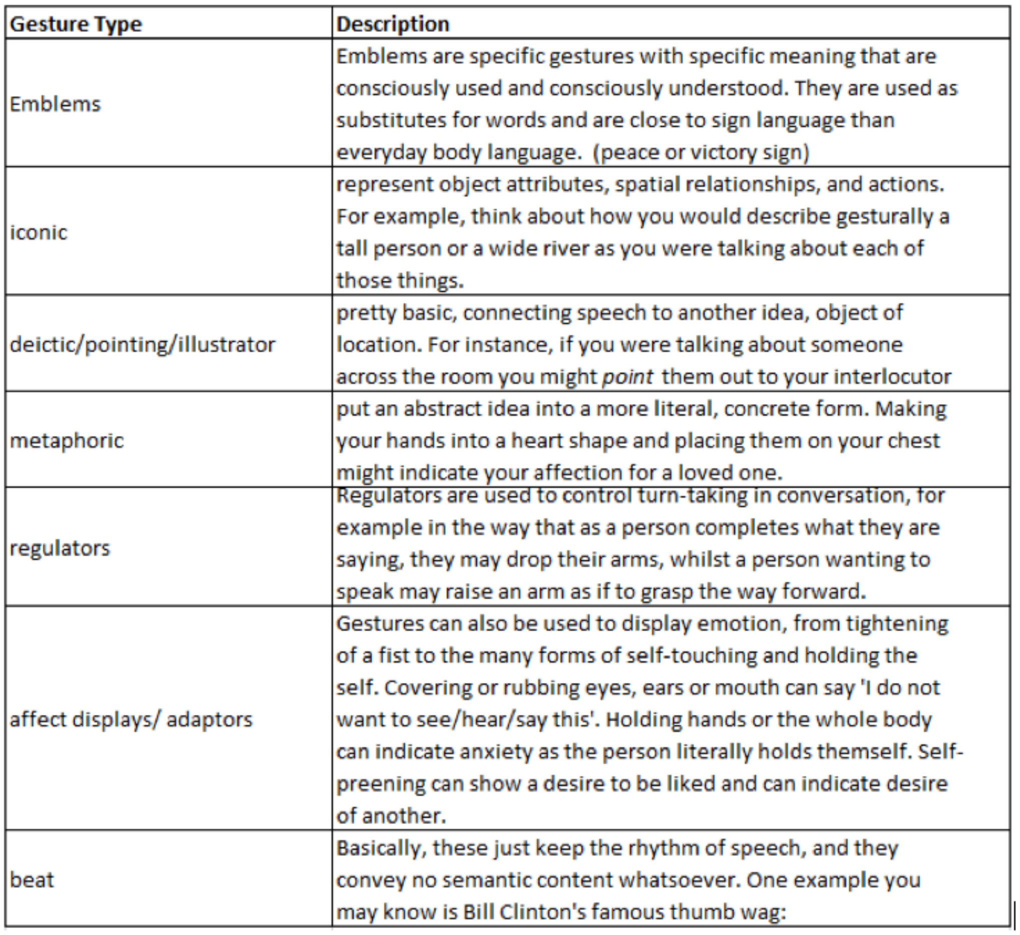


Figure 2.1: Body related social signals

Gesture are of specific interest because it is a motion of the body that contains information [73, 74, 75, 76]. Approximately 35% of hand actions are communicative gestures[66]. Gestures occur synchronous and co-expressive with speech, and are not redundant. Gestures are frequent and accompany about 90% of spoken utterances in descriptive discourse[77].

Gestures have been classified into co-speech gestures, emblems, iconic, deictic, metaphoric, and display gestures, on which we focus as they express mental states, regulators, adapters, beat[67, 68, 78, 79, 80]. Table 2.1 gathers these gestures.

All of the afore mentioned studies discuss a very small set of potential classifiers and do not make full



Gesture Type	Description
Emblems	Emblems are specific gestures with specific meaning that are consciously used and consciously understood. They are used as substitutes for words and are close to sign language than everyday body language. (peace or victory sign)
iconic	represent object attributes, spatial relationships, and actions. For example, think about how you would describe gesturally a tall person or a wide river as you were talking about each of those things.
deictic/pointing/illustrator	pretty basic, connecting speech to another idea, object of location. For instance, if you were talking about someone across the room you might *point* them out to your interlocutor
metaphoric	put an abstract idea into a more literal, concrete form. Making your hands into a heart shape and placing them on your chest might indicate your affection for a loved one.
regulators	Regulators are used to control turn-taking in conversation, for example in the way that as a person completes what they are saying, they may drop their arms, whilst a person wanting to speak may raise an arm as if to grasp the way forward.
affect displays/ adaptors	Gestures can also be used to display emotion, from tightening of a fist to the many forms of self-touching and holding the self. Covering or rubbing eyes, ears or mouth can say 'I do not want to see/hear/say this'. Holding hands or the whole body can indicate anxiety as the person literally holds themself. Self-preening can show a desire to be liked and can indicate desire of another.
beat	Basically, these just keep the rhythm of speech, and they convey no semantic content whatsoever. One example you may know is Bill Clinton's famous thumb wag:

Table 2.1: Gesture types related to engagement

usage of the amount of qualitative research conducted on non-verbal body language and its indication of mental states.

2.1.5 Engagement Classification Framework

Our goal for engagement classification is providing a framework for hand posture and gesture recognition. Based on engagement frameworks and investigations on engagement that are mentioned, we have developed a new engagement scale for interaction in meeting room scenario[52, 53].

Engagement for this application is the level of participation in a meeting interaction. It consists of six stages: *Disengagement*, no participation and attention; *Relaxed Engagement*, attention but no participation; *Involved Engagement*, attention and non-verbal feedback; *Intention to Act*, preparation for active participation in protagonistic role; *Action*, speaking and/or interacting with display or participants; *Involved Action*, highly engaged and involved interaction with intense gesture and voice.

The six stages which is shown in Figure 2.2 help understand different team dynamics and user states that affect the meeting interaction, productivity and effectiveness.

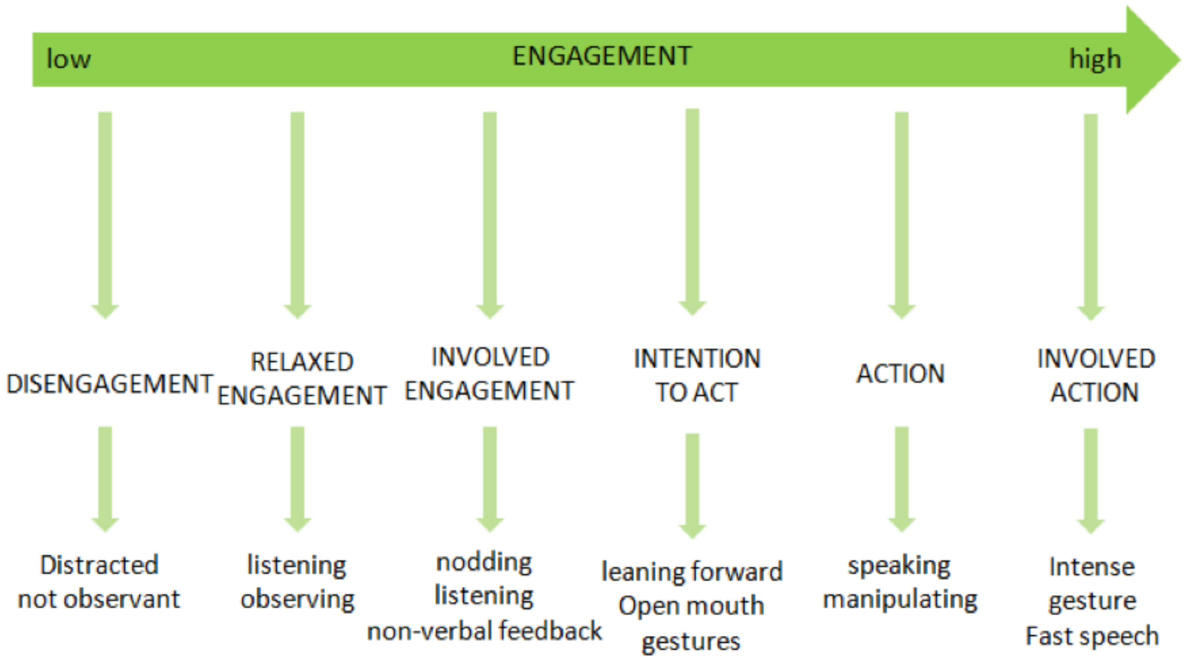


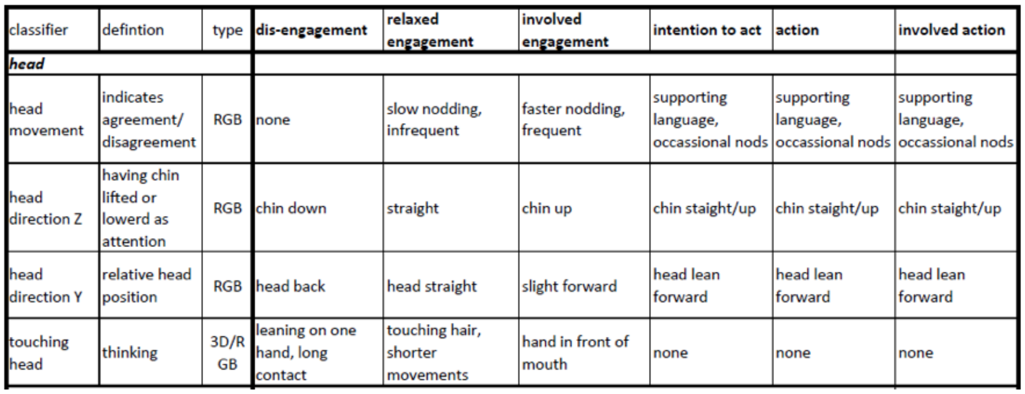
Figure 2.2: Engagement Scale

2.1.6 Multi-modal Classifiers

Literature reports specific observable patterns that indicate mental states. This report collects a variety of indicators for engagement presented in various sources [19, 20, 21, 42, 81, 82]. These indicators are abstracted into classifiers that are easy to observe. Each classifier has a specific expression for different engagement levels. The *Intention to Act*, as a stage specifically relevant to HCI, uses a binary classification. List of suggested classifiers in our technology report [53] for head (Table 2.2), face (Table

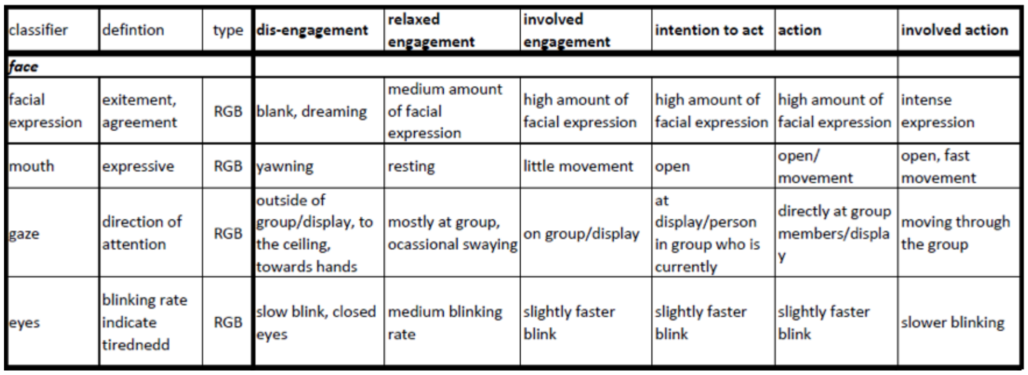
2.3), arms (Table 2.4), body (Table 2.5), and voice (Table 2.6) in our previous research [52, 53] for each scale of engagement in Figure 2.2.

Table 2.2: Head Classifiers



classifier	defintion	type	dis-engagement	relaxed engagement	involved engagement	intention to act	action	involved action
head								
head movement	indicates agreement/ disagreement	RGB	none	slow nodding, infrequent	faster nodding, frequent	supporting language, occassional nods	supporting language, occassional nods	supporting language, occassional nods
head direction Z	having chin lifted or lowerd as attention	RGB	chin down	straight	chin up	chin staight/up	chin staight/up	chin staight/up
head direction Y	relative head position	RGB	head back	head straight	slight forward	head lean forward	head lean forward	head lean forward
touching head	thinking	3D/RGB	leaning on one hand, long contact	touching hair, shorter movements	hand in front of mouth	none	none	none

Table 2.3: Face Classifiers



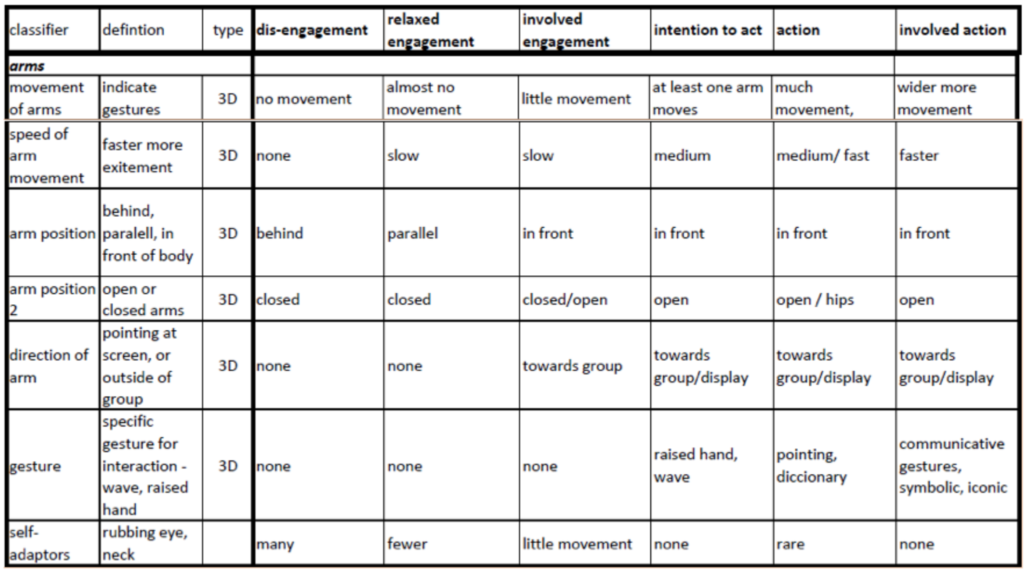
classifier	defintion	type	dis-engagement	relaxed engagement	involved engagement	intention to act	action	involved action
face								
facial expression	exitement, agreement	RGB	blank, dreaming	medium amount of facial expression	high amount of facial expression	high amount of facial expression	high amount of facial expression	intense expression
mouth	expressive	RGB	yawning	resting	little movement	open	open/ movement	open, fast movement
gaze	direction of attention	RGB	outside of group/display, to the ceiling, towards hands	mostly at group, ocassional swaying	on group/display	at display/person in group who is currently	directly at group members/display	moving through the group
eyes	blinking rate indicate tirednedd	RGB	slow blink, closed eyes	medium blinking rate	slightly faster blink	slightly faster blink	slightly faster blink	slower blinking

2.1.7 Engagement Classification Approaches

Mota used both neural networks for posture detection and Hidden Markov Models to detect engagement at an overall accuracy of 77% which needs an expensive computation[70]. Michalowski et al. offered a spatial model combined with gaze tracking to detect user engagement with a robot receptionist [28].

Schwarz et al. (2014[19]) used a linear regression approach to calculate weight factors to evaluate the relative importance of five binary classifiers and defined a threshold for the sum of the classifiers to evaluate intention to act as it is shown in Figure 2.3.

Table 2.4: Arms Classifiers



classifier	defintion	type	**dis-engagement**	**relaxed engagement**	**involved engagement**	**intention to act**	**action**	**involved action**
arms								
movement of arms	indicate gestures	3D	no movement	almost no movement	little movement	at least one arm moves	much movement,	wider more movement
speed of arm movement	faster more exitement	3D	none	slow	slow	medium	medium/ fast	faster
arm position	behind, paralell, in front of body	3D	behind	parallel	in front	in front	in front	in front
arm position 2	open or closed arms	3D	closed	closed	closed/open	open	open / hips	open
direction of arm	pointing at screen, or outside of group	3D	none	none	towards group	towards group/display	towards group/display	towards group/display
gesture	specific gesture for interaction - wave, raised hand	3D	none	none	none	raised hand, wave	pointing, diccionary	communicative gestures, symbolic, iconic
self-adaptors	rubbing eye, neck		many	fewer	little movement	none	rare	none

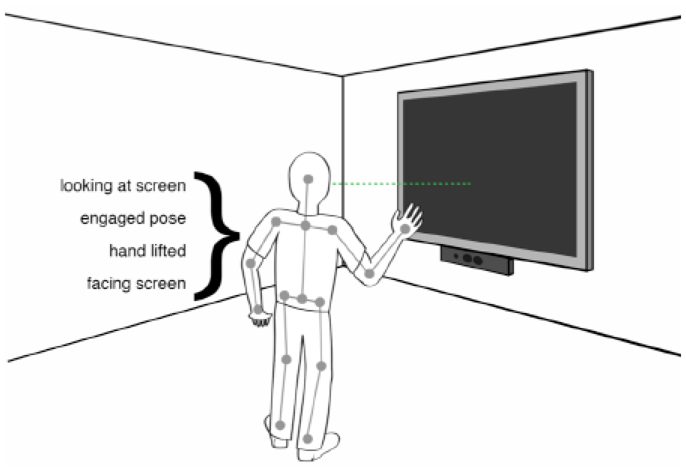
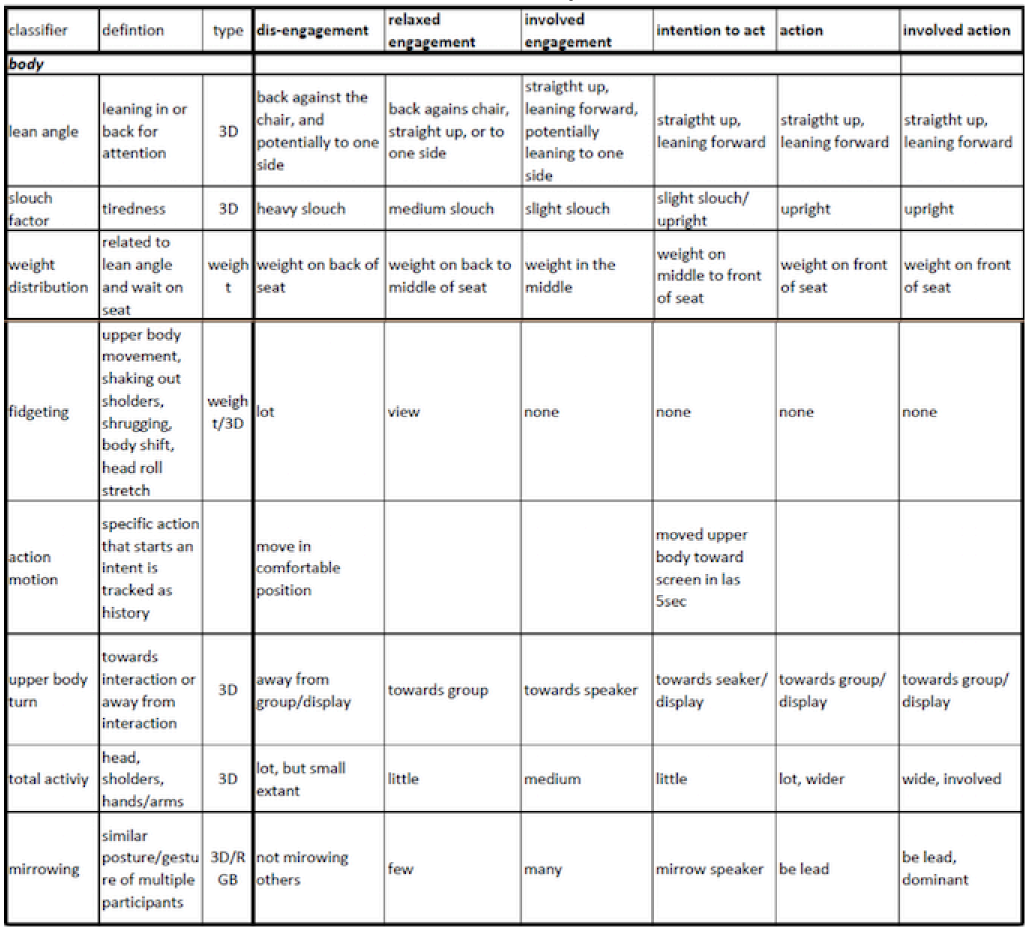


Figure 2.3: Schwarz et al. use collection of individual classifiers to determine a users intention to interact with a vision-based interface.

Table 2.7 shows test set accuracy of binary classifiers used in Schwarz et al. algorithm , and weights assigned to each classifier in their final computation.

After weight calculations, Schwarz et al. have applied thresholds on intention to interact score for intention to interact detection. In this approach, users transitioned from not engaged to engaged when their intention to interact scores reached a threshold. Figure 2.4 illustrates the trade-off between

Table 2.5: Body Classifiers



classifier	defintion	type	dis-engagement	relaxed engagement	involved engagement	intention to act	action	involved action
body								
lean angle	leaning in or back for attention	3D	back against the chair, and potentially to one side	back agains chair, straight up, or to one side	straigtht up, leaning forward, potentially leaning to one side	straigtht up, leaning forward	straigtht up, leaning forward	straigtht up, leaning forward
slouch factor	tiredness	3D	heavy slouch	medium slouch	slight slouch	slight slouch/ upright	upright	upright
weight distribution	related to lean angle and wait on seat	weigh t	weight on back of seat	weight on back to middle of seat	weight in the middle	weight on middle to front of seat	weight on front of seat	weight on front of seat
fidgeting	upper body movement, shaking out sholders, shrugging, body shift, head roll stretch	weigh t/3D	lot	view	none	none	none	none
action motion	specific action that starts an intent is tracked as history		move in comfortable position			moved upper body toward screen in las 5sec		
upper body turn	towards interaction or away from interaction	3D	away from group/display	towards group	towards speaker	towards seaker/ display	towards group/ display	towards group/ display
total activiy	head, sholders, hands/arms	3D	lot, but small extant	little	medium	little	lot, wider	wide, involved
mirrowing	similar posture/gestu re of multiple participants	3D/R GB	not mirowing others	few	many	mirrow speaker	be lead	be lead, dominant

accuracy, false positives, and false negatives at different threshold levels.

Table 2.6: Voice Classifiers

classifier	defintion	type	dis-engagement	relaxed engagement	involved engagement	intention to act	action	involved action
voice								
voice command	direct interaction control	sound	none	none	none	start interaction? I want to speak	to direct action	to direct action
speech	vocal expressions	sound	none	infrequent or no utterances	frequent utterances	humming, no uternace	speech	raised

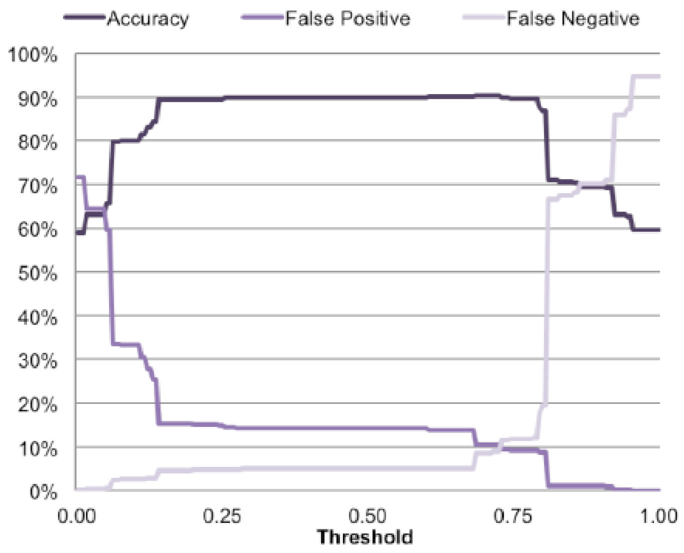
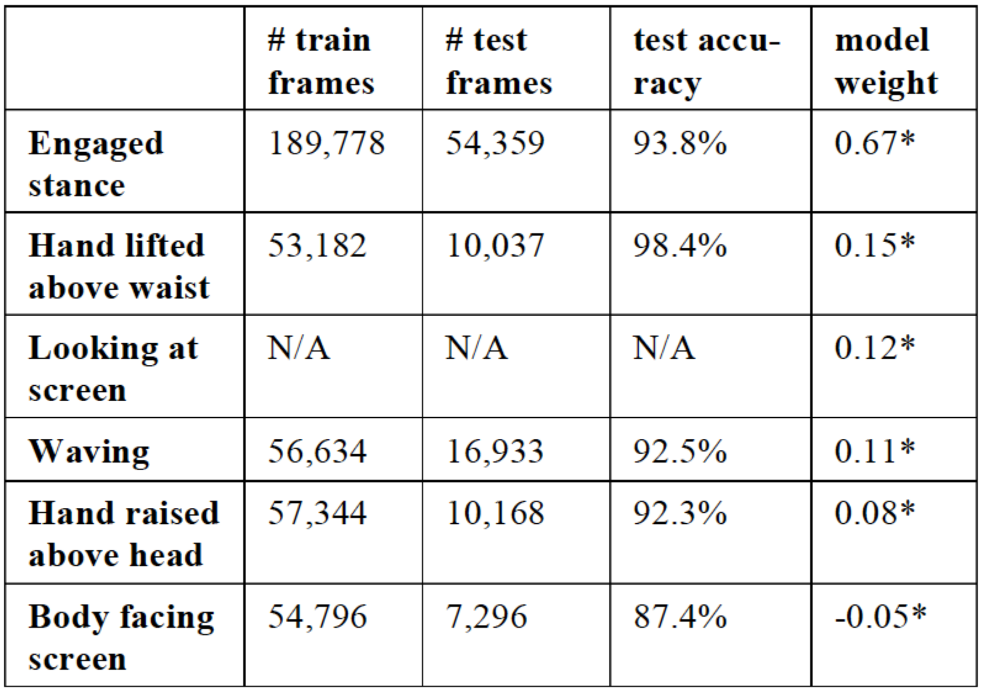


Figure 2.4: Trade-off between accuracy, false positives, and false negatives at different threshold levels.

2.2 Hand Posture Classification

Extensive reviews of hand posture and gesture recognition have been published during the last two decades. Before RGB-D cameras were developed and depth images became available, most of the research was focused on developing algorithms based on RGB images. Pavlovic et al. [83] reviewed more than 100 papers related to visual interpretation of hand gestures in context of HCI. The methods used for modeling, analyzing and recognizing gestures were discussed and integration of hand gestures with gaze, speech and other naturally related modes of communication in multi-modal interfaces were suggested to address the limitations of gestural HCI. Moeslund and Granum [84] published a comprehensive review of 130 papers discussing initialization, tracking, pose estimation and recognition of a motion capture system. Performance characteristics related to system functionality and modern advancements in each

Table 2.7: Test set accuracy of binary classifiers used in Schwarz et al. algorithm , and weights assigned to each classifier in their final computation.



	# train frames	**# test frames**	**test accuracy**	**model weight**
Engaged stance	189,778	54,359	93.8%	0.67*
Hand lifted above waist	53,182	10,037	98.4%	0.15*
Looking at screen	N/A	N/A	N/A	0.12*
Waving	56,634	16,933	92.5%	0.11*
Hand raised above head	57,344	10,168	92.3%	0.08*
Body facing screen	54,796	7,296	87.4%	-0.05*

of these fields were comprehensively evaluated. Moeslund and Granum found predominant issues such as lack of training data, the long time required for gesture capture, and lack of invariance and robustness and introduced possible solutions such as employing an approach similar to speech recognition and abstracting the motion layer that needed to be investigated in detail. Derpanis[85] reviewed the vision-based hand gestures for HCI. The feature set, classification method and underlying representation of gesture set were discussed in detail. The goal of our research is to develop such techniques for new gesture representation and classification methods.

Chaudhary et al. [14] performed a comprehensive study on gesture recognition techniques, specifically focusing on hand and facial movements. In this study Hidden Markov models, particle filtering and condensation, finite-state machines, optical flow, skin color and connectionist models were discussed in detail. The authors suggested there is a need for different recognition algorithms depending on the size of dataset, the gesture performed, and the various combinations of the two. Therefore, any developed system should be both flexible and expandable in order to maximize efficiency, accuracy and understandability.

In our research we are suggesting algorithms that use a small number of training samples to create a model for hand shape that allows accurate and user-independent hand posture recognition. Wachs et al.[9] discussed using soft computing based methods such as artificial neural network, fuzzy logic, genetic algorithms in designing hand gesture recognition method. The model we are presenting employs such soft computing techniques for classification step. Rautaray et al. [86] provided an analysis of comparative surveys that use hand gestures as a natural interface with focus on the three main phases of hand gesture recognition, i.e., detection, tracking and recognition. Different applications of hand gestures for efficient interaction have been discussed under core and advanced application domains. The authors also discussed the future perspectives of vision-based hand gesture recognition for HCI.

In the last decade, most of the studies in this filed have been focused on developing algorithms based on RGB-D sensors and depth images. Suarez et al. [87] presented a literature review on the use of depth for hand tracking and gesture recognition. The survey examined 37 papers describing depth-based gesture recognition systems in terms of: (1) the hand localization and gesture classification methods developed and used, (2) the applications where gesture recognition has been tested, and (3) the effects of the low-cost Kinect and OpenNI software libraries on gesture recognition research. Figure 2.5[87] depicts the components of a video- or depth-based hand gesture recognition and pose estimation system. In our research, we are proposing new algorithms for gesture classification block that is specified with dashed outline in this figure.

Suarez et al. [87] concluded that a total of approximately 10 different methods are commonly used for hand tracking and gesture recognition based on depth-images. Two of these algorithms are used for segmentation, 3 for tracking, and 5 for classification. In the aforementioned studies, gesture classification is done using a variety of classification algorithms. These techniques are standard machine

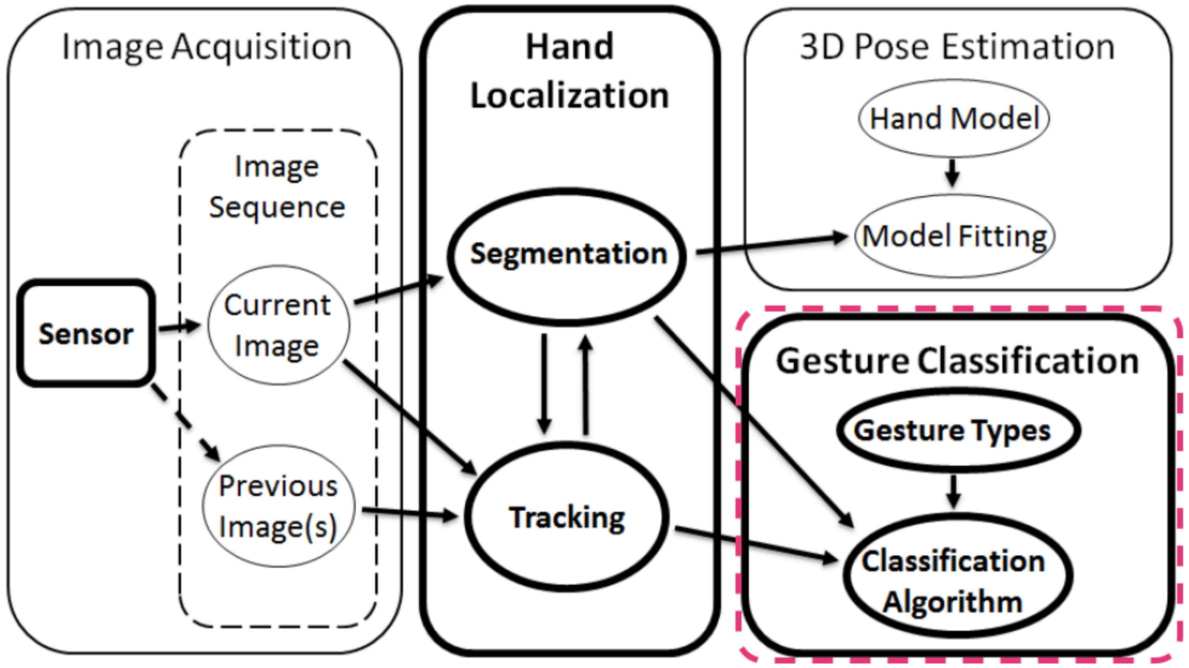


Figure 2.5: The components of a video- or depth-based hand gesture recognition and pose estimation system.

learning algorithms including Hidden Markov Models, k-NN, ANNs, SVMs, and FSMs.

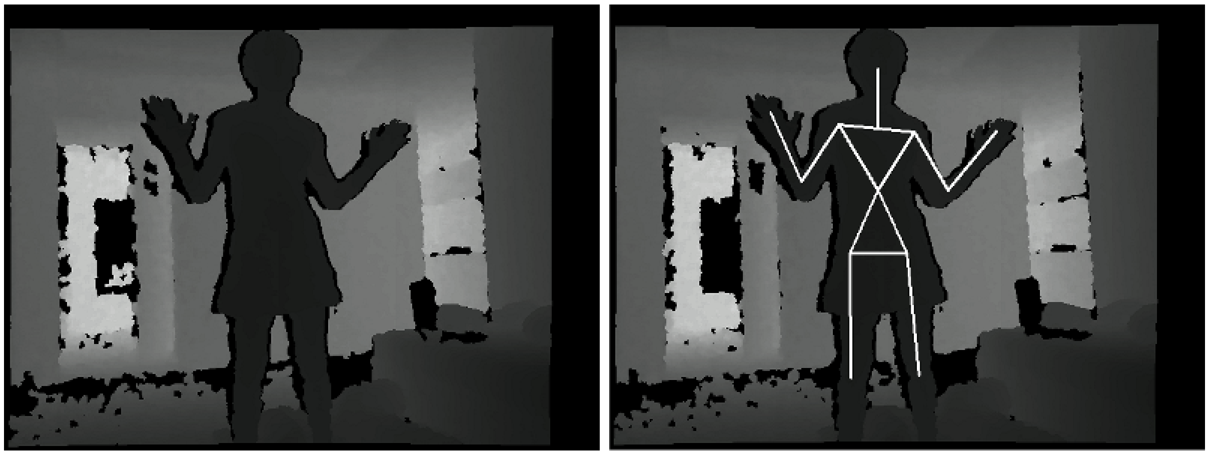


Figure 2.6: OpenNI-compliant real time skeleton detection and tracking by PrimeSense

In addition, detection and tracking methods are being replaced by off-the-shelf solutions such as PrimeSenses NiTE module for the OpenNI framework [87] [88]. OpenNI is publicly available open source

API. Currently OpenNI supports Kinect and Xtion sensors. Moreover the main partner of the OpenNI organization (PrimeSense) provides NiTE - the Natural Interaction Middleware. This middleware allows to perceive the world in 3D, and to comprehend, translate and respond to human movements without any wearable equipment or controls. NiTE provides functionality of human body detection, skeleton extraction and simple gesture recognition. A stick figure which gives the location of body parts, such as hand, in a depth-image can be detected and tracked. Figure 2.6 shows an example of body and hand detection using this module. Further- more, Figure 2.7 also shows an application developed by Evoluce using these features for a multi-touch experience. Based on these recent developments, there seems to be a higher demand for developing powerful hand shape models for feature extraction rather than hand detection or segmentation. Furthermore, there is a greater need for optimized classification algorithms. Therefore, in the present research we focus on developing such algorithms rather than dealing with hand pose estimation, hand tracking or background subtraction.

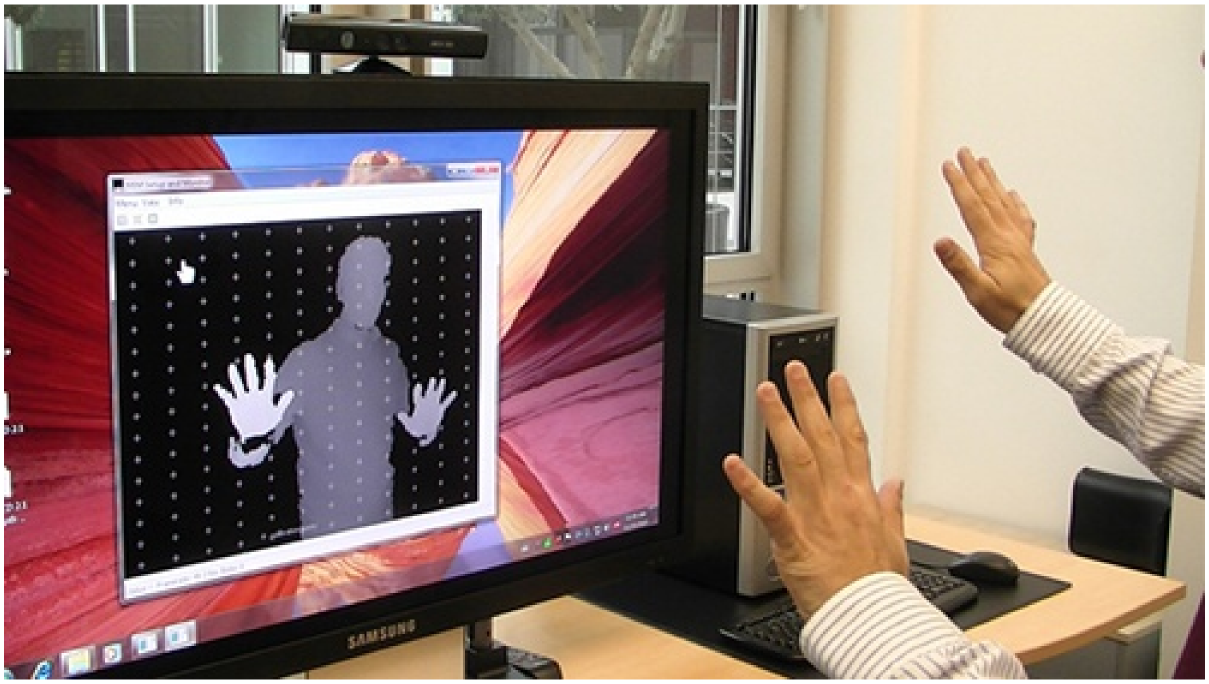


Figure 2.7: Using Kinect for Hand Gesture Recognition

Marnik [89] uses the curvature of a hand edge to create a feature vector describing a model for hand shape. Boundary points which correspond to boundary parts with specified curvature are used to create a feature vector describing a hand shape. Feature vectors corresponding to shapes which are supposed to be recognized by the system are recorded in a model set and serve as patterns in a recognition phase. At the recognition stage, a similarity measure of the model image to the examined image is used to calculate a similarity value of the two feature vectors. Since the number of high curvature points can be different for different images, a special measure is proposed for the method. Figure 2.8 shows the hand shapes used by Marnik [89].

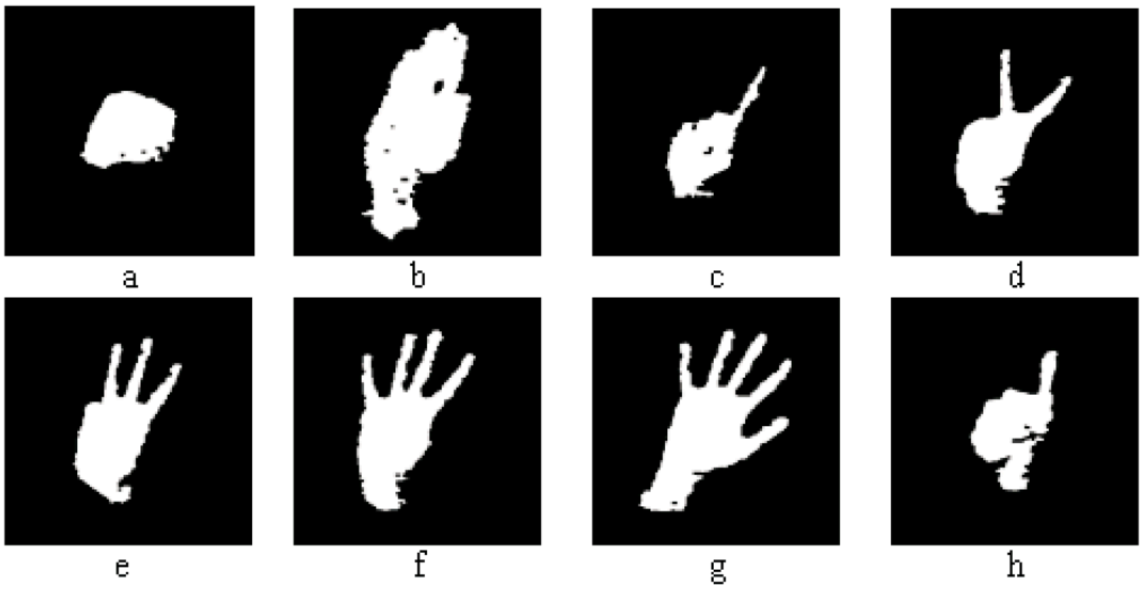


Figure 2.8: Hand shapes used in Marnik's experiments: a) wrist, b) palm, c - g) 1 - 5 fingers, h) thumb

Kelly et al. [90] proposed a hand posture feature based on an eigenspace Size Function. Our presentation of hand shape also uses eigenspace for feature reduction.

Bourennane and Fossati[26] reviewed attempts for modeling the hand shape based on shape descriptors for hand postures. These models consists of Fourier descriptors, Hu and Zernike moments. The authors extensively examined the performance of hand posture recognition on two families of contour-based Fourier descriptors and two sets of region-based moments for modeling the hand shapes. To study the performances of these shape descriptors, they first used Triesch hand posture dataset[33] which is publicly available. Triesch hand posture dataset consists of 10 hand signs performed by 24 people against three backgrounds. Bourennane and Fossati[26] used this dataset with a uniform background in order to compare the shape descriptors without depending on segmentation. Thus, they used 479 images with black and white backgrounds (one image for v gesture was missing).Figure 2.9 shows Triesch hand posture database for some alphabets of ASL. The images are 128×128 pixels in gray scale. While there was very limited variability in the sizes of the hands and the orientation of hand postures, the shapes of a same posture from different users could be very different. Bourennane and Fossati[26] also developed an internal dataset and tested all of the methods on this newly developed dataset . However, this dataset has not been published publicly for further examination or comparison. Figure 2.10[26] depicts 11 different postures introduced in the internal dataset. Furthermore, Trigueiros et al.[27] present a comparative study of seven different algorithms for hand feature extraction, for static hand gesture classification to find the best learner. They also defined their own gesture vocabulary, with 10 gestures, and recorded videos from 20 people performing the gestures for later processing.

In all of the aforementioned studies, the number of public hand posture shapes with uniform background is lower than the desired number for a good comparison. Furthermore, in each study the authors have examined hand gestures used for creating a different set of letters.

Since different gestures are used to create different letters with some of them having obvious discriminative features that makes them easier to classify, it is difficult to compare the results of these studies. Therefore, we introduce a hand posture dataset [91] which is publicly available for further reference and future research. This dataset extends the number of hand postures in the dataset proposed by Triesch [33].

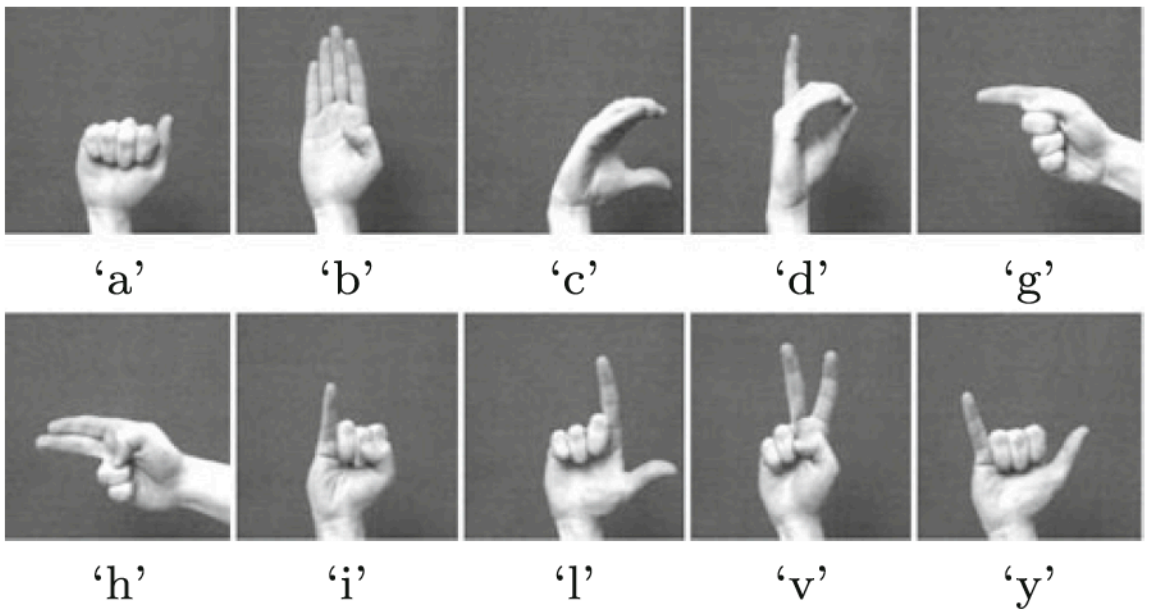


Figure 2.9: Triesch hand posture database from some alphabets of ASL

2.2.1 Hand Shape Modeling

In this section, computation of several shape descriptors, the theory behind them, and the classification steps used in [26] are presented.

2.2.2 Fourier Descriptors

Fourier descriptors are extensively used for the description and also classification of shapes with a closed contour. FD are so useful in $2D$ modeling because they represent the shapes correctly, and have invariance properties[92, 93]. These descriptors are contour-based shape descriptors. Here, We consider the case of closed planar curves under the action of Euclidean transformations. If $\gamma_2(l)$ and $\gamma_2(l)$ denote the respective arclength parametrization of two closed contour objects with, the same shape but and different poses, we can write:

$$\gamma_2(l) = ae^j\theta\gamma_1(l + l_0) + b \tag{2.1}$$

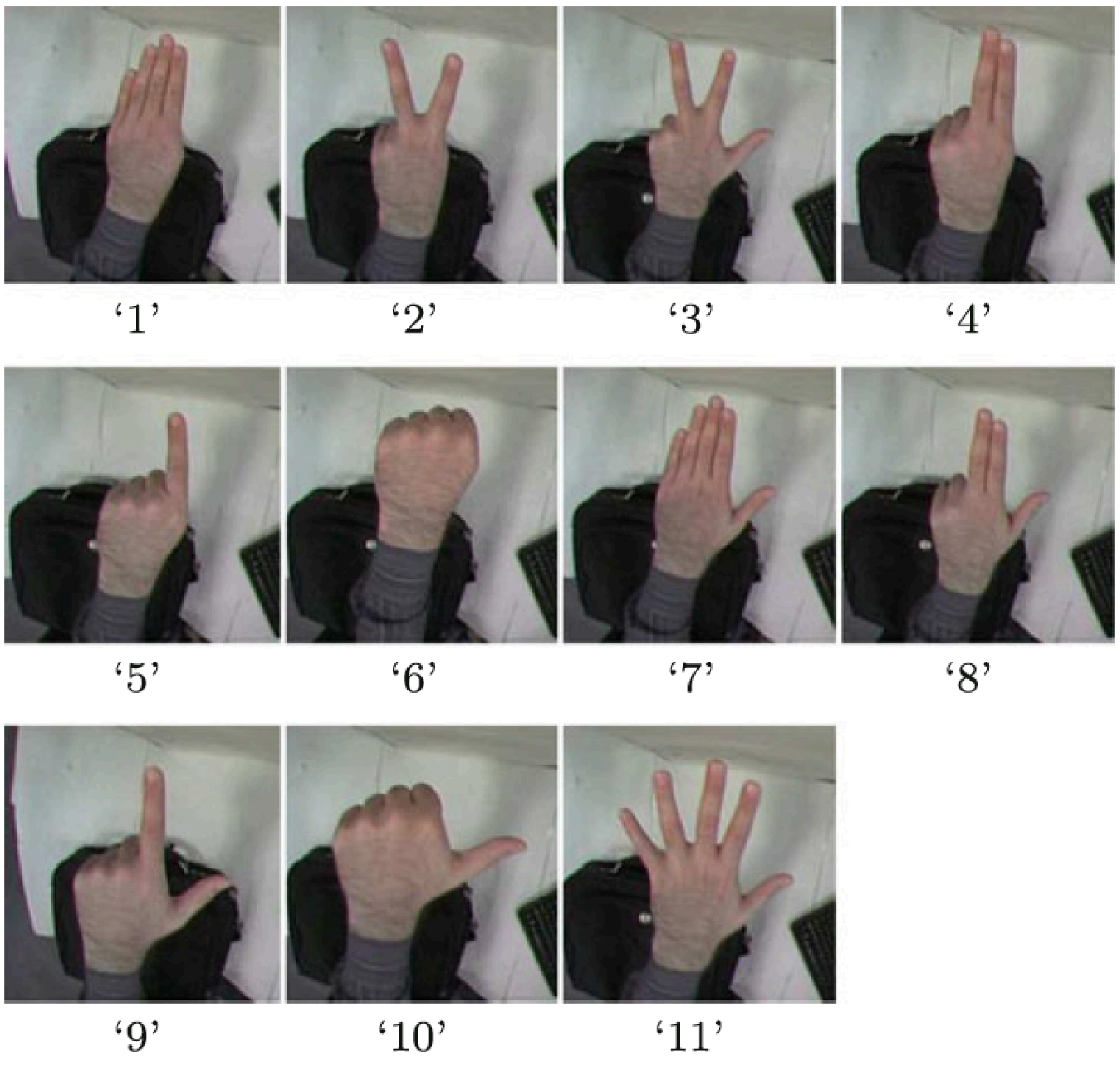


Figure 2.10: The 11 postures of internal database (cropped images)

with a the scale factor, θ the rotation angle, b the translation and l_0 the starting description points difference, $l_0 \in [0, L]$ with L the length of the contour. Scale invariance is achieved by normalizing the arc-length parametrization with an equal length of 1 which leads to $l_0 \in [0, 1]$. The translation invariance is given by describing the contours according to their center of mass[26]. Before applying Fourier transform to the shape signature, shape is first sampled to fixed number of points. In general, object shape and model shape can have different sizes. Consequently, the number of data points of the object and model representations will also be different. The shape boundary of the objects and models must be sampled to have the same number of data points. The number of resolution levels at which the shape signature will be decomposed is determined by the length of the shape boundary. By varying the number of sampled points, the accuracy of the shape representation can be adjusted. The larger the number of sampled points, the more details in the representation of the shape; consequently, the matching result will be more accurate. In contrast,fewer sampled points reduce the accuracy of the matching results while improving the computational efficiency. There are generally three methods of normalization (i) equal points sampling; (ii) equal angle sampling;and (iii) equal arc-length sampling[26]. If N is the total number of candidate points to be sampled along the shape boundary, the equal points sampling method selects candidate points spaced at equal number of points along the shape boundary. The distance between two consecutive candidate points is given by P/N, where P is the total number of boundary points. The equal arc-length sampling method selects candidate points spaced at equal arc length along the shape boundary. The distance between two consecutive candidate points is given by L/N, where L is the perimeter of the shape boundary. The equal angle sampling selects candidate points spaced at equal angle $\theta = 2\pi/N$. The equal arc-length sampling method gives the best equal space effect Among the three sampling methods among the three sampling methods.

In this problem, the complex coordinates is employed. Each point of the shape contour is represented by a complex number $z_i = x_i + jy_i, i \in [1, N]$, with N being the number of points of the contour. This number must be chosen as a compromise between a reliable description of the shape, with enough details and shape smoothing which eliminates the finest details mostly regarding the noise. Therefore, the equal arc-length sampling is chosen to normalize the size of the shapes. For each shape, 64 candidate points with equal arc-length space between them are selected. The computation time is another important factor. The computation time increases with the higher number of points. For computational efficiency of the fast Fourier transform, the number of points is chosen to be a power of two. Therefore, the Fourier transform gives N Fourier coefficients C_k:

$$C_k(\gamma) = \sum_{i=0}^{N-1} z_i e^{-j\frac{2\pi ik}{N}},\ k = 0, \dots,\ N-1 \tag{2.2}$$

Equation in the frequency domain gives:

$$C_k(\gamma_2) = e^{j\theta} e^{j\frac{2\pi k l_0}{N}} C_k(\gamma_1) + b\delta_k \tag{2.3}$$

where δ_k is the Kronecker delta. The first coefficient, $C_0(\gamma_2)$, contains the shape position. That is no longer taken into account. Hence, rotation of the shape and difference in the starting point affect only the phase information. So equation 2.3, can be written as follows:

$$C_k(\gamma_2) = e^{j\theta} e^{j\frac{2\pi k l_0}{N}} C_k(\gamma_1),\ k = 1, \dots,\ N-1 \tag{2.4}$$

Figure hyperref[FD-cutoff]2.11[87] shows that the low-frequency coefficients contain information on the general aspect of the shape, whereas the high-frequency coefficients contain the finer details of the shape. On the Triesch database, we can notice that with more than 20 coefficients, the hand shape is well reconstructed.

Now we can define two different sets of invariants which are $FD1$ and $FD2$. In case of $FD1$, FD invariants to similarities is to take the magnitude of Fourier coefficients and obtain the $N-2$ $FD1$ coefficients in equation

$$I_k(\gamma) = \frac{|C_k(\gamma)|}{|C_1(\gamma)|}, k = 2, \dots, N-1 \tag{2.5}$$

However, this set of invariants is not complete as it does not hold the phase information of the shape. In case of $FD2$, a complete and stable set of invariants is obtained. The completeness of a set of invariant features expresses the fact that two objects have the same shape if and only if they have the same set of

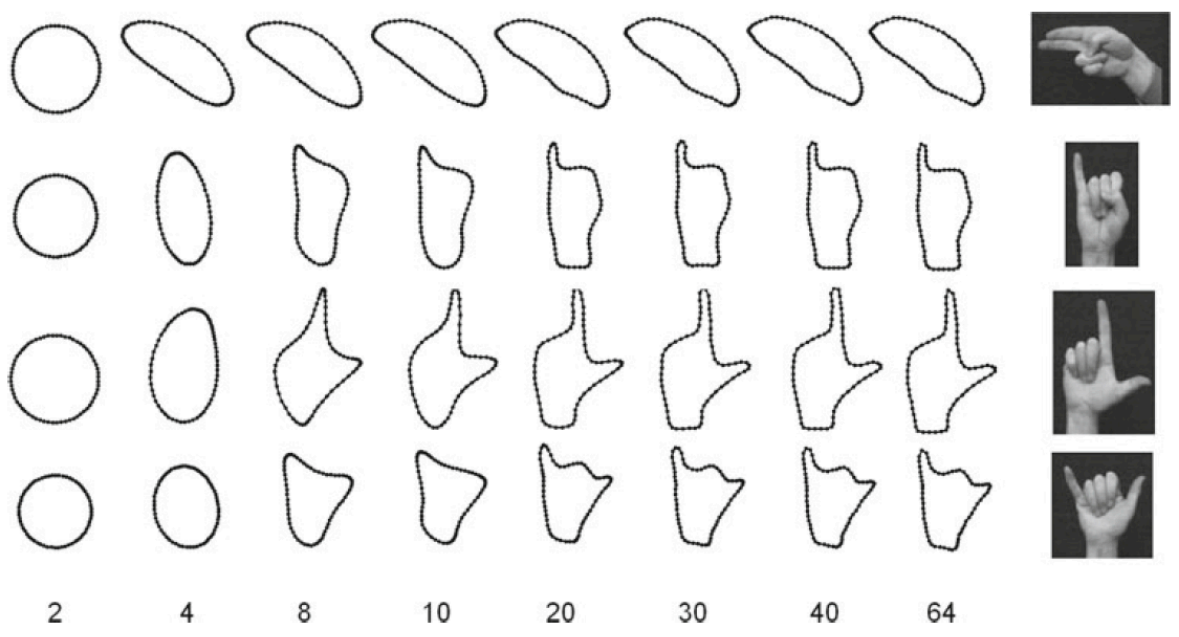


Figure 2.11: Examples of reconstruction as a function of the cutoff frequency, with an initial contour sampled with $N = 64$ points

features. A set of features that is complete but not stable is proposed in Stability means that a small distortion of the shape does not induce a noticeable divergence in the values of invariant features. The complete and stable set of invariant descriptors is defined by:

$$I_{k_0}(\gamma) = |C_{k_0}(\gamma)|,\ for\ k_0\ such\ that \tag{2.6}$$

$$C_{k_0}(\gamma) \neq 0, \tag{2.7}$$

$$I_{k_1}(\gamma) = |C_{k_1}(\gamma)|,\ for\ k_1 \neq k_0\ such\ that \tag{2.8}$$

$$C_{k_1}(\gamma) \neq 0, \tag{2.9}$$

$$I_k(\gamma) = \frac{C_k(\gamma)^{k_0-k_1} C_{k_0}(\gamma)^{k_1-k} C_{k_1}(\gamma)^{k-k_0}}{I_{k_0}(\gamma)^{k_1-k-p} I_{k_1}(\gamma)^{k-k_0-q}}, \tag{2.10}$$

$$\forall k \neq k_0, k_1, \tag{2.11}$$

with $p, q \in \mathbb{R}^+$ and $k1 \leq k0$. For experiments, in order to simplify the expression of $I_k(\gamma)$, following

[94], we take $k_0 = 2, k_1 = 1, p = q = 0.5$. Notice that the cepstral descriptors can be investigated as used in speech recognition front-ends to enhance the robustness.

2.2.3 Hu and Zernike Moments

Hu and Zernike Moments are region-based descriptors. *Hu* invariants are used in pattern recognition to provide a scale-, orientation- and position-invariant representation of an objects shape. These moments are computed with the geometrical moments of the hand region. The zero-mean moments are invariant to translations. For scale invariance descriptor , the normalized moments are calculated; $\eta_{pq} : \eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}$ with $\gamma = \frac{p+q}{2} + 1, \forall p + q \geq 2$. Using the first three normalized moments, we can calculate the seven *Hu invariant moments*[95]:

$$I_1 = \eta_{20} + \eta_{02} \tag{2.12}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{2.13}$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{2.14}$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{2.15}$$

$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \tag{2.16}$$

$$+ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{2.17}$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{2.18}$$

$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \tag{2.19}$$

$$- (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{2.20}$$

The first six invariant moments characterize the shape with invariance to translation, rotation and scale changes. The seventh invariant moment allows to distinguishing the symmetrical shapes. Complex Zernike moments are built using a set of complex polynomials that span a complete orthogonal basis defined on the unit disk $(x^2 + y^2) \leq 1$. They are presented as:

$$A_{mn} = \frac{m+1}{\pi} \int_x \int_y f(x, y) \left[V_{mn}(x, y)\right]^* dxdy \tag{2.21}$$

where $m = 0, 1, 2, ..., \infty$ defines the order, and $f(x, y)$ is the function being described. Coefficient n is an integer depicting the angular dependence, subject to the conditions $m - |n| = even$ and $|n| \leq m$. The Zernike polynomials are

$$V_{mn}(r,\ \theta) = R_{mn}(r)e^{jn\theta} \tag{2.22}$$

where $(r,\ \theta)$ are defined over the unit disk and $R_{mn}(r)$ is obtained by the following equation:

$$R_{mn}(r) = \sum_{s=0}^{\frac{m-|n|}{2}} (-1)^s \frac{(m-s)!}{s!(\frac{m+|n|}{2} - s)!(\frac{m-|n|}{2} - s)!} r^{m-2s} \tag{2.23}$$

Magnitudes of Zernike moments are invariant to rotation. To obtain translation and scale invariance, images are normalized using the zeroth and first order moments. Therefore, for recognition, $|A_{00}|$ and $|A_{11}|$ are not considered. Taking the moments from order 2 to 15 is sufficient [96], leading to 70 moments. One major drawback of Zernike moments is their computation complexity which is very time consuming. Different approaches have been proposed to speed up the computation [97].

2.2.4 Classification

In most cases, classification is achieved with a distance metric and nearest neighbor rules[98, 99, 100]. Furthermore, classifiers such as radial basis function are used for classification. FD have also been employed as input features for dynamic gesture recognition with Hidden Markov Models[101, 102] and Neural Networks[101, 103]. The complete and stable set of invariant features induces a distance on the

Table 2.8: Recognition rates per gesture with internal database and Euclidean distance, on the testing set [26]

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	Total
Hu	79.2	60.28	64.52	60.04	90.8	100	45.29	62.42	76.78	62.35	40.57	67.5
Zer.	74.9	61.7	64.2	64.8	88.4	96.5	52.7	62.2	76.2	68.3	39.2	68.1
FD1	92.8	75.3	78.4	92.4	93.8	94.5	89.1	70.7	85.5	75.9	76.2	83.9
FD2	76.3	68.4	70.8	72.4	86	85.1	56.5	66.4	81.5	61.6	48.5	70.3

shape space, which is given by the Euclidean distance:

$$d_2(\gamma_1, \gamma_2) = \sqrt{\sum_{k=0}^{N-1} \left| I_k(\gamma_1) - I_k(\gamma_2) \right|^2} \tag{2.24}$$

It is used as an error measure, while not theoretically valid for $FD1$.Because the number of images in the Triesch database is small, Bourennane and Fossati[26] performed a k-fold cross-validation to estimate the recognition rates. For Triesch database, they performed a a leave-one-out cross-validation, in which one image for the validation and the rest for the training. For internal database, they also performed a cross-validation with sub-samples containing 50 images.

Table 2.8 shows the recognition rates for each gesture on the test set. Based on this table, the $FD1$ outperforms the other shape descriptors when discriminating between visually close gestures. Either moment invariants or Fourier coefficients are computed from the segmented hand posture. When the postures lead to similar segmentation results, some details of the hand contour are smoothed, and both moment invariants and Fourier coefficients are affected. Therefore, the recognition rate is reduced.

These results show that the best recognition rates with Euclidean distance are obtained by $FD1$. Bourennane and Fossati[26] also proposed evaluation of different classifiers with $FD1$ in order to see whether they give better results than Euclidean distance. The classifiers tested on their internal dataset are presented in Table 2.9.

Based on the results presented in Table 2.9, for internal database, the recognition rates are similar for

Table 2.9: Recognition rates (%) with $FD1$ and different classifiers: Bayesian classifier (BAYES), support vector machine (SVM), k-nearest neighbors (k-NN) and Euclidean distance (EUCL)

	BAYES	*SVM*	*k-NN*	*EUCL*
Internal database, test set	84.7	84.2	87.9	83.9

the three classifiers with a small advantage for k-NN. Furthermore, Ronald and Mannes[100] provided a comparison on human posture recognition among FD with complex coordinates, shape context histograms, and Hu moments. The authors perform tests with deformed shapes to determine the robustness of each descriptor separately, and showed that FD outperform Hu moments.

2.3 Summary

In this chapter, a comprehensive review for engagement modeling and detection is presented. Engagement theories and modeling for different applications and approaches is reviewed and several literatures that offers observable engagement feature extraction from human body is discussed. Because we mainly are interested to the use of engagement detection for hand posture and gesture recognition, those previous research which gives some clues for extracting meaningful features based on body postures and gestures are highlighted. Afterwards, different classification approaches for engagement detection is provided.

In addition, the background research for hand posture detection is discussed in details. First we argued hand shape modeling based on several welknown approaches and afterwards classification approaches are reviewed.

Chapter 3

Theory

3.1 Dimension Reduction Using PCA

The problem with the image representation we are given is its high dimensionality. Two-dimensional $p \times q$ gray-scale images span an $m = pq$-dimensional vector space. Therefore, an image in our dataset with 128×128 pixels falls in a $16,384$-dimensional image space already. The question is: Are all dimensions equally useful for us? We can only make a decision if theres any variance in data, so what we are looking for are the components that account for most of the information. The PCA is proposed to turn a set of possibly correlated variables into a smaller set of uncorrelated variables. The idea is that a high-dimensional dataset is often described by correlated variables, and therefore, only a few meaningful dimensions account for most of the information. The PCA method finds the directions with the greatest variance in the data, called principal components.

3.1.1 Algorithmic Description

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a random vector with observations $x_i \in R^d$.

1. Compute the mean μ

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3.1}$$

2. Compute the the Covariance Matrix S

$$S = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T \tag{3.2}$$

3. Compute the eigenvalues λ_i and eigenvectors :v_i of S

$$S v_i = \lambda_i v_i, i = 1, 2, \ldots, n \tag{3.3}$$

4. Order the eigenvectors descending by their eigenvalue. The k principal components are the

eigenvectors corresponding to the k largest eigenvalues.

The k principal components of the observed vector x are then given by:

$$y = W^T(x - \mu) \tag{3.4}$$

where $W = (v_1, v_2, \ldots, v_k)$.

The reconstruction from the PCA basis is given by:

$$x = Wy + \mu \tag{3.5}$$

where $W = (v_1, v_2, \ldots, v_k)$.

Inspired from Eigenfaces method[104], we can

- Project all training samples into the PCA subspace.
- Project the query image into the PCA subspace.
- Find the nearest neighbor between the projected training images and the projected query image. For finding the nearest neighbor, different types of classifiers, such as Neural Networks, SVM, k-NN, could be employed.

One more problem still left to solve. Imagine we are given 400 images sized100 $\times$ 100 pixel. The Principal Component Analysis solves the covariance matrix $S = XX^T$, where $size(X) = 10000 \times 400$ in our example. We would end up with a 10000 $\times$ 10000 matrix, roughly 0.8GB. Solving this problem is not feasible. From linear algebra we know that when M is greater than N, an $M \times N$ matrix can only have $N - 1$ non-zero eigenvalues. So its possible to take the eigenvalue decomposition $S = X^T X$ of size $N \times N$ instead:

$$X^T X v_i = \lambda_i vi \tag{3.6}$$

and get the original eigenvectors of $S = XX^T$ with a multiplication of the data matrix:

$$XX^T(Xv_i) = \lambda_i(Xv_i) \tag{3.7}$$

The resulting eigenvectors are orthogonal, and need to be normalized to unit length to get orthonormal eigenvectors. More comprehensive explanations and examples are discussed in [105]. The derivation and proof of the equations are also presented in [106].

3.2 Support Vector Machine (SVM)

A Support Vector Machine (SVM) [107, 108] is a discriminative classifier formally defined by a separating hyperplane. In other words, given some labeled training data in a supervised learning task, the algorithm outputs an optimal hyperplane which categorizes new examples. For finding out in which sense the hyperplane obtained is optimal, Let's consider the following simple problem. For a linearly separable set of 2D points which belong to one of two classes (Figure 3.1), find a separating straight line.

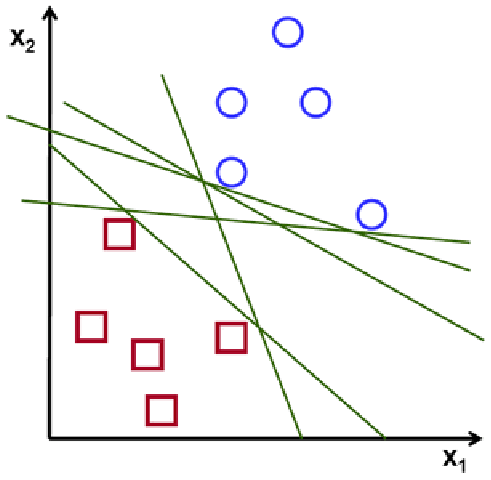


Figure 3.1: A separation example

In this example we deal with lines and points in the Cartesian plane instead of hyperplanes, and vectors in a high dimensional space. This is a simplification of the problem. It is important to understand that this is done only because our intuition is better built from examples that are easy to imagine. However, the same concepts apply to tasks where the examples to classify lie in a space whose dimension is higher than two. In the above figure, there are multiple lines that offer a solution to the problem. Is any of the lines better than the others? We can intuitively define a criterion to estimate the value of each line. A line is bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, our goal should be to find the line passing as far away from all points as possible. Thereforen, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of mar- gin within SVMs theory. Therefore, the optimal separating hyperplane *maximizes* the *margin* of the training data (Figure 3.2).

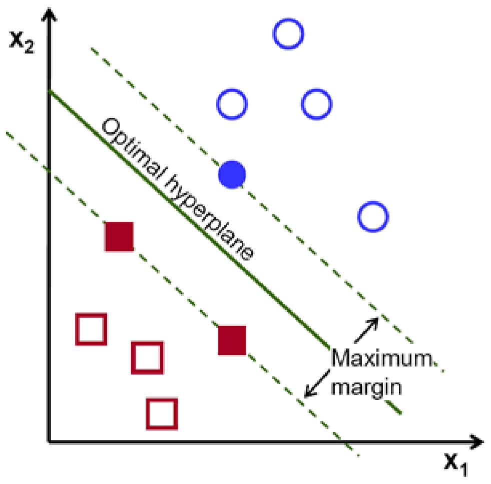


Figure 3.2: The Optimal hyperplane

3.2.1 Computing Optimal Hyperplane

The notation used to define formally a hyperplane can be introduced by:

$$f(x) = \beta_0 + \beta^T x \tag{3.8}$$

where β is known as the **weight vector** and β_0 as the **bias**.

A more in depth description of hyperplanes could be find in section 4.5, Separating Hyperplanes, of The Elements of Statistical Learning[109].

The optimal hyperplane can be represented in an infinite number of different ways by scaling β and β_0. From all possible representations of hyperplane, we chose the following

$$|\beta_0 + \beta^T x| = 1 \tag{3.9}$$

where x symbolizes the training examples closest to the hyperplane. In general, the training examples which are closest to the hyperplane are called **support vectors**. This representation is known as the **canonical hyperplane**.

Now, we use distance between a point x and a hyperplane (β, β_0) using the following equation:

$$\text{distance} = \frac{|\beta_0 + \beta^T x|}{||\beta||} \tag{3.10}$$

In particular, for the canonical hyperplane, the numerator is equal to one and the distance to the support vectors is

$$\text{distance}_{\text{ support vectors}} = \frac{|\beta_0 + \beta^T x|}{||\beta||} = \frac{1}{||\beta||} \tag{3.11}$$

The margin introduced in the previous section, here denoted as M, is twice the distance to the closest examples:

$$M = \frac{2}{||\beta||} \tag{3.12}$$

Finally, the problem of maximizing M is equivalent to the problem of minimizing a function $L(\beta)$ subject to some constraints. The constraints model the requirement for the hyperplane to correctly classify all the training examples x_i. Formally:

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}||\beta||^2 \text{ subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \ \forall i \tag{3.13}$$

where y_i represents each of the labels of the training examples.

This is *Lagrangian optimization* problem that can be solved using Lagrange multipliers to obtain the weight vector β and the bias β_0 of the optimal hyperplane.

3.2.2 Nonlinear Classification and Kernel Functions

Bernhard E. et al.[107] suggested a way to create nonlinear classifiers by applying the kernel method (originally proposed by Aizerman et al.[110]) to maximum-margin hyperplanes[107]. The resulting algorithm is similar to the origin one, except that in newly developed algorithm every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional. Therefore, although the classifier is a hyperplane in the high-dimensional feature space, it may be nonlinear in the original input space. Figure 3.3 shows a kernel function used to transform a non-linearly separable functions into a higher dimension linearly separable function.

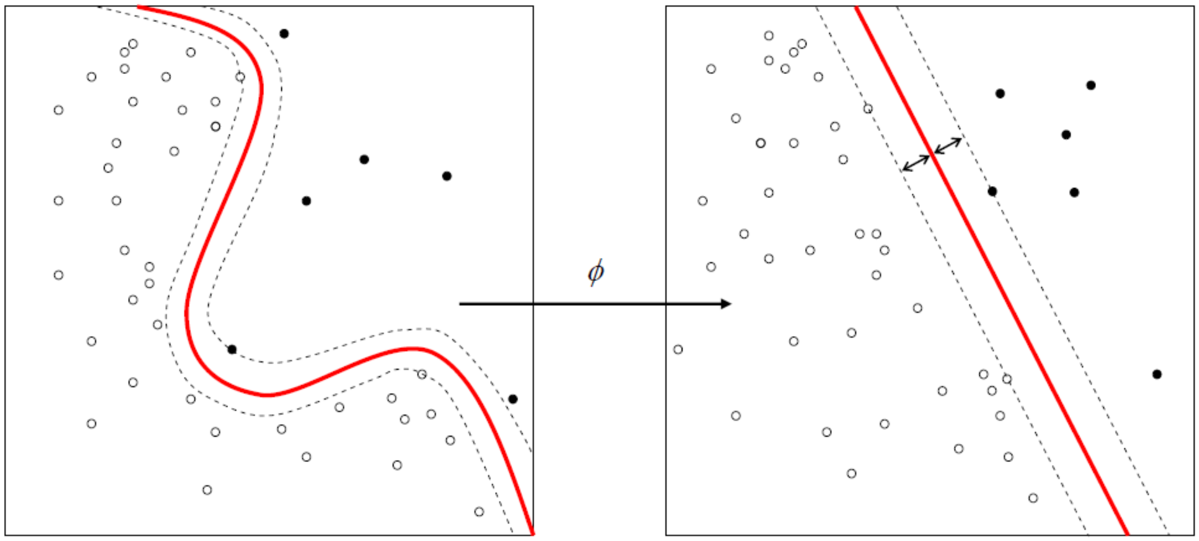


Figure 3.3: Kernel functions are used to transform non-linearly separable functions into a higher dimension linearly separable functions.

Kernel machines are used to convert non-linearly separable functions into a higher dimension linearly separable functions.

A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one *target value* (i.e., the class label), and several *attributes* (i.e., the features or observed variables). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data using only its attributes[111].

Given a training set of instance-label pairs $(\mathrm{x}_i,\ y_i), i = 1, \dots, l$ where $\mathrm{x}_i \in R^n$ and $\mathrm{y} \in \{1,\ -1\}^l$, the support vector machines require the solution to the following optimization problem:

$$\begin{aligned} \min_{\mathrm{w,b},\xi} L(\beta) &= \frac{1}{2}\mathrm{w}^T\mathrm{w} + \mathrm{C}\sum_{i=1}^{l}\xi i \\ \text{subject to } y_i(\mathrm{w}^T\phi(x_i) + b) &\geq 1 - \xi_i \\ \forall\xi_i &\geq 0 \end{aligned} \tag{3.14}$$

Here training vectors x_i are mapped into a higher (maybe infinite) dimensional space by function ϕ. SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. Furthermore, $K(\mathrm{x}_\eta,\ \mathrm{x}_j) \equiv \phi(\mathrm{x}_i)^T\phi(\mathrm{x}_j)$ is called the kernel function. Though new kernels are proposed by researchers, the following four basic kernels are more common:

- Linear: $K(\mathrm{x}_i,\ \mathrm{x}_j) = \mathrm{x}_i^T\mathrm{x}_j$.
- Polynomial: $K(\mathrm{x}_i,\ \mathrm{x}_j) = (\gamma\mathrm{x}_i^T\mathrm{x}_j + r)^d, \gamma > 0$.
- Radial Basis Function (RBF): $K(\mathrm{x}_i,\ \mathrm{x}_j) = \exp(-\gamma\|\mathrm{x}_i - \mathrm{x}_j\|^2), \gamma > 0$.
- Sigmoid: $K(\mathrm{x}_i,\ \mathrm{x}_j) = \tanh(\gamma\mathrm{x}_{i^T}\mathrm{x}_j + r)$.

Here, γ, r, and d are kernel parameters.

3.2.3 Multi-class SVM

Multi-class SVM aims to assign labels to instances by using support vector machines, where the labels are drawn from a finite set of several elements.

The dominant approach for doing so is to reduce the single multi-class problem into multiple binary classification problems[112]. The most common methods for such reduction include[112, 113]:

1. Building Binary Classifiers which distinguish between
    (a) **One-Versus-All**: One of the labels and the rest
    (b) **One-Versus-One**: Between every pair of classes
2. Directed acyclic graph SVM (DAGSVM)[114]
3. Error-correcting output codes[115]

In one-versus-all method, new instances are classified using a winner-takes-all strategy in which the classifier with the highest output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). In the one-versus-one approach, classification is done by a max-wins voting strategy. In this strategy each classifier assigns the instance to one of the two classes. Subsequently, the number of votes for the assigned class is increased by one. The class with the most votes determines the instance classification.

3.2.4 Implementations of Support Vector Machines

Kernel SVMs are available in many machine learning toolkits including LIBSVM, MATLAB, SVMlight, scikit-learn, Shogun, Weka, Shark , JKernelMachines. Since the speed of algorithm is important for us, we based our approach on LIBSVM and LIBLINEAR.

LIBSVM and LIBLINEAR are two popular open source machine learning libraries, both developed at the National Taiwan University and both written in C++ though with a C API. LIBSVM implements the Sequential minimal optimization (SMO) algorithm for kernelized SVMs, supporting classification and regression[116]. LIBLINEAR implements linear SVMs and logistic regression models trained using a coordinate descent algorithm.

LIBSVM does support multi-class classification. implements the **One-Versus-One** approach (Knerr et al., 1990) for multi-class classification. If k is the number of classes, then $\frac{k(k-1)}{2}$ classifiers are constructed, each one training data from two classes.

In classification a voting strategy is employed. Each binary classification is considered to be a voting where votes can be cast for all data points. At the end, a point is designated to a class with the maximum number of votes. In cases where a point has the same number of votes for two classes , the class appearing first in the array of storing class names will be reported.

3.3 Finite State Machines (FSM)

State machines are the description of a life cycle of a system. They can describe the different states of the lifeline, the events which influence it, and what it does when a particular event is detected at a any states as the transition condition for particular state change. They offer the complete specification of the dynamic behavior of the system.

For instance, a simple mechanism that can be modeled by a state machine is a turnstile[117, 118]. A turnstile (Figure 3.4) is a a gate with three rotating arms at waist height, one across the entryway. Common use of turnstiles is controlling access to subways. In its initial state, the arms are locked and blocking the entry preventing from passing through. Depositing a coin or token in a slot unlocks the arms and allows a single patron to push and pass. After the person passes through, the arms are locked again until another coin or token is inserted.

Figure 3.4: A common turnstile for controlling access to subway

Modeling a turnstile as a state machine, the it has two states: Locked and Unlocked[118]. The initial state of the turnstile is Locked state. Two inputs affect its state:

- Putting a coin in the slot (coin)
- Pushing the arm (push)

In the locked state, pushing on the arm has no effects. Putting a coin in shifts the state from Locked

Table 3.1: Turnstile State Transition table

Current State	Input	Next State	Output
Locked	coin	Unlocked	Unlock turnstile and customer can push through
	push	Locked	None
Unlocked	coin	Unlocked	None
	push	Locked	Lock turnstile when customer has pushed through

to Unlocked. In contrast, it the unlocked state, putting additional coins in has no effect; On the other hand, inserting additional coin inputs does not change the state. However, giving a push input shifts the state back to Locked.

3.3.1 Finite State Machine Representation

Finite state machine could be presented in several ways. The most common representations are as follows:

- State Transition table (State/Event table)
- State Diagram
- UML State Machines

State Transition table (State/Event table)

The turnstile state machine can be represented by a state transition table which is shown in Table 3.1. This table shows for each state and from each input, what will be the new state and the output result.

State Diagram

FSM can also be represented by a directed graph called a state diagram. Each of the states is represented by a node (shown as circle) and edges (shown as arrows) are the transitions from one state to another. Each arrow is labeled with the input or condition that triggers that transition. Inputs or conditions that don't trigger a change of state (such as a coin input in the Unlocked state) are represented by a circular arrow returning to the original state. Figure 3.5 depicts state diagram of a common turnstile. The initial state is presented as an arrow into the Locked node from the black dot.

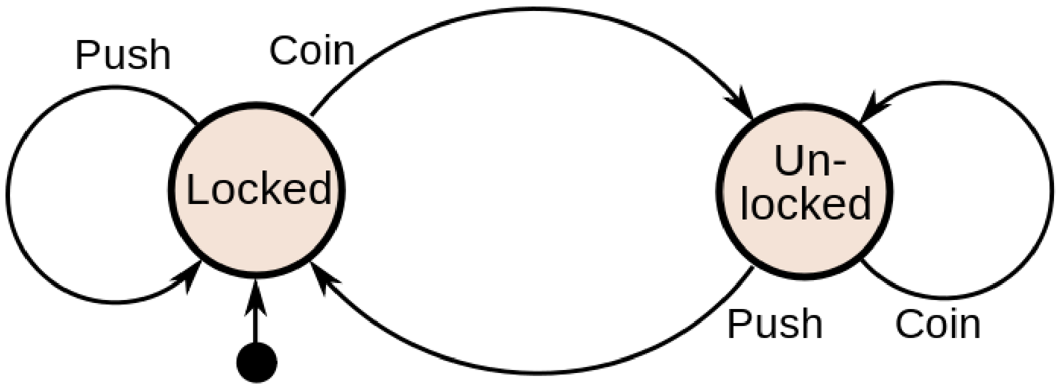


Figure 3.5: Turnstile State Diagram

UML State Machines

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system[119]. For describing state machines, UML has a notation. UML state machines solve the limitations of traditional FSM while retaining their main benefits. UML state machines propose the new concepts of hierarchically nested states and orthogonal regions, while extending the notion of actions. These method of modeling an FSM have the characteristics of both Mealy machines and Moore machines. They support actions that depend on both the state and triggering events which are similarly defined in Mealy machines, as well as entry and exit actions, which are associated with states rather than transitions, as they are available in Moore machines. In addition, after defining an FSM for software programming, there are several high performance libraries for realtime applications to implement UML state machines such as Boost library Meta State Machine and StateChart [120]. Figure 3.6 shows turnstile UML tate machine diagram.

```
+----------------------------------------------------------------+
|                                                                |
|         +----------------+          +----------------+         |
|     *-->|     LOCKED     |          |    UNLOCKED    |         |
|         +----------------+          +----------------+         |
|     +---| entry/         |          | entry/         |---+     |
|     |   | exit/          |          | exit/          |   |     |
|     |   |                |          |                |   |     |
| PUSH|   |                |---COIN-->|                |   |COIN |
|     |   |                |          |                |   |     |
|     |   |                |          |                |   |     |
|     |   |                |<--PUSH---|                |   |     |
|     +-->|                |          |                |<--+     |
|         |                |          |                |         |
|         +----------------+          +----------------+         |
|                                                                |
+----------------------------------------------------------------+
```

Figure 3.6: Turnstile UML State Diagram

3.3.2 UML State Machines Implementation

The following terms describe a state machine based on UML State Machines implementation:

- **State**: a stage in the life cycle of a state machine. A state (like a submachine) can have an entry and exit behaviors.
- **Initial State**: The state in which the state machine starts.
- **Transition**: a specification of how a state machine reacts to an event. It specifies a source state, the event triggering the transition, the target state, guard and actions.
- **Action**: an operation executed during the triggering of the transition.

- **Guard**: a boolean operation being able to prevent the triggering of a transition which would otherwise fire.

3.3.3 Finite State Machine (FSM)

A finite-state machine (also called state machine), is a mathematical model of computation which is used for designing computer programs. The machine is in only one state at a time. The state of machine at any given time is called the current state. A triggering event or *Condition* can change from one state to another and is called a *Transition.* A particular FSM can be defined by a list of its states, and the triggering *Condition* for each *Transition.*[121, 122]

3.3.4 FSM Mathematical Model

A finite state machine is a quintuple $(\Sigma, S, s_0, \delta, F)$, where:

- Σ is the input alphabet (a finite non-empty set of symbols).
- S is a finite, non-empty set of states.
- s_0 is the initial state, an element of S. In a nondeterministic finite automaton, s_0 is a set of initial states.
- δ is the state-transition function: $\delta : S \times \Sigma \to S$.
- F is the set of final states, a subset of S

3.4 Deep Learning

Hierarchical or structured deep learning is a modern branch of machine learning that was inspired by human brain. It has been developed based on complicated algorithms that model high-level features and extract those abstractions from data by using similar neural network architecture but much complicated. The neuro-scientists discovered the neocortex which is a part of the cerebral cortex concerned with sight and hearing in mammals, process sensory signals by propagating them through a complex hierarchy over time. That was the main motivation to develop the deep machine learning focusing on computational models for information representation that exhibit similar characteristics to that of the neocortex [123] [124] [125].

3.4.1 Convolutional Neural Networks (CNNs / ConvNets)

Convolutional Neural Networks (CNNs) that are inspired by the human visual system are similar to classic neural networks. This architecture has been specifically designed based on the explicit assumption that raw data are comprised of two-dimensional images that enable certain properties to be encoded while also reducing the amount of hyper parameters. The topology of CNNs utilizes spatial relationships to reduce the number of parameters that must be learned, thus improving upon general feed-forward backpropagation training [14] [15]. Equation 3.15 demonstrates how the gradient component for a given weight is calculated in the backpropagation step, where E is error function, y is the $neuron_{ij}$, x is the input, l represents layer numbers, w is filter weight with a and b indices, N is the number of neurons in a given layer, and m is the filter size.

$$\frac{\partial E}{\partial \omega_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{\ell}} \frac{\partial x_{ij}^{\ell}}{\partial \omega_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{\ell}} y_{(i+a)(j+b)}^{\ell-1} \tag{3.15}$$

Equation 3.15 describes backpropagation error for the previous layer using the chain rule. This equation is similar to the convolution definition where $x_{(i+a)(j+b)}$ is replaced by $x_{(i-a)(j-b)}$. It demonstrates the backpropagation results in convolution while the weights are rotated. The rotation of the weights derives from delta error in Convolutional Neural Network.

$$\begin{aligned}\frac{\partial E}{\partial y_{ij}^{\ell-1}} &= \sum_{a=0}^{m-1}\sum_{b=0}^{m-1}\frac{\partial E}{\partial x_{(i-a)(j-b)}^{\ell}}\frac{\partial x_{(i-a)(j-b)}^{\ell}}{\partial y_{ij}^{\ell-1}} \\ &= \sum_{a=0}^{m-1}\sum_{b=0}^{m-1}\frac{\partial E}{\partial x_{(i-a)(j-b)}^{\ell}}\omega_{ab}\end{aligned} \tag{3.16}$$

In CNNs, small portions of the image (dubbed a local receptive field) are treated as inputs to the lowest layer of the hierarchical structure. One of the most important features of CNN is that complex architecture provides a level of invariance to shift, scale and rotation as the local receptive field allows the neuron or processing unit access to elementary features such as oriented edges or corners. This network is basically made up of neurons having learnable weights and biases forming Convolutional Layer. It also includes other network structures such as Pooling Layer, Normalization Layer and Fully-Connected Layer. As briefly mentioned above, Convolutional or so called CONV layer computed the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume. Pooling or so called POOL Layer performs a down sampling operation along the spatial dimensions. The Normalization Layer or RELU layer applies an element-wise activation function, such as the max (0, x) thresholding at zero. This layer does not change the size of the image volume. Fully-Connected Layer (FC) layer computes the class scores, resulting in volume of number of classes. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume [123] [124]. The Convolutional Layer plays an important role in CNN architecture and is the core building block in this network. The CONV layer's parameters consist of a set of learnable filters. Every filter is spatially small but extends through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, producing a 2D activation map of that filter. During this convolving, the network learns filters that activate when they see some specific type of feature at some spatial position in the input. Next, these activation maps are stacked for all filters along the depth dimension forms the full output volume. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at only a small region in the input and shares parameters with neurons in the same activation map [123] [124] [125].

A Pooling layer is usually inserted in-between successive Conv layers in ConvNet architecture. Its function is to reduce (down sample) the spatial size of the representation in order to reduce the amount of network hyper parameters, and hence to also control over-fitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. Recently, more successful CNN have been developed such as LeNet, AlexNet, ZF Net, GoogleNet, VGGNet and ResNet. The major bottleneck of constructing ConvNet architectures is the memory restrictions of GPU [123] [124] [125].

3.4.2 Popular Deep Learning Networks

Several network architectures are more common in the field of Convolutional Neural Networks:

- **LeNet**: The first successful applications of Convolutional Networks were developed by Yann LeCun[124]. Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc.

- **AlexNet**. The first work that popularized Convolutional Networks in Computer Vision was the AlexNet, developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton[126]. The AlexNet was submitted to the ImageNet ILSVRC (ImageNet Large Scale Visual Recognition Competition) challenge in 2012[127] and significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error). The Network had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other (previously it was common to only have a single CONV layer always immediately followed by a POOL layer).

- **ZF Net**: The ILSVRC 2013 winner was a Convolutional Network from Matthew Zeiler and Rob Fergus[128]. It became known as the ZFNet (short for Zeiler & Fergus Net). It was an improvement on AlexNet by tweaking the architecture hyper-parameters, in particular by expanding the size of the middle convolutional layers and making the stride and filter size on the first layer smaller.

- **GoogLeNet**: The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al[129]. from Google. Its main contribution was the development of an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M). Additionally, this paper uses Average Pooling instead of Fully Connected layers at the top of the ConvNet, eliminating a large amount of parameters that do not seem to matter much. There are also several follow-up versions to the GoogLeNet, most recently Inception-v4[130].

- **VGGNet**: The runner-up in ILSVRC 2014 was the network from Karen Simonyan and Andrew Zisserman that became known as the VGGNet[131]. Its main contribution was in showing that the depth of the network is a critical component for good performance. Their final best network contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. Their pre-trained model is available for plug and play use in Caffe. A downside of the VGGNet is that it is more expensive to evaluate and uses a lot more memory and parameters (140M). Most of these parameters are in the first fully connected layer, and it was since found that these FC layers can be removed with no performance downgrade, significantly reducing the number of necessary parameters.

- **ResNet**: Residual Network developed by Kaiming He et al[132]. was the winner of ILSVRC 2015. It features special skip connections and a heavy use of batch normalization. The architecture is also missing fully connected layers at the end of the network. ResNets are currently by far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice.

3.4.3 LeNet

As you can see in the Figure 3.7, LeNet was firstly designed by Y. LeCun et Al. [124] and this famous network successfully classified digits and was applied to hand-written check numbers. The application of this network was expanded to more complicated problems and their hyper parameters adjusted for new issues. However, more developed versions of LeNet have been successfully tested. In this work, we dealt with a binary but very complicated binary classification of Alzheimer's and Normal data. In other word, we needed a complicated network but for two classes which was leading us to choose LeNet and adjust this architecture for binary image data.

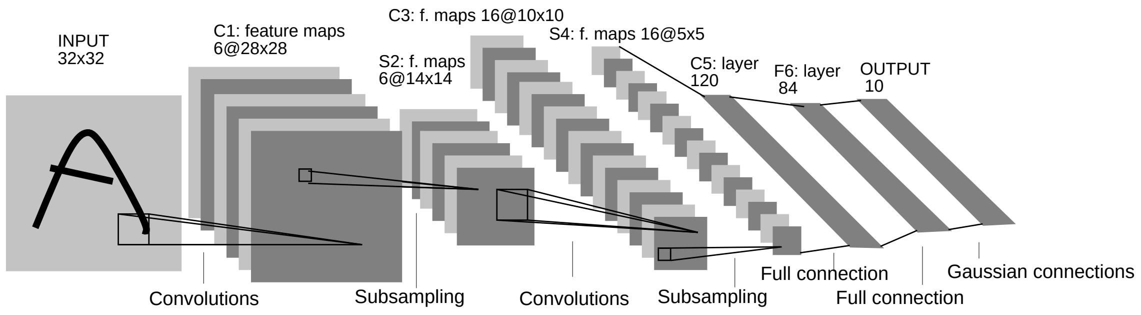


Figure 3.7: Architecture of LeNet-5, a convolutional neural network

The implemented network is shown in Figure 3.8 in details.

3.5 Summary

This chapter contains basic information about two important classifiers, SVM, and Convolutional Neural Network, and also feature reduction method (Principal Component Analysis (PCA)) which are used in our proposed methods. We have also presented Finite State Machines and different definitions for our application. After a short review of each concept, we propose our approach and configurations in the following chapters.

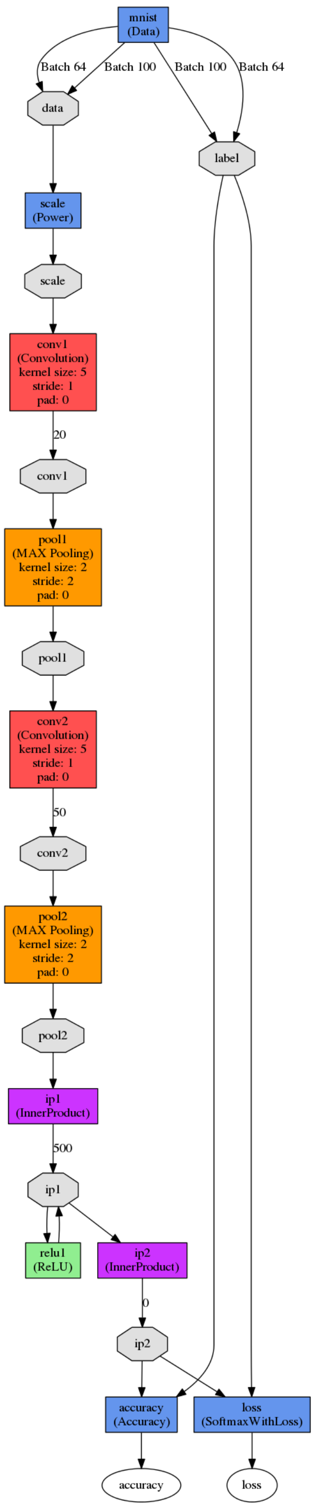


Figure 3.8: LeNet network implemented for binary image classification

Chapter 4

The Proposed Technique

4.1 Our Approach

For hand gesture and posture recognition, we have presented a framework that initially detects user's engagement and afterwards employs related classifiers for posture or gesture classification. Figure 4.1 depicts the framework of our approach.

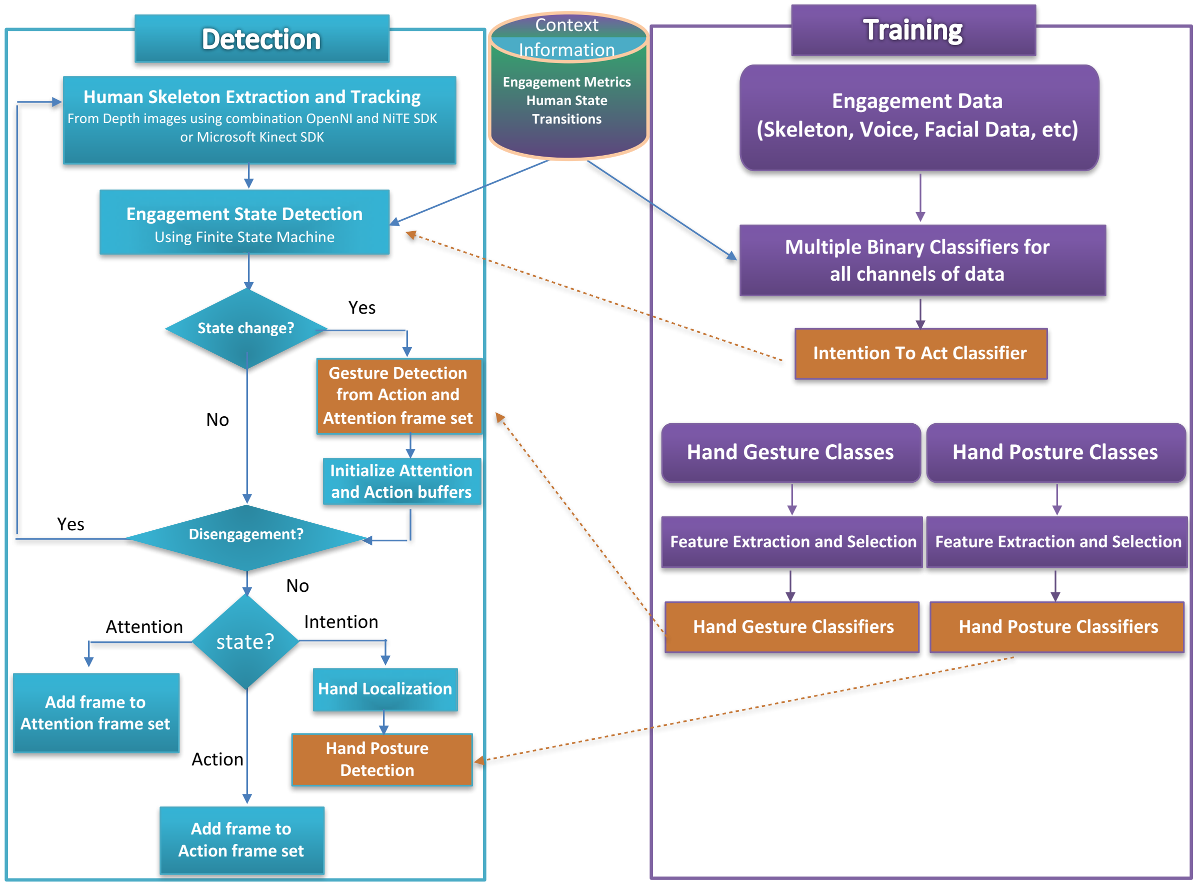

Detection
Context Information
Engagement Metrics
Human State Transitions
Training
Human Skeleton Extraction and Tracking
From Depth images using combination OpenNI and NiTE SDK or Microsoft Kinect SDK
Engagement Data (Skeleton, Voice, Facial Data, etc)
Engagement State Detection
Using Finite State Machine
Multiple Binary Classifiers for all channels of data
State change?
Yes
No
Gesture Detection from Action and Attention frame set
Intention To Act Classifier
Initialize Attention and Action buffers
Hand Gesture Classes
Hand Posture Classes
Yes
Disengagement?
Feature Extraction and Selection
Feature Extraction and Selection
No
Attention
state?
Intention
Hand Gesture Classifiers
Hand Posture Classifiers
Add frame to Attention frame set
Hand Localization
Action
Hand Posture Detection
Add frame to Action frame set

Figure 4.1: Engagement Detection Framework for hand gesture and posture recognition

Engagement Detection Framework: Context Information

Context information in Figure 4.1 refers to those facts we have from literature reviews regarding important factors in engagement detection, especially based on studies of engagement metrics in real world applications such as video games, virtual reality and people's interaction in meeting rooms. This knowledge helps us in designing effective classifiers that extract patterns of engagement for the purpose of hand posture and gesture recognition. Also, it helps us to understand how engagement changes during time to design suitable framework for human state transitions.

Engagement Detection Framework: Terms and Definitions

Engagement metrics and states is discussed in previous sections in details. We can categorize different states of engagement as it is presented in state machine of Figure 4.2. This state machine is discussed in details in section 4.3.6 and consists of four different distinguishable mental states. We proposed these states as *Disengagement*, *Attention*, *Intention*, and *Action*. *Disengagement* occurs when user is disengaged from target object. *Attention* happens when user has attention to the target, but doesn't have intention to perform any actions. For instance, user is observing the objects without any intentions to manipulate them. In *Intention* state, user intends to perform an action, but still not doing it. For example, user is showing an ASL sign or pointing to an object to select it, but not performing any manipulative actions yet. These static hand postures in *Intention* state could be classified using hand posture classifiers which is discussed in section 4.4. Finally, in *Action* state, user is performing a manipulative action with hand. Those consecutive frames in *Action* state could be imported to hand gesture classifier which is in our future research.

Engagement Detection Framework: Training

In training part of Figure 4.1, binary classifiers are designed based on upper-body postures. The output of these binary classifiers are used to design the most important classifier in engagement detection framework which is *Intention to Act classifier*, and plays essential role for distinguishing different states of engagement in Figure 4.2. Training section in Figure 4.1 also shows classifiers for hand posture and gesture recognition. In this research, classifiers for hand posture classification is proposed and discussed in section 4.4. Hand posture classifiers are based on single frames. For hand gestures such as raising

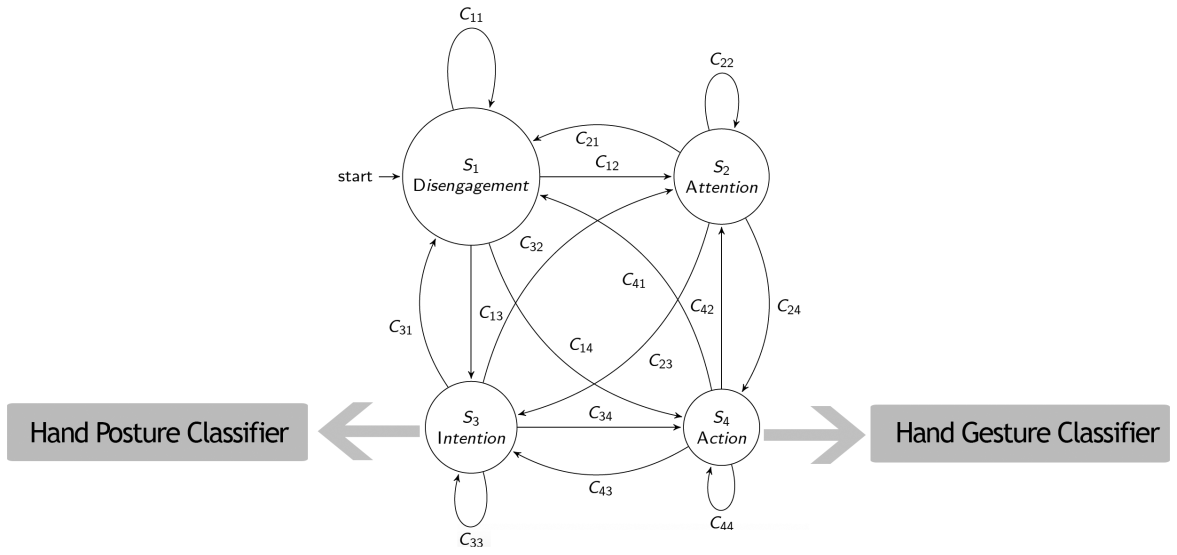


Figure 4.2: Engagement Detection State Machine for Hand Posture and Gesture Recognition

hand or swiping from right to left, we need a extract features from sequence of frames in *Action* state. Hand gesture recognition is in our future research.

Engagement Detection Framework: Detection

Detection part of Figure 4.1 starts with extracting human skeleton and important joints using depth images. OpenNI and Nite SDK provide human skeleton from depth data for each frame. The human skeleton is provided by detection of 15 important joints in human body which are world locations of head, neck, left and right shoulders, left and right elbows, left and right hand, torso, left and right hips, left and right knees, and left and right feet. Among all these joints, upper-body joints play important role in engagement detection which is comprehensively discussed in section 4.2. The binary classifiers trained in *Training* section of the framework also used this joint information.

The joints location provided by OpenNI and Nite SDK is not stable all the time and could be misleading. The joints also could be provided by Microsoft Kinect SDK, which provides more stable joints, but also with some errors. Because our engagement classifiers are highly dependent to this information, we have to design a system that provides minimum error. The classification of engagement state is frame-based and frames are classified independently. When we are processing 30 frames per second for our real-time application, it should considered that states couldn't transit to each other immediately. This comes from our knowledge of human state transition which is a continuous process and two consecutive frames tend to be in the same state for a very short amount of time between two frames. For a normal human being it takes more than hundred milliseconds to change body posture and therefore engagement state. But when we are just looking each frame independently, and the joints provided by the SDKs are noisy, our engagement classifiers may report wrong state for a frame. Therefore, there should be a minimum number of frames that each state contains to transit to the next state. Furthermore, the user may show some engagement postures for a short amount of time unintentionally, and we can't figure out it's a meaningful engagement or not unless we look at the sequence of frames. These considerations lead us to propose a finite state machine with transition and guard criteria to improve engagement state classification. One of the immediate results of proposed finite state machine implementation is gesture segmentation. Gesture segmention helps us to improve results of hand posture and gesture classifiers and speed up the algorithm. For example, hand posture classifiers are just used when we are in *Intention* state. In the same way, gesture classifiers are used at the end of *Action* state when state is transiting from *Action* to another state. The *Action* state frames could be gathered in a buffer to be used for gesture classification. Also we can have *Attention buffer* to gather those continuous frames related to *Attention* state. Attention frames could be used for other types of meaningful gestures in *Attention* state. For example, in a meeting room scenario, we can see how much user is involved in discussions by analyzing and classifying face expressions, gaze and nodding of the user in *Attention buffer* frames. Both *Action* and *Attention* classifiers are in our future research.

Hand Posture Classification

For real-time performance, our approach for posture classification is based on 2D appearance-based approach. Earlier in this research, we talked about a new dataset presented for training and testing the system and will be presented in details in section 4.4.1. In section 4.4.3, preprocessing steps, such as background subtraction for preparing images for feature extraction, is discussed. In section 4.4.4, a method for extracting features from hand postures is introduced. Section 4.4.5 details dimension reduction of extracted features using PCA. Afterwards, deep learning-based approach for hand posture recognition is presented.

4.2 Engagement Modeling and Metrics

User mental status detection has variety of applications in human activity recognition systems. Without a proper algorithm for detecting human intention to interact, the vision-based system is always on, therefore any kind of activity may interpreted as an interaction [133]. Group meetings which are frequent business events is modeled as a case study. In this case study, among all available data streams, a combination of tracking 3D gesture data is combined for user engagement detection with the vision-based screen in meetings. Multiple binary classifiers are implemented to detect user intention for performing an action. The output of these binary classifiers are used to create transition and guard conditions in FSM. Characteristics of engagement will be discussed and biometric data which can be used for this purpose will be introduced. 3D skeleton tracking will be introduced as one of the channels of biometric information for engagement detection. Although we just use this only channel of biometric data, experiment results show we still can predict engagement with high accuracy.

In addition, DAIA, the Finite State Machine (FSM) of engagement detection helps system to flow among states smoothly. FSM is a predefined structure based on our knowledge of human activities that helps system predict engagement state more accurately. Furthermore, FSM algorithm is computational efficient. This property of FSM allows achievable on-line and real-time performance.

In this section, we proposed a multi-modal engagement state detection process. In our intended scenario, multiple people are within the operating range of the sensor, e.g. field of view of a depth camera. For the purpose of this research, we have just monitored one user's biometric information to for engagement detection. The same procedure could be processed in parallel to detect engagement state of other participants. In our proposed framework, user'engagement scales from disengaged up to performing an action.

4.2.1 Engagement Modeling: DAIA

Figure 4.3 shows user's engagement scale. From left to right, the level of engagement increases. For hand gesture and posture classification, we can quantize the spectrum of this mental state in finite number of states as it is shown in Figure 4.3. We called this model of engagement DAIA which quantized engagement in four different states which are Disengagement, Attention, Intention, and Action.

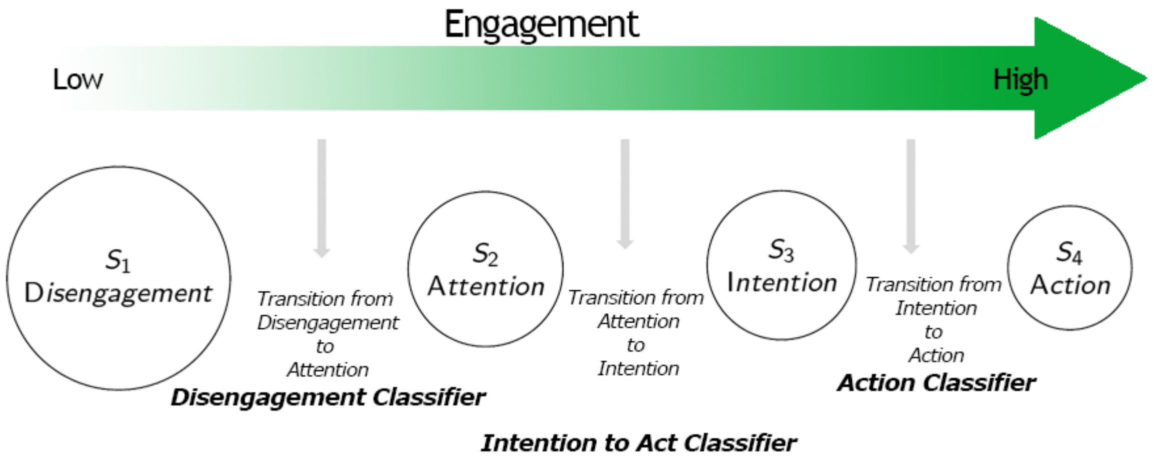

Engagement
Low
High
S_1 Disengagement
Transition from Disengagement to Attention
S_2 Attention
Transition from Attention to Intention
S_3 Intention
Transition from Intention to Action
S_4 Action
Disengagement Classifier
Action Classifier
Intention to Act Classifier

Figure 4.3: Engagement scales from Disengagement to Action. Engagement is quantized in four different states which are Disengagement, Attention, Intention, and Action.

4.2.2 Engagement Metrics

From literature survey, we know Upper-body joints play important role in engagement detection. Multiple classifiers are designed to detect and classify upper-body direction. In addition, hand movements such as raising hand, pointing, swiping, pushing or pulling are used to manipulate in vision-based interfaces. Therefore, different classifiers are designed to detect various hand movements such as raise hand above waist and also different levels of hand speed. These classifiers help to detect user intention for performing an action. Furthermore, leaning forward when observing is an important sign for higher engagement. Also, the direction of body shows the direction that user intends to engage. Obviously, for hand posture and gesture recognition, location and speed of hands are important factors. Based on these assumptions, a set of binary classifiers are designed.

Binary Classifiers

Binary classifiers are mostly designed based on heuristic information extracted from 3D joints location. For example, by knowing the location of hand joint and head joint we can easily say the hand is above head or not. Another example is body direction classifier which is made using the normal vector of the plane containing right and left shoulder and torso joint.

Using the biometric information, the engagement detection framework identifies if the user exhibits a specific combination of classifiers. The analysis is occurring on a frame-by-frame basis. Each frame is analyzed regarding all classifiers. The classifiers, e.g. body posture components, are chosen based on literature review.

Table 4.1 shows binary classifiers which are designed for important engagement metrics. For each engagement metrics, multiple binary classifiers are designed. The 0 or 1 output of these 37 classifiers are used to make our feature vector for intention to action classifier.

Classifiers are evaluated as being exhibited or not exhibited in a specific frame as a binary value. Among all available biometric information such as gaze, voice and gesture to extend the proposed framework to detect engagement state of the user, we just used 3D joint information provided by a depth camera (Asus Xtion Pro) and NiTE SDK by Primesense. However, we believe using more biometric channels of information will make system more accurate, our experiment results showed even use this

Table 4.1: Binary classifiers based on engagement metrics

Engagement Metric	Binary Classifiers
Hand Horizontal	Right of Body, Close to Body, Left of Body
Hand Vertical	Below Hip, Below Torso, Below Shoulder, Below Head
Hand Depth	Back of Body, Close to Body, Front of Body
Hand Speed	Stopped, Slow, Fast, Too Fast
Body Direction	Facing Sensor
Leaning	Lean back, No Lean, Lean Forward
Special Postures	Hands folded, Hands on Head, Hands on Torso

only channel of information could result in a high performance user engagement detection system.

3D Skeleton Data

An action video with T frames and N joints in each frame can be represented as a set of 3D points sequence, written as $p = \{x_n^t \in \mathbb{R}^3 | n = 1, .., N, t = 1, 2, ..., T\}$. The 3D sensor provides us fifteen joints, and T varies for different sequences. However, in our system $N = 10$, because our classifiers only use ten upper-body joints from this set which are head, left and right shoulders, left and right elbows, left and right hand, torso and left and right hips. Figure 4.4 shows all joints provided by NiTE SDK by Primesense. Upper-body joints which are used to design our classifiers are shown as filled circles.

For each joint in volume of interest, 3 dimensional position X, Y, D is obtained. X and Y are horizontal and vertical distances of the joints from the center of sensor. D is the distance of the joint from sensor in Z direction as it is shown in Figure 4.5.

Upper-body 3D Joint Time Series

The output of sensor could provides a time series of 3D Joints. In equation 4.1, 3D joint time series, $P_{3\times N\times T}$ is presented. For upper-body, $N = 10$, and T (frame number in time sequence) varies for

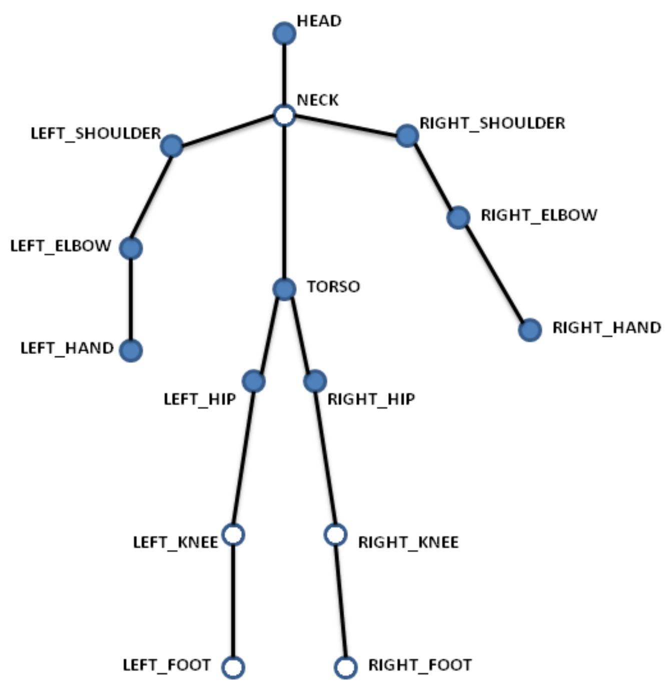


Figure 4.4: Fifteen joints provided by Nite SDK. Joints which are depicted by filled circles are upper-body joints which are used in binary classifier design

different sequences.

$$P_{3\times N\times T} = \{ \begin{bmatrix} X_1 & X_2 & ... & X_N \\ Y_1 & Y_2 & ... & Y_N \\ D_1 & D_2 & ... & D_N \end{bmatrix}_t = p_n^t \in \mathbb{R}^3 | \quad n = 1,..,N \quad \text{and} \quad t = 1,2,...,T\} \tag{4.1}$$

Equation 4.2 shows matrix which contains N different joints in $t = t_0$, and is called $Frame_{t_0}$.

$$Frame_{t_0} = \begin{bmatrix} X_1 & X_2 & ... & X_N \\ Y_1 & Y_2 & ... & Y_N \\ D_1 & D_2 & ... & D_N \end{bmatrix}_{t=t_0} \tag{4.2}$$

The first basic step of feature extraction is computing basic feature for each frame, which describes the

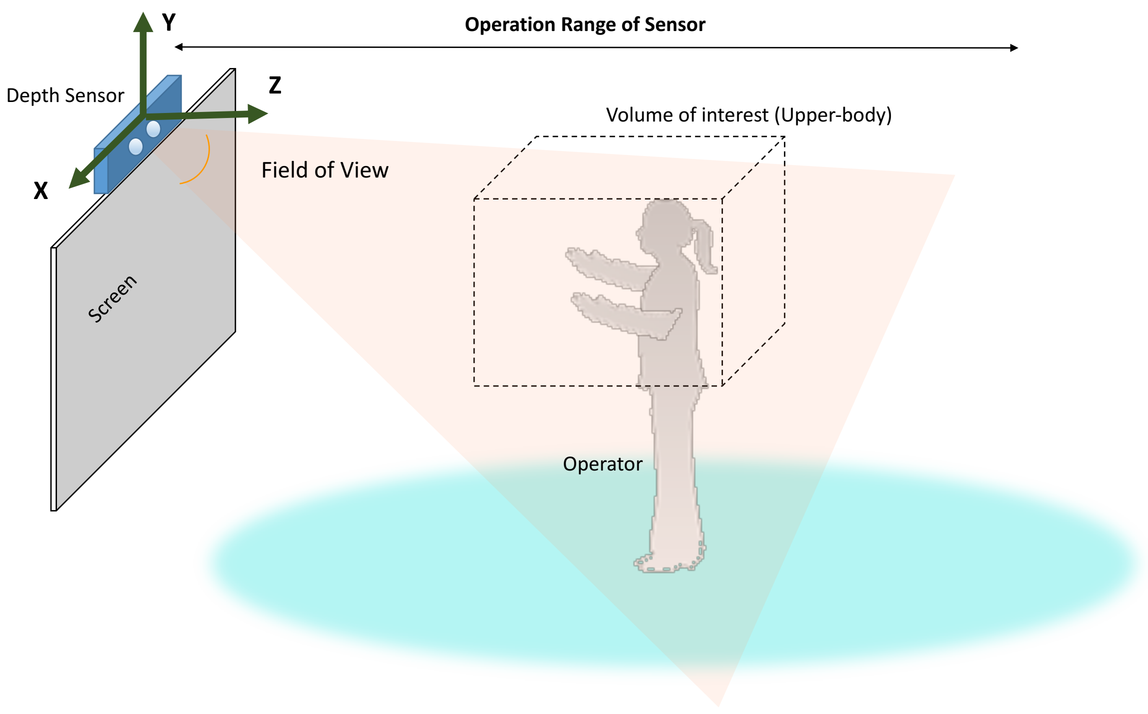


Figure 4.5: Operator in the range of operation and field of view of a depth sensor. The volume of interest is the volume that track joints world locations are tracked

pose information of every of these ten joints in a single frame. These feature extractions are performed using binary classifiers. All the binary classifiers at time $t = t_0$ is calculated based on matrix operations on $Frame_{t_0}$ itself.

The second step is calculating of left and right hand speed information. This features are obtained by calculating 3D distance that each hand moves in two consecutive frames, namely $Frame_{t_k}$ and $Frame_{t_{k+1}}$.

Classification Based on Engagement Score

One way for evaluating the level of engagement is calculating an engagement score based on the features of each frame. The binary values for the individual binary classifiers are weighted based on the relative influence in the training data and summed up to. The relative influence is applied using a weight factor for the output of the classifier. After normalization, the engagement score could have a value between 0

Table 4.2: Binary classifiers for engagement score calculation. Model weights are calculated based on statistical analysis and applies to the output of binary classifiers.

Binary Classifier	Model Weight
Engaged stance	0.67
Hand lifted above waist	0.15
Looking at screen	0.12
Waving	0.11
Hand raised above head	0.08
Body facing screen	-0.05

and 1.

Figure 4.6 shows how our system extract features from users and calculate engagement levels of each users.

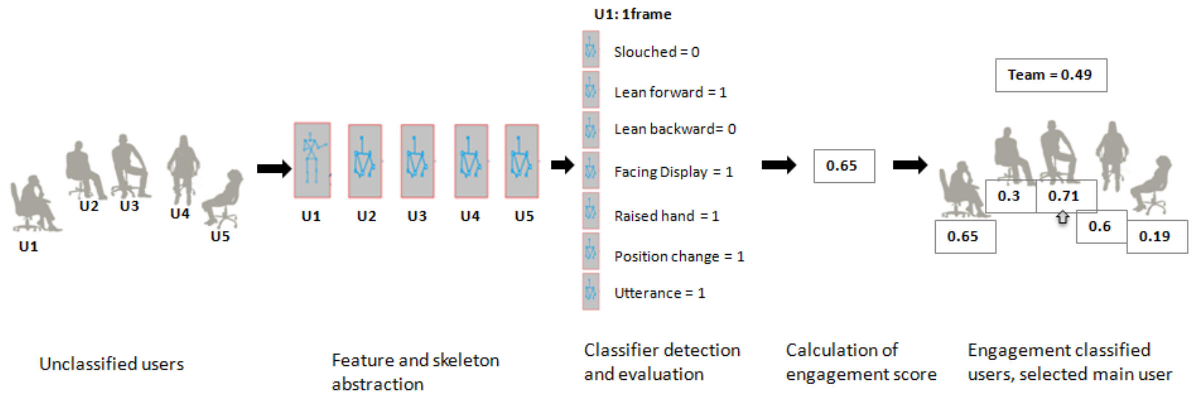


Figure 4.6: Engagement score calculation. Using this approach we can select the most engaged person among other participants. Furthermore, by setting thresholds on engagement score, we can classify engagement states.

Similar but fewer binary classifiers are presented by [19] and engagement score is calculated based on the output of classifiers. These binary classifiers are shown in table 4.2.

In that approach, the engagement score is calculated using $W'.G$ which W is vector of weights $[w_1, w_2, w_3, ..., w_n]$, and G is the vector of binary classifiers $[g_1, g_2, g_3, ..., g_n]$ such that $g_1, g_2, g_3, ..., g_n$ are 0 or 1 based on the output of the binary classifiers. There are several problems if we use this approach. First of all, weights should be defined to apply on each binary classifiers. In addition, we have to define several thresholds for engagement scores to find engagement state. In [19], statistical approach for defining weights is presented and calculated weights are gathered in table 4.2 .

There are some drawbacks in using engagement score for engagement detection. First of all, if a new channel of biometric data will be added for better engagement detection, a whole new statistical analysis is necessary for different weights assignment. In addition, new thresholds for state changes should be calculated. To overcome these problems, instead of calculating engagement scores, we have trained classifiers to directly detect the engagement states which are important to us. For training a classifier, we need a feature vector. We have defined *Engagement Feature Vector* for this purpose.

4.3 Engagement Detection Framework

4.3.1 Engagement Feature Vector

Classifiers depicted in Table 4.1 provide 0 and 1 outputs. For each frame, we can define an *Engagement Feature Vector* which is called E.

Instead of calculating weights that should be applied to the outputs, we have used SVM classifiers for engagement state prediction. Therefore, weight are assigned based on training over ground truth data. The details of our classification approach is presented in the following sections.

Engagement Feature Vector Time Series

Engagement feature vectors during time creates a time series. Equation 4.3 shows this time series. In this equation, C is the number of binary classifiers. In our research, binary classifiers are just based on Upper-body 3D joint data, however, the feature vector could be extended for binary classifiers from other modalities. For instance, we can design binary classifiers based on other biometric information such as gaze, weight sensors on chair, voice levels and its direction toward the target, etc. Therefore, our approach suggests a paradigm for multi-modal information fusion for engagement modeling.

$$E_{C\times T} = \{ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ . \\ . \\ . \\ b_{C-1} \\ b_C \end{bmatrix}_t = e_t \in \{0,1\}^C | c = 1, ..., C \quad \text{and} \quad t = 1, 2, ..., T\} \tag{4.3}$$

Multi-modal Data To Binary Classifiers

Our proposed method maps 3D joint time series,$P_{3\times N\times T}$ to engagement vector time series, $E_{C\times T}$. Equation 4.4 shows this mapping.

$$f : P_{3\times N\times T} \longmapsto E_{C\times T} \tag{4.4}$$

Figure 4.7 shows how 3D joints location are mapped to binary classifiers for a user. As it is already mentioned, biometric information that could extend engagement feature vector is not limited to 3D joints.

Binary Classifiers Based on Hand Location

Hand spatial location is one of the most important parameters in engagement detection for hand posture and gesture recognition. The spatial location is categorized in three space direction which are horizontal, vertical and depth location in 3D space. For each direction, the location of hand versus other parts of the body gives of information about engagement. The summary of these binary classifiers are as follows:

1. Hand's horizontal location

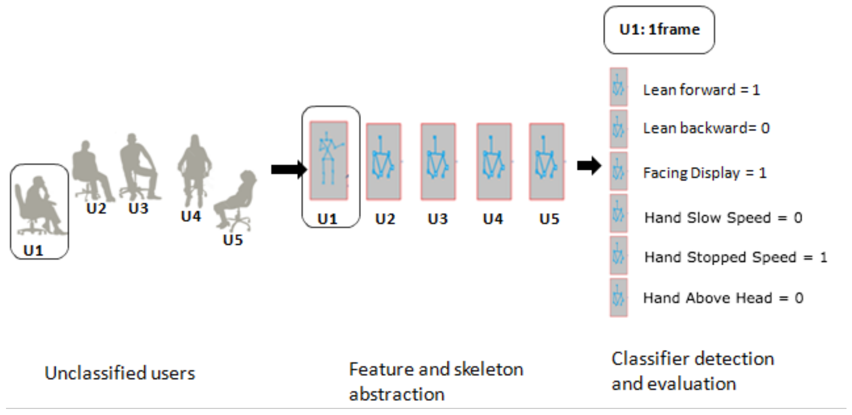


Figure 4.7: 3D Joint Time Series Mapping To Engagement Vector Time Series

- Right of body
- Close to body
- Left of body

2. Hand's vertical location

   - Below Hip
   - Below Torso
   - Below Shoulder
   - Below Head

3. Hand's depth location versus the center of depth sensor

   - Back of body
   - Close to body
   - Front of body

For these binary classifiers, several thresholds are defined. For example, in horizontal location classifiers for each hand, if the location of hand joint is right of the torso point and horizontal distance is grater than a predefined threshold, that *Right of body* binary classifier is 1 and *Close to body* and *Left of body* binary classifiers are 0.

Binary Classifiers Based on Hand Speed

Speed of each hand plays important role for classification between *Action* and *Intention* states. Real hand speed value is calculated from simple matrix operation between two consecutive frames. We have defined several levels for speed of each hand which helps us to extract useful information. In this approach, the speed of each hand is quantized in four different states which are *Stopped*, *Slow*, *Fast* and *Too Fast*. For this purpose, three thresholds are applied to hand speed value based on experimental results.

Facing Classifier Based on Body Direction Vector

Another important classifier is *Facing target classifier* which gives us information about engagement of the body with target object. This classifier tells us if the direction of body is toward target object. Body direction vector could help us in calculating the output of facing classifier. Figure 4.3.1 shows how we calculated body direction.

Equation 4.5 shows how body normal vector is calculated for body direction estimation at frame t. Vectors in Figure 4.3.1 are used in equation 4.5 as follows:

- *Left Shoulder* joint to *Right Shoulder* joint vecor: $\overrightarrow{LS2RS} = \overrightarrow{p_{rs}} - \overrightarrow{p_{ls}}$
- *Left Shoulder* joint to *Torso* joint vecor: $\overrightarrow{LS2TO} = \overrightarrow{p_{ls}} - \overrightarrow{p_{to}}$
- *Body Normal Vector*, $\overrightarrow{N}$, which is made from cross product between $\overrightarrow{LS2RS}$ and $\overrightarrow{LS2TO}$. On the other hand, $\overrightarrow{N}$ is normal vector of the plane consisting of left and right shoulders and torso joints

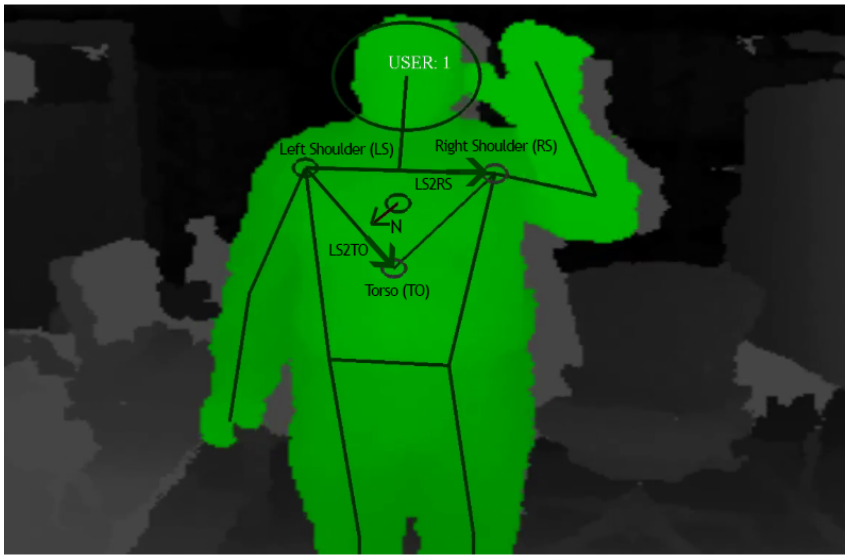


Figure 4.8: Normal vector of the plane consisting of left and right shoulders and torso joints makes *Body Direction Vector*

$$\begin{aligned}
\overrightarrow{LS2RS}^t &= \overrightarrow{p^t_{rs}} - \overrightarrow{p^t_{ls}} \\
\overrightarrow{LS2TO}^t &= \overrightarrow{p^t_{ls}} - \overrightarrow{p^t_{to}} \\
\overrightarrow{N}^t &= \overrightarrow{LS2RS} \times \overrightarrow{LS2TO}
\end{aligned} \tag{4.5}$$

Facing classifier at frame t is defined in equation 4.6. In this equation, *Body Normal Vector* is projected in horizontal and vertical direction parallel to the target. Two predefined thresholds, *MaxDeviationH* and *MaxDeviationV*, denote maximum horizontal and vertical deviation respectively to make the output of facing binary classifier one.

$$FacingClassifier_t = \begin{cases} 1 & \text{if } |proj_{\overrightarrow{x}} \overrightarrow{N}^{\mathbf{t}}| \leq MaxDeviationH \wedge |proj_{\overrightarrow{y}} \overrightarrow{N}^{\mathbf{t}}| \leq MaxDeviationV \\ 0 & \text{if } otherwise \end{cases} \tag{4.6}$$

Leaning Classifiers Based on Body Direction Vector

Leaning directions towards target gives us important information about the level engagement towards a target. Based on this engagement metric, we have designed *Lean back*, *No Lean*, and *Lean Forward* binary classifiers. We could also add two more classifiers namely *Lean left* and *Lean right* in the same manner.

Equations 4.7, 4.8 and 4.9 show how we have calculated the output of these classifiers. Similar to other classifiers in these section, we set *LeanBackTh* and *LeanFwdTh* thresholds based on analyzing real data.

$$LeanBackClassifier_t = \begin{cases} 1 & \text{if } |proj_{\overrightarrow{y}} \overrightarrow{N}^{\mathbf{t}}| \geq LeanBackTh \\ 0 & \text{if } otherwise \end{cases} \tag{4.7}$$

$$LeanFwdClassifier_t = \begin{cases} 1 & \text{if } |proj_{\overrightarrow{y}} \overrightarrow{N}^{\mathbf{t}}| \leq LeanFwdTh \\ 0 & \text{if } otherwise \end{cases} \tag{4.8}$$

$$LeanForwardClassifier_t = \neg(LeanBackClassifier_t \vee LeanFwdClassifier_t) \tag{4.9}$$

Special Postures Binary Classifiers

Several special postures are defined as important engagement metrics. For the current research we have limited special postures to *Hands folded*, *Hands on Head*, and *Hands on Torso*. These postures indicate disengagement from the target and mostly are used in meeting room scenario. For example, hand on head posture could imply the user is thinking and not intending to perform an action with hand. *Hands folded* and *Hands on Torso* postures also could imply disengagement of the user towards target.

4.3.2 Disengagement Classifier

Disengagement classifier detects transition of engagement states between *Disengagement* and *Attention* as it is shown in Figure 4.9. Equation 4.10 shows how *DisengagementClassifier* at time t is calculated using binary operations of several binary classifiers in Table 4.1. The output of *DisengagementClassifier* is 1 if *FacingClassifier* is 0 *or* one of the special posture classifiers in section 4.3.1 is activated. On the other hand, if the user is in *Disengagement state* if not facing the target or is showing a special posture which implies disengagement.

$$\begin{aligned} DisengagementClassifier_t = \; & \neg(FacingClassifier)_t \vee (HandFoldedClassifier)_t \vee \\ & (HandsOnHeadClassifier)_t \vee (HandsOnTorsoClassifier)_t \\ & t = 1, 2, ..., T \end{aligned} \tag{4.10}$$

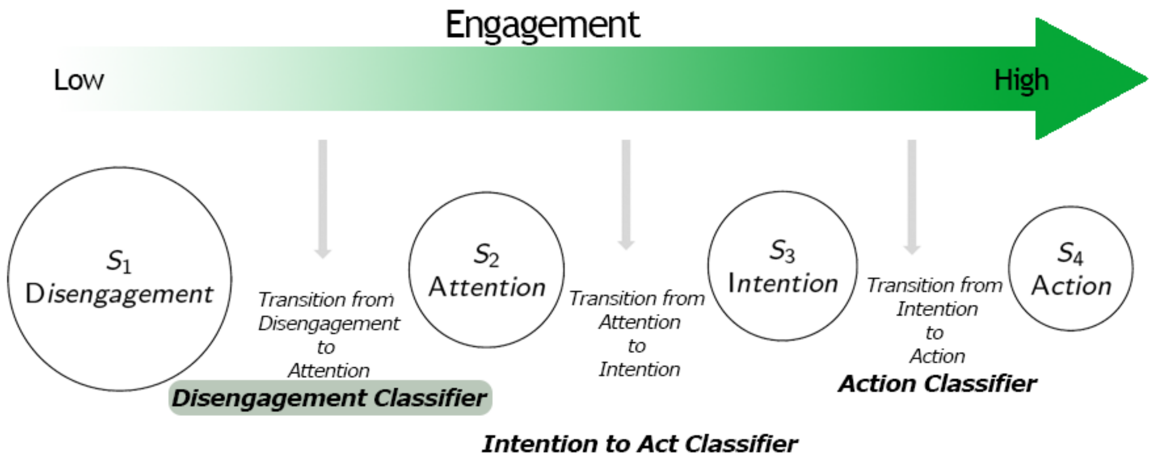

Engagement
Low
High
S_1 Disengagement
Transition from Disengagement to Attention
S_2 Attention
Transition from Attention to Intention
S_3 Intention
Transition from Intention to Action
S_4 Action
Disengagement Classifier
Action Classifier
Intention to Act Classifier

Figure 4.9: Disengagement Classifier

4.3.3 Intention To Act Classifier

The most important transition in DAIA is transition between *Attention* state to *Intention* state as is shown in 4.3.3. For designing this classifier, we have developed a game to capture user *Engagement Feature Vector Attention* and *Intention* states, and also transitions between these two. The game is called *Catch The Box!*, and automatically captures the frames with their *Engagement Feature Vector* and assigns *attention* (0) or *intention* (1) label to them.

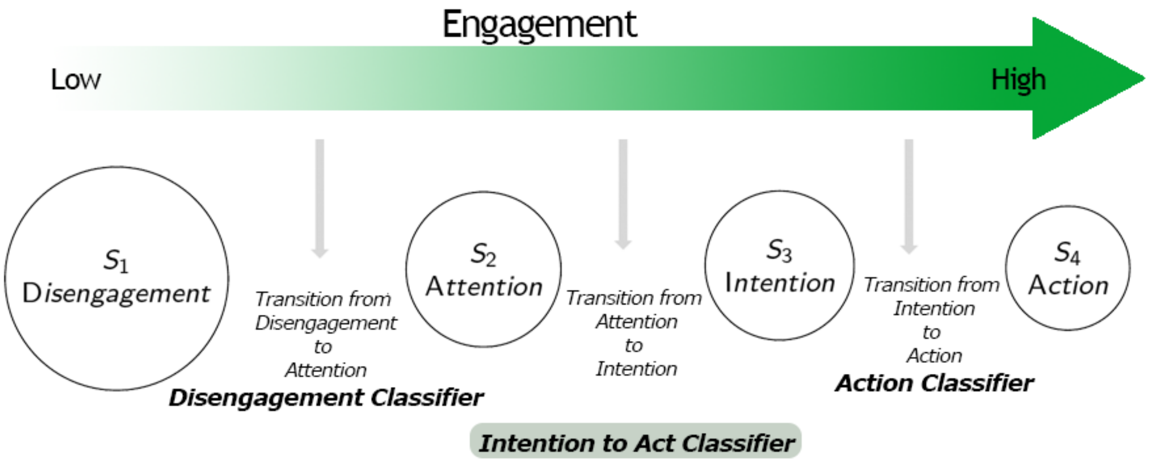

Engagement
Low
High
S_1 Disengagement
Transition from Disengagement to Attention
S_2 Attention
Transition from Attention to Intention
S_3 Intention
Transition from Intention to Action
S_4 Action
Disengagement Classifier
Action Classifier
Intention to Act Classifier

Figure 4.10: Intention To Act Classifier

Catch The Box!

The main purpose of designing this game is creating a dataset of labeled frames for two states of engagement which are *Attention* and *Intention*. These labeled frames help us to design *Intention Classifier*. *Intention Classifier* plays essential role in designing *Intention To Act* and *Action* classifiers. For the purpose of classifier design, the frames are labeled as *attention* (0) or *intention* (1). Therefore, players are supposed to be in of these two states when they are playing the game. Figure 4.11 shows different modes of the game.

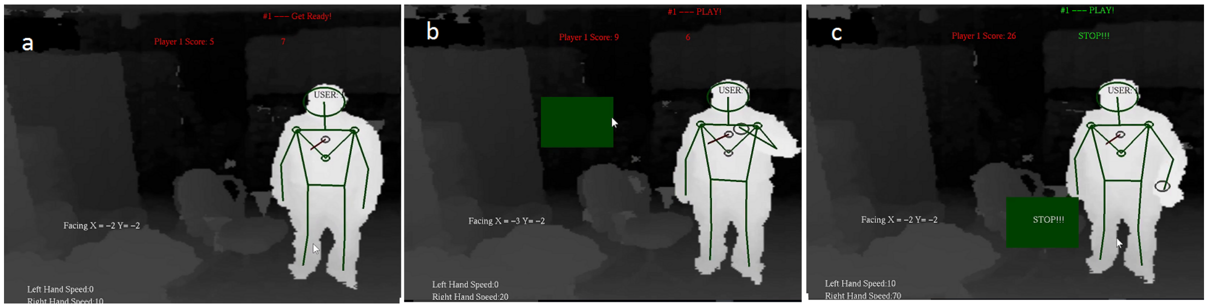


Figure 4.11: *"Catch The Box!"* Game: a) *Getting Ready* mode b) *Play* mode c) *Stop* mode

In this game, we used hand tracking algorithm implemented in NiTE middle-ware by Primesense [88] to move the cursor on screen. The game has 3 modes which are *"Getting Ready"*, *"Play"* and *"Stop"*. In *"Getting Ready"* mode, user sees the countdown counter for 5 seconds following by 5 seconds of *"Play"* mode. At the end of play mode, *"Stop"* sign appears for 2 seconds on screen to ask user to disengage from screen. After *"Stop"* mode, *"Getting Ready"* starts again and game continues in a repetitive loop.

In *"Getting Ready"* mode of the game we have asked user to observe the screen, but do not perform any actions until the countdown reaches 0 and *"Play"* mode starts. These frames in *"Getting Ready"* are automatically labeled as *attention*. When *"Play"* mode starts, a solid rectangle randomly appears on the screen and user should move the cursor towards the rectangle. As soon as the cursor touches the edge of rectangle, the position of rectangle changes randomly to force the user showing *"Intention"* or *"Action"* postures. The frames in *"Play"* mode are labeled as *intention*. In 2 seconds *"Stop"* mode we have asked user to disengage from screen as soon as possible. The frames in first second of *"Stop"* mode is labeled as *intention* and the rest of frames in the second second are labeled as *attention*. However the game assists in labeling the frames automatically, we manually reviewed all frames after capturing them to correct the ground truth labels. If the game mistakenly labeled a frame as a wrong state, we have changed it to the right label.

Intention Classifier

Ground truth data captured in *Catch The Box!* game is divided to training and testing data. Figure 4.12 shows 3-fold cross validation method is employed for training this classifier using SVM. 23,210 frames labeled from five different subjects played the game separately. 18,000 frames were used as training data and the rest were for testing data. This average validation accuracy based on 3-fold cross validation for SVM with linear kernel was 86.38% and the test accuracy was 84.74%.

Equation 4.12 shows how we trained our SVM classifier with linear kernel.

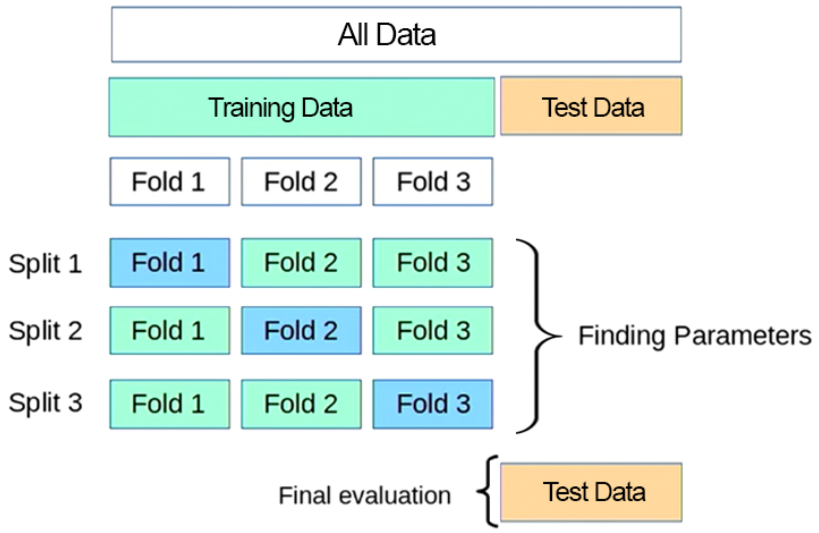


Figure 4.12: 3-fold cross validation

$$
\begin{aligned}
&E_{C\times T} = \{\left[b_1, b_2, b_3, ., ., ., b_{C-1}, b_C\right]_t^{'} = e_t \in \{0,1\}^C | c = 1, ..., C \quad \text{and} \quad t = 1, 2, ..., T\} \\
&E_{C\times T}(traingData) = \{e_t | Frame_t \text{ is labeled as intention (1) or attention (0)} \quad \text{and} \\
&t = 1, 2, ..., T_{traingData}\} \\
&IntentionClassifier = svmTrain(E_{C\times T}(traingData), Linearkernel)
\end{aligned}
\tag{4.11}
$$

Now, based on *IntentionClassifier*, we can define our *IntentionToActClassifier* as it is shown in equation 4.12. In this equation, when *DisengagementClassifier* is 0 and *IntentionClassifier* is 1 and both hands are stopped as we check with *leftHandStoppedClassifier* and *rightHandStoppedClassifier*, the output of *IntentionToActClassifier* becomes 1.

$$\begin{aligned} IntentionToActClassifier_t = \ & \neg DisengagementClassifier_t \wedge IntentionClassifier_t \wedge \\ & (leftHandStoppedClassifier_t \wedge rightHandStoppedClassifier_t) \\ & t = 1, 2, ..., T \end{aligned} \tag{4.12}$$

4.3.4 Action Classifier

Action Classifier detects the transition from *Intention* to *Action* as it is shown in Figure 4.3.4. *Action classifier* is similar to *Intention To Act Classifier* except that at least one of the hands should not be stopped. Equation 4.13 shows the binary operations for *Action classifier* output calculation.

$$\begin{aligned} ActionClassifier_t = \ & \neg DisengagementClassifier_t \wedge IntentionClassifier_t \wedge \\ & \neg(leftHandStoppedClassifier_t \wedge rightHandStoppedClassifier_t) \\ & t = 1, 2, ..., T \end{aligned} \tag{4.13}$$

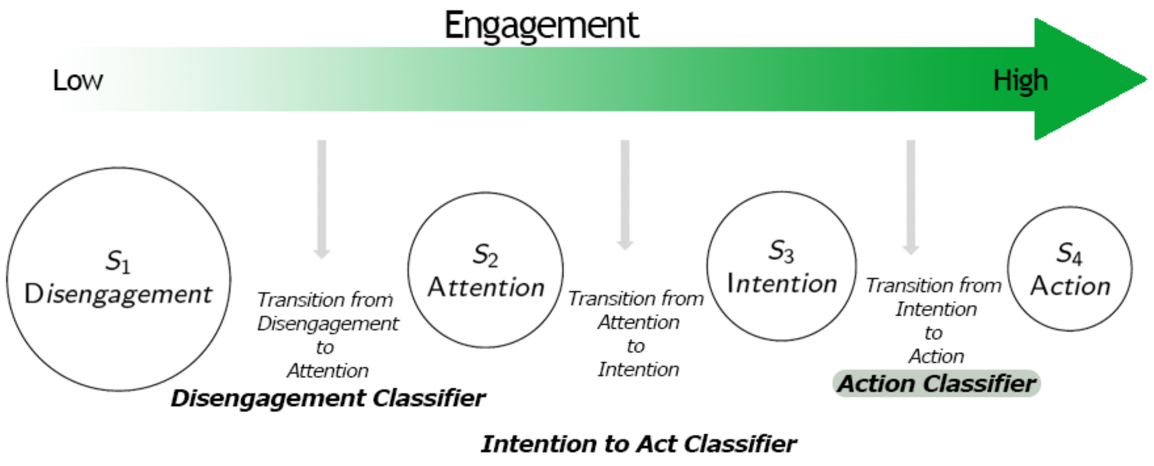

Engagement
Low
High
S1 Disengagement
Transition from Disengagement to Attention
S2 Attention
Transition from Attention to Intention
S3 Intention
Transition from Intention to Action
S4 Action
Disengagement Classifier
Action Classifier
Intention to Act Classifier

Figure 4.13: Action Classifier

4.3.5 Attention Classifier

The output of *Attention Classifier* is simply 1 when all other three classifiers are 0. Equation 4.14 shows binary operations for calculating the output of this classifier.

$$\begin{aligned} AttentionClassifier_t = \;& \neg DisengagementClassifier_t \\ & \wedge \neg IntentionToActClassifier_t \\ & \wedge \neg ActionClassifier_t \\ & t = 1, 2, ..., T \end{aligned} \tag{4.14}$$

4.3.6 Engagement Finite-state Machine

When we are just looking each frame independently, the output of classifier for engagement could be wrong for several reasons.

First of all, it could be hard or impossible to classify engagement state or a gesture that is related to a series of frames. Suppose we want to classify *raising hand* gesture by just looking at a single frame. If we just see the hand position in one frame at the middle of *raising hand*, we may classify it to one of many different gestures such as *putting hand down* or *swiping*.

Secondly, The the joint locations provided by the SDKs are noisy. The joints are even more noisy if an occlusion exists.

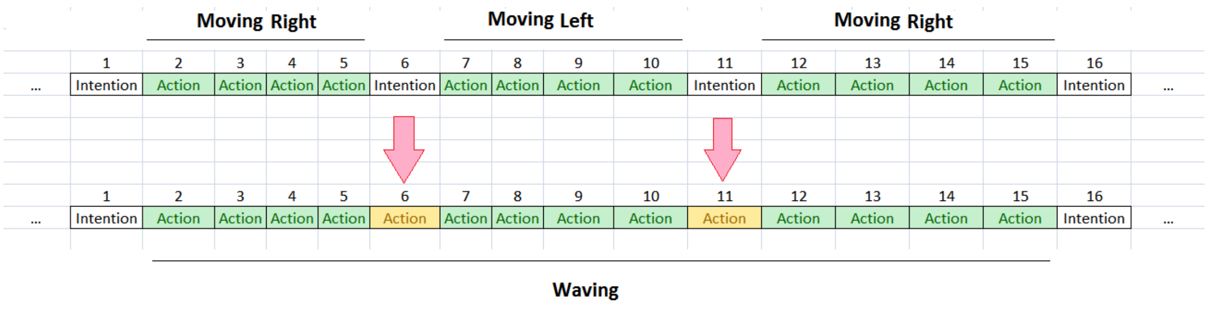
In addition, presence of some body postures that means engagement should be meaningful. A user may show some postures that interpreted as intention to interact for a short amount of time unintentionally. However, can't figure out if it is a meaningful engagement unless we look at the sequence of frames. For instance, the user may face the target for some milliseconds, but does not have any intentions for interaction.

Finally, our engagement classifiers may classify a frame falsely, as their accuracy rate is not 100%.

Therefore, a series of previous frames should be analyzed together to classify a frame more accurately. On the other hand, engagement state classifiers are memoryless and may report wrong engagement state based on the current biometric properties of human body. For addressing these issues, we have designed an *Engagement Finite-state Machine* that keeps record of engagement states based on some hypotheses:

- *Engagement Finite States*: This property describes the FSM design. It starts with disengagement (Initial State). There should be a chain of conditions for engagement state change. State machines are the description of a life cycle of a system. They can describe the different states of the lifeline, the events which influence it, and what it does when a particular event is detected at a any states as the transition condition for particular state change. They offer the complete specification of the dynamic behavior of the system.

- *Engagement Inertia*: We are analyzing frames in real-time. This means we are classifying about 30 frames per second for engagement state detection and time interval for analyzing each frame is about 33ms. For the purpose of engagement detection based on human body postures, it is a very short amount of time. Human body does not move too fast and intends to stay in previous position. Therefore, it's more likely that user is staying in detected state at last frame. For this purpose, we have designed our *Guards* of FSM. Figure 4.3.6 shows how *Guard* conditions assists in keeping the state. In this figure, the output of engagement classifiers are shown in first row for 16 frames for a waving gesture. Waving gesture in our example consists of 3 continuous actions of hand which are *Moving Right*, *Moving Left*, and another *Moving Right*. Because the speed of hand becomes close to zero and classified as *Stopped* with our *HandSpeed* binary classifier, the output of engagement classifier becomes *Intention* at each end. However we know this a continuous gesture, therefore, Guard condition should prevent state change from *Action* to *Intention* for these frames. Second row in Figure 4.3.6 shows a segment of waving action that could be sent to our *Hand Gesture Classifier*. We can also concludes engagement states should transit smoothly that means for each state, we have a minimum number of frames.


Moving Right
Moving Left
Moving Right
Waving

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
...	Intention	Action	Action	Action	Action	Intention	Action	Action	Action	Action	Intention	Action	Action	Action	Action	Intention	...

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
...	Intention	Action	Action	Action	Action	Action	Action	Action	Action	Action	Action	Action	Action	Action	Action	Intention	...

Figure 4.14: Waiving hand example that could be misclassified with swiping right or left in first row. Second row shows how FSM helps in segmenting waving action correctly.

Mathematical model for Engagement FSM

It is noted earlier that a finite state machine is a quintuple $(\Sigma, S, s_0, \delta, F)$, where:

- Σ is the input alphabet (a finite non-empty set of symbols).
- S is a finite, non-empty set of states.

- s_0 is the initial state, an element of S.
- δ is the state-transition function: $\delta : S \times \Sigma \rightarrow S$.
- F is the set of final states, a subset of S

In our Engagement FSM, S is set of S_1, S_2, S_3, S_4 such that S_1 is *Disengagement*, S_2 is *Attention*, S_3 is *Intention*, and S_4 is *Action*. s_0 in this FSM is *Disengagement* as we always start our system supposing the user is not engaged with any targets. δ are set of rules called *Transition Conditions*. These set of rules are presented in Table 4.3.6. Because our system could change among all states and there is no final state, F is empty set in our *Engagement Finite-state Machine*.

Figure 4.15 presents *Engagement Finite-state Machine* and its transition conditions. A state is a description of the mental state or engagement of the user that is anticipated to change over time. A transition is initialized by a change in condition that results in a change of state. In this research, we have modeled engagement states as a finite state Machine with four different states:

- **Disengagement**: User is disengaged from screen or the target object.
- **Attention**: User has attention such as observing the target, but doesn't have any intentions to perform an actions.
- **Intention**: User has intention to do some action, but still not performing it. For example, user is pointing a target for selection.
- **Action**: User is performing an action.

Guard Condition For S_n

Guard conditions are simply for preventing state transition. These guard conditions are defined based on the output of classifiers in previous frames. Suppose we are at state S_n, because the output of engagement classifier is 1 for S_n. If the output of engagement classifier changes for multiple frames, we

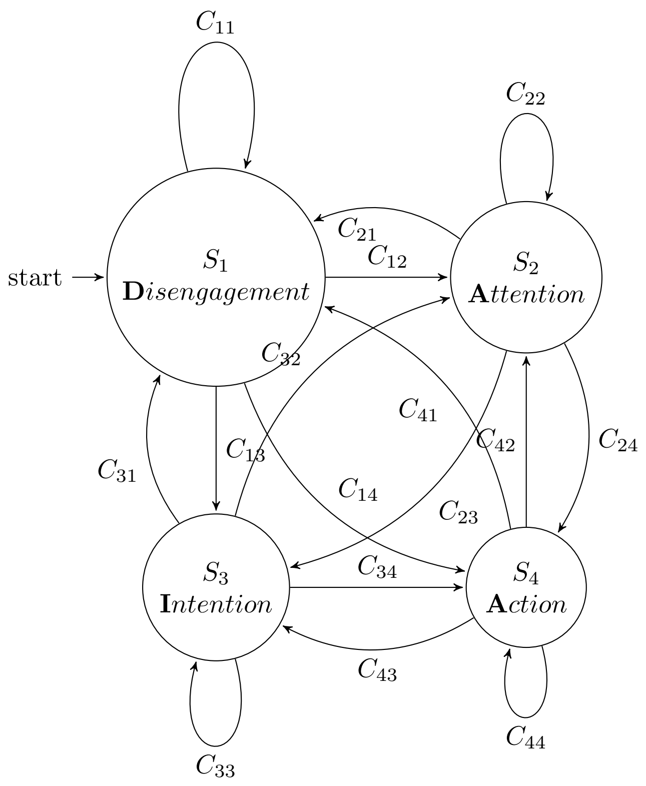


Figure 4.15: Engagement Finite-state Machine

concludes the state also should change. For this purpose, we counted how for many frames the output of engagement classifier is not 1 for S_n as it is shown in equation 4.15. $ClassifierDelay_n^t$ in this equation is the number of consecutive frames that engagement classifier for S_n ($Classifier_n$), is not 1.

$$ClassifierDelay_n^t = \begin{cases} 0 & \text{if } Classifier_n = 1 \\ ClassifierDelay_n^{t-1} + 1 & \text{if } otherwise \end{cases} \tag{4.15}$$

$Guard_n^t$ is defined as guard condition for S_n at frame t. $Guard_n^t$ is 1 if the absence of $Classifier_n$ remains for a predefined number of frames which is called $MaxGrace_n$ for S_n. Based on our experiments, we have set $MaxGrace_n = 10$ for all of states, which means we let 10 frames as grace period for absence of $Classifier_n$.

$$Guard_n^t = \begin{cases} 1 & \text{if } ClassifierDelay_n^t \le MaxGrace_n \\ 0 & \text{if } otherwise \end{cases} \tag{4.16}$$

Transition Condition For S_n

In state S_n at frame t, we use binary operations between output of engagement classifiers and $Guard_n^t$ to decide about state transition as it is shown in equation 4.17. C_{nm}^t in this equation is transition condition from S_n to S_m. This equation indicates if we are in S_n, and $Guard_n$ at frame t is 0 and engagement classifier for State m, S_m is 1, the state transition will occur from S_n to S_m.

$$C_{nm}^t = \begin{cases} Classifier_m \wedge \neg Guard_n^t & \text{if } n \neq m \\ Guard_n^t & \text{if } n = m \end{cases} \tag{4.17}$$

Table 4.3: Transition table

State	S_1	S_2	S_3	S_4
S_1	C_{11}	C_{12}	C_{13}	C_{14}
S_2	C_{21}	C_{22}	C_{23}	C_{24}
S_3	C_{31}	C_{32}	C_{33}	C_{34}
S_4	C_{41}	C_{42}	C_{43}	C_{44}

Table 4.3.6 describes these *Transition Conditions*, C_{nm}, for these state changes. For example, C_{21}, C_{31}, and C_{41} describe *Transition Conditions* from *Attention*, *Intention* or *Action* to *Disengagement*. The interpretation of the equation provided in this section indicates that the state transition from any of these three states to *Disengagement* occurs when their *Guard* conditions are 0 and the engagement classifier is 1 for *Disengagement* classifier. For instance, if we are in *Attention* state, the output of engagement classifier for *Attention* has been absent for several frames and also body direction is not facing target or a special posture *hand folded* exists which indicates disengagement. Similar interpretations for other state changes could be written.

4.3.7 Complexity Analysis

The complexity of our algorithm is mostly bounded by the number of binary classifiers we design for *Engagement Feature Vector* creation. This comes from the fact that binary classifiers are created using matrix operations on frames generated by the software SDK in real-time which each can be done in $O(1)$. Therefore, if we have n binary classifiers, the complexity of our algorithm is $O(n)$. Furthermore, the complexity of FSM calculations is $O(1)$. Therefore, the total complexity of the system is $O(n)$.

4.4 Hand Posture Recognition

In our previous work [134], the global features of hand such as edge contour and convex hull are employed for posture detection. In this research, we use a very basic, yet powerful, global feature of hand shape that allows detecting postures in real-time and with high accuracy by learning a limited number of training samples for each posture. We extend our pixel-wise hand posture classification using Convolutional neural networks.

4.4.1 HandReader Dataset

There are a limited number of datasets for training and testing hand posture recognition. Unfortunately, these datasets are not useful for our purpose since they do not have enough hand posture images with dark background. Hence, we have created a new dataset, i.e., HandReader dataset, for this research. This dataset consists of 500 images from 10 different hand postures. These postures are 10 American Sign Language alphabets. The dataset was created by capturing images of 50 individuals, both males and females, performing the 10 postures in front of a camera. This dataset is publicly available [91]. Figure 4.16shows all the alphabets used in this paper. These are the alphabets A, B, C, D, G, H, L, I, V, and Y from American Sign Language.

Although all images in this dataset have a dark background, the level of illumination differs among the images. Figure 4.17 shows a set of 50 different images of letter V from HandReader dataset.

4.4.2 HandReader2 Dataset

As it is mentioned before, Trieschs static hand posture dataset[33] as a benchmark. Trieschs dataset consists of 10 hand postures (A, B, C, D, G, H, I , L, V, Y) from 24 people with 3 different backgrounds (light, dark, complex) (Figure 4.18). Most real time approaches, after some preprocessing steps convert a hand shape to a 2D binary image called silhouettes, and extract features from these black and white

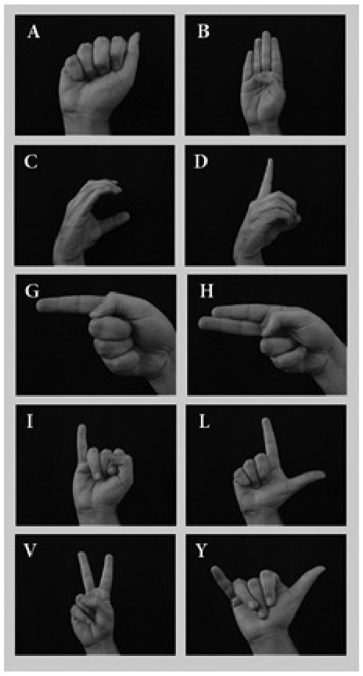


Figure 4.16: Ten different American Sign Language alphabets

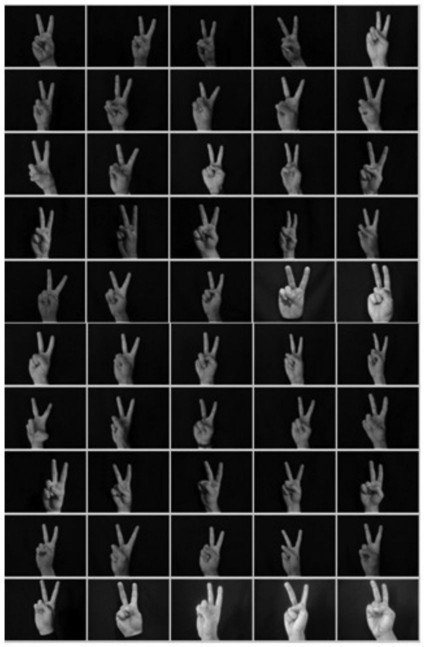


Figure 4.17: A set of 50 different images of V from HandReader dataset.

binary images.

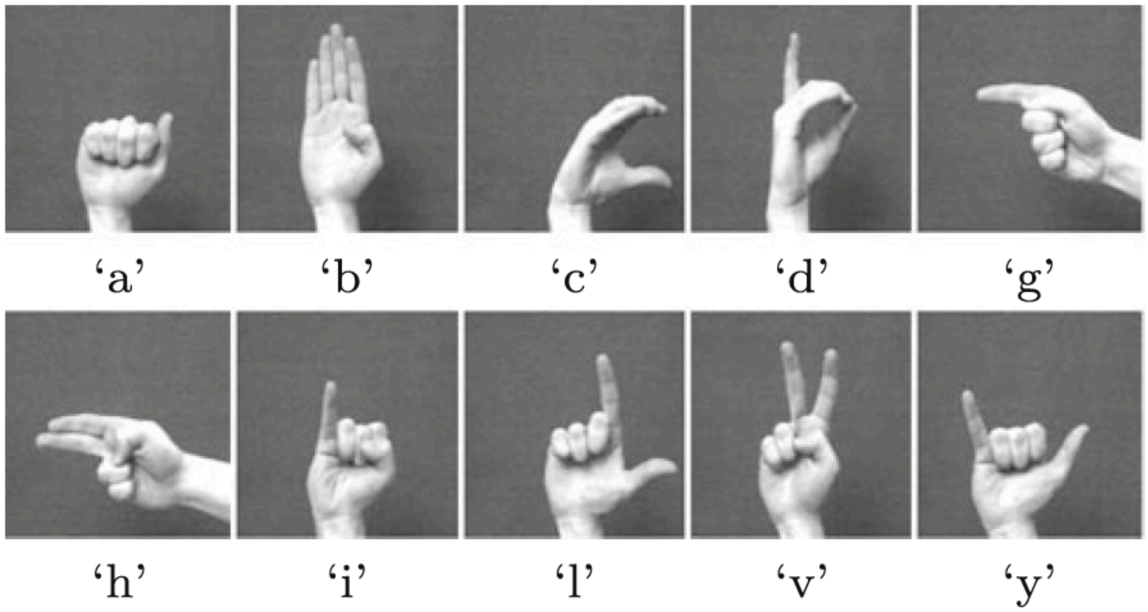


Figure 4.18: Triesch hand posture database from some alphabets of ASL

In addition, as mentioned earlier, after introducing RGB-D sensors, the complex background will not be one of the main challenges for extracting hand from the background. Hence, among all these images, those with dark or light background carry distinct information about the hand shape and posture for different people and are useful for the purpose of hand shape modeling. We chose the images with dark background for extracting the hand shapes. We also extended Trieschs static hand posture dataset by creating HandReader static hand posture dataset. Our dataset consists of 500 images from 10 different hand postures showing 10 different alphabets in American Sign Language. The dataset was created by capturing images from 50 people, both males and females, performing the 10 postures in front of a camera. After some preprocessing steps, we extracted binary images from Trieschs static hand posture dataset with dark background and our own dataset to create a new static hand posture dataset, called HandReader2. Figure 4.19 illustrates 74 different V postures from 74 different people which are used in creation of HandReader2.

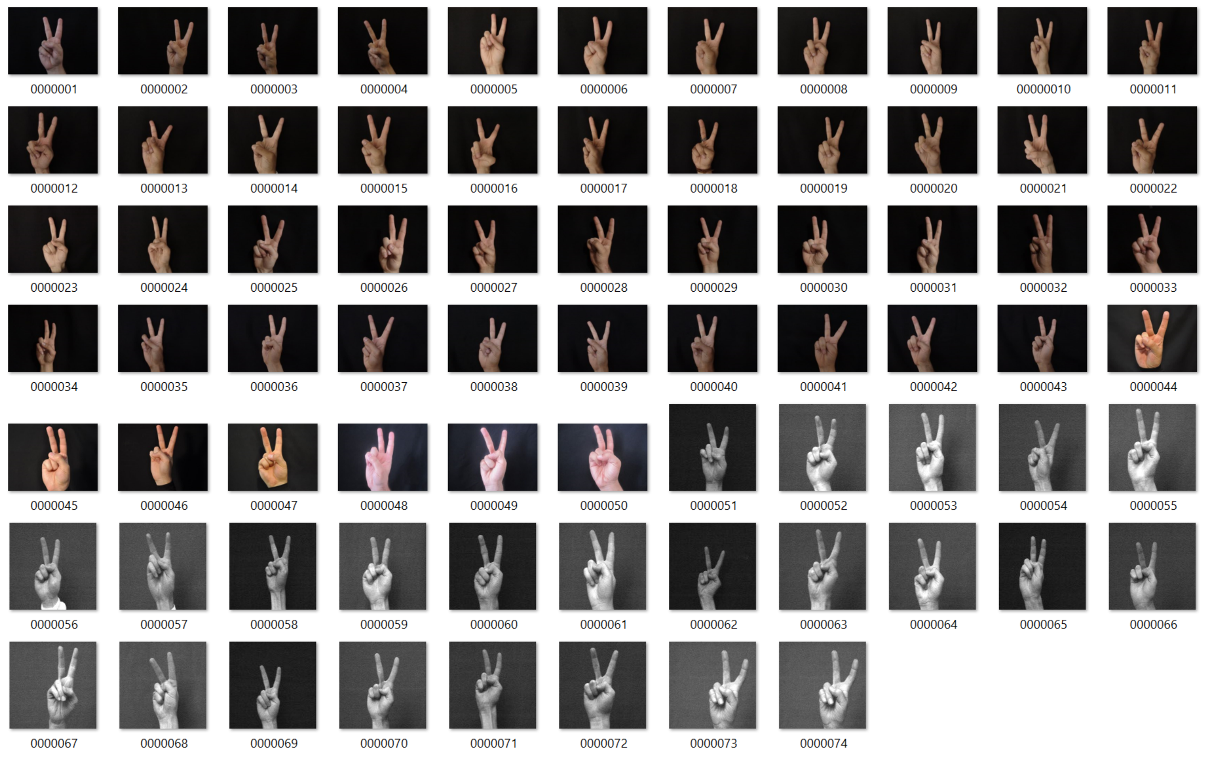


Figure 4.19: HandReader2 is made from 10 RGB hand postures from 74 different people. This figure illustrates 74 different V postures which are used in creation of HandReader2.

HandReader2 is a new dataset of hand-shape silhouettes which contains 128×128 binary square images with the actual shape of hands exactly centered in the squares. This dataset consists of 740 square silhouettes from 10 different hand postures performed by 74 different people. From these 740 images, 500 are extracted from our HandReader dataset and 240 from Trieschs static hand posture dataset. These postures are 10 American Sign Language alphabets (A, B, C, D, G, H, L, I , V, and Y). Creating this dataset allows us to make a more realistic hand shape model based on considerably different hand shapes. HandReader2 is publicly available and can be used by anyone who is interested in hand posture modeling or other related areas.

4.4.3 Preprocessing

After converting all images in the dataset to gray-scale, hand postures should be extracted from the background. Since all images have dark backgrounds, it is possible to subtract the background from the image using image histogram.

Background Subtraction Using Histogram

Suppose histogram of gray-scale image $f(x, y)$ is shown in Figure 4.20 and this image contains a light colored object on a dark background. In this condition, gray-scale pixels of the object and background are separated into two dominant modes. One of the most obvious methods for background subtraction in such conditions is choosing a threshold value, T , which separates these modes. Therefore, if $f(x, y) > T$, then (x, y) are objects pixels; otherwise they are background pixels [135].

In equation 4.18, $f(x, y)$ is gray-scale image and T is the threshold value and the result is a binary image, $g(x, y)$.

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

To find the threshold, T , first the maximum value in the histogram distribution is calculated, because we know a major section of the image is background. Then, the closest point on the x axis of the histogram which is greater than the maximum point and has a value of less than 5% of the maximum value for histogram is chosen as the threshold point. Figure 4.21 shows two samples of finding threshold using histogram.

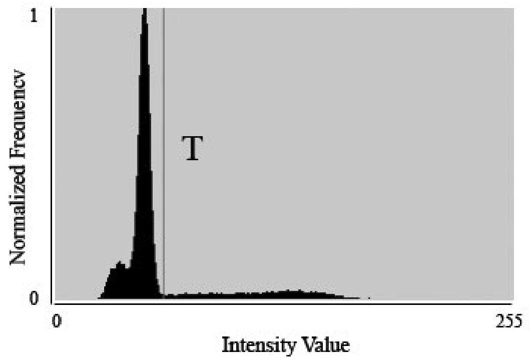


Figure 4.20: Histogram of a gray-scale image which contains a dark background and a light colored object

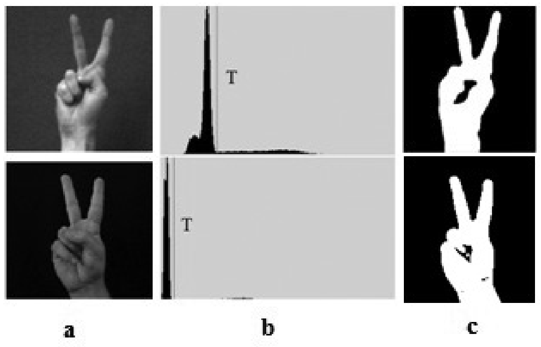


Figure 4.21: Finding Threshold Using Histogram.

Removing Noise and Smoothing Edges

Applying a sharp threshold such as what is done in section 4.4.3 can be quite noisy. It causes a rough edge and maybe some holes in the hand section. To remove holes and some scattered noisy pixels of the background which are remained after applying threshold, a combinations of morphological operations are applied.

A morphological closing operation of A by a structuring element B is obtained by the dilation of A by B, followed by erosion of the result by B as it is shown in equation 4.19:

$$A \bullet B = (A \oplus B) \ominus B, \tag{4.19}$$

where $\oplus$ and $\ominus$ denote dilation and erosion, respectively. For the purpose of this research, a 5×5 pixel rectangle structuring element has been employed. This operation smoothens edges and removes any holes from the structure.

To further smoothen the rough edges, Gaussian blur has been employed. The Gaussian blur is a type of image-blurring filter that uses a Gaussian function to calculate the transformation to be applied to each pixel of the image. The equation of a Gaussian function in one dimension is:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{4.20}$$

In two dimensions, the equation is the product of two such Gaussians, one in each dimension:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.21}$$

Figure 4.22 shows these steps. A Gaussian filter with kernel size of 21×21 and $\sigma = 0.3(n/2 - 1) + 0.8$ which $n = 21$, the size of kernel, works better based on the performance of the system.

4.4.4 Feature Extraction

In section 4.4.3, a binary image for each instance in our dataset has been created. All binary images should be normalized. Figure 4.23 shows the normalization steps. The binary image is shown in Figure 4.23-a. The main idea behind normalization is centering the hand in a square. To find the best size of this square, the bounding box of the main section in the binary image has been found (Figure 4.23-b). Then the width or height of the bounding box, whichever is greater, is selected as the side size of the square of normalized image. Afterwards, we create a black square with this size and copy the main section, which is hand posture, in the center of this square (Figure 4.23-c). To have normalized images of the same size, we scale all normalized images to 128×128 pixels (Figure 4.23-d).

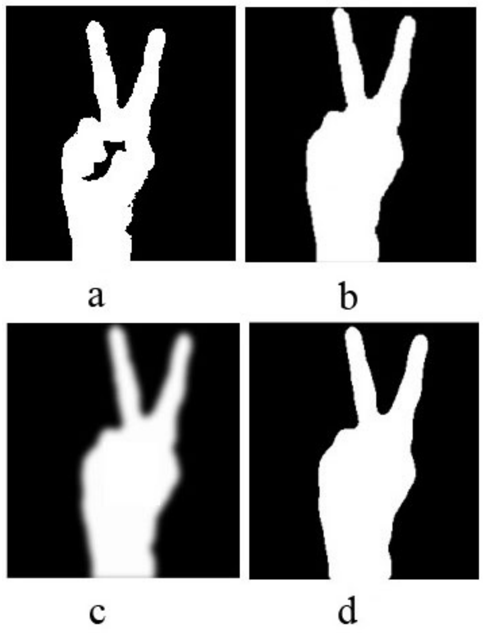


Figure 4.22: Preprocessing steps: (a) Background subtraction (b) Morphological operations (c) Gaussian Blur (d) Binary image by applying a threshold

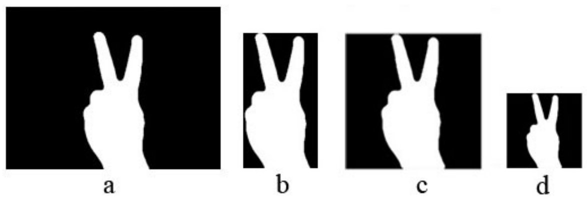


Figure 4.23: Normalization steps: a)Binary image b)Finding bounding box c)Normalization d)Scaling.

Figure 4.24 shows normalized images for V postures.

We have not used any descriptor for extracting features from hand posture images. Instead of using a descriptor, we have just converted $2D$ matrix of our normalized image to vector and used it as our feature vector.

On the other hand, we converted the matrix of image, which is a 128×128 square matrix to 1×16384 vector by putting rows in the vector according to their order in matrix as it is shown in Figure 4.25.

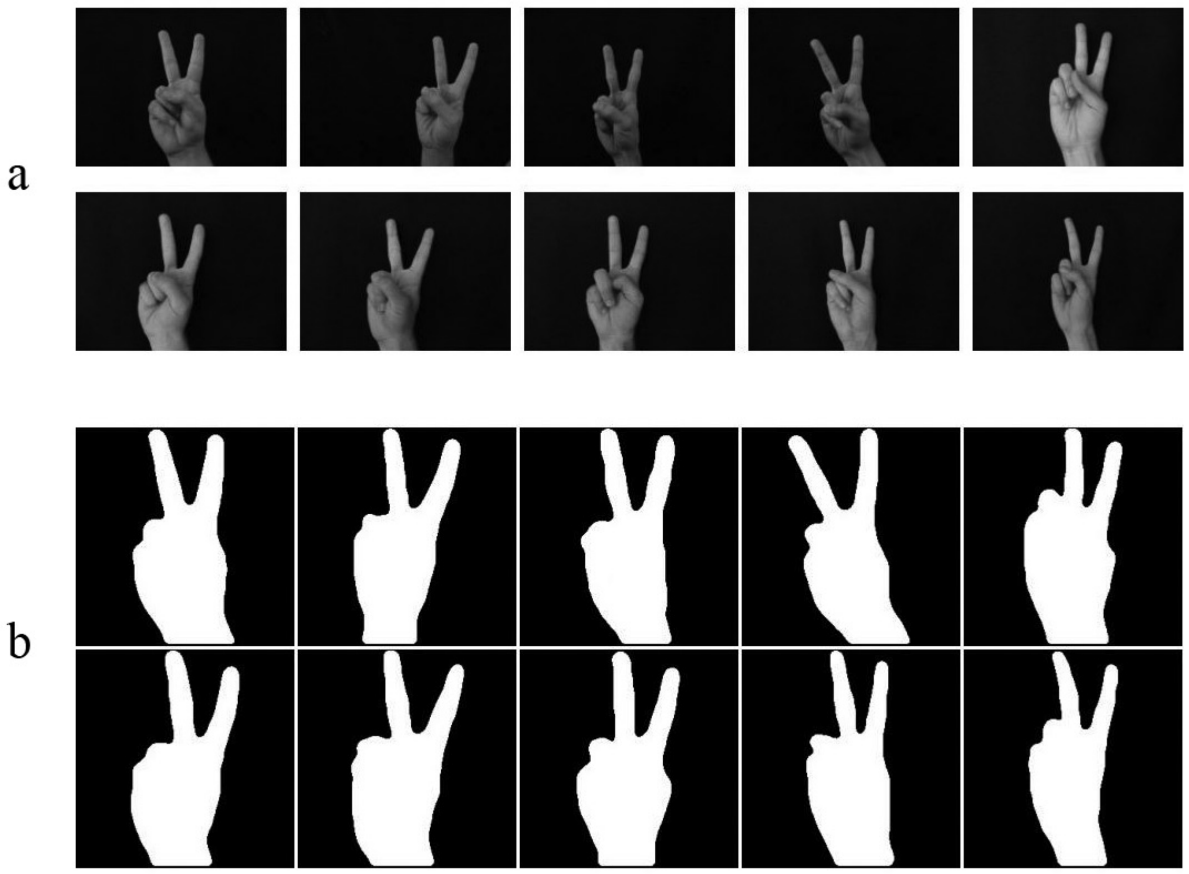


Figure 4.24: Normalized images for V postures: a) Main postures b) Normalized postures.

4.4.5 Dimension Reduction Using PCA

The 1×16384 feature vector is huge. It can be reduced using the PCA, which is one of the most common dimension reduction methods. PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data lies on the first coordinate, called the first principal component, and the second greatest variance on the second coordinate, and so on [105]. Suppose we have a data matrix X^T, with zero empirical mean; i.e., the mean of the distribution has been subtracted from the data set. In this matrix, each of the n rows represents a different repetition of the experiment, and each of the m columns gives a particular kind of datum. The singular value decomposition of X is $X = W\Sigma V^T$, where the $m \times m$ matrix

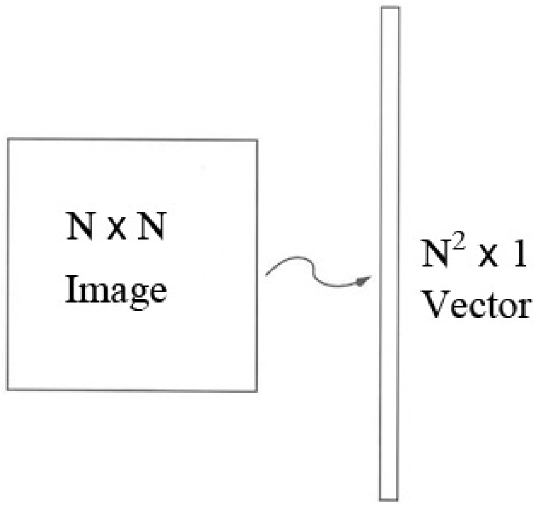


Figure 4.25: Converting an image to a vector

W is the matrix of eigenvectors of the covariance matrix XX^T, the matrix Σ is an $m \times n$ rectangular diagonal matrix with nonnegative real numbers on the diagonal, and the $n \times n$ matrix V is the matrix of eigenvectors of X^TX. The PCA transformation that preserves dimensionality is then given by:

$$Y^T = X^TW = V\Sigma^TW^TW = V\Sigma^T \tag{4.22}$$

For dimension reduction, X should be projected down into the reduced space defined by only the first L singular vectors, W_L: $Y = W_L^TX = \Sigma_LV^T$ where $\Sigma_L = I_{L\times m}\Sigma$ with $I_{L\times m}$ the $L\times m$ rectangular identity matrix. If there is a set of points in Euclidean space, the first principal component corresponds to a line that passes through the multidimensional mean and minimizes the sum of squares of the distances of the points from the line. The second principal component corresponds to the same concept after all correlation with the first principal component has been subtracted from the points.

As mentioned earlier, the PCA transformation preserves dimensionality. This means the original data can be reconstructed if we have the mean of the distribution and its PCA transformation. In general, we should maintain all principal components in PCA transformation to be able to reconstruct the original data. However, when the size of feature vectors are greater than the number of feature vectors used for training the system, the best principal components for reduction is the number of feature vectors used for training the system minus One. Sections 4.4.5 and 4.4.5 clarify the reason.

The PCA Subspace

The direction of maximum separation is the first principal component of a dataset. The direction with the next largest separation, which is perpendicular to the first principal component is the second principal component. In a dataset of $2D$ vectors, at most two principal components exist. The dimensionality is much higher for images. For example, in a dataset of 128×128 images, 1×16384 vectors exist. Therefore, there are more principal components in a dataset made up of images. However, the number of principal components we can find is also limited by the number of data points. Consider a dataset consisting of just one point. Since there is nothing to separate from, there is not a direction of maximum separation for this dataset. Now, suppose a dataset with just two points. The line connecting these two points is the first principal component, and there is no second principal component because there is nothing more to separate from. In this situation, both points are fully on the line.

This idea can be extended indefinitely. Three points define a plane, which is a 2D object. Therefore, a dataset with three data points can never have more than two principal components, even if it consists of 1×16384 vectors.

Computing the eigenvectors

Performing PCA directly on the covariance matrix of images is often computationally expensive. For example, if 128×128 gray-scale images are used, each image is a point in a 16384-dimensional space and the covariance matrix is a matrix of 16384×16384 elements. However the rank of the covariance matrix is limited by the number of training examples. This means if there are N training examples, there will be at most $N - 1$ eigenvectors with non-zero eigenvalues. Turk and Pentland[104] showed that if the number of training examples is smaller than the dimensionality of the training images, the principal components can be computed as follows.

Let T be the matrix of preprocessed training images, where each column contains one mean-subtracted image. The covariance matrix can be computed as $S = TT^T$ and the eigenvector decomposition of S will be:

$$Sv_i = \mathbf{T}\mathbf{T}^T\mathbf{v}_i = \lambda_i\mathbf{v}_i \tag{4.23}$$

However TT^T is a large matrix, and if instead we take the eigenvalue decomposition of

$$\mathbf{T}^T\mathbf{T}\mathbf{u}_i = \lambda_i\mathbf{u}_i \tag{4.24}$$

then we notice that by pre-multiplying both sides of the equation with T, we will obtain:

$$\mathbf{T}\mathbf{T}^T\mathbf{T}\mathbf{u}_i = \lambda_i\mathbf{T}\mathbf{u}_i \tag{4.25}$$

It means if u_i is an eigenvector of TT^T, then $v_i = Tu_i$ is an eigenvector of S. In our dataset, each 128×128 posture image is treated as one data point in a 16384 dimensional space. Therefore, the number of principal components which could be found will never be more than the number of posture images for training minus one. Since our classifier is using 10 images from 5 different posture classes, fifty 1×16384 feature vectors exist. Thus, the best principal components for reducing the dimensionality is $50 - 1$ or 49.

4.4.6 Deep Learning-based Approach

Convolutional Neural Networks requires a large dataset for training the network. In HandReader2 we just have 740 samples for each 10 different posture classes. Furthermore, a posture classifier which is rotation sensitive but insensitive to small deviations is desired. For this purpose, we rotated each sample in HandReader2 45 times from -22 to 22 degrees. We call this new dataset the *Extended HandReader2* dataset. Figure 4.26 shows how a V sample in *Extended HandReader2* is rotated 45 times.

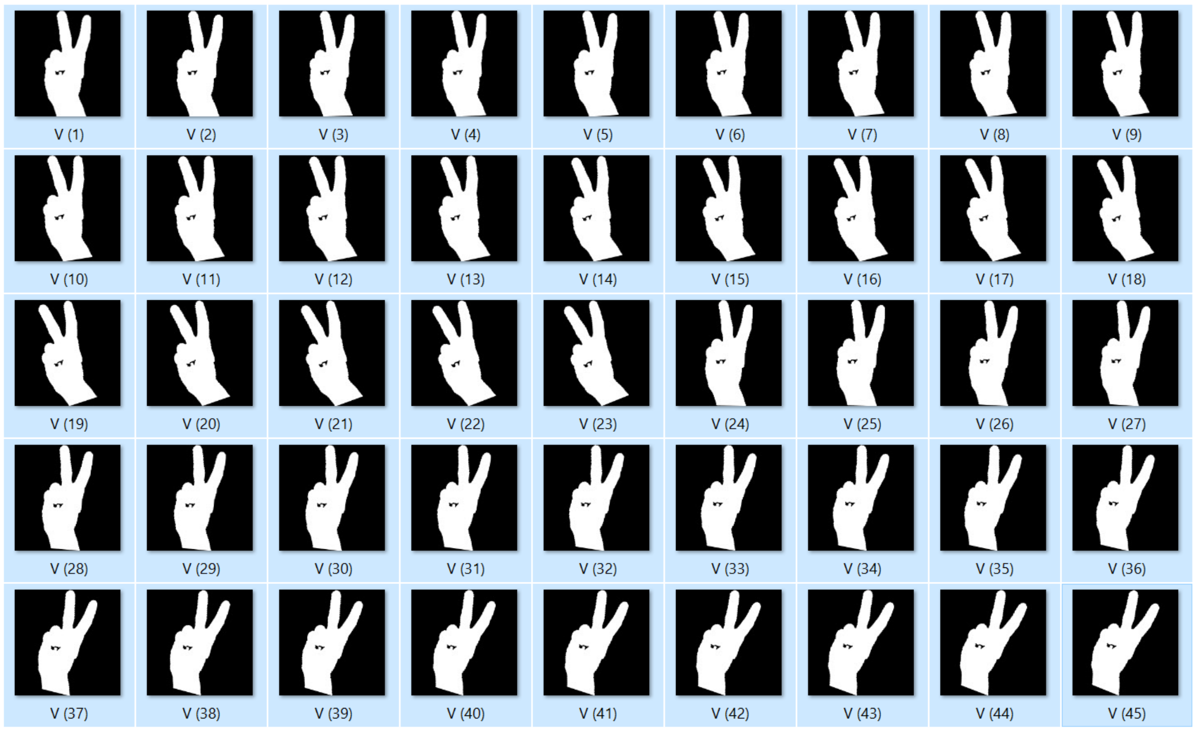


Figure 4.26: 45 degrees rotation from -22 to 22 for a V sample in Extended HandReader2 dataset

By rotating each 740 samples in HandReader2 dataset for 45 times, we have created a new dataset for train and validation which consists of 3330 samples for each 10 postures.

We used 3-fold cross validation for comparing the classifiers. In this research, we have used LeNet and GoogleNet networks for training and validation.

For training, we used about two-third of data and for validation one-third is used. The training and validation repeated 3 times as it is shown in Figure 4.27 for calculating the average accuracy of each method.

It should be noted in each split of data, for training and validation, the subjects of hand postures for training and validation are different. On the other hand, the hand postures for training are from different people rather than validation. Further details about network parameters and average accuracy of each

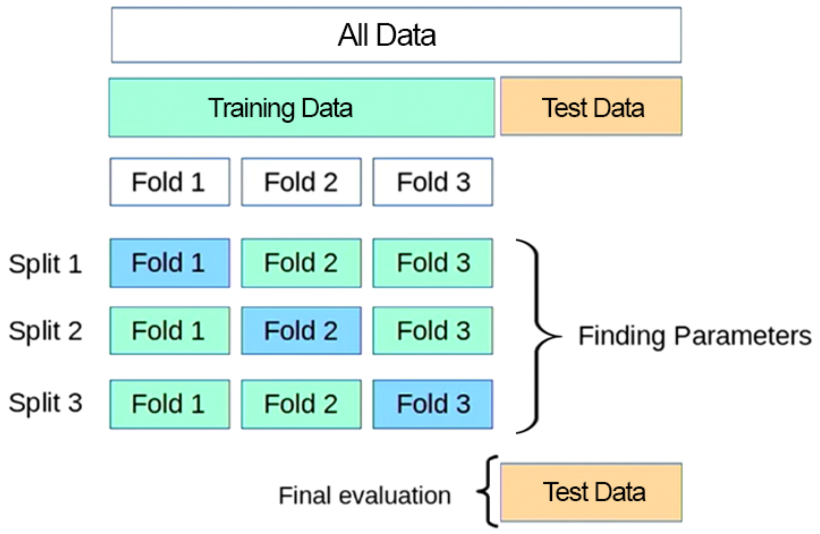


Figure 4.27: 3-fold cross validation

method is presented in experimental results section. Figure 4.28 depicts LeNet architecture adopted for hand posture classification.

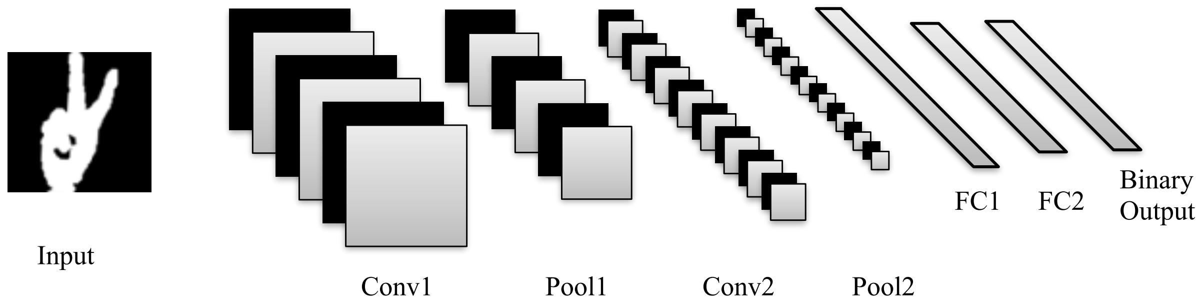


Figure 4.28: LeNet architecture adopted for hand posture classification

4.5 Summary

In the previous chapters, we introduced the general steps of the proposed engagement detection for hand posture and gesture recognition technique. In this chapter, these steps are explained extensively.

Chapter 5

Experimental Results

5.1 Experimental Results for DAIA

DAIA framework was implemented in Visual C++ on Windows workstation.We used ASUS Xtion Pro to capture depth images and track skeletons using Primesense OpenNI and NiTE SDK and OpenGL. Finite State Machine is implemented using Boost Meta State Machine (MSM) Library. The system can process each classify each frame in less than 10ms. Therefore, the method can be used in real-time applications.

5.1.1 Standard Setup For Training and Test

During the train, test and data collection, subjects were standing in front of the screen shown in Figure 5.1.2. 3D skeleton data and was captured by the depth sensor on top center of the screen as it is shown.

5.1.2 Test Data Collection

Thirty different subjects (12 females, 18 males) with engineering background are asked to hear random order of commands from a list of actions such as "raising hand" or "swiping right to left from A to B" and perform them in front of a screen depicted in Figure 5.1.2. In this figure, we have five blue points with numbers 1 to 5 and four red points with letters A, B, C and D. The epicure shown in Figure 5.1.2 was in front of the subjects and they heard and performed the commands individually. The depth sensor which recorded the frames was located on the top center of the screen.

Activity Commands

There are 25 different activity commands based on the signs in Figure 5.1.2. We wrote a small program to randomly select several commands from the set of commands and generate a voice media file reading those commands. List of random commands was generated using a text to speech program that converts our written commands to voice. Subjects heard these commands using a media player and ear-bud. An

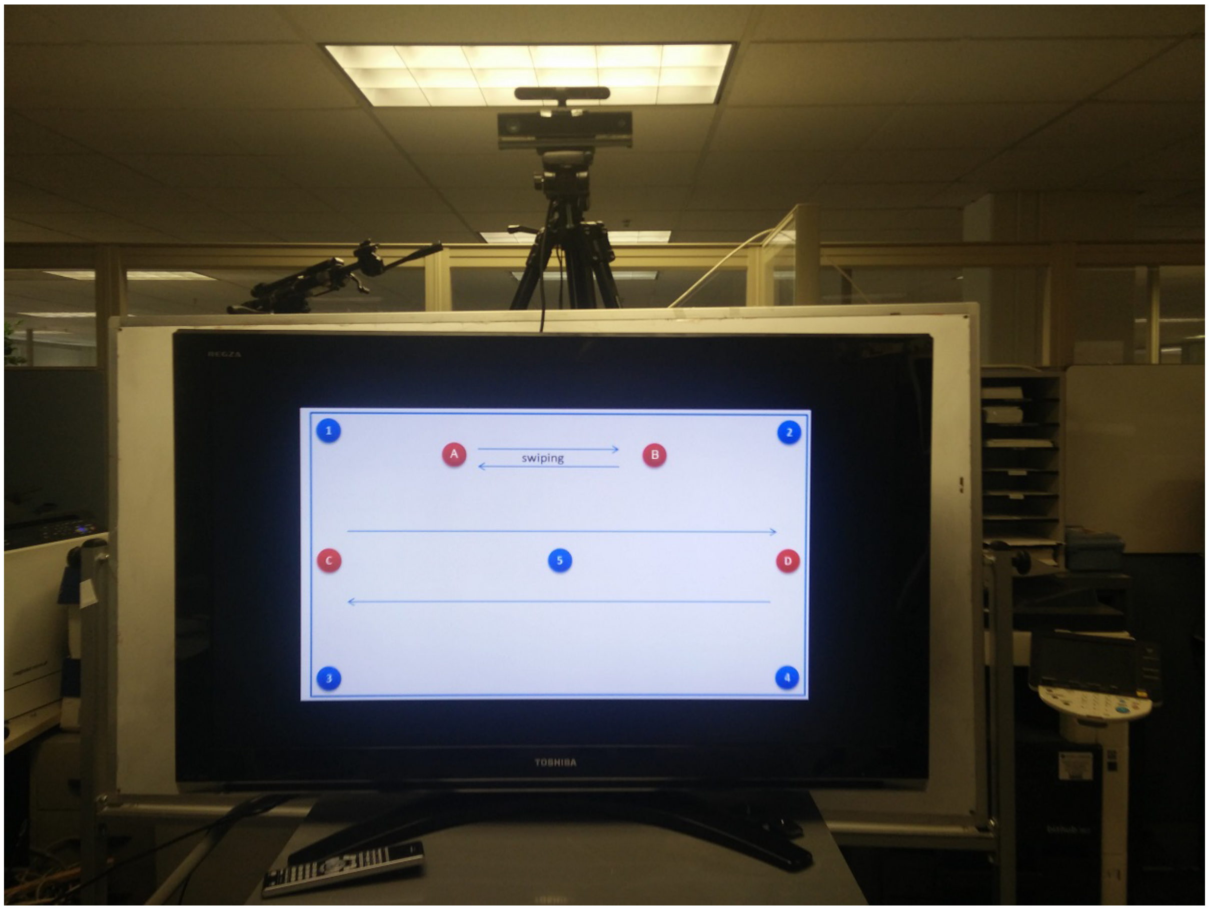


Figure 5.1: The screen is used to guide users in performing actions in front of the depth sensor

example some voice media commands which is generated for one of the subjects includes the following commands:

- Please lean back relax and look at ceiling
- Please point to number 1
- Please push button A
- Please put your hand under chin
- Please raise your hand to ask a question

- Please swipe right from A to B
- Please take some pages from desk and read them
- Please turn chair and look at the person besides you for some seconds
- Please point number 1 and continue pointing to number 4
- Please point number 1 and continue pointing to number 5
- Please stretch both your hands above your head

5.1.3 Test Results

Figure 5.2 shows two different activities that user performed in front of the depth sensor. Figure 5.2.a illustrates the trace of raising hand and 5.2.b depicts swiping activity.

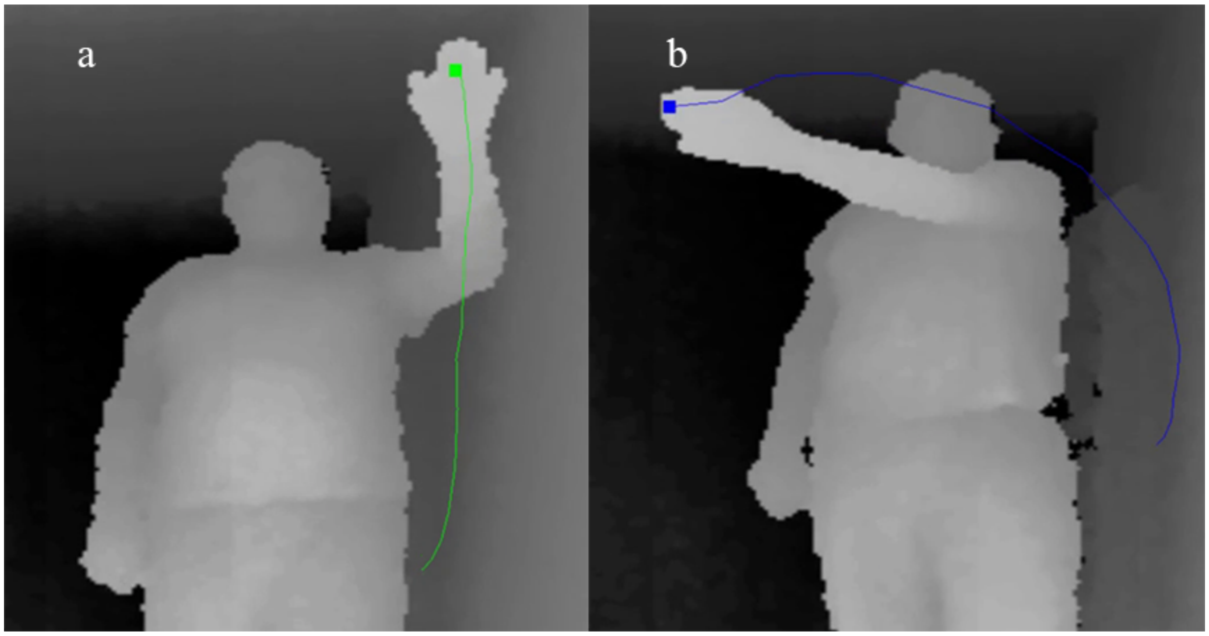


Figure 5.2: Two different example activities: a) Raising hand b) Swiping

An example of state change using *Engagement FSM* is shown in Figure 5.3. Graphs in 5.3 are for the activity of raising and putting down hand during 20 seconds or about 600 frames.

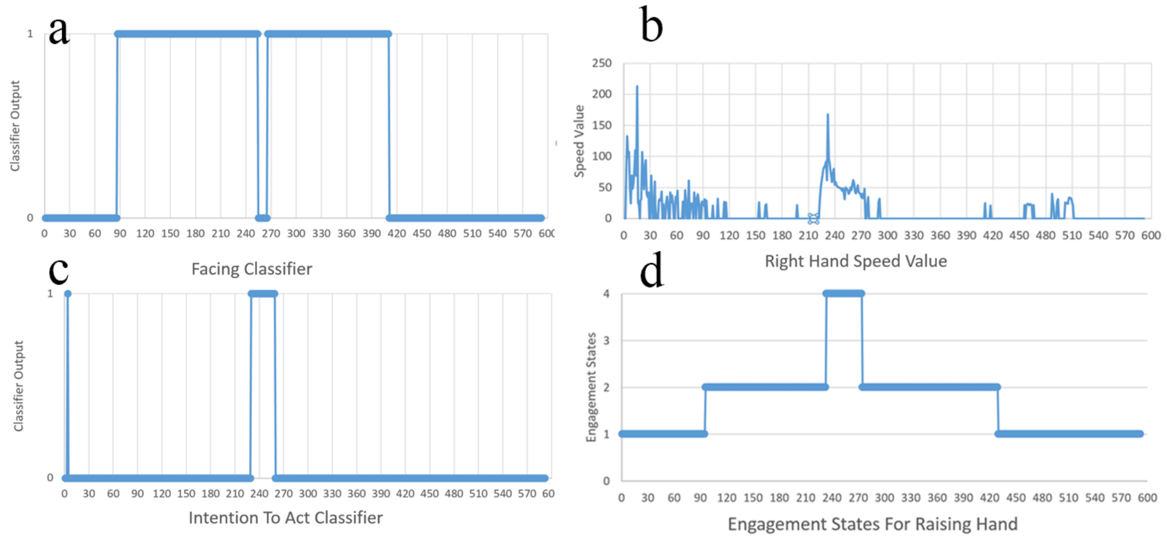


Figure 5.3: Engagement state detection using FSM for raising and putting down right hand in 600 frames: a) Facing classifier b) Hand speed value c) Intention to Act Classifier d) Engagement states for raising and putting down hand

Table 5.1: Performance of FSM

State	FSM Performance
Disengagement	97.3%
Attention	87.2%
Intention	90.8%
Action	94.2%
Average	**92.3%**

Each recording consists of about three minutes of activities that goes through all the engagement states. The results are gathered in Table 5.1.

5.1.4 Comparison and Discussions

Schwarz et al. [19] used a linear regression approach to calculate weight factors to evaluate the relative importance of five binary classifiers and defined a threshold for the sum of the classifiers to evaluate intention to act. Their statistical approach engagement detection is not clear.

We propose the implementation of the classifiers as a feature vector containing the binary values.

Depending on the available sensors the feature vector consists of features from different modules. The feature vector is fed to a Support Vector Machine with a linear kernel as a machine learning algorithm to evaluate the important classifiers and cluster the emerging pattern in the Engagement Framework states. Our approach is based on LIBSVM. LIBSVM implements the Sequential minimal optimization (SMO) algorithm for kernelized SVMs, supporting classification and regression.

As it is mentioned in Table 4.1 we created 37 binary classifiers. For each engagement metrics, multiple binary classifiers are designed. The 0 or 1 output of these 37 classifiers are used to make our feature vector, G, for intention to action classifier. Furthermore, we need to define W or vector of weights to calculate engagement score and afterwards we should define a threshold to classify the frame as intention to act or disengagement similar to the procedure proposed in [19]. It needs extensive research on different body postures to calculate these weights. Furthermore, putting constant weight values for different classifiers may result in wrong classification for complex body postures. Therefore, instead of defining constant values for the weight vector, we used SVM[116] with linear kernel for training our intention to act classification. We used G as the feature vector for training and testing our SVM.

5.2 Hand Posture Recognition

5.2.1 SVM-based results

For testing and evaluating our system, SVM and k-NN classifier with different configurations have been employed. We also trained and tested the system when PCA is not used. k-NN works fine with low numbers of data-point, but with a huge number of data-points it starts to slow down. Applying PCA has no effect on the results of k-NN classifier. Experiments show that $k = 5$ is the best value for this problem. Figure 5.4 shows this result. SVM is a better alternative, which yields better performance in

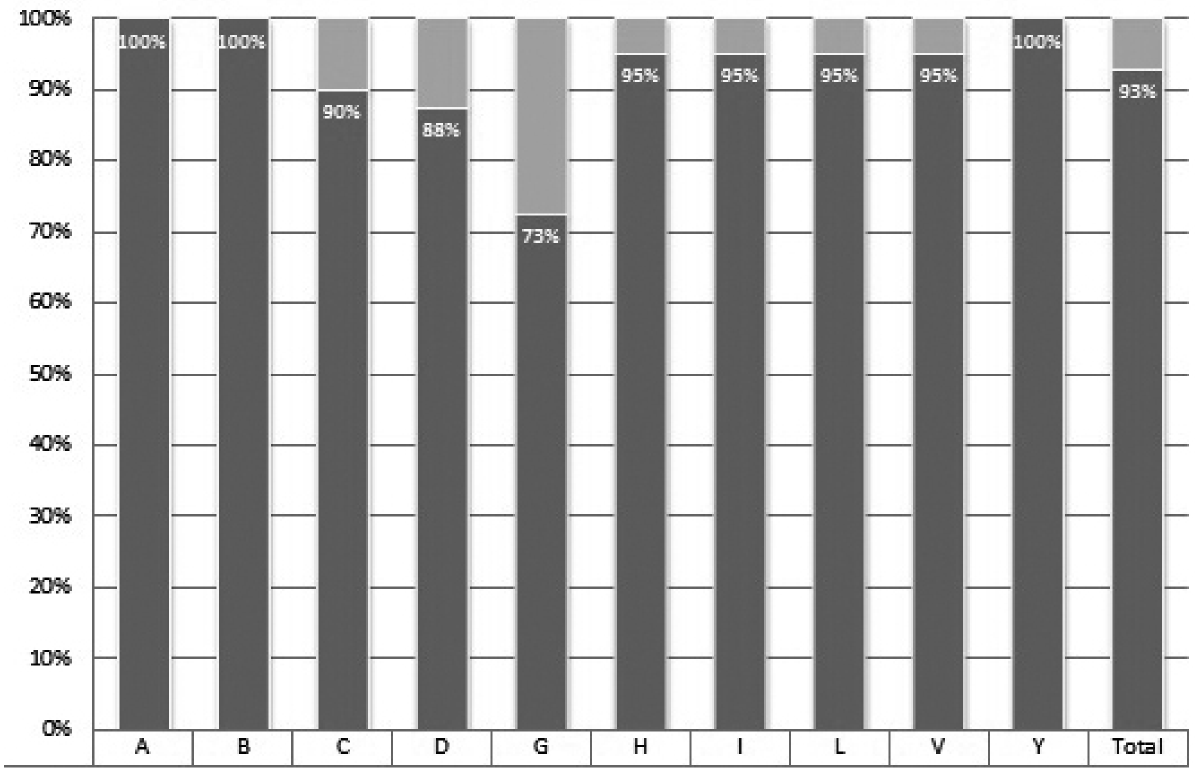


Figure 5.4: Results of classification with k-NN ($k = 5$) classifier and with or without using PCA

classification time and also accuracy. When PCA is employed, RBF kernel can not classify the postures correctly. Figure 5.5 shows this effect. When PCA is not employed, SVM classifier with RBF kernel accomplish similar results in comparison with linear kernel. But the RBF kernel requires the optimization

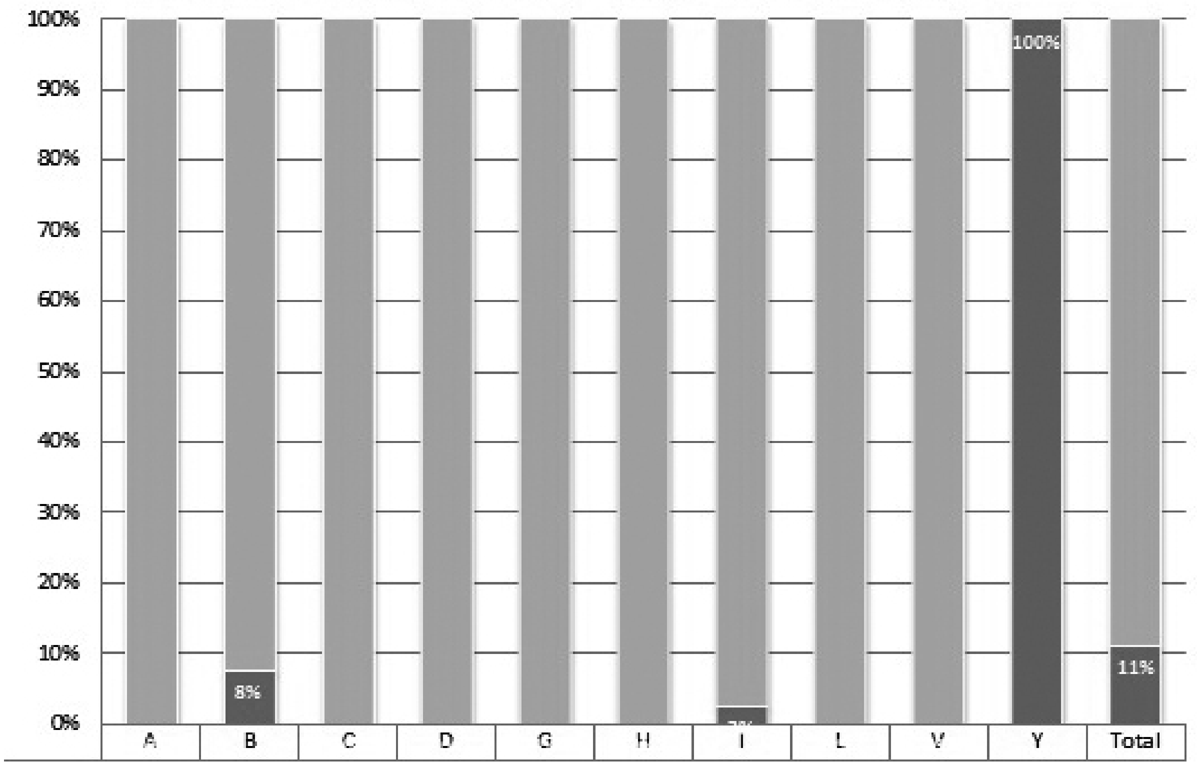


Figure 5.5: Results of classification using SVM with RBF Kernel and PCA

of its parameters (c, γ) . These extra parameters require more work during the training phase, so the linear kernel is preferred. In our experiments, optimum values for theses parameters are $c = 1.5^6$ and $\gamma = 1.5^{-5}$. Figure 5.6 shows results of classification using SVM with RBF or linear Kernel and without using PCA. Using SVM with linear kernel after applying PCA yields best results for classification of our postures. Figure 5.7 shows results of classification using SVM with linear Kernel and PCA.

For training and testing the system, multi-class Support Vector Machine (SVM) with linear kernel produces best results. Table 5.2 summerises our experimental results. Our dataset for this test is HandReader2 which consists of 740 samples. 30% of images are used for training and the rest are used for test. We have used 3-fold cross validation approach and Table 5.2 shows the average accuracy in our experiment for each method. Best results are produced when we are using SVM classifier with linear kernel after applying PCA. Maximum of error occurred for letter G and H. It is because letter G and

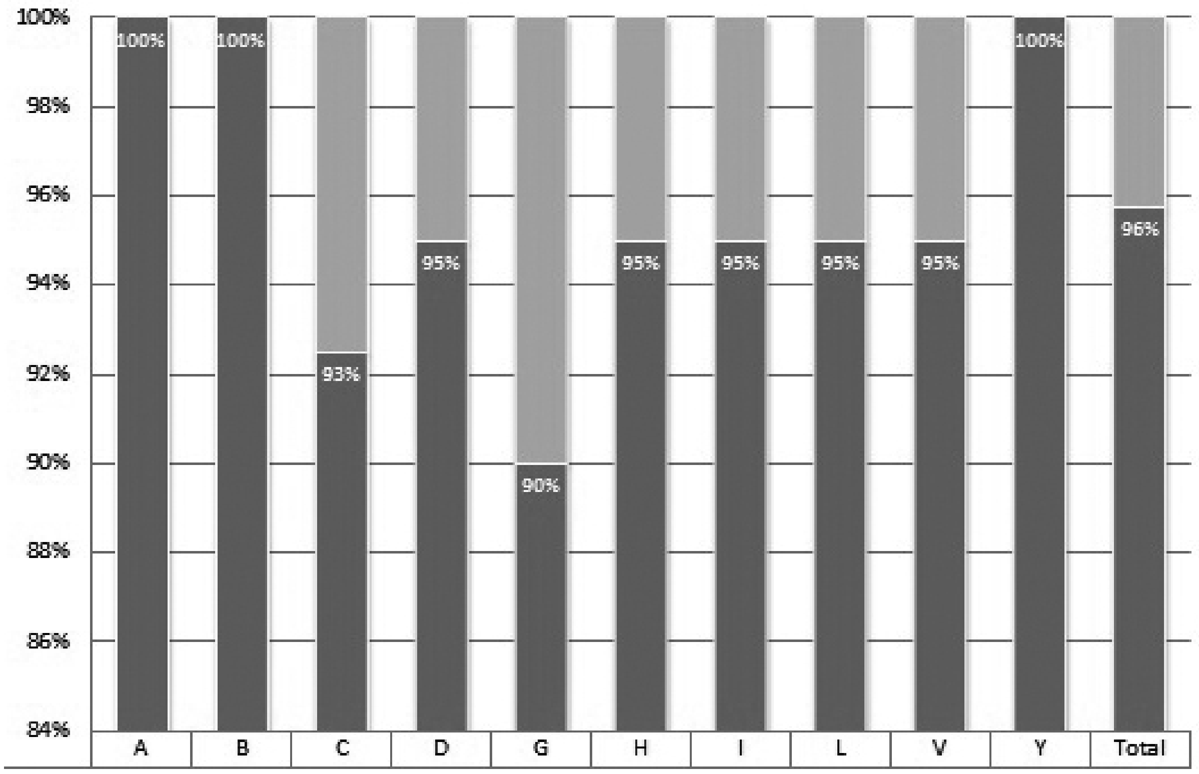


Figure 5.6: Results of classification using SVM with RBF or linear Kernel and without PCA

H in American Sign Language are very close to each other as it is shown in Figure 5.8. Hence, for improving the performance, some methods should be used to extract more discriminating features. In our future work, we will use contour features for extracting these features.

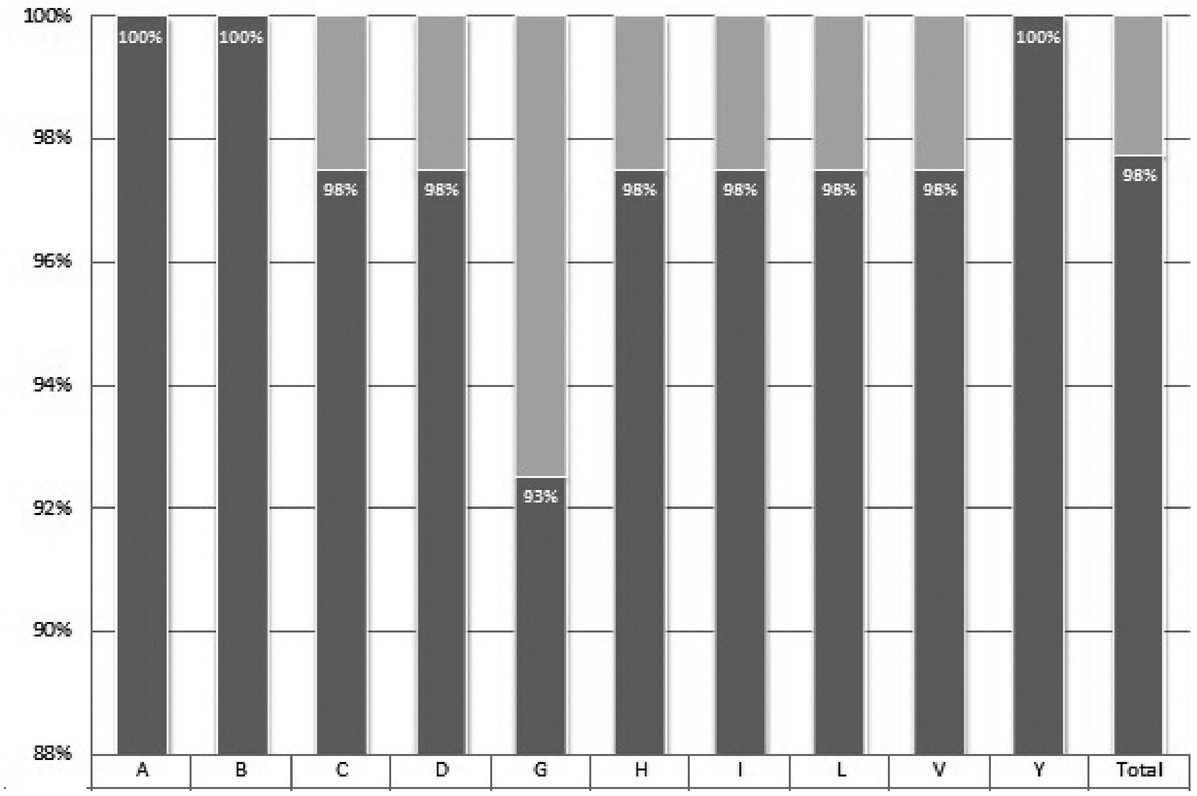


Figure 5.7: Results of classification using SVM with linear Kernel and PCA

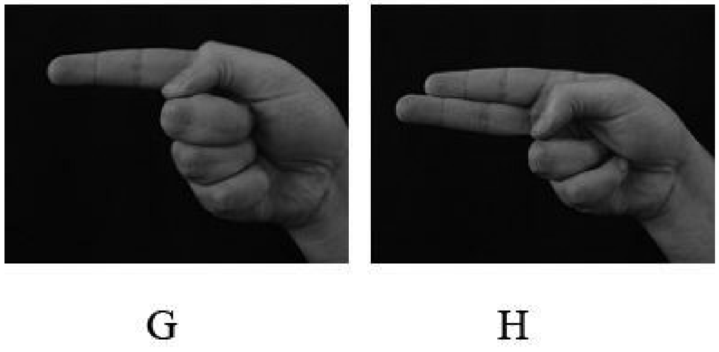


Figure 5.8: letter G and H in American Sign Language.

5.2.2 Deep Learning-based Results

As it is presented in section 4.4.6 we used rotation matrix to generate 45 rotated samples for each sample. This step produced a total number of 33,300 images, with 3,300 belonging to the each of 10 posture

Table 5.2: Average 3-fold cross validation detection accuracies with different classifiers and configurations.

Classifier	PCA	Detection Rate
k-NN ($k = 5$)	no	93.0%
k-NN ($k = 5$)	yes	93.0%
SVM RBF	no	95.75%
SVM RBF	yes	11%
SVM Linear	no	95.75%
SVM Linear	yes	**98.25%**

classes in our HandReader2 dataset, called the *Extended HandReader2* dataset. The data were next converted to the LMDB format and resized to 28×28 pixels. The adopted LeNet model was set for 100 epochs and initiated for Stochastic Gradient Descent with $\gamma = 0.1$, *momentum* $= 0.9$, *base learning rate* $= 0.01$, and *weight decay* $= 0.0005$, and a step learning rate policy dropping the learning rate in steps in each 1/3 of the number epochs by factor of $\gamma = 0.1$ as it is depicted in Figure 5.9.

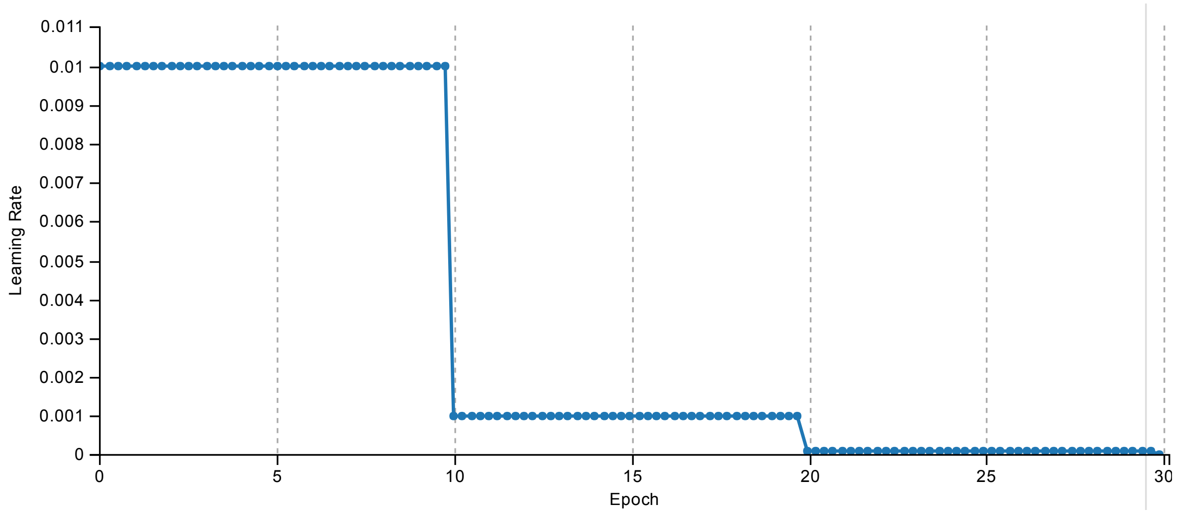


Figure 5.9: Learning rate drops in each 1/3 of the number epochs by factor of $\gamma = 0.1$.

Next, the model was trained and validated by 67% and 33% of the data for Extended HandReader2 dataset. The training and verifying processes were repeated three times based on 3-fold cross validation on Amazon AWS Linux g2.2xlarge to ensure the robustness of the network and achieved accuracy. The average of accuracies was obtained for each experiment separately, as shown in Table 5.3.

Table 5.3: The accuracy of verification datasets is demonstrated below for 10800 test data. As it is shown, a very high level of accuracy in verification datasets was achieved for both LeNet and GoogleNet on Extended HandReader2 dataset

		Accuracy of Verification (%)			
Dataset	Architecture	1	2	3	Average
Extended HandReader2	Adopted LeNet	96.57	96.59	96.66	96.61
Extended HandReader2	Adopted GoogleNet	98.35	98.36	98.35	98.35

The results demonstrate that a high level of accuracy was achieved in all of the experiments, with the average accuracy rate of 96.6102% achieved based on adopted LeNet on Extended HandReader2 dataset as it is shown in Figure 5.10. The accuracy level converged to its final value around epoch 10 and more training epochs based on this network did not increase the accuracy. Therefore, for achieving a higher accuracy we have also employed GoogleNet architecture. The confusion matrix for LeNet is presented in Figure 5.11.

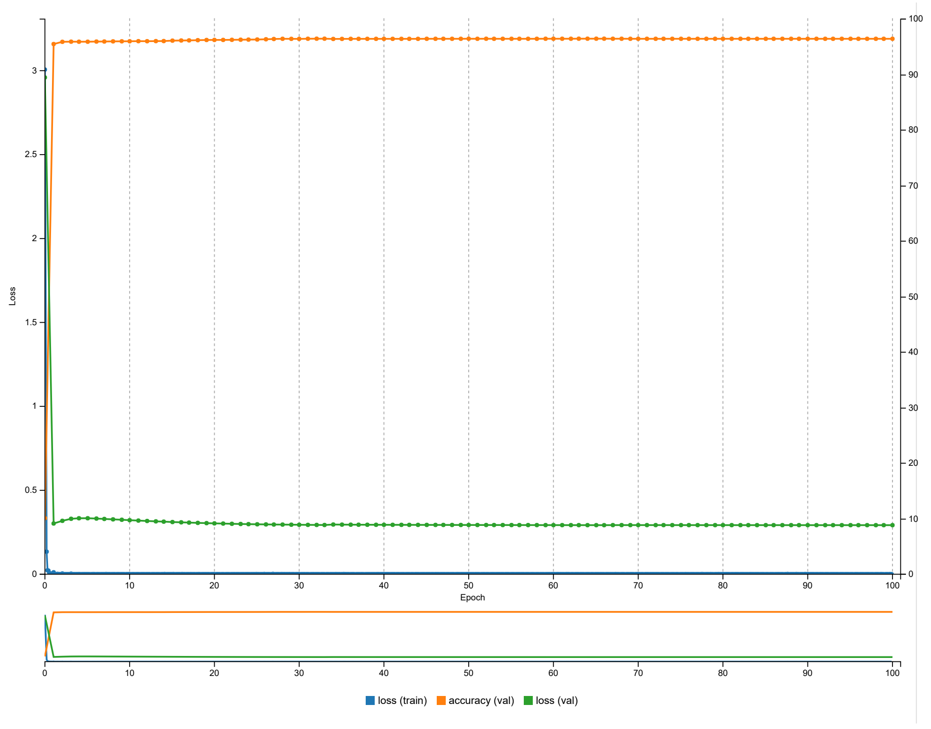
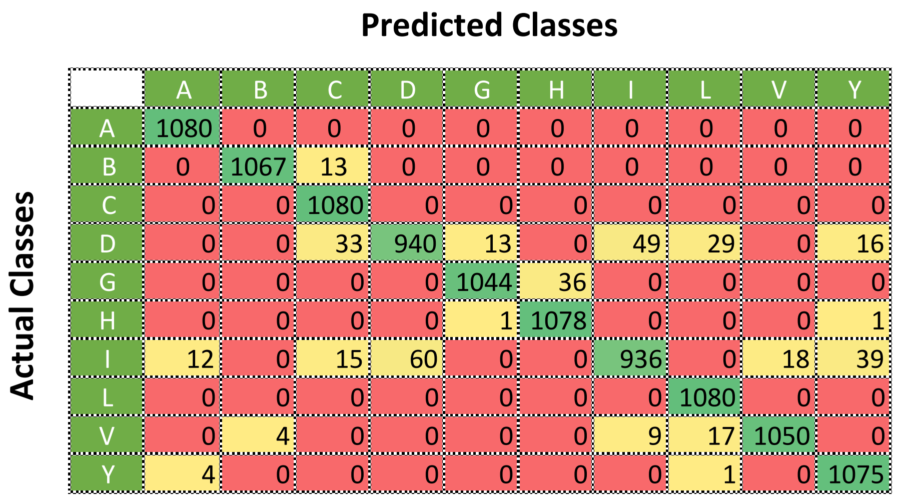


Figure 5.10: Accuracy results after 30 epochs for GoogleNet

For GoogleNet training and verification, the preprocessed datasets were converted to LMDB format



Predicted Classes

Actual Classes

	A	B	C	D	G	H	I	L	V	Y
A	1080	0	0	0	0	0	0	0	0	0
B	0	1067	13	0	0	0	0	0	0	0
C	0	0	1080	0	0	0	0	0	0	0
D	0	0	33	940	13	0	49	29	0	16
G	0	0	0	0	1044	36	0	0	0	0
H	0	0	0	0	1	1078	0	0	0	1
I	12	0	15	60	0	0	936	0	18	39
L	0	0	0	0	0	0	0	1080	0	0
V	0	4	0	0	0	0	9	17	1050	0
Y	4	0	0	0	0	0	0	1	0	1075

Figure 5.11: Confusion matrix for LeNet

and resized to 224×224. The model was adjusted for 30 epochs using *Stochastic Gradient Descent* with $\gamma = 0.1$, $momentum = 0.9$, $base\ learning\ rate = 0.01$, and a step learning rate policy.

The GoogleNet model resulted in a higher level of accuracy than the LeNet model, with the highest average accuracy rate of 98.3531% achieved for Extended HandReader2 dataset. The confusion matrix for GoogleNet is presented in Figure 5.14.

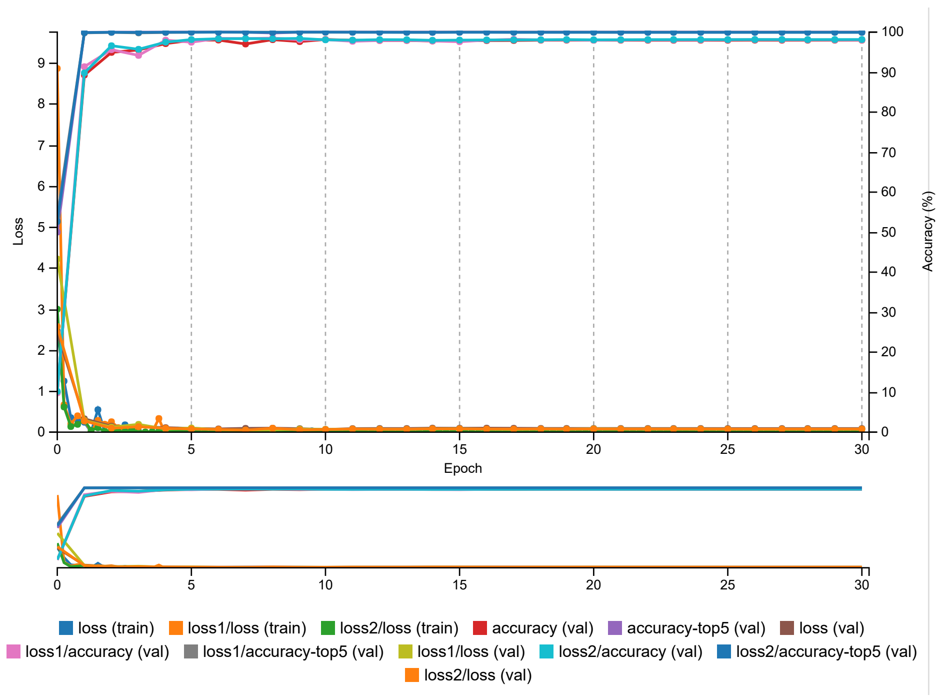
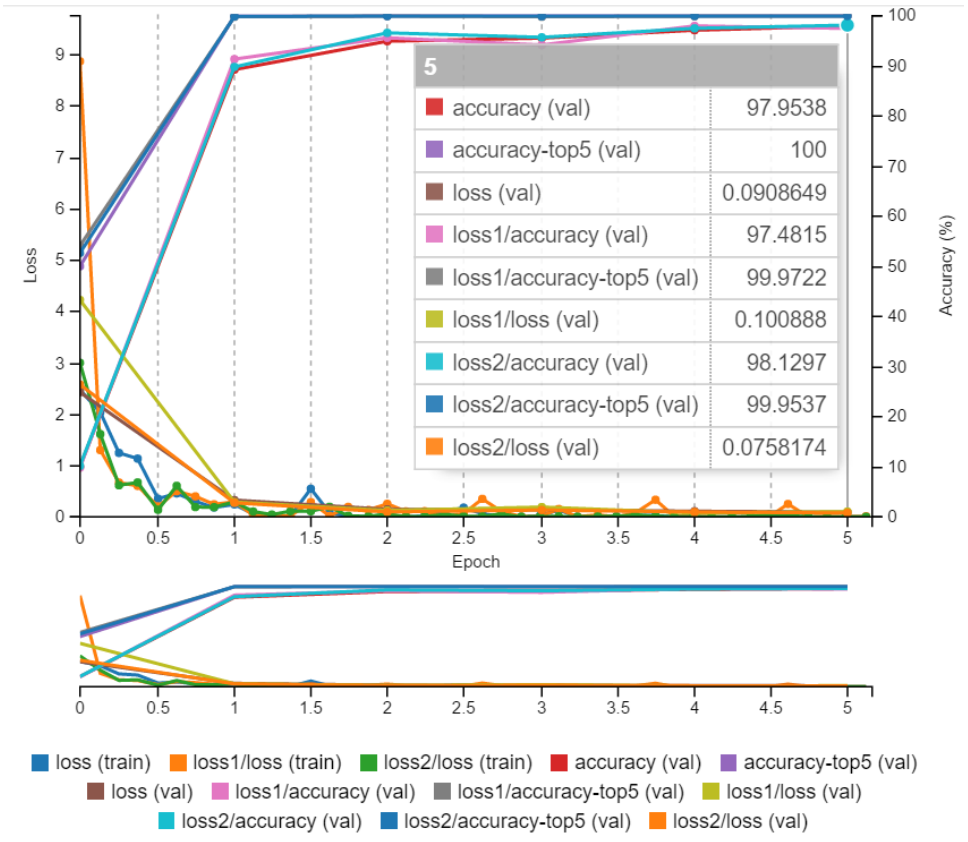
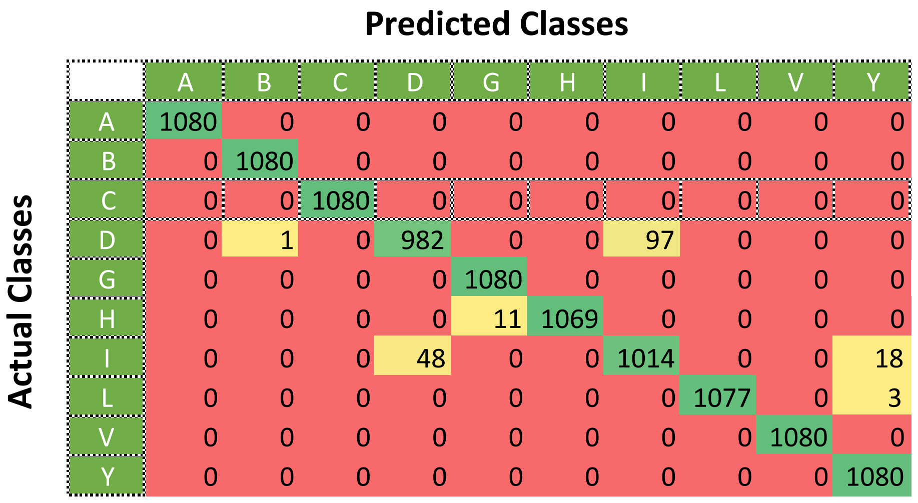


Figure 5.12: Accuracy results after 30 epochs for GoogleNet



5	
🟥 accuracy (val)	97.9538
🟪 accuracy-top5 (val)	100
🟫 loss (val)	0.0908649
🟪 loss1/accuracy (val)	97.4815
⬜ loss1/accuracy-top5 (val)	99.9722
🟨 loss1/loss (val)	0.100888
🟦 loss2/accuracy (val)	98.1297
🟦 loss2/accuracy-top5 (val)	99.9537
🟧 loss2/loss (val)	0.0758174

Figure 5.13: A closer look at accuracy results after 5 epochs for GoogleNet



Predicted Classes

Actual Classes

	A	B	C	D	G	H	I	L	V	Y
A	1080	0	0	0	0	0	0	0	0	0
B	0	1080	0	0	0	0	0	0	0	0
C	0	0	1080	0	0	0	0	0	0	0
D	0	1	0	982	0	0	97	0	0	0
G	0	0	0	0	1080	0	0	0	0	0
H	0	0	0	0	11	1069	0	0	0	0
I	0	0	0	48	0	0	1014	0	0	18
L	0	0	0	0	0	0	0	1077	0	3
V	0	0	0	0	0	0	0	0	1080	0
Y	0	0	0	0	0	0	0	0	0	1080

Figure 5.14: Confusion matrix for GoogleNet

Chapter 6

Conclusions and Future Works

To achieve a natural human-computer interaction for virtual environment applications, the human hand could be considered as an input device. Hand gestures are frequently used in everyday life. They also an important component of body language. There has been extensive research on rapid hand posture recognition focusing on the appearance-based models of hand.

In this research, as our first contribution, a novel multi-modal model for engagement detection, Disengagement, Attention, Intention, Action (DAIA) framework is presented. *Disengagement* happens when user is disengaged from target object. *Attention* occurs when user has attention to the target, but doesnt have intention to do any actions. In *Intention* state, user intends to perform an action, but still not doing it. Finally, in Action state, user is performing an action with hand. Using DAIA, the spectrum of mental status for performing a manipulative action is quantized in a finite number of engagement states. The second contribution of this research is designing multiple binary classifiers based on upper-body postures for state detection. 3D skeleton data is extracted from depth image and is used to extract body posture information. One of these binary classifiers is *Facing classifier*, and designed based on body direction towards the target object. This classifier is used to detect transition between *Disengagement* and *Attention* states. In addition, combining the output of all binary classifiers makes engagement feature vector. This feature vector could be extended using other channels of biometric information such as voice or gaze. Using engagement feature vector, an SVM classifier is trained to detect the most important transition in mental state which is *Intention to Act*, and indicates the transition from *Attention* to *Intention* or *Action*. However these classifiers recognize the state change with acceptable accuracy, minor changes in body postures for some milliseconds may result in transition to other states. For removing this unwanted noise and increasing the accuracy of system, an FSM is designed based on the properties of human activities. The design of *Engagement FSM* is third major contribution of this research.

Different hand postures in Intention state provide useful information for the system. For recognizing different hand postures in this state, a novel algorithm for hand-shape modeling is proposed which is the forth major contribution of this research. Another contribution is introducing a new hand posture dataset, HandReader dataset, which consists of 724 RGB images from 10 different hand postures of 74 individuals. These postures are 10 ASL alphabets; namely $A, B, C, D, G, H, L, I, V$, and Y.

Experimental results for FSM which classifies each frame into one of four different engagement states is 92.3% true detection rate in total. Results also show FSM can segment user hand gestures more robustly. The processing time for each frame is less than 10ms which indicates real-time usability of the algorithm since system processes 30 frames per second. For hand posture recognition, we used ten different classes of posture in the HandReader dataset. After preprocessing, 45 instances from each sample in the dataset are made using rotation matrix and created 33,300 samples in total. two-third of these samples are used for training and one-third is used for validation based on k-fold cross-validation method. In order to train the system, k-NN, SVM classifiers with linear and RBF kernel and Deep Convolutional Neural Network have been employed to find the best classifier. Deep Convolutional Neural Network based on GoogleNet outperforms the other classifiers with average validation accuracy of 98%. The future works for this research are designing classifiers for hand and body gesture recognition and extending the system for multi-user engagement detection. Because we offered a novel time-series for engagement, Recurrent Neural Networks seems to be a promising approach for engagement and gesture recognition in future research.

6.1 Contributions

6.1.1 Multi-modal Engagement Detection Framework

For hand gesture and posture recognition, we have presented a framework that initially detects user's engagement and afterwards employs related classifiers for posture or gesture classification. Figure 6.1 depicts the framework of our approach.

DAIA framework consists of four distinguishable mental states which are *Disengagement*, *Attention*,

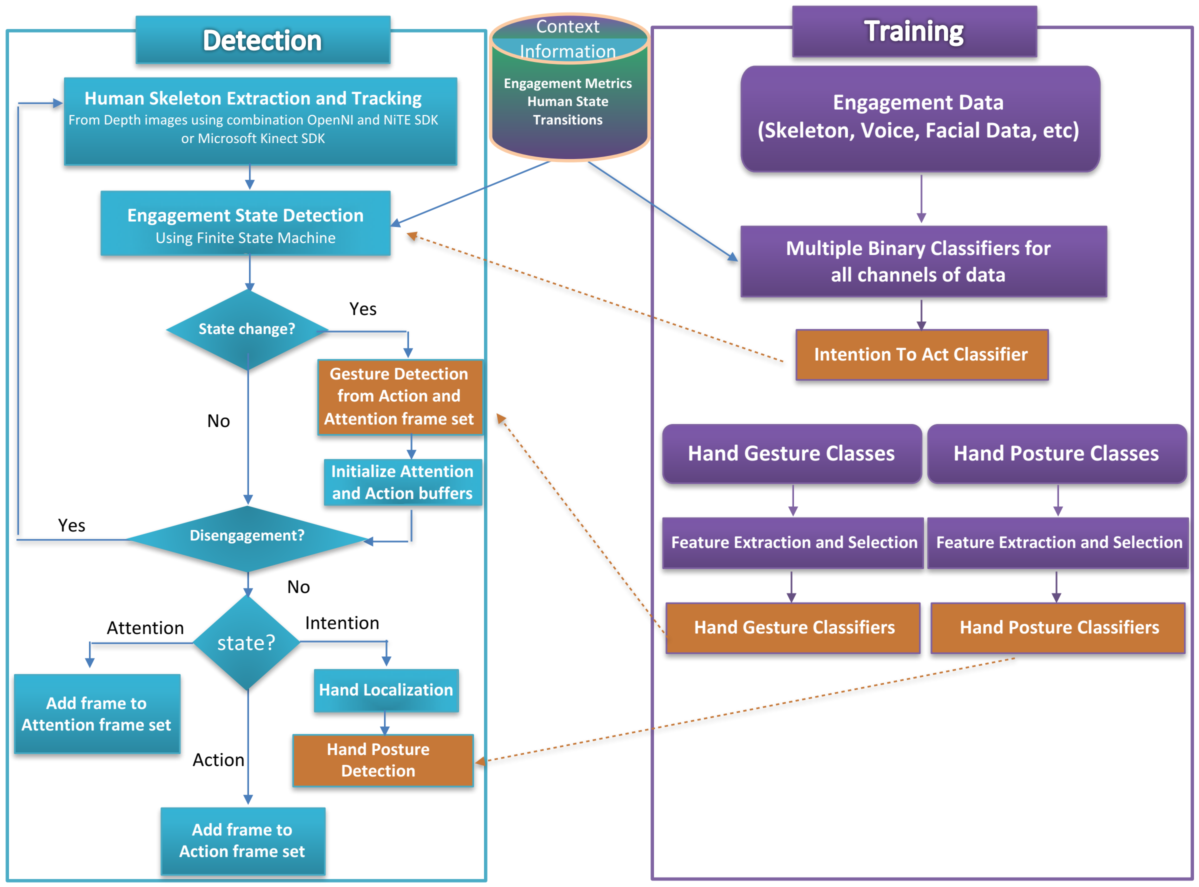

Detection
Human Skeleton Extraction and Tracking
From Depth images using combination OpenNI and NiTE SDK or Microsoft Kinect SDK
Engagement State Detection
Using Finite State Machine
State change?
Yes
No
Gesture Detection from Action and Attention frame set
Initialize Attention and Action buffers
Yes
Disengagement?
No
Attention
state?
Intention
Add frame to Attention frame set
Hand Localization
Hand Posture Detection
Action
Add frame to Action frame set
Context Information
Engagement Metrics
Human State Transitions
Training
Engagement Data (Skeleton, Voice, Facial Data, etc)
Multiple Binary Classifiers for all channels of data
Intention To Act Classifier
Hand Gesture Classes
Hand Posture Classes
Feature Extraction and Selection
Feature Extraction and Selection
Hand Gesture Classifiers
Hand Posture Classifiers

Figure 6.1: Engagement Detection Framework for hand gesture and posture recognition

Intention and *Action*. These four states and their relations are presented in Figure 6.2. From left to right in Figure 6.2, the engagement level is increased.

Disengagement happens when user is disengaged from target object. *Attention* occurs when user has attention to the target, but doesnt have intention to do any actions. In *Intention* state, user intends to perform an action, but still not doing it. Finally, in Action state, user is performing an action with hand. Using DAIA, the spectrum of mental status for performing a manipulative action is quantized in a finite number of engagement states.

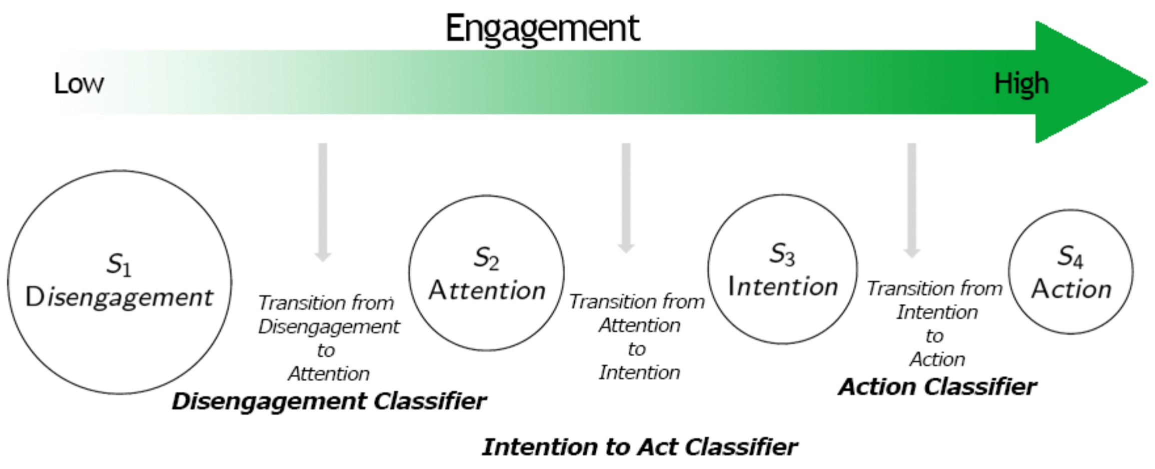

Engagement
Low
High
S_1 Disengagement
Transition from Disengagement to Attention
S_2 Attention
Transition from Attention to Intention
S_3 Intention
Transition from Intention to Action
S_4 Action
Disengagement Classifier
Action Classifier
Intention to Act Classifier

Figure 6.2: Engagement scales from Disengagement to Action

6.1.2 Upper-body Binary Classifiers

The second contribution of this research is designing multiple binary classifiers based on upper-body postures for state detection (Table 6.1). 3D skeleton data is extracted from depth image and is used to extract body posture information. One of these binary classifiers is *Facing classifier*, and designed based on body direction towards the target object. This classifier is used to detect transition between *Disengagement* and *Attention* states. In addition, combining the output of all binary classifiers makes engagement feature vector. This feature vector could be extended using other channels of biometric information such as voice or gaze. Using engagement feature vector, an SVM classifier is trained to detect the most important transition in mental state which is *Intention to Act*, and indicates the transition from *Attention* to *Intention* or *Action*.

6.1.3 Engagement FSM

Although binary classifiers recognize the state change with acceptable accuracy, minor changes in body postures for some milliseconds may result in transition to other states. For removing this unwanted noise

Table 6.1: Binary classifiers based on engagement metrics

Engagement Metric	Binary Classifiers
Hand Horizontal	Right of Body, Close to Body, Left of Body
Hand Vertical	Below Hip, Below Torso, Below Shoulder, Below Head
Hand Depth	Back of Body, Close to Body, Front of Body
Hand Speed	Stopped, Slow, Fast, Too Fast
Body Direction	Facing Sensor
Leaning	Lean back, No Lean, Lean Forward
Special Postures	Hands folded, Hands on Head, Hands on Torso

and increasing the accuracy of system, an FSM is designed based on the properties of human activities. The design of *Engagement FSM* is third major contribution of this research depicted in Figure 6.3. This engagement framework helps in gesture segmentation and providing sequences of frames for hand posture and gesture classifiers.

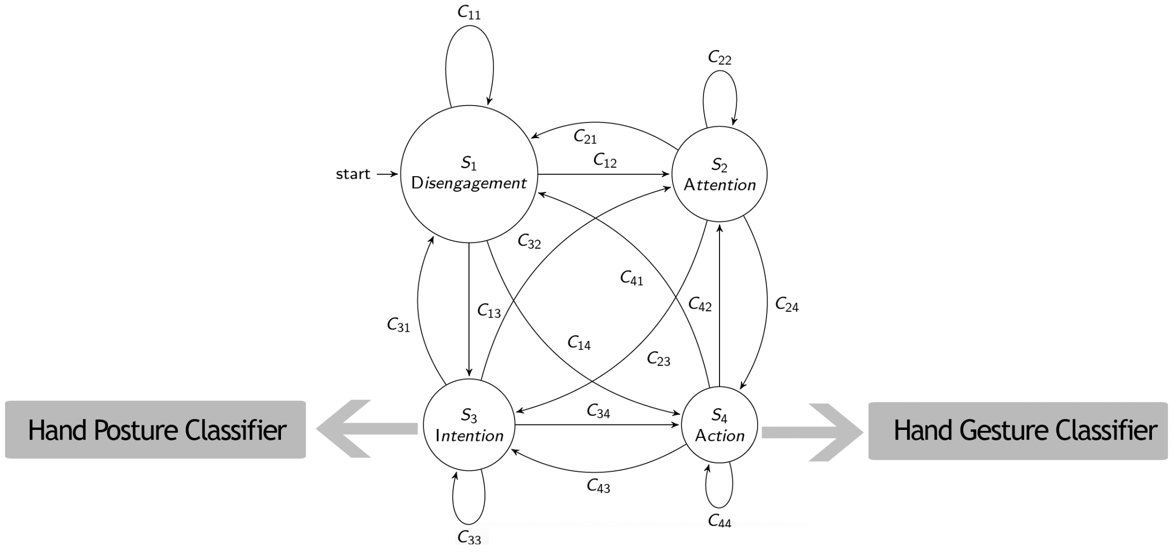


Figure 6.3: Engagement Detection State Machine for Hand Posture and Gesture Recognition

6.1.4 HandReader Datasets

We created HandReader dataset for some alphabets of American Sign Language (ASL). This dataset contains 500 RGB color images of 10 different letters of ASL performed by 50 different people (both males and females). The dataset has a black background. We also want to create HandReader2 dataset which contains 740 128 $\times$ 128 square hand silhouettes. Five hundred of these images are originally from HandReader dataset, and the rest are obtained from Triesch Static Hand Posture Database. We subtracted the dark background from hand shapes and scaled and centered them inside a 128 $\times$ 128 square. After applying a threshold, we converted these hand postures into binary images, i.e., hand silhouettes. This is helpful in feature extraction as any feature extracted from these images would be scale and translation invariant.

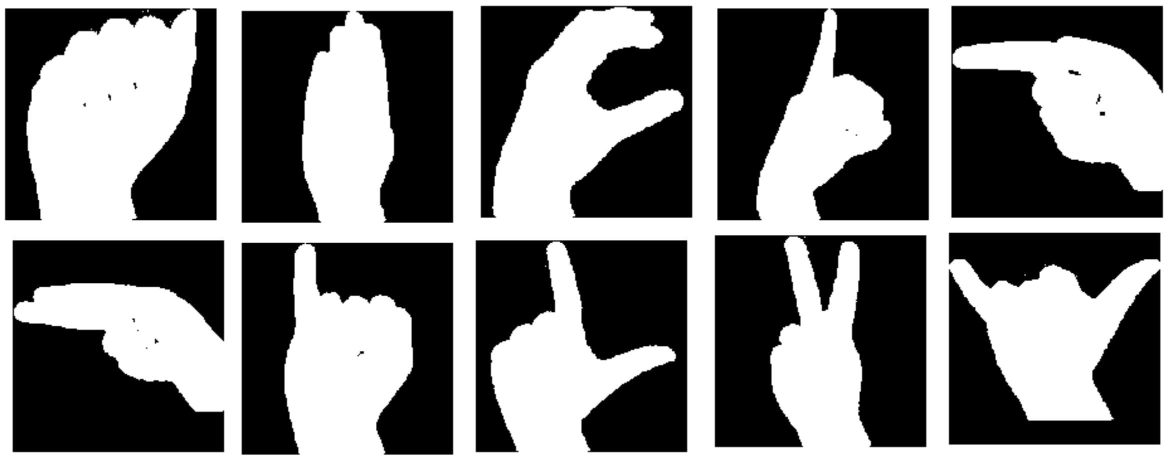


Figure 6.4: HandReader2 Dataset

Afterwards, we combined HandReader dataset with Triesch dataset to create HandReader2. HandReader2 is made from 10 RGB hand postures from 74 different people. Figure 6.5 illustrates 74 different V postures which are used in creation of HandReader2.

6.1.5 Rotation Sensitive Descriptor For Hand Postures

We have normalized hand silhouette into a 128 $\times$ 128 square (HandReader2 dataset). In contrast to all current approaches which are rotation invariant, our shape descriptor is sensitive to rotation; however, it is scale and translation invariant. The rotation sensitive shape descriptor is useful in some applications

Figure 6.5: HandReader2 is made from 10 RGB hand postures from 74 different people. This figure illustrates 74 different V postures which are used in creation of HandReader2.

such as ASL recognition because some letters such as "**I**" and "**J**" have similar postures in different orientations. Although our shape descriptor is rotation sensitive, we accept postures with 22 degrees deviation from the model postures to avoid slight differences. We used the rotation matrix to create samples from -22, to 22 degrees deviation from the main posture sample and label all of them for the same class in training. Please note this configuration even helps to achieve a rotation invariant descriptor for any posture in the test stage of the system. Using the rotation matrix, if we rotate any test posture four times, each time for 45 degrees, it should align once to a posture class since we had trained the system for 45 degrees deviation (-22 to 22) of each class sample in the training step.

6.1.6 Features Reduction

Performing PCA directly on the covariance matrix of the images is often computationally infeasible. If small, say 100×100, gray-scale images are used, each image is a point in a 10,000-dimensional space and the covariance matrix S is a matrix of $10000 \times 10000 = 10^8$ elements. However, inspired by Eigenfaces[104] method, the rank of the covariance matrix is limited by the number of training examples: if there are N training examples, there will be at most $N - 1$ eigenvectors with non-zero eigenvalues. Therefore, in the pilot project with fifty 128×128 training examples ($N = 50$, 5 posture classes with 10 different samples for each class), we can reduce the dimensionality from 16384 to $N - 1 = 49$. This reduction is performed in real-time and increases Signal-to-noise ratio (SNR).

6.2 Future Works

Future works for this research include designing classifiers for hand and body gesture recognition and extending the system for multi-user engagement detection. Because we offered a novel time-series for engagement, Recurrent Neural Networks seems to be a promising approach for engagement and gesture recognition in future research.

Figure 6.2 shows outline of our extended system in our future research.

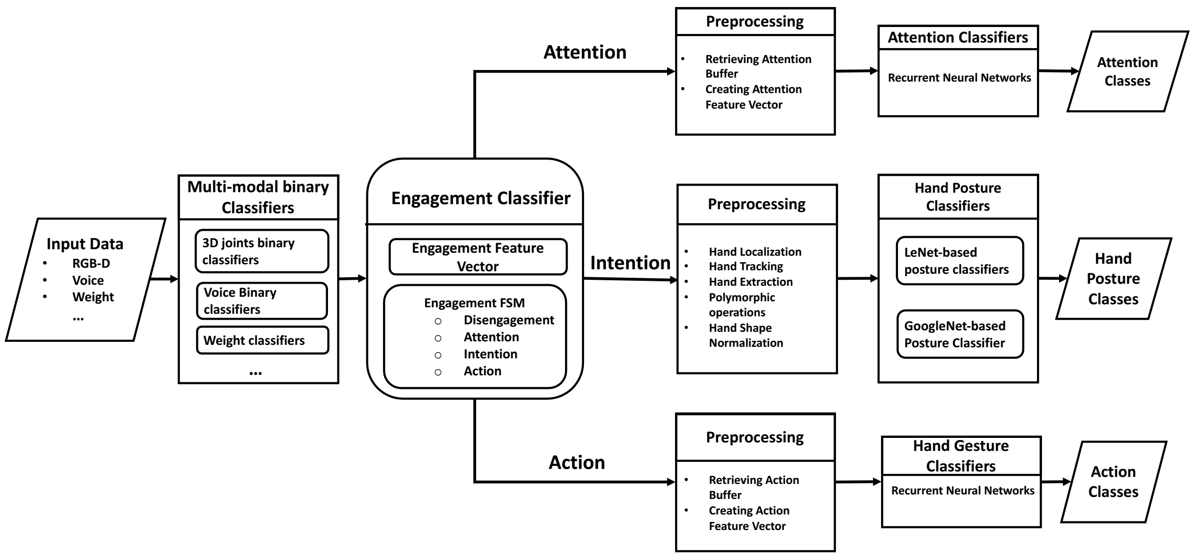


Figure 6.6: Outline of our extended system in our future research.

Summary of our future works are as follows:

- **Multi-modal approach**: Using other channels of biometric information such as voice direction, gaze, facial expressions. We expect as much as more related classifiers we will have for extracting meaningful information from users, we can predict the engagement more precisely.

- **Multi-user platform**: Designing multi-user platform for engagement detection is one of our future goals.(Figure 6.2). For this purpose, an engagement feature vector and a separate FSM for each user should be extracted to segment gestures of users in front of the sensors. This will help in creating a multi-user platform such as smart meeting rooms.

- **Gesture Recognition**: Designing hand gesture classifiers based on Engagement Feature Vector and binary classifiers. Recurrent Neural Networks seems to be a promising approach for engagement and gesture recognition as they are well-known for time-series analysis.

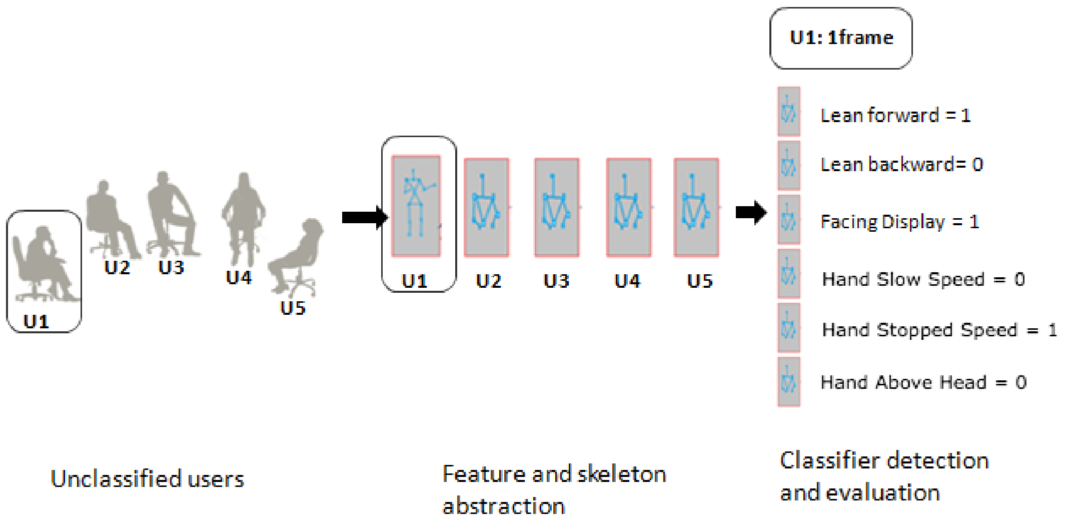


Figure 6.7: Multi-user engagement detection system

References

[1] Candace L Sidner, Cory D Kidd, Christopher Lee, and Neal Lesh. Where to look: a study of human-robot engagement. In *Proceedings of the 9th international conference on Intelligent user interfaces*, pages 78–84. ACM, 2004.

[2] Ying Wu and Thomas S Huang. Hand modeling, analysis and recognition. *Signal Processing Magazine, IEEE*, 18(3):51–60, 2001.

[3] Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition with kinect sensor. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 759–760. ACM, 2011.

[4] Yi Li. Hand gesture recognition using kinect. In *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*, pages 196–199. IEEE, 2012.

[5] Zahoor Zafrulla, Helene Brashear, Thad Starner, Harley Hamilton, and Peter Presti. American sign language recognition with the kinect. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 279–286. ACM, 2011.

[6] Helen Cooper, Eng-Jon Ong, Nicolas Pugeault, and Richard Bowden. Sign language recognition using sub-units. *The Journal of Machine Learning Research*, 13(1):2205–2231, 2012.

[7] Mariusz Oszust and Marian Wysocki. Polish sign language words recognition with kinect. In *Human System Interaction (HSI), 2013 The 6th International Conference on*, pages 219–226. IEEE, 2013.

[8] Paulo Trigueiros, Fernando Ribeiro, and Luís Paulo Reis. Vision based referee sign language recognition system for the robocup msl league. In *RoboCup 2013: Robot World Cup XVII*, pages 360–372. Springer, 2014.

[9] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand-gesture applications. *Communications of the ACM*, 54(2):60–71, 2011.

[10] Michael Van den Bergh, Daniel Carton, Roderick De Nijs, Nikos Mitsou, Christian Landsiedel, Kolja Kuehnlenz, Dirk Wollherr, Luc Van Gool, and Martin Buss. Real-time 3d hand gesture

interaction with a robot for understanding directions from humans. In *RO-MAN, 2011 IEEE*, pages 357–362. IEEE, 2011.

[11] Chun Zhu and Weihua Sheng. Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(3):569–573, 2011.

[12] Ghassem Tofighi, Nasser Ali Afarin, Kamraan Raahemifar, and Anastasios N Venetsanopoulos. Hand pointing detection using live histogram template of forehead skin. In *2014 19th International Conference on Digital Signal Processing*, pages 383–388. IEEE, 2014.

[13] Dai Fujita and Takashi Komuro. Three-dimensional hand pointing recognition using two cameras by interpolation and integration of classification scores. In *Computer Vision-ECCV 2014 Workshops*, pages 713–726. Springer, 2014.

[14] Ankit Chaudhary, Jagdish Lal Raheja, Karen Das, and Sonia Raheja. Intelligent approaches to interact with machines using hand gesture recognition in natural way: a survey. *arXiv preprint arXiv:1303.2292*, 2013.

[15] Gan Lu, Lik-Kwan Shark, Geoff Hall, and Ulrike Zeshan. Immersive manipulation of virtual objects through glove-based hand gesture interaction. *Virtual Reality*, 16(3):243–252, 2012.

[16] HCL Technologies. WHITE PAPER Need for Gesture Recognitions. `http://www.hcltech.com/sites/default/files/gesture_recognition.pdf`, 2014. [Online; accessed 27-March-2015].

[17] Dominique Vaufreydaz, Wafa Johal, and Claudine Combe. Starting engagement detection towards a companion robot using multimodal features. *Robotics and Autonomous Systems*, 2015.

[18] David Klotz, Johannes Wienke, Julia Peltason, Britta Wrede, Sebastian Wrede, Vasil Khalidov, and Jean-Marc Odobez. Engagement-based multi-party dialog with a humanoid robot. In *Proceedings of the SIGDIAL 2011 Conference*, pages 341–343. Association for Computational Linguistics, 2011.

[19] Julia Schwarz, Charles Claudius Marais, Tommer Leyvand, Scott E Hudson, and Jennifer Mankoff. Combining body pose, gaze, and gesture to determine intention to interact in vision-based interfaces. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3443–3452. ACM, 2014.

[20] Ross Mead, Amin Atrash, and Maja J Matarić. Proxemic feature recognition for interactive robots: automating metrics from the social sciences. In *International conference on social robotics*, pages 52–61. Springer, 2011.

[21] Jyotirmay Sanghvi, Ginevra Castellano, Iolanda Leite, André Pereira, Peter W McOwan, and Ana Paiva. Automatic analysis of affective postures and body motion to detect engagement with a game companion. In *2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 305–311. IEEE, 2011.

[22] Nadia Bianchi-Berthouze. Understanding the role of body movement in player engagement. *Human–Computer Interaction*, 28(1):40–75, 2013.

[23] Andrea Kleinsmith and Nadia Bianchi-Berthouze. Affective body expression perception and recognition: A survey. *Affective Computing, IEEE Transactions on*, 4(1):15–33, 2013.

[24] N Bianchi-Berthouze. What can body movement tell us about players engagement. *Measuring Behavior12*, pages 94–97, 2012.

[25] Stylianos Asteriadis, Kostas Karpouzis, and Stefanos Kollias. Feature extraction and selection for inferring user engagement in an hci environment. In *Human-Computer Interaction. New Trends*, pages 22–29. Springer, 2009.

[26] Salah Bourennane and Caroline Fossati. Comparison of shape descriptors for hand posture recognition in video. *Signal, Image and Video Processing*, 6(1):147–157, 2012.

[27] Paulo Trigueiros, António Fernando Ribeiro, and Luís Paulo Reis. A comparative study of different image features for hand gesture machine learning. 2013.

[28] Marek P Michalowski, Selma Sabanovic, and Reid Simmons. A spatial model of engagement for a social robot. In *Advanced Motion Control, 2006. 9th IEEE International Workshop on*, pages 762–767. IEEE, 2006.

[29] Dominique Vaufreydaz, Wafa Johal, and Claudine Combe. Starting engagement detection towards a companion robot using multimodal features. *Robotics and Autonomous Systems*, 75:4–16, 2016.

[30] Iolanda Leite, Marissa McCoy, Daniel Ullman, Nicole Salomons, and Brian Scassellati. Comparing models of disengagement in individual and group interactions. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 99–105. ACM, 2015.

[31] Mingqiang Yang, Kidiyo Kpalma, and Joseph Ronsin. A survey of shape feature extraction techniques. *Pattern recognition*, pages 43–90, 2008.

[32] Zhou Ren, Junsong Yuan, Jingjing Meng, and Zhengyou Zhang. Robust part-based hand gesture recognition using kinect sensor. *Multimedia, IEEE Transactions on*, 15(5):1110–1120, 2013.

[33] Jochen Triesch and Christoph Von Der Malsburg. Robust classification of hand postures against complex backgrounds. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 170–170. IEEE Computer Society, 1996.

[34] Ghassem Tofighi. List of publications on Google scholar. `https://scholar.google.ca/citations?user=qtN8MqoAAAAJ&hl=en&oi=ao`, 2017. [Online; accessed 23-Jan-2017].

[35] Ghassem Tofighi. Ghassem Tofighi's Homepage. `http://ghassem.com`, 2017. [Online; accessed 23-Jan-2017].

[36] Ghassem Tofighi. Ghassem Tofighi's Homepage. `https://sites.google.com/site/tofighi/`, 2017. [Online; accessed 23-Jan-2017].

[37] Saman Sarraf, Cristina Saverino, Halleh Ghaderi, and Jon Anderson. Brain network extraction from probabilistic ica using functional magnetic resonance images and advanced template matching techniques. In *Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on*, pages 1–6. IEEE, 2014.

[38] Saman Sarraf and Ali Mohammad Golestani. A robust and adaptive decision-making algorithm for detecting brain networks using functional mri within the spatial and frequency domain. In *The IEEE International Conference on Biomedical and Health Informatics (BHI)*, pages 1–6. IEEE, 2016.

[39] Saman Sarraf and Jian Sun. Advances in functional brain imaging: A comprehensive survey for engineers and physical scientists. *International Journal of Advanced Research*, 4(8):640–660, 2016.

[40] Saman Sarraf and Ghassem Tofighi. Deep learning-based pipeline to recognize alzheimer s disease using fmri data. *bioRxiv*, page 066910, 2016.

[41] Saman Sarraf and Ghassem Tofighi. Deep learning-based pipeline to recognize alzheimer's disease using fmri data. In *2016 Future Technologies Conference (FTC)*, pages 816–820, Dec 2016.

[42] Martha Davis and Dean Hadiks. Nonverbal behavior and client state changes during psychotherapy. *Journal of Clinical Psychology*, 46(3):340–351, 1990.

[43] Adam Fletcher. Meaningful student involvement. *Retrieved on December*, 1:2012, 2012.

[44] Wilmar B Schaufeli, Arnold B Bakker, and Marisa Salanova. The measurement of work engagement with a short questionnaire a cross-national study. *Educational and psychological measurement*, 66(4):701–716, 2006.

[45] MK Viblis and Kostas J Kyriakopoulos. Gesture recognition: the gesture segmentation problem. *Journal of Intelligent and Robotic Systems*, 28(1-2):151–158, 2000.

[46] Adrian Bulzacki. *Machine Recognition Of Human Gestures Through Principal Joint Variable Analysis*. PhD thesis, Ryerson University, 2015.

[47] Rodie Cowie, Ursula Hess, Shlomo Hareli, Maria Francesca O'Connor, Laurel D Riek, Louis-Philippe Morency, Jonathan Aigrain, Severine Dubuisson, Marcin Detyniecki, Mohamed Chetouani, et al. Cbar 2015: Context based affect recognition. 2015.

[48] Hanan Salam and Mohamed Chetouani. A multi-level context-based modeling of engagement in human-robot interaction. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, volume 3, pages 1–6. IEEE, 2015.

[49] Candace L Sidner, Christopher Lee, Louis-Philippe Morency, and Clifton Forlines. The effect of head-nod recognition in human-robot conversation. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 290–296. ACM, 2006.

[50] Francesco Di Nocera, Bernd Lorenz, AJ Tattersall, and Raja Parasuraman. New possibilities for adaptive automation and work design. *NATO SCIENCE SERIES SUB SERIES I LIFE AND BEHAVIOURAL SCIENCES*, 355:363–372, 2003.

[51] Nele Dael, Marcello Mortillaro, and Klaus R Scherer. Emotion expression in body action and posture. *Emotion*, 12(5):1085, 2012.

[52] Maria Frank, Ghassem Tofighi, Haisong Gu, and Renate Fruchter. Engagement detection in meetings. *arXiv preprint arXiv:1608.08711*, 2016.

[53] Maria Frank and Ghassem Tofighi. Technology report engagement state detection in meeting scenarios. Konica Minolta Research Lab USA, Inc, 2015.

[54] Harry C Triandis. Values, attitudes, and interpersonal behavior. In *Nebraska symposium on motivation*. University of Nebraska Press, 1979.

[55] Paschal Sheeran. Intentionbehavior relations: A conceptual and empirical review. *European review of social psychology*, 12(1):1–36, 2002.

[56] PR Cohen and HJ Levesque. Intention is choice with commitment artificial intelligence, 42 213261. *Cohen, PR and Levesque, HJ*, 1995.

[57] Martin Fishbein and Icek Ajzen. Belief, attitude, intention, and behavior: An introduction to theory and research. 1977.

[58] Wafa Benkaouar and Dominique Vaufreydaz. Multi-sensors engagement detection with a robot companion in a home environment. In *Workshop on Assistance and Service robotics in a human environment at IEEE International Conference on Intelligent Robots and Systems (IROS2012)*, pages 45–52, 2012.

[59] Andrew D Engell and James V Haxby. Facial expression and gaze-direction in human superior temporal sulcus. *Neuropsychologia*, 45(14):3234–3241, 2007.

[60] Carey D Balaban, Joseph Cohn, Mark S Redfern, Jarad Prinkey, Roy Stripling, and Michael Hoffer. Postural control as a probe for cognitive state: Exploiting human information processing to enhance performance. *International Journal of Human-Computer Interaction*, 17(2):275–286, 2004.

[61] Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker, and Preetam Maloor. Match: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 376–383. Association for Computational Linguistics, 2002.

[62] Michael Johnston and Srinivas Bangalore. Finite-state multimodal parsing and understanding. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 369–375. Association for Computational Linguistics, 2000.

[63] Bradley A Singletary and Thad E Starner. Learning visual models of social engagement. In *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings. IEEE ICCV Workshop on*, pages 141–148. IEEE, 2001.

[64] Marie-Luce Bourguet. Designing and prototyping multimodal commands. In *INTERACT*, volume 3, pages 717–720. Citeseer, 2003.

[65] Maria Frank, Renate Fruchter, and Marianne Leinikka. High engagement decreases fatigue effect in global learners. In *SAVI Symposium on New ways to teach and learn for student engagement*. Stanford University, 2015.

[66] Carl Gutwin and Reagan Penner. Improving interpretation of remote gestures with telepointer traces. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 49–57. ACM, 2002.

[67] Paul Ekman. Emotional and conversational nonverbal signals. In *Language, knowledge, and representation*, pages 39–50. Springer, 2004.

[68] Paul Ekman and Wallace V Friesen. Hand movements. *Journal of communication*, 22(4):353–374, 1972.

[69] Harry J Witchel, Carina EI Westling, Julian Tee, Aoife Healy, Robert Needham, and Nachiappan Chockalingam. What does not happen: Quantifying embodied engagement using nimi and self-adaptors. *Participations*, 11(1):304–331, 2014.

[70] Selene Mota and Rosalind W Picard. Automated posture analysis for detecting learner's interest level. In *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*, volume 5, pages 49–49. IEEE, 2003.

[71] Sidney DMello, Patrick Chipman, and Art Graesser. Posture as a predictor of learners affective engagement. In *Proceedings of the 29th annual cognitive science society*, volume 1, pages 905–910. Citeseer, 2007.

[72] Ma, J and Fruchter, Renate. eRing: Body Motion Engagement Detection and Feedback in Global Teams. SAVI Symposium on New ways to teach and learn for student engagement, Stanford University, 2015. April 2015.

[73] Gordon Kurtenbach and Eric A Hulteen. Gestures in human-computer communication. *The art of human-computer interface design*, pages 309–317, 1990.

[74] Mark Billinghurst and Bill Buxton. Gesture based interaction. *Haptic input*, 24, 2011.

[75] Brad A Myers. A brief history of human-computer interaction technology. *interactions*, 5(2):44–54, 1998.

[76] Johann Schrammel, Lucas Paletta, and Manfred Tscheligi. Exploring the possibilities of body motion data for human computer interaction research. In *Symposium of the Austrian HCI and Usability Engineering Group*, pages 305–317. Springer, 2010.

[77] David McNeill. *Gesture and thought.* University of Chicago Press, 2008.

[78] Robert M Krauss, Yihsiu Chen, and Purnima Chawla. Nonverbal behavior and nonverbal communication: What do conversational hand gestures tell us? *Advances in experimental social psychology*, 28:389–450, 1996.

[79] Robert Stephen Feldman and Bernard Rimé. *Fundamentals of nonverbal behavior.* Cambridge University Press, 1991.

[80] David Efron. *Gesture and Environment: A tentative study of some of the spatio-temporal and" linguistic" aspects of the gestural behavior of eastern Jews and southern Italians in New York city, living under similar as well as different environmental conditions.* King's crown Press, 1941.

[81] Chiara Bassetti. Social gatherings: Integrating sociological theory and analysis into computer vision methods and techniques. *Workshop on Group and Growd Behavior Analysis and Understanding*, 2015.

[82] Stefan Scherer, Michael Glodek, Georg Layher, Martin Schels, Miriam Schmidt, Tobias Brosch, Stephan Tschechne, Friedhelm Schwenker, Heiko Neumann, and Günther Palm. A generic framework for the inference of user states in human computer interaction. *Journal on Multimodal User Interfaces*, 6(3-4):117–141, 2012.

[83] Vladimir I Pavlovic, Rajeev Sharma, and Thomas S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):677–695, 1997.

[84] Thomas B Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer vision and image understanding*, 81(3):231–268, 2001.

[85] Konstantinos G Derpanis. A review of vision-based hand gestures. *Feb*, 2004.

[86] Siddharth S Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54, 2015.

[87] Jesus Suarez and Robin R Murphy. Hand gesture recognition with depth images: A review. In *RO-MAN, 2012 IEEE*, pages 411–417. IEEE, 2012.

[88] Soroush Falahati. *OpenNI cookbook.* Packt Publishing Ltd, 2013.

[89] Joanna Marnik. Hand shape recognition for human-computer interaction. In *Man-Machine Interactions*, pages 95–102. Springer, 2009.

[90] Daniel Kelly, John McDonald, and Charles Markham. A person independent system for recognition of hand postures used in sign language. *Pattern Recognition Letters*, 31(11):1359–1368, 2010.

[91] Ghassem Tofighi. Hand Reader Dataset is a hand posture dataset containing 500 images from 10 different hand postures. `https://github.com/tofighi/Hand-Reader-Dataset`, 2012. [Online; accessed 01-May-2015].

[92] T Crimmins. A complete set of fourier descriptors for two-dimensional shapes. *Systems, Man and Cybernetics, IEEE Transactions on*, 12(6):848–855, 1982.

[93] Eric Persoon and King-Sun Fu. Shape discrimination using fourier descriptors. *Systems, Man and Cybernetics, IEEE Transactions on*, 7(3):170–179, 1977.

[94] F Ghorbel. Stability of invariant fourier descriptors and its inference in the shape classification. In *Pattern Recognition, 1992. Vol. III. Conference C: Image, Speech and Signal Analysis, Proceedings., 11th IAPR International Conference on*, pages 130–133. IEEE, 1992.

[95] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[96] Satish Kumar and Chandan Singh. A study of zernike moments and its use in devnagari handwritten character recognition. In *Intl. Conf. on Cognition and Recognition*, pages 514–520, 2005.

[97] Sun-Kyoo Hwang and Whoi-Yul Kim. A novel approach to the fast computation of zernike moments. *Pattern Recognition*, 39(11):2065–2076, 2006.

[98] Attila Licsár and Tamás Szirányi. User-adaptive hand gesture recognition system with interactive training. *Image and Vision Computing*, 23(12):1102–1114, 2005.

[99] Dengsheng Zhang and Guojun Lu. A comparative study on shape retrieval using fourier descriptors with different shape signatures. In *Proc. International Conference on Intelligent Multimedia and Distance Education (ICIMADE01)*, 2001.

[100] Ronald Poppe and Mannes Poel. Comparison of silhouette shape descriptors for example-based human pose recovery. In *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, pages 541–546. IEEE, 2006.

[101] Chan Wah Ng and Surendra Ranganath. Real-time gesture recognition system and application. *Image and Vision computing*, 20(13):993–1007, 2002.

[102] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and vision computing*, 21(8):745–758, 2003.

[103] Peter RG Harding and Tim J Ellis. Recognizing hand gesture using fourier descriptors. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 286–289. IEEE, 2004.

[104] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[105] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[106] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification.* John Wiley & Sons, 2012.

[107] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[108] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[109] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[110] A Aizerman, Emmanuel M Braverman, and LI Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

[111] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.

[112] Kai-Bo Duan and S Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In *Multiple Classifier Systems*, pages 278–285. Springer, 2005.

[113] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[114] John C Platt, Nello Cristianini, and John Shawe-Taylor. Large margin dags for multiclass classification. In *nips*, volume 12, pages 547–553, 1999.

[115] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *arXiv preprint cs/9501101*, 1995.

[116] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[117] David R Wright. Finite state machines. *CSC215 Class Notes. Prof. David R. Wright website, N. Carolina State Univ. Retrieved July*, 14:2012, 2005.

[118] Thomas Koshy. *Discrete mathematics with applications*. Academic Press, 2004.

[119] Grady Booch. *The unified modeling language user guide*. Pearson Education India, 2005.

[120] Vlad Estivill-Castro, René Hexel, and Carl Lusty. High performance relaying of c++ 11 objects across processes and logic-labeled finite-state machines. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 182–194. Springer, 2014.

[121] Paul E Black. *Dictionary of algorithms and data structures*. National Institute of Standards and Technology, 2004.

[122] James H Martin and Daniel Jurafsky. Speech and language processing. *International Edition*, 2000.

[123] Itamar Arel, Derek C Rose, and Thomas P Karnowski. Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *Computational Intelligence Magazine, IEEE*, 5(4):13–18, 2010.

[124] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[125] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[126] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[127] Jia Deng, Alex Berg, Sanjeev Satheesh, H Su, Aditya Khosla, and L Fei-Fei. Imagenet large scale visual recognition competition 2012 (ilsvrc2012), 2012.

[128] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer, 2014.

[129] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[130] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

[131] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[132] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[133] Rick Kjeldsen, Anthony Levas, and Claudio Pinhanez. Dynamically reconfigurable vision-based user interfaces. In *Computer Vision Systems*, pages 323–332. Springer, 2003.

[134] Ghassem Tofighi, S Amirhassan Monadjemi, and Nasser Ghasem-Aghaee. Rapid hand posture recognition using adaptive histogram template of skin and hand edge contour. In *Machine Vision and Image Processing (MVIP), 2010 6th Iranian*, pages 1–5. IEEE, 2010.

[135] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

Glossary

Compound Annual Growth Rate The year-over-year growth rate of an investment over a specified period of time. The compound annual growth rate is calculated by taking the *nth* root of the total percentage growth rate, where n is the number of years in the period being considered.. 5

Degrees of Freedom In statistics, the number of degrees of freedom is the number of values in the final calculation of a statistic that are free to vary. The number of independent ways by which a dynamic system can move, without violating any constraint imposed on it, is called number of degrees of freedom.. 4

Acronyms

ASL American Sign Language. 3, 127

DAIA Disengagement, Attention, Intention, Action. iii, 126

FSM Finite State Machine. iv

HCI Human-Computer Interaction. iii, 2, 13, 16

PCA Principal Component Analysis. 63

SNR Signal-to-noise ratio. 133

SVM Support Vector Machine. 46, 63

ToF Time-of-flight. 3

VR Virtual Reality. 8