MACHINE RECOGNITION OF HUMAN GESTURES THROUGH PRINCIPAL JOINT VARIABLE ANALYSIS

by

Adrian Bulzacki

Master of Engineering in Electrical and Computer Engineering, Toronto, ON, Canada, 2007
Bachelor of Science in Electronics Engineering Technology, Columbus, OH, USA, 2005

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, 2015

Declaration of Authorship

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A DISSERTATION

I hereby declare that I am the sole author of this dissertation. This is a true copy of the dissertation, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this dissertation to other institutions or individuals for the purpose of scholarly research

I further authorize Ryerson University to reproduce this dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Abstract

Program of Electrical and Computer Engineering

Ryerson University

2015

Doctor of Philosophy

by Adrian Bulzacki

Gestures are a fundamental part of human communication and are becoming a key component of human-computer interaction. Traditionally, to teach computers to recognize specific gestures, researchers have used a sensor, usually a camera, to collect large gesture datasets, which are then classified and structured using machine learning techniques. Yet finding a way to confidently differentiate between several gesture classes has proven to be rather difficult for those working in the gesture recognition field.

To capture the samples of movements necessary to train gesture recognition systems, the first step is to provide research participants with appropriate instructions. As collecting gesture data is the crucial first step of creating a robust gesture dataset, this dissertation will examine the modalities of instruction used in gesture recognition research to examine whether appropriate directives are conveyed to research participants. These experiments will result in the creation of a new dataset, the PJVA-20 dataset, comprised of 50 samples of 20 gesture classes sampled from 6 participants.

After collecting the gesture samples of the PJVA-20 dataset, this dissertation will establish the benchmark recognition system PJVA — chiefly comprised of AMFE, Polynomial Motion Approximation, and Principal Component Analysis (PCA) — to contribute to the gesture recognition literature in terms of novel gesture recognition algorithms that can achieve high speed and accuracy results. This also involves examining studies in the gesture recognition literature to determine which machine learning algorithms offer reliability, speed, and accuracy for solving complex gesture recognition problems, as well as experimenting and testing the PJVA approach against other researchers in the Computer Vision and Machine Learning fields.

In particular, the MSRC-12 research provides a benchmark point of comparison for research in this field. To test the quality of samples on the PJVA-20 against the MSRC-12, a new method is established for extracting motion feature vectors through a novel gesture recognition approach, AMFE. This is tested by applying PJVA to extract and label gesture data from both the MSRC-12 and PJVA-20 datasets.

Acknowledgements

At the end of this challenging and arduous journey, I would like to express thanks to so many people, but none more so than my final PhD supervisor, Professor Kamraan Raahemifar. Without his continued support and advice, I would never have completed such a momentous milestone in my life.

I would also like to thank all the Ryerson University instructors and mentors that guided me. First off, Professor Ling Guan who played a pivotal role in my interest in gesture recognition, and allowed me to be part of the Ryerson Media Lab (RML). Secondly, I want to thank Dr. Matthew Kyan; his guidance and instruction were instrumental in helping me take my technical knowledge to the next level. Thirdly, Professor Lian Zhao, who encouraged my academic growth, and Dr. Sean Wise who assisted with my PhD at a critical time.

I would also like to give special thanks to Dr. Sheldon Levy, President of Ryerson University, for his personal encouragement, and for setting up an environment that fostered innovation at Ryerson University. I am not the only beneficiary of his efforts, and in thanking him I know I speak for many people who have benefitted enormously from his perspective on education and entrepreneurship.

I would also like to thank all the people at the Ryerson University DMZ for their support, in particular Valerie Fox and Dr. Hossein Rahnama, as well as everyone at my research groups: the Ryerson Media Lab (RML) and Center for Interactive Multimedia Information Mining (CIM2). I also want to thank the Business Department at Ryerson for supplying me with business student volunteers to support my academic research, and the Federal Economic Development Agency for helping to fund my early research.

Finally, I would like to offer a special thanks to my parents, who have motivated and encouraged me throughout my graduate studies, both intellectually and financially. I could never have completed this dissertation without their tireless support and love.

Dedicated to the foresight to change one's course, and the possibilities of living each day like it's our own future

Contents

List of Tables

List of Figures

Abbreviations

AMFE	**A**nchored **M**otion **F**eature **E**xtraction
CMFE	**C**amera **M**otion **F**eature **E**xtraction
DTW	**D**ynamic **T**ime **W**arping
DOF	**D**egree **O**f **F**reedom
CSV	**C**omma **S**eparated **V**alues
FPS	**F**rames **P**er **S**econd
FSD	**F**ourier **S**hape **D**escriptor
GUI	**G**raphical **U**ser **I**nterface
GDF	**G**esture **D**ata **F**eature
GDP	**G**esture **D**ata **P**oint
GDPS	**G**esture **D**ata **P**oint **S**lice
GS	**G**esture **S**tudio
GSC	**G**esture **S**tudio **C**loud
HCI	**H**uman **C**omputer **I**nteraction
HMM	**H**idden **M**arkov **M**odel
ID	**I**dentity **D**ocument
IR	**I**nfrared
MV	**M**achine **V**ision
PCA	**P**rinciple **C**omponent **A**nalysis
PJVA	**P**rinciple **J**oint **V**ariable **A**nalysis
RF	**R**andom **F**orests
RGB	**R**ed **G**reen **B**lue Colour Model (RGB triplet)
VM	**V**irtual **M**achine

Chapter 1

Introduction

"How can I tell what I think till I see what I say?"

- E. M. Forster

Gestures are an essential aspect of body language and are used in everyday communications. In fact, it would be nearly impossible to avoid making some kind of gesture when communicating face-to-face with another person. Gestures can convey messages easily and seemingly wordlessly. Gestures can represent a variety of different things, from abstract ideas and emotions to representations of more tangible ideas, such as people, places, or things.

This leads us to the question: what actually defines a gesture?

Gestures, like letters and words, tend to represent just a piece of an idea; it is how those pieces are arranged that defines some form of meaning. As such, with reference to scholars in the gesture recognition field, I will define a gesture "as sequences of human skeletal body part movements (represented as body part locations) and the associated meaning that needs to be recognized by the system". Gesture recognition research uses the joints of the human body to anchor positional data of 'configurations' or 'positions' of a human body at particular points in time. Researchers in the gesture recognition field can thus see the gesture sample as as a "3D skeleton" comprised of the spatial relationships between human joints. The motion features of a 3D skeleton moving over a time period can thus be interpreted as reliable patterns by computers, and labeled as a particular "gesture class".

The research conducted for this dissertation has demonstrated that after determining the joint positions of these 3D skeletons (henceforth known as Gesture Data Points, or GDPs), PJVA can to be applied to the skeletal data to extract motion feature vectors, which has the potential to improve the accuracy of gesture recognition algorithms.

1.1 Problem Statement

Finding a way to confidently differentiate between several gesture classes has proven to be rather difficult for those working in the gesture recognition field. This dissertation will create the benchmark recognition system, PJVA, to contribute to the gesture recognition literature in terms of novel gesture recognition algorithms that can achieve high speed and accuracy results. This involved finding studies in the gesture recognition literature to determine which studies offer reliability, speed, and accuracy for solving complex gesture recognition problems.

I found that several researchers employed motion feature extraction algorithms to improve the characterization of multiple gestures in the benchmark MSRC-12 dataset. Also, to classify the resulting gesture data, researchers employed several machine learning techniques, including Hidden Conditional Random Field (HCRF), Hidden Markov Model (HMM), SVM Classification, and Random Forest (RF). Whereas many of these machine learning experiments split the benchmark MSRC-12 into two separate datasets (MSRCa comprised of 6 iconic gestures, and MSRC-12b comprised of 6 metaphoric gestures) for testing, this dissertation will test machine learning algorithms on the full 12 class MSRC-12 dataset to create a more complex gesture recognition problem.

In order to verify these experiments, this dissertation also involves creating a new dataset, PJVA-20, which will be used to 1) test charades as a modality of instruction for creating a dataset that is robust, in terms of comprising all movements necessary to recognize particular gestures, and 2) has an appropriate level of coverage for the machine learning system to cope with a wide array of users and their corresponding abilities. The same gesture recognition and machine learning approaches applied to the MSRC-12 will be applied to the PJVA-20 for comparative purposes.

1.2 Motivation

Through approaches like machine learning, computers have the potential to successfully recognize a gesture quicker and more accurately than a human being because they have the capacity to sample the space with a camera faster than humans are capable of seeing.

Leveraging collected information about the position and orientation of joints during gestures performed by humans, it is possible to employ artificial intelligence systems to learn from this data and make predictions about unseen joint information (outside of the camera's vision) and the type of gesture that it most likely represents.

1.3 Application

A machine-based intelligence capable of detecting and categorizing different types of gestures can be used to expand several essential industry sectors, such as electronic communication, interactive entertainment, and security systems.

Gesture recognition can serve as an interactive tool in environments where keyboards, mice, or vocal commands are not practical or possible. Gesture recognition also offers a non-invasive solution for detecting the movements, gestures, and upcoming actions of an individual. For example, as a future functionality, these systems could automatically detect a moment that signifies someone is beginning to suffer from an injury (e.g., falls at home or in a public place without a caregiver nearby).

For the first time, affecting both industry and academic research, gesture recognition software has the capacity to be used to conduct real-time analytics on gestures.

1.4 Research Objectives

The three objectives of this research are:

i. I want to collect a gesture dataset of 20 human body gestures in order to generate robust 3D skeletal data for gesture recognition applications. This dataset is called the PJVA-20 dataset.

   - To capture the necessary information, I will conduct a literature review in the fields of Computer Vision and Machine Learning to find vision-based approaches for sampling gestures. This will explore several studies in the literature for both hand and full body gesture samples, to refine my technique for sampling gestures and applying gesture recognition techniques to extract static and motion feature vectors. This will be explored in Section 2.2 on page 13.
   - This objective is based on an analysis of several datasets in the literature, but in particular the MSRC-12 dataset, which is a benchmark contribution in the field of gesture recognition (see Section 2.5 on page 23). Thus, this dissertation is partially focused on critiquing this dataset to improve upon its methodology in areas such as the mode of instructions provided to research participants and the orientation chosen for data extraction. This will be further explained in Section 2.5 on page
   - The MSRC-12 dataset includes data from 30 people and is comprised of 20 three-dimensional joints, captured for 12 different gesture classes. The participants were 60% male and 40% female, of which 93% were right-handed, the average height was 5'8", and the average age was 31. More details are provided in Chapter 2 on page 11. 23.
   - I have named this new dataset PJVA-20 after my gesture recognition algorithm approach, Principal Joint Variable Analysis (PJVA), which is used for gesture recognition and machine learning applications. The "20" in PJVA-20 refers to the 20 gesture classes for which positional data has been collected, comprised of 50 samples of each class sampled from 6 participants. Participants included five men and one woman between the ages of 20 to 30 years old with various body build types (see Section 3.3 on page 52).
   - The samples of the PJVA-20 dataset will be sourced from research participants who are playing a game of charades, which differs from other datasets in the literature. Charades has the potential to improve upon instructions consisting of text, video,

and static images provided by the MSRC-12 research team. This comparison will be further explained in Section 2.5.1 on page 29 and Section 3.3.1 on page 53.

ii. I want to create a novel PJVA technique comprised of several algorithms. PJVA is chiefly comprised of AMFE, Polynomial Motion Approximation, and Principal Component Analysis (PCA), and will be used to extract motion feature vectors from the MSRC-12. The novelty of this technique will be shown particularly my in Anchored Motion Feature Extraction (AMFE) technique, which reorients the extraction of motion feature data from static gesture samples (sets of GDP values).

- I aim for this to create a new benchmark in this academic field, advancing the state-of-the-art in motion feature extraction by changing the orientation of many gesture recognition techniques, from the point-of-view of the camera to the centre hip joint of the 3D skeleton generated from the gesture sample. This technique can be used alongside other gesture recognition techniques, which will primarily be explored in Section 4.4 on page 64.
- This objective will involve deleting extraneous data that may confuse the learning algorithm, and compressing the data to increase the speed of the system to be practical in real time recognition applications. See Chapter 4 on page 58.

iii. I need to apply PJVA to the motion feature data for the MSRC-12 to categorize the gesture samples appropriately and increase accuracy for gesture recognition applications.

- I aim for these results to establish PJVA as a benchmark in the literature in terms of recognition accuracy on the MSRC-12. This is accomplished by capturing the required gesture data, implementing a novel approach for extracting motion feature data from these samples, and increasing accuracy by applying machine learning algorithms to improve labelling of gesture classes. These results are described in Chapter 5 on page 72.

iv. I will apply PJVA to the static feature data of the PJVA-20 dataset to demonstrate the high quality of samples collected using charades as a modality for data capture.

- The results of these tests will show the efficacy of charades for capturing gesture samples. By achieving high results on static images alone — whereas the MSRC-12 required the addition of motion feature vectors to increase accuracy — these tests will show that charades and AMFE can capture subtle movements in gesture samples, which can help the recognition system to differentiate between separate gestures.

1.5 Contributions

The contributions of this research to the computer-based gesture recognition literature are the following:

- Created, at the time of collection, the largest dataset of its kind for marker-less skeletal 3D points, at 20 gesture classes. This database (PJVA-20) has many advantages in terms of the quality of samples and the integrity of class labeling over the MSRC-12 (Microsoft Research Cambridge 12) Kinect gesture data (see Section 3.4 on page 54), creating a gesture recognition benchmark after achieving a 98.5% accuracy on static samples alone. This dataset was designed to provide the system with more freedom of motion for participants, encouraging more natural gestures in contrast to MSRC-12's rigid and prescribed gestures.

- Created a benchmark method for designing an effective instructional modality for research participants by having research participants act out charades cards, and record their own gestures without supervision. No other dataset enabled this type of freedom, which served to prevent overfitting of the data and improve computational accuracy in recognition. This technique overcame some issues in the MSRC-12, in which the instructions may have over-explained the gesture to the participant by providing static images and video instructions that limited participants' freedom of movement (see Section 2.6 on page 40).

- Created new method for extracting static and motion feature vectors through a novel anchoring approach, AMFE. This is a foundational shift away from the current state of the

literature and thus creates a benchmark in how to extract feature vector data from 3D skeletal data for gesture recognition applications (see Section 4.2 on page 59). I consider this technique the most significant contribution from this dissertation for creating a system capable of real-time gesture recognition applications.

- Created PJVA as a benchmark in the literature in terms of feature extraction and recognition accuracy. This has been tested by applying PJVA to analyze motion feature data from the MSRC-12 (achieving a 81.49% accuracy result) and static feature data from the PJVA-20 (achieving a 98.5% accuracy result) (see Chapter 5 on page 72). All results are 10-fold cross validated.

- Showed that a state-of-the-art gesture recognition system does not need to be built on many samples of gestures. In fact, a smaller sample size in a dataset can be used very effectively to achieve both high accuracy and recognition speeds, while avoiding oversampling (see Section 2.5 on page 23). When just considering pose, the PJVA approach is capable of recognizing gestures at the speed of >600fps, while motion recognition is fast enough for real time applications.

1.6 Thesis Structure

Here, I present an overview of the structure of this thesis and the topics discussed in the following chapters.

Chapter 2: Literature Review Gesture Recognition Through a literature review, this chapter lays out the considerations for designing gesture recognition systems that can capture, classify, and label gesture samples of research participants.

Chapter 2 is organized as follows. Section 2.2 on page 13 reviews the definitions of the word 'gesture' in the literature, which has guided subsequent studies into gesture recognition. Section 2.3 on page 15 then discusses approaches to collecting gesture data in order to create gesture samples for gesture recognition research. Section 2.4 on page 21 lists several datasets

that have been developed in the gesture recognition literature. Section 2.5 on page 23 introduces the benchmark dataset MSRC-12, discussing the instructions used to convey the gestures to the research participants, as well as the techniques typically used for gesture recognition and machine learning on this dataset. Section 2.6 on page 40 reviews the gesture recognition literature to find examples of other studies that use charades as a modality of instruction to collect gesture samples.

Chapter 3: Experimental Design for Capturing Gesture Samples This chapter details the steps required to design an experiment for capturing gesture samples using a depth camera.

Chapter 3 is organized as follows. Section 3.2 on page 46 outlines the initial considerations for capturing gesture data from research participants using a depth camera and no trackers, touching upon which type of camera and distance measurement techniques were used. Section 3.3 on page 52 discusses the approaches used for capturing gesture samples for the PJVA-20 dataset, focusing on the instructional modality of charades to compare to the methodology of the MSRC-12. Section 3.4 on page 54 conducts a brief statistical comparison of the resulting gesture data to consider whether using charades as a modality of instruction has created suitable gesture samples for gesture recognition applications.

Chapter 4: PJVA Methodology: Extracting Motion Feature Vectors from MSRC-12 This chapter outlines the experiments undertaken to build the novel vision-based approach for gesture recognition, PJVA.

Chapter 4 is organized as follows. Section 4.2 on page 59 describes the novel gesture recognition technique, AMFE, which was created by reorienting the motion feature extraction process to improve gesture recognition accuracy on most datasets. Section 4.3 on page 62 discusses the calculations used to scale the resulting GDP data. Finally, Section 4.4 on page 64 outlines the PJVA approach used to extract motion feature vectors from these scaled GDP values (chiefly comprised of Polynomial Motion Approximation and Principal Component Analysis (PCA)).

Chapter 5: Simulation Results and Discussion This chapter demonstrates the reliability and high-accuracy of the PJVA machine learning approach on both the MSRC-12 dataset and my PJVA-20 dataset.

Chapter 5 is organized as follows. Section 5.2 on page 74 presents the results of the PJVA machine learning approach on my PJVA-20 dataset, including tests of several algorithms including SVM Poly (Section 5.2.1 on page 74), and SVM RBF (Section 5.2.2 on page 74), and Random Forest (Section 5.2.3 on page 76). The high accuracy results of the RF algorithm ensured its use in the final PJVA machine learning approach. These tests also demonstrate the high recognition capabilities of the PJVA approach on static gestures, which has proven to be difficult on the MSRC-12. Other machine learning approaches require the extraction of motion feature vectors to capture subtle movements from gesture samples, to better differentiate between separate gestures and improve recognition. PJVA, on the other hand, achieved a 98.5% accuracy on just static gestures alone. This result primarily serves to demonstrate the quality of the samples captured using charades as a modality of instruction.

Finally, Section 5.3 on page 81 presents the results of the PJVA machine learning approach on the MSRC-12 dataset. By comparing these results to those of Zhang et al. [2] in particular, which were the best results on the MSRC-12 at the time of testing, it can been seen that the PJVA system has advanced the state-of-the art in gesture recognition.

Chapter 6: Conclusions and Future Work In this chapter, potential future directions for this research are discussed and key contributions are summarized. The thesis is concluded with a summary of the work completed to date, a discussion of the contribution to the gesture recognition sector, and a call for action with next steps to pursue to advance this research.

Appendix A: Early PhD Research in Depth Based Hand Gesture Recognition This section provides a brief summary of a collection of original experiments related to hand gesture recognition using a homemade depth camera and infrared (IR) emitter. The approach applies Fourier shape descriptors with SVM. These systematic experiments include my early research into gesture recognition in the literature, particularly considering the limitations of database creation and classification approaches.

Appendix B: Potential for Alternative Inputs These are 3 alternative gesture based inputs, the pointing display, the Gesture Studio mouse emulation, and anchored 3D buttons.

Appendix C: Gesture Studio Features This Appendix details all of the features of the Gesture Studio software, including instructions for use and screen captures of the Gesture Studio site and portal, which are primarily used by researchers and academics.

Appendix D: Gesture Studio Cloud Benchmark These are the benchmark results of Gesture Studio Cloud training gestures for Minecraft, the PC video game. These gestures allow the game to be played entirely through body gestures. The whole process, from recording the gesture to playing the game, takes just under 12 minutes.

Chapter 2

Literature Review Gesture Recognition

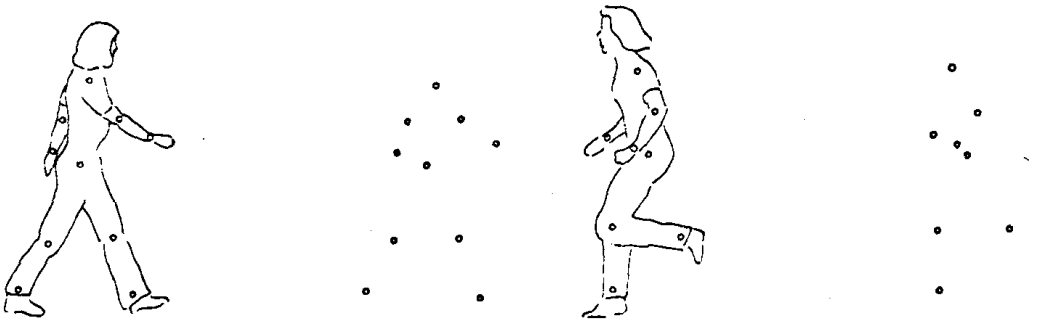
"Look wise, say nothing, and grunt. Speech was given to conceal thought."

- Sir William Osler

2.1 Introduction

With the enormous growth in gesture recognition applications for mainstream use —– such as gaming, security, and health care —– approaches for collecting accurate gesture data has become increasingly more important. To solve gesture recognition problems, researchers must collect data that contains examples of movements and their associated gesture label. Researchers in this field typically use human subjects as research participants to generate datasets used to train and test machine learning systems.

A little more than 40 years ago, Swedish psychologist Gunnar Johansson was the first researcher to raise the possibility of recognizing gestures by deconstructing the motion of the human body into features of gestures. He created an experiment that involved attaching small reflective bulbs to the joints of a person dressed entirely in black and asking them to perform specific movements inside a dark room (e.g., walking up the stairs and performing other gestures). By recording these movements, Johansson captured footage containing gestures consisting of 10



(A) (B)

FIGURE 2.1: This figure is from the ground-breaking experiment proposed by Johansson in 1973. Johansson measures and confirms that 150 ms is all it takes for the humans to register a few moving dots as a moving person, demonstrating onlookers' capacity to effectively connect spatial and motion information to semantic concepts known as 'gestures'

white bulbs moving against a black background, which was a clear enough image for onlookers to recognize the shape as a human participant performing several recognizable gestures [3, 4].

This study demonstrated that 10 bulbs moving in space and 150 ms of viewing time were enough for an observer to recognize that the dots of light are forming a human body, indicating that the human brain has a highly specialized capacity for recognizing gestures from basic movements. This research opened the door for other researchers to begin investigating the possibilities of attaching sensors attached to the joints of the human body (GDPs) to create robust gesture samples, enabling researchers to generate a virtual 3D skeleton of these GDPs to track these gestures for gesture recognition. As such, Johansson is a forerunner of contemporary techniques for sampling gestures for gesture recognition and computer learning.

The goal of capturing gesture samples from research participants is two-fold: to collect data that is a) correct in terms of comprising all movements necessary to recognize a gesture, and b) has an appropriate level of coverage for the machine learning system to cope with a wide array of users and their corresponding abilities [5]. This means that there must be a) appropriate examples of desired gestures (correctness) and b) the dataset must include common, desired variants of the particular movements associated with the gestures (coverage). Achieving both of these goals is important because the appropriate coverage ensures the resulting machine learning system is representative of the target population (in this case, as many body types as possible) while also determining the correct movements which "best match people's common stereotypes of the semantic concept associated with a gesture" [6].

The rest of the chapter is organized as follows. Section 2.2 on page 13 reviews the definitions of the word 'gesture' in the literature, which has guided subsequent studies into gesture recognition. Section 2.3 on page 15 will then discuss approaches to collecting gesture data in order to create gesture samples for gesture recognition research. Section 2.4 on page 21 lists several datasets that have been developed in the gesture recognition literature. Section 2.5 on page 23 introduces the benchmark dataset MSRC-12, discussing the instructions used to convey the gestures to the research participants, as well as the techniques typically used for gesture recognition and machine learning on this dataset Section 2.6 on page 40 reviews the gesture recognition literature to find other examples of other studies that use charades as a data gathering modality.

2.2 Definition of a Gesture in the Literature

Several academic researchers have put forward definitions of the human gesture, which are relevant for designing gesture recognition systems capable of extracting human motion features from gestures samples.

Figure 2.2 shows Kendon's Gesture Continuum [7], which was an early step to understanding and describing the nature of the human gesture. Kendon defined five types of gesture, as follows:

1. Gesticulation represents the unprompted actions of the hands and arms which accompany speech.

2. Language-like gestures are movements that are part of speech, and replace spoken words or phrases.

3. Pantomimes are gestures that portray actions or objects, and may or may not be associated with speech.

4. Emblems are culturally specific gestures, such as a "thumbs up" for "good," or holding up the index and middle fingers in a "V" for victory (during World War II) or peace (since the 1970s).

FIGURE 2.2: Kendon's Gesture Continuum

5. Sign language is a formalized language system which uses manual communication, body language, and lip patterns instead of sound to convey meaning.

As we move to the right in 2.2, the accompaniment of speech with gesture is reduced, impulsiveness decreases, and language properties and social guidelines increase.

Going into further depth in the classification of gestures, McNeill demarcates four sub-types of gesticulation [8, 9], as follows:

1. The "iconic" type represents figurative gestures that depict features of the action, object, or event that is being described. For example, when trying to describe a missing object, often people will say "Have you seen this thing?" followed by an action that physically indicates the object's size.

2. The "metaphoric" type represents familiar metaphor, rather than describing the action, object, or event directly. For example, when you let someone in ahead of you, you would move your hand outward while rolling your wrist, or wave them in.

3. The "beat" is a brief, unformed gesture that is usually related to the stressing of a word. For example, when someone is emphasising a question, they might extend both hands in a quick chop.

4. The "deictic" is a gesture that involves pointing at a person or object in relevance to space or time. For example, when you need a tissue you could just point to the box just out of reach.

McNeill's efforts to sub-categorize gesticulation proved to be a foundational contribution to the future of gesture recognition literature. A seminal paper in the gesture recognition field, written by Fothergill et al. [5], used these 'iconic' and 'metaphoric' classifications when building the MSRC-12 dataset, in order to train computers how to distinguish between different types of gestures.

2.3 Capturing Skeletal Data to Create Gesture Samples

Methods for capturing skeletal data to create gesture samples has been explored extensively in the Computer Vision literature, using both tracker and visual processing methods to capture data from a research participant. Vision-based approaches to data capture aim to provide an accuracy as close as possible to tracker-based approaches, yet several difficulties with this approach remain such as effectively separating the participant from the environment and acquiring a 3D image of the participant from a 2D image.

2.3.1 Tracker-Based Approaches for Data Capture

Since the inception of gesture recognition technology, data sampling has been accomplished by using trackers, sensors, and even special gloves worn by the research participant.

"Trackers" are any type of device that is mounted or attached to a research participant to monitor his/her gestures, and can be considered a transmissive approach to gesture recognition. Trackers are most commonly used in the film industry because they provide a very accurate description of body movement. There are many types of trackers available — in the gesture recognition literature, optic trackers (Figure 6.4) as used by Kawashima et al. [10], transmissive trackers [11] [12], and colour trackers [13] are most commonly used, which have a higher degree of accuracy and also allowing for detailed 3D texture tracking [14] (see 2.3a for an example of passive trackers in a real world application).

Researchers and developers have also made use of ultrasonic trackers, e.g., Brandl et al. [15] uses these types of transmissive trackers to accurately locate the two points of a pen. When a research participant wears ultrasonic transmitters connected to critical joints of the body area, stationary ultrasonic receivers combined can position the transmitters in three dimensions to obtain real time gesture data. By increasing the number of transmitters attached to the gesturer, one can obtain higher accuracy results.

Tracking visual markers requires multiple cameras and a fixed capture area, because the cameras are stationary. For instance, three or more markers are required to collect 3D data with

high accuracy, e.g., to capture rotational movements. There are four types of optical marker available:

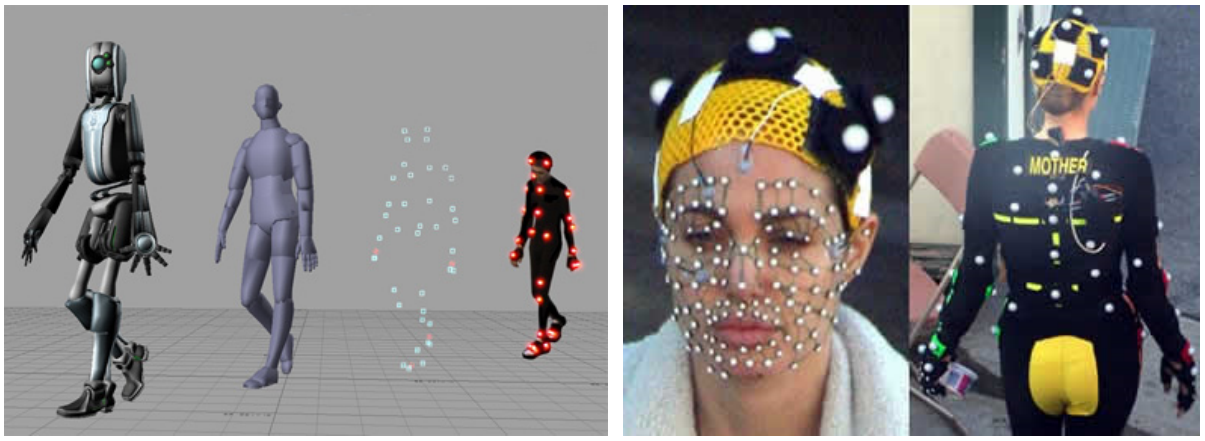
1. Passive Markers: A passive marker is coated with a retro-reflective material that allows light near the camera lens to be reflected back. Adjusting the camera's sensitivity allows most cameras to narrow the viewable footage to just bright markers, ignoring skin and clothing altogether. Centroids of each marker are estimated in a two-dimensional image. Greyscale values of each pixel can provide sub-pixel precision.

2. Active Markers: An active marker is made from LEDs. Triangulation of markers is achieved by illuminating one LED at a time very quickly. Multiple LEDs can be illuminated simultaneously; however, a complex software is required to identify the relative position of each marker. Unlike passive markers, the LEDs do not reflect light; rather they are powered to generate the necessary amount of light. This type of system is better suited for larger capture areas, since the LEDs can generate light that is visible from greater distances than passive markers. However, identifying each marker requires additional processing.

3. Time Modulated Active Markers: A Time Modulated Active Marker uses a quicker method of strobing a single marker or, when considering multiple markers, the markers are tracked over time. By modulating the pulse width, the marker ID can be identified. LEDs are connected to onboard processing and are wirelessly synchronized. The benefit of these markers is that unlike passive markers, they can be applied outdoors.

4. Semi-Passive Imperceptible Markers: This system uses photosensitive marker tags to decode the optical signals. By using photo-sensors, the tags can calculate illumination, location, orientation, and reflectance.

The hands are a critical part of the human body to map, since they are very expressive and are used for many important gestures. To fully capture hand motion using trackers, a data glove, fitting over the hand and embedded with sensors, can faithfully record all the movements of the fingers and wrist. There are many benefits to using this approach: it is easy to use, no line-of-sight is required, and it generates high-quality data. The drawbacks are that the glove



(A) Illustration of motion capture system from markers to point cloud, to model, to rendered avatar

(B) Face gesture tracking with markers, Angelina Jolie from the 2007 film *Beowulf*

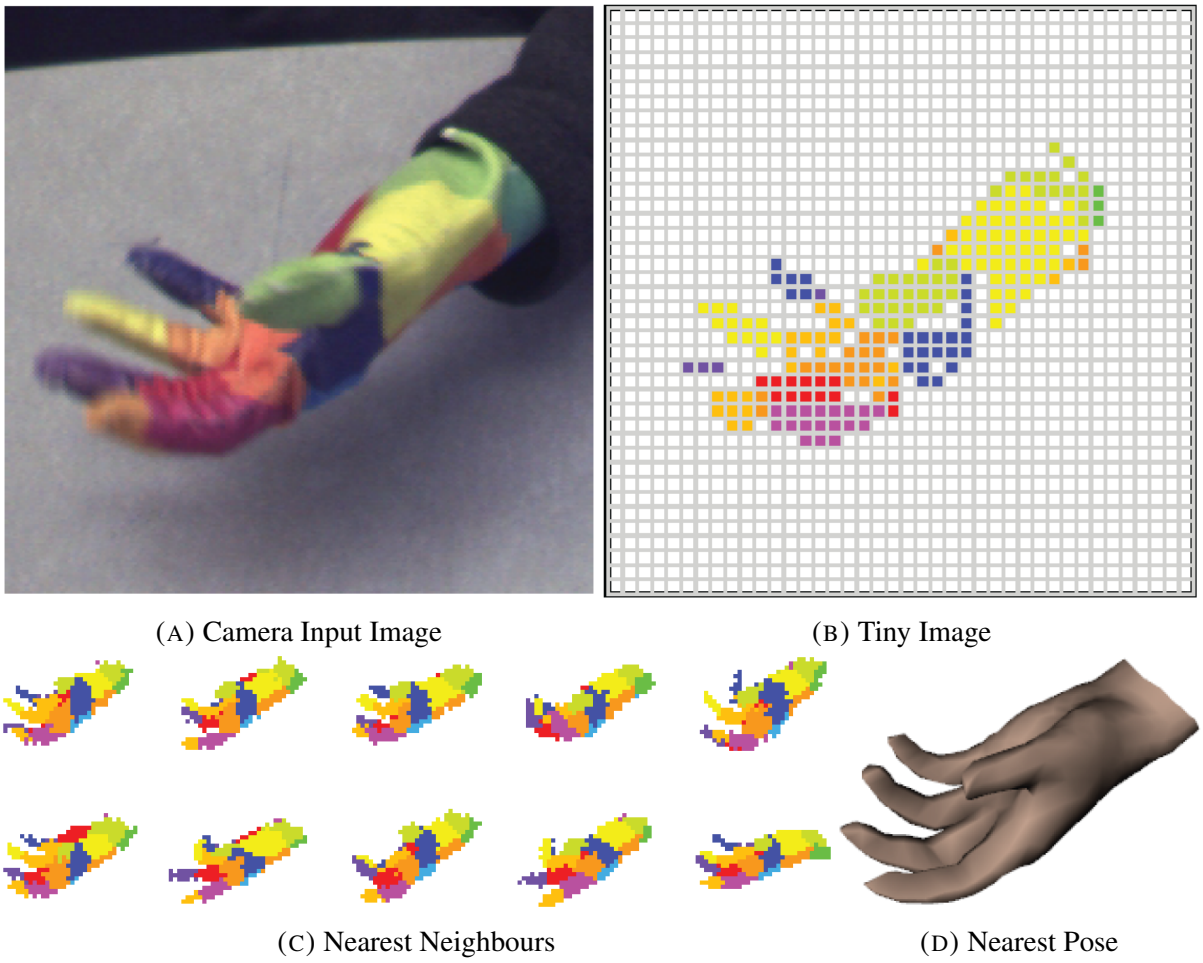
FIGURE 2.3: Example of passive markers used on a motion capture suit

is intrusive and awkward to use, calibration can be difficult, and if the glove is tethered it can reduce range.

Despite these drawbacks, many experiments have used data gloves to create complex gesture-based interfaces. Zhang et al. [16] captured data on hand gestures through multi-channel surface electromyogram (EMG) sensors and a 3D accelerometer. For a set of 18 different gestures, each trained with 10 repetitions, the average recognition accuracy was about 91.7% in real applications. In another study, Chen et al. [17] used only a 2D accelerometer in addition to EMG and found between a 5% to 10% improvement in recognition.

Another technique in the literature is to use the accelerometer in a Nintendo Wii remote, which can be synced to gesture recognition software to track movement and record gestures. Most of these papers use this technology to train new gestures and store them in a database, but one paper notably applied reflective stickers to a user's fingers to make the IR from the Wii remote reflect [18].

Though each type of tracker has a different approach, the idea is consistently the same: if you can model the gesture in 3D, you can identify and categorize it, enabling a computer to learn to recognize these gestures.



(A) Camera Input Image (B) Tiny Image

(C) Nearest Neighbours (D) Nearest Pose

FIGURE 2.4: Hand pose recognition from Wang, based on nearest neighbour calculation, which inspired the gesture recognition approach for PJVA

2.3.2 Vision-Based Approaches for Data Capture

For vision-based approaches for data capture, the biggest limitation is that gestures are complex and require 3D data that is very complicated to extract from a 2D image.

One possibility for addressing this is to approach the hand gesture problem by creating a 3D virtual model of the human hand and attempting to keep that synchronized with a high frequency multi-camera system. For instance, Downton et al. [19] created a system that tracks limbs through an "articulated generalized cylindrical human model," which shows promising results in dynamic gesture recognition. Although they only used a single monochrome camera, gesture recognition was accomplished through a perspective projection of the cylindrical model. The output of this system was kinematic specifications of the joints similar to those generated by the MSRC-12 and PJVA-20, which in this case were used for sign language analysis.

Furthermore, Lee [20] used a 3D hand skeleton model with 27 degrees of freedom. He used five restrictions on the human hand kinematics, creating constraints to help narrow the search space and localize results. In his experiments making a 3D hand skeleton model, Kuch [21] also employed six constraints, allowing for 26 degrees of freedom. The system was adapted to specific users and would extract motion feature vectors out of a long sequence of single-camera images, which is a similar technique to that used by PJVA (see Section 3.2 on page 46).

It is also quite common to see marked glove gesture recognition systems in the literature. Breuer et al. [22] describe a system that employs an IR time-of-flight range camera with the speed of up to 15 fps to measure 3D-surface points captured from the user's hand. The measured data is transformed into a cloud of 3D-points after depth keying and suppression of camera noise by median filtering is completed. The system has 7 degrees of freedom which can further be used to generate a classifier for gloveless hand tracking using raw depth data after applying a simple background segmentation algorithm.

When creating gesture recognition systems, it is important to make the systems as unobtrusive as possible so that people actually want to use them. Brandl et al. [15] has developed a gesture recognition system that can be seamless enough for daily use. Using a rear-projection setup that combines high-resolution pen tracking with simple hand gesture recognition for zooming and moving images, the group claims 1mm accuracy. The system can track multiple users within the camera range by using 8 industrial-grade IR cameras (4 behind the user, 4 in front of the user).

Of particular interest to the present study are Wang's 2008 [47] and 2009 [13] papers, in which he describes a system that can reconstruct the pose of a hand from single frames of the hand wearing a specially made multi-coloured glove (20 patches with 10 unique colours). His work is important to this dissertation as someone conducting breakthrough research using an image-based "anchored" approach, as opposed to orienting the algorithm using the distance between the camera and the glove (see Section 4.2 on page 59 for an analysis of these orientations).

The focus of the work was to provide more positions of freedom for the hand over other the cutting-edge techniques of the time. The glove's design is sufficiently distinctive that the pose of the hand is reliably distinguishable in a single frame. According to Wang, the use of a colour glove was inspired by advances in cloth motion capture, where dense patterns of coloured

Ref.	Author	Task	Method	Hardware	Speed(FPS/Hz)
[23]	Ahmad	Hand Tracking	Fingertip detection	Mono Camera	10-30FPS
[24]	Bulzacki et al.	Gesture Recognition	Vector nodal network	Mono Camera	40FPS
[25]	Baudel et al.	Gesture input for presentation	Interaction model and notation	Dataglove	N/A
[26]	Boehm et al.	Teaching dynamic gestures	Kohonen feature map to reduce data	N/A	N/A
[15]	Brandl et al.	Pen with Gesture	Special whiteboard, Pen, and Gestures	8 Cameras	Real-Time
[22]	Breuer et al.	Hand Gesture	IR camera	Mono Camera	15FPS
[27]	Cipolla et al.	Gesture Recognition	Marked fingertips	Mono Camera	25FPS
[28]	Cho et al.	Posture Recognition	Local shape property learning	Mono Camera	N/A
[29]	Darrell et al.	Gesture Recognition	Set-of-views model	Mono Camera	10FPS
[30]	Davis et al.	Hand Tracking	Fingertip detection model	Mono Camera	N/A
[19]	Downton et al.	Limb Tracking	3D cylindrical limb	Mono Camera	N/A
[31]	Elmezain et al.	Static Hand Gesture	HMM based with Trajectory	Stereo Camera	Real-Time
[32]	Etoh et al.	Hand Tracking	3D cylindrical hand model	Stereo Camera	N/A
[33]	Fels et al.	Hand gestures to speech	Adaptive neural network	N/A	N/A
[34]	Fukumoto et al.	Pointing	Finger detection and virtual projection origin	Stereo Camera	Real-Time
[35]	Freeman et al.	Gesture Recognition	Orientation histograms	Mono Camera	Real-Time
[36]	Hubbard et al.	Voice with Gesture	Camera, Microphone, and MRI	Mono Camera	N/A
[37]	Kervrann et al.	Hand Tracking	Stochastic deformable model	Mono Camera	N/A
[38]	Kjeldsen et al.	Gesture Recognition	ALVINN	Mono Camera	N/A
[39]	Krueger	Object Manipulation	Silhouette	Mono Camera	30FPS
[21]	Kuch et al.	Gesture Recognition	3D NURBS hand model	Mono Camera	3-30FPS
[40]	Kim et al.	Two Handed Marker Glove	Markers, RF transmitters	N/A	N/A
[18]	Lee	Wii remote finger recognition	Reflectors of fingers viewed through Wii	N/A	100Hz
[41]	Maggioni	Gesture Recognition	Marked glove	Mono Camera	25FPS
[42]	Schlenzig et al.	Gesture Recognition	Zernike moments	Mono Camera	0.5FPS
[43], [44], [45]	Segen et al.	Gesture Recognitiong	Silhouette edges	N/A	Real-Time
[46]	Takahashi et al.	Japanese Kana manual alphabet	Real-time 3D model	Dataglove	N/A
[13][47]	Wang et al.	Finger and Hand Recognition	Colour Glove Pose Estimation	Mono	30Hz
[48]	Yoon et al.	Gesture Recognition	HMM geometric parameters	Mono Camera	5FPS
[16]	Zhang et al.	16 hand gestures	EMG and 3D accelerometer	N/A	1kHz
[49]	Väänänen et al.	Teaching static gestures	Neural network	N/A	N/A

TABLE 2.1: Research focused on Hand Gestures Recognition or Tracking

markers enable precise capturing of deformations [14, 50, 51]. Wang collected a set of 18,000 finger configurations using the Cyber-glove II hand motion capture system. The image of each hand pose was indexed as a tiny $(40X40)$ rasterized image (see Figure 2.4). This was done to speed up the system's searching time and to make each pose representable by a standardized description.

For calculating nearest neighbours —– which is a calculation used to model the distance, shape, and depth of the hand —– Wang used an approach similar to how Hausdorff distance [52], which measures how far two subsets of a metric space are from each other. Wang took the average of two divergences when calculating a "symmetrical" neighbour's distance: one divergence from the database to the query and one divergence from the query to the database. In Wang's approach, the camera was geometrically calibrated by establishing a ground plane (e.g., a desk's surface) for successful data captures.

Wang's "anchoring" approach for calculating nearest neighbours had a significant influence on this dissertation. I validated that this calculation can be used to calculate nearest neighbours without the coloured glove, which is not required to accurately capture robust gesture data. This research in particular was a motivating force for advancing the state-of-the-art in gesture recognition, as it demonstrated the potential for the creation of a novel, high-accuracy motion extraction approach (AMFE) to recognize a gesture in as low as a single frame, without using trackers whatsoever (see Section 4.2 on page 59).

2.4 Gesture Datasets in the Literature

The following is a list of several gesture datasets in the literature. The most relevant dataset to this study, the benchmark MSRC-12, will be introduced and analyzed further into the Literature Review in Section 2.5 on page 23.

The KTH dataset [69]. This dataset contains video of six types of human actions (walking, jogging, running, boxing, hand waving, and hand clapping) performed multiple times by 25 subjects in four different scenarios: outdoors (s1), outdoors with scale variation (s2), outdoors with different clothes (s3), and indoors (s4). 2.5 shows examples of the dataset. The dataset

Ref.	Name	Model Parts	Approach
[53]	Agarwal et al.	21	Direct nonlinear regression against shape descriptor vectors extracted from image silhouettes
[54]	Mori et al.	10	Estimating human body configurations in 3D based on matching with multiple 2D exemplars
[55]	O'Rourke et al.	N/A	Feedback loop between high and low levels on segmented image (pioneering research)
[56]	Hogg	14	WALKER model,person is represented by the series of hierarchical levels (pioneering research)
[57]	Yamamoto et al.	4	Motion calculated optically by several points and compare to model (pioneering research)
[58]	Ramanan et al.	10	2D view of the human body as a puppet of colored and textured rectangles
[59]	Siddiqui	7	Seperate approaches for 2D camera, 3D camera and range camera
[60]	Sidenbladh et al.	10	Implicit probabilistic model of human motion through texture synthesis
[61]	Grest et al.	13	depth images using an iterative closest point algorithm
[62]	Felzenszwalb et al.	10	pictorial structure models to find instances of an object
[63]	Rogez et al.	15	3D Mocap data with histograms of orientated gradient and random trees
[13]	Wang et al.	26	Colour glove and tiny image distance through nearest nighbour
[64]	Okada et al.	13	Feature vector of gradient orientation histograms with SVM
[65]	Knoop et al.	10	Iterative closest point and 2D 3D fusion
[66]	Navaratnam	16	Semi-supervised multi-valued regression
[67]	Sigal et al.	10	Probabilistic models for the temporal evolution of each loose limb
[68]	Bregler et al.	10	Region-based motion estimation framework and kinematic chain constraints

TABLE 2.2: Skeletonization (articulatory body model, or pose estimation) Approaches

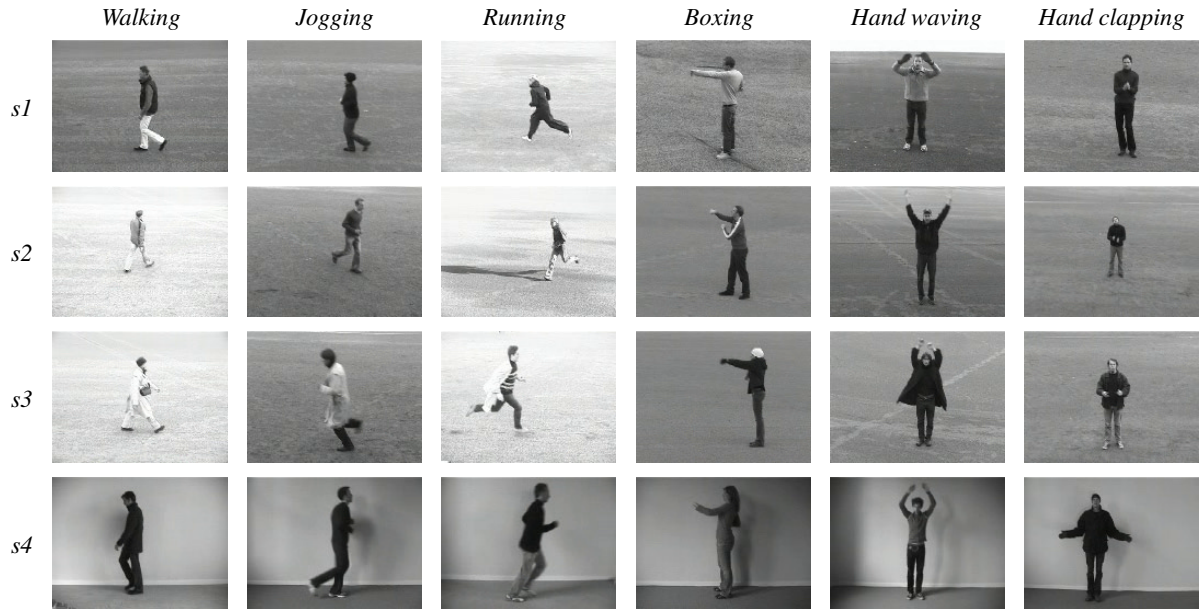


FIGURE 2.5: The KTH dataset includes a video database of 6 gestures preformed in 4 different environments

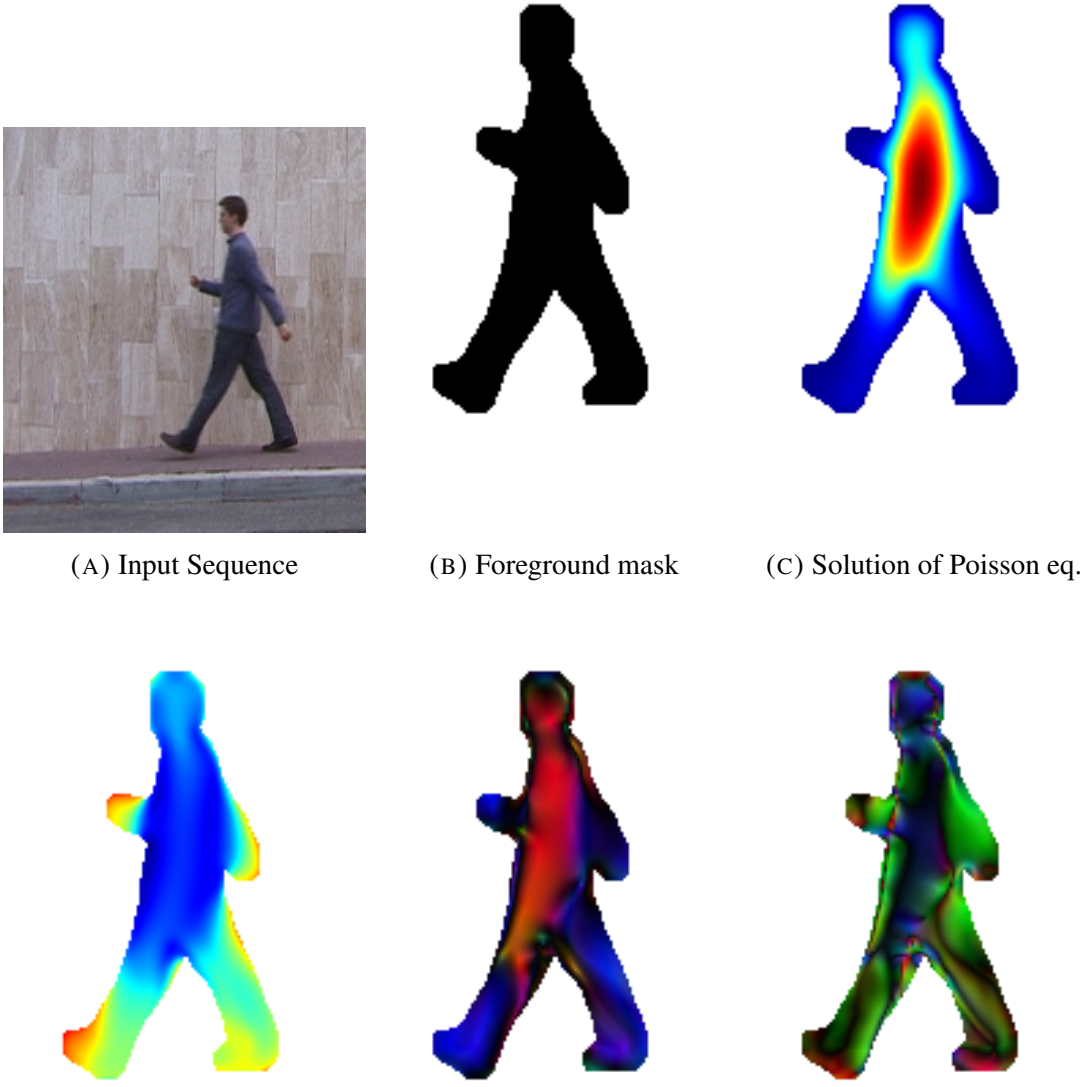
contains 2391 sequences which were taken on homogeneous backgrounds with a traditional video camera capable of 25fps. The dataset was downsampled to the spatial resolution of 160 by 120 pixels and each action has an average length of four seconds.

The Weizmann dataset [70]. This dataset contains 10 walking gestures: normal walking, walking in a skirt, carrying a briefcase, limping, occluded legs, knees up, walking with a dog, sleepwalking, swinging a bag, and occluded by a pole. See 2.6 for examples of these gestures.

The IXMAS dataset [71]. The dataset is sampled using 5 camera angles, and contains 11 types of full body gestures: check watch, cross arms, scratch head, sit down, get up, turn around, walk, wave hand, punch, kick, and pick up. See Figure 2.7 for examples of these gestures.

2.5 MSRC-12 Benchmark Dataset

The MSRC-12 research conducted by Fothergill et al. [5] is a study that developed a benchmark dataset in the gesture recognition literature. In their own words, this dataset "consists of sequences of human skeletal body part movements (represented as body part locations) and the



(A) Input Sequence (B) Foreground mask (C) Solution of Poisson eq.

(D) Space-Time "Saliency" (E) Measure of "Plateness" (F) Measure of "Stickness"

FIGURE 2.6: Walking Sample from the Weizmann dataset

associated meaning that needs to be recognized by the system" [5]. The present study will refer to this skeletal body as a "3D skeleton" that represents the positions of the joints of the human body as these joints move throughout the performance of a gesture.

The MSRC-12 dataset includes data from 30 people and is comprised of 20 three-dimensional joints, captured for 12 different gesture classes. The participants were 60% male and 40% female, of which 93% were right-handed, the average height was 5'8", and the average age

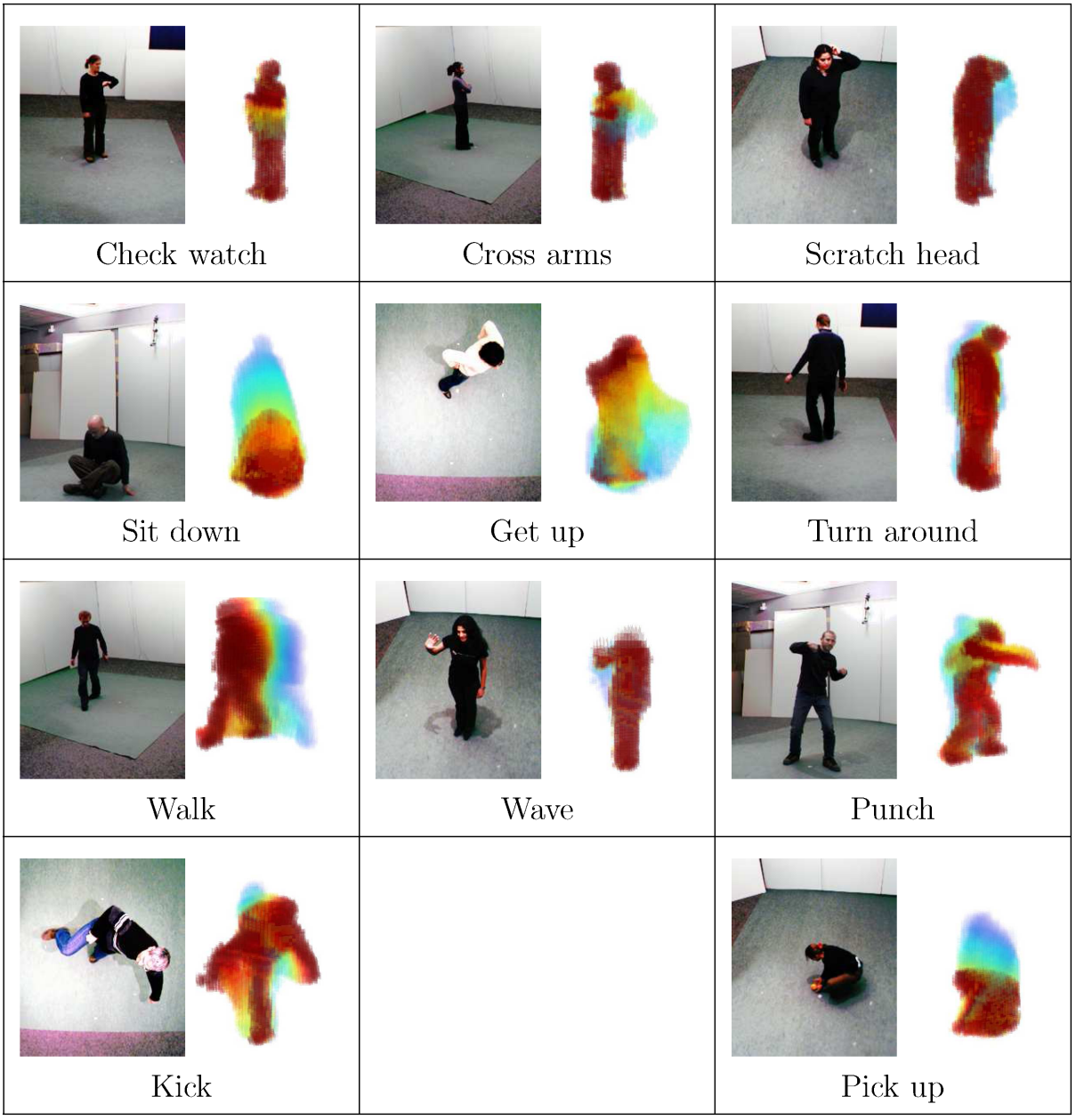


FIGURE 2.7: IXMAS dataset includes 11 actions, performed by 10 actors.

was 31. The data in the MSRC-12 dataset was captured using the Kinect camera, which offers researchers the saved values for each GDP on a tracked 3D skeleton. This allows researchers to obtain positional data from the gesture, known as motion feature vectors, using the positional relations between GDPs.

Research participants were asked to stand and face a 30" LCD TV with a Kinect sensor in front of it. "When they indicated they were ready, the first gesture's instructions appeared on the screen in a PowerPoint slideshow. At the top of each slide, the application category (e.g. music

player) and gesture outcome were displayed... and below that the instruction was placed using the appropriate modality... Questions were not addressed by the experimenter and instead the participants were told to 'do what they wanted' " [1]. Data begins to be recorded as a research participant walks into position, just before beginning the instructed gesture. The recordings are not trimmed, and also include the person walking out of camera view after the gesture is performed.

Fothergill et al. divided the gestures of this dataset into two categories, which are based on McNeill's categories of gesticulation: "The first is iconic gestures —– those that imbue a correspondence between the gesture and the reference. The second was metaphoric gestures —– those that represent an abstract content" [5]. In other words, iconic gestures are recognized easily by most people, representing an action that a participant could typically perform with a familiar motion, e.g., 'crouch,' 'shoot with a pistol,' 'kick to attack,' etc. Metaphoric gestures are abstract so often require further explanation, e.g., 'wind up the music' or 'protest the music,' for which any number of gestures could apply.

Fothergill et al. divided the 12 gestures classes of the MSRC-12 dataset into two separate categories, MSRC-12a comprised of 6 iconic gestures classes and MSRC-12b comprised of 6 metaphoric gesture classes. The iconic gestures are meant for a user to play a first person shooter game, and the metaphoric gestures are meant to convey instructions to a music player. See Table 3.1 for a list of all the statistics of the data collected in the MSRC-12.

However, although all gestures in the benchmark MSRC-12 are categorized and labelled by Fothergill et al., in some cases the label does not represent the actions performed (i.e., right push sometimes is done with the left hand, or in some other cases with an entirely different gesture). In fact, one other case —– "G11_Beat_both" sample, ID 40 —– is actually a kick gesture, which is a critical miscategorization. These observations are corroborated by Gomes et al. [72].

Such miscategorizations in the MSRC-12 is one of the reasons why I experimented with a different modality of instruction, gesture sampling approaches, and gesture recognition techniques. Sampling with charades and using PJVA can help to avoid system errors (see Section

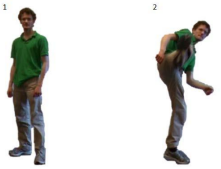
Gesture Outcome	Descriptive Text	Static Images
Crouch or hide	*Squat down or crouch*	
Shoot with a pistol	*Stretching your arms out in front of you and holding your hands together to form a pistol, make a recoil movement*	
Throw an object such as a grenade	*Using your right arm, make an overarm throwing movement*	
Change weapon	*Reach over your left shoulder with your right hand and then bring both hands in front of your body as if you are holding something*	
Kick to attack an enemy	*Karate kick forwards with your right leg*	
Put on night vision googles to change the game mode	*Bring your hands up to your eyes as if they were googles*	

TABLE 2.3: MSRC-12 Iconic Gestures for first-person shooter application.

3.2) on page 46, as I will demonstrate when testing on the resulting PJVA-20 dataset to determine whether the machine learning results in the literature would achieve higher accuracy on more suitable gesture samples for gesture recognition applications.

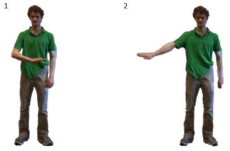
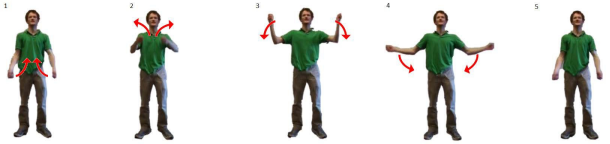
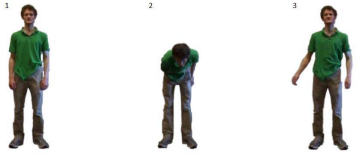
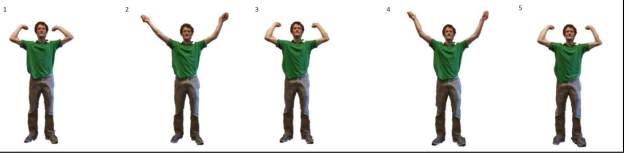
Gesture Outcome	Descriptive Text	Static Images
Start music/raise volume	*Raise outstretched arms*	
Navigate to next menu	*Slide right hand, palm down in front of you, from left to right*	
Wind up the music	*Make circular movements with both arms, in front of your body, clockwise with right hand and counter-clockwise with left hand*	
Take a bow to end the session	*Bend forward at the waist, pause and come back up again*	
Protest the music	*Pause and rest your hands on your head*	
Lay down the tempo of a song	*Beat the air with both your hands*	

TABLE 2.4: MSRC-12 Metaphoric Gestures for music player application.

2.5.1 Modality of Instruction for Capturing Gesture Samples

To source the samples of movements necessary to train gesture recognition systems, the first step is to provide research participants with appropriate instructions. These instructions can be comprised of several different semiotic modalities including text, images, video, and different combinations of these modalities

Yet there is little work in the Human-Computer Interaction (HCI) and Computer Vision (CV) literature on the problem of how to design these instructions to specify which movements need to be performed by research participants for accurate gesture recognition. Fothergill et al. have stated that "there has been no study of what biases are introduced by different modalities on correctness and coverage" [5]. Many researchers simply apply machine learning algorithms to existing datasets. Most do not even consider the important question: "what is the most appropriate semiotic modality of instructions for conveying to research participants the movements that the system developer needs them to perform" [5]. As collecting gesture data is the crucial first step of creating a robust gesture dataset, this requires more examination to ensure that appropriate directives are conveyed to research participants.

To effectively convey information to research participants, Fothergill et al. provided participants with "three familiar, easy-to-prepare instruction modalities and their combinations that did not require the participant to have any sophisticated knowledge: (1) descriptive text breaking down the performance kinematics, (2) an ordered series of static images of a person performing the gesture with arrows annotating as appropriate, (3) video (dynamic images) of a person performing the gesture" [5]. Beyond these three modalities, Fothergill et al. added two new combinations to include (4) Images+Text and (5) Video+Text.

To assess which modality was superior to convey instructions to research participants, Fothergill et al. tested for the intra-modality generalization performance of all of these modalities, "obtain[ing] five F-scores, one for each modality, and each being an average over all 10 repetitive repetitions and 12 gestures" [5]. To analyze this data, Fothergill et al. compared the F-scores from 10 runs with one-way, between-subjects ANOVAs between the five conditions: Text, Images, Video, Images+Text, and Video+Text.

Although Fothergill et al. did not find a strong statistical difference between these instructional modalities for generating suitable data, they did discover that the instructional modality that provided the most information to research participants —– Video+Text —– yielded slightly less sense of freedom of movement than Images+Text. Importantly, they also found that "in terms of capturing natural variation less information was optimal" and that allowing participants to feel more free in their choices helps generate desirable coverage and "may instill confidence that encourages more controlled improvisation" [5].

This finding is corroborated by a review of the interviews conducted following the Fothergill et al. study. After collecting data for the MSRC-12, two preliminary questionnaires were used followed by an open-ended interview. The questionnaires consisted of 11 questions regarding each participant's perception of his/her performance as well as four final questions asking the participants to rank their ability to perform each gesture on a Likert scale.

Participants generally related that the videos were the clearest modality for providing instructions; however, participants also explained that video footage and static image were not necessary for understanding the iconic gestures, since they felt it was fairly obvious what was being requested of them due to common experiences in the real world.

Also, when the discussion turned to enabling more freedom for the research participant to perform the gesture as they saw fit, the participants generally agreed that less information allowed them more freedom to effectively perform metaphoric gestures. This is primarily a reaction to the use of videos as an instructional modality, which is perhaps too prescriptive by showing the participant exactly what to do, whereas other modalities such as text are more open to interpretation. As such, the text approach may create samples that provide better coverage for the gesture recognition system.

Additionally, in the interest of keeping instructions open to interpretation, I believe that the instructions should only consist of 1-2 words, i.e., just the semantic concept without any further instruction. I believe the use of descriptive text in the MSRC-12 is only necessary for clarifying some metaphoric gestures (e.g., the 'Protest the Music' gesture outcome for Fothergill et al.'s music player application, for which the required actions are not immediately obvious). However, even in the metaphoric category of the MSRC-12, there are some gestures that do not

require such prescriptive instructions (e.g., 'Take a bow to end the session,' which is a common, fairly obvious gesture that would not need extra instructions—in fact, this may limit the coverage of the resulting data).

2.5.2 Extracting Motion Features from MSRC-12 Static Gesture Samples

A set of GDPs is a single frame of a gesture sample, representing a static pose of a gesture, which can be enough data for accurate gesture recognition. As such, motion feature extraction is not always necessary for gesture recognition applications; however, similar poses can be present in multiple static gestures and can thus confuse classification algorithms when discerning between different gesture classes. For example, with static gesture data alone, a gesture recognition system would be unable to distinguish between a rotating limb moving clockwise and counterclockwise, because both gestures share identical poses. Thus, when creating a gesture recognition system, extracting motion features can increase accuracy by tracking subtle features that differentiate between similar gesture classes.

A study by Zhang et al. [2] demonstrates the process for extracting motion features from the MSRC-12 using CMFE. In the original MSRC-12 dataset, each frame of gestures is recorded as the absolute position of 20 joints (GDPs) of the human body in xyz-coordinates —– 60 data total per frame. In order to extract informative motion feature vectors from these coordinates, according to Zhang et al., four kinds of factors can be considered as the possible components of a motion feature vector:

- 3 xyz-coordinates per joint (60 total)
- 3 xyz-velocities per joint (60 total)
- 35 joint angles
- 35 joint angular velocities

To clarify the definition of these terms, I will provide the following explanation from Zhang et al.: "The xyz-velocities are straightforwardly defined as the difference between xyz coordinates

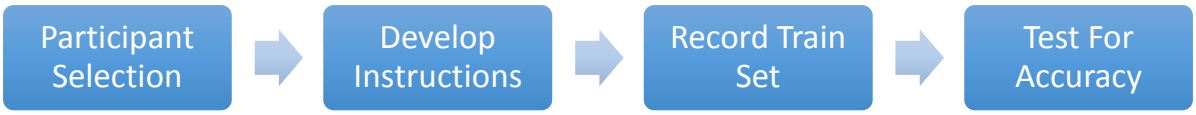


FIGURE 2.8: Fothergill et al. 2012 Experiment Process

of corresponding joints between each pair of adjacent frames. The joint angle is simply the angle between the two segments on either side of the joint... [and] the joint angular velocity is the rate of change of joint angle, which is computed by the difference of the corresponding joint angle in each pair of adjacent frames" [2].

Table 2.5 shows the relationship between different features and accuracy. It shows that, among single features, joint angle has the highest accuracy; and from double features, the combination of xyz-velocity and joint angle has the highest accuracy; and of triple features, xyz-velocity, joint angle, and joint angular velocity have the highest overall accuracy. The joint angular velocity is the rate of change of joint angle, which is computed by the difference of the corresponding joint angle in each pair of adjacent frames.

Feature in Use	**Feature Fixed**	**Accuracy**	**Real Proportion**
1	ϕ	40.3794%	149/369
2	ϕ	23.5772%	87/369
3	ϕ	65.3117%	241/369
4	ϕ	23.5772%	87/369
3 + 1	3	56.6396%	209/369
3 + 2	3	73.9837%	273/369
3 + 4	3	57.4526%	212/369
3 + 2 + 4	3 + 2	79.4038%	293/369
3 + 2 + 1	3 + 2	63.4146%	234/369
3 + 2 + 4 + 1	3 + 2 + 4	63.4146%	233/369

* 1 is xyz position, 2 is xyz velocity, 3 is joint angle, 4 is joint angular velocity.

TABLE 2.5: Zhang feature class selection, where 4 different type of gesture feature are compared separately and in combination with each other.

In this equation, the data captured from the Kinect is oriented relative to the research participant's distance from the camera. This technique —– henceforth known as 'camera motion feature extraction' (CMFE) —– captures the motion features of a gesture sample by "placing an imaginary joint in (0, 0, 0) in world coordinates," which creates an artificial 21st joint to

orient the motion feature extraction process [2]. However, I discovered that this orientation can become problematic for subsequent machine learning. To improve the calculation for extracting motion feature data, I developed a normalization technique called AMFE relative to a single joint on the human body: the centre of the waistline (see Figure 4.4). This technique is an entirely new perspective for motion feature extraction.

After applying AMFE, Principal Component Analysis (PCA) can be used for dimensionality reduction, in order to transform a linear projection of high dimensional data into a low dimensional subspace, such that the variance of the projected data is maximum and the least square reconstruction error is minimized [73] [74]. When testing on the MSRC-12 dataset, I will make use of PCA and polynomial approximation to reduce the dimensions of the skeletal gesture data into eigenvectors to improve both the speed and accuracy of gesture recognition applications (see 4.4).

2.5.3 Machine Learning Algorithms on the MSRC-12

To classify and label the resulting motion features extracted from gesture samples, researchers can apply machine learning algorithms such as HMMs [75],[76],[48],[77],[78],[79],[80],[73], [81], Bayesian networks, dynamic time warping [82],[83],[84], [85],[86],[85], and/or a self-organizing map (SOM) [11] to design dynamic gesture recognition systems that can extract and categorize motion feature vector data for gesture recognition.

I will focus on four machine learning techniques that have been used to categorize motion feature vectors from the MSRC-12 dataset for machine learning: Hidden Conditional Random Field (HCRF), Hidden Markov Model (HMM), SVM Classification, and Random Forest (RF).

2.5.4 Hidden Conditional Random Field (HCRF)

Song et al. [87] conducted a study to test an HCRF algorithm on the MSRC-12 dataset. Instead of testing for all classes simultaneously, they divided the MSRC-12 into two 6-class problems, MSRC-12a (Iconic) and MSRC-12b (Metaphoric).

They evaluated the "distribution-sensitive prior" for learning with imbalanced data, comparing it to three baseline methods: (a) learning with imbalanced data without using the distribution-sensitive prior ($k = 0$), (b) learning with balanced data with random undersampling, and (c) random oversampling. Song et al. studied how sensitive the classification performance is to the degree k of the distribution-sensitive prior. They used the $\alpha = 1$ version of the datasets to simulate highly imbalanced data. In their methodology, Song et al. varied the degree $k = [00.512]$ of distribution-sensitive prior, where $k = 0$ means no distribution-sensitive prior was used.

Song et al. set the number of samples per class as the minimum (and the maximum) of N_i^y's, and discarded (and duplicated) samples at random to create an even sample distribution. Song et al. validated the two hyper parameters of Hidden Conditional Random Field (HCRF): the cardinality of the latent variables $|H| = [6810]$ and the $L2$ regularization factor $\sigma^2 = [110100]$. They then selected, for each split and for each k, the optimal hyper parameter values based on the $F1$ score on the validation split. Song and colleagues performed 5-fold cross validation, and the L-BFGS optimization solver was set to terminate after 500 iterations. Figure 2.9 shows the results of their study on the two 6-class classification problems of the MSRC-12. On the MSRC-12a dataset of iconic gestures, they achieved results between 76-80%, and on the MSRC-12b, they achieved lower results of 56-58.5% accuracy.

As their results on the MSRC-12b were lower than the results of other studies, which achieved higher accuracy results using algorithms such as SVM and RF, further testing on the MSRC-12 using HCRF was deemed outside the scope of this study.

2.5.5 Naïve Bayes and Hidden Markov Model

Another study, Gomes et al. [72], used Naïve Bayes and a Hidden Markov Model (HMM) as a machine learning algorithm to test on the MSRC-12 dataset for gesture recognition, selecting this dataset because it contained a wider range of different people and different gestures when compared to other existing datasets. 60% of the data was used for training, 20% for validation of training data, and the remaining 20% for testing. Gomes et al. used motion feature vector data to train their HMM, assigning one HMM per action class (i.e., actions such as 'kick',

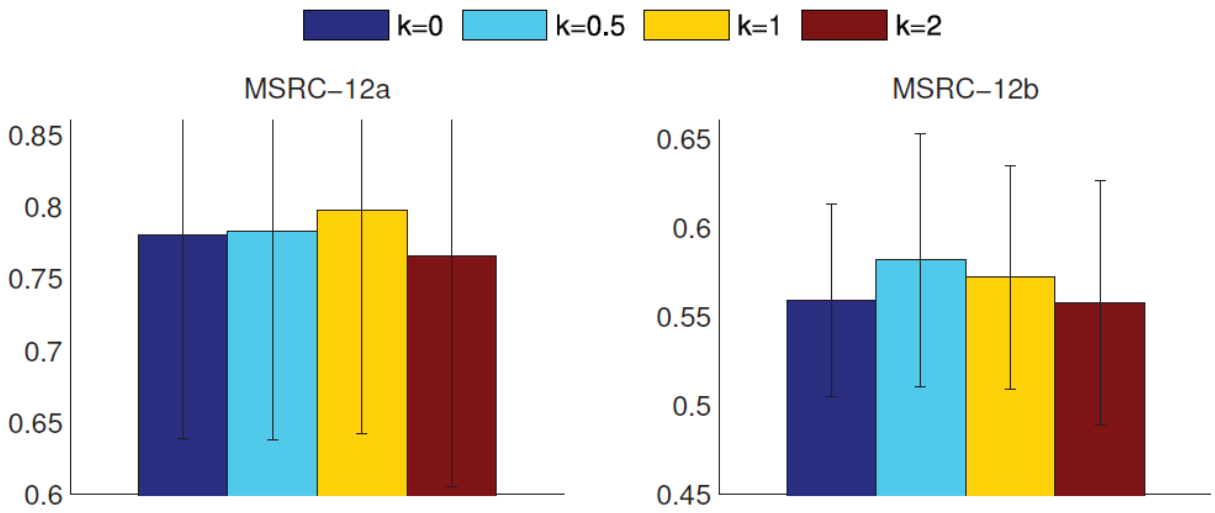


FIGURE 2.9: Results from Song 6-Class MSRC-12 Experiments where the mean F1 scores as a function of k was obtained

'bow', etc.). With this approach, they have configured the classification system so that "each state of the HMMs corresponds to different keyframes of the gesture (e.g. knee bending) plus an idle state. An observation will be the vector of real valued features that [they] extracted, and an observation sequence will be a gesture performance" [72]. After training an HMM for each class, Gomes et al. were able to create a classifier with all 12 HMMs (corresponding to the 12 gesture classes of the MSRC-12 dataset, 6 iconic and 6 metaphoric).

The data was segmented by separate gestures or poses. The segmented data was then converted to a csv file with angular measurements and velocities taken into account. These csv files were then imported into WEKA for classification and training. Using a Naïve Bayes classifier, Gomes could achieve an accuracy of 63.8% using Absolute Position; however, the results confirmed that there were some samples that were mislabelled or did not have a label, which skewed the results. Such results indicate that the the MSRC-12 dataset could be improved in terms of capturing data from research participants.

After "straw man" processing was used to normalize the feature vectors, Gomes et al. conducted computer training using HMM. Each class or gesture was given one HMM based on all of the gestures in class. In order to learn the unknown HMM parameters, the Baum-Welch algorithm was implemented. After training the dataset to a classifier, the researcher was able to achieve an average recognition rate of 65% —— slightly higher than before. Table 2.8 shows

Gomes et al.'s WEKA results, comparing how angle and velocity information affect the accuracy of the WEKA Data Mining Library.

Selected Features	Lift Both Arms	Crouch	Move Hand	Goggles	Circular Motions	Shoot	Take a Bow	Throw Grenade	Hands on head	Change Weapon	Beat Air with Hands	Kick	Total
Absolute positions	71.70%	72.10%	63.80%	78.70%	50.60%	86.70%	82.00%	84.50%	68.90%	83.60%	38.30%	90.30%	71.69%
Absolute Pos Velocity	48.30%	95.10%	82.60%	88.50%	47.20%	83.30%	90.20%	58.60%	65.60%	82.00%	51.70%	83.90%	72.35%
Angles	23.30%	96.70%	95.70%	75.40%	57.30%	81.70%	90.20%	60.30%	29.50%	73.80%	41.70%	93.50%	68.28%
Angular Velocity	46.70%	96.70%	78.30%	88.50%	51.70%	85.00%	88.50%	58.60%	57.40%	72.10%	70.00%	88.70%	72.87%
Abs Position Velocity and Single Angle	53.30%	95.10%	88.40%	91.80%	56.20%	86.70%	91.80%	69.00%	67.20%	86.90%	58.30%	95.50%	76.93%
Single Angle	45.00%	98.40%	46.40%	82.00%	36.20%	86.70%	80.30%	70.70%	29.50%	50.80%	20.00%	90.30%	59.90%

TABLE 2.6: Gomes' WEKA classification experiments on MSRC-12 features. This is similar to Zhang's comparison, however, Gomes' results are highest with a feature made of absolution position velocity combined with single angle value at 76.93%.

A	B	C	D	E	F	G	H	I	J	K	L	Total
76%	75%	52%	65%	47%	62%	89%	75%	64%	63%	27%	100%	65%

TABLE 2.7: Gomes' 12-Class MSRC-12 HMM results

Accuracies													
Data Type	Gesture 1	Gesture 2	Gesture 3	Gesture 4	Gesture 5	Gesture 6	Gesture 7	Gesture 8	Gesture 9	Gesture 10	Gesture 11	Gesture 12	Total
AbsolutePositions	71.70%	72.10%	63.80%	78.70%	50.60%	86.70%	82.00%	84.50%	68.90%	83.60%	38.30%	90.30%	71.69%
AbsolutePositionVelocity	60.00%	41.00%	4.30%	44.30%	56.20%	3.30%	65.60%	15.50%	8.20%	23.00%	28.30%	46.80%	33.68%
Angles	23.30%	96.70%	95.70%	75.40%	57.30%	81.70%	90.20%	60.30%	29.50%	73.80%	41.70%	93.50%	68.28%
AngularVelocity	31.70%	24.60%	8.70%	50.80%	41.60%	1.70%	80.30%	12.10%	39.30%	24.60%	51.70%	58.10%	35.50%
Both	65.00%	95.10%	30.40%	77.00%	59.60%	41.70%	80.30%	46.60%	52.50%	31.10%	30.00%	88.70%	58.06%
SingleAngle	45.00%	98.40%	46.40%	82.00%	32.60%	86.70%	80.30%	70.70%	29.50%	50.80%	20.00%	90.30%	59.90%

TABLE 2.8: Gomes Table 2.9 MSRC-12 WEKA results

Accuracies													
Selected Features	Lift Both Arms	Crouch	Move Hand	Goggles	Circular Motions	Shoot	Take A Bow	Throw Grenade	Hands on Head	Change Weapon	Beat Air with Hands	Kick	Total
Absolute Positions	71.70%	72.10%	63.80%	78.70%	50.60%	86.70%	82.00%	84.50%	68.90%	83.60%	38.30%	90.30%	71.69%
Absolute Pos Velocity	48.30%	95.10%	82.60%	88.50%	47.20%	83.30%	90.20%	58.60%	65.60%	82.00%	51.70%	83.90%	72.35%
Angles	23.30%	96.70%	95.70%	75.40%	57.30%	81.70%	90.20%	60.30%	29.50%	73.80%	41.70%	93.50%	68.28%
Angular Velocity	46.70%	96.70%	78.30%	88.50%	51.70%	85.00%	88.50%	58.60%	57.40%	72.10%	70.00%	88.70%	72.87%
Abs Position Velocity and Single Angle	53.30%	95.10%	88.40%	91.80%	56.20%	86.70%	91.80%	69.00%	67.20%	86.90%	58.30%	95.50%	76.93%
SingleAngle	45.00%	98.40%	46.40%	82.00%	32.60%	86.70%	80.30%	70.70%	29.50%	50.80%	20.00%	90.30%	59.90%

TABLE 2.9: Gomes Table 2.9 MSRC-12 WEKA results with averaging of frames before the hand labeled frame and after hand labeled

In Table 2.9, Gomes et al. made an alteration to their approach. They captured two averages, the average of motion feature vectors before the hand-labeled frame and the average of motion feature vectors after the hand-labeled frame, to make a $1x2M$ feature matrix for each feature set. Gomes et al. also attempted recognition through HMM, achieving a peak accuracy of 65%.

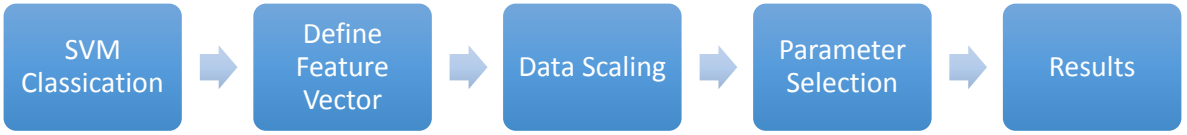


FIGURE 2.10: Zhang et al. SVM Learning approach

However, as both their Naïve Bayes and HMM approach was produced lower accuracy results than Zhang et al.'s [2] tests using SVM and RF [2], further testing on the MSRC-12 dataset using HMM was deemed outside the scope of the present study.

2.5.6 Support Vector Machines

Support Vector Machines (SVMs) are a nonlinear learning model often used for classification and regression analysis. In machine learning, SVMs are very effective solutions for teaching computers to analyze data and recognize patterns. For gesture recognition, SVM can be used to label the gesture classes and associate them with feature vector data in order to train gesture recognition applications. This machine learning process involves defining the motion feature vector, scaling the data, selecting the appropriate kernel, and finally selecting the appropriate parameters.

Zhang et al. [2] applied SVM to the MSRC-12 dataset to test recognition accuracy on gesture samples. Zhang et al. first used xyz-velocity, joint angle, and joint angle velocity to define the motion feature vector for each frame, with the dimensions 60 + 35 + 35 = 130. As 35 frames are included in each training sample, this creates a dimension of a motion feature vector for training samples that is 4420. After scaling the data, Zhang et al. tried three kernels before finalizing their selection: linear kernel, polynomial kernel, and radial basis function (RBF) kernel. They achieved the highest results with the RBF kernel, eventually selecting two parameters that enable it to accurately classify unknown data.

Zhang et al. achieved an accuracy of prediction of 67.33% after applying SVM. Although this is significantly lower than their experiments using RF, SVM has shown potential as a machine learning algorithm for classifying gesture motion features, and will be further applied to the MSRC-12 in Section 5.3 on page 81.

2.5.7 Random Forest

In these tree structures, leaves represent gesture classes and their labels, and branches represent conjunctions of motion features that lead to those gesture classes. For each tree, a decision function splits the training data that reach a node at a given level in the tree. Then each tree provides a classification; Zhang et al. construe this as saying "the tree 'votes' for that class" [2]. The forest then chooses the classification having the most votes. The resulting forest classifier is used to classify a given motion feature vector by taking the mode of all the classifications made by the tree classification for all the forest.

For the Random Forest decision tree method, each tree is grown as follows:

- If the number of classes in the training set is N, sample N classes at random but with replacement from the original data. This sample will be the training set for growing the tree.
- If there are M input variables, a number m«M is specified such that at each node, m variables are selected at random out of the M, and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
- Each tree is grown to the largest extent possible. There is no pruning.

The RF algorithm (for both classification and regression) is as follows:

1. Draw n tree bootstrap samples from the original data.
2. For each of the bootstrap samples, grow an unpruned classification or regression tree with the following modification: at each node, rather than choosing the best split among all predictors, randomly sample the m-tree of the predictors and choose the best split from among those variables. (Bagging can be thought of as the special case of RFs obtained when m-tree = p, the number of predictors.)
3. Predict new data by aggregating the predictions of the n-trees (i.e., majority votes for classification, average for regression).

An estimate of the error rate can be obtained, based on the training data, by the following:

1. At each bootstrap iteration, predict the data not in the bootstrap sample (what Breiman [88] calls "out-of-bag", or OOB, data) using the tree grown with the bootstrap sample.
2. Aggregate the OOB predictions (on the average, each data point would be out-of-bag around 36% of the times, so these predictions must be aggregated).
3. Calculate the error rate, and call it the OOB estimate of error rate.

In the original paper on RF [89], it was shown that the forest error rate depends on two things:

- The correlation between any two trees in the forest. The higher the correlation, the greater the forest error rate would be.
- The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

There are many benefits to using RF as proposed in this thesis:

- It can handle thousands of input variables without variable deletion.
- It provides an estimate of which variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data is missing.
- It has the ability to grow trees of different lengths.
- It has methods for balancing error in class population unbalanced datasets.
- Generated forests can be saved for future use on other data.
- Prototypes are computed that give information about the relation between the variables and the classification.

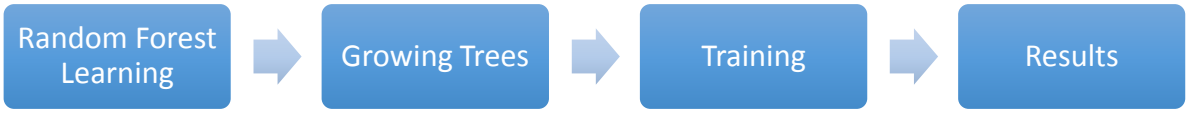


FIGURE 2.11: Zhang et al. Random Forest Learning approach

- It computes proximities between pairs of cases that can be used in clustering and locating outliers.
- The above capabilities can be extended to unlabeled data, leading to unsupervised clustering, data views, and outlier detection.
- It offers an experimental method for detecting variable interactions.

After growing a decision forest of 300 decision trees, Zhang et al. were able to achieve a recognition accuracy of 80.69%.

The Zhang et al. paper achieved high results, but the fact that not one researcher achieved above an 80.69% alerted me that there may be some fundamental errors in the MSRC-12 dataset in terms of sample quality. To test this, I created a new dataset, PJVA-20, to see whether the modality of instruction, data sampling, gesture recognition, and machine learning techniques demonstrated in the literature would achieve higher accuracy on another dataset.

2.6 Charades as a Modality of Instruction to Collect Gesture Samples

As explained by Fothergill et al., the primary meaning of instructions are "to convey the kinematics (the features of motion of the body)" [5]. After a scan of the literature —— in several academic fields such Machine Learning, HCI, & Computer Vision, as well as a general search of Ryerson University's library database using the keywords 'Charades', 'Data', and 'Gesture' —— I found several studies that attempted to use charades as a modality of instruction for capturing gesture samples.

One recent study [90] used charades as a Game with a Purpose (GWAP) to crowdsource gesture data for analysis. In this study, Spiro discussed the need for "sophisticated [machine learning]

algorithms to push the accuracy of automated visual tasks... [using charades] to address the need for large datasets to drive machine learning solutions" [90]. Unfortunately, though this methodology directly aligns with the goals of the present study, he did not analyze the resulting data using gesture recognition or machine learning techniques. Therefore, there is little basis for comparing my approach with Spiro's crowdsourcing approach.

Another study, by Roemmele et al. [91] in the field of action recognition, attempted to use a game of charades as a data capturing technique that "abstracts away from the full complexity of human motion, reducing it to a simple trajectory displayed by a 2-dimensional shape". In this study, research participants created motion trajectories for "Actions that are highly recognizable and highly distinguishable" (including quiver, wiggle, sneeze, bounce, etc.) by animating a triangle to depict those actions. Although the format of charades worked well for their study to gather research data, the use of animated triangles instead of human kinematics is too abstract for comparison with the present study.

Finally, I determined that a study by Hsieh et al. [6] is the most relevant to the present study, as they construed charades similarly as a natural form of data collection, "an acting game with the idea for one player to use physical (e.g. gesture or action) rather than verbal language to act out a semantic concept (i.e. a word or a phrase as the charades topic) for the other players to guess." Also, their purpose is identical to my own: to collect kinematic data in the interest of creating a machine learning system capable of recognizing gestures, i.e., automatically identifying the patterns between semantic concepts and human actions.

For their experiments, Hsieh et al. also built a video dataset. Listed in Table 2.10, eight research participants acted out 32 gesture classes, including the actions of imitating a celebrity's signature moves, playing a sport, and interacting with a common physical object —— all iconic actions with potential for varying styles of performance. Each video sample was rated by 10 other participants based on their subjective judgement, maintaining the methodology of the traditional game of charades.

First of all, Hsieh et al. construe an action as a sequence of poses performed by a research participant. They then extract a "silhouette" of the pose, which is an outline of their body rather than a 3D skeleton comprised of GDPs (see 2.12). The dimensions of this silhouette is then used to determine its viability in gesture recognition applications. Consequently, each pose of

Celebrity/Famous Character	Sport/Action Event	Daily Object
Bruce Lee	Baseball	Bicycle
David Beckham	Basketball	Toothbrush
Michael Jackson	Boxing	Glasses
Michael Jordan	Taekwondo	Electric Drill
Super Mario	Karate	Watch
E.T.	Soccer	Chair
Zorro	Golf	Mug
Superman	Swimming	Machine Gun
Spiderman	Shooting	Keyboard
	Archery	Cap
	Morra	Hairbrush
		Mirror

TABLE 2.10: Hsieh et al.'s list of semantic concepts to test, divided into three subsections.

a gesture is represented by these features. To label and categorize data, Hsieh et al. cluster similar poses from each semantic concept into the same group via affinity propagation.

It is important to note that this study did not intend to collect gesture samples to generate a 3D skeleton capable of providing specific GDP values: In contrast to human action recognition, the goal of our study is to evaluate the iconic level of a performing action video, i.e. whether it is vivid and expressive to describe the corresponding concept" [6]. In other words, Hsieh et al. are specifically testing the viability of charades as a modality of instruction for gesture recognition research, and they have determined that it is suitable for conveying the patterns between semantic concepts and human actions.

To validate their model, they used Kendall's tau coefficients to measure the similarity between the ranking results and the ground truth data. These results imply that around 60-70% of their rank order lists correctly match the human rank made during charades. Although they admit their results could be higher, Hsieh et al. have achieved their research goal and successfully demonstrated that charades has potential as an instruction modality.

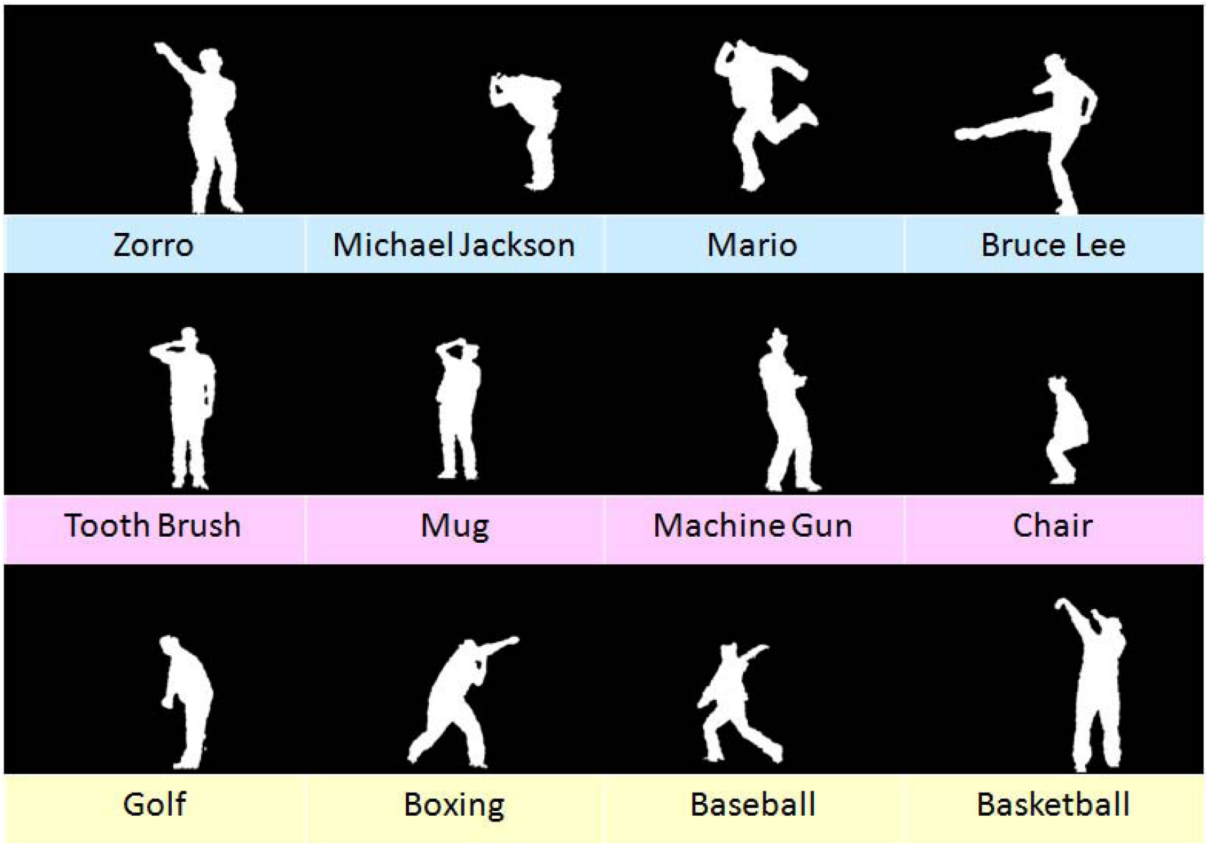


FIGURE 2.12: The "silhouettes" generated by Hsieh et al. These do not track the 20 GDPs of the research participant. This data cannot generate a 3D skeleton such as those from MSRC-12 and PJVA-20.

2.7 Summary

As gestures tend to vary from person to person, a database must contain invariant data to widen the scope of the system; it is important to capture multiple performances for each gestures to stop the system from becoming too gesturer-specific. Overall, Fothergill et al. provided instructions to research participants that were generally suitable for creating a gesture dataset, however, there are a few instances that I have identified either bias or errors in the methodology. As a result, I view Fothergill et al.'s instructions to create a 'supervised learning' approach, which limits the freedom of research participants and makes it more difficult to acquire samples with the appropriate coverage for gesture recognition applications.

With this is mind, I would like to offer a gentle criticism of the MSRC-12 methodology, which I think creates bias in their data gathering process. When finalizing which static images would be presented to research participants in the instructions, Fothergill et al. allowed the software

designer to extract individual video frames at their own discretion, at moments which "the designer considered necessary to fully define the gesture" [5]. As such, although the experimenter told the participant to 'do what they wanted', this statement seems to conflict with the use of static images that stipulated very specific poses for the research participants to emulate. With charades as the foundation of the present study, the use of static images as instructions is eliminated to enable fuller freedom of motion for the research participants.

With reference to the gesture recognition literature, I have demonstrated the prospective of using the game of charades to collect robust data from research participants. This approach was used to collect the PJVA-20, which is a very different methodology from the MSRC-12 approach. By collecting several different iterations of a gesture from different users, numerous variations of how to represent the same semantic concept was collected in the PJVA-20 dataset, increasing the coverage of the resulting gesture samples.

In Chapter 3, I will outline the approach taken for sampling gestures and conduct a brief statistical comparison of the resulting gesture data against that of the MSRC-12, to discuss whether using charades as a modality has created suitable gesture samples for gesture recognition applications.

Chapter 3

Theory: Experimental Design for Capturing Gesture Samples

"The statue is there for nothing but the sum of all it has acquired.
May not this be the same with Man?"

- Étienne Bonnot de Condillac

3.1 Introduction

To solve gesture recognition problems, researchers must collect data that contains examples of movements and their associated gesture label. Gesture recognition researchers and developers typically use human subjects as research participants to generate datasets used to train and test machine learning systems.

Though tracker-based data capturing systems can render the 3D skeleton of a research participant very accurately, they are not realistic for general users, who often are in positions in which wearing trackers is impractical. One of the most important hurdles in solving the gesture recognition problem is how to capture tracker-quality data of a gesture through vision-based approaches.

After collecting the GDP values to generate static feature data from a single frame of a gesture sample, researchers can use gesture recognition techniques to extract motion feature data from several frames. Therefore, the role of gesture extraction methods and machine learning approaches — as the means to capture the motion feature data from gesture samples, and then appropriately categorize and label the resulting data for gesture recognition — is becoming more important.

This chapter is organized as follows. Section 3.2 will outline the initial considerations for capturing gesture data from research participants using a depth cameras and no trackers, touching upon which type of camera and distance measurement techniques were used. Section 3.3 will discuss the approaches used for capturing gesture samples for the PJVA-20 dataset, focusing on the instructional modality of charades to compare this approach that of the MSRC-12. Section 3.4 conducts a brief statistical comparison of the resulting gesture data to consider whether using charades as a modality has created suitable gesture samples for gesture recognition applications.

3.2 Considerations for PJVA-20 Sampling Approach

In order to develop a system capable of gesture recognition, the most important problem is how to categorize these samples in a way that highlights the most important features for computer training. A single set of GDPs represents a single frame of a gesture sample, which is a static pose. However, similar poses can exist in multiple gestures and can thus confuse classification accuracy between different gesture classes. For example, with static image data alone, a gesture recognition system would be unable to distinguish between a rotating limb moving clockwise and counterclockwise, because both gestures share the identical pose while being distinct gesture classes. As such, when creating a gesture recognition system, extracting motion features to account for several frames of a gesture sample can increase accuracy by tracking subtle features that differentiate between similar gesture classes.

When sampling gestures, I classified the data into two different types of feature that complement each other for better recognition accuracy: 1) “static” features, which is a set of GDPs

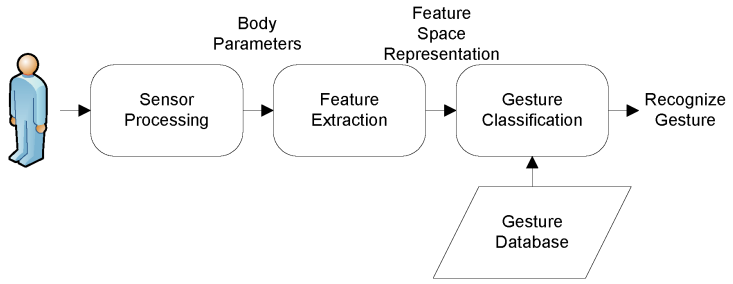


FIGURE 3.1: Pattern Recognition System (ideal for static gesture recognition)

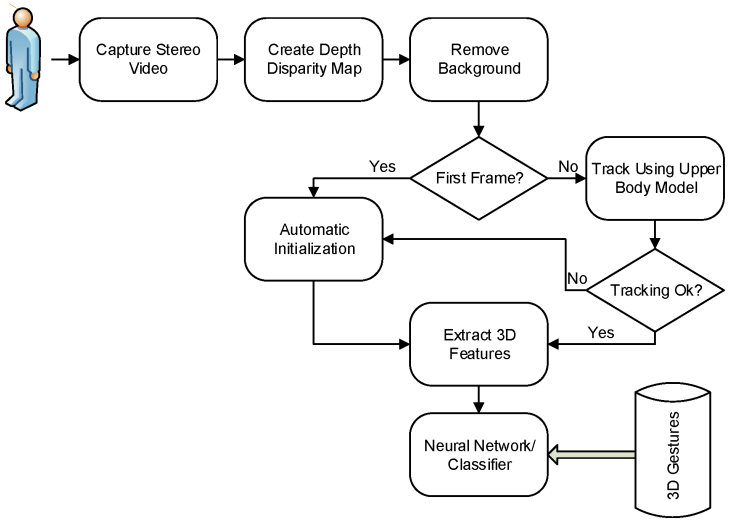


FIGURE 3.2: Dynamic Gesture Recognition Process

that represent a participant's pose extracted from a single video frame, and 2) "motion" features, which are sequential frames extracted from a gesture sample, to track the GDPs of a research participant as they perform a gesture. A gesture sample can thus be either be captured by a Kinect sensor as dynamic data (i.e., research participant is carrying out a movement) or as static data (i.e., participant is frozen in a special pose). Dynamic data are intended to capture as much positional data on the gesture as possible by examining a moving gesture represented by a sequence of images. Figure 3.2 shows a schematic of a dynamic gesture data being processed by gesture recognition algorithms.

In dynamic systems, background data and image effects are applied to the gesture to help separate only the necessary features from their background data. Thus, dynamic data requires the use of sophisticated algorithms to accurately track the features and movement of the body throughout a gesture sample.

When choosing effective classifiers for the data, I determined that the easiest approach is to

construct a 2D feature matrix with a constant feature length. The complexity of the problem is illustrated in 3.10.

In Chapter 5, I will test the static features of samples from the PJVA-20 dataset and motion features of samples from the MSRC-12 dataset. For PJVA-20, only static features were required to achieve high accuracy, demonstrating the efficacy of using charades to acquire high-quality gesture samples. For the MSRC-12, the results after testing on static features was fairly low, which gave researchers the opportunity to extract motion features from the samples in order to increase accuracy. In Chapter 4, I will compare the PJVA approach for extracting motion features to that of other researchers in the literature.

3.2.1 Type of Camera

In early experiments, I used a custom built IR camera, however, this approach limited the ability to compare my results with those in the literature because of the novelty and uniqueness of this camera. Things are very different today with the lowered prices for this camera technology —– a popular commercial camera system, Microsoft Kinect, is now widely available. This means that all researchers are using these "depth cameras," ensuring consistency of data input for comparative research purposes.

The Kinect's depth comparison features are inspired by Lepetit's approach in [92] and are represented in 3.1, where the features are calculated for a given pixel x where $d_I(x)$ is the depth of a pixel x in image I, and $\Theta = (u, v)$ describes the offset of u and v. The offset is normalized by $\frac{1}{d_I(x)}$ which ensures the features are depth invariant. 3.5 shows the skeletonization compared to ground truth. OpenNI is PrimeSense's opensource driver that is paired with the NiTE2 skeletonization — its full body tracking is slightly more accurate than the Kinect approach, as it has been optimized to work better with side views.

$$f_\Theta(I, x) = d_I(x + \frac{u}{d_I(x)}) - d_I(v + \frac{u}{d_I(x)}) \tag{3.1}$$

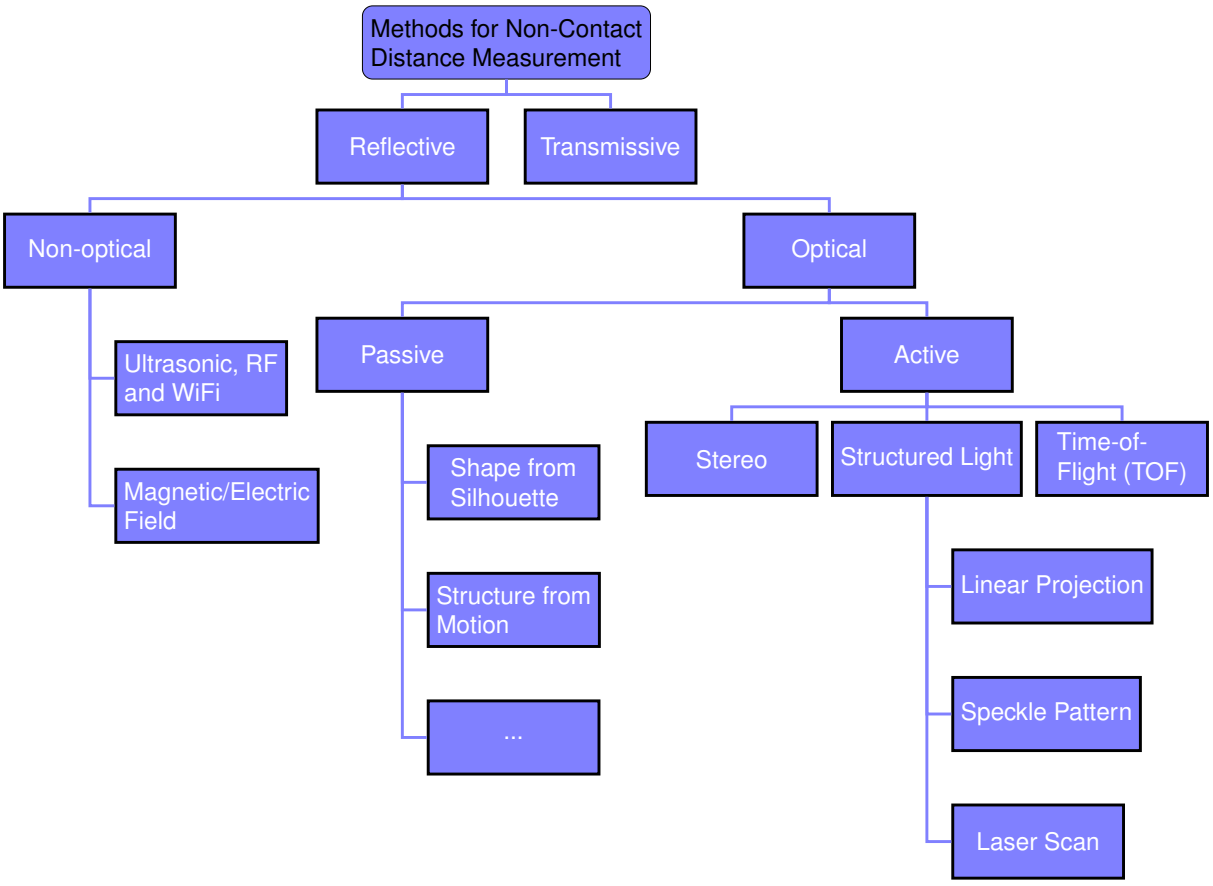


FIGURE 3.3: Taxonomy of distance measurement devices.

3.2.2 Distance Measurement Techniques

The ability to calculate accurate distance measurements without sensors or trackers was one of the important requirements of designing the PJVA-20 dataset. To calculate distance measurements, the distance measurement approaches are classified into different types of "passive" systems (see Figure 3.3, which shows the taxonomy of distance measurement devices used in this research).

For instance, the "stereo vision" device is an example of a passive approach, using at least two cameras at different angles to capture data that enables PJVA to refer to distance as a 3D positional measurements. This allows PJVA to calculate the GDP values of a research participant as they are scanned in real time by the passive sensing devices.

Combined, these passive sensing solutions enable PJVA to convert measurements of depth/distance into depth maps, enabling the system to recognize a gesture based on several vision-based systems including structured light and Time of Flight (ToF).

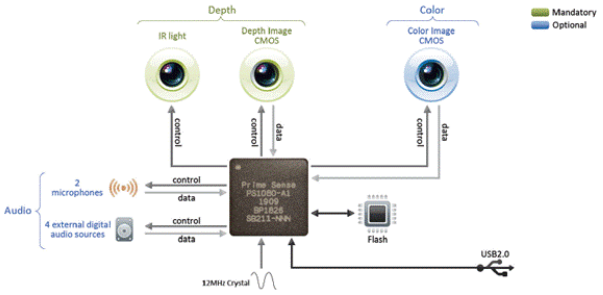


FIGURE 3.4: PrimeSense IC Layout

3.2.2.1 Structured Light

Structured light is an approach that involves projecting a known pattern of pixels (e.g., grids, horizontal or vertical bars, or speckles) onto an environment, e.g., a barcode reader used in a retail store. Based on the deformations of these projections when striking a surface, computers use a vision system to calculate depth and surface information from object(s) in the environment.

One structured light system that must be mentioned is Microsoft's Project Natal (hardware from PrimeSense and software from Microsoft), because this system was a game changer and is considered the first commercially available depth camera. The hardware involves a depth and RGB sensor combined with a speckled structured light laser emitter. The original PrimeSense patent [93] outlines how the hardware operates: the speckle pattern of the camera can be seen if viewed in the dark by a camera with an optical bandpass filter capable of viewing a center wavelength in the range of 830nm (specifically 833nm) in the IR spectrum.

To get the most accurate sample, the data capture is taken with two Kinect cameras: one for projecting the speckle, and the second one configured to only focus on the first camera's speckle pattern when in IR viewing mode Figure 3.6. As can be seen in Figure 3.4, the system is comprised of a single chip that controls two cameras, one IR and the other RGB. Most of the processing is done on camera through PrimeSense's PS 1080-A1 chip, which conducts depth calculation and simple image processing based on the two input cameras.

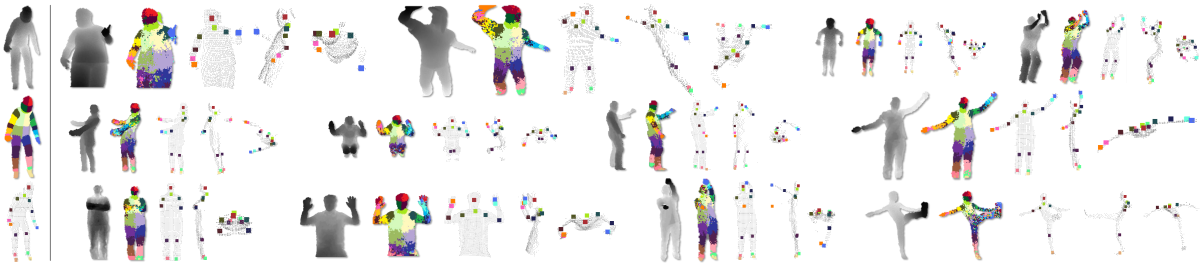


FIGURE 3.5: Kinect skeletonization approach. Top row represents synthetic, middle row is real, and the bottom shows failure modes. The left column is the ground truth for a neutral pose. For each example, we see the depth image, the inferred most likely body part labels, and the joint proposals shown as front, right, and top views.

FIGURE 3.6: PrimeSense/Microsoft Kinect speckle pattern projecting on a research participant

3.2.2.2 Time of Flight (ToF)

Although speckle patterning is an effective way for cameras to determine the depth of an object in its environment, ToF is even more effective, using lasers instead of structured light to improve a camera's calculation of depth. This technique is now more widely used than structured light: for instance, the new Kinect cameras use ToF, whereas older Kinect models used speckled patterns.

Below is a typical measurement setup for a scene point using ToF. The sensor estimates the radial distance by ToF or RADAR principal. The distanceρ, is calculated at time τ with electromagnetic radiation at light speed c, which have a relationship of $\rho = c\tau$. The transmitter emits radiation that travels towards the scene, which is then reflected back by the surface to the sensor receiver. The distance covered is now 2ρ at time T. The relationship can be written as:

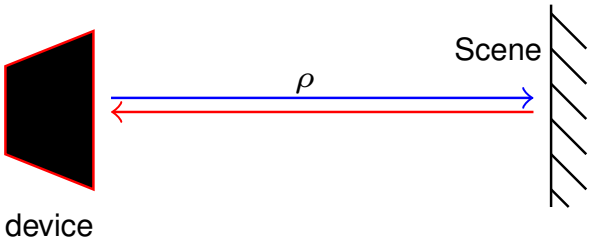


FIGURE 3.7: ToF Basic prinple

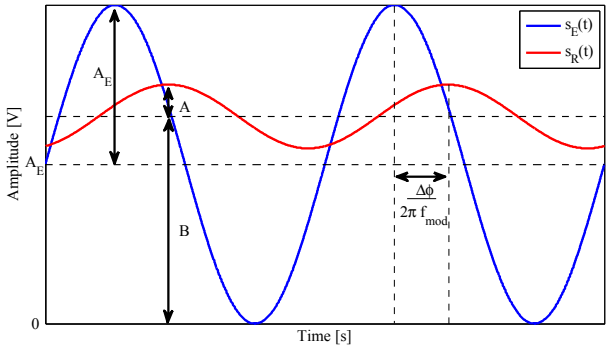


FIGURE 3.8: Example of ToF emitted signal in blue and received signal in red.

$$\rho = \frac{c\tau}{2} \tag{3.2}$$

$$S_E(t) = A_E[2\pi f_{\text{mod}}\, t] \tag{3.3}$$

Using ToF, I was able to capture data and deliver depth maps to the computer as video rates or positional measurement matrices, with entries giving the distance between the matrix pixel and the corresponding scene point.

3.3 PJVA-20 Dataset

The PJVA-20 is a dataset comprised of 20 gesture classes, comprised of 50 samples of each class sampled from 6 participants. Participants included five men and one woman between the ages of 20 to 30 years old with various body build types.

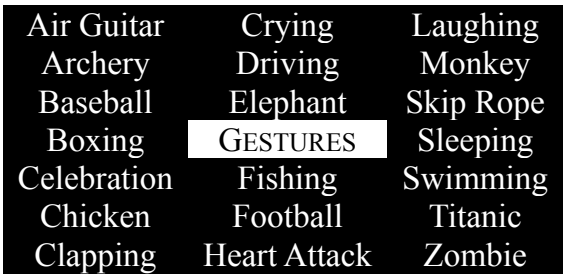


FIGURE 3.9: Semantic concepts of PJVA-20 dataset. These gesture names were taken from a charades game and provided to a sample group of 5 male individuals and 1 female individual.

3.3.1 Modality of Instruction

The gestures of both the MSRC-12 and PJVA-20 were collected in a very similar manner —— sampled using the Kinect camera —— but the modality of instructions for the PJVA-20 dataset were transmitted in a different format following a different methodology. Participants were not instructed on how to perform the gestures, and stood at different locations relative to the camera.

First, the author selected 20 charades cards after a scan of several publicly available editions of the game —— primarily iconic gestures (see Figure 3.9).

Unlike the MSRC-12, which provided their research participants with both a 'gesture outcome' and 'descriptive text' (see Table 2.3 for MSRC-12a and Table 2.4 for MSRC-12b), the PJVA-20 only provided these 1-2 word descriptions to its research participants. This was done in the interest of improving the coverage of the resulting data, and also subsequently made it easier to label the gesture classes for computing learning applications.

Instead of supervising the research participants, they were asked to conduct the data-capturing experiment on their own outside of a laboratory setting. This involved taking a Kinect sensor to a location of their choosing —— any location at which they had enough space to set up the sensor, e.g., office, home living room, bedroom, library —— to record gestures onto a computer loaded with basic video editing software (all of the controls required for recording the gesture footage were made familiar to participants in advance of these experiments). Each participant was asked to record 1 of the 20 gestures, labeling the gesture by manually typing in the name of the gesture class as it appears on the charades card. Participants would then press

the ‘Start/Record’ button on the tool, walk into scene, and physically perform the gesture in whichever way they felt natural after reading the description on the card. Participants could repeat the gesture as many times as they wanted within the same sample (e.g., if a participant was asked to swing a bat, they could keep swinging until they thought the motion was effectively conveyed). When they felt satisfied with their actions, they would walk out of scene and press ‘Stop’ on the computer terminal running the tool.

After completing this recording, the participants were then asked to review the resulting 3D skeleton, instead of the video footage of the gesture, to determine whether it accurately depicted the gesture they had just performed (this methodology hearkens back to the research from Johansson, Section 2.1, for which research participants were able to recognize gestures from just moving dots against a black background). If they could not recognize their own 3D skeleton as representing the gesture, they were allowed to erase their sample and start over. If the skeleton looked appropriate, they were required to trim the beginning and end of the sample to remove the extraneous data that was captured as they walked in and out of the setting, i.e., before and after they performed the gesture.

The results of testing on the resulting gesture samples of PJVA-20 for recognition accuracy is 98.5% (see 5.2). This demonstrates that charades is an appropriate modality of instruction for capturing high-quality gesture samples.

3.4 Statistical Comparison of MSRC-12 and PJVA-20

By comparing the characteristics of the MSRC-12 to the PJVA-20, it will be possible to evaluate the potential for charades to be used as a data capturing modality. The resulting data can help determine which method should be used to create a dataset more suitable for machine learning applications in gesture recognition.

Table 3.1 provides the resulting data for both the MSRC-12 and the PJVA-20 datasets for creating gesture samples for computer training, comparing the datasets over several categories:

While the majority of these statistics do not have an effect on the quality of gesture sample for machine learning results, three of these statistics stand out for comparison when generating

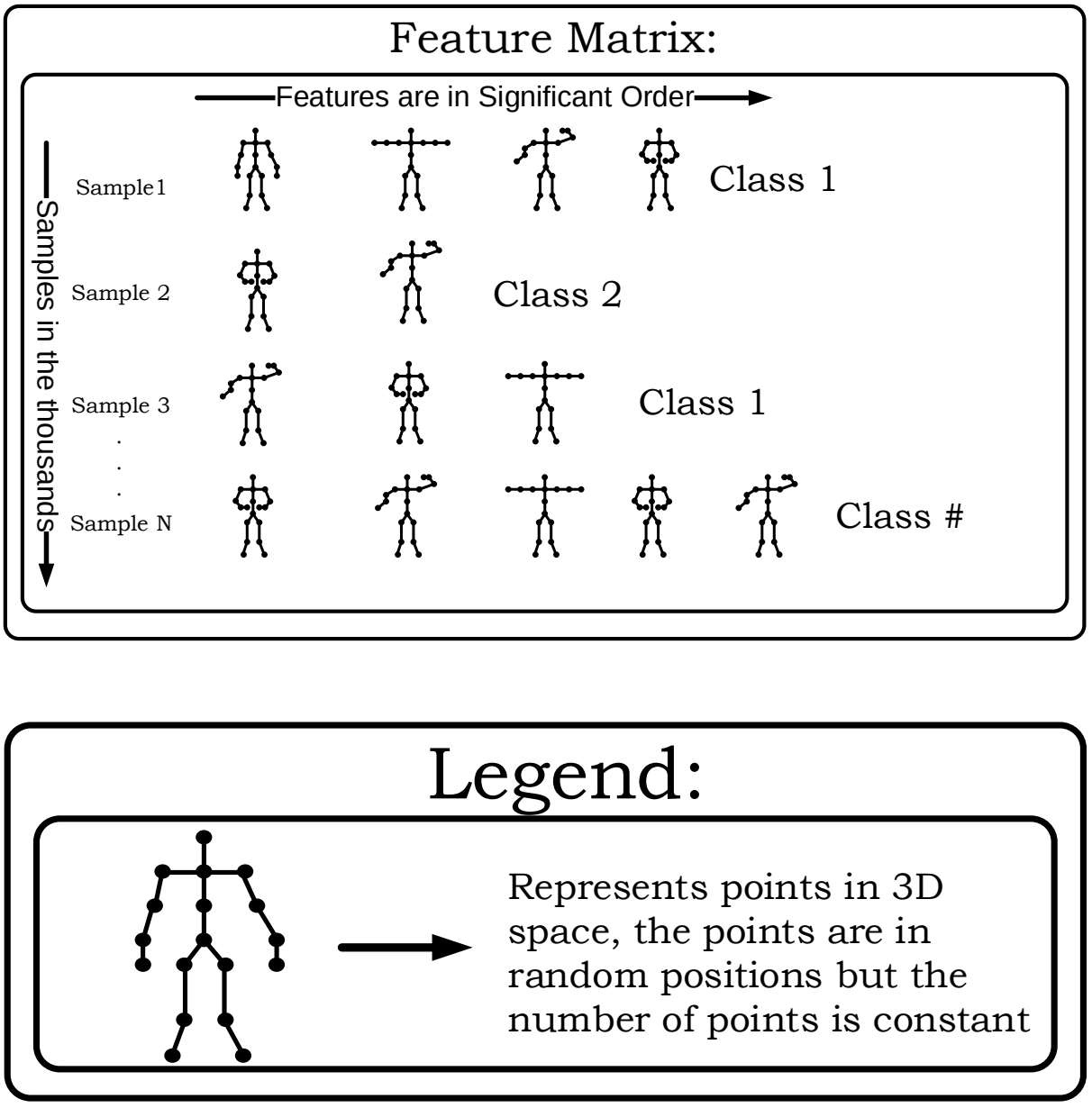


FIGURE 3.10: Technical Representation of Motion Feature Data of PJVA-20 Dataset

samples for training recognition systems: number of gestures instances per sample, number of frames per sample, and number of GDP tracked.

For the number of gestures instances per sample statistic, I did not set an initial number. By initially (and seemingly randomly) setting the number of gesture instances required per sample, I believe Fothergill et al. has demonstrated bias in determining the appropriate number of instances. The actual number of instances required may vary depending on the gesture; in other words, the gesture recognition system may actually require more or less instances to achieve reliable recognition results in terms of correctness. This bias can be eliminated using

	MSRC-12	**PJVA-20**
# of gesture classes	12	20
# of GDPs tracked	21	19
Avg. # of frames per sample	800	300-900
# of gesture instances per sample	10	N/A
# of research participants	30	6
# of frames per sample used as training data	35	15 & 45
Total # of frames in dataset	385,732	719,359
Total # of gesture instances in dataset	3054	6244

TABLE 3.1: Comparison of MSRC-12 and PJVA-20

the charades methodology, because the correct number of instances is only determined when the second participant is able to guess the correct gesture —– not through any intervention from the developer whatsoever. Similarly, for the number of frames per sample, I used multiple values (15 & 45 frames) to allow for some variance in the gesture sample to account for different gesture lengths. By pre-setting the number of frames at exactly 35, it reduces the coverage of the system by limiting the potential for longer gestures that may require extra time to be performed fully. This will be further explored in Section 4.4 on page 64.

For the number of GDPs tracked per sample, this denotes how many GDPs are accounted for when extracting motion features from a gesture sample. The difference between the datasets represents a major divergence from gesture recogition techniques in the literature. Whereas the CMFE approach in the literature creates an artificial joint to capture GDP values (20 GDPs + 1 artificial GDP = 21 GDPs), AMFE technique places the (0, 0, 0) joint at the waistline of the research participant. As this joint thus never generates a GDP value, the PJVA approach tracks 19 GDPs when generating samples for gesture recognition applications. This will be further explored in Section 4.2 on page 59.

Finally, although the number of research participants statistic demonstrates a fairly large difference between the two datasets, the results of the PJVA approach demonstrates that datasets do not require a large number of participants to capture robust data for gesture recognition applications (as long as there are several different body types among the participants).

3.5 Summary

This chapter discussed the considerations for capturing gesture data, including the difference between static and motion feature data, distance measurement calculation, and the modality of instruction for research participants.

After a gesture dataset has been collected, providing the GDP values for static gestures, gesture recognition algorithms can be applied to the resulting dataset in order to extract motion feature vectors. This is not necessary for all gesture recognition systems, but was conducted in this dissertation to test the novel gesture recognition approach proposed in this thesis —– the PJVA technique chiefly comprised of AMFE, Polynomial Motion Approximation, and Principal Component Analysis (PCA) —– on the MSRC-12 (see Chapter 4) in order to compare the results of the PJVA approach to those in the literature.

Chapter 4

The Proposed Technique: PJVA

"The human brain had a vast memory storage. It made us curious and very creative. Those were the characteristics that gave us an advantage - curiosity, creativity and memory. And that brain did something very special. It invented an idea called 'the future'."

- David Suzuki

4.1 Introduction

After a gesture dataset is collected, gesture recognition algorithms can be used to extract the required features from a gesture sample for gesture recognition applications. In this section, I will demonstrate the PJVA technique for applying gesture recognition techniques —– chiefly comprised of AMFE, Polynomial Motion Approximation, and Principal Component Analysis (PCA) —– on the MSRC-12.

In the original MSRC-12 dataset, each frame of gestures is recorded as the absolute position of 20 joints (GDPs) of the human body in xyz-coordinates —– 60 data total per frame. Motion features are extracted from these static gesture samples by taking polynomial approximation of the motion of each GDP along three axes: X, Y, & Z. To reduce the size of these motion feature vectors while maintaining the same quality of data, the polynomial values are transformed into eigenvectors, thereby compressing the number of values in the motion feature vectors. This is

demonstrated in Figure 4.11. In the PJVA approach, the resulting motion features are formatted in 15-frame and 45-frame lengths for training purposes, thereby representing fast and slow moving gestures comprised of different motion feature vectors in the same sample (see 4.9 and 4.10 in Section 4.4 on page 64).

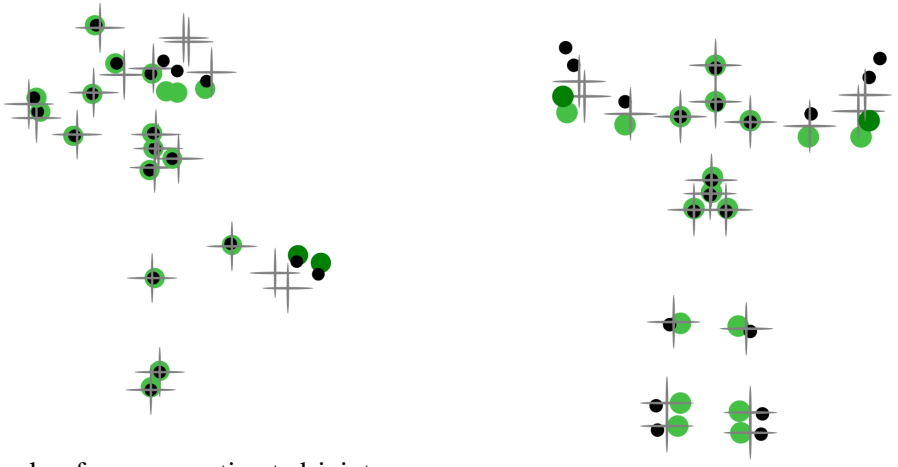
This chapter is organized as follows. Section 4.2 on page 59 will describe the novel gesture recognition technique, AMFE, which was created to reorient the motion feature extraction process to improve gesture recognition accuracy on the MSRC-12 dataset. Section 4.3 on page 62 will discuss the calculations used to scale the resulting GDP data. Finally, Section 4.4 on page 64 will outline the approach used to extract motion feature vectors from these scaled GDP values, which involves polynomial approximation and PCA.

4.2 Novel AFME Gesture Recognition Technique for Reorienting Feature Extraction

For extracting motion feature data from the MSRC-12 dataset, the AMFE technique places the (0, 0, 0) joint at the waistline of the research participant instead of creating an artificial joint to capture GDP values.

In order to demonstrate the efficacy of this technique, I had to first investigate which GDP had the least amount of noise generated when tracking a research participant. I developed an application that would monitor joint positioning over a period of time and average the xyz-positions. By running the PJVA recording software with a user standing still in front of the camera, I was able to discover that the centre of the waistline (the "Hip Centre" joint) generated the smallest amount of noise over time (see 4.3 and Table 4.1). The hands, head, and feet GDPs, for instance, deviated from their original xyz-coordinates more then 5cm over time. This test was also repeated with a user standing on one leg and the same results were found.

Thus, using AMFE, the positional data of GDPs are extracted as static poses representing specific gestures.



(A) Example of average estimated joint error of user standing still over time while right leg is extended

(B) Example of average estimated joint error of user standing still over time with arms raised

FIGURE 4.1: In these recordings of a still person, the green circles are the actual XYZ positions, the black circles represent the last frame position and the cross hairs represent the average position

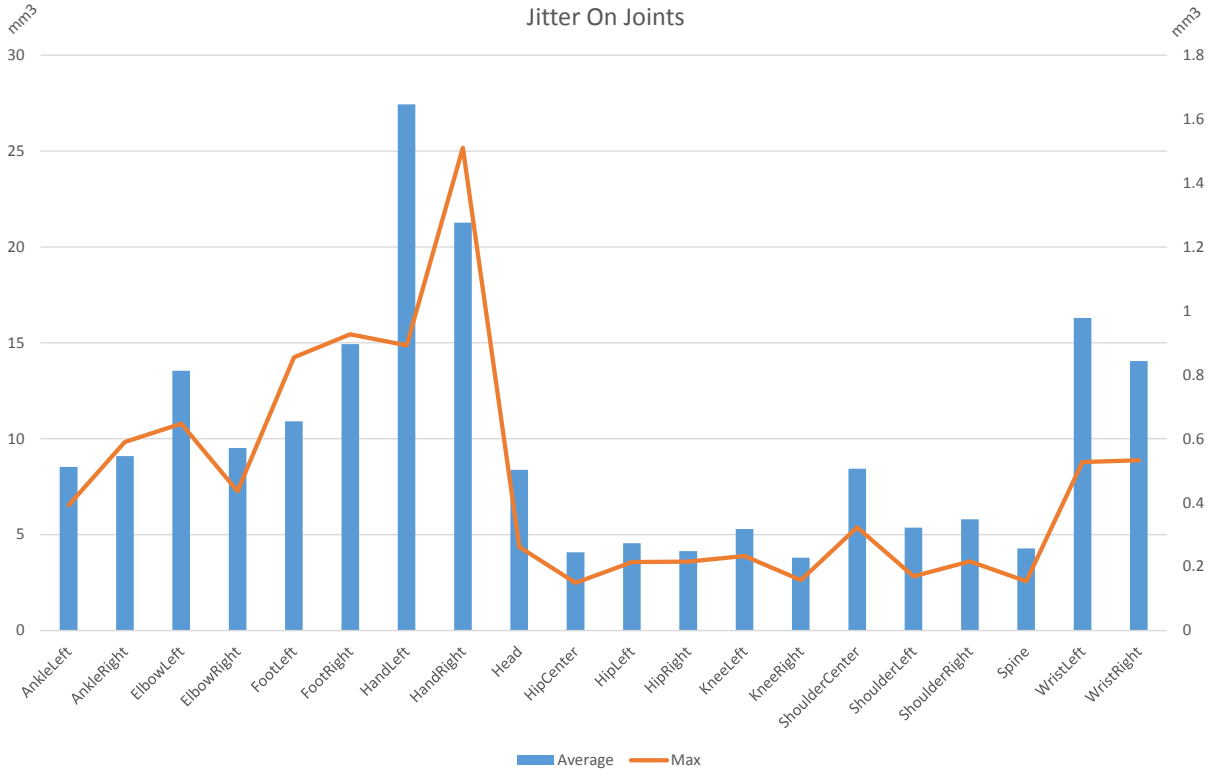


FIGURE 4.2: Plot of average jitter of joints on still user.

Joint	Average	Max	Sum
AnkleLeft	0.511914075	6.536789358	255.9570374
AnkleRight	0.545578392	9.830658081	273.3347746
ElbowLeft	0.812832309	10.785525	406.4161547
ElbowRight	0.571012953	7.267045283	285.5064767
FootLeft	0.654230841	14.23966792	327.7696512
FootRight	0.896174283	15.44382623	448.0871416
HandLeft	1.645970322	14.86438852	822.9851612
HandRight	1.275785543	25.16949099	637.8927715
Head	0.502328221	4.360165227	251.1641103
HipCenter	0.244428226	2.489307051	122.214113
HipLeft	0.27306771	3.570031104	136.5338551
HipRight	0.248500511	3.58938828	124.2502556
KneeLeft	0.317170662	3.87962978	158.5853308
KneeRight	0.228131084	2.626484906	114.0655419
ShoulderCenter	0.5057643	5.376494407	252.88215
ShoulderLeft	0.321710926	2.825308916	160.855463
ShoulderRight	0.347584683	3.600692871	173.7923413
Spine	0.25713056	2.569161855	128.5652799
WristLeft	0.977868484	8.778208411	488.9342419
WristRight	0.843096027	8.873226336	422.3911096
	Max Average	**Max Max**	**Max Sum**
	1.645970322	25.16949099	822.9851612
	Min Average	**Min Max**	**Min Sum**
	0.228131084	2.489307051	114.0655419

TABLE 4.1: Jitter values of joint estimation of still user over 30 seconds measured in mm

Consider the following matrix f_{st} populated with an example of 3D joints succeeding skeletonization:

$$f_{st} = \begin{bmatrix} -0.4234 & -0.5235 & 0.7234 \\ \vdots & \vdots & \vdots \\ 0.4864 & 0.2341 & -0.1241 \end{bmatrix}$$

All joints oriented to j_a:

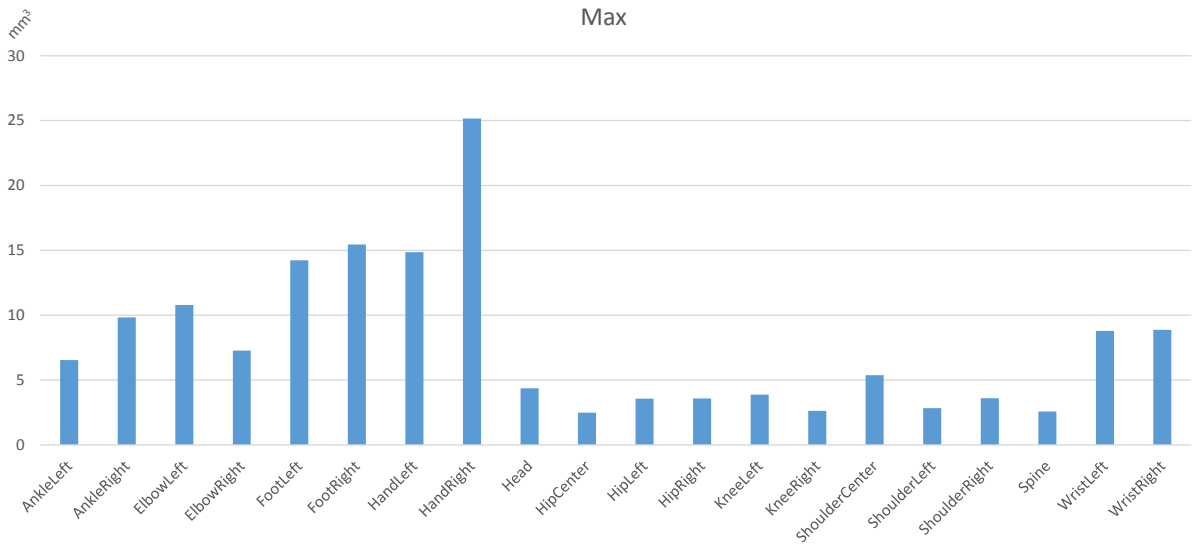


FIGURE 4.3: Plot of which joints had the greatest amount of jitter on still user.

$$f_{stj_a} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

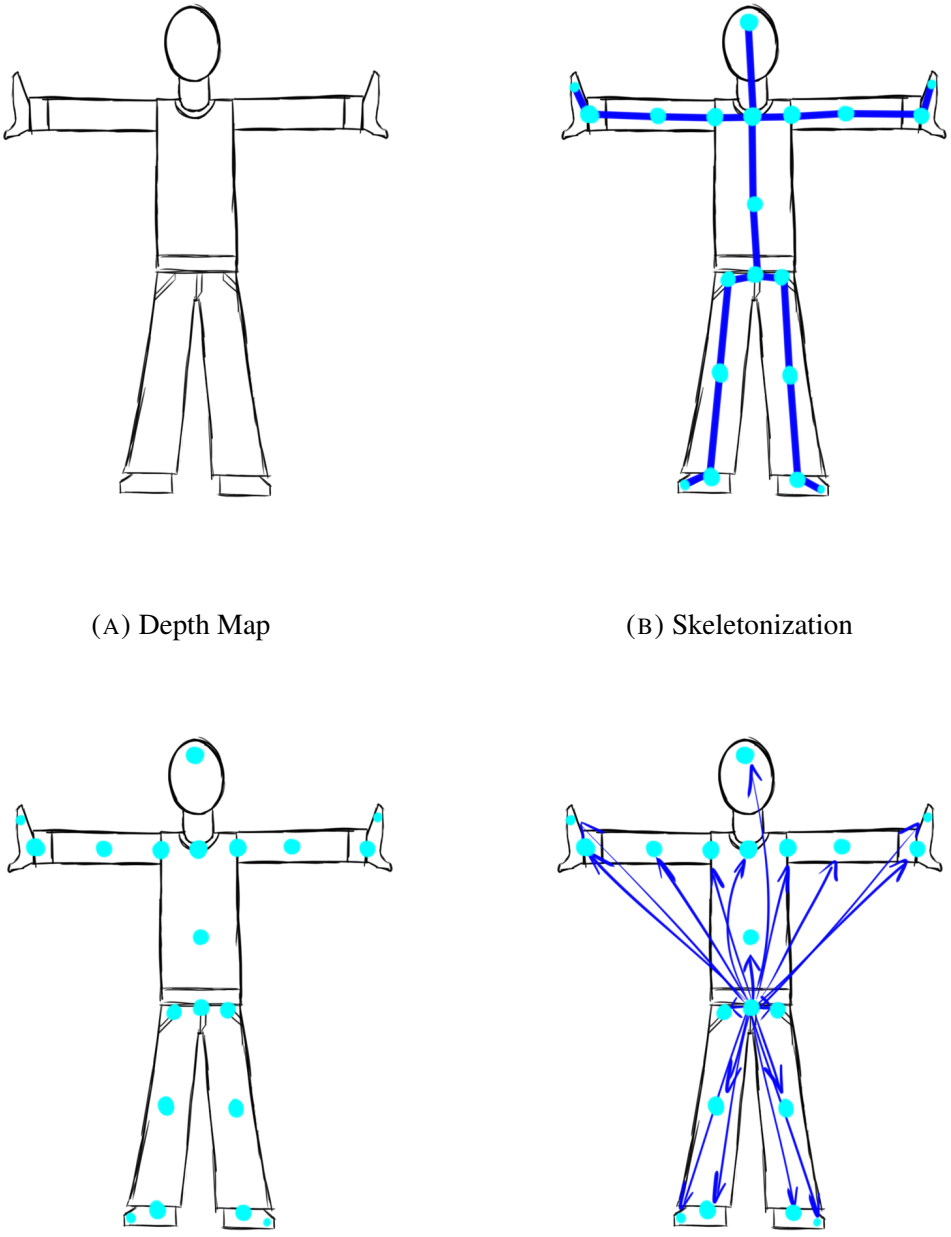
$$f_{st} = \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0.063 & 0.7576 & -0.8475 \end{bmatrix}$$

4.3 Scaling GDP Data

Data scaling is applied to the resulting GDP values to diversify the learning algorithm testing (as some algorithms have difficulty with or incompatibility with negative values) and to improve gesture compression for faster transmission by using fewer bits to represent each frame.

For computer training, a labeled matrix of size N rows and 60 columns of floating point anchored GDPs serves as a scalable input. The output consists of a column vector of integers denoting gesture class ID. Each input column (each of the 60 features) was scaled across all samples to lie in range [0-1] relative to the Min and Max of the values of the dataset.

The following is the normalization vector equation used to scale GDP data:



(A) Depth Map

(B) Skeletonization

(C) Joints relative to source

(D) Anchoring Least Variant Joint

FIGURE 4.4: Anchoring approach in order of process. The approach assumes that participant's orientation in the environment is not necessary. In brief overview, the participant's shoulders, hips, elbows, palms, fingers, knees, heels, toes, head, neck, and pelvis are indicated with respect to his/her waist (0,0,0)

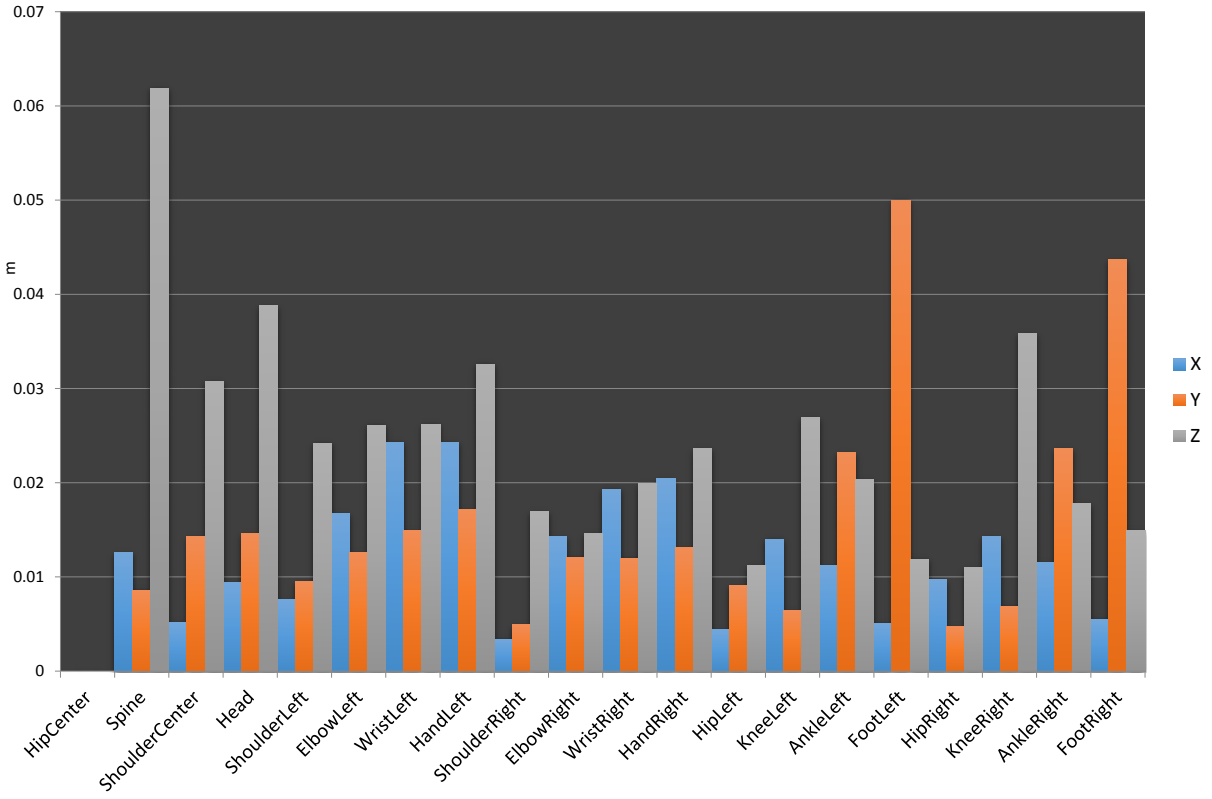


FIGURE 4.5: 2D plot of 20 GDPs from PJVA-20 a data samples

$$\hat{u} = \frac{u}{\|u\|} \tag{4.1}$$

This allows scaling of the feature matrix using the following equation:

$$f_{st} = \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ \hat{x}_{st} & \hat{y}_{st} & \hat{z}_{st} \end{bmatrix} where, 0 \leq x, y, z \leq 1 \tag{4.2}$$

4.4 Extracting Motion Feature Vectors from Scaled GDP Values

A set of GDPs is a single frame of a gesture sample, representing a static pose of a gesture, which can be enough data for accurate gesture recognition. As such, motion feature extraction

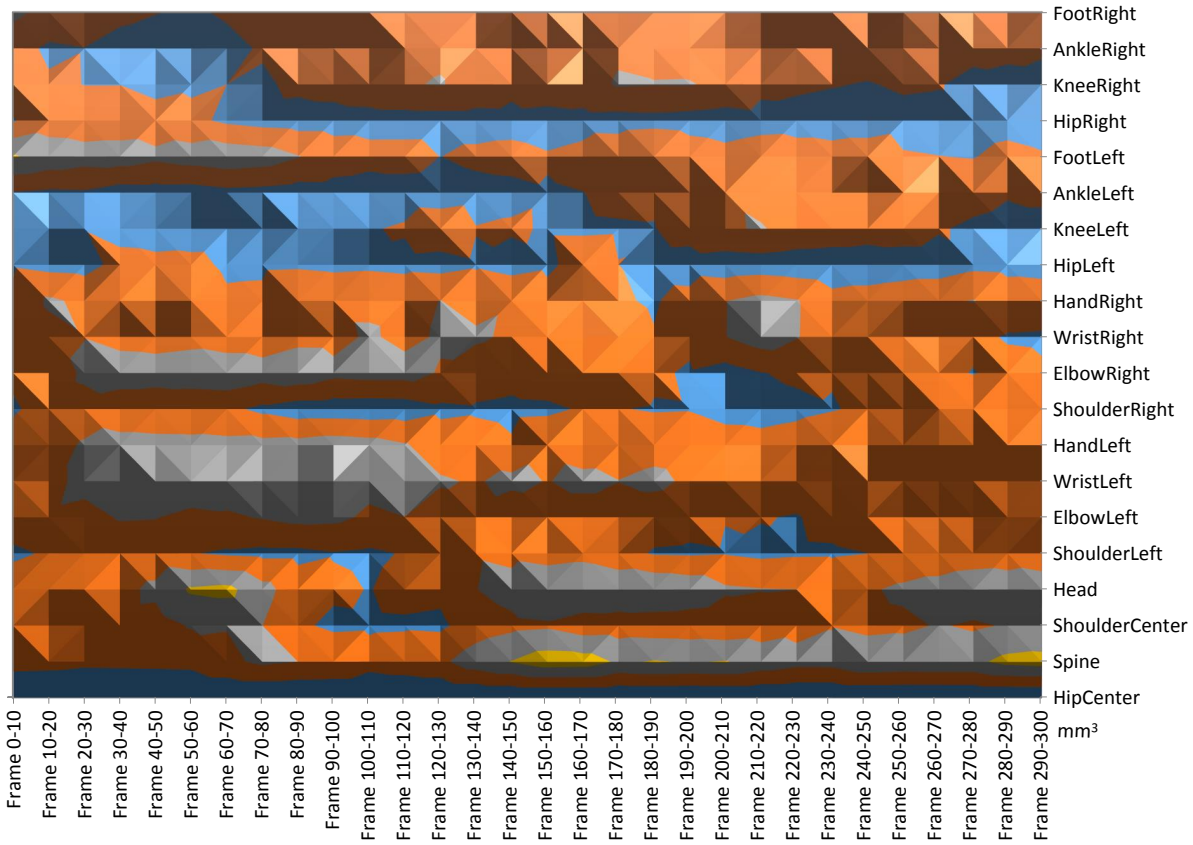
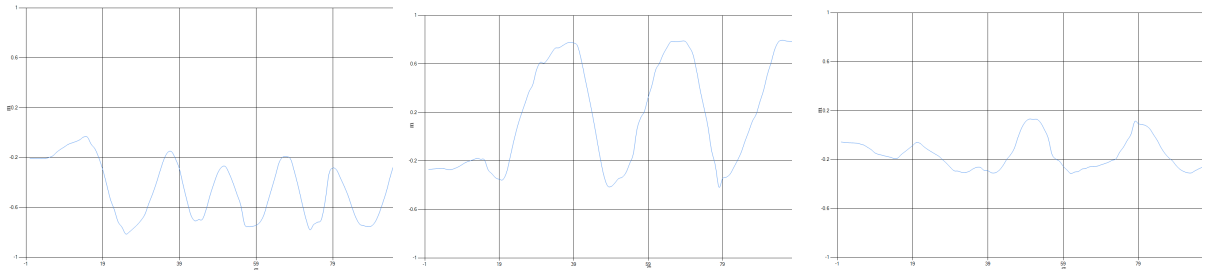


FIGURE 4.6: 3D view of 20 GDPs moving though time. the colours represent values between 0 and 1. This was done to reduce negative numbers out of the training sets as some learning algorithms may encounter learning difficulties.

Example of 20 GDPs

0 , 0 , 0
0.03179324 , 0.0703515 , 0.03879285
-0.004698373 , 0.4598181 , 0.07642901
-0.01441655 , 0.6236759 , 0.01204717
-0.1083868 , 0.3688791 , 0.1732941
-0.2167371 , 0.2107168 , -0.03710222
-0.32348 , 0.1906061 , -0.2504768
-0.3532059 , 0.1369402 , -0.2306565
0.1497031 , 0.3551072 , -0.08072317
-0.08622038 , 0.2413698 , -0.1561334
-0.2722499 , 0.1856924 , -0.2028849
-0.3458772 , 0.1695999 , -0.2467974
-0.05885772 , -0.05490351 , 0.005712748
-0.1134296 , -0.4642195 , 0.06664348
-0.1626059 , -0.8685466 , 0.08456528
-0.2237435 , -0.9078944 , 0.0222106
0.05694254 , -0.05767353 , -0.02099109
0.08054152 , -0.460308 , -0.05231309
0.1543252 , -0.8435999 , 0.0467453
0.09809816 , -0.8882636 , -0.004412532

FIGURE 4.7: Groups of GDPs are used to extract both static and motion feature vectors for gesture recognition applications



(A) Left Hand GDP X-Axis (B) Left Hand GJP Y-Axis (C) Left Hand GJP Z-Axis

FIGURE 4.8: Plot of the left hand GDP of a user performing a jumping jack

is not always necessary for gesture recognition, however, similar poses can exist in multiple gestures and can thus confuse classification accuracy between different gesture classes. For example, with static gesture data alone, a gesture recognition system would be unable to distinguish between a rotating limb moving clockwise and counterclockwise, because both gestures share identical poses. Thus, when creating a gesture recognition system, extracting motion features can increase accuracy by tracking subtle features that differentiate between similar gesture classes.

To maintain continuity in regards to motion feature vector length when training gesture recognition systems, constants need to be defined. Picking a length that is too short would make it difficult to distinguish similar gestures, while making the length too long would result in difficulty recognizing fast or subtle gestures. To increased the coverage of the resulting gesture data, I presumed that a gesture has two lengths: 900 motion feature vectors (45 frames, which is enough time to capture most gestures) and 300 motion feature vectors (15 frames, isolating the end of the gesture to capture very subtle gestures), as described in the following notation:

Consider a gesture that is comprised of 10 frames of three-dimensional data. Each frame therefore comprises a matrix having three columns corresponding to X, Y, and Z axes, with each column comprising about 10 rows and each row corresponding to a particular GDP. Each motion feature vector may correspond to a particular GDP. For this gesture, of a set of 10 frames of 3D skeleton data where each dimension includes only 10 GDP entries, the total number of motion feature vectors to be calculated by the system may be expressed as:
The total motion feature vectors = (10 frames) X (3 dimensions/frame) X (10 GDPs/dimension) = 300 motion feature vectors in total.
Therefore, for 10 frames of three-dimensional matrices of 10 GDPs, the system would need to

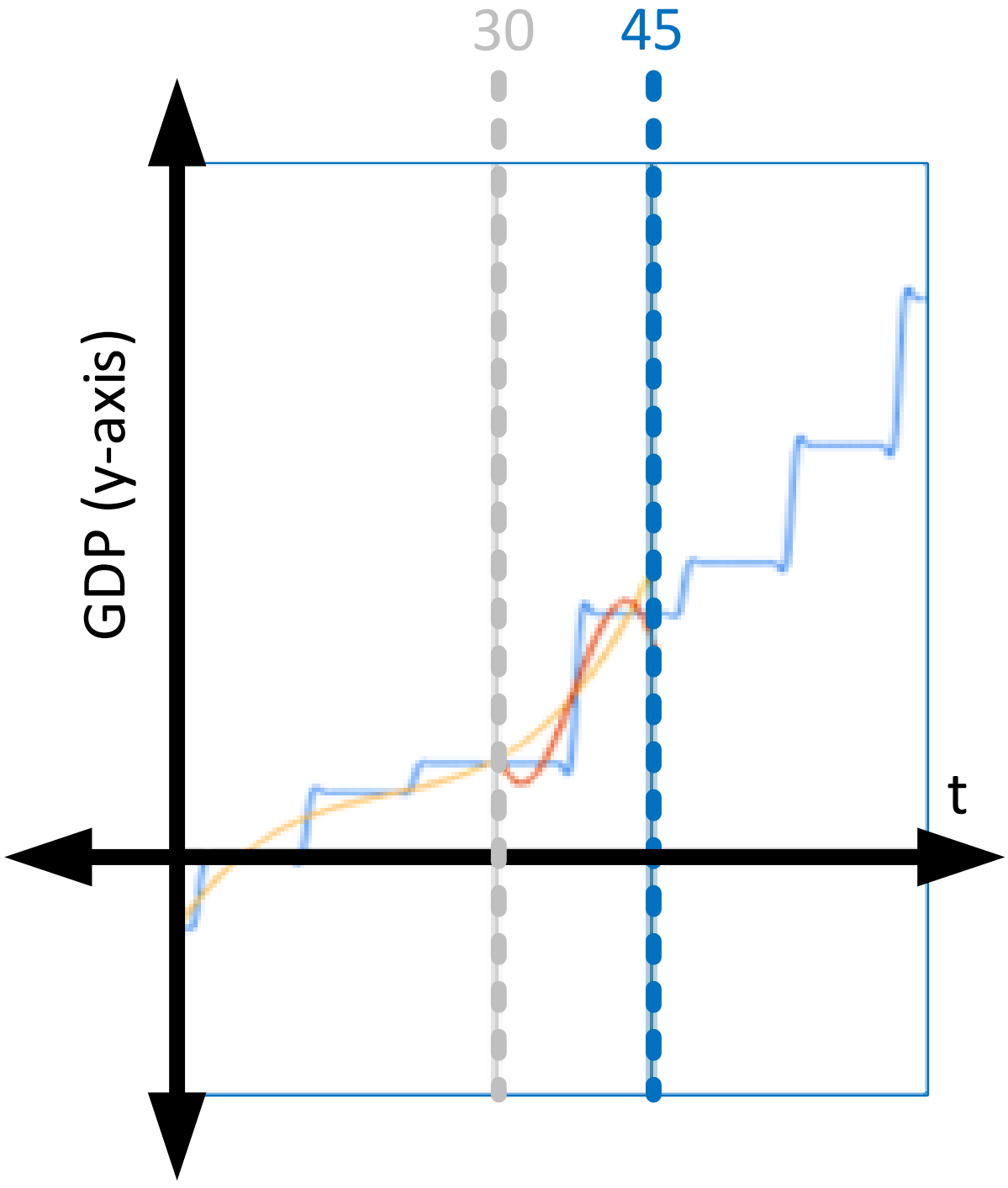


FIGURE 4.9: Plot of the gesture “clapping” from the perspective of the left hand GDP y-axis vs time

calculate or keep track of a total of 300 motion feature vectors.

The procedure uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called 'principal components'. Steps for transforming an N by x matrix into X into an N by x matrix Y:

1. Centralize the data by subtracting the mean value of each column from each element of the column
2. Calculate d by d covariance matrix

$$C = \frac{1}{N-1} X^T X \tag{4.3}$$

3. Calculate the eigenvectors of the covariance matrix C
4. Select m eigenvectors that correspond to the largest m eigenvalues to be the new basis.

Solving for a third dimension polynomial, both the 45-frame and 15-frame gesture classes are reduced to polynomial approximated values (see Table 5.7). For a motion gesture sample comprised of 1200 xyz-values, the motion data can be transformed into 160 motion feature vectors. Data representation for the sample types was tested. In these experiments, the recognition accuracy was highest with 2nd degree polynomials, 3rd degree polynomials, and 4th degree polynomials based on the data provided.

$$[GDPS_{i-l}, GDPS_{i-(l-2)}, ...GDPS_i, GDPS_{i-s}, GDPS_{i-(s-1)}, ...GDPS_i] \tag{4.4}$$

$$[GDPS_{i-45}, GDPS_{i-44}, ...GDPS_i, GDPS_{i-15}, GDPS_{i-14}, ...GDPS_i]$$

To reduce the noise and enhance the quality of the motion feature vector data, PCA was conducted on gesture samples to account for the variability (v). Additionally, the first and last 100 frames are trimmed off each sample to discard any redundant motions performed at the start or end of recording.

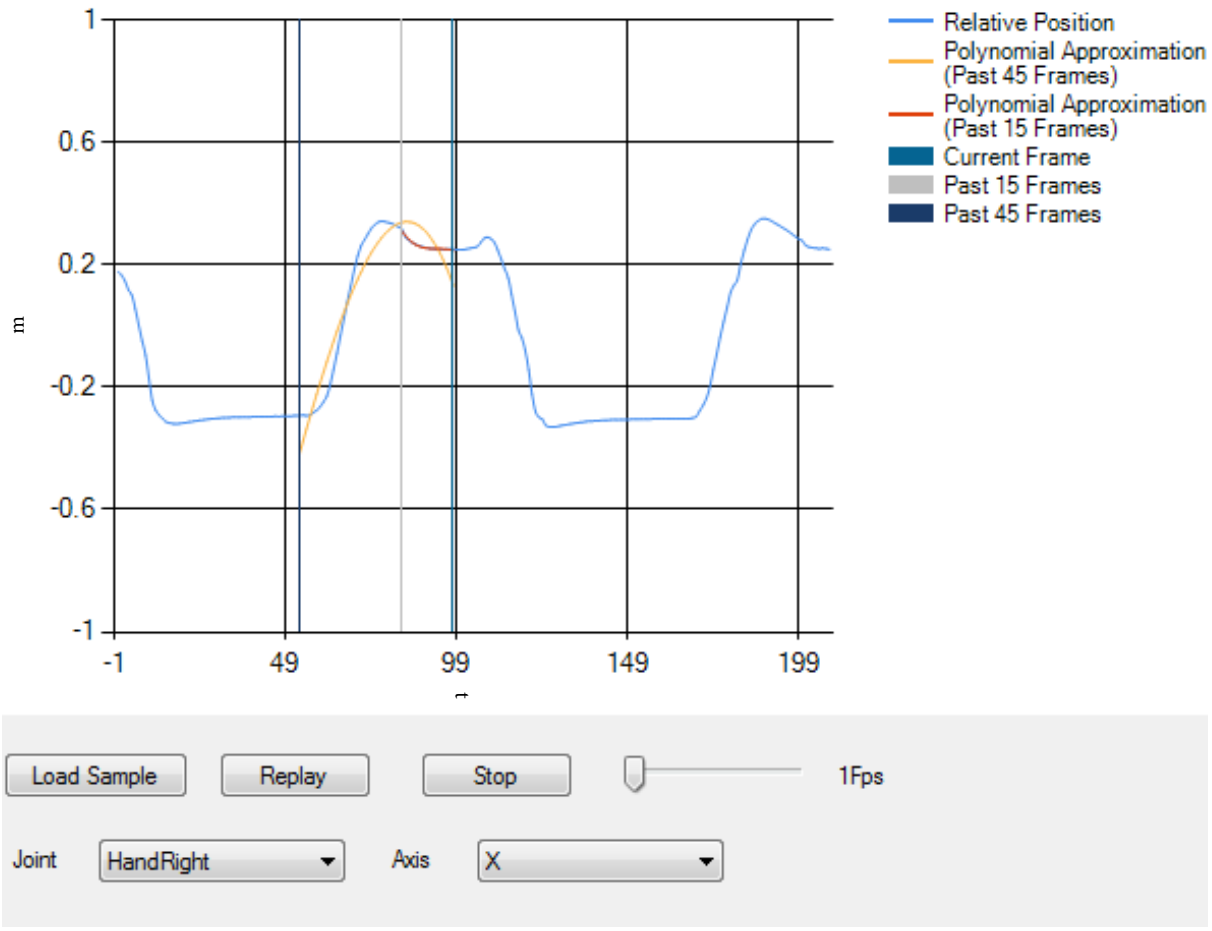


FIGURE 4.10: Third dimensional polynomial approximation of 45 and 15 frames of an x-axis right hand GJP

For PCA, X is an N by 480 Matrix, and N is the number of gesture feature samples. Each feature sample has 480 feature points where the feature sample is derived by approximating temporal motion by 4th degree polynomials. Two types of time frames are used (60 frames and 45 frames), as well as 20 body joints, 3 axes for each body joint, and a 4^{th} degree polynomial, which results in each feature vector having 480 feature points. The following are the steps performed:

$$Cv_i = \gamma_i v_i \tag{4.5}$$

30 eigenvectors with the largest eigenvalues selected:

$$V = [v_1, v_2, ...v_{30}],$$

$X(N$ by 480) sample feature matrix is multiplied by V, to dimensionally reduce $X'(N$ by 30).

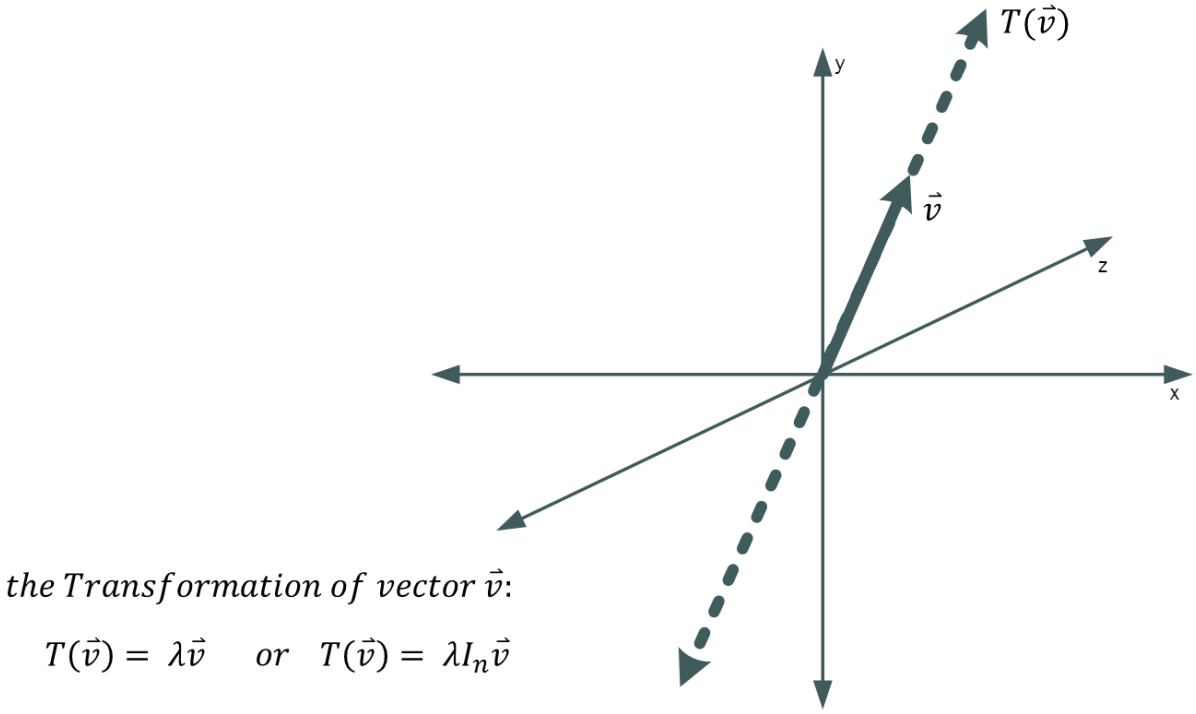


FIGURE 4.11: Plot of Eigenvector with legend

The resulting eigenvectors, which are being used as motion feature vectors that account for 98% of the variability in a dataset in some experiments, reduces the size of the motion feature vectors. For instance, when testing on the MSRC-12 dataset, rather than having 480 polynomial approximation numerators reduced from 1200 xyz-values, PCA transforms these into approximately 20-30 eigenvectors that represent all of the required information for gesture recognition. These eigenvectors then get fed into a feature matrix to categorize the gesture classes in a way that the computer can easily recall when recognizing gestures.

4.5 Summary

In this chapter I explained how I extracted motion features from the MSRC-12 dataset. After acquiring the GDP values from the 3D skeleton of a gesture sample, the PJVA approach uses

the gesture recognition techniques AMFE, polynomial approximation, and principal component analysis (PCA) to extract motion feature vectors.

In Chapter 5, I will test the PJVA approach by applying machine learning algorithms found on the WEKA database to label and classify the motion feature vectors extracted from the MSRC-12 data samples. I will present these PJVA results alongside the results of several of other studies that tested on the MSRC-12 dataset. I thus will thus justify the use of AMFE as a novel gesture recognition approach, demonstrate the potential of charades for sampling gestures, and I will establish Random Forest as the most effective machine learning algorithm for appropriately labeling the static PJVA-20 dataset that I constructed.

Chapter 5

Simulation Results and Discussion

"Shape Without form, shade without color,
Paralyzed force, gesture without motion;"

- T. S. Eliot

5.1 Introduction

In this chapter, I will test machine learning approaches for classifying and labeling gestures for recognition applications. I tested the RF algorithm with AMFE on the static samples from PJVA-20 dataset, achieving 98.5% accuracy for gesture recognition. I also tested my PJVA gesture recognition approach on the MSRC-12 dataset to see if RF combined with AMFE could exceed other academics' gesture recognition accuracy results. PJVA achieved 81.49% accuracy on the MSRC-12 dataset, which will be compared to the results of other researchers in this chapter.

This chapter is organized as follows. Section 5.2 on page 74 presents the results of the PJVA machine learning approach on my PJVA-20 dataset, including tests of several algorithms including SVM Poly (Section 5.2.1 on page 74), and SVM RBF (Section 5.2.2 on page 74), and Random Forest (Section 5.2.3 on page 76). The high accuracy results of the RF algorithm ensured its use in the final PJVA machine learning approach. These tests also demonstrate the

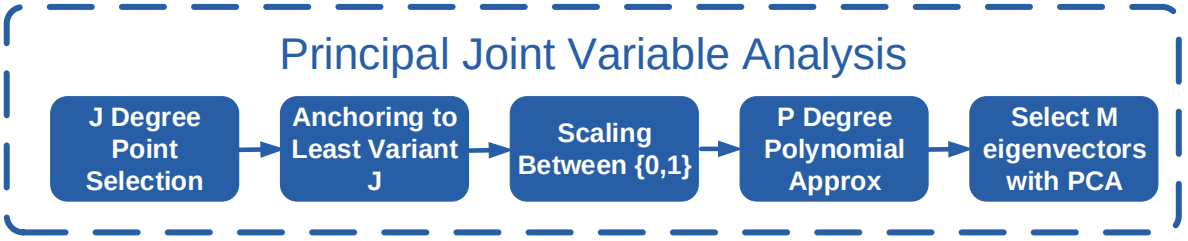


FIGURE 5.1: PJVA Flowchart explaining methodology

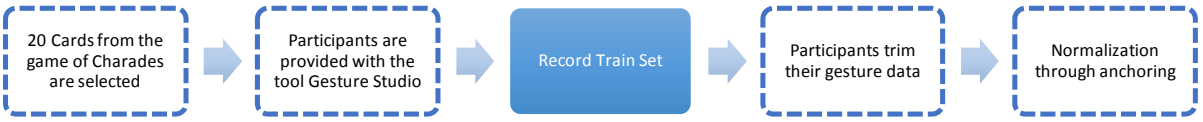


FIGURE 5.2: Proposed Approach compared to Fothergill et al. 2012 Experiment Process

high recognition capabilities of the PJVA approach on static images of the PJVA-12. This has proven to be difficult on the MSRC-12, which required the extraction of motion feature vectors to better differentiate between separate gestures and improve recognition accuracy. PJVA, on the other hand, achieved a 98.5% accuracy on just static gestures alone. This result primarily serves to demonstrate the quality of the samples captured using charades as a modality of instruction.

Finally, Section 5.3 on page 81 presents the results of the PJVA machine learning approach on the MSRC-12 dataset. By comparing these results to those of Zhang et al. [2] in particular, which was the best results on the MSRC-12 at the time of testing, it can been seen that the PJVA system has advanced the state-of-the art in gesture recognition. This is primarily attributed to the AMFE approach for orienting motion feature extraction approaches.

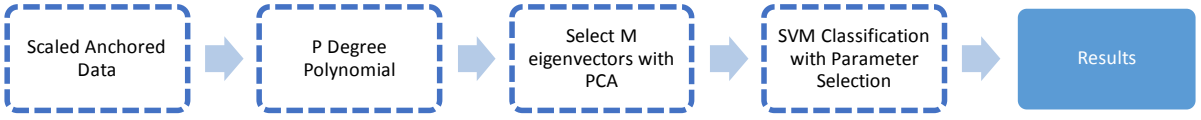


FIGURE 5.3: The Proposed Approach for SVM Learning compared to Zhang et al.

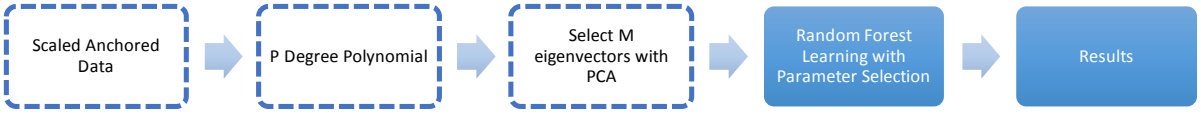


FIGURE 5.4: The Proposed Approach for RF Learning

5.2 Results of PJVA Machine Learning Approach on PJVA-20 dataset

5.2.1 SVM Poly Results

The Polynomial SVM classifier is non-linear and is a hyperplane in the high-dimensional feature space, defined by the Homogeneous Polynomial and Non-homogeneous Polynomial equations:

$$k(x_i, x_j) = (x_i \cdot x_j)^d \tag{5.1}$$

$$k(x_i, x_j) = (x_i \cdot x_j + 1)^d \tag{5.2}$$

Then, I calculated the Polynomial Kernel Grid Search Parameter:

1. Cost varied between .1 and 7812.5, scaling in steps of 5.
2. Gamma serves as inner product coefficient in the polynomial. Gamma varied between $1e^5$ and 113.90625, scaling in steps of 15.
3. Degree of polynomial varied between .01 and 4, scaling in steps of 7.
4. Coefficient varied between .1 and 274.4, scaling in steps of 3.

The most accurate recognition (97.64% accuracy) was obtained with a Cost value of 0.5, Gamma of 0.50625, Degree of 3.43, and coefficient of 0.1.

5.2.2 SVM RBF Results

The Radial Basis Function (RBF) SVM classifier is non-linear and the corresponding feature space is a Hilbert space of infinite dimensions, defined as the following function:

Gamma/Cost	0.1	0.5	2.5	12.5	62.5	312.5	1562.5	7812.5
0.00001	11.9088	11.0895	11.0895	11.0895	11.0895	28.017	65.6136	83.3715
0.00015	11.9088	11.0895	11.0895	11.9163	48.0545	80.878	89.702	93.8928
0.00225	11.9088	11.0895	37.1109	72.714	88.26	93.2538	95.5032	96.3559
0.03375	29.7226	67.0234	85.2106	92.8481	96.1389	96.9349	96.808	96.7915
0.50625	83.73	93.0102	96.5956	98.0217	98.3722	98.1005	97.8376	97.8376
7.59375	73.5057	92.8436	95.8249	95.921	95.9305	95.8808	95.8312	95.8312
113.90625	11.3813	19.893	40.9047	40.9047	40.9047	39.7976	38.6905	38.6905

TABLE 5.1: SVM RBF Kernel Performance Table for Gamma (horizontal) and Cost (vertical). These are variables that can be fitted to a SVM RBF recognition problem. PJVA was designed to automatically select these parameters during recognition, which in turn fine-tunes the accuracy of the SVM RBF learning algorithm's parameters. However, RF still turned out to be a more effective solution.

$$k(x_i, x_j) = exp(-\gamma \mid\mid x_i - x_j \mid\mid^2), \gamma > 0 \tag{5.3}$$

The RBF Kernel has been used to calculate the grid search for parameters:

1. The computational cost varied between 0.1 and 7812.5, scaling in steps of 5. The computational cost effects a soft margin that permits some misclassifications, increasing the cost of misclassifying points and forcing the creation of a more accurate model that may not generalize well.

2. Gamma varied between $1e^5$ to 113, scaling in steps of 15. The gamma parameter determines the RBF width.

3. The highest correct prediction was obtained for Cost 312.5 and Gamma .50625. Table 5.1 summarizes the RBF performance.

The SVM RBF is a complex model to illustrate. Nonetheless, a plot of the first 4 alphabetical classes is shown in Figure 5.5. This is a 2D plot created using the z-axis values of the spine and the y-axis values of the left foot. These axes were selected because the recognition system prioritized these points for accurate identification.

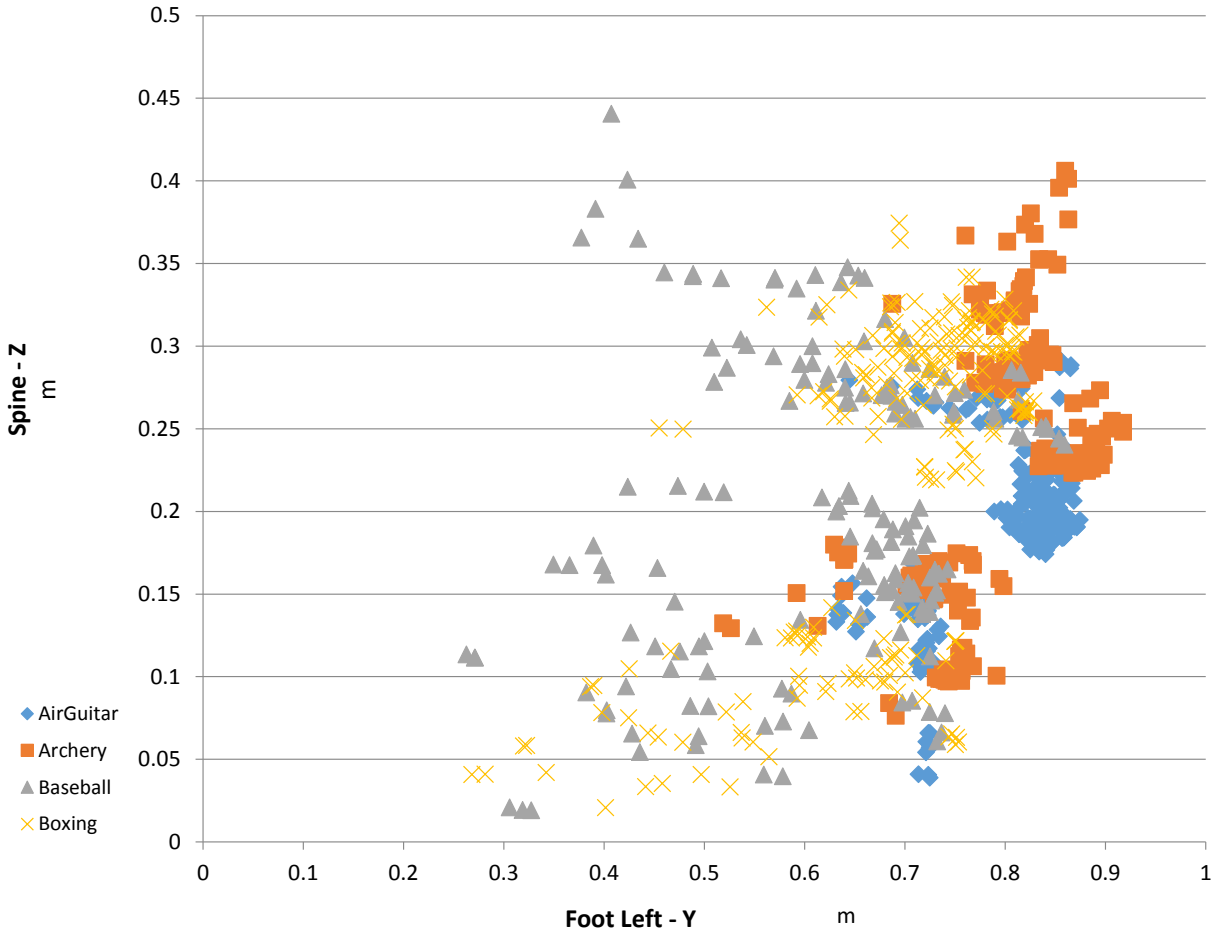


FIGURE 5.5: According to the SMV recognition results of this database, the y-coordinate of the left foot and z-coordinate of spine are the most useful features for classifying gestures. This figure shows heavy overlaps between classes due to projection to feature space of lower dimensions.

5.2.3 Random Forest Results

RF Parameter Selection using Polynomial Kernel:

1. Tree Height varied between 2 and 64, scaling in steps of 2.

2. The number of features considered varied between 4 and 12, with a multiple step of 2.

The most accurate prediction (98.13% accuracy) was obtained for Max Tree Height of 32 and 10 random features (see Figures Figure 5.6, Figure 5.7, Figure 5.8, and Table 5.2).

FIGURE 5.6: Full Tree View of 20 Gesture Classes with Random Forest Growth Results

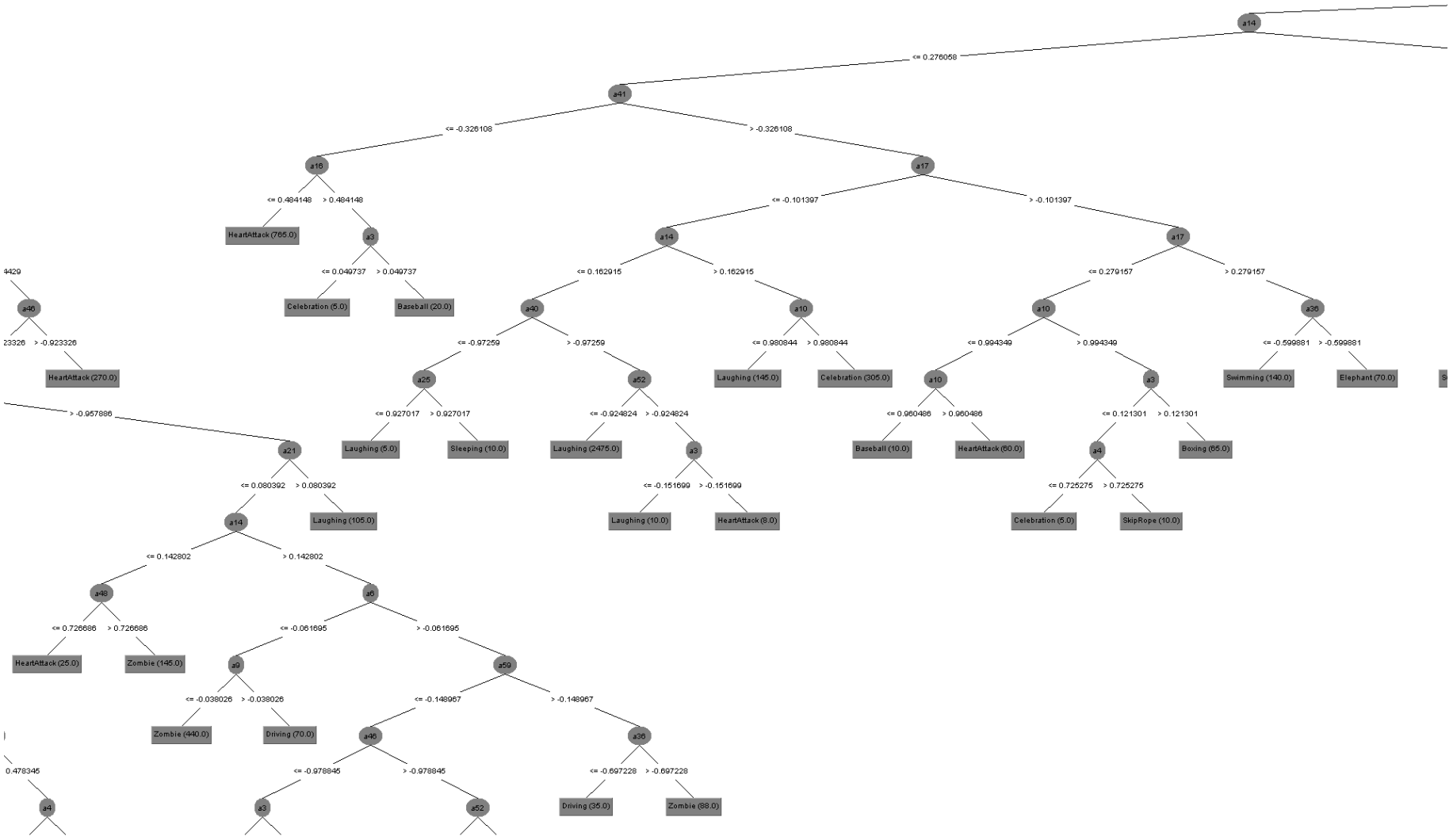


FIGURE 5.7: Tree View of 20 Gesture Classes with Random Forest Growth Results

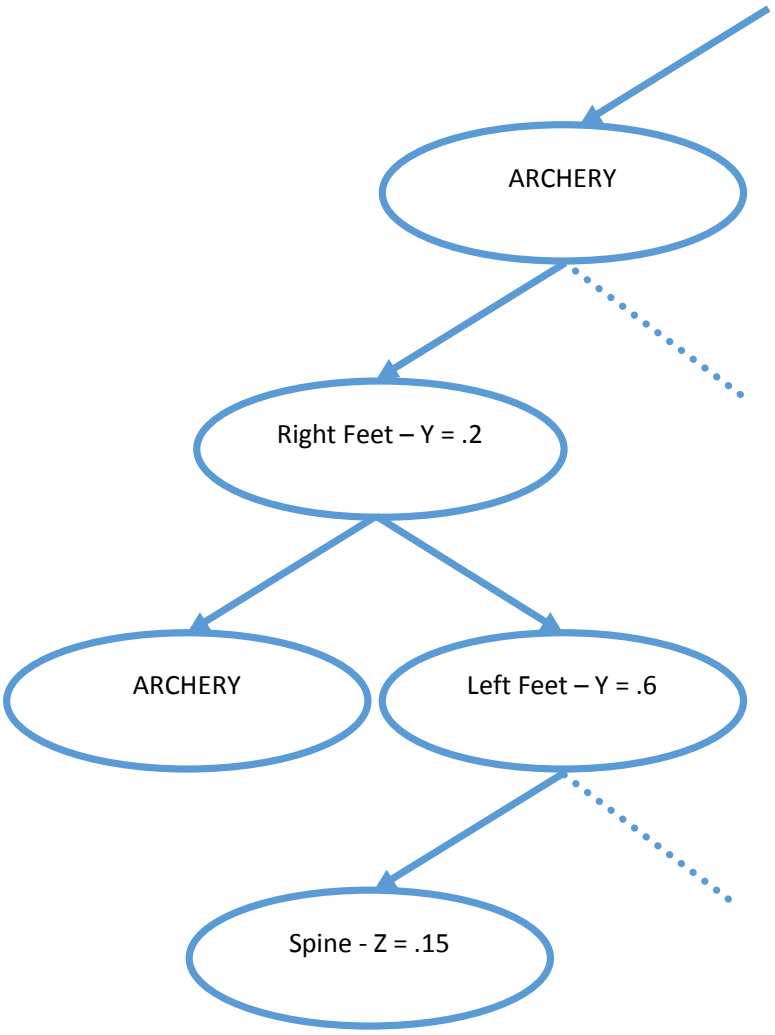


FIGURE 5.8: A small portion of a decision tree. For Left Feet - Y = .6 if motion feature vector has z coordinated of spine larger than .15, traverse to the right side; otherwise go to the left side of that node

Figure 3.9 is a screen capture of WEKA Explorer that was used for accuracy validation of the PJVA approach. WEKA [1] was developed at the University of Waikato, New Zealand.

Features/Max Tree Height	2	4	8	16	32	64
4	24.38	46.72	90.09	97.73	97.89	97.89
6	26.27	46.48	89.51	97.92	97.97	97.97
8	27.93	45.19	89.36	98.01	98.11	98.11
10	30.32	46	89.25	98.03	98.13	98.13
12	31	44.89	89.16	97.95	98.02	98.02

TABLE 5.2: Performance table with features(horizontal) and max tree height(vertical)

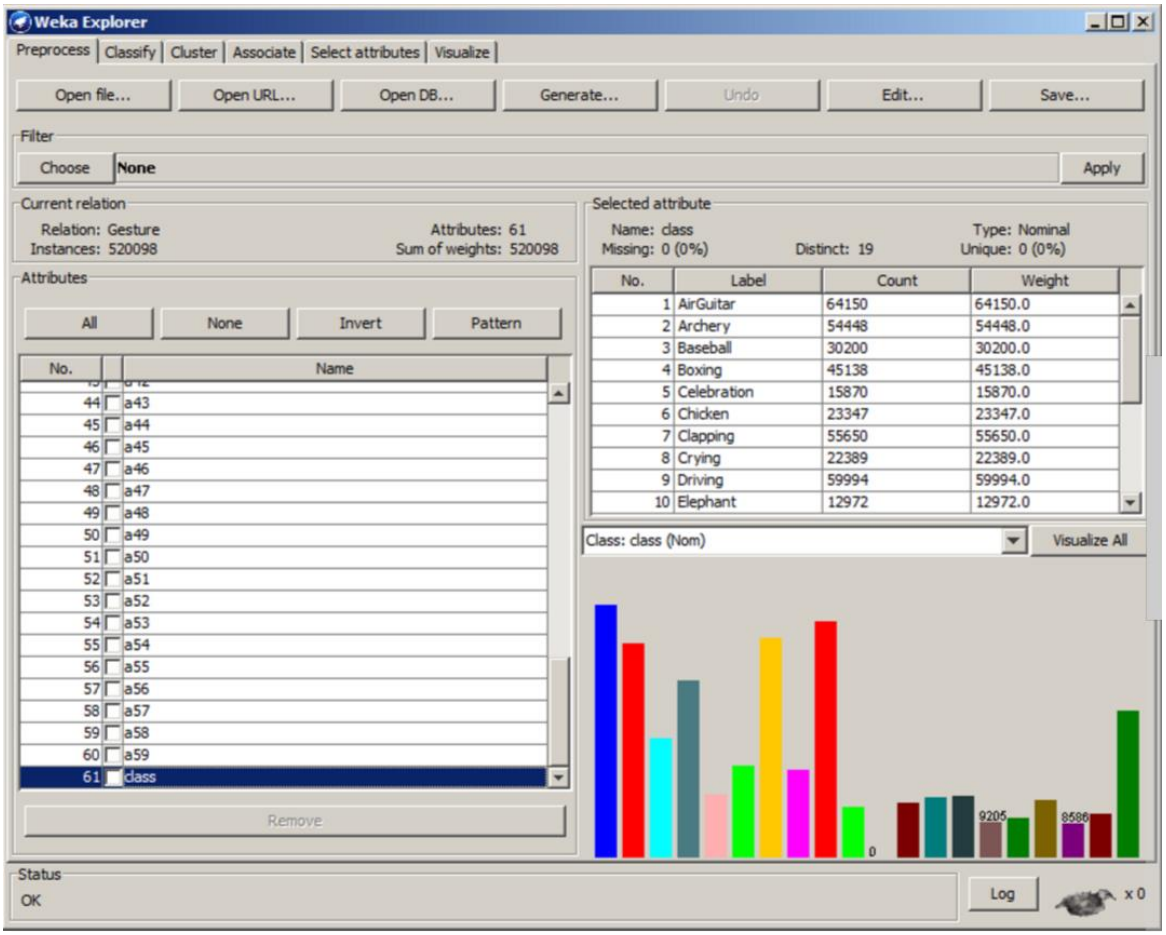


FIGURE 5.9: WEKA comparitive learning results on the gestures listed in Figure 3.9. WEKA [1] was developed at the University of Waikato, New Zealand and is the leading free and publicly available machine learning library. WEKA's quick comparison and extensive library make it the tool of choice for applied research in machine learning

5.3 Results of PJVA Machine Learning Approach on MSRC-12 dataset

This section will show the results of testing the PJVA machine learning approach for recognizing the 12 classes of the MSRC-12 dataset, on the combined iconic and metaphorical gesture dataset. PJVA results on MSRC-12 are shown with CMFE (Figure 5.8) and AMFE gesture recognition approaches (Figure 5.9).

	G10_Change _weapon	G12_Kick	G2_Duck	G4_Goggles	G6_Shoot	G8_Throw
G10_Change_weapon	68.20%	1.20%	1.30%	7.30%	19.50%	2.60%
G12_Kick	0.40%	91.80%	4.90%	0.90%	0.10%	1.90%
G2_Duck	1.30%	3.50%	87.00%	5.80%	0.50%	1.90%
G4_Goggles	2.30%	1.80%	6.30%	79.80%	6.70%	3.00%
G6_Shoot	1.30%	3.90%	0.70%	13.80%	80.20%	0.20%
G8_Throw	2.40%	19.20%	2.30%	0.70%	0.70%	74.70%
					Overall: 80.45%	

TABLE 5.3: Confusion Matrix results for PJVA system testing on the MSRC-12 dataset for recognizing Iconic Gestures with CMFE

	G10_Change _weapon	G12_Kick	G2_Duck	G4_Goggles	G6_Shoot	G8_Throw
G10_Change_weapon	79.70%	0.40%	3.20%	1.20%	9.10%	6.40%
G12_Kick	1.70%	87.30%	4.50%	0.20%	0.80%	5.50%
G2_Duck	0.90%	7.00%	86.70%	1.30%	2.90%	1.10%
G4_Goggles	1.90%	0.30%	5.50%	88.40%	3.40%	0.40%
G6_Shoot	6.90%	1.10%	1.30%	9.00%	80.40%	1.20%
G8_Throw	2.20%	11.20%	3.60%	0.70%	0.20%	82.00%
					Overall: 84.42%	

TABLE 5.4: Confusion Matrix results for PJVA system testing on the MSRC-12 dataset for recognizing Iconic Gestures with AMFE

A few specific gestures have proven to be much more difficult to recognize than the others. Wind it up (G5), Lift outstretched arm (G1), and Beat Both Hands (G11) have very low accuracy in recognition. In fact, discarding these three gestures, the accuracy of PJVA will rise to as

	G11_Beat_both	G1_LOA	G3_Push_Right	G5_Wind_it_up	G7_Bow	G9_HE
G11_Beat_both	33.60%	23.70%	2.20%	12.80%	1.90%	25.70%
G1_LOA	23.10%	47.60%	5.20%	14.60%	2.20%	7.20%
G3_Push_Right	8.80%	1.10%	64.50%	13.50%	6.20%	5.90%
G5_Wind_it_up	19.60%	11.30%	3.90%	49.90%	5.40%	10.00%
G7_Bow	6.40%	4.30%	5.30%	2.80%	77.00%	4.20%
G9_HE	20.70%	11.50%	0.30%	4.60%	1.80%	61.20%
					Overall: 54.58%	

TABLE 5.5: Confusion Matrix results for PJVA system testing on the MSRC-12 dataset for recognizing Metaphorical Gestures with CFME

	G11_Beat_both	G1_LOA	G3_Push_Right	G5_Wind_it_up	G7_Bow	G9_HE
G11_Beat_both	51.50%	22.60%	0.10%	12.70%	2.80%	10.20%
G1_LOA	12.20%	64.70%	0.30%	7.40%	7.40%	8.00%
G3_Push_Right	1.00%	1.70%	78.40%	10.20%	8.20%	0.50%
G5_Wind_it_up	14.20%	8.70%	0.30%	74.30%	1.50%	1.00%
G7_Bow	1.20%	3.70%	1.60%	5.80%	87.40%	0.20%
G9_HE	17.80%	7.30%	0.10%	4.10%	0.90%	69.80%
					Overall: 69.55%	

TABLE 5.6: Confusion Matrix results for PJVA system testing on the MSRC-12 dataset for recognizing Metaphorical Gestures with AMFE

high as 92%. 'Beat both hands' and 'lift outstretched arms' both involve lifting of arms above the head and bringing them down sideways. A similar problem exists with 'Wind it up', which partially resembles many other gestures.

5.4 Summary

This chapter summarizes the PJVA experiments undertaken to achieve a benchmark in gesture recognition in terms of correctness and coverage — not only on my self-built PJVA-20 dataset, but also on the MSRC-12 dataset.

Description	N1	N2	D	V (Eigen vectors)	Test Accuracy
Random Forest, 200 Trees	30	10	4	.95 (18)	76.79%
Random Forest, 200 Trees	30	10	4	.92 (14)	69.87%
Random Forest, 200 Trees	30	10	4	.98(30)	74.73%
SVM, RBF Kernel, c = 1, Gamma = 9.25	30	10	4	.95 (18)	62.45%
Random Forest, 200 Trees,	30	10	2	.95(26)	71.81%
Random Forest, 200 Trees Not normalized data	60	30	3	.95(17)	74.75%
Random Forest, 200 Trees	60	30	3	.95(22)	79.09%

TABLE 5.7: 12 Class MSRC-12 results. N1, N2 are the past frame count, D is the Degree of fitted polynomial, V is Variability accounted for by the selected eigenvectors

	G10	G11	G12	G1	G2	G3	G4	G5	G6	G7	G8	G9
G10	82.20%	0.70%	0.10%	0.10%	0.00%	5.10%	4.30%	3.80%	0.90%	0.30%	1.70%	0.70%
G11	0.50%	69.10%	0.00%	8.50%	0.70%	0.10%	7.20%	3.00%	0.70%	0.00%	0.00%	10.00%
G12	1.10%	0.50%	90.20%	2.60%	1.10%	0.10%	0.00%	0.30%	0.00%	0.20%	3.80%	0.00%
G1	0.10%	25.20%	0.00%	54.50%	7.00%	0.30%	0.10%	3.10%	0.40%	2.80%	0.10%	6.50%
G2	0.50%	0.60%	2.60%	1.90%	83.30%	0.30%	1.10%	0.40%	0.00%	6.30%	3.00%	0.00%
G3	13.80%	4.60%	1.30%	0.40%	0.90%	69.40%	0.00%	2.60%	1.70%	3.30%	1.80%	0.00%
G4	0.40%	0.20%	0.00%	0.30%	0.00%	0.00%	91.80%	1.70%	2.50%	0.00%	0.00%	3.20%
G5	0.80%	16.90%	0.10%	9.30%	0.30%	0.50%	7.30%	57.50%	6.20%	0.60%	0.10%	0.50%
G6	2.20%	0.10%	0.50%	0.40%	0.00%	0.10%	9.40%	0.90%	85.40%	0.10%	0.00%	1.00%
G7	1.00%	0.20%	4.70%	6.10%	10.20%	2.10%	0.10%	0.50%	0.00%	74.00%	0.90%	0.20%
G8	3.90%	0.00%	0.40%	3.50%	0.00%	1.40%	0.00%	0.50%	0.00%	0.00%	90.10%	0.20%
G9	0.00%	6.90%	0.00%	10.10%	0.00%	0.10%	13.30%	1.10%	0.60%	0.10%	0.00%	67.90%
												Overall: 76.28%

TABLE 5.8: Confusion Matrix results of PJVA machine learning approach on MSRC-12 with CMFE

	G10	G11	G12	G1	G2	G3	G4	G5	G6	G7	G8	G9
G10	81.90%	0.00%	0.10%	1.00%	0.20%	1.70%	2.20%	2.00%	10.60%	0.30%	0.00%	0.00%
G11	0.00%	62.00%	0.00%	13.90%	0.00%	0.00%	0.20%	5.50%	0.00%	0.20%	0.30%	17.90%
G12	0.00%	0.00%	95.80%	1.90%	0.10%	0.50%	0.10%	0.10%	0.00%	0.60%	0.80%	0.00%
G1	0.00%	39.30%	0.00%	52.20%	0.10%	0.00%	0.30%	6.30%	0.10%	0.20%	0.00%	1.50%
G2	0.00%	0.00%	0.30%	0.00%	98.50%	0.00%	0.20%	0.00%	0.00%	0.90%	0.00%	0.00%
G3	1.00%	0.00%	0.80%	0.20%	0.10%	93.40%	0.00%	0.20%	0.00%	2.30%	1.90%	0.00%
G4	0.30%	0.20%	0.00%	0.40%	0.50%	0.00%	88.00%	2.90%	1.60%	0.00%	0.00%	6.10%
G5	8.80%	7.80%	4.40%	5.30%	2.50%	14.80%	4.70%	44.60%	2.50%	2.00%	2.30%	0.30%
G6	0.00%	0.00%	0.00%	0.10%	0.20%	0.00%	1.10%	0.10%	98.30%	0.10%	0.10%	0.00%
G7	0.60%	0.40%	4.70%	3.60%	7.10%	1.40%	0.30%	1.00%	0.20%	80.20%	0.60%	0.00%
G8	0.60%	0.00%	0.00%	0.40%	0.20%	0.70%	0.00%	0.10%	0.00%	0.00%	98.10%	0.00%
G9	0.00%	2.00%	0.00%	5.10%	1.20%	0.00%	5.80%	0.70%	0.00%	0.30%	0.00%	84.90%
												Overall: 81.49%

TABLE 5.9: Confusion Matrix results of PJVA machine learning approach on MSRC-12 with AMFE

Zhang et al.'s results (see Sections 2.5.6 on page 37 and 2.5.7 on page 37) encouraged me to apply both SVM and RF to my own experiments on both the MSRC-12 dataset and the PJVA-20, in order to determine the most effective computer learning algorithms for my novel anchoring approach. I used several classifying algorithms to test my 20 class PJVA dataset: SVM RBF, SVM Polynomial, and RF. Of those three, RF was deemed the fastest and most accurate. Different learning parameters for these 3 classifiers have been tested. For classification, 70% of the database are trained while 30% are tested in real time recognition. AMFE and CMFE approaches are compared, and AMFE is confirmed as a benchmark approach after achieving higher accuracy results in testing on both the PJVA-20 and the MSRC-12 datasets.

Researcher and Approach	MSRC'12 Iconic Dataset	MSRC'12 Metaphorical Dataset	MSRC'12 Full Dataset
Song	76%-80%	56%-58.5%	-
Gomes WEKA			
Absolute Positions	-	-	71.69%
Absolute Position and Velocity	-	-	33.68%
Angles	-	-	68.28%
Angular Velocity	-	-	35.50%
Absolute Position Velocity and Single Angle	-	-	58.06%
Single Angle	-	-	59.90%
Gomes with averaging of frames			
Absolute Positions	-	-	71.69%
Absolute Position and Velocity	-	-	72.35%
Angles	-	-	68.28%
Angular Velocity	-	-	72.87%
Absolute Position Velocity and Single Angle	-	-	76.93%
Single Angle	-	-	59.90%
Gomes HMM	-	-	65%
Zhang Niave Bayes	56.33%	-	-
Zhang SVM	67.33%	-	-
Zhang Random Forest	80.69%	-	-
Zhang			
X,Y,Z Position	40.3794%	-	-
X,Y,Z Velocity	23.5772%	-	-
Joint Angle	65.3117%	-	-
Joint Angular Velocity	23.5772%	-	-
Joint Angle with X,Y,Z Position	56.6396%	-	-
Joint Angle with X,Y,Z Velocity	73.9837%	-	-
Joint Angle with Joint Angular Velocity	57.4526%	-	-
Joint Angle with X,Y,Z Velocity with Joint Angular Velocity	79.4038%	-	-
Joint Angle with X,Y,Z Velocity with X,Y,Z Position	63.4146%	-	-
Joint Angle with X,Y,Z Velocity with JAV with X,Y,Z Position	63.4146%	-	-
Bulzacki PJVA without Anchoring	80.45%	54.58%	76.28%
Bulzacki PJVA with Anchoring	84.42%	69.55%	81.49%

TABLE 5.10: PJVA recognition accuracy results compared to other experiments. These results demonstrate the potential of the AMFE approach for improving recognition accuracy on the MSRC-12 dataset

Chapter 6

Conclusions and Future Work

"Nothing happens until something moves"

- Albert Einstein

In this research, I first performed a comprehensive literature review of hand gesture and body gesture recognition techniques in the literature, in order to compare my results and help refine my approach. In particular, I conducted numerous tests on the MSRC-12 dataset —– the state-of-the-art for gesture recognition at the time of writing this dissertation —– to ensure that the proposed methodology (PJVA) and approach could compare. When tested on the MSRC-12 dataset —– driven by the CFME approach —– the proposed PJVA approach successfully recognized the 12 gestures with 81.49% accuracy. Additionally, testing on the static images of PJVA-20, the PJVA achieves an average accuracy of 98.5% on the 20 gesture class dataset.

I showed that charades is an appropriate modality of instruction for gesture recognition research and that the PJVA approach to gesture recognition —– chiefly comprised of AMFE, Polynomial Motion Approximation, Principal Component Analysis (PCA), and RF machine learning —– has created a benchmark in the gesture recognition field.

This dissertation involved experimenting and testing this approach against other researchers in the Computer Vision and Machine Learning fields. In particular, the MSRC-12 research gave me a benchmark point of comparison for research in this field. This study standardized a set of

instructions to be used when collecting gesture data from research participants. This gave me the motivation to find alternative modalities of instruction to see if I could build a robust dataset to meet and exceed the correctness and coverage requirements laid out by Fothergill et al.

This dissertation offers several contributions to the literature. At the time of collection, PJVA was the largest dataset of its kind for marker-less skeletal 3D points, at 20 gesture classes, and will be henceforth offered publicly as an open-source dataset for testing machine learning approaches and gesture recognition techniques. I have also established a benchmark method for modality of instruction by having research participants act out charades cards to capture skeletal data for gesture recognition. To create wider coverage of the data, I also enabled participants with the freedom to record their own gestures and self-edit the resulting gesture sample to isolate the gesture instance, which helped prevent overfitting of the data and improved computational accuracy in recognition.

Most importantly, I established a new method for extracting motion feature vectors through a novel recognition approach, AMFE. This is a foundational shift away from the current state of the literature and thus creates a benchmark in how to extract motion feature data from static 3D skeletal data for machine learning applications. This has been tested by applying PJVA to extract and label gesture data from both the MSRC-12 and PJVA-20 datasets, with high accuracy results.

I recognize that the meaning of a particular gesture may be subjective; however, this research has demonstrated that gestures are predictable over a large population and can be identified by one or more sequences of movements over a short period of time. When just considering pose, the PJVA approach is capable of recognizing gestures at the speed of >600fps.

Similar to a researcher that I admire, Robert Wang, I believe that the future potential in research and industry applications are significant.

6.1 Call to Action

To further advance the benchmark of gesture recognition, I offer the following calls to action:

1. I call for more testing using the PJVA approach on different gesture datasets in order to widen the scope of testing this approach. In particular, the AMFE approach should now be employed to increase the accuracy of gesture recognition systems on several datasets in the literature. These tests would further demonstrate the new benchmark by testing the PJVA gesture recognition approach in further experiments and comparing them to other experiments in the field.

2. I call for further analysis of the PJVA-20 dataset with different combinations of gesture recognition and machine learning algorithms, which will put the quality of my gesture samples to further testing.

6.2 Future Work

In the future, this research has the potential for practical applications that are not confined to the laboratory, providing functionality in the real world for a wide variety of situations that cannot afford to have false positive results, e.g., recognizing the gestures of doctors during surgery, the movement of patients during physiotherapy sessions, and detecting a burglar or a dangerous threat before the situation has escalated.

The PJVA system has the potential to achieve a machine-based intelligence strong enough to categorize and detect different types of gestures in many industry settings such as electronic communication, interactive entertainment, and security systems. Gesture recognition can also serve as an interactive tool in environments where keyboards, mice, or vocal commands are not practical or possible. As such, gesture recognition can be used for human body and gesture tracking in any business industry, science laboratory, or technological application —– particularly with its potential for collecting real time analytics of body language at public events.

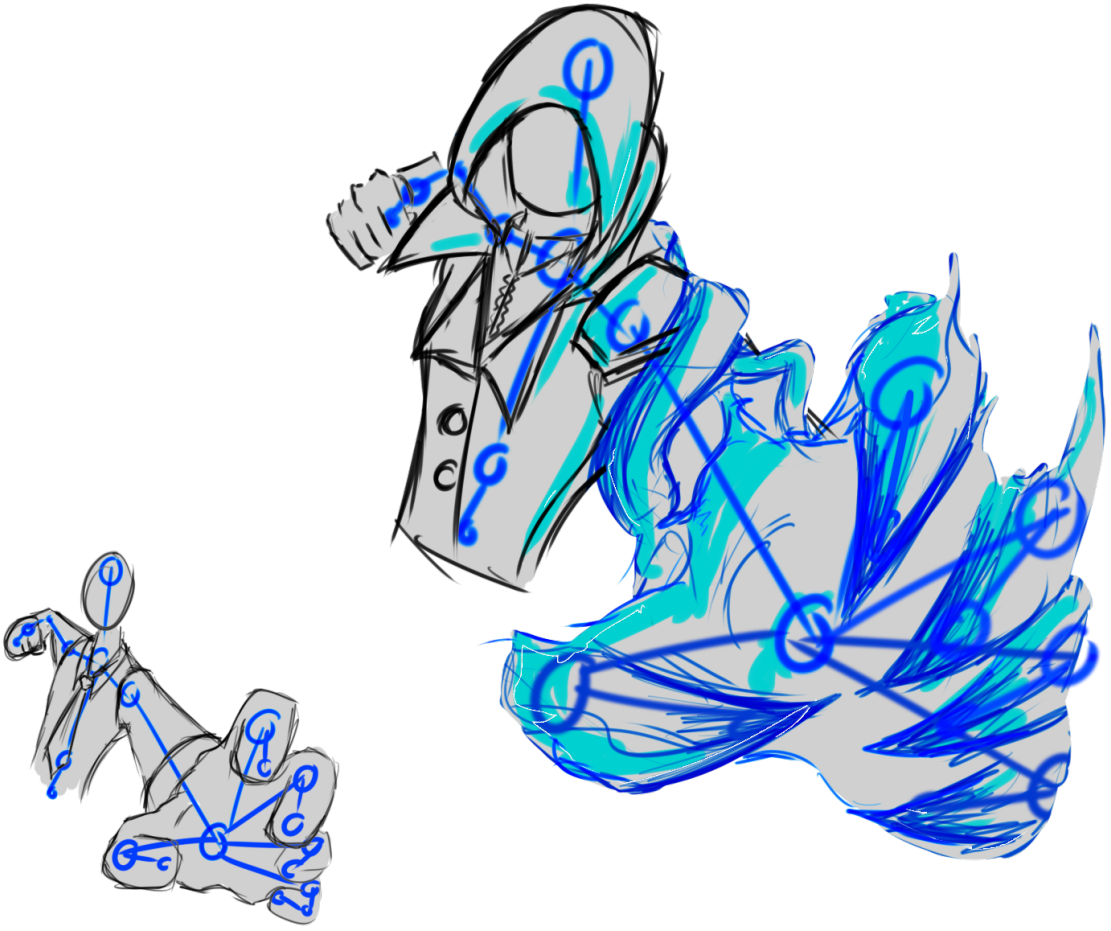


FIGURE 6.1: The proposed approach could be used to make games more realistic by reducing the time it takes to recognition a gesture and provide near immediate feedback for a greater sense or immersion

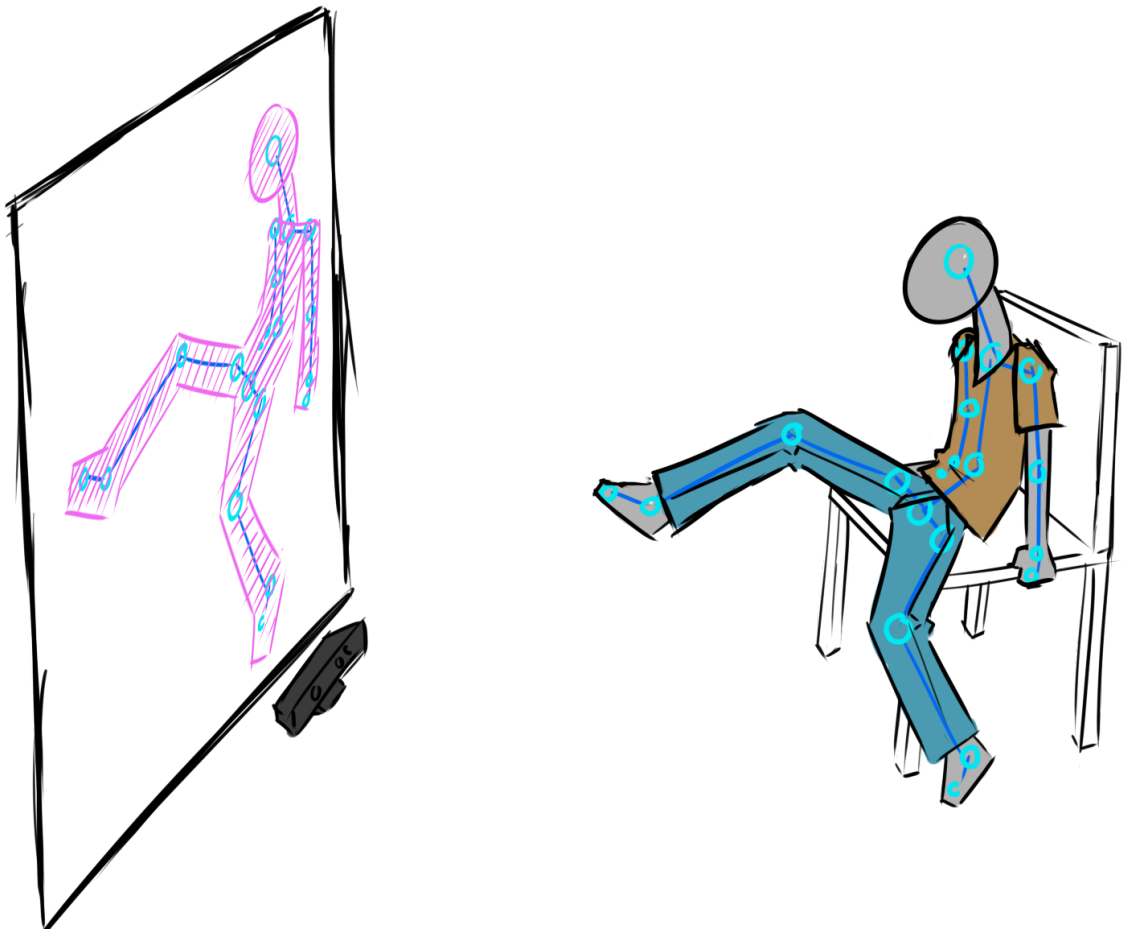


FIGURE 6.2: The proposed approach could be used to enhance the recognition accuracy of physiotherapy treatment in hospitals or in the home to allow quicker recovery and improved training regiment

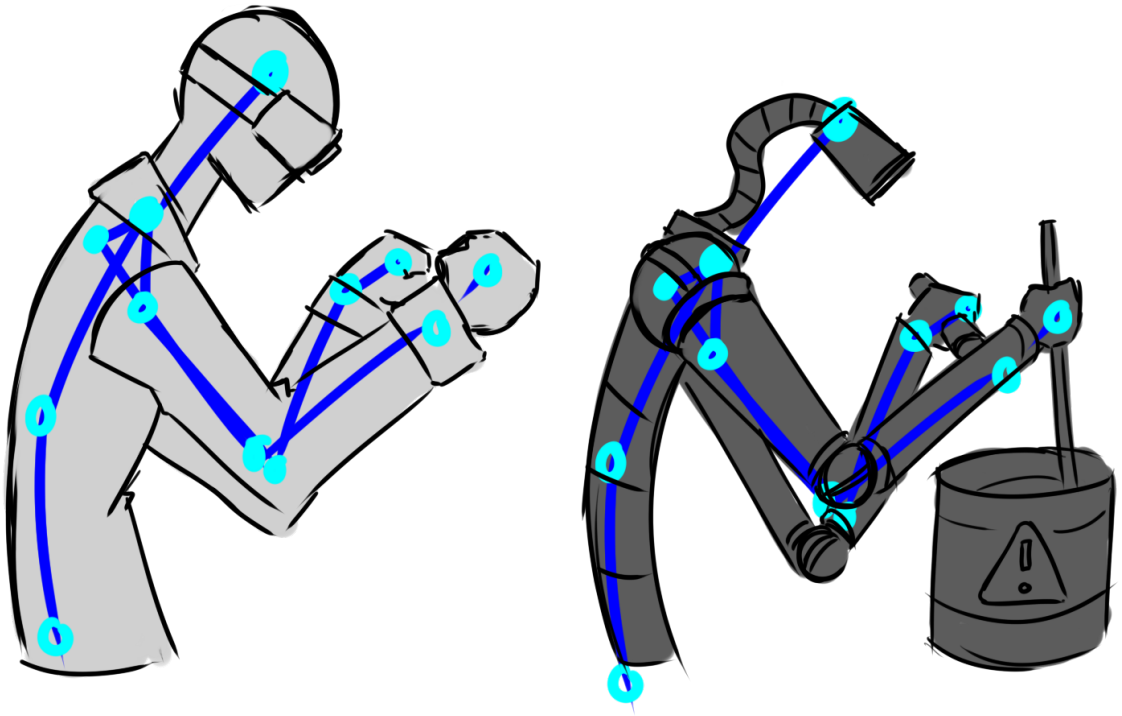


FIGURE 6.3: The proposed approach could be used to allow industrial control of machinery and the linking of more complex controls than are currently available through traditional approaches

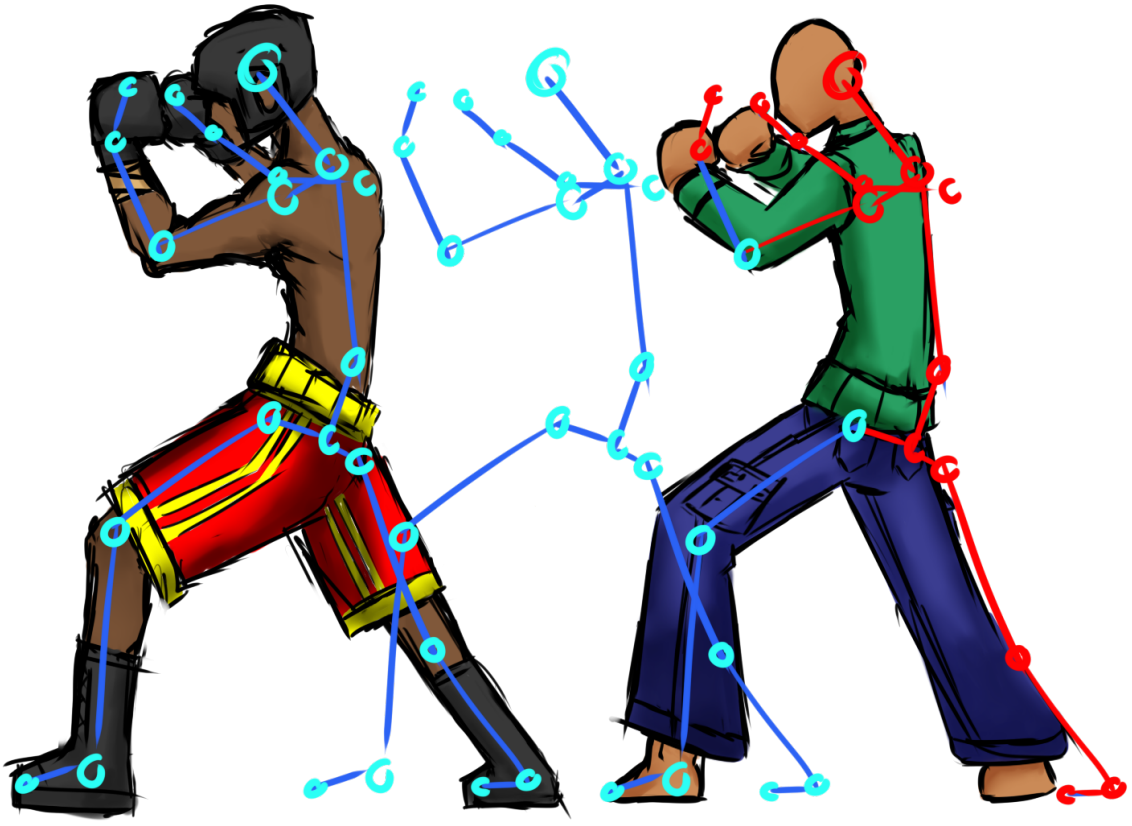


FIGURE 6.4: The proposed approach could be used to search and index motion capture datasets/collections to allow for faster retrieval

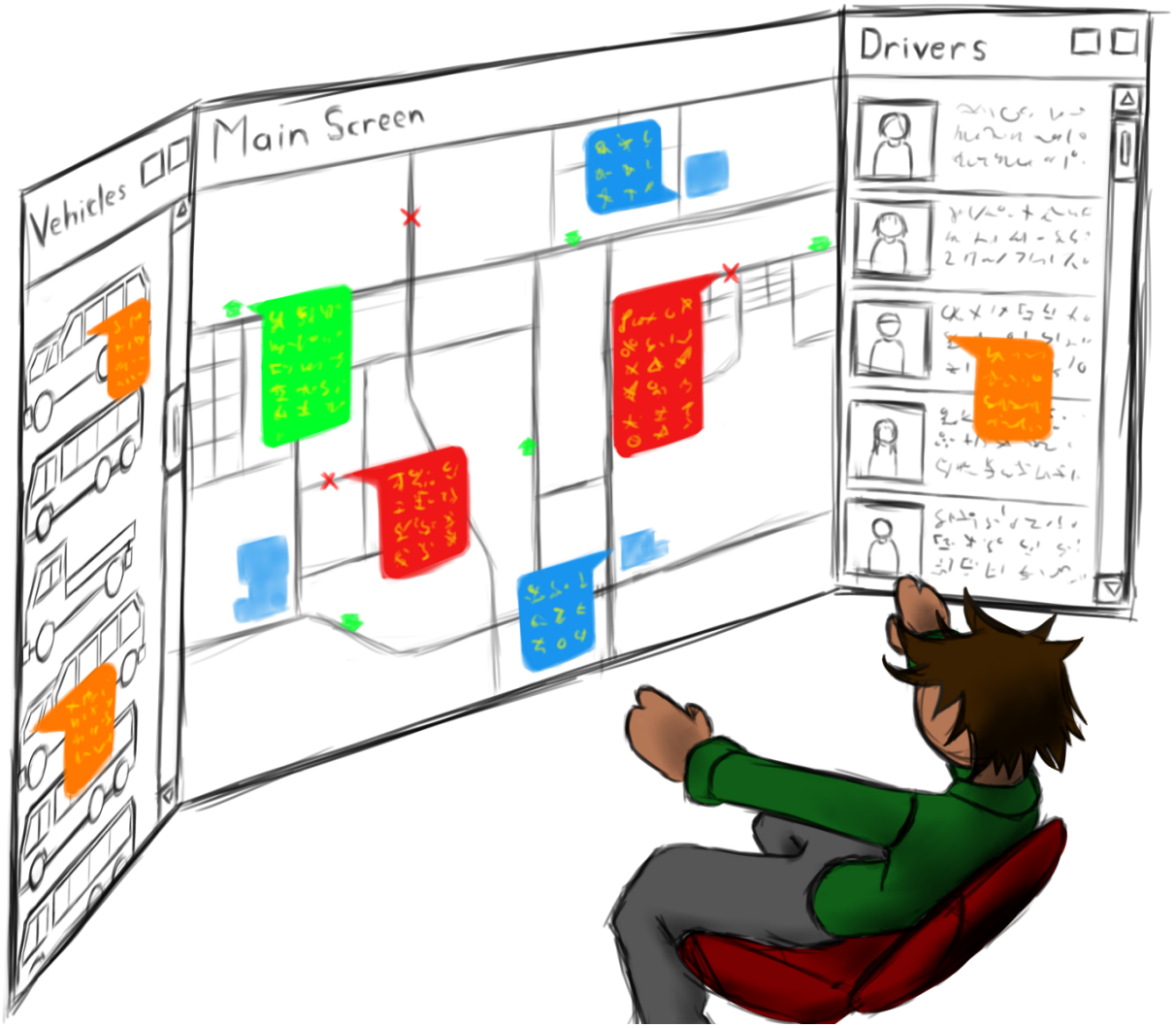


FIGURE 6.5: The proposed approach could be used as an interface tool for transit control operator to reroute transit vehicle and move data between many screens

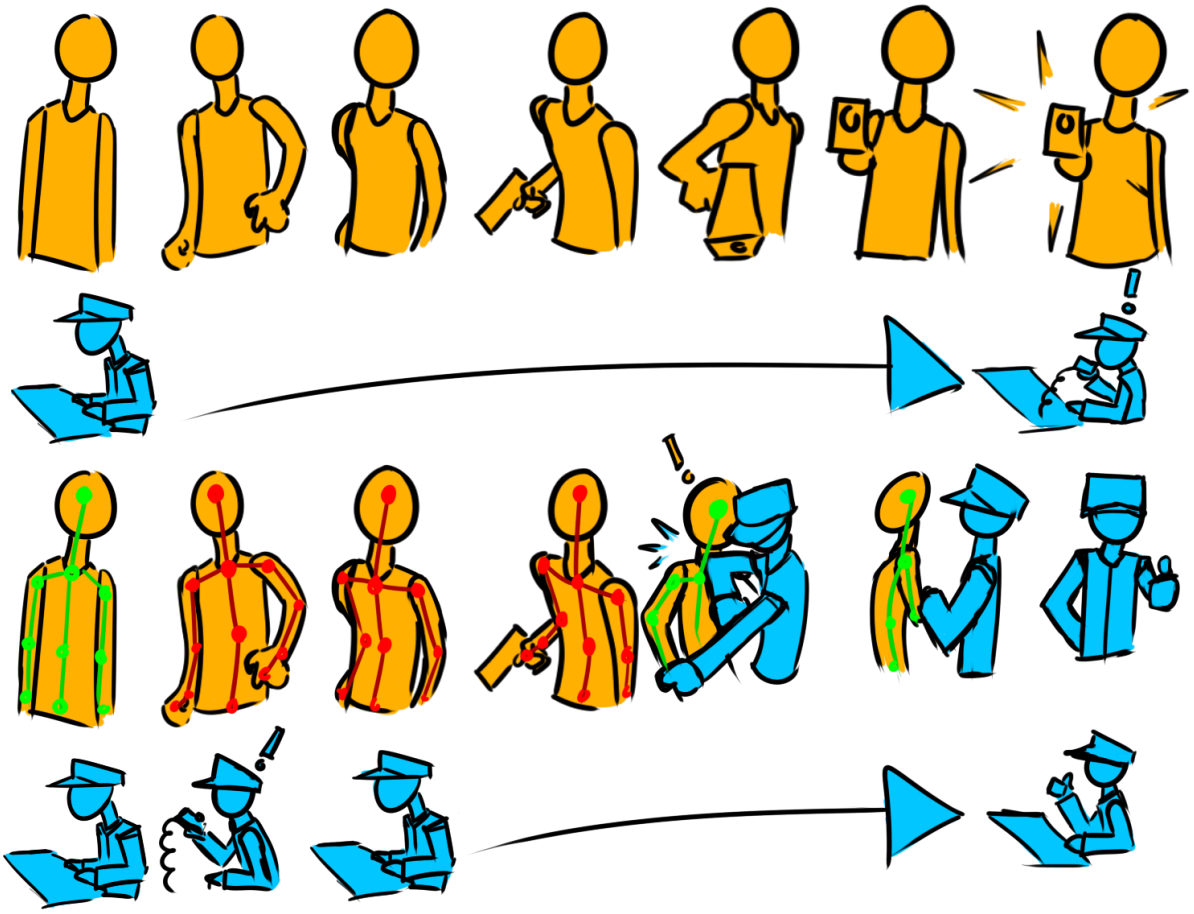


FIGURE 6.6: The proposed approach could be used to identify dangerous or criminal gestures in a space and allow for real time alerts and aid in quicker response time

Appendix A

Experiments in Depth Based Hand Gesture Recognition

In earlier experimentation(2009), several problems were discovered with a vectorization approach published around the same time period[24, 94], as illustrated in the top portion of Figure A.1. The colour vector approach proved to be computationally heavy, and though in later experiments we were able to optimize the approach of outlining just the hand, it added even more to the computation when using x86 processors. One additional problem came from the similarity of colour between the hands and faces of users, which is almost always impossible to distinguish, even with prior frame tracking. Colour sampling is the most common approach to gesture recognition, but so far has been only a stop-gap method and proven to be a dead-end approach.

A very important assumption about hand gestures is that they are never behind the face or body of the user and rarely beside, but always in front. This suggests that depth/distance from a source can play a major role in locating the gesture, perhaps far more than the colour sampling approach commonly utilized, hence the benefit of using IR. The middle row (Figure A.1) was sourced by an IR camera built in-house using a standard Microsoft Lifecam webcam. The IR camera solves several problems, the most important being that colour data that is otherwise not required is now replaced with depth data relative to the brightness in the stream, Fanello[95] describes a similar approach to building an IR camera. The bottom row is a vectorization of the live IR camera stream. Notice the two rightmost gestures on the bottom row of Figure A.1. The

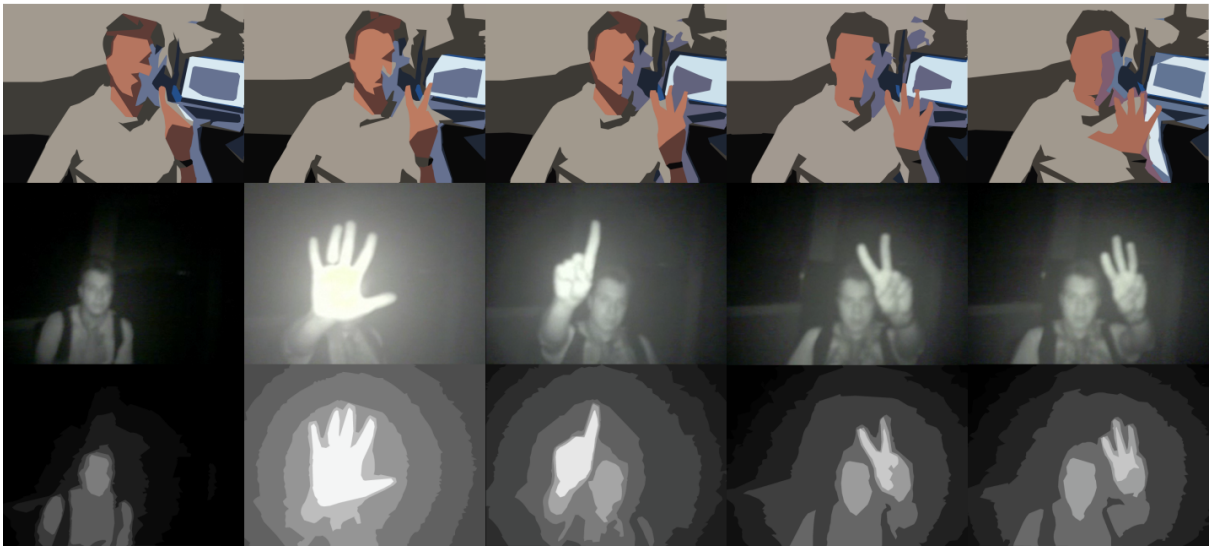


FIGURE A.1: Original Colour Vector vs. IR Camera vs. IR Camera Vector

complete gesture is not in the brightest layer, it is actually surrounded by the second brightest area. The segmentation algorithm knows to check only the brightest (presumed closest) and its surrounding second brightest (presumed second closest) area of the source. None of the Vector IR experiments contained relative gesture data in a layer brighter/closer than the second layer. This is better illustrated in Figure A.2, where we see a 3D mesh model of both the IR source and its vector equivalency. The IR source is visibly very smooth; the IR conversion is far flatter, with fewer gradients that are more distinct, and easier to quantify. Looking at the right side of Figure A.2, we can see that by using just the IR source, we would have to design a variable threshold system. The IR vector allows us to check only the top two distinct tiers (red and orange in this case) for the gesture, regardless of threshold.

Acquisitioning/Segmentation

Between 2009 and 2010 dozens of segmentation methods have been tested on IR video frames and though testing results were positive on many, ultimately the segmentation approach was narrowed down to an autosegmentation approach. The two key reasons being, segmentation is not the focus of the original research, and the approach was compatible for extracting a single hand contour shape. For testing, the Graph Cut approach was implemented, Grieg first applied graph cut to image processing in [96]. Figure A.3 and Figure A.4 show two different gestures, five fingers and two fingers, with four Graph Cut samples where the region parameters differ.

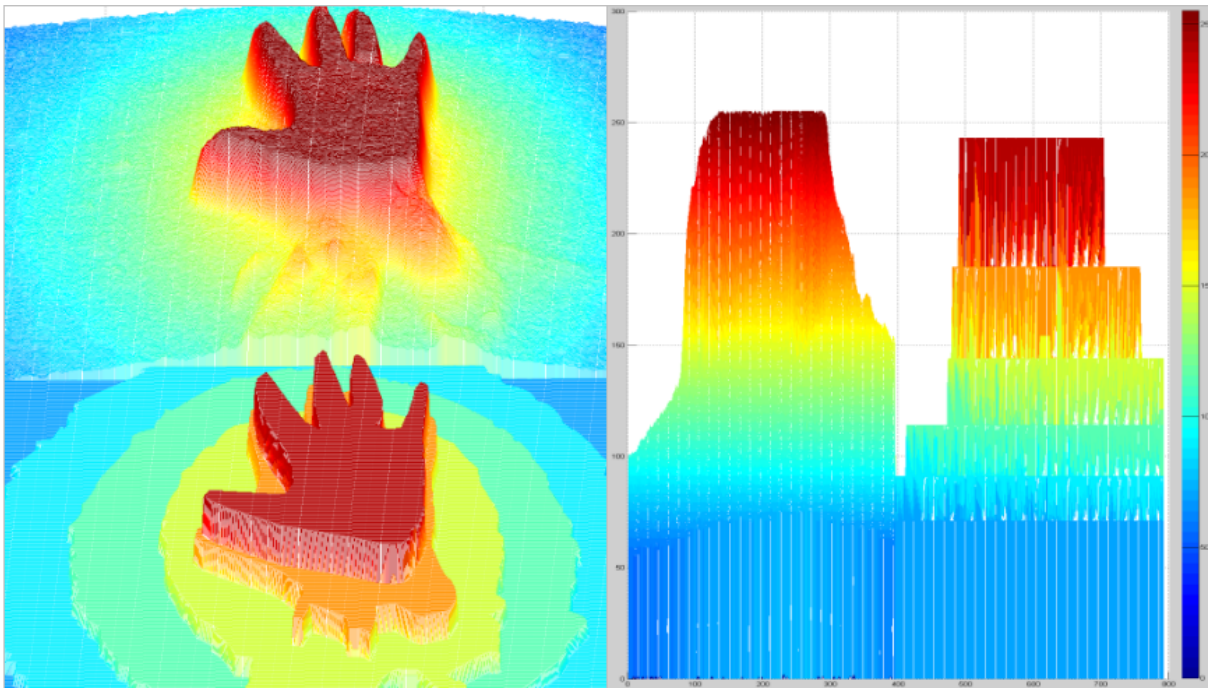


FIGURE A.2: 3D Depth Model based on IR Camera and IR Vector Conversion

From top left, five regions, top right, four regions, bottom left, three regions, and bottom right is two regions. Top right, four regions became the best choice because visually it seemed to have the most completed hand shape. Though Graph Cut is an autosegmentation technique, the regions are randomly labelled. K-means clustering can be used to identify the brightest region. Rather than applying the K-mean approach, to save time, the following steps removed the other three regions by using the mean average of white pixels to find the most probable center of the hand gesture. Figure A.5 shows the approach which takes the Graph Cut with random labelled regions to a contour. Top left is the Graph Cut, while top right uses the original source frame with a fast gray threshold applied to it. Mean average is applied to the gray threshold image which points to a pixel region that is likely to be where the hand is located. Once the algorithm knows the pixel location, it point the Graph Cut to the correct region which results in the bottom left images. Once a single shape is segmented, a contour image can be created, and the building of a descriptor begins.

Final approach will more than likely vary because the acquisition tool(camera) will be changed very shortly. Since there is no shortage of segmentation approaches, and because of the extensive testing done to narrow down this methodology, segmentation should not be a major problem during the next step of experimentation.

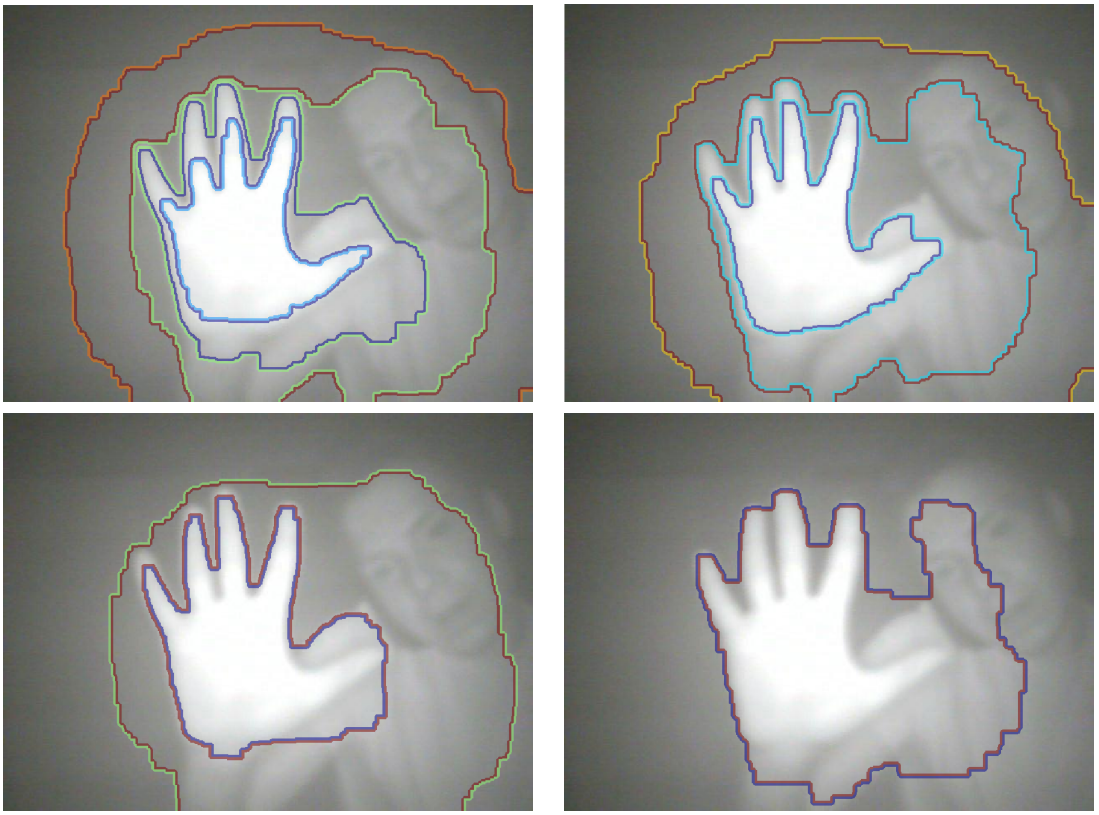


FIGURE A.3: Graph Cut 5 Finger Example

Descriptor Design

For early testing a Fourier Shape Descriptor (FSD) was implemented. Before converting the contour to a shape descriptor, all conversion must be implemented in the same way. For our testing, to begin the shape descriptor, the topmost y axis contour pixel was selected, based on the mean average x axis. From there, the contour moves clockwise in a south east direction. Figure A.6 shows two Fourier Shape Descriptors. The contour on the right has more noise associated with the sample. It is important to note the values of the axes. Mean averaging was used to center the contour on the x and y axes. Having the axis centered allows our descriptors to be consistent. When building a good descriptor, scaling and rotation are important. So far, a solution to the scaling problem has been devised by scaling the starting point of the contour to 1 for the y axis and 0 for the x axis. This has not been implemented on the samples in Figure A.6. A solution for the rotation problem is still being researched, though solvable. When considering a shape descriptor, it is worth remembering that noise in the shape tends to be high frequency, while the general shape of the hand tends to be low frequency. Figure A.7 shows the

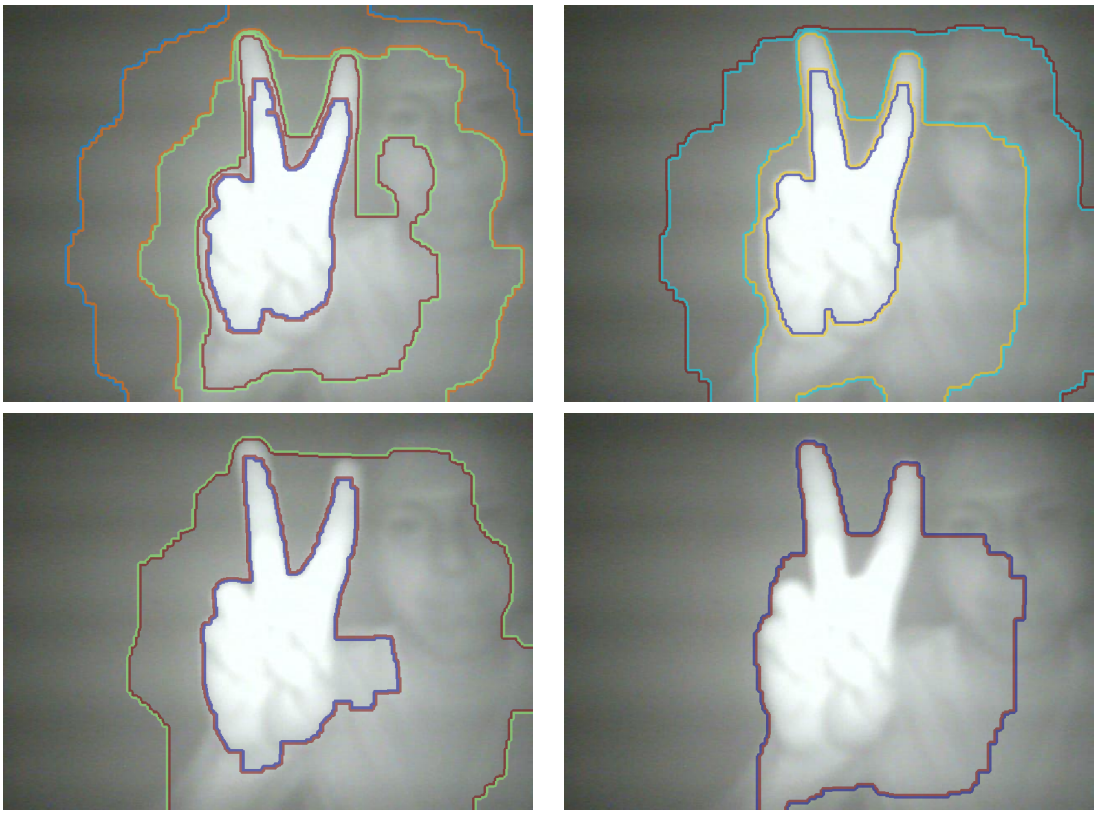


FIGURE A.4: Graph Cut 2 Finger Example

Fast Fourier Transform(FFT) of the X-axis of the FSD. The example has over 1200 features to describe to the shape. Figure A.8 shows the key points. We will try to cut the 1200 features down to under 40 features. Comparative results of different feature numbers will be discussed in the next section.

The FSD is being used only temporarily. One original contribution of the methodology will be in a new descriptor approach. Though the FSD has worked well, it is a 2D approach. A new descriptor approach will need to contain 3.5D data. The 3.5D data will benefit by being represented in a 2D format; therefore, the data will need to be downconverted. Downconverting is helpful because it simplifies classification, as discussed in the next section.

Describing 3D Shapes

[97] uses a Spherical Fourier transformation (Spherical harmonics) to describe the features of 3D shapes. Vranic applies a Spherical Fourier transformation to the function $r(u)$. The Fourier transformation on the sphere uses the spherical harmonics functions $Y_{l,m}$ to represent

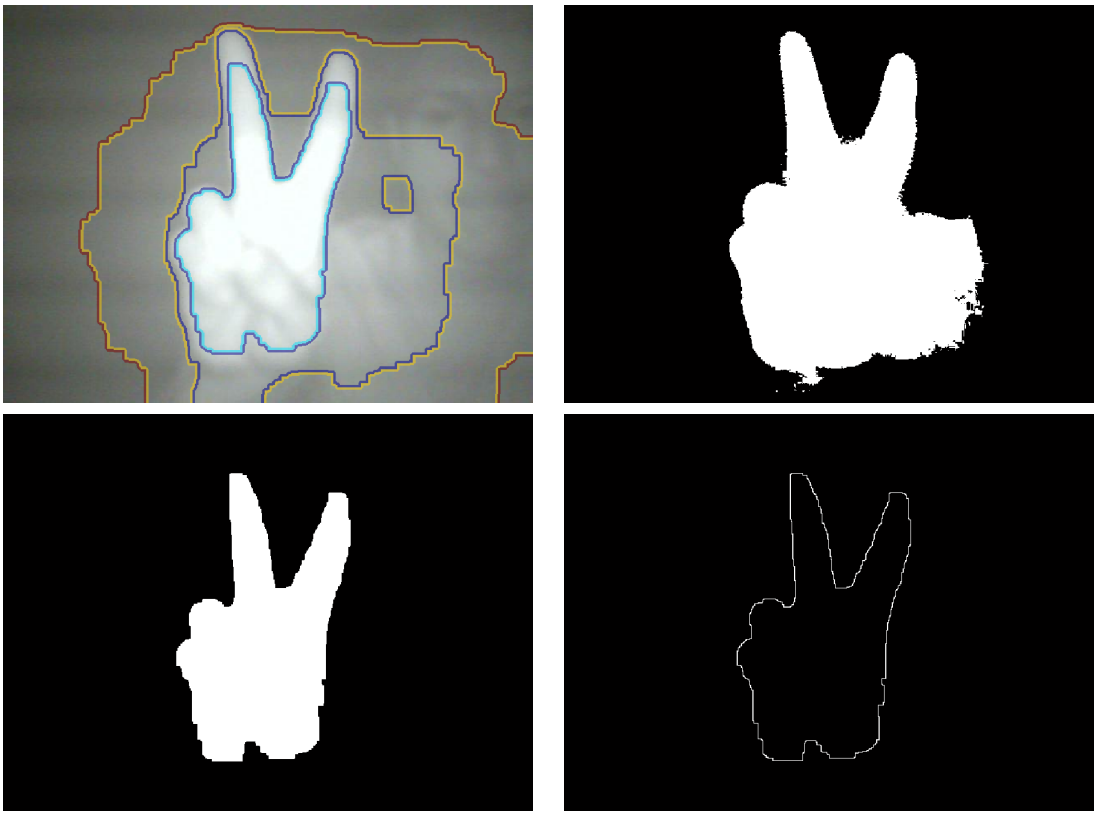


FIGURE A.5: Graythresh Mean Average Region Identification

any function $r \in L^2(S^2)$ as,

$$r = \sum_{l \geq 0} \sum_{|m| \leq l} \hat{r}(l,m) Y_{l,m} \tag{A.1}$$

where $\hat{r}(l,m)$ denotes a Fourier coefficient. The spherical harmonics are orthogonal with respect to integration over the surface of the unit sphere. The complex Fourier coefficients can be calculated by a spherical Fast Fourier Transform algorithm applied to samples taken. Spherical harmonics are quite commonly used throughout 3D modeling classification.

In [98], Shih proposes a novel feature descriptor called elevation descriptor for 3D model retrieval. The approach is designed for invariant translation and scaling of 3D models and rotation. "Six elevations are obtained to describe the altitude information of a 3D model from six different views. Each elevation is represented by a gray-level image which is decomposed into several concentric circles. The elevation descriptor is obtained by taking the difference between the altitude sums of two successive concentric circles. Shih describes experimental results which show that the proposed method is superior to other descriptors, including spherical harmonics, the MPEG-7 3D shape spectrum descriptor, and D2."

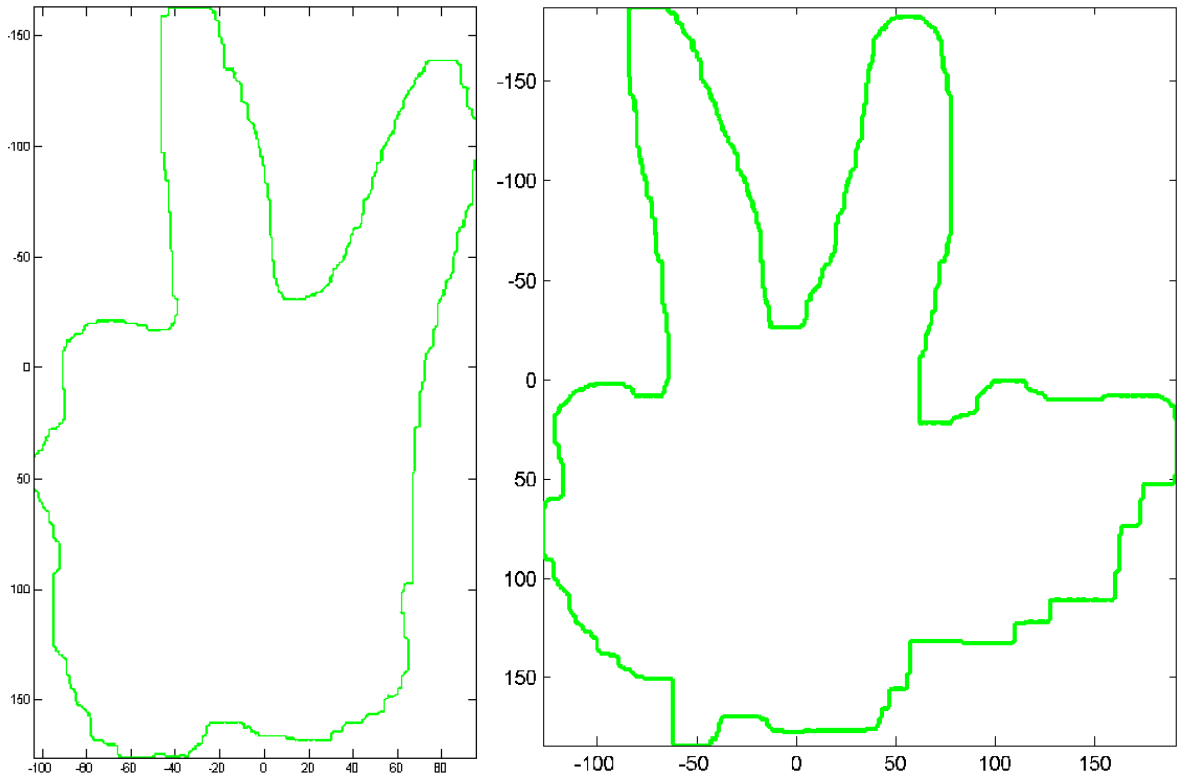


FIGURE A.6: Contour Centering where the axis represent pixels

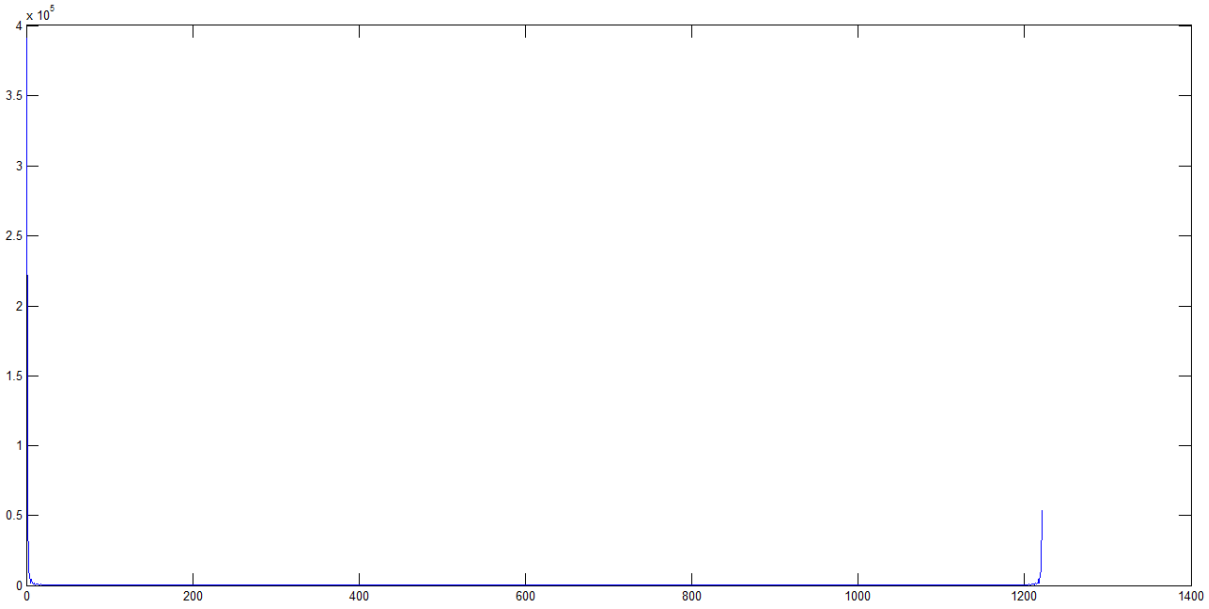


FIGURE A.7: Plot of FFT of X-axis

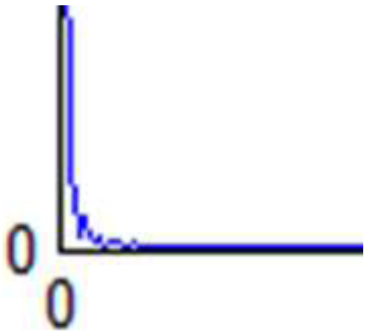


FIGURE A.8: Key Features of Descriptor

[98] proposed a novel 3D shape descriptor based on reflective symmetry. This refers to a measure of reflective symmetry for an arbitrary 3D model for all planes through the model's center of mass (regardless of the uniformity of their shape). Kazhdan shows how to obtain a voxel grid from arbitrary 3D shapes and, using Fourier methods, presents an algorithm that computes the symmetry descriptor in $O(N^4 logN)$ time for an $N \times N \times N$ voxel grid and computes a multiresolution approximation in $O(N^3 logN)$ time.

Princeton University has proposed several 3D model searching and indexing approaches. In [99], Funkhouser proposes a search repository of 3D surface models with queries describing geometric properties. He found that the main challenge in matching and indexing shapes is accounting for arbitrary rotations. Previous methods that require alignment into a common coordinate system (e.g., with principal axes) are not robust. He claims other methods that rely upon rotationally invariant shape descriptors are usually not very discriminating. Funkhouser proposed a novel rotation invariant shape-descriptor based on spherical harmonics. The main idea is to decompose a 3D model into a collection of functions defined on concentric spheres, and to use spherical harmonics to discard orientation information (phase) for each one. This yields a shape descriptor that is both orientation invariant and descriptive. The proposed method involves using skeleton model data, where each node connector (stick) is represented by a 3 by 3 matrix containing 3D orientation, as shown in Figure A.9. The 3 by 3 matrices are transpositions of three 2 by 2 matrices, each representing position, rotation and scale. The feature matrix will be populated with all 9 real numbers from each stick. When the classifier is run, each individual feature will be tested for accuracy of classification. It will be possible to narrow down only the most important features for each gesture. Table A.1 illustrates a four-class experiment, where a Naive Bayes classifier with 80 percent training and 20 percent testing achieved a total accuracy of 71.25 percent with 40 features. By applying forward sequential feature selection,

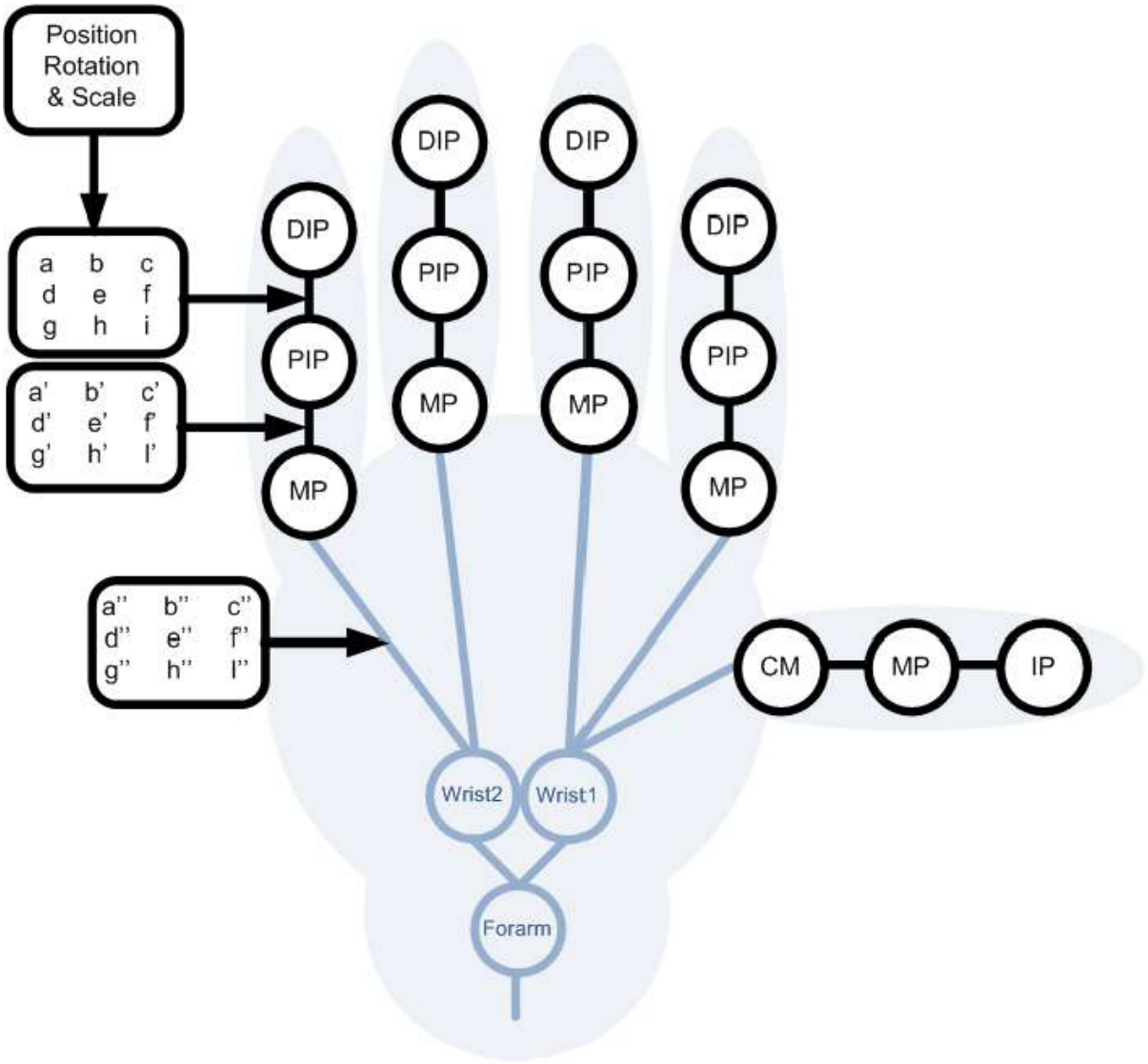


FIGURE A.9: Hand Skeleton Model with 3X3 Matrix Description

an accuracy of 69.75 percent was achievable using only one feature. In this example, adding any one of the other 39 features would not increase the accuracy. This problem stops the forward sequential feature selection algorithm. Though inputting all 9 values of every stick in the hand may seem computationally heavy at first, once prime features are selected, the recognition and classification process can become much quicker.

Time/Motion

In [100], Guangqi presents an approach for the capturing of motion by using the location feature. (Location of pixel is derived by segmentation by way of colour sampling). The descriptor

Accuracy=71.25%	
Accuracy of Class 9 Classification: 85%	
Accuracy of Class 10 Classification: 70%	Forward Sequential Feature Selection (% of error)
Accuracy of Class 11 Classification: 90%	Step 1, added column 14, criterion value 0.3025
Accuracy of Class 12 Classification: 40%	Final columns included:14
Elapsed time is 1.125224 seconds	

TABLE A.1: Forward Sequential Feature Selection

contains orientation features, which are made up of directional angles. It's worth mentioning that a simple HMM was used to accurately track hands drawing shapes.

Classification

Before classification can begin, a feature matrix needs to be constructed. Table A.2 shows the feature matrix format used, while Figure A.10 shows an invalid feature matrix format. The feature matrix has each sample with a label at the end (right side). Each gesture is a horizontal data set, with some type of feature information associated with it. The rightmost value is always the class number. The feature matrix is passed to a classifier, which has an implementation of a multivariate normal distribution with the probability distribution function of:

$$f_x(x) = \frac{1}{2\pi^{\frac{k}{2}}|\Sigma|^{\frac{1}{2}}} e^{\left(-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right)} \tag{A.2}$$

Where k is number of gestures (classes).

	Feature1	Feature 2	...	Feature m	Class
Sample 1	$X_{F1,S1}$	$X_{F2,S1}$	...	$X_{Fm,S1}$	Class
Sample 2	$X_{F1,S2}$	$X_{F2,S2}$	...	$X_{Fm,S2}$	Class
...	...	...	...	...	...
Sample n	$X_{F1,Sn}$	$X_{F2,Sn}$	...	$X_{Fm,Sn}$	Class k

TABLE A.2: Feature Matrix

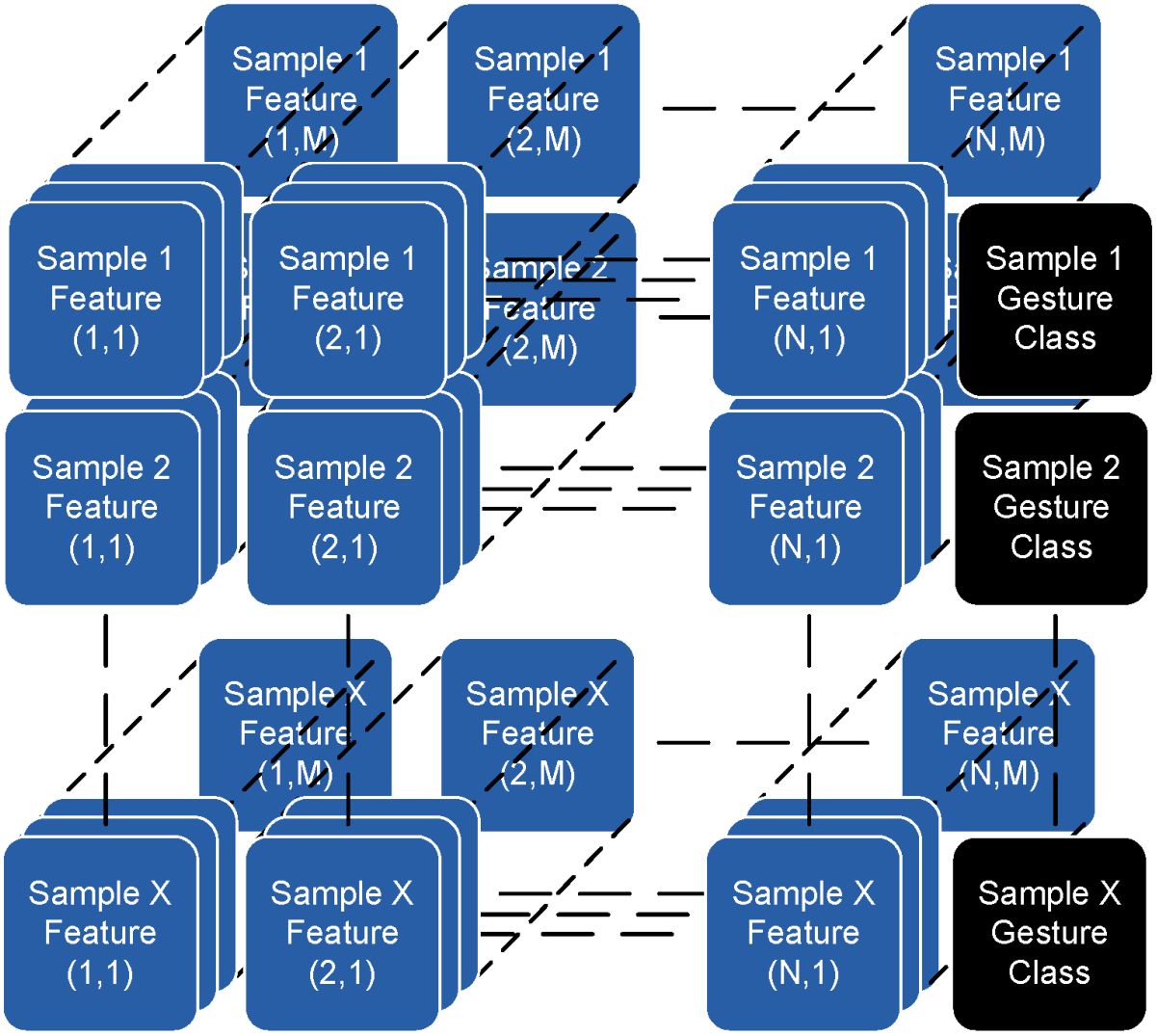


FIGURE A.10: Representation of Invalid Feature Matrix with Contour and Temporal Data (2.5D)

Descriptor features are real values based on Figure A.8. Mean values are calculated for the entire class set. Covariance must be calculated instead of just variance because this is a multivariate problem. In the experimentation results shown Table A.3, the feature matrix contains 200 samples, 100 samples per class. Results were tested using 2, 10, 20, 30, and 40 features. As yet, there is no know optimal number of features. Based on the limited number of samples in the table, one could extrapolate that having more features is better. The risk with providing too many features (original contour average is around 1600 features) is that the classifier will be over fitted, the accuracy will decrease, and processing time might increase.

This approach used a Naive Bayes classifier. Other classification methods will still be designed and tested during the methodology phase in the search for the most accurate results. One

# of Feature in Descriptor	Accuracy of Class 1	Accuracy of Class 2	Time Cost
2	73,00%	44,00%	0.113229sec
10	60,00%	84,00%	0.125898sec
20	86,00%	91,00%	0.118099sec
30	96,00%	96,00%	0.130884sec
40	98,00%	97,00%	0.125588sec

TABLE A.3: Two Class Naive Bayes Accuracy Comparison

major limitation to the Naive Bayes classifier, and many other classification methods, is that our feature matrix is 2D, which ultimately limits our descriptor. One potential solution to this limitation might be to combine multiple features into feature sets capable of representing the 3.5D data in a single descriptor. Even though the representation is in 2D, we would not lose the 3.5D data. This theory would still need to be tested for validity. Figure A.10 illustrates the complexity of a 2.5D feature matrix. This format is not compatible without downconverting the data into the format of Table A.2 Feature Matrix. "Sample 1 Feature(1,1)" to "Sample 1 Feature(1,M)" represents just the contour data, while temporal data in linear in this example.

Picking a Classifier

When selecting a classifier, exploratory data analysis is always a good idea. Currently, Naive Bayes, SVM and Bagging Decision Trees (type of bootstrap aggregation) have been tested as multi-class and multi-feature classifiers, though each approach has many sub-variations by way of kernel.

Testing Accuracy

Many approaches exist to test accuracy, including confusion matrices and rock curves. A confusion matrix is a visualization tool which displays the predicted values and actual values overtop each other. In Table A.4, a simple confusion matrix is shown, where the principal diagonal represents the correct prediction. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e. commonly mislabelling one as another). When a data set is unbalanced, the error rate of a classifier is not representative of the true performance of the classifier. The table allows for easy identification of biases towards classes.

		Predicted Value	
		Class 1	**Class 2**
Actual value	**Class 1**	True Positive	False Negative
	Class 2	False Positive	True Negative

TABLE A.4: Confusion Matrix Example

Test Gestures

Static gestures were selected to begin the classifier testing. Table A.5 shows twelve gesture classes that were tested over the following sections. 100 samples of each gesture are sampled into the database.

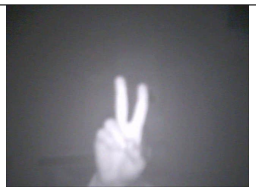
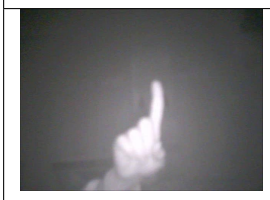
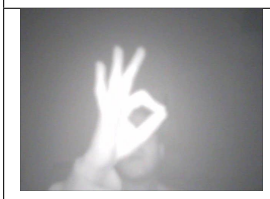
			
Gesture Class 1	Gesture Class 2	Gesture Class 3	Gesture Class 4
			
Gesture Class 5	Gesture Class 6	Gesture Class 7	Gesture Class 8
			
Gesture Class 9	Gesture Class 10	Gesture Class 11	Gesture Class 12

TABLE A.5: First 12 Gesture Classes of Test Database

Naive Bayes

Naive Bayes with Gaussian distribution has been tested with up to 12 gestures. All training classification has been in the near 100 percent range, as with all of the following approaches. Table A.6 shows the results of six two-class testing problems for comparative analysis. Accuracy ranges from 70% to 85% in approximately half a second total computation time.

Naive Bayes Results Two-Class Problem with Randomly Distributed 80% Training 20% Testing					
Class 1 vs 2	**Class 3 vs 4**	**Class 5 vs 6**	**Class 7 vs 8**	**Class 9 vs 10**	**Class 11 vs 12**
Accuracy = 82.5%	Accuracy = 80%	Accuracy = 80%	Accuracy = 70%	Accuracy = 85%	Accuracy = 72.5%
Accuracy of C 1 Classification: 80%	Accuracy of C 1 Classification: 60%	Accuracy of C 1 Classification: 70%	Accuracy of C 7 Classification: 55%	Accuracy of C 9 Classification: 75%	Accuracy of C 11 Classification: 90%
Accuracy of C 2 Classification: 85%	Accuracy of C 2 Classification: 100%	Accuracy of C 2 Classification: 90%	Accuracy of C 8 Classification: 85%	Accuracy of C 10 Classification: 95%	Accuracy of C 12 Classification: 55%
0.463599 seconds.	0.475739 seconds.	0.884039 seconds.	0.465314 seconds.	0.888766 seconds.	0.540778 seconds.
Confusion Matrix $\begin{bmatrix} 16 & 4 \\ 3 & 17 \end{bmatrix}$	Confusion Matrix $\begin{bmatrix} 12 & 8 \\ 0 & 20 \end{bmatrix}$	Confusion Matrix $\begin{bmatrix} 14 & 6 \\ 2 & 18 \end{bmatrix}$	Confusion Matrix $\begin{bmatrix} 11 & 9 \\ 3 & 17 \end{bmatrix}$	Confusion Matrix $\begin{bmatrix} 15 & 5 \\ 1 & 19 \end{bmatrix}$	Confusion Matrix $\begin{bmatrix} 18 & 2 \\ 9 & 11 \end{bmatrix}$

TABLE A.6: Naive Bayes 2 Class Results

Table A.7 uses the same classes but lists the results of four classes at the same time. Accuracy drops to between 57% and 71% with the increase of class numbers. Table A.8 shows the results of a twelve-class problem. Accuracy has dropped to 32%.

Feature Selection

Following Naive Bayes classification, a forward sequential feature selection technique was run on the feature matrix. [101] outlines such a technique. The bottom of Table A.7 and Table A.8 shows the results of the approach. What is interesting is that in some cases, specifically in Table A.7, less than two features out of the original forty are capable of describing the class well enough to achieve equal accuracy to using all forty features. In the case of Table A.8, five features are collected before the accuracy cannot be increased any further. Feature selection will be very useful in the future, once feature matrices begin to contain hundreds of features.

Naive Bayes Results Four-Class Problem with Randomly Distributed 80% Training 20% Testing		
Class 1 vs 2 vs 3 vs 4	**Class 5 vs 6 vs 7 vs 8**	**Class 9 vs 10 vs 11 vs 12**
Accuracy = 57.5%	Accuracy = 61.25%	Accuracy = 71.25%
Accuracy of Class 1 Classification: 15%	Accuracy of Class 5 Classification: 45%	Accuracy of Class 9 Classification: 85%
Accuracy of Class 2 Classification: 60%	Accuracy of Class 6 Classification: 70%	Accuracy of Class 10 Classification: 70%
Accuracy of Class 3 Classification: 75%	Accuracy of Class 7 Classification: 60%	Accuracy of Class 11 Classification: 90%
Accuracy of Class 4 Classification: 80%	Accuracy of Class 8 Classification: 70%	Accuracy of Class 12 Classification: 40%
1.153229 seconds.	1.572986 seconds.	1.125224 seconds.
Confusion Matrix $\begin{bmatrix} 3 & 4 & 8 & 5 \\ 1 & 12 & 4 & 3 \\ 0 & 0 & 15 & 5 \\ 1 & 3 & 0 & 16 \end{bmatrix}$	Confusion Matrix $\begin{bmatrix} 9 & 3 & 4 & 4 \\ 2 & 14 & 1 & 3 \\ 0 & 1 & 12 & 7 \\ 5 & 1 & 0 & 14 \end{bmatrix}$	Confusion Matrix $\begin{bmatrix} 17 & 0 & 2 & 1 \\ 0 & 14 & 6 & 0 \\ 0 & 0 & 18 & 2 \\ 2 & 0 & 10 & 8 \end{bmatrix}$
Forward Sequential Feature Selection:	Forward Sequential Feature Selection:	
Step 1, added feature 32, accuracy: 53.5%	Step 1, added feature 6, accuracy: 53.75%	Forward Sequential Feature Selection:
Step 2, added feature 5, accuracy: value 57.75%	Step 2, added column 22, accuracy: 56.25%	Step 1, added feature 14, accuracy: 69.75%
Final features included:5 & 32	Final features included: 6 & 22	Final feature included: 14

TABLE A.7: Naive Bayes 4 Class Results

SVM

To begin testing SVM, a linear kernel was used, though research suggests trying the RBF kernel first. In [102], Keerthi suggests that if an RBF kernel is used with model selection, there is no need to test a linear kernel. RBF still needs to be tested. A good SVM tutorial is available at [103]. Table A.9 shows the results of SVM two-class results. Accuracy tends to be higher than with the Naive Bayes approach. Computation time is also comparable, though the Class 1 & Class 2 experiment contradicts accuracy and speed observations.

Naive Bayes Results **Twelve-Class Problem with Randomly Distributed 80% Training 20% Testing**	
	ConfusionMatrix 0 4 0 4 0 2 0 4 2 0 4 0 0 9 0 2 0 0 0 3 5 0 1 0 0 0 2 7 0 6 0 1 4 0 0 0 0 4 0 11 0 0 0 4 0 0 1 0 0 4 0 4 0 1 0 3 4 4 0 0
Accuracy = 32.5%	0 0 0 2 0 7 0 6 3 1 1 0 0 1 0 5 0 7 0 1 4 0 2 0 0 1 0 0 0 6 0 11 0 2 0 0
Accuracy of C 1 Classification: 0% Accuracy of C 2 Classification: 45% Accuracy of C 3 Classification: 10% Accuracy of C 4 Classification: 55%	0 1 0 1 0 1 0 0 17 0 0 0 0 3 0 3 0 0 0 0 0 14 0 0 0 2 0 2 0 6 0 3 0 0 7 0 0 6 0 1 0 7 0 3 2 0 1 0
Accuracy of C 5 Classification: 0% Accuracy of C 6 Classification: 35% Accuracy of C 7 Classification: 0% Accuracy of C 8 Classification: 55% Accuracy of C 9 Classification: 85% Accuracy of C 10 Classification: 70% Accuracy of C 11 Classification: 35% Accuracy of C 12 Classification: 0% C is Class 7.186955 seconds.	Forward Sequential Feature Selection: Step 1, added feature 2, accuracy: 29.8333% Step 2, added feature 40, accuracy: 35.3333% Step 3, added feature 1, accuracy: 36.1667% Step 4, added feature 15, accuracy: 36.5% Step 5, added feature 3, accuracy: 36.5833% Final features included: 1 2 3 15 40

TABLE A.8: Naive Bayes Twelve-Class Results

Table A.10 shows the results of three SVM four-class problems. Results are very comparable to the results of Naive Bayes approach, but with exponentially longer computation time. Table A.11 shows the results of a twelve-class problem, with accuracy improving to 50

Bagging Decision Trees (Bootstrap Aggregation)

The Naive Bayes and SVM results hinted that the classifier had over-fitted the classes. The approach of bagging decision trees, or more commonly known as bootstrap aggregations, was

Support Vector Machine Classification Results Two-Class Problem with Evenly Distributed 80% Training 20% Testing					
Class 1 vs 2	**Class 3 vs 4**	**Class 5 vs 6**	**Class 7 vs 8**	**Class 9 vs 10**	**Class 11 vs 12**
Accuracy = 55% (22/40)	Accuracy = 90% (36/40)	Accuracy = 90% (36/40)	Accuracy = 70% (28/40)	Accuracy = 97.5% (39/40)	Accuracy = 77.5% (31/40)
Accuracy of C 1 Classification: 60%	Accuracy of C 1 Classification: 80%	Accuracy of C 1 Classification: 85%	Accuracy of C 7 Classification: 60%	Accuracy of C 9 Classification: 100%	Accuracy of C 11 Classification: 90%
Accuracy of C 2 Classification: 50%	Accuracy of C 2 Classification: 100%	Accuracy of C 2 Classification: 95%	Accuracy of C 8 Classification: 80%	Accuracy of C 10 Classification: 95%	Accuracy of C 12 Classification: 65%
10.918108 seconds.	0.359613 seconds.	1.212346 seconds.	0.799946 seconds.	0.037085 seconds.	1.550604 seconds.

TABLE A.9: Support Vector Machine Two-Class Results

also tested, because the approach reduces variance and helps to avoid over fitting. Table A.12 shows the results of three four-class problems. Results are higher than the previous classifiers and even more noticeable in Table A.13.

Support Vector Machine Classification Results Four-Class Problem with Randomly Distributed 80% Training 20% Testing		
Class 1 vs 2 vs 3 vs 4	**Class 5 vs 6 vs 7 vs 8**	**Class 9 vs 10 vs 11 vs 12**
Accuracy = 50% (40/80)	Accuracy = 60% (48/80)	Accuracy = 77.5% (62/80)
Accuracy of Class 1 Classification: 55%	Accuracy of Class 5 Classification: 75%	Accuracy of Class 9 Classification: 90%
Accuracy of Class 2 Classification: 35%	Accuracy of Class 6 Classification: 60%	Accuracy of Class 10 Classification: 90%
Accuracy of Class 3 Classification: 50%	Accuracy of Class 7 Classification: 35%	Accuracy of Class 11 Classification: 90%
Accuracy of Class 4 Classification: 60%	Accuracy of Class 8 Classification: 70%	Accuracy of Class 12 Classification: 40%
26.809825 seconds.	18.000260 seconds.	4.245733 seconds.

TABLE A.10: Support Vector Machine Four-Class Results

Support Vector Machine Classification Results Twelve-Class Problem with Randomly Distributed 80% Training 20%
Accuracy = 50% (120/240) Accuracy of Class 1 Classification: 30% Accuracy of Class 2 Classification: 25% Accuracy of Class 3 Classification: 50% Accuracy of Class 4 Classification: 50% Accuracy of Class 5 Classification: 50% Accuracy of Class 6 Classification: 40% Accuracy of Class 7 Classification: 25% Accuracy of Class 8 Classification: 60% Accuracy of Class 9 Classification: 70% Accuracy of Class 10 Classification: 85% Accuracy of Class 11 Classification: 85% Accuracy of Class 12 Classification: 30% Elapsed time is 283.466870 seconds.

TABLE A.11: Support Vector Machine 12-Class Results

Support Vector Machine Classification Results Four-Class Problem with Randomly Distributed 80% Training 20% Testing		
Class 1 vs 2 vs 3 vs 4	**Class 5 vs 6 vs 7 vs 8**	**Class 9 vs 10 vs 11 vs 12**
Accuracy = 83.75%	Accuracy = 82.1875%	Accuracy = 88.1250%
Confusion Matrix $\begin{bmatrix} 19 & 0 & 1 & 0 \\ 0 & 19 & 0 & 1 \\ 5 & 0 & 15 & 0 \\ 2 & 0 & 0 & 18 \end{bmatrix}$	Confusion Matrix $\begin{bmatrix} 19 & 0 & 1 & 0 \\ 0 & 18 & 2 & 0 \\ 0 & 0 & 19 & 1 \\ 3 & 4 & 0 & 13 \end{bmatrix}$	Confusion Matrix $\begin{bmatrix} 20 & 0 & 0 & 0 \\ 0 & 18 & 0 & 2 \\ 0 & 0 & 18 & 2 \\ 3 & 0 & 3 & 148 \end{bmatrix}$

TABLE A.12: Bagging Decision Trees Four-Class Results

Bagging Decision Trees Results
Twelve-Class Problem with Randomly Distributed 80% Training 20% Testing

Accuracy = 76.0417%

ConfusionMatrix

2	11	0	0	0	0	3	3	0	0	1	0
3	0	17	0	0	0	0	0	0	0	0	0
0	0	3	15	0	0	0	1	0	0	1	0
0	0	1	0	15	0	1	0	1	2	0	0
0	0	0	1	0	15	0	3	0	0	0	1
2	0	5	0	0	2	11	0	0	0	0	0
0	1	0	2	0	2	0	15	0	0	0	0
0	0	1	0	1	0	2	0	15	0	0	1
0	0	0	1	2	0	0	0	0	15	0	2
0	0	0	1	0	1	3	1	0	0	13	1
1	1	0	0	0	0	0	1	0	0	1	16

TABLE A.13: Bagging Decision Trees Twelve-Class Results

Appendix B

Alternative Inputs

B.1 Pointing Display

The researcher has envisioned a pointing display for use on store fronts that makes use of gesture recognition software, allowing customers to point at items on a screen display outside of a store in order to add items to their 'virtual cart', similar to an online store like Amazon.

However, this interface is difficult to configure because it has always been difficult for a computer to recognize an individual pointing finger due to the noise of angular data – as a result of this noise, the recognition system often sees a pointing finger as a 'spike' that distorts its recognition of a user's hand altogether. In practice, many researchers have found difficulties in achieving real-time, accurate results of a pointing finger using their gesture recognition system. Kim [104] [105] describes a 'pointing gesture recognition method' that fails because it only uses basic static gestures, always treating one index finger extended up in air as a pointing gesture. This conclusion is premature and not representative of how pointing is done in real life: in this dissertation, the researcher has determined that people do not commonly point with their limb in tandem with their finger – the 'pointing' gesture is often performed with just the finger, and does not require synchronization with the entire arm, which Kim has incorrectly assumed.

Figure B.1 shows a proposed solution for effectively creating such a system.

B.2 Mouse

When it comes to mouse interaction, an algorithm that presumes a single user in the scene would be very difficult to use in the majority of use cases. Figure B.2 demonstrates how to lock a mouse to one user. The mouse cursor control resembles the experience of a classic joystick. Using a joystick interface was a learned process: after several live demonstrations of the technology, it was determined that skelatonization algorithms (just like many image processing techniques) have many "hiccups", e.g., the locations of the hand is usually not "smooth" and can jump several camera pixels in any direction, causing a mouse with "relative tracking" to jitter and jump. By using the distance of the hands location to a fixed point (e.g., shoulder height and semi-extended arm), the speed to move the mouse in a direction is determined and the mouse cursors movement can be smoothed.

B.3 Anchored 3D Buttons

In a similar approach to how anchoring has been done in the previous chapters, Figure B.3 is a concept drawing of a 3D button with proposed anchoring approach. 3D shapes can be treated like buttons and controls when anchored to the body, and are able to move with the user while still being located in relatively the same area for the user. These button elements can be seen when combined with augmented reality, but even when not visible these buttons can become second nature for a user.

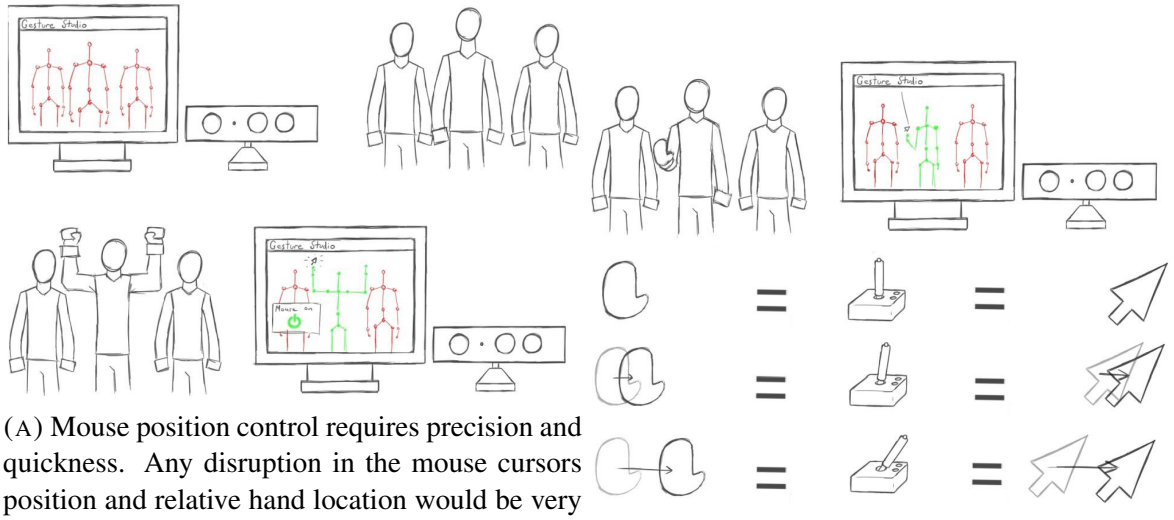
(A) A storefront window has been freshly arranged for display. The storefront has been enabled and trained by Gesture Studio. The storefront is display enabled so that additional information can be presented to customers.

(B) Customer stands in a predefined location and paint at the object they want to purchase or now more information about.

(C) After the pointing gesture relative to the item pointed at, more information is provided.

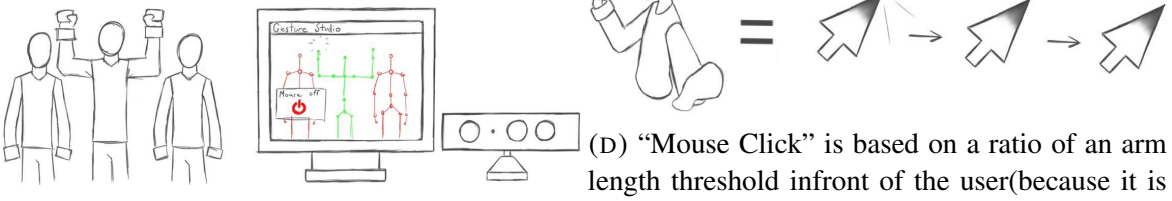
(D) A customer is now able to complete the purchase by scanning a code on the screen.

FIGURE B.1: Gesture Studio enabled storefront window. Allows users/customers to point at objects through a window and be provided with a interface.



(A) Mouse position control requires precision and quickness. Any disruption in the mouse cursors position and relative hand location would be very disorienting. Hard locking the cursor to one user at a time is presently the most accurate way to start a hand-to-mouse session.

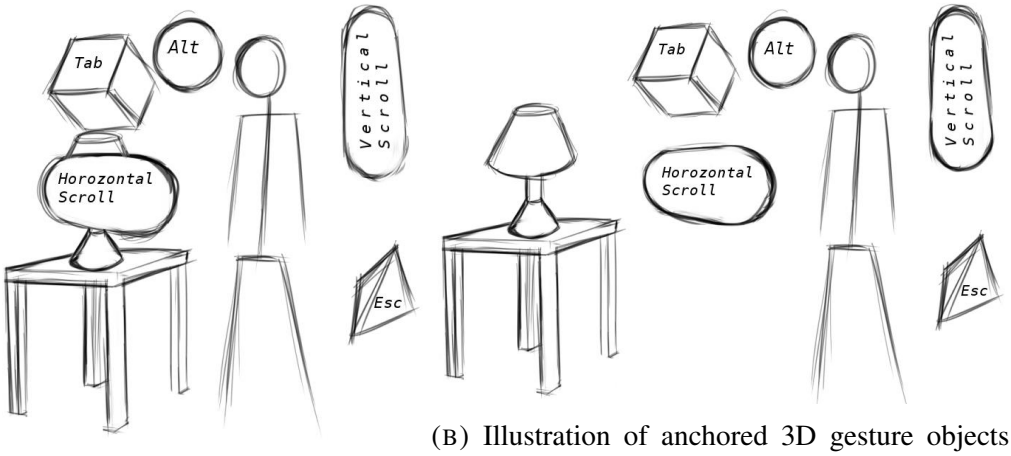
(B) A mouse movement hand is operated similar to two-axis joystick, whereby the distance from center effects the mouse movement exponentially



(C) Manual mouse session deactivation, activated by repeating the initialization gesture or by stepping out of camera view

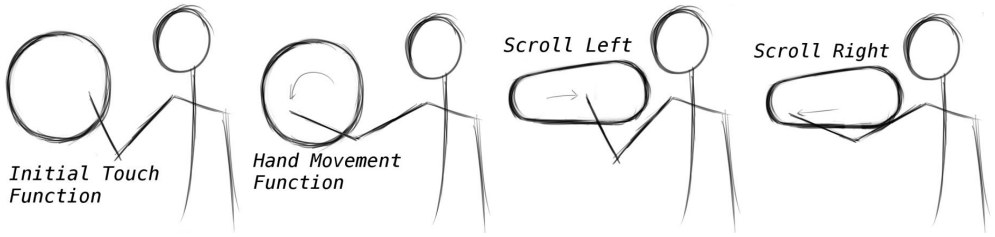
(D) "Mouse Click" is based on a ratio of an arm length threshold infront of the user(because it is used more often.) While, "Mouse Hold" is controlled by a similar threshold but to the side of the user.

FIGURE B.2: Gesture Studio Lite's Mouse Emulation Controls



(A) Visualization of anchored 3D gesture objects surround a user. These can be arranged around a user in various shapes and sizes. These objects can be used as triggers.

(B) Illustration of anchored 3D gesture objects moving with the user as the user explores the environment. This allows virtual tools to be seamlessly adapted to users moving through tasks and knowing where to expect triggers with their varying poses.



(C) Simple touch and hold with variable threshold

(D) Using 3D object as radio knob style control.

(E) Using a 3D objects to scroll to the left.

(F) Using a 3D objects to scroll to the right.

FIGURE B.3: Anchored 3D gesture objects

Appendix C

Gesture Studio

This Appendix details all of the features of the Gesture Studio software, which was initially coded in C#, including instructions for use and screen captures of the Gesture Studio site and portal, which are primarily used by researchers and academics.

C.1 Recording

C.1.1 Start Recording

To start recording, users can either press the 

button or use the voice command “start recording”. The button icon should then change to a blinking  icon to indicate Gesture Studio is recording.

C.1.2 Provide Gesture

As the recording starts right away, the user should be ready to perform the gesture before starting the recording. However, if necessary, the recording can be trimmed later on to remove unnecessary portions. Editing will be further discussed in the Optimization section.

C.1.3 Saving the Gesture

After the gesture has been recorded, the user can click the recording button to stop the system. If the gesture has been successfully recorded, a name will be randomly generated, unless the examiner has already given the recording a name using the 'Gesture Name' textbox. The gesture should also appear in the panel on the right under the gesture name.

Following successful completion of recording, the user can rename the recording, giving it a more descriptive name. There is also a feature that enables the user to remove certain "bad" frames, e.g., where either nothing is happening or a frame is not part of the gesture.

When training a computer, Gesture Studio creates a classifier file, which the classifier can then read and compare in real time to match gestures. Training requires at least 2 different gestures (gestures of different names) before it can train. To train, the user simply presses the 

button.

C.1.4 Classification

After the system is successfully trained, the examiner can run the classifier by pressing the 

button, which will start comparing gestures in real time.

(A) (B) (C) (D) (E)

FIGURE C.1: Gesture Studio Control Buttons

C.1.5 Voice Commands

Gesture Studio has a variety of voice commands which can be used to help perform different actions while standing far away from the computer. Gesture Studio is most efficient and least

time consuming when recording and editing is done by two people: one person populating the database through pantomiming and the other one running Gesture Studio.

If voice commands are not required, the user may turn them off by pressing the  button. The additional “lock” option removes the option of turning the voice commands on again through the voice command “voice on”. Each action may be recorded using a variety of the following “command variations”:

Action	Command Variations
Start Recording	“Start Recording” “Start” “Record”
Stop Recording	“Stop Recording” “Stop”
Default Tracking Mode (Standing)	“Default Mode” “Default” “Standing”
Seated Tracking Mode (Sitting)	“Seated Mode” “Seated”
Custom Tracking Mode	“Custom Mode” “Custom”
Toggle Save RGB Image	“Toggle RGB” “Toggle Color” “Toggle Color Image”
Toggle Save Depth Image	“Toggle Depth” “Toggle IR”
Turn on voice commands*	“Voice on” “Speech on” “Voice Recognition on” “Yes Voice” “Yes Speech” “Yes Voice Recognition” “Yes Speech Recognition”
Turn off voice commands*	“Voice off” “Speech off” “Voice Recognition off” “No Voice” “No Speech” “No Voice Recognition” “No Speech Recognition”

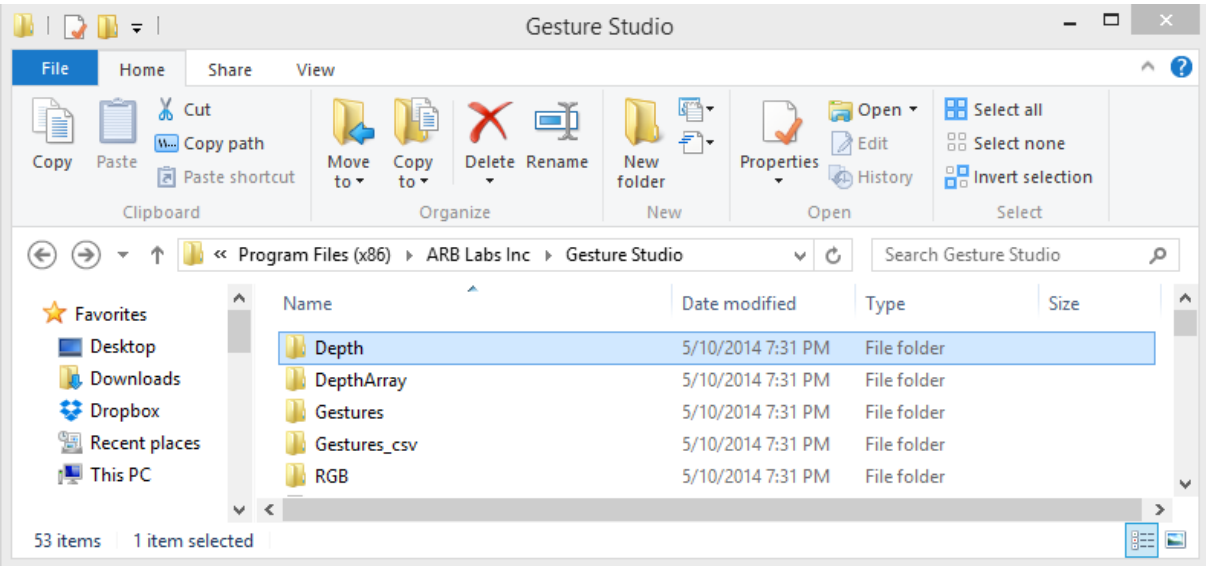


FIGURE C.2: Gesture Studio Folder structure of saved data

*Does not work if voice command lock checkbox is checked.

C.1.6 File and Directory Structure

Once created, the ability to move the gesture files efficiently is critically important, as uploading gestures to the cloud database becomes "transmissively costly." On the other hand, the availability of files to academics, developers, and enthusiasts is important for growing the database and creating a gesture benchmark. Therefore, Gesture Studio is designed to allow saving several gesture formats. The following figures show and describe the structure and use of the local client gesture database.

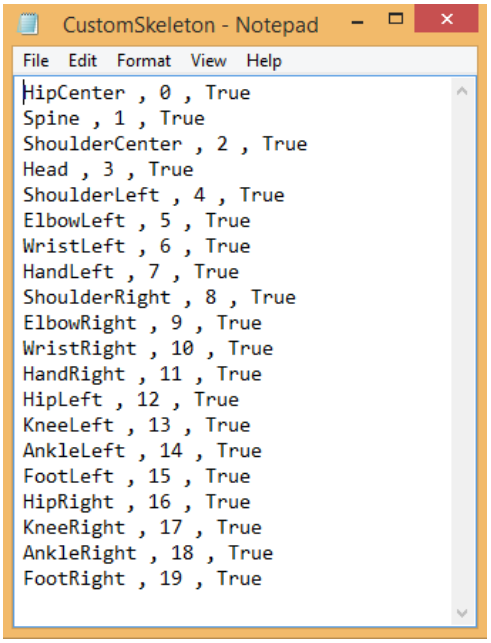


FIGURE C.3: Gesture Studio Principle Joint Index for Custom Skeleton

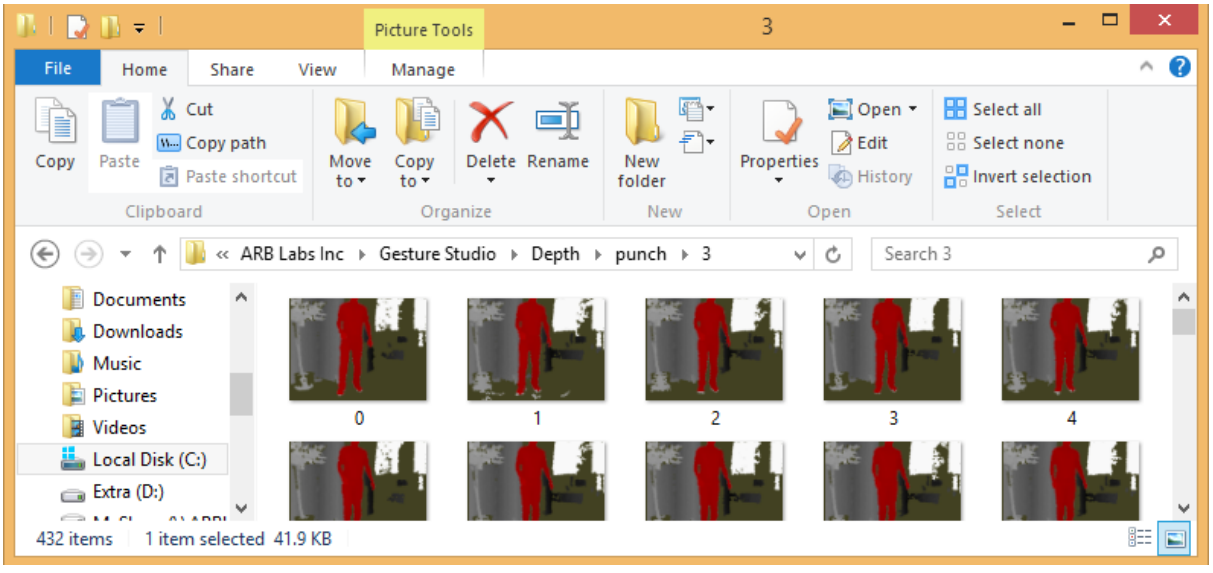


FIGURE C.4: Gesture Studio Depth folder is for saved images from the depth stream, pre-skeletonization.

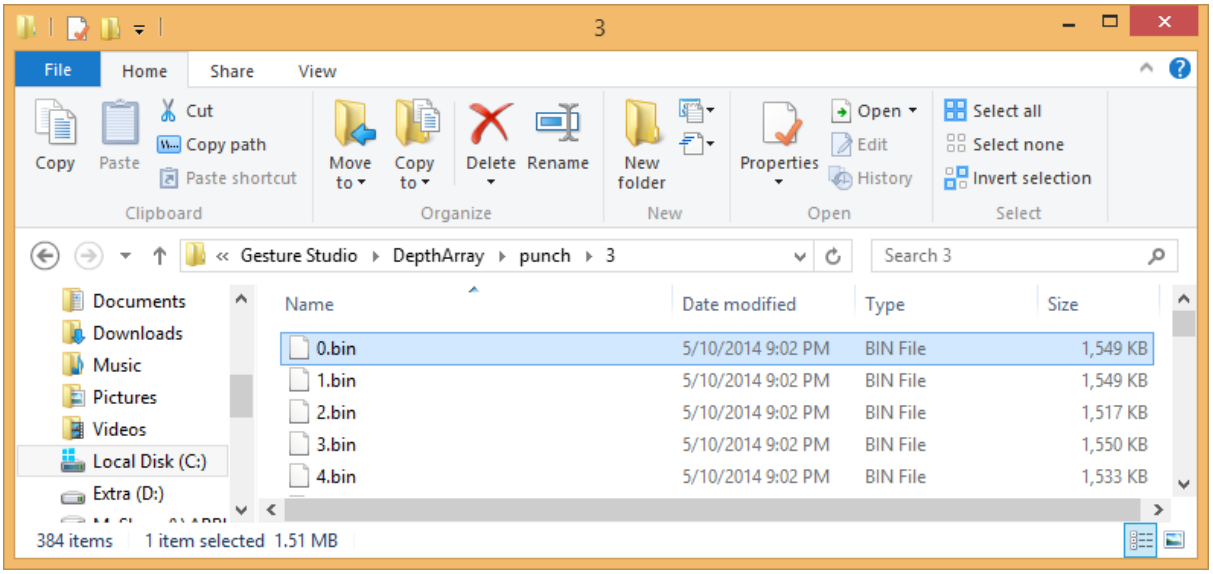


FIGURE C.5: Gesture Studio 'DepthArray' folder content

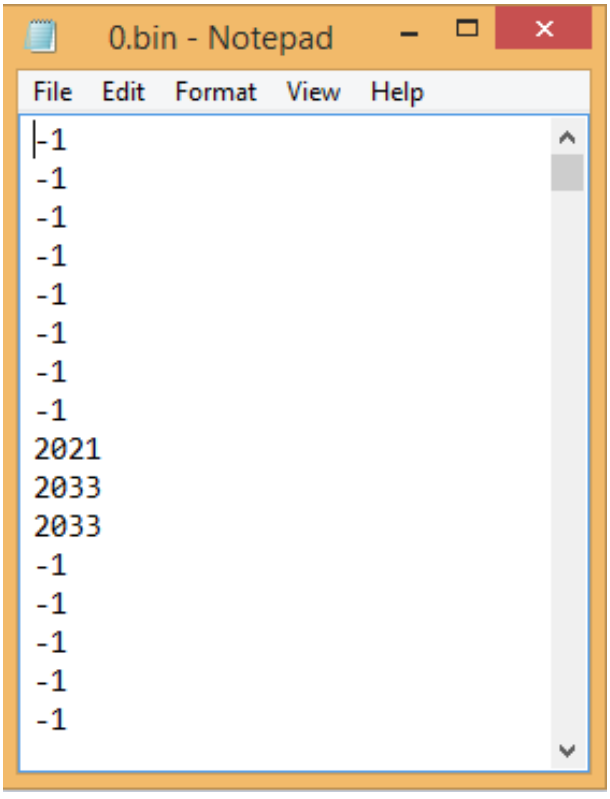


FIGURE C.6: Each 'DepthArray' file holds the raw numerical values of one frame of depth stream

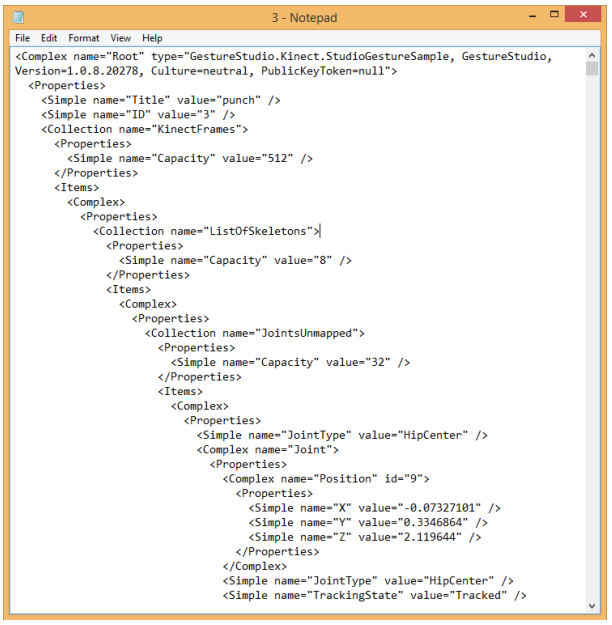


FIGURE C.7: Complex Gesture Default Data file; this contains one full gesture with motion Gesture Studio file. This approach is used for speed and compression of gesture data

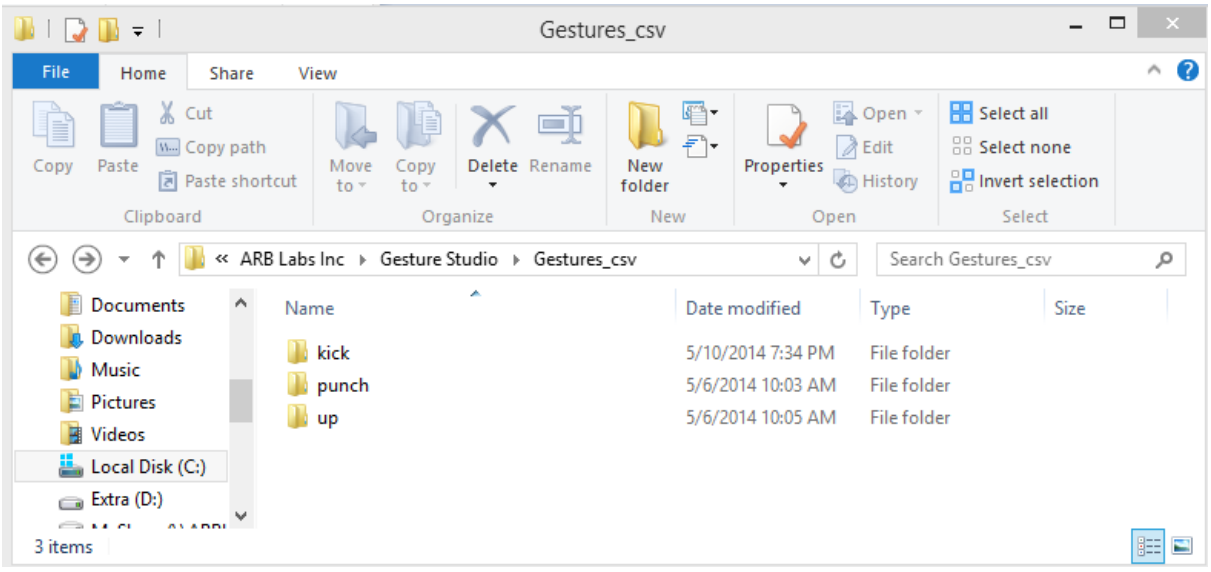


FIGURE C.8: Gesture Studio Gesture Comma Separated Values Folder Content

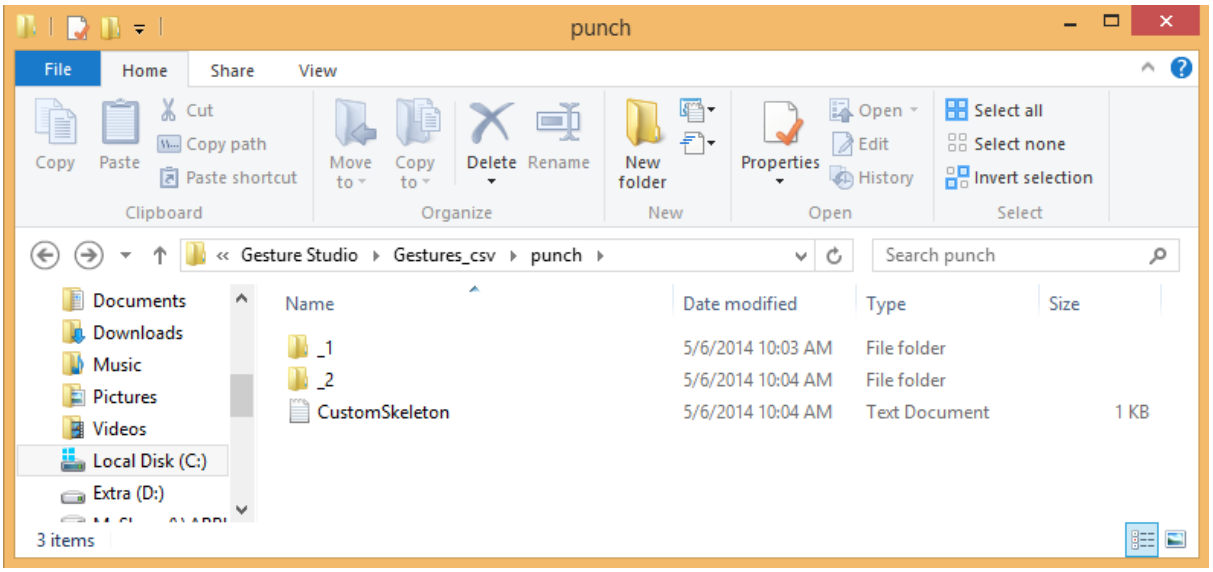


FIGURE C.9: Gesture Studio Gesture Comma Separated Sample Separated Folder

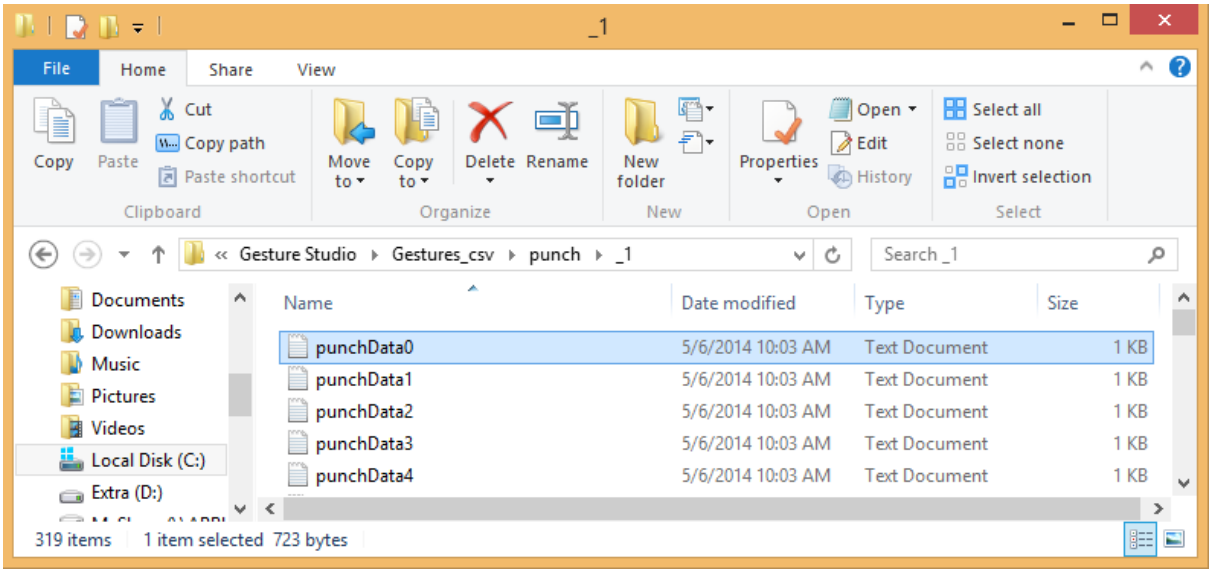


FIGURE C.10: Gesture Studio Gesture Comma Separated Values, each frame is a seperate file

C.2 Gesture Studio Lite

Gesture Studio Lite is my Graphical User Interface (GUI) for non-developers to quickly use the classifier they have trained in Gesture Studio with existing keyboard and mouse buttons. Figure C.13 is a screen capture of Gesture Studio Lite, demonstrating how gestures have been assigned to keyboard inputs.

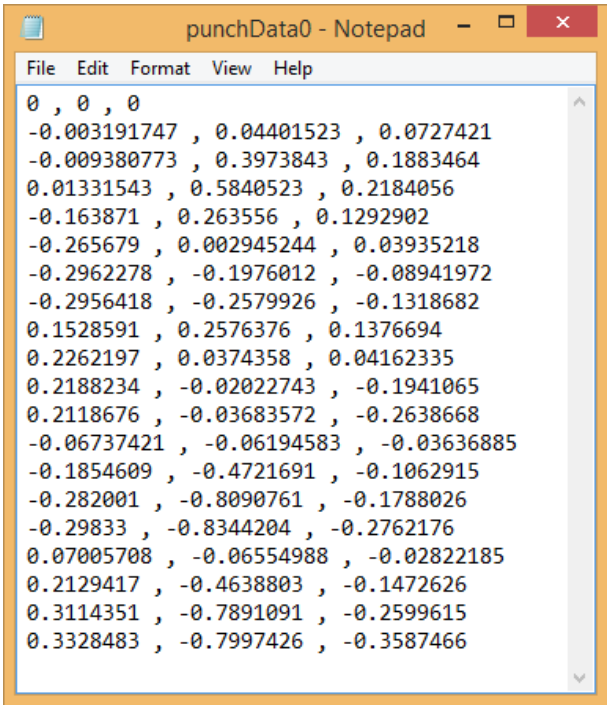


FIGURE C.11: Gesture Studio Gesture Comma Separated Values, the file is a single instance

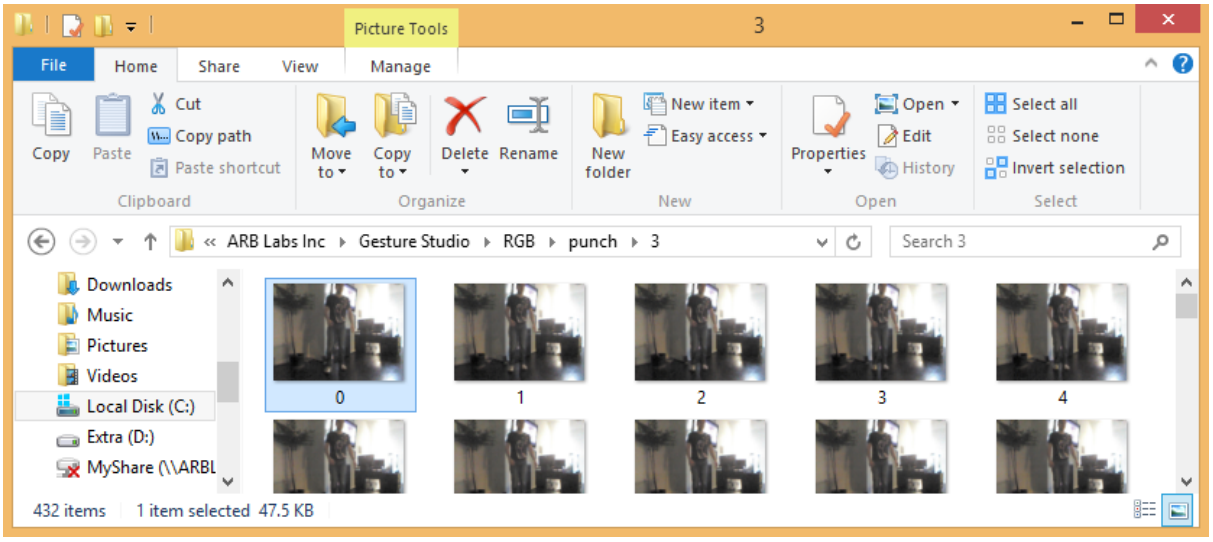


FIGURE C.12: Gesture Studio Gesture RGB Folder Content separated by folders for each full sample

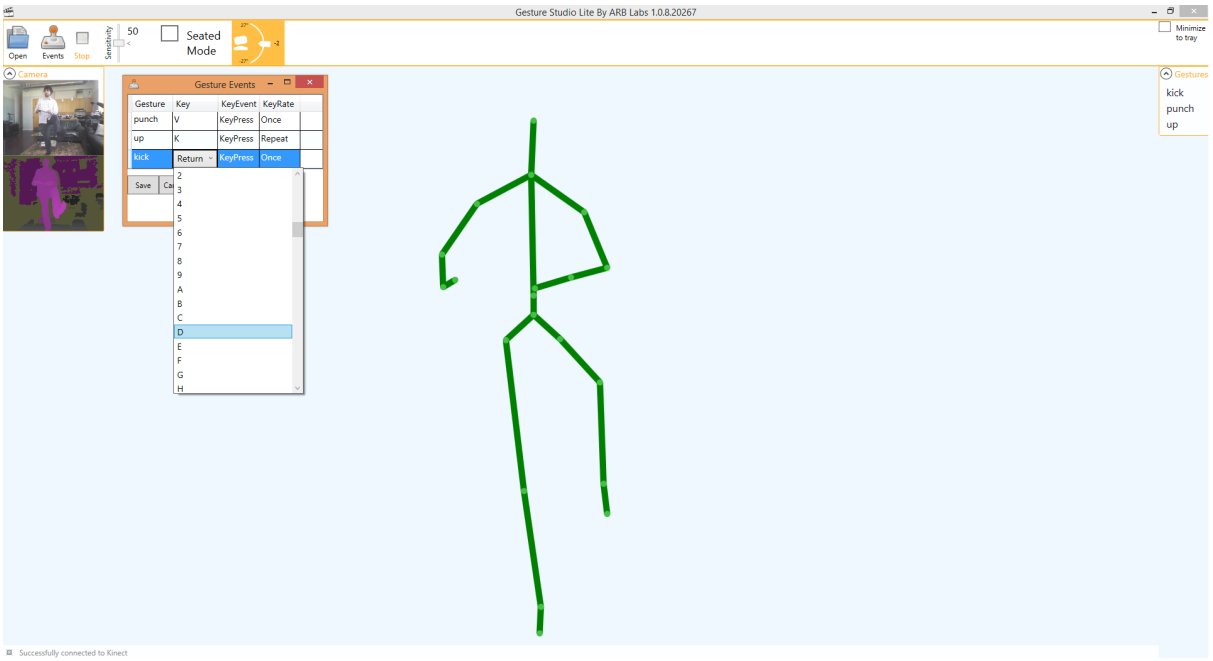


FIGURE C.13: Gesture Studio Lite is a tool that allows gestures to be connected to keyboard and mouse buttons

C.3 Gesture Studio Qt

Gesture Studio Qt is a cross platform version of Gesture Studio built on the OpenNI and NiTE2 drivers from PrimeSense. Gesture Studio Qt is compiled for Windows, Linux, and OSX. Because the Kinect and OpenNI drivers do not work well together, it was decided to keep the versions separate. The purpose of Gesture Studio Qt was to make the platform more open to developers. Figure C.14 is a screen capture of the Windows build of Gesture Studio Qt from October 2013.

C.4 Gesture Studio Visualizer

The proposed approaches are not ideal for visualizing why the PJVA recognizer chose to identify a source as a particular gesture. However, an algorithm that can determine the differences between the two samples of the same gesture would be a useful database creating tool. Therefore, experiments were conducted to use a nearest neighbour approach based on the work by

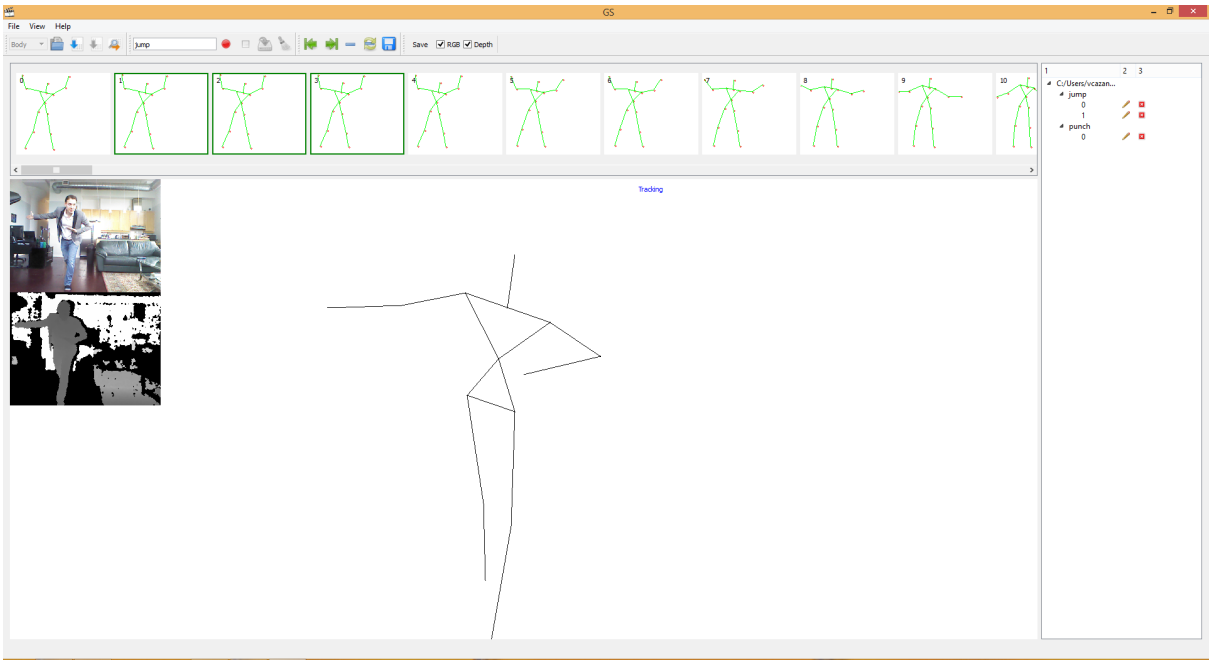


FIGURE C.14: Gesture Studio Qt and PrimeSense OpenNI Nite2 Skeletonization was developed for Windows, OSX and Linux.

Gonsales [106], and Khan [107] (Note, Gonsales' experiments are based off of the original dataset created and outlined in this dissertation). Kyan shows how a spherical SOM (SSOM) can be applied to solve gesture motion classification by learning the path of a gesture. In Gesture Studio spherical, SOM is an OpenGL visualizer paired with OpenCL SOM code. Figure C.15 includes various views of a SOM based on my PJVA-20 dataset.

The algorithm is able to find similar entries, arranging them closer together on the surface over time as it acquires more testable information. SSOM was thus used to examine why some gestures are alike and can indicate typical actions, and demonstrate how one gesture can trigger another — effectively representing how a computer can recognize gestures reliably as a sequence over time.

This technique has been used among researchers in this field. Another study also used self-organizing map in gesture recognition applications, with excellent result. A study by Oshita and Matsunaga [11] utilized Nintendo's Wii remote to learn two different gestures moving through time, implementing a Self-Organizing Map (SOM) to divide the sample data into phases and a SVM to learn the transition conditions between nodes. The results they reported for the supervised system scored an accuracy of 100% for class one and 84% for class two. The unsupervised system scored an accuracy of 98% for class one and 80% for class two.

C.5 Gesture Studio Cloud

Gesture Studio Cloud allows for integration of a large, ever-expanding database of gestures, without having to distribute the classifier algorithms to each client. Gestures are uploaded in the following structure:

<gesture name, id of uploader, hash of sample>

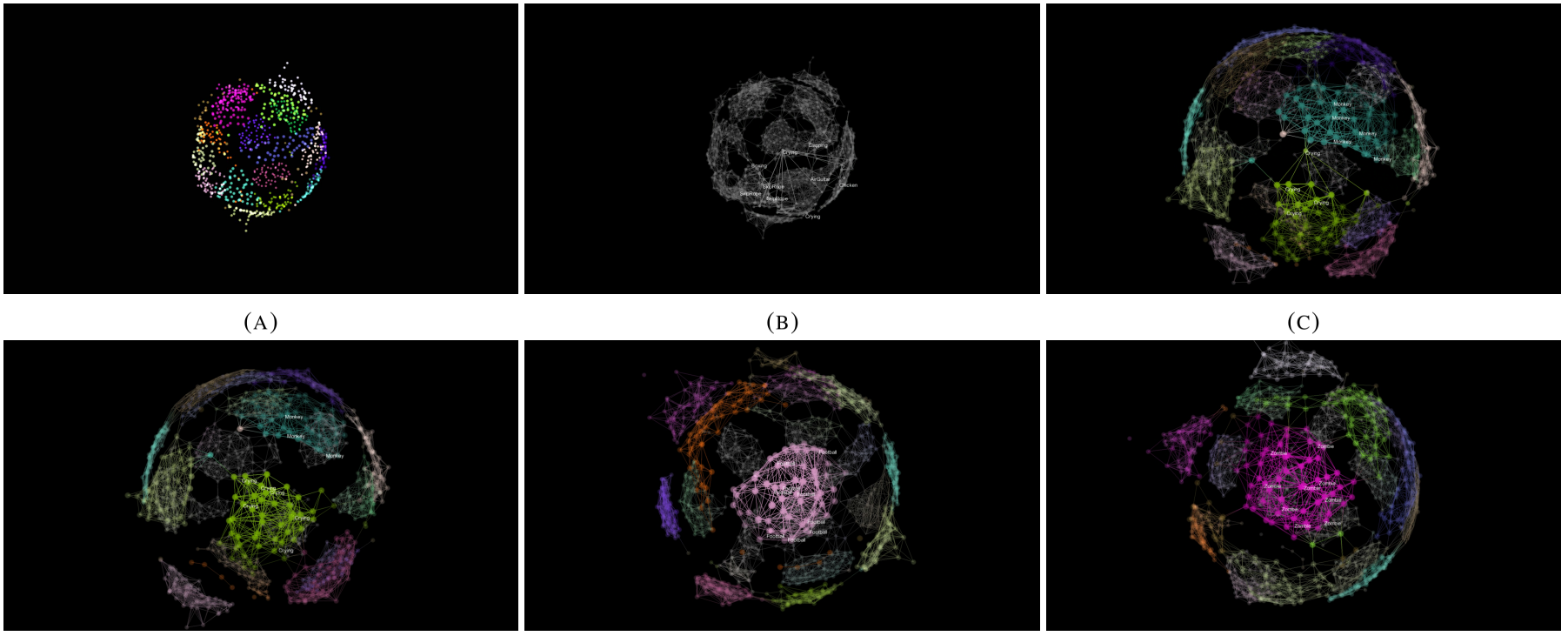
When training a classifier, a user can customize which samples to test for, e.g., all "Boxing" samples uploaded by user or all "Boxing" samples uploaded by everyone. A user can then add authentication and restriction of usage. Figure C.19 and Figure C.18 (see Chapter 4) show the high-level cloud topology and the high-level client topology, respectively. The Cloud is hosted on a Microsoft Azure server (see Figure C.17) running an AMD Opteron Processor 4171HE 2.09GHz single thread with 1.75GB of RAM running Windows Server 2008 R2 Datacenter. The VM is operated by a C# application dedicated to listening for training samples and returning classified binaries, as shown in Figure C.16.

Gesture Studio was developed on the Microsoft Azure Cloud platform. I (representing the ARB Labs Inc.) was a finalist in a Best Kinect Business Idea contest in 2013. Microsoft awarded ARB Labs Inc. with Azure credit. Appendix D shows the cloud learning of the gestures for the game Minecraft.

Advantages of using the cloud to train gestures vs. a plain PC:

- Sharing sample data between users
- Building an interconnected global database that is potentially capable of understanding the typical gestures performed in certain regions based on body language
- Creating an experimental database that allows for complex machine learning testing and training.
- Building a searchable and selectable database of gestures customized for each project requirement

- Allowing the local users to store only a very small recognizer file (about a hundred kilobytes) regardless of number of samples for each class
- Serving as an initial step for building an all-encompassing recognizer, similar to Google voice recognition.



(A) (B) (C)

(D) (E) (F)

FIGURE C.15: Gesture Studio Visualizer based on the 2012 Gesture SSOM proposed by Matthew Kyan

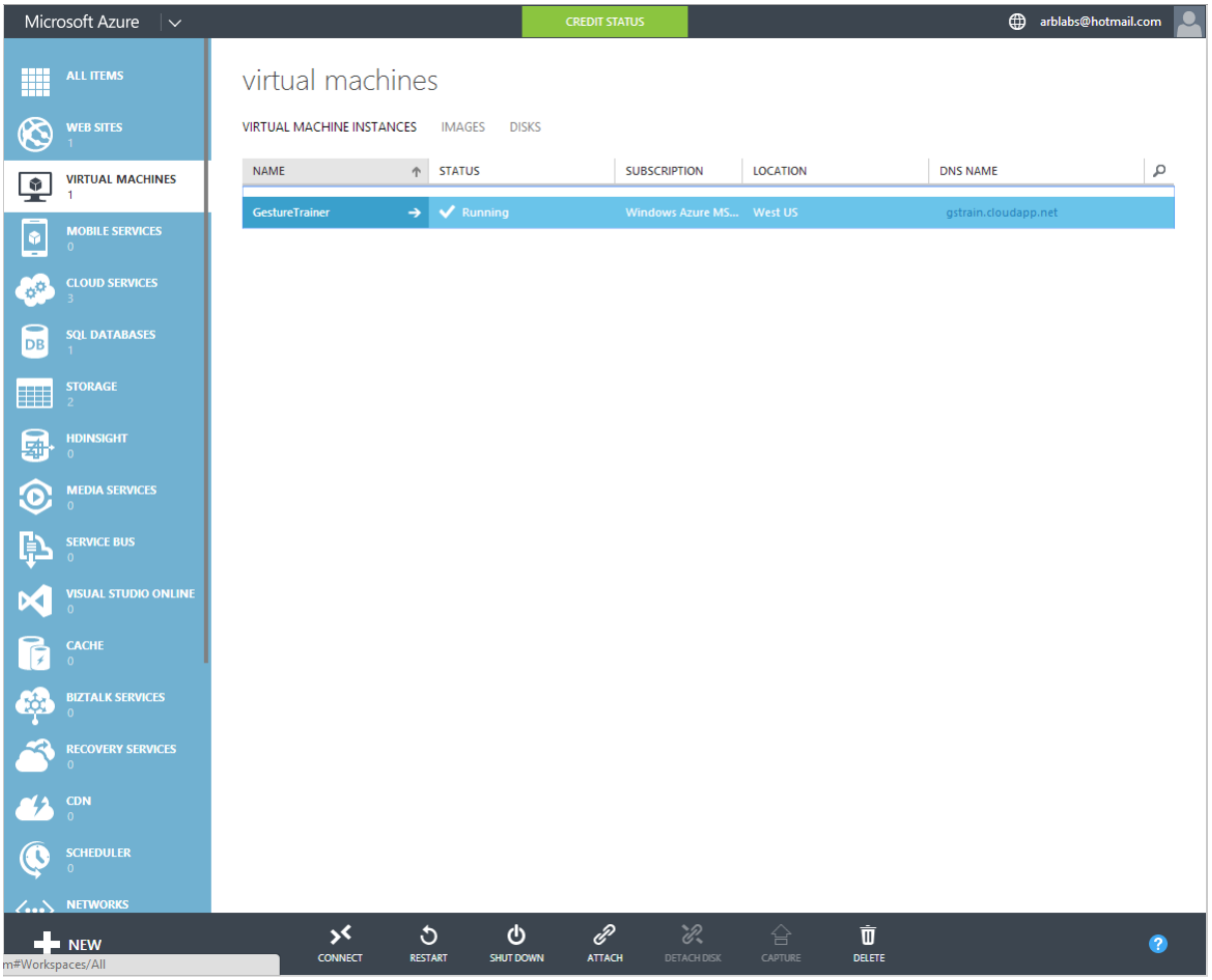


FIGURE C.16: Screen capture of Gesture Studio Cloud Azure Environment Back-end.

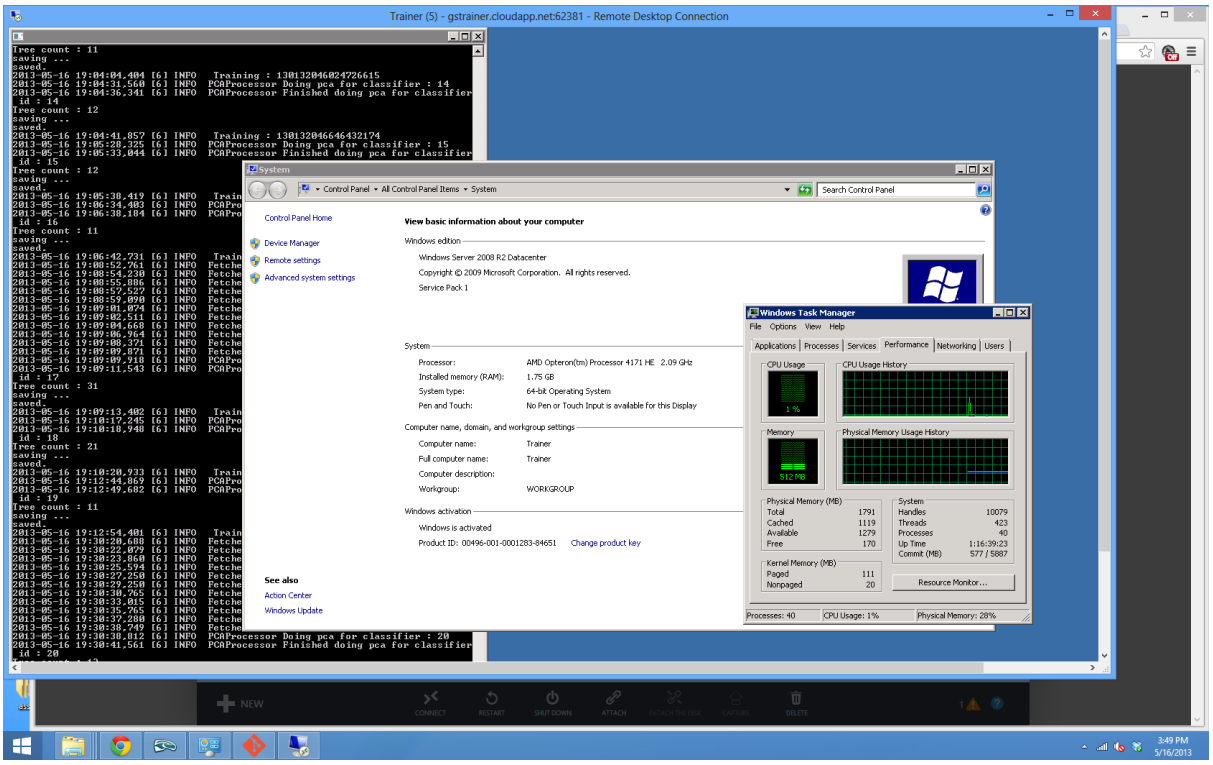


FIGURE C.17: Gesture Studio Cloud host hardware specifications

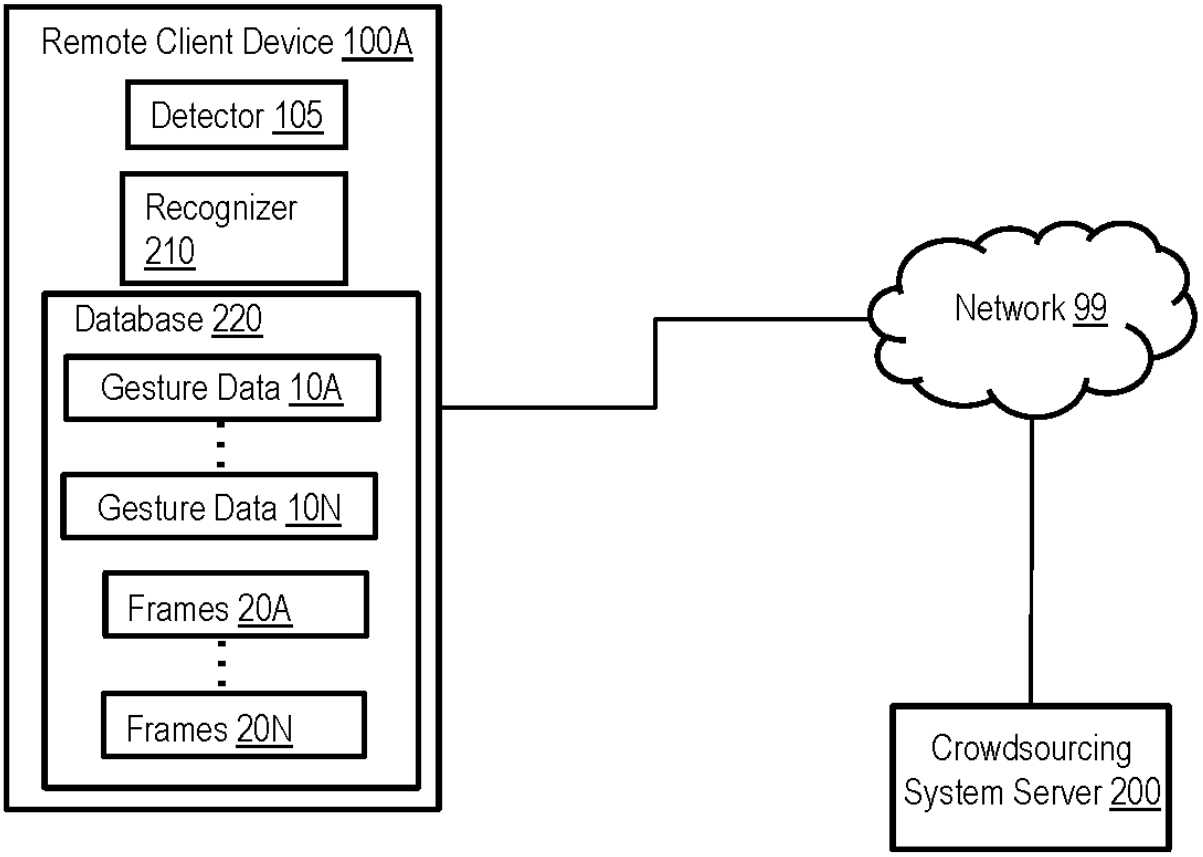


FIGURE C.18: Gesture Studio Cloud client user high level overview.

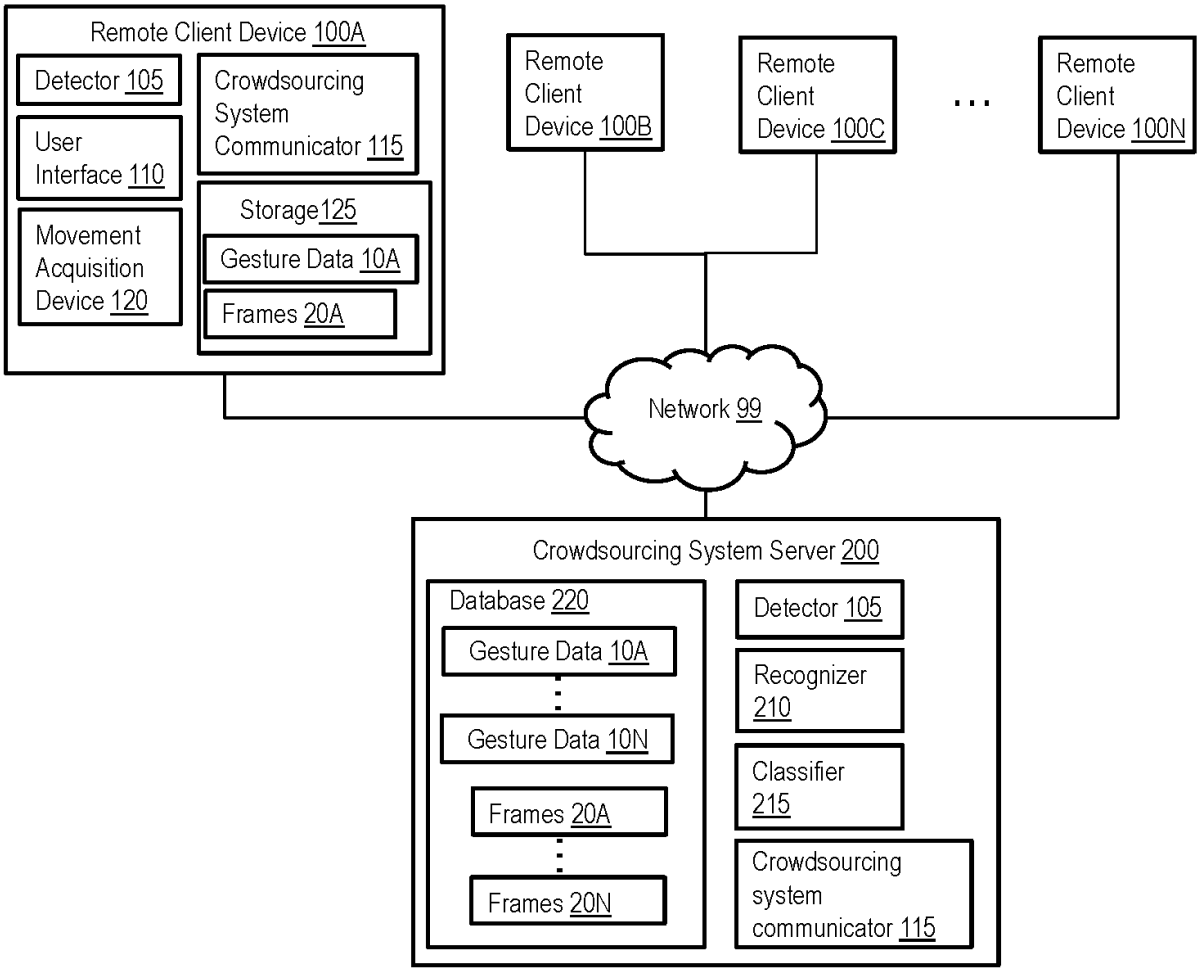
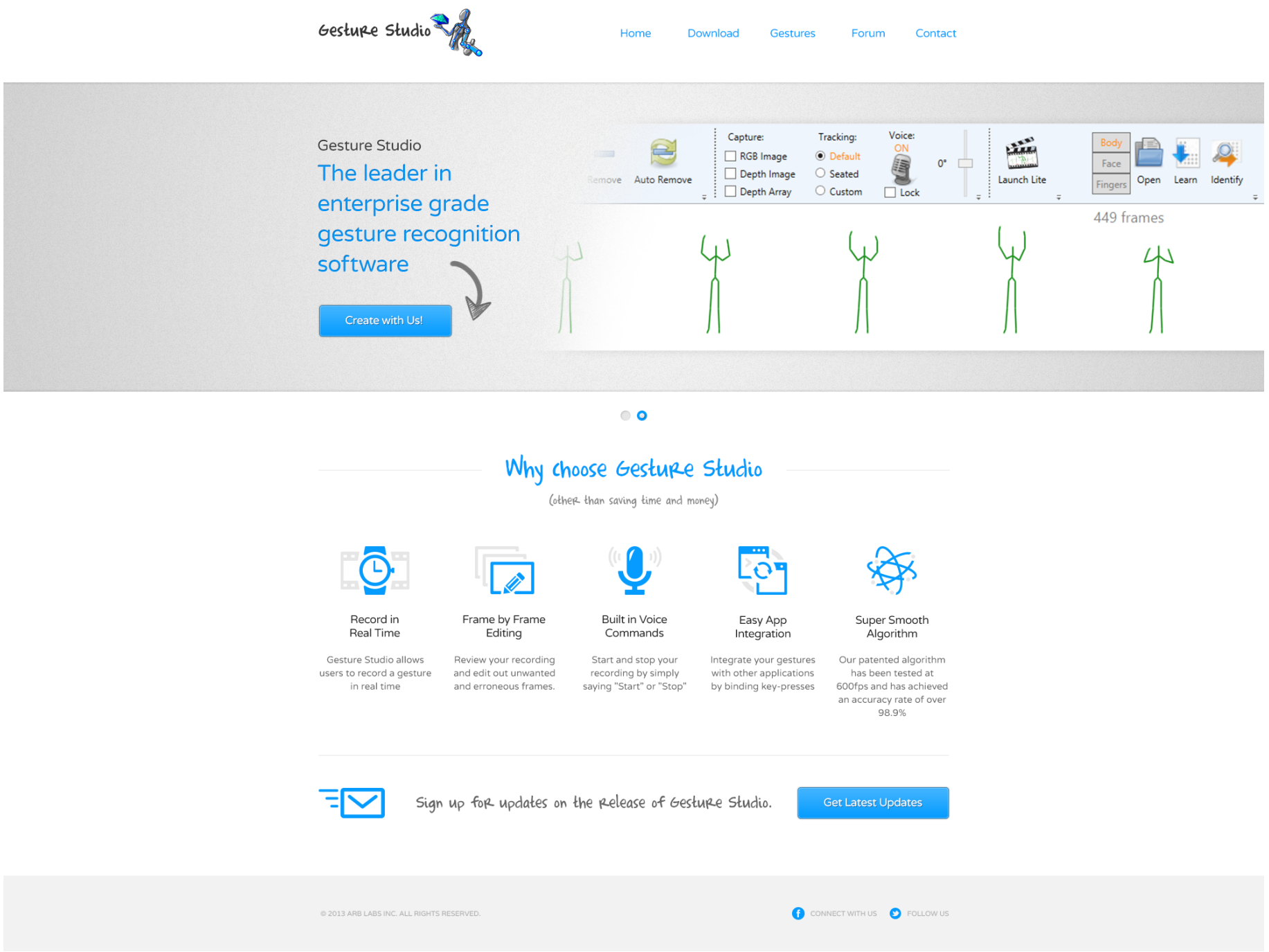


FIGURE C.19: Gesture Studio Cloud host level overview.


Gesture Studio
Home
Download
Gestures
Forum
Contact
Gesture Studio
The leader in enterprise grade gesture recognition software
Create with Us!
Remove
Auto Remove
Capture:
RGB Image
Depth Image
Depth Array
Tracking:
Default
Seated
Custom
Voice:
ON
Lock
0°
Launch Lite
Body
Face
Fingers
Open
Learn
Identify
449 frames
Why choose Gesture Studio
(other than saving time and money)
Record in Real Time
Gesture Studio allows users to record a gesture in real time
Frame by Frame Editing
Review your recording and edit out unwanted and erroneous frames.
Built in Voice Commands
Start and stop your recording by simply saying "Start" or "Stop"
Easy App Integration
Integrate your gestures with other applications by binding key-presses
Super Smooth Algorithm
Our patented algorithm has been tested at 600fps and has achieved an accuracy rate of over 98.9%
Sign up for updates on the Release of Gesture Studio.
Get Latest Updates
© 2013 ARB LABS INC. ALL RIGHTS RESERVED.
CONNECT WITH US
FOLLOW US


Gesture Studio
Home
Download
Gestures
Forum
Contact
Getting Started Guide
Teaching Gestures Tutorial
Why choose Gesture Studio
(other than saving time and money)
Record in Real Time
Gesture Studio allows users to record a gesture in real time
Frame by Frame Editing
Review your recording and edit out unwanted and erroneous frames.
Built in Voice Commands
Start and stop your recording by simply saying "Start" or "Stop"
Easy App Integration
Integrate your gestures with other applications by binding key-presses
Super Smooth Algorithm
Our patented algorithm has been tested at 600fps and has achieved an accuracy rate of over 98.9%
Sign up for updates on the Release of Gesture Studio.
Get Latest Updates
© 2013 ARB LABS INC. ALL RIGHTS RESERVED.
CONNECT WITH US
FOLLOW US

Appendix D

Gesture Studio Cloud Benchmark Log for Minecraft Game Gestures

2013-05-16 15:48:32,974 [1] INFO Starting worker GS.Cloud.TrainerRole.TrainWorker

2013-05-16 15:48:34,052 [6] INFO Fetcher Extracting feature from left-pocket:10

2013-05-16 15:48:37,021 [6] INFO Fetcher Extracting feature from left-pocket:11

2013-05-16 15:48:38,692 [6] INFO Fetcher Extracting feature from left-pocket:12

2013-05-16 15:48:40,052 [6] INFO Fetcher Extracting feature from left-tuck:13
2013-05-16 15:48:41,973 [6] INFO Fetcher Extracting feature from left-tuck:14
2013-05-16 15:48:43,551 [6] INFO Fetcher Extracting feature from neutral:15
2013-05-16 15:48:44,754 [6] INFO Fetcher Extracting feature from neutral:16
2013-05-16 15:48:46,629 [6] INFO Fetcher Extracting feature from neutral:17
2013-05-16 15:48:48,332 [6] INFO Fetcher Extracting feature from right-pocket:18
2013-05-16 15:48:50,817 [6] INFO Fetcher Extracting feature from right-pocket:19
2013-05-16 15:48:52,161 [6] INFO Fetcher Extracting feature from right-pocket:20

2013-05-16 15:48:54,004 [6] INFO Fetcher Extracting feature from right-tuck:21

2013-05-16 15:48:55,582 [6] INFO Fetcher Extracting feature from right-tuck:22

2013-05-16 15:48:57,004 [6] INFO Fetcher Extracting feature from right-tuck:23

2013-05-16 15:48:59,317 [6] INFO PCAProcessor Doing pca for classifier : 6

2013-05-16 15:49:01,333 [6] INFO PCAProcessor Finished doing pca for classifier

id : 6

Tree count : 16

saving ...

saved.

2013-05-16 15:49:04,583 [6] INFO Training : 130130200857196884

2013-05-16 15:49:05,192 [6] INFO PCAProcessor Doing pca for classifier : 7

2013-05-16 15:49:07,114 [6] INFO PCAProcessor Finished doing pca for classifier

id : 7

Tree count : 17

saving ...

saved.

2013-05-16 15:49:09,551 [6] INFO Training : 130130203228441595

2013-05-16 15:49:10,036 [6] INFO PCAProcessor Doing pca for classifier : 8

2013-05-16 15:49:11,864 [6] INFO PCAProcessor Finished doing pca for classifier

id : 8

Tree count : 15

saving ...

saved.

2013-05-16 15:49:14,270 [6] INFO Training : 130130320578212456

2013-05-16 15:49:15,442 [6] INFO PCAProcessor Doing pca for classifier : 9

2013-05-16 15:49:19,379 [6] INFO PCAProcessor Finished doing pca for classifier

id : 9

Tree count : 51

saving ...

saved.

2013-05-16 15:49:24,676 [6] INFO Training : 130131919364343832

2013-05-16 15:50:15,427 [6] INFO PCAProcessor Doing pca for classifier : 10

2013-05-16 15:50:19,177 [6] INFO PCAProcessor Finished doing pca for classifier

id : 10

Tree count : 79

saving ...

saved.

2013-05-16 15:50:25,286 [6] INFO Training : 130131930161531396

2013-05-16 16:41:18,594 [6] INFO PCAProcessor Doing pca for classifier : 11

2013-05-16 16:41:20,594 [6] INFO PCAProcessor Finished doing pca for classifier

id : 11

Tree count : 16

saving ...

saved.

2013-05-16 16:41:23,031 [6] INFO Training : 130131960783597331

2013-05-16 16:49:24,422 [6] INFO PCAProcessor Doing pca for classifier : 12

2013-05-16 16:49:26,204 [6] INFO PCAProcessor Finished doing pca for classifier

id : 12

Tree count : 14

saving ...

saved.

2013-05-16 16:49:28,516 [6] INFO Training : 130131965627332066

2013-05-16 19:03:28,249 [6] INFO Fetcher Extracting feature from left-pocket:24

2013-05-16 19:03:30,202 [6] INFO Fetcher Extracting feature from left-pocket:25

2013-05-16 19:03:31,858 [6] INFO Fetcher Extracting feature from left-pocket:26

2013-05-16 19:03:33,311 [6] INFO Fetcher Extracting feature from left-tuck:27

2013-05-16 19:03:35,139 [6] INFO Fetcher Extracting feature from left-tuck:28

2013-05-16 19:03:36,968 [6] INFO Fetcher Extracting feature from neutral:29

2013-05-16 19:03:38,139 [6] INFO Fetcher Extracting feature from neutral:30

2013-05-16 19:03:39,952 [6] INFO Fetcher Extracting feature from neutral:31

2013-05-16 19:03:42,483 [6] INFO Fetcher Extracting feature from right-pocket:32

2013-05-16 19:03:45,155 [6] INFO Fetcher Extracting feature from right-pocket:33

2013-05-16 19:03:46,514 [6] INFO Fetcher Extracting feature from right-pocket:34

2013-05-16 19:03:48,639 [6] INFO Fetcher Extracting feature from right-tuck:35

2013-05-16 19:03:50,170 [6] INFO Fetcher Extracting feature from right-tuck:36

2013-05-16 19:03:51,811 [6] INFO Fetcher Extracting feature from right-tuck:37

2013-05-16 19:03:54,217 [6] INFO PCAProcessor Doing pca for classifier : 13

2013-05-16 19:03:59,076 [6] INFO PCAProcessor Finished doing pca for classifier

id : 13

Tree count : 11

saving ...

saved.

2013-05-16 19:04:04,404 [6] INFO Training : 130132046024726615

2013-05-16 19:04:31,560 [6] INFO PCAProcessor Doing pca for classifier : 14

2013-05-16 19:04:36,341 [6] INFO PCAProcessor Finished doing pca for classifier

id : 14

Tree count : 12

saving ...

saved.

2013-05-16 19:04:41,857 [6] INFO Training : 130132046646432174

2013-05-16 19:05:28,325 [6] INFO PCAProcessor Doing pca for classifier : 15

2013-05-16 19:05:33,044 [6] INFO PCAProcessor Finished doing pca for classifier

id : 15

Tree count : 12

saving ...

saved.

2013-05-16 19:05:38,419 [6] INFO Training : 130132047203624044

2013-05-16 19:06:34,403 [6] INFO PCAProcessor Doing pca for classifier : 16

2013-05-16 19:06:38,184 [6] INFO PCAProcessor Finished doing pca for classifier

id : 16

Tree count : 11

saving ...

saved.

2013-05-16 19:06:42,731 [6] INFO Training : 130132047943937245

2013-05-16 19:08:52,761 [6] INFO Fetcher Extracting feature from forward:38

2013-05-16 19:08:54,230 [6] INFO Fetcher Extracting feature from forward:39

2013-05-16 19:08:55,886 [6] INFO Fetcher Extracting feature from jump:40

2013-05-16 19:08:57,527 [6] INFO Fetcher Extracting feature from jump:41

2013-05-16 19:08:59,090 [6] INFO Fetcher Extracting feature from left:42

2013-05-16 19:09:01,074 [6] INFO Fetcher Extracting feature from left:43

2013-05-16 19:09:02,511 [6] INFO Fetcher Extracting feature from mine:44

2013-05-16 19:09:04,668 [6] INFO Fetcher Extracting feature from mine:45

2013-05-16 19:09:06,964 [6] INFO Fetcher Extracting feature from none:46

2013-05-16 19:09:08,371 [6] INFO Fetcher Extracting feature from none:47

2013-05-16 19:09:09,871 [6] INFO Fetcher Extracting feature from none:48

2013-05-16 19:09:09,918 [6] INFO PCAProcessor Doing pca for classifier : 17

2013-05-16 19:09:11,543 [6] INFO PCAProcessor Finished doing pca for classifier

id : 17

Tree count : 31

saving ...

saved.

2013-05-16 19:09:13,402 [6] INFO Training : 130132049266662043

2013-05-16 19:10:17,245 [6] INFO PCAProcessor Doing pca for classifier : 18

2013-05-16 19:10:18,948 [6] INFO PCAProcessor Finished doing pca for classifier

id : 18

Tree count : 21

saving ...

saved.

2013-05-16 19:10:20,933 [6] INFO Training : 130132050112540424

2013-05-16 19:12:44,869 [6] INFO PCAProcessor Doing pca for classifier : 19

2013-05-16 19:12:49,682 [6] INFO PCAProcessor Finished doing pca for classifier

id : 19

Tree count : 11

saving ...

saved.

2013-05-16 19:12:54,401 [6] INFO Training : 130132051601233607

Bibliography

[1] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, p. 10, 2009.

[2] H. Zhang, W. Du, and H. Li, "Kinect Gesture Recognition for Interactive System," *Cs229.Stanford.Edu*, pp. 1–5.

[3] G. Johansson, "Visual perception of biological motion and a model for its analysis," *Perception & Psychophysics*, vol. 14, no. 2, pp. 201–211, 1973.

[4] A. Goycoolea, "Visual motion," *Scientific American*, 1998.

[5] S. Fothergill, H. Mentis, P. Kohli, and S. Nowozin, "Instructing people for training gestural interactive systems," 2012.

[6] Y.-H. Hsieh, S. C. Hidayati, W.-H. Cheng, M.-C. Hu, and K.-L. Hua, "Who's the Best Charades Player? Mining Iconic Movement of Semantic Concepts," in *MultiMedia Modeling*, pp. 231–241, Springer, 2014.

[7] A. Kendon, "Some relationships between body motion and speech: An analysis of an example," *Studies in dyadic communication*, vol. 7, p. 177, 1972.

[8] D. McNeill, *Hand and Mind: What Gestures Reveal About Thought*. University of Chicago Press, 1992.

[9] D. McNeill, "So you think gestures are nonverbal?," *Psychological Review*, vol. 92, no. 3, pp. 350–371, 1985.

[10] M. Kawashima, A. Shimada, and R. I. Taniguchi, "Early recognition of gesture patterns using sparse code of self-organizing map," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5629 LNCS, pp. 116–123, 2009.

[11] M. Oshita and T. Matsunaga, "Automatic learning of gesture recognition model using {SOM} and {SVM}," *6th International Symposium on Visual Computing 2010 (Lecture Notes in Computer Science 6453)*, pp. 751–760, 2010.

[12] J. Wu, G. Pan, D. Zhang, and G. Qi, "Gesture recognition with a 3-d accelerometer," *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing*, pp. 25–38, 2009.

[13] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Transactions on Graphics*, vol. 28, p. 1, 2009.

[14] V. Scholz, T. Stich, M. Keckeisen, M. Wacker, and M. Magnor, "Garment motion capture using color-coded patterns," *Computer Graphics Forum*, vol. 24, no. 3, pp. 439–447, 2005.

[15] P. Brandl, M. Haller, M. Hurnaus, V. Lugmayr, J. Oberngruber, C. Oster, C. Schafleitner, and M. Billinghurst, "An adaptable rear-projection screen using digital pens and hand gestures," in *Proceedings 17th International Conference on Artificial Reality and Telexistence, ICAT 2007*, pp. 49–54, IEEE, 2007.

[16] X. Zhang, X. Chen, W. Wang, and J.-h. Yang, "Hand gesture recognition and virtual game control based on 3D accelerometer and EMG sensors," in *Proceedings of the 14th international conference on Intelligent user interfaces*, pp. 401–406, ACM, 2009.

[17] X. Chen, X. Zhang, Z.-Y. Zhao, J.-H. Yang, V. Lantz, and K.-Q. Wang, "Hand Gesture Recognition Research Based on Surface EMG Sensors and 2D-accelerometers," in *2007 11th IEEE International Symposium on Wearable Computers*, pp. 11–14, IEEE, 2007.

[18] J. C. Lee, "Hacking the Nintendo Wii remote," *IEEE Pervasive Computing*, vol. 7, no. 3, pp. 39–45, 2008.

[19] a. C. Downton and H. Drouet, "Image analysis for model-based sign language coding,"," in *6th International Conference on Image Analysis and Processing.*, p. 637, 1991.

[20] J. Lee and T. Kunii, "Constraint-based hand animation," *Models and techniques in computer animation*, pp. 110–127, 1993.

[21] J. Kuch and T. Huang, "Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration," in *Proceedings of IEEE International Conference on Computer Vision*, pp. 666–671, 1995.

[22] P. Breuer, C. Eckes, S. Muller, and S. M, "Hand Gesture Recognition with a novel IR Time-of-Flight Range Camera–A pilot study," *Computer Vision/Computer Graphics Collaboration Techniques*, vol. 4418, pp. 247–260, 2007.

[23] S. Ahmad, "A usable real-time 3D hand tracker," in *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1257–1261, IEEE, 1994.

[24] A. Bulzacki, L. Zhao, and L. Guan, "Introduction to Gesture Recognition through Vectorization," *Visual Information Processing 2007*, 2007.

[25] T. Baudel and M. Beaudouin-Lafon, "Charade: remote control of objects using free-hand gestures," *Communications of the ACM*, vol. 36, no. 7, pp. 28–35, 1993.

[26] K.Boehm, W. Broll and M.Sokolewicz, "Dynamic gesture recognition using neural networks; a fundament for advanced interaction construction," in *Dynamic gesturere cognition using neural networks; af un-dament for advanced interaction construction*, vol. 2177, pp. California,USA, 1994.

[27] R. Cipolla, Y. Okamoto, and Y. Kuno, "Robust structure from motion using motion parallax," in *1993 (4th) International Conference on Computer Vision*, pp. 374–382, IEEE, 1993.

[28] K. Cho and S. M. Dunn, "Learning shape classes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, pp. 882–888, 1994.

[29] T. Darrell and A. Pentland, "Space-time gestures," pp. 335–340, IEEE, 2002.

[30] J. Davis, "Visual gesture recognition," in *IEE Proceedings - Vision, Image, and Signal Processing*, vol. 141, p. 101, IET, 1994.

[31] M. Elmezain, a. Al-Hamadi, and B. Michaelis, "Real-time capable system for hand gesture recognition using hidden markov models in stereo color image sequences," *Winter School of Computer Graphics*, vol. 16, no. 1, pp. 65–72, 2008.

[32] M. Etoh, a. Tomono, and F. Kishino, "Stereo-based description by generalized cylinder complexes from occluding contours," *Systems and Computers in Japan*, vol. 22, no. 12, pp. 79–89, 1991.

[33] S. Fels and G. Hinton, "Glove-TalkII: an adaptive gesture-to-formant interface," in *Proceedings of the SIGCHI conference on Human factors in computing systems.*, pp. 456–463, ACM Press/Addison-Wesley Publishing Co., 1995.

[34] M. Fukumoto, Y. Suenaga, and K. Mase, ""Finger-Pointer": Pointing interface by image processing," *Computers & Graphics*, vol. 18, no. 5, pp. 633–642, 1994.

[35] W. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *International Workshop on Automatic Face and Gesture Recognition*, vol. 12, pp. 296–301, Citeseer, 1995.

[36] A. L. Hubbard, S. M. Wilson, D. E. Callan, and M. Dapretto, "Giving speech a hand: Gesture modulates activity in auditory cortex during speech perception," *Human Brain Mapping*, vol. 30, no. 3, pp. 1028–1037, 2009.

[37] C. Kervrann and F. Heitz, "Learning structure and deformation modes of nonrigid objects in long image sequences," 1995.

[38] F. C. M. Kjeldsen, *Visual Interpretation of Hand Gestures as a Practical Interface Modality*. PhD thesis, Columbia University, 1997.

[39] M. W. Krueger, "Environmental technology: making the real world virtual," *Communications of the ACM*, vol. 36, no. 7, pp. 36–37, 1993.

[40] M. E. Latoschik and I. Wachsmuth, "Exploiting Distant Pointing Gestures for Object Selection in a {V}irtual {E}nvironment," in *Gesture and Sign Language in Human-Computer Interaction*, vol. 1371, pp. 185–196, Springer Berlin Heidelberg, 1998.

[41] C. Maggioni, "A novel gestural input device for virtual reality," in *Proceedings of IEEE Virtual Reality Annual International Symposium*, pp. 118–124, IEEE, 1993.

[42] J. Schlenzig, E. Hunter, and R. Jain, "Vision based hand gesture interpretation using recursive estimation," in *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1267–1271, IEEE, 1994.

[43] J. Segen, "Controlling computers with gloveless gestures," *Proceedings of virtual reality systems*, 1993.

[44] J. Segen and S. Kumar, "Video acquired gesture interfaces for the handicapped," in *Proceedings of the sixth ACM international conference on Multimedia*, pp. 45–48, ACM Press, 1998.

[45] J. Segen and S. Kumar, "Gesture vr: vision-based 3d hand interace for spatial interaction," in *Proceedings of the sixth ACM international ...*, pp. 455–464, ACM Press, 1998.

[46] T. Takahashi and F. Kishino, "Hand gesture coding based on experiments using a hand gesture interface device," *ACM SIGCHI Bulletin*, vol. 23, no. 2, pp. 67–74, 1991.

[47] R. Wang, "Real-time hand-tracking as a user input device," *Massachusetts Institute of Technology, Cambridge, MA*, 2008.

[48] H. S. Yoon, J. Soh, Y. J. Bae, and H. Seung Yang, "Hand gesture recognition using combined features of location, angle and velocity," *Pattern Recognition*, vol. 34, pp. 1491–1501, Jan. 2001.

[49] K. Vaananen and K. Bohm, "Gesture driven interaction as a human factor in virtual environments-an approach with neural networks," *Virtual reality systems*, pp. 93–106, 1993.

[50] D. Forsyth and R. White, "Deforming objects provide better camera calibration," tech. rep., Citeseer, 2005.

[51] I. Guskov, S. Klibanov, and B. Bryant, "Trackable surfaces," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 251–257, Eurographics Association, 2003.

[52] W. Groß, "Grundzüge der Mengenlehre," *Monatshefte für Mathematik und Physik*, vol. 26, no. 1, pp. A34–A35, 1915.

[53] A. Agarwal and B. Triggs, "3D human pose from silhouettes by relevance vector regression," *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. 882–888, 2004.

[54] U. Hillenbrand and G. Hirzinger, "Probabilistic Search for Object Segmentation and Recognition," *Proceedings of the 7th European Conference on Computer Vision (ECCV '02) Part III*, p. 18, 2002.

[55] J. ORourke and N. Badler, "Model-based image analysis of human motion using constraint propagation," *Ieee Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 6, pp. 522–536, 1980.

[56] D. Hogg, "Model-based vision: a program to see a walking person," *Image and Vision Computing*, vol. 1, no. 1, pp. 5–20, 1983.

[57] M. Yamamoto and K. Koshikawa, "Human motion analysis based on a robot arm model," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 664–665, IEEE, 1991.

[58] D. Ramanan and D. Forsyth, "Finding and tracking people from the bottom up," *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, pp. II–467–II–474, 2003.

[59] M. Siddiqui, *Human pose estimation from a single view point*. PhD thesis, Diss. University of Southern California, 2009.

[60] L. S. H. Sidenbladh, J. Blanck, "Implicit Probabilistic Models of Human Motion for Synthesis and Tracking," *Computer Vision - ECCV 2002*, vol. 2350, pp. 784–800, 2002.

[61] D. Grest, J. Woetzel, and R. Koch, "Nonlinear Body Pose Estimation from Depth Images," *Pattern Recognition*, pp. 285–292, 2005.

[62] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision*, vol. 61, pp. 55–79, Jan. 2005.

[63] G. Rogez, J. Rihan, S. Ramalingam, C. Orrite, and P. H. S. Torr, "Randomized trees for human pose detection," *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 1–8, June 2008.

[64] R. Okada and S. Soatto, "Relevant feature selection for human pose estimation and localization in cluttered images," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5303 LNCS, pp. 434–445, 2008.

[65] S. Knoop, S. Vacek, and R. Dillmann, "Sensor fusion for 3D human body tracking with an articulated 3D body model," *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 1686–1691, 2006.

[66] R. Navaratnam, A. W. Fitzgibbon, and R. Cipolla, "The joint manifold model for semi-supervised multi-valued regression," *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[67] L. Sigal, S. Bhatia, and S. Roth, "Tracking loose-limbed people," *Computer Vision and Image Understanding*, no. Figure 1, 2004.

[68] C. Bregler and J. Malik, "Tracking people with twists and exponential maps," *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, pp. 8–15, 1998.

[69] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proceedings - International Conference on Pattern Recognition*, vol. 3, pp. 32–36, IEEE, 2004.

[70] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, pp. 1395–1402, IEEE, 2005.

[71] D. Weinland, R. Ronfard, and E. Boyer, "Free viewpoint action recognition using motion history volumes," *Computer Vision and Image Understanding*, vol. 104, no. October 2006, pp. 249–257, 2006.

[72] P. Gomes, S.-m. Morgens, and S.-r. Smith, "Gesture Classification from Kinect Data," pp. 1–15, 2012.

[73] A. Kurakin, Z. Zhang, and Z. Liu, "A real time system for dynamic hand gesture recognition with a depth sensor," *20th European Signal Processing Conference (EUSIPCO 2012)*, no. Eusipco, pp. 1975–1979, 2012.

[74] S.-s. Learning, C. M. Christoudias, T. J. Darrell, and T. P. Orlando, "Probabilistic Models for Multi-View by Certified by ...," *Work*, 2009.

[75] T. E. Starner, *Visual Recognition of American Sign Language Using Hidden Markov Models*. PhD thesis, 1995.

[76] Y. Nam and K. Wohn, "Recognition of Space-Time Hand-Gestures Using Hidden Markov Model," in *ACM Symposium on Virtual Reality Software and Technology*, pp. 51–58, Citeseer, 1996.

[77] L. Xia, C. C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3D joints," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 20–27, IEEE, 2012.

[78] Y. Zhou, Z. Cheng, L. Jing, J. Wang, and T. Huang, "Pre-classification based hidden Markov model for quick and accurate gesture recognition using a finger-worn device," *Applied Intelligence*, vol. 40, no. 4, pp. 613–622, 2014.

[79] J. Beh, D. K. Han, R. Durasiwami, and H. Ko, "Hidden Markov Model on a unit hypersphere space for gesture trajectory recognition," *Pattern Recognition Letters*, vol. 36, pp. 144–153, 2014.

[80] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1290–1297, June 2012.

[81] D. Weinland, E. Boyer, and R. Ronfard, "Action recognition from arbitrary views using 3D exemplars," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1–7, IEEE, 2007.

[82] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 657–675, 2009.

[83] H.-R. Choi, E.-J. Kim, and T.-Y. Kim, "A DTW gesture recognition system based on gesture orientation histogram," in *Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on*, pp. 1–2, IEEE, 2014.

[84] M. Reyes, G. Domínguez, and S. Escalera, "Featureweighting in dynamic timewarping for gesture recognition in depth data," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1182–1188, IEEE, 2011.

[85] S. Sempena, N. U. Maulidevi, and P. R. Aryan, "Human action recognition using Dynamic Time Warping," *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, vol. 30, no. 3, pp. 1–5, 2011.

[86] S. Celebi, A. S. Aydin, T. T. Temiz, and T. Arici, "Gesture Recognition Using Skeleton Data with Weighted Dynamic Time Warping," in *8th International Conference on Computer Vision Theory and Applications (VISAPP 2013)*, p. Paper #79, 2013.

[87] Y. Song, L. P. Morency, and R. Davis, "Distribution-sensitive learning for imbalanced datasets," *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, FG 2013*, 2013.

[88] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[89] L. U. o. C. Breiman, "Random forest," *Machine Learning*, vol. 45, no. 1, pp. 1–35, 1999.

[90] I. Spiro, "Motion chain: a webcam game for crowdsourcing gesture collection," in *CHI'12 Extended Abstracts on Human Factors in Computing Systems*, pp. 1345–1350, ACM, 2012.

[91] M. Roemmele, H. Archer-McClellan, and A. S. Gordon, "Triangle charades: a data-collection game for recognizing actions in motion trajectories.," in *IUI*, pp. 209–214, 2014.

[92] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 775–781, 2005.

[93] B. Freedman, S. Alexander, M. Machline, and Y. Arieli, "Depth Mapping Using Projected Patterns," 2010.

[94] A. Bulzacki, L. Zhao, L. Guan, and K. Raahemifar, "Computerized recognition of human gestures," in *Canadian Conference on Electrical and Computer Engineering*, pp. 1003–1006, IEEE, 2008.

[95] S. R. Fanello, T. Paek, C. Keskin, S. Izadi, P. Kohli, D. Kim, D. Sweeney, A. Criminisi, J. Shotton, and S. B. Kang, "Learning to be a depth camera for close-range human capture and interaction," in *ACM Transactions on Graphics*, vol. 33, pp. 1–11, ACM – Association for Computing Machinery, July 2014.

[96] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact Maximum A Posteriori Estimation for Binary Images," *Journal of the Royal Statistical Society*, vol. 51, no. 2, pp. 271–279, 1989.

[97] D. Vranic and D. Saupe, "Description of 3D-shape using a complex function on the sphere," in *Proceedings. IEEE International Conference on Multimedia and Expo*, vol. 1, pp. 177–180 vol.1, 2002.

[98] J.-L. Shih, C.-H. Lee, and J. T. Wang, "A new 3D model retrieval approach based on the elevation descriptor," *Pattern Recognition*, vol. 40, no. 1, pp. 283–295, 2007.

[99] T. Funkhouser, M. Kazhdan, P. Min, and P. Shilane, "Shape-based retrieval and analysis of 3d models," *Communications of the ACM*, vol. 48, no. 6, p. 58, 2005.

[100] G. Ye, J. Corso, and G. Hager, "Gesture Recognition Using 3D Appearance and Motion Features," in *2004 Conference on Computer Vision and Pattern Recognition Workshop*, p. 160, 2004.

[101] D. Ververidis and C. Kotropoulos, "Sequential forward feature selection with low computational cost," *Proceedings of the 8th European Signal Processing Conference*, 2005.

[102] S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with Gaussian kernel.," *Neural computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[103] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[104] H. Kim, Y. Kim, D. Ko, J. Kim, and E. C. Lee, "Pointing Gesture Interface for Large Display Environments Based on the Kinect Skeleton Model," in *Future Information Technology*, pp. 509–514, Springer, 2014.

[105] H. Kim, Y. Kim, and E. C. Lee, "Method for User Interface of Large Displays Using Arm Pointing and Finger Counting Gesture Recognition," *The Scientific World Journal*, vol. 2014, 2014.

[106] A. Gonsales and M. Kyan, "Trajectory Analysis on Spherical Self-Organizing Maps with Application to Gesture Recognition," in *Advances in Self-Organizing Maps* (P. A. Estévez, J. C. Príncipe, and P. Zegers, eds.), vol. 198 of *Advances in Intelligent Systems and Computing*, pp. 125–134, Springer, 2012.

[107] N. M. Khan, M. Kyan, and L. Guan, "Intuitive volume exploration through spherical self-organizing map," *Advances in Intelligent Systems and Computing*, vol. 198 AISC, pp. 75–84, 2013.