Computer Vision-based Real-time 3D Full-body
Motion Gesture Recognition for Aircraft Handling

A thesis submitted in partial fulfilment of the requirements for the Degree of
Master of Science in Computer Science and Engineering

Submitted By
Alina Zaman
Exam Roll: 1623
MS 2013-14
Registration No: Ha-1186
Session: 2008-09

Supervised By
Dr. Md. Hasanuzzaman
Professor
Department of Computer Science and Engineering
University of Dhaka

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY OF DHAKA, DHAKA-1000, BANGLADESH

January 2016
Declaration

I, hereby, declare that the work presented in this thesis is the outcome of the investigation performed by me under the supervision of Dr. Md. Hasanuzzaman, Professor, Department of Computer Science and Engineering, University of Dhaka. I also declare that no part of this thesis and thereof has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned                                             Signature

(Dr. Md. Hasanuzzaman)                         (Alina Zaman)
Supervisor                                             Candidate
Acknowledgements

First of all, thanks to Almighty Allah for giving me the potency and energy to complete this thesis successfully.

I want to express out gratefulness towards my thesis supervisor Dr. Md. Hasanuzzaman for his valuable advices and important suggestions. His regular and active supervision and erudite directions from the beginning to the end were the driving forces for the successful completion of the research work.

I would like to convey my gratitude to all of my teachers at the Department of Computer Science and Engineering, University of Dhaka. Discussions with many of them have helped me tremendously in improving the quality of the work. I am particularly thankful to Mr. Hasnain Heickal and Md. Shiplu Hawlader for their valuable suggestions. I also thank the department for providing me with resources which were necessary for the preparation of the thesis.

And last but not the least, I would like to express thanks to my parents and family members for their tremendous support and inspiration.
Abstract

Gestures are considered as one of the most natural expressive way for communications between human and computers in virtual system. To implement human-machine natural interaction system, the machine should understand 2D/3D gesture happened in real-time consisting both posture and motion. This thesis presents a vision-based real-time 3D gesture recognition system for aircraft handling that integrates information from body movements and hand postures. A Kinect motion sensor device has been used to collect 3D images of human body, and tracks body and hand together. Kinect for Windows version 2.0 SDK helps to detect the 3D body-joint points. Motion trajectory along with the body joint coordinates used as a feature to recognize 3D motion gesture. A Kinect skeletal coordinate system is established and each 3D joint points are scaled to make the system invariant of person’s distance from the sensor, height, etc. For gesture learning and recognition, four classification algorithms such as Naive Bayes, Neural Network, Hidden Markov Model and Support Vector Machine are used. The system is trained and tested to recognize 20 NATOPS gestures for an aircraft carrier flight deck environment, where humans interact with unmanned vehicles using predefined body and hand gesture vocabulary. The accuracy of the system is 93.7% in Naive Bayes, 98.1% in Support Vector Machine, 97% in Neural Network and 90.8% in Hidden Markov Model classifier. To test the consistency of output of all the four classifiers kappa-test has been applied to the system and the result was satisfactory with the kappa value 0.91.
# Contents

1 Introduction .......................................................... 1
  1.1 Motivation ......................................................... 3
  1.2 Objectives ......................................................... 4
  1.3 Overview of the Proposed System ............................... 5
  1.4 Contributions ..................................................... 6
  1.5 Outline of the Thesis ............................................ 7

2 Literature Review .................................................. 9
  2.1 Definitions and Descriptions ................................. 10
    2.1.1 Gesture ................................................... 10
    2.1.2 Hand Posture ............................................ 11
    2.1.3 Human-Computer Interaction .......................... 12
    2.1.4 Human-Robot Interaction ............................. 13
  2.2 Gesture Recognition ........................................... 14
    2.2.1 Glove-based Approaches ............................... 16
    2.2.2 Vision-based Approach ................................ 19
    2.2.3 Kinect-based Approach ................................ 26
    2.2.4 Gesture Recognition for Aircraft Handling on NATOPS Database 37
  2.3 Gesture Classification ........................................ 41
    2.3.1 Hidden Markov Model ................................. 42
    2.3.2 Neural Network-based Classifier ....................... 44
    2.3.3 Naive Bayes Classifier ................................ 45
    2.3.4 Support Vector Machine ............................... 47
  2.4 Summary .......................................................... 48

3 Proposed System Descriptions ................................. 50
  3.1 Problem Formulation ........................................... 50
  3.2 Methods of Implementation .................................. 52
    3.2.1 3D Data Acquisition .................................. 53
    3.2.2 Feature Extraction ..................................... 66
    3.2.3 Gesture Modeling ....................................... 69
    3.2.4 Gesture Learning ....................................... 70
    3.2.5 Gesture Recognition ................................... 73
  3.3 Gesture Domain of The Thesis ................................ 74
    3.3.1 Gesture of “I Have Command” ........................ 75
    3.3.2 Gesture of “All Clear” ................................. 75
3.3.3 Gesture of “Not Clear” ........................................... 75
3.3.4 Gesture of “Spread Wings” ................................... 75
3.3.5 Gesture of “Fold Wings” ........................................ 75
3.3.6 Gesture of “Lock Wings” ....................................... 81
3.3.7 Gesture of “UP Hook” .......................................... 81
3.3.8 Gesture of “Down Hook” ....................................... 81
3.3.9 Gesture of “Remove Chocks” ................................. 81
3.3.10 Gesture of “Insert Chocks” .................................... 81
3.3.11 Gesture of “Move Ahead” ..................................... 81
3.3.12 Gesture of “Turn Left” ........................................ 88
3.3.13 Gesture of “Turn Right” ....................................... 88
3.3.14 Gesture of “Slow Down” ....................................... 88
3.3.15 Gesture of “Stop” .............................................. 88
3.3.16 Gesture of “fire” ............................................ 88
3.3.17 Gesture of “Brakes On” ........................................ 88
3.3.18 Gesture of “Brakes Off” ....................................... 98
3.3.19 Gesture of “Next Marshaler” ................................. 98
3.3.20 Gesture of “Install Tiedown” ................................. 98

3.4 Summary ............................................................. 98

4 Experimental Results Discussions .................................. 99
4.1 Experimental Setup .................................................. 99
4.1.1 Kinect Sensor .................................................... 99
4.1.2 Kinect SDK ..................................................... 100
4.1.3 Programming Environment .................................... 102
4.2 Implementation ...................................................... 103
4.2.1 Learning Module ............................................... 103
4.2.2 Training Module ............................................... 105
4.2.3 Gesture Recognition Module ................................. 106
4.2.4 Parameters for Performance Analysis ....................... 107
4.3 Results of Gesture Recognition ................................... 110
4.3.1 Confusion Matrix ............................................... 110
4.3.2 Precision ........................................................ 113
4.3.3 TP Rate or Recall .............................................. 117
4.3.4 FP Rate .......................................................... 121
4.3.5 F-measure ....................................................... 126
4.3.6 Accuracy ........................................................ 130
4.3.7 $\kappa$-value(Kappa Value) ................................... 131
4.4 Summary ............................................................. 132

5 Conclusion .................................................................. 133
5.1 Contributions ......................................................... 133
5.2 Limitation of the System .......................................... 134
5.3 Future Works ......................................................... 135

References ..................................................................... 137
List of Tables

3.1 Mapping of all joints for a single frame .......................... 59
3.2 Mapped data with joint position translation ......................... 65
3.3 List of feature using joint point position .......................... 67

4.1 Sample Frame of Training Dataset .................................. 106
4.2 Mean Training Error for All Classifiers for Different N ............. 107
4.3 Confusion Matrix for Naive Bayes Classifier: (N = 10) ............. 110
4.4 Confusion Matrix for Neural Network-based Classifier: (N = 10) .... 111
4.5 Confusion Matrix for Support Vector Machine Classifier: (N = 10) .... 112
4.6 Confusion Matrix for Hidden Markov Model Classifier: (N = 10) .... 113
4.7 Comparison of Precision for 20 Gestures .......................... 114
4.8 Precision for Different Value of N ................................ 117
4.9 Recall for Different Gestures ....................................... 118
4.10 Recall for Different Value of N ................................... 121
4.11 FP Rate for Different Gestures .................................... 122
4.12 FP Rate for Different Value of N ................................... 125
4.13 F-measure for Different Gestures .................................. 126
4.14 F-measure for Different Value of N ................................ 129
4.15 Accuracy for Different Values of N ............................... 130
4.16 κ-value(Kappa Value) for N = 10 ................................. 132
4.17 Average Response Time for Different Classifiers ................... 132
## List of Figures

1.1 Overview of the Proposed System. ........................................... 5

2.1 The P5 Data Glove with a 3D Positioning Sensor. ©Rutgers University [1]. ................................................................. 18
2.2 VPL Data Glove [2]. .......................................................... 18
2.3 Different Representation of Gesture. (Courtesy: Mohamed Bécha [3]) . 21
2.4 Gesture Processing Stages. ................................................ 23
2.5 Kinect Sensor for Xbox 360 [4]. ........................................... 28
2.6 Kinect Sensor for Xbox One [4]. .......................................... 28
2.7 Block Diagram of Kinect Sensors [4]. ................................... 30
2.8 Recommend Playable Distance for Kinect [5]. ......................... 31
2.9 Example of NATOPS Gestures: (a) brakes on, (b) brakes off, (c) insert chocks, (d) remove chocks [6]. ................................. 37
2.10 NATOPS Aircraft Handling Signals [6]. .............................. 40
2.11 Architecture of Hidden Markov Model. ................................. 42
2.12 A Probable Artificial Neural Network ................................. 45
2.13 Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. ............................................. 47

3.1 Input-Process-Output Model .................................................. 51
3.2 A step by step model of problem formulation .......................... 52
3.3 Example of RGB image and depth image of a Human .............. 55
3.4 Skeleton on the color image (skeleton in the left image is not aligned with human body and the right one is aligned with the help of coordinate mapper [7]) ................................................... 55
3.5 Skeleton Joints of a Human identified by Kinect for Windows version 2.0 SDK [8] ............................................................. 57
3.6 Example of tracking of a Human in a single frame .................. 58
3.7 The body coordinates system referencing the SpineMid point as origin [9]. ........................................................................... 60
3.8 Kinect camera space coordinate system [4]. ......................... 61
3.9 Geometric translation of coordinate .................................... 62
3.10 The Coordinate System before translation ............................ 63
3.11 The Coordinate System after translation ............................... 64
3.12 Feature extraction in Gesture Recognition model .................. 66
3.13 Model of a gesture with N=8 (part-1) ................................. 71
3.14 Model of a gesture with N=8 (part-2) ................................. 72
4.23 F-measure Graph for different values of N ................. 129
4.24 Accuracy Graph for Different Values of N .................. 130
Chapter 1

Introduction

Gesture-based interaction is a paradigm shifting approach to interacting with computer systems. It allows users to interact with systems by waving their arms, articulating specific body gestures, giving a hand pose, making facial expressions, etc.; skills that most of us have naturally learned and used since birth.

Gesture recognition has become a significant research field with the current focus on interactive emotion recognition and Human Computer Interaction. Research on gesture recognition has many motivations, all of which are related to improving the interface between humans and computers. If a computer can detect and recognize a set of gestures, it can infer the senders message and respond appropriately. For example, a conductor can control a virtual orchestra by gesturing commands to a camera. The system responds by appropriately varying the volume and tempo of the prerecorded music being played.

Moreover Gestures are an important form of human interaction and communication: hands are usually used to interact with things and our body gesticulates to communicate with others. Thus, a wide range of gesture recognition applications has been experienced up to now thanks to a certain level of maturity reached by sub-fields of machine intelligence (Machine learning, Cognitive Vision, Multi-modal monitoring). Human gesture is most naturally expressed with body and hands, ranging from the simple gestures we use in normal conversations to the more elaborate gestures used by cricket umpire giving signal for conducting a match; soldiers gesturing for tactical
tasks; and traffic police giving body and hand signals to drivers. Current technology for gesture understanding is still sharply limited, with body and hand signals typically considered separately, restricting the expressiveness of the gesture vocabulary and making interaction less natural [6].

However, shifting from the controller-based interaction to the gesture-based interaction is getting growing attention in many research and industrial fields. Gesture recognition allows people to use their own body parts to give commands via a sequence of articulated body poses. Therefore, in order to control a television or play an interactive video game, users no longer need to sit in front of a system holding a remote control or a joystick; they can simply wave their arms or hands, or hold a specific body pose in a predefined manner.

With the recent invention of depth sensors, human gesture recognition has gained significant interest in the fields of computer vision and human computer interaction. Robust gesture recognition is a difficult problem because of the spatiotemporal variations in gesture formation, subject size and subject location in the frame, image fidelity, and subject occlusion. Gesture boundary detection, or the automatic detection of the beginning and the end of a gesture in a sequence of gestures, is critical toward achieving robust gesture recognition. This thesis attempts to help bridge the visual communication gap between computers and humans by designing and implementing a vision-based gesture recognition system.

The thesis proposes computer vision-based real-time 3D full-body motion gesture recognition for aircraft handling. The output of the research can be used for understanding the Naval Air Training and Operating Procedures Standardization (NATOPS) Aircraft Handling signals[10] which is a multi-signal gesture database performed by using both body and hand movements. This database includes 20 body-and-hand gestures and also focuses on a clearly defined gesture vocabulary from a real world scenario.

The motivation behind conducting this thesis is presented in section 1.1. Section 1.2 states the objective of the thesis. Section 1.3 presents an overview of the proposed system. Section 1.4 states contributions of this work in this field of research. Section 1.5 gives the outline for the rest of the document.
1.1 Motivation

Human gesture recognition is an active and challenging research topic. A successful gesture recognition technology has a great potential in many application areas, including visual surveillance, virtual reality, home appliances, human-computer interaction (HCI), and analysis of sign languages, gaming, and robotic control.

The role of gesture on its own and the expressiveness it adds when used with verbal interaction, suggests that gesture recognition systems have the potential to open a new and effective communication channel in human-computer interaction. Gesture recognition systems identify human gestures and the information they convey. Gesture recognition systems are in some cases the optimal choice for the user and can potentially revolutionize some problem domains.

Controller-based interaction (e.g., keyboard, mouse, joystick, etc.) has been a primary way to deal with computer systems. It has allowed us to interact with complex systems efficiently, especially when one needs fine control of a system. However Controller-based interaction is often both labor intensive (e.g., getting familiar with input devices and learning all the provided functionalities of the system) and requires an awkward interaction modality or an unintuitive definition of a functionality.

On the other side, Close proximity sensors like data glove[11, 12] often facilitate detection of hand configuration and movement. But the major disadvantage of the system is that additional hardware is required which makes intrusive approaches less attractive and troublesome. Moreover the system is limited to recognizing only hand and finger movements.

Increases in computer processing power and the miniaturization of sensors have increased the possibilities of varied, novel inputs in HCI. Gestures input is one of the important way in which users can communicate with machines, and such a communication interface can be even more intuitive and effective than traditional mouse and keyboard, or even touch interfaces. Just as human gesture when they speak or react to their environment, ignoring gestures will result in a potent loss of information.

The use of gesture as a natural interface serves as a motivating force for research in modeling, analyzing and recognition of gestures. In particular human computer...
intelligent interaction needs vision-based gesture recognition[13].

Due to the rapid development of computer vision and machine learning, recognizing human actions through image or motion sensors has gained increasing popularity. Gesture-based user interfaces that are carefully designed to resemble how humans naturally interact with objects in the real world, therefore, can open a new horizon of future computing.

However, many of the previous approaches to gesture recognition consider either body pose or hand pose alone, limiting their practicality for many real-world problems. In this work, we combine body and hand poses, allowing gesture recognition to deal with a richer gesture vocabulary, extending its practicality.

1.2 Objectives

The objective of this thesis work is to design and implement a real-time 3D full-body motion gesture recognition system that attends to multiple information channels, specifically a combination of body and hand poses. At the same time, to avoid obtrusive and unnatural interaction, the system was built not to require any marker to be attached to the human body, but to perform motion tracking solely based on a single Kinect sensor along with various computer vision techniques[14].

In the real world scenarios, gestures are continuous in nature, without any explicit pause or break between individual gestures. A real-time gesture recognition has the challenge of spotting and recognizing a gesture in continuous stream [15]. Thus the recognition of such gestures closely depends on the segmentation; i.e. determining start and end frame of each gesture in a continuous stream of gestures [16].

Our system performs 3D body and hand pose estimation and classification in a unified fashion, and uses results to recognize gestures. The gestures that we intend to recognize are motion gestures. Motion gesture is a sequence of meaningful movements of hands, arms, heads, legs etc. To recognize motion gestures, we use a vision-based approach, which is the natural way of interaction. The thesis also focuses on exploring the ability of Kinect sensor of human action recognition. The proposed method uses skeleton joints along with motion trajectory as features for classifying gesture. Finally
some classification algorithms (Naive Bayesian Classifier, Artificial Neural Network, Hidden Markov Model and Support Vector Machine) used to recognize gesture, making the system useful for real-time application.

We aim to test the system using gesture database: the NATOPS [10] aircraft handling signals [6]. The database includes 20 gestures and also focuses on a clearly defined gesture vocabulary from a real world scenario. These gestures also includes a whole variety of motion gestures which utilizes hands, fingers, arms etc. of the human body.

1.3 Overview of the Proposed System

In this thesis a computer vision-based real-time 3D full-body motion gesture recognition system is proposed. Our system will perform 3D upper body pose estimation along with hand pose classification and finally gesture recognition. It will be evaluated on the scenario like an aircraft carrier flight deck environment, where humans interact with unmanned vehicles using existing body and hand gesture vocabulary. Figure 1.1 shows overview of the proposed system. Implementation of the proposed gesture recogn-

![Figure 1.1: Overview of the Proposed System.](image)

ation system is spanned around skeleton tracking and gesture recognition functions. Kinect for Windows version 2.0 SDK used to detect the body joint points. The SDK
supports all standard inputs and in addition includes a Skeletal Tracking. Total 25 human body joint is detected by Kinect sensor along with the SDK. The joints locations are actually coordinates relative to the sensor and values of X, Y, Z coordinates are in meters. The skeletal tracking allows the Kinect to recognize people and follow their actions [21]. In the next step, frame by frame skeleton joint-point coordinates are collected and required joint-points are saved. From the joint point coordinates a Kinect Skeletal Coordinate System (KSCS) is generated. At the same time, Kinect skeletal joint points are scaled by a scale factor to get robust data.

The general technique to tackle the gesture recognition problem is to deal with it as a pattern recognition problem where a set of features are extracted from data captured that in turn are matched (i.e., Classification) to a predefined representation of the gesture (i.e., pattern). Gestures are modeled using sequence of frames. The size of the sequence is a predefined fixed integer. The predefined gestures are performed by volunteers to prepare the training data set. This training data set is used to train pattern classification algorithms such as Naive Bayes, Neural Network, Hidden Markov Model and Support Vector Machine. The system is trained using almost 1500 instances of gestures. 15 volunteers performed each of the 20 gestures 5 times.

In the testing stage, a sequence of frame is served from the continuous observing frames and classified by an appropriate classifier such as Naive Bayes, Hidden Markov Model, Support Vector Machine and Neural Network-based classifiers. These classifiers predict the class of the predefined gesture or declare it as unknown gesture.

### 1.4 Contributions

This thesis presents and develops a computer vision based real-time 3D motion gesture recognition system consisting multiple signals that follows the movement of bodies and hands including fingers. The output of this research can be used for interpreting the Naval Air Training and Operating Procedures Standardization (NATOPS) aircraft handling signals [10]. There are previous work in this field by Song et al. Those works uses a stereo camera to collect 3D images considering body pose estimation and hand pose classification individually where real-time tracking ability was not considered. In
this research a real-time gesture recognition approach is developed which is a natural way of interaction.

For 3D body pose estimation, the system used Kinect 3D joint points and calculated motion trajectory for accuracy. Image frame sequences are used to represent motion gestures, where each frame is mapped to a matrix of 3D (X,Y,Z) coordinates of 25 body joints of a human. A Kinect Skeletal Coordinate System (KSCS) is established with the mapped data and the joint points are scaled for developing a system which is invariant of person’s distance from the sensor, height etc. Motion trajectory along with the body joint coordinates used as a feature to recognize 3D motion gesture.

Four pattern classification algorithms such as Naive Bayes, Neural Network, Hidden Markov Model and Support Vector Machine has been used to recognize. The accuracy of the system is 93.7% in Nave Bayes classifier, 98.1% in Support Vector Machine, 97% in Neural Network and 90.8% in Hidden Markov Model. Considering the accuracy and complexity of the system with respect to the number of frame(N), the best response time of the system is 16.1 ms in Naive Bayes, 29.5 ms in Neural Network, 27.1 ms in Support Vector Machine, 32.3 ms in Hidden Markov Model. To test the consistency of output of all the four classifiers kappa-test has been applied to the system and the result was satisfactory with kappa value 0.91.

The system is evaluated on a real-world scenario: we tested the performance of this gesture recognition system is evaluated with a subset of the NATOPS aircraft handling signals, a challenging gesture vocabulary that involves both body and hand pose articulations. Experimental result show that combining body and hand poses significantly improved the gesture recognition accuracy.

1.5 Outline of the Thesis

Chapter 2 reviews previous works in gesture recognition considering appearance based and 3D model based approaches. It also focuses on gesture recognition using kinect sensor and gesture recognition in aircraft handling. It describes pattern classification methods used in this proposed system. It describes a several pattern classification methods used in related research.
Chapter 3 describes the proposed methodologies for the proposed gesture recognition system.

Chapter 4 discusses the experimental setup, implementations of the system, result and related discussion. It also analyzes the performance of the proposed system.

Chapter 5 concludes this research work by mentioning the limitations of this system and future scope to work.
Chapter 2

Literature Review

Gesture recognition has been a prominent domain of research since the last three decades as a major fulfillment of machine intelligence.

Gestures are an important form of human interaction and communication. With gesture recognition technique we can mimic the communications between human, and involve gesture as a natural and intuitive way to interact with machines.

Gesture is a non-vocal communication method that can be used to carry information to a viewer. Gestures used for communicating between human and machines as well as between people using sign language. A wide range of gesture recognition applications has been experienced up to now thanks to a certain level of maturity reached by sub-fields of machine intelligence (Machine learning, Cognitive Vision, Multi-modal monitoring).

From the advancement in computer system, the definition of intelligent system has become obsolete. Systems that interact and communicate in the real world need sophisticated visual capabilities for orientation, recognition, understanding, reference resolution, exploration, and more. Even the simplest verbal communication is an inherent multimodal affair typically coming along with a facial expression or gesture.

In this chapter a comprehensive background on gesture and gesture recognition methods are presented. Section 2.1 presents definitions and descriptions of some keywords that are fundamental to the context of gesture recognition. Section 2.2 describes various approaches used in researches to recognize gestures. Section 2.3 briefly de-
scribes about classification methods used in the proposed system.

2.1 Definitions and Descriptions

This thesis presents a real-time computer vision-based 3D motion gesture recognition system for aircraft handling. To understand the background of the work, basic concept of gestures, hand posture, human-computer interaction, gesture recognition and other related research topics must be known in advance. This section presents brief knowledge about this topics.

2.1.1 Gesture

To gain insight into gesture recognition, it is important to understand the nature of gestures. We emphasize the importance of gestures in communication, as often, gestures not only communicate, they also help the speaker formulate coherent speech by aiding in the retrieval of elusive words from lexical memory. Gestures are expressive, meaningful body motions involving physical movements of the fingers, hands, arms, head, face, or body with the intent of conveying meaningful information or interacting with the environment.

Gesture acts a medium of communication for non-vocal communication in conjunction with or without verbal communication is intended to express meaningful commands. These gestures may be articulated with any of the body parts or with combination of one or many of them.

Gesture is an action, which consists of a sequence of body postures. Gestures are restricted to a discrete set of postures recognized in static form. Gestures being major constituent of human communication may serve as an important means for human computer interaction too.

Bobick and Wilson have defined gestures as the motion of the body that is intended to communicate with other agents[17]. For a successful communication, a sender and a receiver must have the same set of information for a particular gesture. Gesture can be defined as an expressive movement of body parts which has a particular message,
to be communicated precisely between a sender and a receiver. Gestures can be static (posture or certain pose) which require less computational complexity [18] or dynamic (sequence of postures) which are more complex but suitable for real-time environments [18][19]. Some gesture also have both static and dynamic elements as in sign languages.

Gesture recognition can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse. The recognition of natural gestures requires their temporal segmentation. Often one needs to specify the start and end points of a gesture in terms of the frames of movement, both in time and space. Sometimes a gesture is also affected by the context of preceding as well as following gestures.

2.1.2 Hand Posture

Static morphs of the hand are called hand postures. Hand gestures are a collection of movements of the hand and arm that vary from the static posture of pointing at something to the dynamic ones used to communicate with others. Recognizing these hand movements needs modeling them in the spatial and temporal domains.

Hand posture is the static structure of the hand while its dynamic movement is called hand gesture and both are particularly crucial for human-computer interaction. The methods used for understanding these structures and movements are among the most classifying researches that still in progress.

The information related with hand gestures during a talk has a spatial as well as a temporal structure. The gesture recognition methods are primarily divided into Data-Glove Based and Vision Based methods. The Data-Glove based approaches use mechanical or optical sensors connected to a glove that converts finger flexions into electrical signals for recognizing the hand posture. In [11, 12], gloves or finger marks have been used to extract the hand posture information from the image and facilitate the hand segmentation process by removing the varying skin color issue.

Even though hand postures and gestures are frequently considered as being identi-
cal, there are actually differences [20, 21]. While the hand posture is a static motionless pose, such as making a palm posture and holding it in a certain position, the hand gesture is a dynamic process consisting of a sequence of changing hand postures over a short duration, as for instance waving the hand. In addition to the motion-based features, most research focuses on the recognition of hand postures, which is, extracting features directly from the hands.

### 2.1.3 Human-Computer Interaction

Human computer interaction (HCI) also named Man-Machine Interaction (MMI) [22] refers to the relation between the human and the computer or more precisely the machine [22]. Since the machine is insignificant without suitable utilize by the human, HCI is an important field of research. There are two main characteristics should be deemed when designing a HCI system: functionality and usability [22]. System functionality referred to the set of functions or services that the system equips to the users [22], while system usability referred to the level and scope that the system can operate and perform specific user purposes efficiently [22]. The system that attains a suitable balance between these concepts considered as influential performance and powerful system.

Majority of the human-computer interaction (HCI) is based on mechanical devices such as keyboard, mouse, joysticks or gamepads. Special input and output devices have been designed over the years with the purpose of easing the communication between computers and humans. Every new device can be seen as an attempt to make the computer more intelligent and making humans able to perform more complicated communication with the computer. This has been possible due to the result oriented efforts made by computer professionals for creating successful human computer interfaces. As the complexities of human needs have turned into many folds and continues to grow so, the need for complex programming ability and intuitiveness are critical attributes of computer programmers to survive in a competitive environment. The computer programmers have been incredibly successful in easing the communication between computers and human. With the emergence of every new product in the
market attempts to ease the complexity of jobs performed.

Earlier, Computer programmers were avoiding such kind of complex programs as the focus was more on speed than other modifiable features. However, a shift towards a user friendly Environment has driven them to revisit the focus area. The idea is to make computers understand human language and develop a user friendly human computer interfaces (HCI). Making a computer understand speech, facial expressions and human gestures are some steps towards it. In recent years there has been a growing interest in a class of methods based on computational vision due to its ability to recognize human gestures in a natural way [23]. Gestures are the non-verbally exchanged information. A person can perform innumerable gestures at a time. Since human gestures are perceived through vision, it is a subject of great interest for computer vision researchers. Determining human gestures by creating an HCI as well as coding of these gestures into machine language demands a complex programming algorithm.

In the present world, the interaction with the computing devices has advanced to such an extent that as humans it has become necessity and we cannot live without it. The technology has become so embedded into our daily lives that we use it to work, shop, communicate and even entertain our self. It has been widely believed that the computing, communication and display technologies progress further, but the existing techniques may become a bottleneck in the effective utilization of the available information flow. To efficiently use them; most computer applications require more and more interaction. For that reason, human computer interaction (HCI) has been a lively field of research in the last few years. The interaction consists of the direct manipulation of graphic objects such as icons and windows using a pointing device. Even if the invention of keyboard and mouse is a great progress, there are still situations in which these devices are incompatible for HCI.

2.1.4 Human-Robot Interaction

Human-Robot Interaction (HRI) is a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans[24]. The HRI problem is to understand and shape the interactions between one or more humans and one or
more robots[24]. A significant topic of interest for gesture interaction systems is human-robot interaction with a huge number of published papers [25, 26, 27, 28, 29]. Gesture recognition is a significant feature for fixed and portable robots [30]. In [31], robotic manipulators and human users were integrated with hand gesture commands for recognizing four postures such as when a user points at an object, the robot can detect it. In [32], a human-machine interaction system for game playing allows the visual recognition of three postures with different rotation angles and scales with 95% accuracy. In [33], a robotic-assistant interaction application was designed by combining gesture recognition and voice to generate commands. Once the hand posture is detected, six gestures are trained using a hand contour as the major feature of each gesture. In [34], a human-robot system was designed to recognize dynamic gestures by tracking head and arm direction. The system employs Hidden Markov Model (HMM) to recognize trajectories of the detected hands. In [35], a programming-by-demonstration method allows the robot to learn a human users gestures. The system employs eight hand postures to control a hybrid service robot system. In [36], a programming-by-demonstration idea was also utilized to allow users to help the robot make gestures by kinesthetic teaching. The user trains the robot to make 10 dynamic gestures captured by sensors connected to the users torso, upper- and lower-arm.

Thus, Human-Robot Interaction (HRI) is a relatively young discipline that has attracted a lot of attention over the past few years due to the increasing availability of complex robots and people’s exposure to such robots in their daily lives, e.g. as robotic toys or, to some extent, as household appliances (robotic vacuum cleaners or lawn movers). Also, robots are increasingly being developed for real world application areas, such as robots in rehabilitation, eldercare, or robots used in robot-assisted therapy and other assistive or educational applications.

2.2 Gesture Recognition

Gesture recognition is an alternative user interface for providing real-time data to a computer. Instead of typing with keys or tapping on a touch screen, a motion sensor perceives and interprets movements as the primary source of data input.
Gesture recognition is a broad term that is increasingly used in natural human-computer interaction research. To gain insight into gesture recognition, it is important to understand the nature of gestures. The meanings of gestures can range from dynamic human body motion [37, 38] through pointing device gestures [39, 40] to sign language[41]. Different beings also gesticulate differently, therefore increasing the difficulty of gesture recognition. Gestures differ both temporally, and spatially. Gestures are ambiguous and incompletely specified, and hence, machine recognition of gestures is non-trivial.

Gesture recognition is a type of perceptual computing user interface that allows machines to capture and interpret human gestures as commands. The general definition of gesture recognition is the ability of a machine to understand gestures and execute commands based on those gestures. Gesture recognition allows people to use their own body parts to give commands via a sequence of articulated body poses. Research on gesture recognition has many motivations, all of which are related to improving the interface between humans and computers. If a computer can detect and recognize a set of gestures, it can infer the senders message and respond appropriately. For example, a conductor can control a ”virtual orchestra” by gesturing commands to a video camera or a Kinect sensor. The system responds by appropriately varying the volume and tempo of the prerecorded music being played. As another example, a system can annotate video clips of athletic events with meaningful descriptions. When requested for an example of a triple salchow, another system responds by quickly finding the figure skating jump among an annotated database of video sequences. As a final example, a karate instruction system can visually evaluate the performance of a students kick.

There are two categories of gesture recognition, isolated recognition and continuous recognition [16]. Isolated gesture recognition is based on the assumption that each gesture can be individually extracted in time, from the beginning to end of gesture [15]. Continuous recognition has the additional challenge of spotting and recognizing a gesture in continuous stream [15]. In the real world scenarios, gestures are continuous in nature, without any explicit pause or break between individual gestures. Thus the recognition of such gestures closely depends on the segmentation; i.e. determining start and end frame of each gesture in a continuous stream of gestures [16]. Spatiotem-
poral variability [42] refers to dynamic variations in movement and duration, even for same gesture. The intermediate motion between two consecutive gestures is termed as transitional motion.

Many researchers have proposed various techniques for gesture recognition systems. Usually, these systems are divided into two techniques, namely the glove-based and respectively the vision-based methods. When data gloves are used to track hand gestures, the user has to wear burdensome and expensive data gloves to track hand and finger movements. Vision-based methods can detect, track and recognize gestures more naturally and efficiently.

2.2.1 Glove-based Approaches

The Data-Glove based approaches utilize sensor devices for digitizing hand and finger movements into multi-parametric data. This additional sensors facilitate detection of hand configuration and movement.

Close proximity sensors like data glove often provide high quality and accurate information. However, they tend to impose additional time to prepare for usage, and may restraint natural behavior while using the system. Natural behavior could be restrained as these devices often have wires attached to them, and even holding an object may change the way a person moves. In addition, this sort of equipment is usually rather expensive as it is produced to perform very specific tasks.

In glove based analysis, detection of the hand is eliminated by the sensors on the hand and 3D model of the hand is easily subjected to the virtual world and analysis comes next. Such systems are optimal for body motion capture purposes and widely used in industry.

Early approaches to gesture recognition are sensor-based and rely on measurements from position sensors and data gloves. With these, detailed measurements of the body position and of the hand shape can be obtained, including flexing angles of the joint-points, as well as posture of the hands. Due to these exact measurements, sensor-based methods can cope with larger vocabulary sizes (i.e. the number of different signs) while reporting higher classification accuracies than vision-based methods.
There exists a variety of data gloves [1, 43, 44, 45] which provide real time information about a hands current configuration. These data gloves consist of a glove covered with sensors, typically at the joints of each finger. Some of these systems also provide information about the placement and orientation of the hand, and some rely on a supporting system for this kind of information. As finger joints angles are measured directly, extraction of measurements requires low software complexity and calculation imposes small overhead. In addition the measurements are generally of high quality and the measurement frequency is high.

Data gloves, however, are generally quite expensive with the P5 glove as a notable exception [1]. Furthermore, as these sort of devices typically are connected to a computer by cables they might be cumbersome to put on and, more importantly, may hinder natural movement [19]. If full body gestures are necessary, a data suit [46] could be used. This is an extension of data gloves, which provides measurements of multiple limbs at once. As a result, information about the whole body configuration is measured. According to Goto and Yamasaki a performer wears this suit, but doesn’t hold a controller in his hands. Therefore, his gesture could be liberated to become a larger gesture, like a mime. [46]

Christopher Lee and Yangsheng Xu [47] developed a glove-based gesture recognition system that was able to recognize 14 of the letters from the hand alphabet, learn new gestures and able to update the model of each gesture in the system in online mode, with a rate of 10Hz. Over the years advanced glove devices have been designed such as the Sayre Glove, Dexterous Hand Master and PowerGlove [48]. The most successful commercially available glove is by far the VPL DataGlove as shown in figure 2.2. It was developed by Zimmerman [2] during the 1970s. It is based upon patented optical fiber sensors along the back of the fingers. Star-ner and Pentland [49] developed a glove-environment system capable of recognizing 40 signs from the American Sign Language (ASL) with a rate of 5Hz.

Murakami and Taguchi [50] distinguish between 42 isolated gestures from Japanese finger spelling. A neural network classifier is trained and an accuracy of 98% reported. This accuracy drops to 72% in a signer-independent setting. Kim [51] adopt a similar approach wherein they test their system on 25 different Korean Sign Language gestures;
and Fels and Hinton [52] which recognize 66 different hand shapes derived mainly from American Sign Language. A simple nearest neighbor classifier is used by Kadous [51] to distinguish between 95 signs. The authors report classification accuracies of 80 percent, which drops to 15 percent in a signer independent scenario. In more recent work,
Hidden Markov Models (HMM) gained increased popularity. Liang and Ouhyoung [53] use a lexicon of 250 signs where each sign is represented in a viseme-like way using hand shape, hand position, hand orientation and hand motion. Wang et al. [54] use a significantly larger vocabulary of 5119 signs from Chinese sign language. The authors manually identify 2439 shared sub-words in this vocabulary and train a HMM for each sub-word. With these subword recognizers, sign recognition is performed for the 5119 signs with an accuracy of 92.8%. While these results are astonishing for such a large database, providing ground truth for each of the subwords is very time-consuming. For this reason, Fang et al. [55] use a data-driven approach to automatically segment the 5119 signs into 238 sub-words and then train sub-word classifiers to perform sign recognition. Sub-word classifiers are then trained and classification results of 90.5% reported for sub-word recognition.

However, the sensor devices are quite expensive and wearing gloves or trackers is uncomfortable and enlarges the time-to interface or setup time. The major disadvantage though is that additional hardware is required which makes sensor-based approaches less attractive for commercial products.

These devices are not precise and have low resolution but they are useful for environment that lacks light and has magnetic obstacles or noises. On the other hand, vision-based methods are more natural and useful for real-time applications.

2.2.2 Vision-based Approach

When systems cannot rely on attached or held sensors, information about the user must necessarily be obtained from a sensor at some distance from the user. This approach gives users freedom to move naturally, as they are not bound by cables or inhibited by potentially heavy equipment.

Several Vision based gesture recognition models have been proposed to abstract and model human body parts motion. We can split the models in two kinds according to the spatial and temporal aspects of gestures: (1) posture automaton models in which the spatial and the temporal aspects are modeled separately and (2) motion models in which there is a unique spatial-temporal model.
Wu and Huang [13] in their survey have identified four gesture categories for different application scenarios: conversational, controlling, manipulative and communicative. These gestures can be represented by temporal movements and static postures as two separate input channels or a mixed channel of data. Postures express certain concepts through body pose or hand configuration while temporal gestures represent certain actions of body movements. An applications constraints may further enforce specific requirements that may limit the selection of technologies and methods. For instance, hand and face tracking for human-computer interaction requires applying fast image processing techniques while other applications like hand or body posture recognition have more relaxed constraints in terms of required processing power. Availability of the required hardware and robustness against environmental noise are other constrain which may dictated by the application.

Vision based gesture recognition system is able to provide the required functions for recognition of static or temporally sequenced patterns of hand, arm, head, body and other body parts from an image sequence. In this context, a general purpose vision based gesture recognition system can be decomposed to three distinctive sub-systems: 1) detection 2) tracking and 3) recognition. The role of the detection sub-system is finding the object of interest within the image (i.e. the hand, face, or body). In other words a set of features is extracted from every frame captured. Since gestures are a dynamic sequence of postures connected though continuous movements, a classifier can be trained against a possible grammar. Moreover, the tracking sub-system indicates the movement trajectory of the interest within a sequence of images and collects the necessary information for the recognition sub-system. The recognition system interprets the gesture meaning by finding a known pattern in the information collected by the other two sub-systems.

Vision based approaches rely on one or more cameras to detect and analyze body movement from the video sequences. There are different camera sensors:

- Infrared camera: can detect movement with no light condition.
- Monocular camera: is the most popular and cheap camera. Typically monocular cameras are used with sensor markers. Markers can be passive if don’t emit light
or active otherwise.

- Stereo cameras: stereovision can detect 3D model of the objects from the scene allowing object movement detection in three dimensions.

- PTZ cameras: pan-tilt-zoom camera embodies a robotic movement engine that enables the movement along three axis.

There are mainly two categories for vision based gesture recognition, namely the three dimensional (3D) model-based methods and, respectively, the appearance-based methods[56]. Figure 2.3 overviews the different representations of gestures.

![Figure 2.3: Different Representation of Gesture. (Courtesy: Mohamed Bécha [3])]({})

### 2.2.2.1 3D Model-based Methods

A 3D model defines the 3D spatial description of the human body parts. The temporal aspect is generally handled by an automaton which generally divides the gesture time into 3 phases [57]: (1) the preparation or pre-stroke phase, (2) the nucleus or stroke phase and (3) the retraction or post-stroke phase. Each phase can be represented as one or several transitions between the spatial states of the 3D human model.

The main advantage of 3D model based methods is to recognize gestures by synthesis: during the recognition process, one or more cameras are looking at the real
target and then compute the parameters of the model that matches spatially the real
target and then follows the latter motion (i.e. update the model parameters and check
whether it matches a transition in the temporal model). Thus, the gesture recognition
is generally precise (specially the start and the end time of the gesture). However,
these methods tend to be computationally expensive unless implemented directly in
dedicated hardware. Some methods [58] combine silhouette extraction with 3D model
projection fitting by finding the target self-orientation. Generally, three kinds of model
are usually used:

- Textured kinematic/volumetric model: these models contain very high details of
  the human body: skeleton and skin surface information.
- 3D geometric model: these models are less precise than the formers in terms of
  skin information but still contain essentially skeleton information.
- 3D skeleton model: these are the most common 3D models due to their simplicity
  and higher adaptability: The skeleton contains only the information about the
  articulations and their 3D degree of freedom (DoF).

Many approaches that use the 3D model based technique [59, 60, 61, 62] depends on the
3D kinematic model with significant degrees of freedom and calculate the parameters
by comparing the input frames and the 2D appearance projected by the 3D model
of the body. This will be suitable for realistic interactions in virtual environments.
The 3D model based technique provides a rich description that permits a wide class
of gestures. However, since the 3D models are articulated deformable objects with
many DOFs, a large image database is required to deal with the entire characteristic
shapes under several views. This method has many drawbacks that have prevented it
from practical use. first, for every image the initial parameters must be close to the
solution; otherwise, the method is likely to find a suboptimal solution such as local
minima. Second, the fitting process is sensitive to noise such as lens aberrations and
sensor noise in the imaging process. The final drawback is the difficulty of feature
extraction and inability to handle singularities that occur from unclear views.
2.2.2.2 Appearance-based Methods

Appearance-based techniques extract image features to model the visual appearance of the body pose and compare these features with the extracted features from the video frames using a pattern classification module. Appearance-based techniques are generally known as an instance of the general problem of pattern recognition, which includes three tasks:

1. Feature Extraction.
2. Classifier (learning to use training samples).
3. Classification of unknown samples.

A large number of gesture recognition methods fall in the system of the block diagram given in figure 2.1, such as [63, 64]. These systems use a module for feature extraction such as a feature vector and a module for the classification of the set of N buffered feature vectors.

![Image](image.png)

Figure 2.4: Gesture Processing Stages.

Appearance based approaches have real-time performance because of the simpler 2D image features are used. A simple method consisting of searching for skin colored regions in the image, was used in [65]. However, this method has some limitations; first, it is highly sensitive to lighting conditions. Secondly, it is assumed that there are no other skin like objects within the image.

In [54], scale-space color features were used to recognize hand gestures, which are based on feature detection and user independence. However, the system shows real-
time performance only when no other skin colored objects exist in the image. The authors of [66] obtained a clear-cut and integrated hand contour to recognize hand gestures, and then computed the curvature of each point on the contour. Due to noise and unstable illumination in the cluttered background, the segmentation of the integrated hand contour had some difficulty. Eigen space approaches can provide an effective illustration of a large group of high dimensional points using a small group of basis vectors. However, eigen space methods are not invariant to translation, scaling, and rotation.

Concerning appearance-based methods, two main sub-categories exist: (1) 2D static model based methods and (2) motion-based methods. Each sub-category contains several variants. For instance, the most used 2D models are:

- **Color-based models:** methods with this kind of model use generally body markers to track the motion of the body or the body part. For example, Bretzner et al. proposed a method for gesture recognition using multi-scale color features, hierarchical models and particle filtering [54].

- **Silhouette geometry based models:** such models may include several geometric properties of the silhouette such as perimeter, convexity, surface, compacity, bounding box/ellipse, elongation, rectangularity, centroid and orientation. Birdal et al. used the geometric properties of the bounding box of the skin to recognize gestures [67].

- **Deformable gabarit based models:** they are generally based on deformable active contours (i.e. snake parameterized with motion and their variants [68]. Ju et al. used snakes for the analysis of gestures and actions in technical talks for video indexing[69]).

For motion models, we can split them in two variants:

- **Global motion descriptor:** Yilmaz et al. have proposed to encode an action by an action sketch” extracted from a silhouette motion volume obtained by stacking a sequence of tracked 2D silhouettes[70]. The action sketch” is composed of a collection of differential geometric properties (e.g. peak surface, pit surface,
ridge surface) of the silhouette motion volume. For recognizing an action, the authors use a learning approach based on a distance and epipolar geometrical transformation for viewpoint changes. Lu et al. propose to recognize gestures via maximum likelihood estimation with hidden Markov models and a global HOG descriptor computed over the whole body[71]. The authors extend their method in [71] by reducing the global descriptor size with principal component analysis. Gorelick et al. extract space-time saliency, space-time orientations and weighted moments from the silhouette motion volume [72]. Gesture classification is performed using nearest neighbors algorithm and Euclidean distance. Recently, Calderara et al. introduce action signatures[73]. An action signature is a 1D sequence of angles, forming a trajectory, which are extracted from a 2D map of adjusted orientation of the gradients of the motion history image. A similarity measure is used for clustering and classification. As these methods are using global motion, they depend on the segmentation quality of the silhouette which influences the robustness of the classification. Furthermore, local motion, which can help to discriminate similar gestures, can easily get lost with a noisy video sequence or with repetitive self-occlusion.

- Local motion descriptor: local motion based methods overcome these limits by considering sparse and local spatial-temporal descriptors more robust to brief occlusions and to noise. For instance Scovanner et al. proposed a 3-D (2D+time)SIFT descriptor and apply it to action recognition using the bag of word paradigm[74]. Schuldt et al. proposed to use Support Vector Machine classifier with local space-time interest points for gesture categorization[75]. Luo et al. introduced local motion histograms and use an Adaboost framework for learning action models[76]. More recently, Liu et al. apply Support Vector Machine learning on correlogram and spatial temporal pyramid extracted from a set of video-word clusters of 3D interest points[77].

These methods are generally not robust enough since the temporal local windows (with short size and fixed spatial position) do not model the exact local motion but several slices of that motion instead. So, the spatial position is not fixed and the resulting
trajectory of the local descriptor represents faithfully the local motion. The generated descriptors are used for gesture learning-clustering using the bag of word paradigm. Thus, the advantages of global and local gesture descriptors are combined to improve the quality of recognition.

There have been a number of research efforts recently focusing on local invariant features [78, 79, 80]. In [78], Adaboost learning algorithm and SIFT features were used to achieve in-plane rotation invariant hand detection. In addition, a sharing feature concept was used to speed up the testing process and increase the recognition accuracy. Therefore, efficiency of 97.8% was achieved. However, several features such as a contrasted context histogram had to be used to achieve gesture recognition in real-time.

2.2.3 Kinect-based Approach

The apparition of reasonably inexpensive image and depth sensors has encouraged researchers in the area of object detection, tracking and gesture recognition. The Kinect based approach has improved the user interactivity, user comfort, system robustness, and ease of deployment.

Microsoft Kinect is a motion sensing input devices which initially released for the Microsoft Xbox 360 console gaming system. Kinect for Xbox One is the latest version of Kinect From Microsoft. It can recognize the whole body and build an avatar of the player, so the player can play full body games without any controller. Nowadays, Kinect has been widely used to replace RGB cameras to provide new opportunities in many applications.

Prior to the Kinect, to capture accurate 3D depth data of a scene, a 3D laser scanner was the main device to be applied in non-contact measuring situations. However, the massive volume and price of the laser scanner limit the potential usage in many applications. Instead a stereo vision system consisting of two cameras is often employed to get 3D information, for example in robotics [81, 82]. Nevertheless, the resolution of the cameras, the calibration of the system and the required heavy computations increase the complexity of the system and significantly influence the accuracy of the
3D depth data.

Figure 2.5 and 2.6 shows the physical appearance and sensor components of the first generation and 2nd generation of the Kinect device respectively. The Kinect sensor contains a depth sensor, a color camera, and a four-microphone array that provide full-body 3D skeleton tracking, face recognition, and voice recognition capabilities [83]. The color sensor is a RGB camera. An infrared (IR) emitter emits infrared light beams, and the reflected IR beams from the environment come back to the IR depth sensor. The distances between different objects and the sensor are obtained based on the reflected beams. A multi-array microphone containing four microphones can be used for capturing sound. The tilt motor is capable of vertically tilting the sensor bar with a range of $\pm 27^\circ$. The first generation of the sensor uses structured light for depth sensing. The color stream has a resolution of $640 \times 480 \text{px}$ at $30\text{Hz}$. The depth sensing video stream has a resolution of $320 \times 240 \text{px}$ at $30\text{Hz}$. With the default range, the Kinect sensor (version one) has a depth sensing range limit of about $0.8 - 4 \text{m}^2$. The random error of its depth measurements increases quadratically with increasing distance from the sensor, and ranges from a few millimeters at $0.5\text{m}$ distance to about $4\text{cm}$ at the maximum range of $5\text{m}$ [84]. The angular field of view of the sensor is $43^\circ$ vertically and $57^\circ$ horizontally. It can track 20 body joints including hands. The second generation Kinect has the color stream of resolution of $1920 \times 1080 \text{px}$ at $30\text{Hz}$. The depth sensing video stream has a resolution of $512 \times 424 \text{px}$ at $30\text{Hz}$. The Kinect sensor (version two) has a depth sensing about $4.5\text{m}$. The angular field of view of the sensor is $60^\circ$ vertically and $70^\circ$ horizontally. Kinect for Xbox One can track 25 body joints including hands.

In addition to the depth information provided by Kinect, another data modality is also often provided the skeleton model. Comparing to extracting the skeleton model from RGB videos, the depth information makes the extraction more feasible and stable. Several algorithms have been proposed and applied to extract the skeleton from the depth data [85, 86, 87]. The basic idea underlying these methods is to segment the human depth image into multiple body parts with dense probabilistic labeling. This segmentation of the body parts can be considered as a classification task for each pixel in the depth image. The 3D joint positions are computed based on the spatial modes
of the inferred per-pixel distribution. Due to the characteristics of the algorithms, the skeletal tracking is optimized for when the user is facing the Kinect. Sideways poses with parts of the user invisible to the Kinect make the skeletal tracking challenging.

Kinect can recognize up to six users in the field of view of the sensor. Of these, up to two users can be tracked in detail. An application can locate the joints of the tracked users in space and track their movements overtime.

Microsoft has also provided a Software Development Kit (SDK) for Kinect through which the low-level data streams from the Kinect video, microphone, and depth sensors can be accessed. This SDK provided by Microsoft is capable of tracking skeletal data, too. It can track the skeleton image of people moving within the Kinects field of view.

In [88], the performance of the skeleton from the Microsoft SDK is measured from the aspects of noise, accuracy, resolution and so on. However, the ground truth for the measurements is not from the true skeletal model, but instead from some measurements
with physical measuring tools, e.g. a wooden meter stick.

The skeleton generated from the depth data is less accurate and stable compared to mocap data. Still, one can adopt the methodology developed for mocap skeletons to work with RGB-D skeleton data as well in most cases. Different algorithms often generate different skeleton models. For example, the Kinect for Windows SDK [89] provides skeletons with 20 joints, and the NiTE library [90] generates skeletons with 15 joints. The Kinect for Windows version 2.0 SDK provides skeletons with 25 joints.

Patsadu et al. [91] demonstrated the high potential of using Kinect for human gesture recognition. They used twenty joint locations returned by Kinect to classify three different human gestures of stand, sit down and lie down. They showed that using different machine learning algorithms such as Support Vector Machines (SVM), decision trees, and naive Bayes, they could achieve an average accuracy rate of 93.72%. A neural network trained with back propagation achieved 100% accuracy.

Huang et al. [92] used skeletal joints from the Microsoft Kinect sensor along with SVM classification to achieve a recognition rate of 97% on signed gestures. The dataset comprised of fifty samples from two users each. They implemented ten ASL signs, all of which were based on arm movement. Their approach fails in case of gestures that involve finger and facial movement as skeleton tracking does not provide finger and facial movement information. Another limitation was that they used a fixed duration of ten frames, restricting the speed of which a sign can be made.

Biswas et al. [93] used depth images from the Kinect sensor for gesture recognition. They showed that depth data along with the motion profile of the subject can be used to recognize gestures like clapping and waving hands. They presented a lesser compute intensive approach for gesture recognition, and shown the accuracy of the system can be further improved by using skin color information from RGB data.

Zafrulla et al. [94] investigated the use of Kinect for their currently existing CopyCat system which is based on the use of a colored glove. They achieved accuracy rate of 51.5% and 76.12% on ASL sentence verification in seated and standing mode respectively which is comparable to the 74.82% verification rate when using the current CopyCat system. They used a larger dataset of 1000 ASL phrases. They used the depth information for ASL phrase verification, whereas the RGB image stream is used
to provide live feedback to the user. They used the skeleton tracking capability of the Kinect along with depth information to accurately track the arm and hand shape respectively.

In [95], the Kinect, a depth sensor combined with color camera are used to detect hand gestures, which are then translated into interaction commands and 3D pointing directions.

The block-diagram of Kinect is shown in figure 2.7. It has an infrared camera and a PrimeSense sensor to calculate the depth of the object, and an RGB camera to capture frames. The depth images and RGB image of the object could be got simultaneously. This 3D scanner system named Light Coding uses a variant of image-based 3D reconstruction [96].

Despite many achievements in using the Kinect sensor for face recognition and the human body tracking, it is still a challenge to employ Kinect for hand gesture recognition.

Kinect has several limitation and some of those are implicit in every manual of a Kinect game. first of all, there is a limitation on distance usage (0.8m minimum
to 3.5m maximum user distance). This can restrictive for families that do not have large living spaces (a common situation in Japan and Europe). In a Kinect game manual, it recommends 1.8m (6feet) for one player and 2.4m (8feet) for two players 2.8. Furthermore when playing together, the players have to give each other some space.

![Figure 2.8: Recommend Playable Distance for Kinect [5].](image)

Some minor issues like being in front of the camera in order to track the players body correctly have to be insured too. For instance, the best way for the Kinect to track the players hands for menus interface is to move the hands in front of the players body. Unfortunately another issue arises from this. If the depth camera does not identify a body part (e.g. head), that body part would be hidden in the game as well [97]. For this issue, the xbox team wrote a tracking algorithm that rejects these bad skeletons and only accepts the good ones. So theoretically Kinect could track the players skeletal body with its depth camera and the Kinect database of poses. This is in fact true, but
only for standing poses. This is due to Kinect database having standing poses only. Therefore, the Kinect will only work when the player is standing, even when navigating through menus [86]. Another issue is lighting exposure, i.e. the sunlight. Good lighting helps the Kinect sensor to recognize the player, but direct exposure to sunlight might interfere with the sensor. Wearing a jacket with big, oppy sleeves, a skirt, or a dress can also confuse the Kinect sensor. It may think they are extra body parts.

2.2.3.1 Kinect for XBox 360

Developed for the Xbox 360 console and released it is the first in the line of Kinect sensors. Kinect for Xbox 360 was built to track body that are up to 4.5 meters away from the sensor, but fails to track objects that are closer than 0.8 meters [98].

2.2.3.2 Kinect for Windows v1

The Kinect for Windows sensor comes with an external power supply for additional power and a USB (universal serial bus) adapter to connect with a PC or tablet. In Kinect for Windows v1 new firmware is added which enabled Near Mode tracking. This support to track objects as close as 40 centimeters in front of the device without losing accuracy or precision [99]. The Kinect for Windows SDK is designed to expand the purposes of the Kinect sensor so that it can be used to develop software for real-life applications with human gestures and voice commands by using C++, C#, Visual Basic or any other .NET language of Windows Store projection.

2.2.3.3 Kinect for XBox One

Kinect for Xbox One had major upgrades compared to its predecessor for Xbox 360. The new Kinect also uses infrared sensor to read its environment, but uses a wide-angle time-of-flight camera and processes 2 gigabits of data per second. The sensor can track without visible light by using an active infrared sensor. Kinect for Xbox One has no separate Near Mode, but has greater accuracy (resolution is 20 times greater) and an improved field of view, reducing the amount of distance needed between the player and the sensor for optimal Kinect performance. The new Kinect no longer contains a tilt
motor and the physical design of the new sensor is more similar to the architecture of the Xbox console.

2.2.3.4 Kinect for Windows v2

The second-generation Kinect for Windows alongside the Kinect for Windows version 2.0 SDK is intended for those developing Kinect-enabled software for Windows 8, Windows 8.1 and Windows 10. Higher depth fidelity makes it easier to see objects more clearly and in 3D. Microsoft introduced an adapter kit that enables users to attach their Kinect for Xbox One sensors to Windows PCs and tablets over USB 3.0.

2.2.3.5 Kinect Architecture

**Depth Data:** The Kinect uses the depth information to track the position of people in front of the sensor. The depth sensor consists of an infrared laser projector and monochrome CMOS sensor. The CMOS sensor captures light and converts it into electrical signals.

Inferring body position for Kinect for Xbox 360 works by first computing a depth map using structured light and then using machine learning. The depth map is constructed by analyzing a speckled pattern of infrared laser light that is invisible to the eye. The depth computation is done by the PrimeSense hardware built into Kinect. It works by projecting a known pattern onto the scene and inferring depth from the deformation of that pattern (structured light) through comparing the image from the camera with the pattern it knows it is displaying. The dots are arranged in a pseudo-random pattern that is hardwired into the sensor. The infrared sensor is fitted with a filter that keeps out ordinary light, which is how it can see just the infrared dots, even in a brightly lighten room.

The first Kinect combines structured light with two classic computer vision techniques: depth from focus and depth from stereo. Depth from focus uses the principle that objects that are more blurry are further away. The Kinect uses a special lens with different focal length in x- and y- directions. A projected circle then becomes an ellipse whose orientation depends on depth. Depth from stereo - if you look at the scene from
another angle, objects that are close get shifted to the side more than objects that are far away. The Kinect analyzes the shift of the speckled pattern by projecting from one location and observing from another [100].

As the reflecting surface gets farther away from the camera, the dots do not look smaller and closer together because the camera and the laser projector are epipolar rectified. This means, the camera and projector have matching fields of view. As the reflecting surface gets farther away from the sensor, the light from the laser is a cone of light that is getting larger as you get farther from its source [101].

Body parts are inferred by machine learning feeding the computer data in the form of millions of images of people and it learns how to understand them. The Kinect guesses which parts of the body are which based on all of its experience with body poses. Then, based on the probabilities assigned to different areas, Natal comes up with possible skeletons that could fit the body parts. Ultimately it selects the most probable one and outputs the data into a simplified 3D avatar shape. To get proper coordinates, the sensor calculates the three views of the same image: front, left and top view, by which it defines the 3D body proposal. Analyzing the data is done 30 times in a second [102].

The new Kinect has a time-of-flight camera. It consists of the following:

- **Illumination unit**: Illuminates the scene using infrared.
- **Optics**: A lens gathers the reflected light and images the environment onto the image sensor (focal plane array). An optical band-pass filter only passes the light with the same wavelength as the illumination unit. This helps suppress non-pertinent light and reduce noise.
- **Image sensor**: Each pixel measures the time the light has taken to travel from the illumination unit to the object and back to focal plane array.
- **Driver electronics**: Controls and synchronizes the illumination unit and image sensor.
- **Computation/Interface**: Distance is calculated directly in the camera and no manual depth computation is required.
The basic idea is that the round-trip time is measured in which the photons (quantum in which light and other forms of electromagnetic radiation are transmitted) are emitted by the sensor and reflected back and this time is used to calculate the distance. Because the sensor does this measurement with every one of its quarter million pixels at the same time, it can generate a 3D image with an amazing level of accuracy. This technology also improves the ability to see objects more clearly, recognize smaller objects and enhances the stability of body tracking, even fingers on the hands [103]. The sensor can see from 0.5 meters to 4.5 meters by default with the optimal distance for body detection is from 0.8 to 3.5 meters. The depth sensor can also see from 4.5 to 8 meters, but body detection does not work in this extended range. The field of view of the new sensor is 60% wider. As a result users can be closer to the camera and still in view and the camera is effective over a larger total area. Also the remarkable improvement in resolution has helped make considerable advancements in facial recognition with the Kinect sensor.

The circuitry design of the time-of-flight are very challenging making the sensor costly. 3DV Systems found a way to lower the costs dramatically and planned to release a RGB-Depth sensor, called ZCam [104].

**Color Camera:** The color camera is responsible for capturing and streaming the color video data. Its function is to detect the red, green and blue colors from the source. It simply acts like a basic webcam and records the room. The first Kinect captures images 30 frames per second and projects at 640x480 resolution. The color camera of the Kinect for Xbox One captures 1080p HD (high definition) video at 30 frames per second. The value of frames per second may vary depending on the resolution used for the image frame [105].

**New Active Infrared:** The new Kinect has an active infrared, which gives it the ability to see in the dark. The variations in the room lighting can make it difficult to see who is in the frame, so the active infrared removes ambient room lighting. Using the feed from the high definition color camera and active infrared, it is possible to detect a heart rate by looking for subtle variations in color and intensity in the skin of
the face. This can be used in exercise scenarios and when measuring exertion [106].

**Sensor Positioning:** The placement requirements of both Kinects are similar. For the old Kinect it was important to place the sensor so it can tilt and automatically adjust freely. With the Kinect for Xbox One, users are able to stand closer to the sensor and it offers a wider field of view. [83][107]

- The sensor works best when it is positioned between 0.6 and 1.8 meters off the floor. In smaller rooms the sensor should be as close to 1.8 meters as possible.

- The Kinect should stand near the edge of a flat, stable surface to see the feet and make sure no nearby objects are obstructing the field of view.

- The sensor should not be positioned near a speaker, vibrating surface or glass doors.

- Avoid positioning the sensor in direct sunlight. The depth camera works in all lighting situations, but with large amounts of natural light skeleton tracking becomes less reliable. Ideally the light source should be from behind the sensor.

### 2.2.3.6 Kinect for Windows SDK

The Kinect for Windows SDK is used to write programs that use Kinect devices. The SDK contains all the drivers needed to link a Kinect to your computer and provides a set of libraries that can be added to the software so it can interact with the sensor. It gives direct access to low-level video and depth signals and the powerful library features built into the SDK make it possible for a program to identify and track users. Building applications can be done using C# or Visual Basic or C++. The first two versions of Kinect for Windows SDK (Beta 1 and Beta 2) were non-commercial releases and were meant for hobbyists. On February 2012 the first commercial version (v1.0) was released and Kinect for Windows version 2.0 SDK was released on October 2014. The SDK also includes samples of good practices for using a Kinect sensor and example codes that breaks down samples into user tasks.
2.2.3.7 OpenNI

The open-source framework called OpenNI can also gather and use the depth data provided by Kinect. By feeding the depth data into OpenNI it allows several joints to be tracked in 3D with joint recognition algorithms. This framework is only one of the many hacked Kinect SDKs out there, but is by far the most used [108].

Together with the open-source driver called Sensor Kinect (by PrimeSense) and NITE (OpenNI Compliant Middleware Binaries and Libraries), the OpenNI framework can be used with Microsoft Windows [97] too. It also comes along with some interesting sample programs to test.

2.2.4 Gesture Recognition for Aircraft Handling on NATOPS Database

![Example of NATOPS Gestures: (a) brakes on, (b) brakes off, (c) insert chocks, (d) remove chocks [6].](image)

Figure 2.9: Example of NATOPS Gestures: (a) brakes on, (b) brakes off, (c) insert chocks, (d) remove chocks [6].

Aircraft handling signal recognition is an interesting and challenging real-world problem in the domain of human gesture recognition system. A real-time recognition system for aircraft handling is allowing two-way communication between humans and systems. In order to provide natural gesture-based interaction, it is important for a system to be able to recognize human gestures. At the same time the robustness of the system is truly necessary for a real-world problem. Figure 2.10 describes the NATOPS
In the aircraft carrier flight deck environment, deck personnel and pilots are required to have eye contact when communicating; pilots are in fact not permitted to take any action without having eye contact with one of the deck personnel confirming the action. This prevents misinterpretation of a command and aids in correcting mistakes immediately, keeping the environment safe.

To develop a vision-based real-time gesture recognition system which is comparable to a human pilots on an aircraft carrier flight deck environment, it is required the system to attend multiple information channels, specifically, a combination of body and hand poses and capable of avoiding obtrusive and unnatural interaction.

The Naval Air Training and Operating Procedures Standardization (NATOPS) manual standardizes general flight and operating procedures for the US naval aircraft. There are a large number of publications that belong to NATOPS; among the many, an aircraft signals manual (NAVAIR 00-80T-113 [10]) contains information about all aircraft systems including general aircraft and carrier flight deck handling signals. Several things make it interesting to use the NATOPS aircraft handling signals as a gesture vocabulary.

The aircraft handling signals are the ones that are currently used on the flight deck which is a realistic scenario. There are a lot of things going on in the carrier flight deck environment, which indicates that the aircraft handling signals should be designed to handle a wide range of situations to communicate. Also, the aircraft handling signals have been refined, modified, and optimized over the years, suggesting that the signals can be thought of as a well-defined gesture set.

The aircraft handling signals presented in the figure 2.10 can be an interesting problem domain for gesture recognition. There are many similar gesture pairs that are having completely opposite meanings. For example, the “brakes on” and “brakes off” gestures are performed by raising both hands, with either open palms that are closed (“brakes on”), or closed hands that are opened (“brakes off”) (figure 2.9). Here, the role of hand pose is crucial to differentiating these two very similar gestures with completely opposite meanings.

As a more subtle case: the “insert chocks” and “remove chocks” gestures are per-
formed with both arms down and waving them in/outward (figure 2.9). The only difference is the position of thumbs (inward and outward). The velocity of waving arms might be another indicator for differentiating the two gestures (going faster for the direction of thumbs), but it is not obvious whether even human eyes can catch the differences.

Gestures performed by Aviation Boatswains Mate Handlers (ABH) need to be perceived in 3D space. Many gestures are performed with self-occluding body poses, which are in general hard to reconstruct with a 2D image. Moreover, some gestures include directional information with pointing actions, which are lost in a 2D image.

Song et al. presented a unified framework for body and hand tracking, the output of which can be used for understanding simultaneously performed body-and-hand gestures. The framework uses a stereo camera to collect 3D images, and tracks body and hand together, combining various existing techniques to make tracking tasks efficient. They introduced a multi-signal gesture database: the NATOPS aircraft handling signals. Unlike previous gesture databases, this data requires knowledge about both body and hand in order to distinguish gestures. It is also focused on a clearly defined gesture vocabulary from a real-world scenario that has been refined over many years. The database includes 20 body and hand gestures, and provides both gesture video clips and the body and hand features we extracted (figure 2.10).

In addition to the challenges in recognizing gestures in the NATOPS manual, there are many constraints and challenges on the carrier flight deck environment that make the problem of gesture recognition even more interesting.

We cannot rely on the system detecting colors for locating body parts in captured images, although flight deck personnel wear color-coded helmets and jerseys (e.g., yellow helmet and jersey for aircraft directors, as in figure 2.9). This is because lighting conditions change through the day and the colors usually wear out.

Gestures performed by ABHs differ from person to person and from time to time, senior ABHs often simplify the standard gestures when they communicate with well experienced pilots; the fatigue effect simplifies the standard gestures as well. Therefore, we need a gesture recognition system that is robust to individual variations.
Figure 2.10: NATOPS Aircraft Handling Signals [6].
2.3 Gesture Classification

Gesture Classification is the final stage of gesture recognition in which the output of current gesture model is compared with each model in gesture database where the most matched gesture is selected as final recognition result. Different gesture modeling methods have diverse recognition approaches.

Gestures are almost always unique, as humans are unable to create identical gestures every single time. Humans, having an extraordinary ability to process visual signals and filter noise, have no problem understanding gestures which look alike. However, ambiguous gestures as such pose a big problem to machines attempting to perform gesture recognition, due to the injective nature of gestures to meanings. Similar gestures vary both spatially and temporally, hence it is non-trivial to compare gestures and determine their nature.

Most of the tools for gesture recognition originate from statistical modelling, including Principle Component Analysis, Hidden Markov Models, Kalman filtering, and Condensation algorithms [19]. In these methods, multiple training samples are used to estimate parameters of a statistical model.

Gesture classification has been taken to be primarily a solution for gesture recognition problem in the field of computer vision. Mutual disambiguation has been applied to improve recognition by constraining the gesture recognition candidates based on a set of possible semantic frames. But the idea that gesture classes themselves are fundamentally multimodal entities defined not only by the body motion but also by the role of gesture within the linguistic context has not yet been given full consideration.

The extracted features from images are exposed to different classifiers, from generic Support Vector Machines (SVMs) invented by Vladimir Vapnik [31] to highly customized shape classifiers such as in [35]. Some features carry out classification implicitly such as the LucasKanade-based tracker, which removes unreliable” patches, and Camshift [109], which determines a decision boundary in space and color histograms.

Classification can be integrated with feature extraction such as the boosting method including a combination of weak detectors [110]. Other approaches include a distinct translation step into feature space and subsequent classification. To illustrate, con-
sider tracking a gesture, with its spatial location over time providing feature vector and a hidden Markov model classifying movement trajectory into different temporal/dynamic gestures [34, 111, 112].

Lots of gesture classification approaches has been proposed and implemented by several researcher. The most popular methods are reviewed below.

### 2.3.1 Hidden Markov Model

Mappings between high dimensional feature sets and gestures are carried out by machine learning methods. The most well-known approach for these methods is using Hidden Markov Models (HMMs) in which gestures are dealt with as the output of a stochastic process. The Hidden Markov Model was extensively implemented in gesture recognition. For example in [113, 114, 115] has concentrated on HMMs for gesture recognition.

Hidden Markov Models assume the first order Markov property of time domain processes, i.e. 

\[ P(s_t|s_{t-1}s_{t-2}...s_1) = P(s_t|s_{t-1}) \]

![Figure 2.11: Architecture of Hidden Markov Model.](image)

The current event only depends on the most recent past event. The model is a double-layer stochastic process, where the underlying stochastic process describes a "hidden" process which cannot be observed directly, and an overlying process, where observations are produced from the underlying process stochastically and then used to estimate the underlying process. This is shown in figure 2.11, the hidden process
being \( x(t) \) and the observation process being \( y(t) \). Each HMM is characterized by \( \lambda = (A, B, \pi) \), where

- \( A = \{a_{ij}\} \) is a state transition matrix.

\[ A = \{a_{ij}\} = P(s_t|s_{t-1}) \]

- \( B = \{b_{ij}\} \) is the probability of observing symbol \( v_k \) from state \( s_j \).

- \( \pi = \{\pi_i\} \) is the initial state distribution.

\[ \pi = \{\pi_i\} = P\{s_i\} \text{att = 1} \]

Given the Hidden Markov Model and an observation sequence \( O = o_1, o_2...o_t \), three main problems need to be solved in its application,

1. Adjusting \( \lambda = (A, B, \pi) \) to maximize \( P(O|\lambda) \), i.e. adjusting the parameters to maximize the probability of observing a certain observation sequence.

2. In the reverse situation, calculate the probability \( P(O|\lambda) \) given \( O \) for each HMM model \( \lambda_i \).

3. Calculate the best state sequence which corresponds to an observation sequence for a given HMM.

In gesture recognition, we concern ourselves more with the first two problems. Problem 1 corresponds to training the parameters of the HMM model for each gesture with a set of training data. The training problem has a well-established solution, the Baum-Welch algorithm [116] (equivalently the Expectation Modification method) or the gradient method. Problem 2 corresponds to the evaluation of the probability of the various HMMs given a certain observation sequence, and hence determining which gesture was the most probable.

There have been many implementations of the Hidden Markov Model in various gesture recognition experiments. Simple gestures, such as drawing various geometry
shapes, were recorded using the Wii remote controller, which provides only accelerometer data, and accuracy was between 84% and 94% for the various gestures [112]. There have also been various works involving hand sign language recognition using various hardware, such as glove-based input [117, 2], and video cameras [118].

### 2.3.2 Neural Network-based Classifier

Neural Network (also known as artificial Neural Network) is a mathematical or computational model. This is one of the most extensively used models for machine learning. The model is hugely inspired from biological Neural Networks. A Neural Network consists of artificial neurons. These neurons are connected to each other via edges. The learning structure is mostly dependent on the weights that control the flow of data through the network during the learning phase.

Neural Networks can be applied to many tasks. Few broad categories are Regression analysis, Classification, Data processing and Robotics.

Neural Network can be viewed as directed graph with many nodes and arcs between them. Nodes are the processing elements and the arcs are the interconnection between them. Each of these nodes functions independently and uses only the input and output to that node to direct its processing.

Neural Network nodes can be divided in three groups. Input, output and internal or hidden nodes. Input nodes exists in the input layer, output nodes exists in the output layer and the hidden nodes exists in one or more hidden layers. A tuple is input through the input layer and the output nodes determine the prediction for class. An approximate figure 2.12 is shown below.

The output of each node is determined based on a function \( f \). This function is called the activation function. Activation functions are associated with every node. This function is also called a processing element function. Many proposals for activation functions include threshold, sigmoid, symmetric sigmoid, hyperbolic tangent and Gaussian.

Also a cost function is associated with a Neural Network. We try to minimize this cost to find an optimal Neural Network.
In this work, mean squared error is used as the cost function which tries to minimize the average squared error between the network’s output and the target value. It uses gradient descent to minimize cost. It uses back-propagation for training Neural Network.

2.3.3 Naive Bayes Classifier

Naive Bayes classifier is a simple probabilistic classifier. It uses the Bayes theorem. The “Naive” part of the name comes because it assumes strong (naive) independence. To be more precise, it takes the assumption that the feature values are independent. In spite of the simple design and naive assumptions, Naive Bayes classifiers have worked pretty well in many complex real-life situations.

The probabilistic model of Naive Bayes classifier can be derived using the Bayes theorem. Suppose the class variable is $C$ with limited number of outcomes and the feature values are $F_1$ through $F_n$. Using Bayes theorem we can write,

$$p(C|F_1,...,F_n) = \frac{p(C)p(F_1,...,F_n|C)}{p(F_1,...,F_n)}$$
Where,
\[ p(C|F_1,\ldots,F_n) = \text{posterior probability}, \]
\[ p(C) = \text{prior probability}, \]
\[ p(F_1,\ldots,F_n|C) = \text{likelihood}, \]
\[ p(F_1,\ldots,F_n) = \text{evidence}. \]

In practice, we are only interested about the numerator of the fraction, since the denominator does not depend on \( C \) and the values of the feature \( F_i \) is given. So that the denominator value is constant. The numerator value is equivalent to the joint probability \( p(C, F_1,\ldots,F_n) \). Using conditional probability we can write,

\[
p(C, F_1,\ldots,F_n) \\
\propto p(C)p(F_1,\ldots,F_n|C) \\
\propto p(C)p(F_1|C)p(F_2,\ldots,F_n|C,F_1) \\
\propto p(C)p(F_1|C)p(F_2|C,F_1)p(F_3,\ldots,F_n|C,F_1,F_2) \\
\propto p(C)p(F_1|C)p(F_2|C,F_1)p(F_3|C,F_1,F_2)\ldots p(F_n|C,F_1,F_2,F_2,\ldots,F_{n-1})
\]

Now the “naive” conditional independence assumes that the feature values are independent of each other. That means if \( i \neq j \) then

\[ p(F_i|C,F_j) = p(F_i|C) \]

Therefore the joint probability model can be expressed as

\[
p(C, F_1,\ldots,F_n) \\
\propto p(C)p(F_1|C)p(F_2|C)p(F_3|C)\ldots p(F_n|C) \\
\propto p(C) \prod_{i=1}^{n} p(F_i|C)
\]

This means under the above independence assumption the conditional distribution over the class variable can be expressed like this:
\[
p(C|F_1, F_2, ..., F_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(F_i|C)
\]

where \(Z\) (the evidence) is a scaling factor dependent only on \(F_1, F_2, ..., F_n\).

### 2.3.4 Support Vector Machine

Support Vector Machines are a learning algorithm which utilize hyper plane boundaries to separate classes based on their feature representation. Although SVMs are binary classifiers, which in their most fundamental state find a line through a feature space \(\mathbb{R}\) that separates two classes of data, they are able to expand to multi class classification problems (figure 2.13) [119]. For multi-dimensional expansion it is necessary to adopt a method known as one against all classification. In this method \(k\) models are built, where \(k\) is the number of classes being classified. Each model is trained with a single class containing positive labels while all other classes contain negative labels [120]. For

![Figure 2.13: Maximum-margin hyperplane and margins for an SVM trained with samples from two classes.](image)

clarity, let’s assume that the feature space to be classified is linearly separable. In this case a new gesture instance can be classified by training the SVM and observing which side of the division the new datum falls on. The calculations for classification are as
follows:

\[ D(x) = \sum_{i \in S} a_i y_i w_i^T x + b_i \]

Where S is a set of support vectors, these are the vectors which form the division boundary, \( w_i^T \) is the support vector, \( a_i \) are a set of Lagrange multipliers that help unconstrain the problem, and \( b \) is a bias term given by:

\[ b = \frac{1}{|S|} \sum_{i \in S} (y_i - w_i^T x_i) \]

Using this method a new feature vector, representing a new gesture, can be classified as:

\[
c = \begin{cases} 
  \text{Class 1} & \text{if } D(x) > 0 \\
  \text{Class 2} & \text{if } D(x) < 0 
\end{cases}
\]

Though this method was originally presented as a linear classifier, it is able to classify non-linear/high dimensional data through the use of nonlinear kernel transforms. Using these methods, data is mapped into a higher dimensional feature space where a linear separation is possible. In this study two kernels will be tested. The first is the radial basis function kernel (RBF), which has been found to offer optimal results in gesture recognition [121]. This kernel is given by:

\[ H(x, x') = \exp(-\gamma ||x - x'||^2) \]

where \( \gamma \) is a positive parameter controlling the radius of the kernel. The second is the linear kernel given by the function:

\[ H(x, x') = x^T x'^T + c \]

The linear kernel is the simplest of all the kernel functions; in fact the use of the linear kernel is often equivalent to an algorithms non-kernel counterpart. As seen in its formulation above, it is given by the inner product of \( x \) and \( x' \) plus an optional constant \( c \) [122].

2.4 Summary

In this chapter, we have discussed about gestures and various gesture recognition approaches. We have also discussed Pattern classification techniques like Naive Bayes,
Neural Network, Support Vector Machine and Hidden Markov Model classification algorithm. In the next chapter, we will describe the proposed system and the gesture domain.
Chapter 3

Proposed System Descriptions

Automatic gesture recognition in a real-world environment has a variety of potential applications from smart surveillance to human-machine interaction to biometrics. The main objective of gesture recognition research is to build a system which can recognize human gestures and utilize them to control an application. This thesis attempts to help bridge the visual communication gap between computers and humans by designing and implementing a gesture recognition system.

This chapter describes the proposed gesture recognition system in detail. Section 3.1 presents problem formulation. This section has worked to setup the basic problem that this thesis tries to tackle. The methodologies of implementation are explained in detail in the next section. Section 3.2 describes each module of the proposed gesture recognition system including approaches for grouping different feature representations. Section 3.3 represents Gesture Domain that the proposed system recognizes. In section 3.4 we have summarize the whole proposed system.

3.1 Problem Formulation

Gesture recognition from a real-world scenario is one of the most important challenges in computer vision and behavior understanding since it offers to the machine the ability to identify, recognize and interpret the human gestures in order to control some devices, to interact with some human machine interfaces (HMI) or to monitor some human
activities.

The goal of this thesis is computer vision-based real-time 3D full-body motion Gesture Recognition and comparative analysis of the system. Our research focuses on vision based approach of understanding simultaneously performed body-and-hand gestures.

To solve the gesture recognition problem we have to formulate it as a modularized computational problem and solve each module independently. Like all other computational problem this problem has a set of input and a set of output. In this case, the set of input is set of gesture taken by some imaging or sensor device. To solve this problem we are taking the help of Kinect, part of Xbox One from Microsoft, well renowned gaming device. The output of this problem will be a type of gesture which is recognized by the proposed system.

We used Kinect sensor camera for the devices capabilities of the depth sensing and body tracking. The input data are streams of vectors of twenty-six body-joint positions obtained by standard Application Programming Interface (API) of the Kinect Software Development Kit (SDK). These joints represent the human body captured by Kinect camera.

![Input-Process-Output Model](image)

Figure 3.1: Input-Process-Output Model

As shown in the figure 3.1, in between the input and output there should be a processing system, which will receive the input data, process the data, calculate the result and the show the output.

In the processing system, we have to match the pattern of the gesture with a predefined gesture database. But as the input gestures is not a straight forward data, the matching process will be a complex system. But before the matching process we have to make the input gesture images or motion images system readable by converting them into matrices of numbers. The next step is very important that we perform a distance adaption and position normalization to deal with parameters of different units and scales of body-joint positions.
This can be done using a Kinect sensor device to collect 3D images, and track body and hand together, followed by some calculation to make the input data usable. The complex part of this system is the pattern matching component. There are lots of techniques as discussed in the chapter, but several machine learning approaches get the priority as all state of the art researches are based on machine learning techniques. Figure 3.2 shows a step by step model of problem formulation.

![Figure 3.2: A step by step model of problem formulation](image)

A multi-signal gesture database: The Naval Air Training and Operating Procedures Standardization (NATOPS) aircraft handling signals has taken as a gesture vocabulary from a real-world scenario. Our system will be tested with a subset of standard aircraft handling signals defined in NATOPS [10].

To recognize human gesture defined in NATOPS dataset various pattern classification methods (such as, SVM, ANN, Naive Bayes and HMM) are then applied. Then we compare the performance of each method to find the optimal classifier.

### 3.2 Methods of Implementation

The structure of a gesture recognition system is usually determined by the characteristics of the used features and classifiers. Therefore, in this section we review features
and classifiers separately, while the whole system is revealed indirectly through the introduction of both features and classifiers.

Gestures are considered as the most natural expressive way for communications between human and computers in virtual system. This work formulates a computer vision-based method that will recognize 3D full-body motion gestures in real-time.

A general method of recognition should be able to recognize gestures performed by any human user. This system performs real-time mapping with a vision based approach through a 3D visual sensor and classify a set of human movements as particular gestures using machine learning techniques.

We will explore the methods used in conducting the experiments specific to this thesis. first a depth view of the features extracted from the input dataset will be given, and how they relate to the classifiers, and the gestures in the set. Following this, a series of experiments will be outlined which cover the rest of the system from feature selection to the optimization of each of the four individual classifiers.

The core idea of the system procedure are:

- 3D Data Processing
- Feature Extraction
- Gesture Modeling
- Gesture Learning
- Gesture Recognition

### 3.2.1 3D Data Acquisition

The implementation is based on the data structures which represent data provided by the sensor. The Kinect for Windows v2.0 SDK implements its own data structures but these structures have limited implementation. The data of these structures is not possible to clone and regarding to its implementation with non public constructors, it is not possible to instantiate them instantly and thus it is not possible to use them for
custom data. This limitations have been overcame by encapsulating data from these native structures into the own object data representation.

The key points of the method are:

- Complete human body tracking using Kinect
- Mapping All Joints of a Human in a Frame
- Kinect Skeleton Coordinate System Generation
- Kinect Skeleton Scaling in 3D Space

### 3.2.1.1 Complete human body tracking using Kinect

In skeletal tracking, a human body is represented by a number of joints representing body parts such as head, neck, shoulders, and arms (see figure 3.5 ). Each joint is represented by its 3D coordinates. The goal is to determine all the 3D parameters of these joints in real time to allow fluent interactivity. Kinect for Windows version 2.0 SDK allows us to track up to 25 body joints including the fists and thumbs. A joint corresponds to a part of the human body tracked by the Kinect: The sensor provides us with the 3D position (X, Y, Z) and the rotation information for each one of them. Moreover, Kinect lets us know whether the joints are tracked, hypothesized or not tracked. Its a good practice to check whether a body is tracked before performing any critical functions. A depth sensor with infrared (IR) light make Kinect capable of providing the depth image of the scene in front of it. An example output of skeleton tracking is shown in figure 3.3.

The skeletal tracking allows the Kinect to recognize people and follow their actions [21]. The body joints are not always perfectly aligned to the background image (figure 3.4). Because the color, infrared and depth sensors are not one above the other, so they have a slightly different point of view. We used the coordinate mapper of to align them. Kinect act as a real time 3D vision sensor camera. Kinect for Windows v2.0 SDK provides these 25 skeletal body-joints. The supported joints by Kinect 2 are the following:

- **SpineBase**: Middle point of two hips.
Figure 3.3: Example of RGB image and depth image of a Human

Figure 3.4: Skeleton on the color image (skeleton in the left image is not aligned with human body and the right one is aligned with the help of coordinate mapper [7])

- **SpineMid**: Center of the spine. Positioned near the naval region.
- **Neck**: Middle point of two shoulders.
- **Head**: Coordinate of the center of the head. Approximately positioned near the nose.
- **ShoulderLeft**: Coordinate of the left shoulder. End position of the left collar bone.
- **ElbowLeft**: Coordinate of the left shoulder. End position of the left collar bone.

- **WristLeft**: Coordinate of the left wrist.

- **HandLeft**: Positioned in the center of the left palm.

- **ShoulderRight**: Coordinate of the right shoulder. End position of the right collar bone.

- **ElbowRight**: Coordinate of the right elbow.

- **WristRight**: Coordinate of the right wrist.

- **HandRight**: Positioned in the center of the right palm.

- **HipLeft**: Positioned at the starting of the left leg. Its the outer joint where the left leg is connected with the hip.

- **KneeLeft**: Coordinate of the left knee.

- **AnkleLeft**: Coordinate of the left ankle.

- **FootLeft**: Positioned at the center of the left foot.

- **HipRight**: Positioned at the starting of the right leg. Its the outer joint where the right leg is connected with the hip.

- **KneeRight**: Coordinate of the right knee.

- **AnkleRight**: Coordinate of the right ankle.

- **FootRight**: Positioned at the center of the right foot.

- **SpineShoulder**: Middle point of the left shoulder and right shoulder.

- **HandTipLeft**: Coordinate of the fingertip of left hand.

- **ThumbLeft**: End position of the left thumb.

- **HandTipRight**: Coordinate of the fingertip of right hand.
• **ThumbRight:** End position of the right thumb.

Neck and thumbs are new joints added in the second version of Kinect. Figure 3.5 shows skeleton joints of a human identified by Kinect for Windows version 2.0 SDK.

![Skeleton Joints of a Human](image)

Figure 3.5: Skeleton Joints of a Human identified by Kinect for Windows version 2.0 SDK [8]

### 3.2.1.2 Mapping All Joints of a Human in a Frame

A frame refers to a still shot of the 3D motion data. A collection of frames with certain speed of transition creates the illusion of movement or motion. A single frame thus describes the current properties of the scene.

In this work, we have tracked the skeleton of the human using kinect sensor along
with SDK. It provide 25 body-joints. An example of tracking human skeleton in a frame is given in figure 3.6.

Figure 3.6: Example of tracking of a Human in a single frame

A frame is represented by a mapping of 3D coordinates of the 25 joints that can be tracked by using kinect sensor along with SDK. This mapping is represented as a matrix of X, Y, Z coordinate of all joints. The mapping of all joints of the frame from figure 3.6 is shown in table 3.1.
Table 3.1: Mapping of all joints for a single frame

<table>
<thead>
<tr>
<th>Joint ID</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpineBase</td>
<td>-0.385</td>
<td>-0.143</td>
<td>1.478</td>
</tr>
<tr>
<td>SpineMid</td>
<td>-0.391</td>
<td>0.204</td>
<td>1.542</td>
</tr>
<tr>
<td>Neck</td>
<td>-0.392</td>
<td>0.533</td>
<td>1.591</td>
</tr>
<tr>
<td>Head</td>
<td>-0.384</td>
<td>0.651</td>
<td>1.617</td>
</tr>
<tr>
<td>ShoulderLeft</td>
<td>-0.466</td>
<td>0.367</td>
<td>1.465</td>
</tr>
<tr>
<td>ElbowLeft</td>
<td>-0.467</td>
<td>0.102</td>
<td>1.353</td>
</tr>
<tr>
<td>WristLeft</td>
<td>-0.299</td>
<td>-0.113</td>
<td>1.276</td>
</tr>
<tr>
<td>HandLeft</td>
<td>-0.282</td>
<td>-0.149</td>
<td>1.298</td>
</tr>
<tr>
<td>ShoulderRight</td>
<td>-0.215</td>
<td>0.416</td>
<td>1.578</td>
</tr>
<tr>
<td>ElbowRight</td>
<td>-0.142</td>
<td>0.220</td>
<td>1.591</td>
</tr>
<tr>
<td>WristRight</td>
<td>-0.219</td>
<td>0.100</td>
<td>1.433</td>
</tr>
<tr>
<td>HandRight</td>
<td>-0.254</td>
<td>0.057</td>
<td>1.369</td>
</tr>
<tr>
<td>HipLeft</td>
<td>-0.434</td>
<td>-0.138</td>
<td>1.423</td>
</tr>
<tr>
<td>KneeLeft</td>
<td>-0.466</td>
<td>-0.467</td>
<td>1.482</td>
</tr>
<tr>
<td>AnkleLeft</td>
<td>-0.434</td>
<td>-0.738</td>
<td>1.467</td>
</tr>
<tr>
<td>FootLeft</td>
<td>-0.512</td>
<td>-0.779</td>
<td>1.383</td>
</tr>
<tr>
<td>HipRight</td>
<td>-0.319</td>
<td>-0.141</td>
<td>1.465</td>
</tr>
<tr>
<td>KneeRight</td>
<td>-0.302</td>
<td>-0.466</td>
<td>1.430</td>
</tr>
<tr>
<td>AnkleRight</td>
<td>-0.277</td>
<td>-0.737</td>
<td>1.434</td>
</tr>
<tr>
<td>FootRight</td>
<td>-0.311</td>
<td>-0.797</td>
<td>1.390</td>
</tr>
<tr>
<td>SpineShoulder</td>
<td>-0.392</td>
<td>0.453</td>
<td>1.581</td>
</tr>
<tr>
<td>HandTipLeft</td>
<td>-0.257</td>
<td>-0.149</td>
<td>1.285</td>
</tr>
<tr>
<td>ThumbLeft</td>
<td>-0.253</td>
<td>-0.119</td>
<td>1.246</td>
</tr>
<tr>
<td>HandTipRight</td>
<td>-0.272</td>
<td>0.004</td>
<td>1.311</td>
</tr>
<tr>
<td>ThumbRight</td>
<td>-0.216</td>
<td>0.002</td>
<td>1.405</td>
</tr>
</tbody>
</table>
3.2.1.3 Kinect Skeleton Coordinate System Generation

A coordinate system referenced to an object (i.e. human body) is called an object coordinate system. It often helps to define a coordinate system for each object in order to make it easier to adjust the program if the object is moved. Kinect skeleton coordinate system refers to a coordinate system referencing the point in the center of the skeleton known as SpineMid as the origin (0,0,0). Figure 3.7 shows the object coordinate system of a human body referencing the SpineMid point of kinect skeleton as the origin of the coordinate system.

Figure 3.7: The body coordinates system referencing the SpineMid point as origin [9].

According to Kinect for Windows version 2.0 SDK documentation, in front of the Kinect is the positive Z-axis, above the Kinect is the positive Y-axis and the right of the Kinect is the positive X-axis. Thus, the 3D coordinate system used by Kinect is a right-handed coordinate system, and is similar to a view space in computer graphics. Figure 3.8 showing the camera space of Kinect version 2. Camera space refers to the
3D coordinate system used by Kinect. The coordinate system is defined as follows: The origin \((x = 0, y = 0, z = 0)\) is located at the center of the IR sensor on Kinect, \(X\) grows to the sensors left, \(Y\) grows up, \(Z\) grows out in the direction the sensor is facing, 1 unit = 1 meter.

![Kinect camera space coordinate system](image)

Figure 3.8: Kinect camera space coordinate system [4].

However, people can stand at any position in the real application. For classification, it will be a little trouble to set the value of feature. Therefore, we need to translate the coordinates to adapt the distance of the user from the kinect sensor, so that all the joint points can have a uniform coordinates independent of the user position for a specific body posture. So the system will calculate the geometric translation in 3D Cartesian coordinates and define SpineMid (body-joint coordinate near Center of the spine) as origin.

In computer graphics, it always applies the homogeneous coordinate system to simplify the operation of the matrix. Its convenient to record the translation, resize and rotation by the matrix. The homogeneous coordinate system use \(d + 1\) dimensional coordinates to represent the point on \(d\)-dimensional space. For example, three dimensional coordinates point \((x, y, z)\) is become \((wx, wy, wz, w)\) in the homogeneous coordinate system. The forth coordinate value \(w\) is called homogenous coordinate and cannot equal to zero. For a geometric translation \(w\) is expressed the distance parameter. The translation is to move a graph in the certain distance and direction. Also \(v\) is the position before translation; \(v'\) is the position after translation; \(dx\) is the translational component of \(x\)-axis; \(dy\) is the translational component of \(y\)-axis; \(dz\) is the translational component of \(z\)-axis. Furthermore, they can be represented by transla-
tion matrix. The 3D Cartesian coordinates is expressed by the homogeneous coordinate system. The position after translation is the dot product of the translation matrix and the original homogeneous coordinate matrix in 3D Cartesian coordinates. If we move the original 3D Cartesian coordinate position \((x_{SpineMid}, y_{SpineMid}, z_{SpineMid})\) of the SpineMid (almost center point of the skeleton) as origin \((0, 0, 0)\) of the Cartesian coordinates, it can generate an coordinate system translating the SpineMid to origin.

The original 3D coordinate position \((x_i, y_i, z_i)\) of the other 25 joints take the original 3D Cartesian coordinate position \((x_{SpineMid}, y_{SpineMid}, z_{SpineMid})\) of the SpineMid as the reference point. They are translated from the originial 3D Cartesian coordinate position to the new coordinate \((x'_i, y'_i, z'_i)\). Geometric Translation has been applied on 3D Cartesian coordinates as figure 3.9.

![Geometric translation of coordinate](image)

**Figure 3.9: Geometric translation of coordinate**

We perform the three-dimensional coordinate’s translation of geometry conversion as:

\[
v = [x, y, z]  
v' = [x', y', z']  
D = [dx, dy, dz]
\]
\[ v' = v + d \]
\[ = T(dx, dy, dz). [x, y, z, 1]^T \]
\[ = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \]
\[ = [x + dx, y + dy, z + dz, 1]^T \]

Figure 3.10 shows the coordinate of SpineMid and ShoulderLeft before translation. And figure 3.11 shows the coordinates after translation.

Figure 3.10: The Coordinate System before translation

Translated joint positions are shown in table 3.2.

3.2.1.4 Kinect Skeleton Scaling in 3D Space

It is not promising for a human being that body movement against a specific gesture will be strictly same at different time or perspective or even pose given by different people. Rather it can vary slightly time to time, person to person. To avoid this circumstance Kinect skeletal joint points are scaled by a scale factor, so that the skeletal joints that
are supposed to be in the same position get into the same range. The ratio of any two corresponding lengths in two similar geometric figures is called as scale factor. Joint-points lies in a specific range will be observed in a discrete coordinate after scaling.

The scale operator performs a geometric transformation which can be used to shrink or zoom the size of an image (or part of an image). Image reduction, commonly known as subsampling, is performed by replacement or by interpolating between pixel values in a local neighborhoods.

To scale an object by a vector \(v = (v_x, v_y, v_z)\), each point \(p = (p_x, p_y, p_z)\) would need to be multiplied with this scaling matrix. Scaling matrix can be represented like this:

\[
S_v = \begin{bmatrix}
v_x & 0 & 0 \\
0 & v_y & 0 \\
0 & 0 & v_z \\
\end{bmatrix}
\]

The multiplication will give the expected result:
Table 3.2: Mapped data with joint position translation

<table>
<thead>
<tr>
<th>Joint ID</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpineBase</td>
<td>6</td>
<td>-347</td>
<td>-64</td>
</tr>
<tr>
<td>SpineMid</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Neck</td>
<td>-1</td>
<td>329</td>
<td>49</td>
</tr>
<tr>
<td>Head</td>
<td>7</td>
<td>447</td>
<td>75</td>
</tr>
<tr>
<td>ShoulderLeft</td>
<td>-75</td>
<td>163</td>
<td>-77</td>
</tr>
<tr>
<td>ElbowLeft</td>
<td>-76</td>
<td>-102</td>
<td>-189</td>
</tr>
<tr>
<td>WristLeft</td>
<td>92</td>
<td>-317</td>
<td>-266</td>
</tr>
<tr>
<td>HandLeft</td>
<td>109</td>
<td>-354</td>
<td>-244</td>
</tr>
<tr>
<td>ShoulderRight</td>
<td>176</td>
<td>212</td>
<td>36</td>
</tr>
<tr>
<td>ElbowRight</td>
<td>249</td>
<td>16</td>
<td>49</td>
</tr>
<tr>
<td>WristRight</td>
<td>172</td>
<td>-104</td>
<td>-109</td>
</tr>
<tr>
<td>HandRight</td>
<td>137</td>
<td>-147</td>
<td>-173</td>
</tr>
<tr>
<td>HipLeft</td>
<td>-42</td>
<td>-342</td>
<td>-119</td>
</tr>
<tr>
<td>KneeLeft</td>
<td>-75</td>
<td>-671</td>
<td>-60</td>
</tr>
<tr>
<td>AnkleLeft</td>
<td>-43</td>
<td>-943</td>
<td>-75</td>
</tr>
<tr>
<td>FootLeft</td>
<td>-121</td>
<td>-983</td>
<td>-159</td>
</tr>
<tr>
<td>HipRight</td>
<td>72</td>
<td>-346</td>
<td>-77</td>
</tr>
<tr>
<td>KneeRight</td>
<td>89</td>
<td>-671</td>
<td>-112</td>
</tr>
<tr>
<td>AnkleRight</td>
<td>114</td>
<td>-942</td>
<td>-108</td>
</tr>
<tr>
<td>FootRight</td>
<td>80</td>
<td>-1001</td>
<td>-152</td>
</tr>
<tr>
<td>SpineShoulder</td>
<td>-1</td>
<td>249</td>
<td>39</td>
</tr>
<tr>
<td>HandTipLeft</td>
<td>134</td>
<td>-353</td>
<td>-258</td>
</tr>
<tr>
<td>ThumbLeft</td>
<td>138</td>
<td>-324</td>
<td>-296</td>
</tr>
<tr>
<td>HandTipRight</td>
<td>119</td>
<td>-201</td>
<td>-231</td>
</tr>
<tr>
<td>ThumbRight</td>
<td>175</td>
<td>-202</td>
<td>-137</td>
</tr>
</tbody>
</table>
\[
S_v p = \begin{bmatrix}
v_x & 0 & 0 \\
0 & v_y & 0 \\
0 & 0 & v_z \\
\end{bmatrix}
\begin{bmatrix}
p_x \\
p_y \\
p_z \\
\end{bmatrix} = \begin{bmatrix}
v_x p_x \\
v_y p_y \\
v_z p_z \\
\end{bmatrix}
\]

3.2.2 Feature Extraction

The most important part of gesture recognition is feature extraction where the observation of a pattern is transformed into a vector, whose components are called features. Features are easily tractable for the system, but should contain most of the information necessary for classification of the patterns. The procedures for feature extraction may be based on intuition or physical considerations of the problem, or they may be purely mathematical techniques for simply reducing the dimensionality of the observations. Figure 3.12 represents a gesture recognition model including the feature extraction model in between gesture observation and gesture classification phase of the model.

Figure 3.12: Feature extraction in Gesture Recognition model.

Gestures are usually described as sequence of features per sample. Feature points enable us to localize points where descriptors have to be computed since they usually correspond to body parts where the movement can be easily discernable. Identification and computations of good features are important for the 3D motion gesture recognition. The commonly used motion features include trajectory, location, orientation and velocity. This thesis emphasized on motion trajectory and location for full-body gesture recognition. Skeletal data has the advantage of simple calculation of the motion
features, e.g. trajectory, and location. These features help to detect and track gesture given by people from Kinect sensor device.

The modeling of the feature and feature space is an implementation issue and may vary based on target application. An automatic methods for gesture recognition requires to obtain good features. In this thesis two different class has been used for describing features. Section 3.2.2.1 and 3.2.2.2 will show different approaches for grouping two different feature representations, Body Joint Coordinate and Motion Trajectory respectively.

### 3.2.2.1 Body Joint Coordinate

The features extracted from gesture domain are the position of the 13 joint points of different limbs and fingertips of human body relative to the centroid of the spine. Table 3.3 contains the list of joint point position names which are the actual feature points with respect to the domain of this thesis.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ShoulderLeft</td>
</tr>
<tr>
<td>2</td>
<td>ElbowLeft</td>
</tr>
<tr>
<td>3</td>
<td>WristLeft</td>
</tr>
<tr>
<td>4</td>
<td>HandLeft</td>
</tr>
<tr>
<td>5</td>
<td>HandTipLeft</td>
</tr>
<tr>
<td>6</td>
<td>ThumbLeft</td>
</tr>
<tr>
<td>7</td>
<td>ShoulderRight</td>
</tr>
<tr>
<td>8</td>
<td>ElbowRight</td>
</tr>
<tr>
<td>9</td>
<td>WristRight</td>
</tr>
<tr>
<td>10</td>
<td>HandRight</td>
</tr>
<tr>
<td>11</td>
<td>HandTipRight</td>
</tr>
<tr>
<td>12</td>
<td>ThumbRight</td>
</tr>
<tr>
<td>13</td>
<td>Head</td>
</tr>
</tbody>
</table>
At each time frame, the human body tracking module takes raw data from the sensor and estimates the body-joint location of the gesturing body. After obtaining the skeletal model the next step is computing its feature vector. The feature extraction module computes feature descriptors from the localized body joints and sends the calculated features to the gesture recognition module.

3.2.2.2 Motion Trajectory

The main goal of the system is to extract people motion features in order to analyze them for gesture recognition. Compared to the Body Joint Coordinate, the motion of the hands and arms provides further options to be used as features for the gesture recognition. Thus some specific body-joint coordinate should be chosen to determine the regions of the human body where relevant motion can be extracted.

For a given individual in a scene, feature points (Body Joint Coordinates) are tracked over objects whole body to extract the motion of the body parts. Capturing correct gestures requires the feature points to be distributed sufficiently over the body. The gesture recognition module estimates the current most likely gesture label and gesture phase information based on the input stream of feature vectors.

For each feature points from the body joint position in a particular frame, four more feature will be added. They are angle in left elbow $\theta_{le}$, angle in left shoulder $\theta_{ls}$, angle in right elbow $\theta_{re}$ and angle in right shoulder $\theta_{rs}$. These angles are measured using the vectors mentioned below:

$\vec{V}_{lew} =$ Vector from left elbow to left wrist.

$\vec{V}_{les} =$ Vector from left elbow to left shoulder.

$\vec{V}_{lsh} =$ Vector from left shoulder to left hand tip.

$\vec{V}_{lam} =$ Vector from left shoulder to spine mid.

Similarly
\vec{V}_{rew} = \text{Vector from right elbow to right wrist.}

\vec{V}_{res} = \text{Vector from right elbow to right shoulder.}

\vec{V}_{rsh} = \text{Vector from right shoulder to right hand tip.}

\vec{V}_{rsm} = \text{Vector from right shoulder to spine mid.}

\theta_{le} = \cos^{-1} \frac{\vec{V}_{lew} \cdot \vec{V}_{les}}{\| \vec{V}_{lew} \| \| \vec{V}_{les} \|}

\theta_{ls} = \cos^{-1} \frac{\vec{V}_{lsh} \cdot \vec{V}_{ism}}{\| \vec{V}_{lsh} \| \| \vec{V}_{ism} \|}

\theta_{rc} = \cos^{-1} \frac{\vec{V}_{rew} \cdot \vec{V}_{res}}{\| \vec{V}_{rew} \| \| \vec{V}_{res} \|}

\theta_{rs} = \cos^{-1} \frac{\vec{V}_{rsh} \cdot \vec{V}_{rsm}}{\| \vec{V}_{rsh} \| \| \vec{V}_{rsm} \|}

\subsection{3.2.3 Gesture Modeling}

Gesture modeling is concerned with the fitting and fusing the input gesture into the model used. In this modular design, the length of the gesture is not a limiting factor for feature extraction. When the features are extracted based on each frame, more detailed information can be preserved in the features. Also, the classification can be performed immediately as each frame arrives, which leaves only the light computation of the sequence level model to the end of the gesture. This design greatly facilitates real-time recognition of the gestures.

To effectively distinguish actions, the actors perform a resting gesture for a certain amount of time before performing a concrete action. Gesture is modeled using a sequence of N frames, where N is predefined. The value of N needs to be fixed for gesture learning and recognition purpose. Because if the size is dynamic, the classifier will never know when to compare for matching. So the value of N must be fixed.
The sequence needs to be ordered, because the meaning of a gesture depends on the sequence of the motion. The order in which the frames occur is very important in recognizing gesture. Same set of frames occurring in different order means different gestures.

So, a gesture is represented as \((F_1, F_2, ..., F_N)\) Where, \(f_i\) is the \(i\)-th frame and \(N\) is the number of image frames in the sequence. An example of a gesture model can be shown in figure 3.13 and 3.14.

This step may require some pre-processing steps to ensure the successful convergence to a unified set of gestures. The pre-processing steps are typically normalization, sampling, translation etc. which are already been done. In this stage the input data is already invariant to the subject’s size, shape and location in the frame. Making gesture interaction feel natural requires a system that responds at the correct moment, meaning that we have to consider the temporal characteristics of a gesture. The gesture modeling technique is based on sampling the gradient of the gesture movement trajectory and presenting the gesture trajectory as a sequence of numbers. This techniques have some important features for gesture recognition including robustness against slight rotation, a small number of required samples, invariance to the start position and device independence.

3.2.4 Gesture Learning

After the features are extracted from the skeleton data, the next step is to classify the features to recognize the gestures. The former is to classify the features frame by frame; the latter is to use all the outputs from the frame level classifiers to evaluate the final classification result for the gesture. In this modular design, the length of the gesture is not a limiting factor for feature extraction. When the features are extracted based on each frame, more detailed information can be preserved in the features. Also, the classification can be performed immediately as each frame arrives, which leaves only the light computation of the sequence level model to the end of the gesture. This design greatly facilitates real-time recognition of the gestures.

In this work we used pattern classification algorithms for recognizing gestures.
Figure 3.13: Model of a gesture with N=8 (part-1)
Figure 3.14: Model of a gesture with N=8 (part-2)
Naive Bayes, Support Vector Machine, Hidden Markov Model and Neural Network-based classifier is used for the recognition purposes because it can easily learn and train from the features computed for gesture recognition. Before applying classification algorithms, a gesture is first converted into a single row vector. This is done by linearizing each frame independently and then concatenating each frame in the sequence.

### 3.2.5 Gesture Recognition

The final phase of the proposed system is gesture recognition. This stage usually has a temporal classifier that can attach each input testing gesture to the closest matching class. The input to this stage is an unseen test data sequence along with the model parameters learned from training. The general technique to tackle the gesture recognition problem is to deal with it as a pattern recognition problem where a set of features are extracted from images captured from a video file or webcam that in turn are matched (i.e., Classification) to a predefined representation of the gesture (i.e., pattern). A reliable set of characteristic features and relevant information of how they correlate in representing gestures are needed in order to successfully recognize gestures. A set of features describing a given object is used to learn a classifier, and in turn, the trained classifier is responsible for recognizing the (distorted) object by matching the extracted features.

To recognize gestures, we need to observe the ordered frame sequences. From the N number image sequences, a linearized row vector is generated and compared with the training dataset to find out which class it belongs. For classification we used Naive Bayes, Support Vector Machine, Hidden Markov Model and Neural Network-based classifier. A list of frames is kept for the current candidate. As long as the list size is not equal to N, we keep populating it with the frames sequentially in order they occur. While the list sizes N, it becomes a candidate for testing. This testing is done by already trained classifiers. If it is recognized as a predefined gesture, the gesture name and id is shown and the whole list is flushed to empty. Otherwise the oldest frame (in front of the list) is popped out and the latest one is added at the end. This process runs continuously. The process can be better understood from figure 3.15.
3.3 Gesture Domain of The Thesis

The Naval Air Training and Operating Procedures Standardization (NATOPS) manual standardizes general flight and operating procedures for the US naval aircraft. There
are a large number of publications that belong to NATOPS; among the many, an aircraft signals manual [10] contains information about all aircraft systems including general aircraft and carrier flight deck handling signals. We have selected 20 NATOPS aircraft handling signals, the gestures most often used in routine practice on the deck environment.

3.3.1 Gesture of “I Have Command”

If there is any available commands to show for any aircraft this command should be given first then other commands will follow. Gesture of “I have command” given in figure 3.16. The right hand raised and waits for sometime indicate this command.

3.3.2 Gesture of “All Clear”

When everything is clear for the pilot to do his work this sign is given. The gesture is given in figure 3.17. The sign consist of raised hand with thumbs up.

3.3.3 Gesture of “Not Clear”

This sign indicates negative result. The sign consist of arms held out on the level of waist and thumbs downwards. The gesture of the signal “Not Clear” is given in figure 3.18.

3.3.4 Gesture of “Spread Wings”

The gesture of “spread wings” deliver the message when to command the pilot to spread the wings of the aroplane. The arms swept straight out to the sides from the front. The gesture is given in figure 3.19.

3.3.5 Gesture of “Fold Wings”

The gesture of “fold wings” deliver the message when to command the pilot to fold the wings of the aroplane. The arms swept forward from straight out states. The gesture is given in figure 3.20.
Figure 3.16: Gesture of the sign “I have command”
Figure 3.17: Gesture of the sign “All Clear”
Figure 3.18: Gesture of the sign “Not Clear”
Figure 3.19: Gesture of the sign “Spread Wings”
Figure 3.20: Gesture of the sign “Fold Wings”
3.3.6 Gesture of “Lock Wings”

The gesture is to hit right elbow with the palm of left hand. The gesture of “Lock wings” is given in figure 3.21.

3.3.7 Gesture of “UP Hook”

In this gesture right fist, thumb extended upward, raised suddenly to meet horizontal palm of left hand. The gesture of “up hook” is given in figure 3.22.

3.3.8 Gesture of “Down Hook”

In this gesture right fist, thumb extended downward, raised suddenly to meet horizontal palm of left hand. The gesture of “down hook” is given in figure 3.23.

3.3.9 Gesture of “Remove Chocks”

This gesture is like arms down, fists closed, thumbs extended outwards, swing arms outwards. The gesture of “remove chocks” is given in figure 3.24.

3.3.10 Gesture of “Insert Chocks”

This gesture is like arms down, fists closed, thumbs extended inwards, swing arms from extended position inwards. The gesture of “insert chocks” is given in figure 3.25.

3.3.11 Gesture of “Move Ahead”

The gesture is shown as like: arms extended from body and held horizontal to shoulders with hands up-raised and above eyelevel, palms facing backwards, execute beckoning arm motion angled backward, rapidly indicates speed desired of aircraft. The gesture of move ahead is given in figure 3.26.
Figure 3.21: Gesture of the sign “Lock Wings”
Figure 3.22: Gesture of the sign “Up Hook”
Figure 3.23: Gesture of the sign “Down Hook”
Figure 3.24: Gesture of the sign “Remove Chocks”
Figure 3.25: Gesture of the sign “Insert Chocks”
Figure 3.26: Gesture of the sign “Move Ahead”
3.3.12 Gesture of “Turn Left”

The gesture of “turn left” is given in figure 3.27. It is given by acting the following task: extend right arm horizontally, left arm is repeatedly moved upward, speed of arm movement indicating rate of turn.

3.3.13 Gesture of “Turn Right”

The gesture of “turn right” is given in figure 3.28. It is given by acting the following task: extend left arm horizontally, left arm is repeatedly moved upward, speed of arm movement indicating rate of turn.

3.3.14 Gesture of “Slow Down”

The gesture of “slow down” is given in figure 3.29. The gesture shown by arms down with palms towards ground, then moved up and down several times.

3.3.15 Gesture of “Stop”

The gesture of “Stop” is given in figure 3.30. In this gesture arm crossed above the head with palms facing forward.

3.3.16 Gesture of “fire”

This gesture describes large figure eight with one hand and points to the fire area with the other hand. The gesture of “fire” is given in figure 3.31.

3.3.17 Gesture of “Brakes On”

The signal “brake on” means hold the hand straight up the head and close the palm in a fist. figure 3.32 showing the gesture breaks on.
Figure 3.27: Gesture of the sign “Turn Left”
Figure 3.28: Gesture of the sign “Turn Right”
Figure 3.29: Gesture of the sign “Slow Down”
Figure 3.30: Gesture of the sign “Stop”
Figure 3.31: Gesture of the sign "fire"
Figure 3.32: Gesture of the sign “brakes on”
Figure 3.33: Gesture of the sign “Brakes Off”
Figure 3.34: Gesture of the sign “Next Marshaler”
Figure 3.35: Gesture of the sign “Install Tiedown”
3.3.18 Gesture of “Brakes Off”

The signal “brake off” means hold the hand straight up the head with a fist and open the palm. figure 3.32 showing the gesture “brakes off”.

3.3.19 Gesture of “Next Marshaler”

The gesture “Next Marshaler” is shown by fixing left arm down and right arm moved across the body and extended to indicate direction to next marshal. figure 3.34 showing the gesture “Next Marshaler”.

3.3.20 Gesture of “Install Tiedown”

In the signal “Install Tiedown” crew rotates hands in a circle perpendicular to and in front of the body. figure 3.32 showing the gesture “Install Tiedown”.

3.4 Summary

In this chapter we have described detail of the proposed gesture recognition system. In the next chapter we will present experimental results with appropriate discussion.
Chapter 4

Experimental Results Discussions

A computer vision-based real-time 3D motion gesture recognition system for aircraft handling has been proposed in the Chapter 3. In this chapter we are going to describe the experimental setup and results in terms of measurement of the accuracy and complexity of the proposed system. Before going through the result the experimental setup and performance measure parameters will be discussed.

4.1 Experimental Setup

The experiment was conducted in a computer with “Intel Core i7” processor with 2.4 GHz clock rate and 8 GB RAM. The operating system was “Windows 10 Pro, 64-bit”. The “Kinect for XBox One” motion sensor is used to capture the motion data of the user, which is the latest version of Kinect from Microsoft Corporation Limited[123]. The proposed system is developed using C# (C-Sharp) language with the help of Kinect for Windows version 2.0 SDK provided by Microsoft, dot NET framework for graphical user interfaces and machine learning API, Accordion.NET.

4.1.1 Kinect Sensor

The Kinect sensor, originally known as Project Natal, is a motion-sensing device developed by Microsoft. The sensor is a horizontal device with a color camera, depth sensors and a set of microphones (Figure 4.1) designed to be positioned lengthwise
above or below the video display. The Kinect provides a NUI (natural user interface) that can detect body motion, gestures, spoken commands and even facial recognition. This enables users to interact with their console or personal computer without using hand-controlled devices[124].

Required 3D body-joint points was achieved by projecting the Kinect sensor to the bodies of the user and the most challenging part was to track their motions and rapid movements without any delay in real-time[124].

4.1.2 Kinect SDK

The Kinect for Windows version 2.0 SDK is a free downloadable software development toolkit. It is used to write programs that use Kinect devices. The SDK contains all the drivers needed to link a Kinect to the computer and provides a set of libraries that can be added to the software so it can interact with the sensor (shown in figure 4.2). It gives direct access to low-level video and depth signals and the powerful library features built into the SDK make it possible for a program to identify and track users. Building applications can be done using C# or Visual Basic or C++ [125]. The SDK also includes samples of good practices for using a Kinect sensor and example codes that breaks down samples into user tasks [8].

The Kinect SDK uses depth information to track the position of people in the field of view of the sensor. Skeletons can be tracked whether the user is standing (25 joints) or seated (10 joints). With the SDK v2.0 the tracked body positions are more anatomically correct and stable (shown in figure 4.3). The Kinect skeleton is made up of 25 joints including new joints for hand tips, thumbs and shoulder center. Each joint
Figure 4.2: Microsoft Kinect SDK Architecture: interaction between application and different layers of components

is positioned in 3D space relative to the Kinect sensor. The position of each joint is given as an offset from the Kinect sensor. From the point of view of the user the x-axis extends to the right, the y-axis upward and z-axis is oriented from the sensor to the user and expressed in meters.

Some important features of Kinect for Windows version 2.0 SDK are:

- It is possible to create Windows Store applications and offer them to consumers.
- It also offers support for Unity Pro users for building and publishing apps.
- Multiple apps can now access a single sensor.
- Representation of a person's face is more lifelike.
- Visual Gesture Builder enables developers to build their own gestures that can be recognized decreasing the time to prototype and test solutions.
In our experiment, Kinect is the primary device for motion sensing. It is used for capturing the skeleton data of the user. While in use, the Kinect was put in at height around 1 to 1.5 meters. The distance between the Kinect and the user was approximately 2.5 to 4 meters. If the distance is less than this, then the full body can’t be tracked by Kinect. So its required to keep at least 2.5 meter distance from the Kinect.

4.1.3 Programming Environment

The implementation of this thesis is done using C# and built with a WPF (Windows Presentation Foundation) rendering engine from Microsoft Dot NET framework in Microsoft Visual Studio 2013 IDE. C# dot NET is one of the core and native language of different Microsoft products.

4.1.3.1 Accord.NET Framework

Accord.NET is a framework for scientific computing in .NET. The framework builds upon the popular AForge.NET Framework, focusing on providing statistical methods, machine learning, pattern recognition, audio processing and computer vision algorithms. This framework is written completely in .NET and is not simply a wrapper around native libraries. It offers an extensive list of examples and sample applications to get the user cope up with the framework easily. Source code is readily available and
it be used in commercial applications with minor restrictions.

4.2 Implementation

The system uses a learn-and-predict strategy which tries to classify a test sample from the training data-set or integrate it to the learning process if it fails to do so. Then, the new test sample can be considered as a classified one or can serve to update the training data-set.

The major parts of implementation of the system are: Learning Module, Recognition Module and Performance Analysis Parameters.

4.2.1 Learning Module

Supervised learning method is used in this experiment to train the system and classify gestures. To generate the training dataset we ask 15 volunteers to play the predefined gestures. 9 of them are male and 6 of them are female. The height of those volunteers vary from 152 cm to 182 cm with average being 170 cm. These volunteers are well familiar with the gesture domain that we are using as we brief them about the gesture domain.

These volunteers were instructed to show the 20 predefined gestures one by one for five times each. While showing the gestures, they were instructed to remain relaxed and to show the gesture assuming they were acting as a real aircraft carrier flight deck personnel. This was encouraged to simulate the real life speed and longevity of a gesture. They were requested to avoid any personal idiosyncrasies.

With the procedure, a total of almost 15000 frames were captured and stored. These frames constitutes almost 1500 gesture instances.

Whenever the volunteers were showing a gesture, the data of their skeleton was being labeled with the appropriate gesture id manually and then saved in a file. When the gesture was being saved, it was maintained that the duration of the gestures be sufficiently long, so that, later the system can be tested upon different sizes of gesture.

This module has several important components. Such as:
• Kinect Interface
• User Detection
• Data Collection
• Data Preprocessing
• Training

4.2.1.1 Kinect Interface

This interface is used to interact with the Kinect sensor and acquisition of the required data. It is written in visual studio 2013 using C# dotNET and utilizes Kinect for Windows version 2.0 SDK. Skeleton coordinates are produced from Kinect sensor interface through body frame reader (one of the four readers: depth frame reader, body frame reader, inferred frame reader, color frame reader).

4.2.1.2 User Detection

Body frame reader can detect a human presence in front of the Kinect sensor. It returns BodyFrame type data when kinect sensor detect any human in front of the camera. The computed data provided by this frame type includes skeletal joints and orientations, hand states, and more for up to 6 people at a time. These tracking features provide a great baseline for getting started with human interaction. The Body class is distinct from other frame payloads. Instead of simply being an image stored as bytes or shorts in an array, it stores all the information the runtime has on a tracked body as well as associated functionality. This capability also gives us 3D coordinates of 25 joints of a human body which is mainly used in the experiment. And it will be expanded as the runtime starts tracking new things about a body.

4.2.1.3 Data Collection

Though Kinect for Windows version 2.0 SDK can be used to detect multiple users at a time, this experiment works with only one user. The user skeleton data (meaning: coordinates of the joints) is saved in file upon operator command.
Usually Kinect for Windows version 2.0 SDK provides 30 frames per second (fps) which is plenty of information to process. For this experiment we used 3 frames per second to reduce the computational cost.

### 4.2.1.4 Data Preprocessing

After collecting the data, it is processed before used for training or testing. Processing includes:

- Extracting $N$ tuples of data, where $N$ is the number of frames in a gesture sequence. Each tuples consists of the mapping of 3D coordinates of required 13 body joints of an user.

- Generating a Kinect Skeleton Coordinate System (KSCS). It refers to a coordinate system referencing SpineMid (the point in the center of the skeleton) as the origin $(0, 0, 0)$ as shown in figure 3.7.

- Scaling the 3D coordinates of each joints using scaling matrix.

- Labeling with predefined gesture ID.

### 4.2.2 Training Module

After the processing of data is finished, the training have the training dataset consists of 13 3D joint point coordinates, 4 angle values and two fist status values as shown in table 4.1. This dataset represents a single frame of a particular gesture. There are $N$ such frames for each gesture in training dataset. This training dataset is used to train classifiers such as Neural Network, Naive Bayes Classifier, Hidden Markov Model, Support Vector Machine. This is described in detail in section 3.2.4.

We have trained the classifier with the training dataset using 20-fold cross validation. The mean error rate for different $N$ of each classifier has been mentioned in the table 4.2.
Table 4.1: Sample Frame of Training Dataset

<table>
<thead>
<tr>
<th>#</th>
<th>Feature Name</th>
<th>Feature Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Coordinate of ShoulderLeft</td>
<td>(-3, 480, 3)</td>
</tr>
<tr>
<td>2</td>
<td>Coordinate of ElbowLeft</td>
<td>(-169, 191, -65)</td>
</tr>
<tr>
<td>3</td>
<td>Coordinate of WristLeft</td>
<td>(-204, -59, -59)</td>
</tr>
<tr>
<td>4</td>
<td>Coordinate of HandLeft</td>
<td>(-209, -273, -153)</td>
</tr>
<tr>
<td>5</td>
<td>Coordinate of HandTipLeft</td>
<td>(-210, -348, -171)</td>
</tr>
<tr>
<td>6</td>
<td>Coordinate of ThumbLeft</td>
<td>(157, 187, -59)</td>
</tr>
<tr>
<td>7</td>
<td>Coordinate of ShoulderRight</td>
<td>(206, -75, -35)</td>
</tr>
<tr>
<td>8</td>
<td>Coordinate of ElbowRight</td>
<td>(250, -264, -112)</td>
</tr>
<tr>
<td>9</td>
<td>Coordinate of WristRight</td>
<td>(249, -334, -142)</td>
</tr>
<tr>
<td>10</td>
<td>Coordinate of HandRight</td>
<td>(-191, -415, -172)</td>
</tr>
<tr>
<td>11</td>
<td>Coordinate of HandTipRight</td>
<td>(-228, -361, -193)</td>
</tr>
<tr>
<td>12</td>
<td>Coordinate of ThumbRight</td>
<td>(225, -399, -170)</td>
</tr>
<tr>
<td>13</td>
<td>Coordinate of Head</td>
<td>(267, -374, -142)</td>
</tr>
<tr>
<td>14</td>
<td>angle of left elbow, $\theta_{le}$</td>
<td>80</td>
</tr>
<tr>
<td>15</td>
<td>angle of left shoulder, $\theta_{ls}$</td>
<td>90</td>
</tr>
<tr>
<td>16</td>
<td>angle of right elbow, $\theta_{re}$</td>
<td>175</td>
</tr>
<tr>
<td>17</td>
<td>angle of right shoulder, $\theta_{rs}$</td>
<td>95</td>
</tr>
<tr>
<td>18</td>
<td>Right Fist Status</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>Left Fist Status</td>
<td>1</td>
</tr>
</tbody>
</table>

4.2.3 Gesture Recognition Module

The recognition module recognizes gesture in real-time mode. That means, as the user is moving in front of Kinect, whenever he is showing a gesture, this module tries to recognize it. N-number of image frame are used to define a gesture. This value of N is predefined. Kinect Sensor has frame rate of 30 Hz, that means 30 image frames are generated in a second. After studying the gesture domain it has been figured out that each gesture takes about 3 seconds in average. So, with nine or ten frame in average on
Table 4.2: Mean Training Error for All Classifiers for Different N

<table>
<thead>
<tr>
<th>N</th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.29</td>
<td>0.23</td>
<td>0.18</td>
<td>0.44</td>
</tr>
<tr>
<td>8</td>
<td>0.17</td>
<td>0.11</td>
<td>0.09</td>
<td>0.34</td>
</tr>
<tr>
<td>10</td>
<td>0.023</td>
<td>0.019</td>
<td>0.019</td>
<td>0.12</td>
</tr>
<tr>
<td>12</td>
<td>0.021</td>
<td>0.018</td>
<td>0.017</td>
<td>0.09</td>
</tr>
<tr>
<td>14</td>
<td>0.021</td>
<td>0.017</td>
<td>0.017</td>
<td>0.08</td>
</tr>
</tbody>
</table>

a regular interval we can represent a particular gesture without losing any important information. We have decided to conduct our experiment with value of N 6 to 14. The function of this module is predefined in the flow chart described in figure 3.15.

Here a frame is mapped to the 3D coordinate of 13 required joints of the user skeleton, 4 angles and 2 fist status. A list or queue is maintained to store the frames received from the Kinect sensor. List size is the number of frames currently in the list. N-number of image frames is used to define a gesture. If list size and N is equal, then we may have a potential candidate for gesture. So in this case the list is fed to the already trained classifier (Naive Bayes, Neural Network, Hidden Markov Model and Support Vector Machine) as test dataset and the classifier guesses its gesture ID. If the ID is not one of the gesture in the domain (1 to 20, ID 0 indicates every other gesture or garbage gesture) then the first (oldest) frame of the list is removed. Otherwise, when we find a match, the list is initialized to empty again.

4.2.4 Parameters for Performance Analysis

This module is designed to test and verify the performance and accuracy of the proposed gesture recognition system. Its written in C# dotNET, using the C# dotNET API of Accord Framework.

Though we have a recognition module to recognize the gestures shown in front of Kinect, but that is ill-suited for testing and analysis of performance and accuracy. Rather we used 20-fold cross validation of the training dataset for testing. N-number
of image frames are used to define a gesture. For Performance analysis we varied the value of N from 5 to 12.

Several performance parameters are used to analyze the performance of the proposed gesture recognition system such as: confusion matrix, precision, TP rate or recall, FP rate, F-measure, accuracy and average response time,

4.2.4.1 Confusion Matrix

Confusion matrix is a specific table to visualize the performance of an algorithm. Each column of the matrix represents the instances in a predicted class, which each row the instances in an actual class. Its also called the error matrix.

4.2.4.2 Precision

The Precision is the proportion of the examples classified as class x which truly have class x among all those which were classified as class x. Precision describes the degree with which a certain result can be repeated with unchanged condition. It can be formulated as:

\[
Precision = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of False Positives}} \quad (4.1)
\]

4.2.4.3 TP Rate or Recall

The True Positive (TP) rate or Recall or sensitivity is the proportion of examples classified as class x which truly have class x, among all examples which are of class x. Its formulated as:

\[
Recall = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of False Negatives}} \quad (4.2)
\]

4.2.4.4 FP Rate

The False Positive (FP) rate is the proportion of examples which were classified as class x, but belong to a different class, among all examples which are not of class x. The formula for FP Rate is:
\[ FPRate = \frac{\text{Number of False Positives}}{\text{Number of False Positives} + \text{Number of True Negatives}} \quad (4.3) \]

**4.2.4.5 F-Measure**

F-Measure is the harmonic mean of Precision and Recall. It's used to combine those two measures.

\[ F - Measure = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4.4) \]

**4.2.4.6 Accuracy**

Accuracy is the percentage of correctly classified data. In our case, the correctly recognized number of gestures. It can be written as:

\[ \text{Accuracy(\%)} = \frac{\text{Number of Correctly Recognized Gesture Instances}}{\text{Total Number of Gesture Instances}} \times 100 \quad (4.5) \]

**4.2.4.7 Average Response Time**

Average response time is the amount of time taken in average by the system to recognize a gesture shown by the human. This doesn’t include time taken for training.

**4.2.4.8 κ-value**

The Kappa statistic (or value) is a metric that compares an observed accuracy with an expected accuracy. The kappa statistic is used not only to evaluate a single classifier, but also to evaluate classifiers amongst themselves. In addition, it takes into account random chance (agreement with a random classifier), which generally means it is less misleading than simply using accuracy as a metric. Computation of observed accuracy and expected accuracy is integral to comprehension of the kappa statistic, and is most easily illustrated through use of a confusion matrix. The equation for \( \kappa \) is:

\[ \kappa = \frac{\text{observed accuracy} - \text{expected accuracy}}{1 - \text{expected accuracy}} \quad (4.6) \]
4.3 Results of Gesture Recognition

The performance analysis module (section 4.2.4) tests the performance of the proposed gesture recognition system. Four classification algorithms such as Naive Bayes, Neural Network, Hidden Markov Model and Support Vector Machine is used for gesture learning and recognition.

4.3.1 Confusion Matrix

Table 4.3 presents the confusion matrix for image sequence of 20 gestures, where Naive Bayes classification algorithm is used and the number of Kinect body frame (N) for each gesture is 10.

Table 4.3: Confusion Matrix for Naive Bayes Classifier: (N = 10).

|    | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  |
|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1  | 444 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 12  | 9   | 0   | 0   | 0   | 0   | 0   | 0   |
| 2  | 0   | 412 | 28  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 3  | 0   | 33  | 407 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 4  | 0   | 0   | 435 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 5  | 0   | 0   | 0   | 435 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 6  | 0   | 0   | 0   | 0   | 390 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 7  | 0   | 0   | 0   | 0   | 0   | 331 | 79  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 8  | 0   | 0   | 0   | 0   | 66  | 344 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 9  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 10 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 375 | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 11 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 341 | 0   | 0   | 49  | 0   | 0   | 0   | 0   |
| 12 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 430 | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 13 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 430 | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 14 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 34  | 0   | 0   | 336 | 0   | 0   | 0   | 0   |
| 15 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 18  | 0   | 358 | 0   |
| 16 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 302 | 64  | 0   |
| 17 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 95  | 270 |
| 18 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 375 |
| 19 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 360 |
| 20 | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 360 |

Table 4.4 presents the confusion matrix for image sequence of 20 gestures, where Neural Network classification algorithm is used and the number of Kinect body frame
(N) for each gesture is 10.

Table 4.4: Confusion Matrix for Neural Network-based Classifier: (N = 10).

|   | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 461| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 3  | 1  | 0  | 0  | 0  | 0  |
| 2 | 0  | 418| 15 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 7  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 3 | 0  | 19 | 419| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 2  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 4 | 0  | 0  | 0  | 131| 3  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 5 | 0  | 0  | 0  | 2  | 433| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 6 | 0  | 0  | 0  | 0  | 0  | 390| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 7 | 0  | 0  | 0  | 0  | 0  | 0  | 402| 8  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 8 | 0  | 0  | 0  | 0  | 0  | 0  | 7  | 403| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 9 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 362| 9  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 10| 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 5  | 379| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 11| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 376| 0  | 0  | 0  | 14 | 0  | 0  | 0  | 0  | 0  |
| 12| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 430| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 13| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 376| 0  | 0  | 0  | 14 | 0  | 0  | 0  | 0  | 0  |
| 14| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 6  | 0  | 0  | 364| 0  | 0  | 0  | 0  | 0  | 0  |
| 15| 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 6  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 16| 7  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 17| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 18| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 69 | 0  | 0  | 0  |
| 19| 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 3  | 0  | 0  | 369| 2  | 0  |
| 20| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 2  | 358|

Table 4.5 presents the confusion matrix for image sequence of 20 gestures, where Support Vector Machine classification algorithm is used and the number of Kinect body frame (N) for each gesture is 10.

Table 4.6 presents the confusion matrix for image sequence of 20 gestures, where Hidden Markov Model classification algorithm is used and the number of Kinect body frame (N) for each gesture is 10.

The confusion matrix gives us insight about the performance of the classification algorithms against each individual gesture. From the confusion matrices shown in table 4.3, 4.4, 4.5 and 4.6 we can see that other than a few gestures the overall performance is appreciable. Some gestures like gesture-17(brakes on), gesture-18(brakes off) are difficult to interpret for the classifier. These gestures are predicted with about 76% to 81% precision. The reason behind the unsatisfactory accuracy is the difficulties to
Table 4.5: Confusion Matrix for Support Vector Machine Classifier: \((N = 10)\).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>464</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>423</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>7</td>
<td>432</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>431</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>435</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>387</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>409</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>406</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>362</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>368</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>385</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>430</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>430</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>367</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>375</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>369</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>23</td>
<td>338</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

interpret the fist status(open or close) by the Kinect sensor.
4.3.2 Precision

Gesture wise precision is calculated for Naive Bayes, Support Vector Machine, Hidden Markov Model and Neural Network-based classifier for this system. Table 4.7 compares achieved precision for 20 gestures for all four classifiers.

Gesture wise precision column chart gives us better understanding about which gesture can be properly recognized and which is not. This column chart is shown in four parts in figure 4.4, 4.5, 4.6 and 4.7. N is fixed to 10 for this column chart.

According to the column chart it can be claimed that almost all the gestures have a satisfactory result. The system has an average precision of 93.7%, 96.9%, 98.0% and 90.7% for Naive Bayes, Neural Network, Support Vector Machine and Hidden Markov Model classifier respectively when Kinect body frame (N) is set to 10. Support Vector Machine achieved the highest precision among all these classifiers because the features
Table 4.7: Comparison of Precision for 20 Gestures

<table>
<thead>
<tr>
<th>Gesture Id</th>
<th>Gesture Name</th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I have Command</td>
<td>0.967</td>
<td>0.983</td>
<td>0.985</td>
<td>0.920</td>
</tr>
<tr>
<td>2</td>
<td>All Clear</td>
<td>0.926</td>
<td>0.957</td>
<td>0.981</td>
<td>0.961</td>
</tr>
<tr>
<td>3</td>
<td>Not Clear</td>
<td>0.936</td>
<td>0.963</td>
<td>0.975</td>
<td>0.935</td>
</tr>
<tr>
<td>4</td>
<td>Spread Wings</td>
<td>1.000</td>
<td>0.984</td>
<td>0.993</td>
<td>0.911</td>
</tr>
<tr>
<td>5</td>
<td>Fold Wings</td>
<td>1.000</td>
<td>0.993</td>
<td>0.998</td>
<td>0.973</td>
</tr>
<tr>
<td>6</td>
<td>Lock Wings</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.997</td>
</tr>
<tr>
<td>7</td>
<td>Up Hook</td>
<td>0.834</td>
<td>0.983</td>
<td>0.988</td>
<td>0.933</td>
</tr>
<tr>
<td>8</td>
<td>Down Hook</td>
<td>0.813</td>
<td>0.981</td>
<td>0.985</td>
<td>0.928</td>
</tr>
<tr>
<td>9</td>
<td>Remove Chocks</td>
<td>1.000</td>
<td>0.986</td>
<td>0.984</td>
<td>0.926</td>
</tr>
<tr>
<td>10</td>
<td>Insert Chocks</td>
<td>1.000</td>
<td>0.976</td>
<td>0.971</td>
<td>0.968</td>
</tr>
<tr>
<td>11</td>
<td>Move Ahead</td>
<td>0.909</td>
<td>0.984</td>
<td>0.992</td>
<td>0.897</td>
</tr>
<tr>
<td>12</td>
<td>Turn Left</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.923</td>
</tr>
<tr>
<td>13</td>
<td>Turn Right</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.929</td>
</tr>
<tr>
<td>14</td>
<td>Slow Down</td>
<td>0.834</td>
<td>0.953</td>
<td>0.984</td>
<td>0.912</td>
</tr>
<tr>
<td>15</td>
<td>Stop</td>
<td>0.968</td>
<td>0.969</td>
<td>0.964</td>
<td>0.804</td>
</tr>
<tr>
<td>16</td>
<td>Fire</td>
<td>0.976</td>
<td>0.987</td>
<td>0.961</td>
<td>0.811</td>
</tr>
<tr>
<td>17</td>
<td>Brakes On</td>
<td>0.761</td>
<td>0.824</td>
<td>0.935</td>
<td>0.788</td>
</tr>
<tr>
<td>18</td>
<td>Brakes Off</td>
<td>0.811</td>
<td>0.873</td>
<td>0.911</td>
<td>0.748</td>
</tr>
<tr>
<td>19</td>
<td>Next Marshaler</td>
<td>1.000</td>
<td>0.995</td>
<td>0.995</td>
<td>0.875</td>
</tr>
<tr>
<td>20</td>
<td>Install Tiedown</td>
<td>1.000</td>
<td>0.994</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

are integer values, so that the linear kernel used in Support Vector Machine Classifier can easily separates the feature values in hyperplane.

Precisions of all classifier increases over N (table 4.8). The precision graph for the system is described in figure 4.8 for different values of N (6,8,10,12,14).

114
The precision value for N greater than 10 remains almost same or change very slightly but the computational complexity and response time increase rapidly. The
change is negligible comparative to high computational complexity. But if the value of
N is set below 10 the precision decreases to an unsatisfactory result.
Table 4.8: Precision for Different Value of N

<table>
<thead>
<tr>
<th>N</th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.518</td>
<td>0.563</td>
<td>0.594</td>
<td>0.400</td>
</tr>
<tr>
<td>8</td>
<td>0.848</td>
<td>0.818</td>
<td>0.892</td>
<td>0.759</td>
</tr>
<tr>
<td>10</td>
<td>0.937</td>
<td>0.969</td>
<td>0.980</td>
<td>0.907</td>
</tr>
<tr>
<td>12</td>
<td>0.941</td>
<td>0.971</td>
<td>0.989</td>
<td>0.910</td>
</tr>
<tr>
<td>14</td>
<td>0.943</td>
<td>0.977</td>
<td>0.990</td>
<td>0.910</td>
</tr>
</tbody>
</table>

Figure 4.8: Precision Graph for different values of N

4.3.3 TP Rate or Recall

Gesture wise recall is calculated for Naive Bayes, Support Vector Machine, Hidden Markov Model and Neural Network-based classifier for this system. Table 4.9 compares achieved recall for 20 gestures for all four classifiers.

Gesture wise recall column chart gives us better understanding about which gesture can be properly recognized and which is not. This column chart is shown in four parts in figure 4.9, figure 4.10, figure 4.11 and figure 4.12. N is fixed to 10 for this column chart.

High recall means that an algorithm returned most of the relevant results. Recall
<table>
<thead>
<tr>
<th>Gesture Id</th>
<th>Gesture Name</th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I Have Command</td>
<td>0.955</td>
<td>0.991</td>
<td>0.998</td>
<td>0.966</td>
</tr>
<tr>
<td>2</td>
<td>All Clear</td>
<td>0.936</td>
<td>0.950</td>
<td>0.961</td>
<td>0.948</td>
</tr>
<tr>
<td>3</td>
<td>Not Clear</td>
<td>0.925</td>
<td>0.952</td>
<td>0.982</td>
<td>0.975</td>
</tr>
<tr>
<td>4</td>
<td>Spread Wings</td>
<td>1.000</td>
<td>0.991</td>
<td>0.995</td>
<td>0.989</td>
</tr>
<tr>
<td>5</td>
<td>Fold Wings</td>
<td>1.000</td>
<td>0.995</td>
<td>1.000</td>
<td>0.993</td>
</tr>
<tr>
<td>6</td>
<td>Lock Wings</td>
<td>1.000</td>
<td>1.000</td>
<td>0.992</td>
<td>0.928</td>
</tr>
<tr>
<td>7</td>
<td>Up Hook</td>
<td>0.807</td>
<td>0.980</td>
<td>0.998</td>
<td>0.946</td>
</tr>
<tr>
<td>8</td>
<td>Down Hook</td>
<td>0.839</td>
<td>0.983</td>
<td>0.990</td>
<td>0.926</td>
</tr>
<tr>
<td>9</td>
<td>Remove Chocks</td>
<td>1.000</td>
<td>0.965</td>
<td>0.965</td>
<td>0.938</td>
</tr>
<tr>
<td>10</td>
<td>Insert Chocks</td>
<td>1.000</td>
<td>0.984</td>
<td>0.981</td>
<td>0.896</td>
</tr>
<tr>
<td>11</td>
<td>Move Ahead</td>
<td>0.874</td>
<td>0.964</td>
<td>0.987</td>
<td>0.987</td>
</tr>
<tr>
<td>12</td>
<td>Turn Left</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.916</td>
</tr>
<tr>
<td>13</td>
<td>Turn Right</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.916</td>
</tr>
<tr>
<td>14</td>
<td>Slow Down</td>
<td>0.908</td>
<td>0.984</td>
<td>0.992</td>
<td>0.868</td>
</tr>
<tr>
<td>15</td>
<td>Stop</td>
<td>0.942</td>
<td>0.987</td>
<td>0.987</td>
<td>0.895</td>
</tr>
<tr>
<td>16</td>
<td>Fire</td>
<td>0.971</td>
<td>0.981</td>
<td>0.984</td>
<td>0.863</td>
</tr>
<tr>
<td>17</td>
<td>Brakes On</td>
<td>0.827</td>
<td>0.882</td>
<td>0.907</td>
<td>0.725</td>
</tr>
<tr>
<td>18</td>
<td>Brakes Off</td>
<td>0.740</td>
<td>0.811</td>
<td>0.926</td>
<td>0.690</td>
</tr>
<tr>
<td>19</td>
<td>Next Marshaler</td>
<td>1.000</td>
<td>0.984</td>
<td>0.995</td>
<td>0.879</td>
</tr>
<tr>
<td>20</td>
<td>Install Tiedown</td>
<td>1.000</td>
<td>0.994</td>
<td>0.956</td>
<td>0.830</td>
</tr>
</tbody>
</table>

is the probability that a gesture is recognised correctly. The system has achieved 93.6%, 96.9%, 97.9% and 90.4% recall for Naive Bayes, Neural Network, Support Vector Machine and Hidden Markov Model classifier respectively when Kinect body frame (N) is set to 10.
We can see the value of recall rising with the increase of N (table 4.10). On the other hand decrease in N gives a poor TP rate. The number of frame N=10 is the best choice if we make a tradeoff to accuracy, recall and precision with system complexity.

The recall graph for the system is described in figure 4.13 for different values of N.
Figure 4.11: Recall Column Chart for third 5 Gestures

Figure 4.12: Recall Column Chart for fourth 5 Gestures

(6,8,10,12,14).
Table 4.10: Recall for Different Value of N

<table>
<thead>
<tr>
<th>N</th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.591</td>
<td>0.574</td>
<td>0.646</td>
<td>0.443</td>
</tr>
<tr>
<td>8</td>
<td>0.845</td>
<td>0.817</td>
<td>0.884</td>
<td>0.743</td>
</tr>
<tr>
<td>10</td>
<td>0.936</td>
<td>0.969</td>
<td>0.979</td>
<td>0.904</td>
</tr>
<tr>
<td>12</td>
<td>0.939</td>
<td>0.972</td>
<td>0.980</td>
<td>0.916</td>
</tr>
<tr>
<td>14</td>
<td>0.943</td>
<td>0.977</td>
<td>0.983</td>
<td>0.921</td>
</tr>
</tbody>
</table>

Figure 4.13: Recall Graph for different values of N

4.3.4 FP Rate

Gesture wise FP rate for Naive Bayes, Support Vector Machine, Hidden Markov Model and Neural Network-based classifier is given in table 4.11.

The column chart is shown in four parts in figure 4.14, 4.15, 4.16, 4.17. FP rates of all four classifiers decreases over N (table 4.12). Average FP rate over different N (6,8,10,12,14) is given in table 4.12 and the graph is shown in figure 4.18.
Table 4.11: FP Rate for Different Gestures

<table>
<thead>
<tr>
<th>Gesture Id</th>
<th>Gesture Name</th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I Have Command</td>
<td>0.002</td>
<td>0.001</td>
<td>0.001</td>
<td>0.005</td>
</tr>
<tr>
<td>2</td>
<td>All Clear</td>
<td>0.004</td>
<td>0.003</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>3</td>
<td>Not Clear</td>
<td>0.004</td>
<td>0.002</td>
<td>0.001</td>
<td>0.004</td>
</tr>
<tr>
<td>4</td>
<td>Spread Wings</td>
<td>0.000</td>
<td>0.001</td>
<td>0.000</td>
<td>0.006</td>
</tr>
<tr>
<td>5</td>
<td>Fold Wings</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.002</td>
</tr>
<tr>
<td>6</td>
<td>Lock Wings</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>7</td>
<td>Up Hook</td>
<td>0.009</td>
<td>0.001</td>
<td>0.001</td>
<td>0.004</td>
</tr>
<tr>
<td>8</td>
<td>Down Hook</td>
<td>0.010</td>
<td>0.001</td>
<td>0.001</td>
<td>0.004</td>
</tr>
<tr>
<td>9</td>
<td>Remove Chocks</td>
<td>0.000</td>
<td>0.001</td>
<td>0.001</td>
<td>0.004</td>
</tr>
<tr>
<td>10</td>
<td>Insert Chocks</td>
<td>0.000</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>11</td>
<td>Move Ahead</td>
<td>0.004</td>
<td>0.001</td>
<td>0.000</td>
<td>0.006</td>
</tr>
<tr>
<td>12</td>
<td>Turn Left</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.004</td>
</tr>
<tr>
<td>13</td>
<td>Turn Right</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.004</td>
</tr>
<tr>
<td>14</td>
<td>Slow Down</td>
<td>0.009</td>
<td>0.002</td>
<td>0.001</td>
<td>0.004</td>
</tr>
<tr>
<td>15</td>
<td>Stop</td>
<td>0.002</td>
<td>0.002</td>
<td>0.002</td>
<td>0.011</td>
</tr>
<tr>
<td>16</td>
<td>Fire</td>
<td>0.001</td>
<td>0.001</td>
<td>0.002</td>
<td>0.010</td>
</tr>
<tr>
<td>17</td>
<td>Brakes On</td>
<td>0.012</td>
<td>0.009</td>
<td>0.003</td>
<td>0.009</td>
</tr>
<tr>
<td>18</td>
<td>Brakes Off</td>
<td>0.008</td>
<td>0.006</td>
<td>0.004</td>
<td>0.011</td>
</tr>
<tr>
<td>19</td>
<td>Next Marshaler</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.006</td>
</tr>
<tr>
<td>20</td>
<td>Install Tiedown</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
Figure 4.14: FP Rate Column Chart for first 5 Gestures

Figure 4.15: FP Rate Column Chart for second 5 Gestures
Figure 4.16: FP Rate Column Chart for third 5 Gestures

Figure 4.17: FP Rate Column Chart for fourth 5 Gestures
Table 4.12: FP Rate for Different Value of N

<table>
<thead>
<tr>
<th>N</th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.482</td>
<td>0.437</td>
<td>0.406</td>
<td>0.600</td>
</tr>
<tr>
<td>8</td>
<td>0.152</td>
<td>0.182</td>
<td>0.102</td>
<td>0.241</td>
</tr>
<tr>
<td>10</td>
<td>0.063</td>
<td>0.031</td>
<td>0.02</td>
<td>0.093</td>
</tr>
<tr>
<td>12</td>
<td>0.059</td>
<td>0.029</td>
<td>0.011</td>
<td>0.090</td>
</tr>
<tr>
<td>14</td>
<td>0.057</td>
<td>0.023</td>
<td>0.010</td>
<td>0.090</td>
</tr>
</tbody>
</table>

Figure 4.18: FP Rate Graph for different values of N
4.3.5 F-measure

Gesture wise F-measure for Naive Bayes, Support Vector Machine, Hidden Markov Model and Neural Network is given in table 4.13. The column chart is shown in four parts in figure 4.19, 4.20, 4.21 and 4.22.

Table 4.13: F-measure for Different Gestures

<table>
<thead>
<tr>
<th>Gesture Id</th>
<th>Gesture Name</th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I Have Command</td>
<td>0.961</td>
<td>0.987</td>
<td>0.991</td>
<td>0.942</td>
</tr>
<tr>
<td>2</td>
<td>All Clear</td>
<td>0.931</td>
<td>0.953</td>
<td>0.971</td>
<td>0.954</td>
</tr>
<tr>
<td>3</td>
<td>Not Clear</td>
<td>0.930</td>
<td>0.958</td>
<td>0.978</td>
<td>0.954</td>
</tr>
<tr>
<td>4</td>
<td>Spread Wings</td>
<td>1.000</td>
<td>0.987</td>
<td>0.994</td>
<td>0.948</td>
</tr>
<tr>
<td>5</td>
<td>Fold Wings</td>
<td>1.000</td>
<td>0.994</td>
<td>0.999</td>
<td>0.983</td>
</tr>
<tr>
<td>6</td>
<td>Lock Wings</td>
<td>1.000</td>
<td>1.000</td>
<td>0.996</td>
<td>0.961</td>
</tr>
<tr>
<td>7</td>
<td>Up Hook</td>
<td>0.820</td>
<td>0.982</td>
<td>0.993</td>
<td>0.939</td>
</tr>
<tr>
<td>8</td>
<td>Down Hook</td>
<td>0.826</td>
<td>0.982</td>
<td>0.988</td>
<td>0.927</td>
</tr>
<tr>
<td>9</td>
<td>Remove Chocks</td>
<td>1.000</td>
<td>0.976</td>
<td>0.974</td>
<td>0.932</td>
</tr>
<tr>
<td>10</td>
<td>Insert Chocks</td>
<td>1.000</td>
<td>0.980</td>
<td>0.976</td>
<td>0.931</td>
</tr>
<tr>
<td>11</td>
<td>Move Ahead</td>
<td>0.892</td>
<td>0.974</td>
<td>0.990</td>
<td>0.940</td>
</tr>
<tr>
<td>12</td>
<td>Turn Left</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.919</td>
</tr>
<tr>
<td>13</td>
<td>Turn Right</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.922</td>
</tr>
<tr>
<td>14</td>
<td>Slow Down</td>
<td>0.869</td>
<td>0.968</td>
<td>0.988</td>
<td>0.889</td>
</tr>
<tr>
<td>15</td>
<td>Stop</td>
<td>0.955</td>
<td>0.978</td>
<td>0.975</td>
<td>0.847</td>
</tr>
<tr>
<td>16</td>
<td>Fire</td>
<td>0.973</td>
<td>0.984</td>
<td>0.972</td>
<td>0.836</td>
</tr>
<tr>
<td>17</td>
<td>Brakes On</td>
<td>0.793</td>
<td>0.852</td>
<td>0.921</td>
<td>0.755</td>
</tr>
<tr>
<td>18</td>
<td>Brakes Off</td>
<td>0.774</td>
<td>0.841</td>
<td>0.918</td>
<td>0.718</td>
</tr>
<tr>
<td>19</td>
<td>Next Marshaler</td>
<td>1.000</td>
<td>0.989</td>
<td>0.995</td>
<td>0.877</td>
</tr>
<tr>
<td>20</td>
<td>Install Tiedown</td>
<td>1.000</td>
<td>0.994</td>
<td>0.977</td>
<td>0.907</td>
</tr>
</tbody>
</table>
Average F-measure over different N is given in table 4.14 and the graph is shown in figure 4.23.
Figure 4.21: F-measure Column Chart for third 5 Gestures

Figure 4.22: F-measure Column Chart for forth 5 Gestures
Table 4.14: F-measure for Different Value of N

<table>
<thead>
<tr>
<th>N</th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.601</td>
<td>0.587</td>
<td>0.651</td>
<td>0.449</td>
</tr>
<tr>
<td>8</td>
<td>0.848</td>
<td>0.820</td>
<td>0.889</td>
<td>0.751</td>
</tr>
<tr>
<td>10</td>
<td>0.936</td>
<td>0.969</td>
<td>0.980</td>
<td>0.905</td>
</tr>
<tr>
<td>12</td>
<td>0.940</td>
<td>0.975</td>
<td>0.981</td>
<td>0.917</td>
</tr>
<tr>
<td>14</td>
<td>0.949</td>
<td>0.986</td>
<td>0.982</td>
<td>0.922</td>
</tr>
</tbody>
</table>

Figure 4.23: F-measure Graph for different values of N
4.3.6 Accuracy

The accuracy graph for the system is the most important one. The vertical axis represents different N values. Figure 4.24 represents the accuracy graph for Naive Bayes, Support Vector Machine, Hidden Markov Model and Neural Network classifier for different values of N. The graph is formed using the data used in table 4.15.

Table 4.15: Accuracy for Different Values of N

<table>
<thead>
<tr>
<th>N</th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>52.2</td>
<td>56.8</td>
<td>59.9</td>
<td>40.5</td>
</tr>
<tr>
<td>8</td>
<td>85.3</td>
<td>82.2</td>
<td>89.8</td>
<td>76.4</td>
</tr>
<tr>
<td>10</td>
<td>93.7</td>
<td>97</td>
<td>98.1</td>
<td>90.8</td>
</tr>
<tr>
<td>12</td>
<td>94.1</td>
<td>97.2</td>
<td>98.9</td>
<td>91.1</td>
</tr>
<tr>
<td>14</td>
<td>94.1</td>
<td>97.8</td>
<td>99</td>
<td>91.1</td>
</tr>
</tbody>
</table>

Figure 4.24: Accuracy Graph for Different Values of N

From the graph of figure 4.24, we can see that the highest accuracy achieved for Naive Bayes is 94.1% (for N = 12 and N=14), for Neural Network is 97.8% (for N = 14), for Support Vector Machine is 99% (for N=14) and for Hidden Markov Model is
94.1% (for N=12 and N=14). Though, starting from N=10, a reasonable accuracy has been achieved for all the classifiers.

It is notable that, the performance for Hidden Markov Model and Naive Bayes classifier is significantly less accurate than Neural Network and Support Vector Machine. Naive Bayes classifier makes an naive assumption that the features (joint coordinates in our work) are independent of each other. But in case of gestures, the joints are hugely dependent. For example, a hand joint position will always depend on the position of arm joint. Same thing is true for all the other joints. For this reason, Naive Bayes performs worse than other classifiers. Hidden Markov Model gave almoast 100% accuracy for some gesture but it performs very poorly for the other gesture with a accuracy of almoast 50% which decease the overall performance of the Hidden Markov Model classifier. As Neural Network uses non linear regression, it is capable of building complex models. On the other hand we used linear kernel Support Vector Machine which works as a sequence classifier as our input data is long sequence of integer value and the integer are not independent to each other. Thats why, in our work, Support Vector Machine and Neural Network based classifier outperforms Naive Bayes classifier and Hidden Markov Model.

4.3.7 $\kappa$-value(Kappa Value)

Table 4.17 presents all possible $\kappa$-value for Naive Bayes, Hidden Markov Model, Support Vector Machine and Neural Network.

We can see from table , almoast all the $\kappa$-values are over 80% which conclude that the all four classifier’s result agreed with each other.

4.3.7.1 Average Response Time

Table 4.17 presents the average response time for Naive Bayes, Hidden Markov Model, Support Vector Machine and Neural Network.

We can see that, for all classifier, average response time is below 100 ms. Naive Bayes classifier is more faster than other classifiers. So the proposed system is capable of responding in real-time.
Table 4.16: $\kappa$-value(Kappa Value) for $N = 10$

<table>
<thead>
<tr>
<th></th>
<th>Naive Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>1.0</td>
<td>0.83</td>
<td>0.91</td>
<td>0.78</td>
</tr>
<tr>
<td>Neural Network</td>
<td>0.83</td>
<td>1.0</td>
<td>0.96</td>
<td>0.88</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>0.91</td>
<td>0.96</td>
<td>1.0</td>
<td>0.89</td>
</tr>
<tr>
<td>Hidden Markov Model</td>
<td>0.78</td>
<td>0.88</td>
<td>0.89</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 4.17: Average Response Time for Different Classifiers

<table>
<thead>
<tr>
<th>N</th>
<th>Nave Bayes</th>
<th>Neural Network</th>
<th>Support Vector Machine</th>
<th>Hidden Markov Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>10.7</td>
<td>17.8</td>
<td>15.3</td>
<td>25.7</td>
</tr>
<tr>
<td>8</td>
<td>12.0</td>
<td>22.1</td>
<td>20.9</td>
<td>26.6</td>
</tr>
<tr>
<td>10</td>
<td>16.1</td>
<td>29.5</td>
<td>27.1</td>
<td>32.3</td>
</tr>
<tr>
<td>12</td>
<td>24.4</td>
<td>44.9</td>
<td>36.2</td>
<td>46.0</td>
</tr>
<tr>
<td>14</td>
<td>33.5</td>
<td>71.8</td>
<td>53.0</td>
<td>74.8</td>
</tr>
</tbody>
</table>

4.4 Summary

In this chapter we have presented experimental setup and results with appropriate discussion. In the next chapter we will conclude this thesis mentioning the contribution of this research, limitation of the proposed system and future works related to this research.
Chapter 5

Conclusion

Building a natural human-machine interaction is important for developing intelligent machine. The aim of this thesis was to develop a computer vision-based real-time 3D gesture recognition. In this thesis, we presented a novel method for continuous gesture recognition that support a challenging dataset (NATOPS database) for aircraft handling.

In section 5.1 contributions of the proposed system are presented. Section 5.2 describes the limitations of the proposed system regarding 3D gesture recognition. Finally section 5.3 focuses on future work and scope to extend this research.

5.1 Contributions

In this thesis, we designed and implemented a computer vision-based real-time gesture recognition system that uses information obtained from 3D body and hand movement. The output of this research can be used for interpreting the Naval Air Training and Operating Procedures Standardization (NATOPS) aircraft handling signals[10]. However, many of the previous approaches to gesture recognition consider either body pose or hand pose alone, limiting their practicality for many real-world problems. In this work, we combine body and hand poses, allowing gesture recognition to deal with a richer gesture vocabulary, extending its practicality.

A Kinect sensor is used to capture 3D images. Kinect along with Windows version
2.0 SDK is used for collecting 3D joint position and depth information. The developed application as a main part of the thesis provides an ability to recognize aircraft handling signals as like an aircraft carrier flight deck environment.

For 3D body pose estimation, the system used Kinect 3D joint points and calculated motion trajectory for accuracy. Image frame sequences are used to represent motion gestures, where each frame is mapped to a matrix of 3D (X,Y,Z) coordinates of 25 body joints of a human. From mapped data for 3D position of 25 joints, the system established a skeleton coordinate system. Mapped data are scaled for developing a system which is invariant of person’s distance from the sensor, height and size.

The system is evaluated on a real-world scenario: we tested the performance of this gesture recognition system is evaluated with a subset of the NATOPS aircraft handling signals, a challenging gesture vocabulary that involves both body and hand pose articulations. Experimental result show that combining body and hand poses significantly improved the gesture recognition accuracy. The system distinguished the performance of different types of gestures. Motion Trajectory along with the body joint coordinates used as feature to recognize 3D motion gestures.

Four pattern classification algorithms such as Naive Bayes, Neural Network, Hidden Markov Model and Support Vector Machine has been used to train and test the system. The accuracy of the system is 93.7% in Nave Bayes classifier, 98.1% in Support Vector Machine, 97% in Neural Network and 90.8% in Hidden Markov Model. Considering the accuracy and complexity of the system with respect to the number of frame(N), it can be concluded that the best response time of the system is 16.1 ms in Naive Bayes, 29.5 ms in Neural Network, 27.1 ms in Support Vector Machine, 32.3 ms in Hidden Markov Model. Also to test the consistency of output of all the four classifiers kappa-test has been applied to the system and the result was satisfactory with kappa value 0.91.

5.2 Limitation of the System

To use Kinect sensor to detect full body of a human, the distance between Kinect sensor and human needs to be at least 1.5 meters. Also as Kinect uses Infrared light to detect human, if the distance it more than 6 meters, then performance will deteriorate.
While tracking the skeleton of a human, the inherent limitations of Kinect also limits the proposed system of gesture recognition. Proper dress makes it easier to track human. In case of females wearing long traditional clothes, tracking is not accurate for leg joints.

Kinect does not have a good accuracy in detecting open or close fist. For this reason the system have some difficulties to distinguish between the gesture-17 (brakes on) and gesture-18 (brakes off) (as shown in figure 3.32 and 3.35).

5.3 Future Works

The future work consists of advancing in the different interconnected lines of research presented in this thesis work belonging to pattern matching and computer vision fields. For the feature level, it could be interesting to compare with more descriptors and methods for emphasize the benefits. In addition, we think that it would be interesting to train the classifier with more data.

This gesture recognition system assumes no background knowledge or context information; it performs the recognition solely based on the observation. However, humans make extensive use of these to perform many types of inference tasks, allowing them to make decisions rapidly or to come up with an answer that is impossible to infer solely based on the observation. This intuition has led to a large body of research that concerns modeling and integrating human knowledge into the system. Using this context information might improve the performance of this system in many ways.

For gesture recognition in the context of the flight deck environment, there is a certain sequence of customary gestures, and we can reason about what gestures would make sense based on the context. For example, it is less likely that an ABH (Aviation Boatswains Mate Handlers) will command “brakes off to a pilot while taxing an aircraft, because the brake should have been already off while moving. Therefore, we believe that a gesture recognition system that is able to consider the context information (i.e., reason about routine procedures on the flight deck) can improve its performance.

Similarly, in each gesture there are a certain set of body and hand poses we can expect, so once we know which gesture is being performed or what gestures we can
expect next, we can narrow down the search space, which can improve the performance of a gesture classifier. To do so, we need an approach to describing the knowledge in a concise and consistent manner. Also, there needs to be an easy way to add or modify the knowledge manually, or even a way to reason about new things automatically based on the existing knowledge. It is important that the whole process should not be ad-hoc, i.e., it should be generalizable to many types of context information.

Most of the current approaches concentrate on learning from observations. Designing and implementing a statistical estimation and inference framework that incorporates the context information is also an interesting future direction of this work.

Allowing two-way communication between humans and intelligent machine is of particular interest in this work. In order to provide natural gesture-based interaction, it is important for a system to be able to recognize human gestures. At the same time, it is also necessary for the system to have an appropriate feedback mechanism, that is, the machine have to be able to response with gesture, just as a human would do in the same situation. This concept can be a future task for the research work.
References


[46] Suguru Goto and Fuminori Yamasaki. Integration of percussion robots” robot-music” with the data-suit” bodysuit”: technological aspects and concepts. In


[71] Wei-Lwun Lu and James J Little. Tracking and recognizing actions at a distance. In *Proceedings of the ECCV Workshop on Computer Vision Based Analysis in Sport Environments (CVBASE06), Graz, Austria*, volume 1, 2006.


[81] Raphael Labayrade, Didier Aubert, and Jean-Philippe Tarel. Real time obstacle detection in stereovision on non flat road geometry through” v-disparity” representation. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 646–651. IEEE, 2002.


[104] Logan Scott and Juan Sebastian Casallas. Wedjat: Project proposal.


