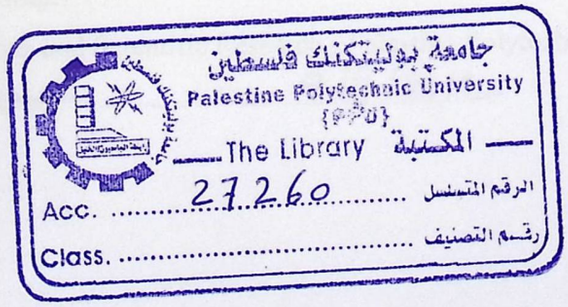
Palestine Polytechnic University

Deanship of Graduate Studies and Scientific Research
Master of Informatics

Real-Time Gesture Recognition Using 3D Images

Submitted by:
Osama 'Mohammad Yaser' Dweik

In Partial Fulfillment of the Requirements for the Degree Master of Informatics
June 2012



The undersigned hereby certify that they have read and recommend to the Deanship of Graduate Studies and Scientific Research at Palestine Polytechnic University the acceptance of a thesis entitled: **Real-Time Gesture Recognition using 3D Images** submitted by **Osama M.Yaser Dweik** in partial fulfillment of the requirements for the degree of Master in Informatics Graduate Advisory Committee.

Committee Member Name (Supervisor), University:

Dr. Hashem Tamimi, Palestine Polytechnic University

Signature:..... Date: 29/08/2012

Committee Member Name, University:

Dr. Faisal Khamayseh, Palestine Polytechnic University

Signature: Date: 3/9/2012

Committee Member Name, University:

Dr. Rasheed Jayousi, Al-Quds University

Signature:..... Date: 2/9/2012

Approved for the Deanship:

Dean of Graduate Studies and Scientific Research - Palestine Polytechnic University

Signature:..... Date: 9/9/2012

Abstract

KINECT has been recently introduced in the market as a low cost 3D acquisition device, so it will be interesting to discover the power of this device when we use it for gesture recognition. In this thesis, we propose a real-time gesture recognition system using 3D sensor that transforms gestures into a set of useful words using different machine learning algorithms and taking into consideration temporal features. A depth image, which is provided by KINECT, will be used to construct a skeleton of the human body. We have used Nearest Neighbor (NN) with different distance formulas, Self Organizing Map (SOM) and Hidden Markov Model (HMM) for recognition. The result of this thesis using the 10 fold cross validation shows that HMM may provide recognition accuracy up to 96 percent, while using NN algorithm with Spearman distance we can obtain around 90 percent accuracy and around 75 percent of accuracy using the SOM algorithm. All three algorithms has works in real-time.

Abstract

KINECT has been recently introduced in the market as a low cost 3D acquisition device, so it will be interesting to discover the power of this device when we use it for gesture recognition. In this thesis, we propose a real-time gesture recognition system using 3D sensor that transforms gestures into a set of useful words using different machine learning algorithms and taking into consideration temporal features. A depth image, which is provided by KINECT, will be used to construct a skeleton of the human body. We have used Nearest Neighbor (NN) with different distance formulas, Self Organizing Map (SOM) and Hidden Markov Model (HMM) for recognition. The result of this thesis using the 10 fold cross validation test, HMM may provide recognition accuracy up to 96 percent, while using NN with Spearman distance we can obtain around 90 percent accuracy and 87 percent of accuracy using the SOM algorithm. All three algorithms work in real-time.

الملخص

تهدف هذه الدراسة الى بحث وتحليل مدى القدرة الممكن الحصول عليها من خلال استخدام جهاز KINECT للتعرف على الحركات والإيماءات التي يقوم بها الإنسان.لقد تم في هذه الرسالة العمل على استخدام مجموعة مختلفة من خوارزميات تعلم الآلة وهي خوارزمية الجار الأقرب (Nearest Neighbor) , وخوارزمية خريطة التنظيم الذاتي (Self-organizing Map) ونموذج ماركوف المخفي (Hidden Markov Model) مع الأخذ بعين الاعتبار الخصائص الزمنية المفيدة لعملية التعرف على الحركات ودراسة هذه الخوارزميات ومقارنتها وتحليلها من خلال استخدامنا للصورة ثلاثية الأبعاد واستخراج الهيكل الممثل للجسم البشري وتزويد البيانات الناتجة للخوارزميات الثلاث.واستطعنا من خلال هذه الدراسة تحقيق نتائج بدقة تصل الى 96 بالمائة باستخدام نموذج ماركوف المخفي و حوالي 90 بالمائة باستخدام خوارزمية الجار الأقرب وحوالي 75 بالمائة باستخدام خوارزمية خريطة التنظيم الذاتي وقد تم فحص صحة النتائج باستخدام خوارزمية التحقق من الصحة (10 fold cross validation).كذلك استطعنا تحقيق عملية التعرف على الحركات والإيماءات التي يقوم بها الإنسان في وقت حقيقي متزامن باستخدام الخوارزميات الثلاث.

DECLARATION

I declare that the Master Thesis entitled "Online Gesture Recognition Using 3D Images" is my own original work, and hereby certify that unless stated, all work contained within this thesis is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgment is made in the text.

Osama Mohammad Yaser Dweik

Signature:..... Date: 9.Sep.2012

DECLARATION

I declare that the Master Thesis entitled "Online Gesture Recognition Using 3D Images" is my own original work, and hereby certify that unless stated, all work contained within this thesis is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgment is made in the text.

Osama Mohammad Yaser Dweik

Signature:..... Date: 9.Sep.2012

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for the master degree in Informatics at Palestine Polytechnic University, I agree that the library shall make it available to borrowers under rules of the library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made. Permission for extensive quotation from, reproduction, or publication of this thesis may be granted by my main supervisor, or in his absence, by the Dean of Graduate Studies and Scientific Research when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Osama Mohammad Yaser Dweik

Signature:..... Date: 9.Sep.2012

DEDICATION

I dedicate this work to my parents and my wife. Without their patience, understanding, there continuous support, and most of all love, the completion of this work would not have been possible during my vital educational years. Also their endless patience and encouragement when it was most required.

ACKNOWLEDGMENT

I would like to express my deep-felt gratitude to my advisor, Dr. Hashem Tamimi for his advice, encouragement, enduring patience and constant support. He continuously provided me with help, encouragement, and with extensive knowledge. I also wish to thank the other members of my committee, Dr. Feisal Khamayseh and Dr.Rasheed Jayoosy for their suggestions, comments and additional guidance were invaluable to the completion of this work.

Additionally, I want to thank the Palestine Polytechnic University Computer Science Department staff for all their hard work and dedication, providing me the means to complete my degree and prepare for a career as a computer scientist. Also I want to thank the following individuals for their help: My brother: Abdullah Dweik. My friend: Hamzah Karajah. My cousine: Hadeel Tamimi.

Finally, I must thank my parents and my dear wife for putting up with me during the development of this work with continuing, loving support and no complaint.

NOTE: This thesis was submitted to my Supervising Committee on the May 31, 2012.

ACKNOWLEDGMENT

I would like to express my deep-felt gratitude to my advisor, Dr. Hashem Tamimi for his advice, encouragement, enduring patience and constant support. He continuously provided me with help, encouragement, and with extensive knowledge. I also wish to thank the other members of my committee, Dr. Feisal Khamayseh and Dr.Rasheed Jayoosy for their suggestions, comments and additional guidance were invaluable to the completion of this work.

Additionally, I want to thank the Palestine Polytechnic University Computer Science Department staff for all their hard work and dedication, providing me the means to complete my degree and prepare for a career as a computer scientist. Also I want to thank the following individuals for their help: My brother: Abdullah Dweik. My friend: Hamzah Karajah. My cousine: Hadeel Tamimi.

Finally, I must thank my parents and my dear wife for putting up with me during the development of this work with continuing, loving support and no complaint.

NOTE: This thesis was submitted to my Supervising Committee on the May 31, 2012.

Contents

Abstract v

Arabic Abstract vi

1 Introduction 1

1.1 Gesture recognition 1

1.1.1 Problem definition of gesture recognition in general 4

1.1.2 Contribution 4

1.1.3 Thesis structure 5

2 Theoretical background 7

2.1 Review to machine learning approaches 7

2.1.1 K-Mean clustering 8

2.1.2 Nearest Neighbor(NN) classification 10

2.1.3 Self Organizing Map (SOM) 14

2.1.4 Hidden Markov Model (HMM) 15
2.2 Cross validation 24
2.3 KINECT and Infra-red technology 25
2.3.1 KINECT sensor overview 25
2.3.2 KINECT limitation 29
2.3.3 Body part recognition using KINECT 30
2.3.4 Data collection and capturing: 30

3 Literature review **34**
3.1 Gloves: 35
3.2 Stereo camera: 37
3.3 3D camera: 39
3.4 KINECT: 41

4 Gesture Recognition using KINECT **45**
4.1 Data acquisition using KINECT as a 3D imaging device. 46
4.2 Scale and transition invariant: 49
4.3 Gesture recognition using Nearest Neighbor (NN): 51
4.3.1 Data preparation 53
4.3.2 Apply Nearest neighbor algorithm 53
4.4 Gesture recognition using Self Organizing Map (SOM): 54
4.4.1 Data preparation: 54

4.4.2 Training phase: 55
4.4.3 Testing Phase: 56
4.5 Gesture recognition using hidden markov model (HMM): 56
4.5.1 Data preparation: 57
4.5.2 Training Phase: 58
4.5.3 Testing Phase: 60
4.5.4 Incremental approach 61
4.6 Online gesture recognition using HMM: 62
4.6.1 Building the models: 62
4.6.2 Capturing data: 62
4.6.3 Two enhancements: 63

5 Experimental results **65**

5.1 Gesture recognition using Nearest Neighbor algorithm 66
5.2 Gesture recognition using Self Organizing Map(SOM) algorithm . . . 68
5.3 Gesture Recognition using Hidden Markov Model(HMM) algorithm . 70
5.4 Time analysis 72

6 Conclusion and future work **73**

6.1 Conclusions 73
6.2 Future work 75

A Appendix 83

A.1 Preparing the KINECT 83

List of Figures

2.1 The induction step of the forward algorithm [4] 23

2.2 KFold algorithm where $K = 3$ [32] 25

2.3 KINECT sensor 26

2.4 Camera with infrared sensor sends the infrared beams 27

2.5 3D image 27

2.6 Skeleton joints relative to the human body. [20] 29

2.7 3D Image features 32

2.8 Random decision forests. 33

4.1 General block diagram 46

4.2 Skeleton joints relative to the human body. [20] 48

4.3 Depth image from KINECT 48

4.4 Colored image from KINECT 49

4.5 Detected skeleton 49

4.6 Gesture samples 52

4.7 Gesture recognition using NN block Diagram 52
4.8 Gesture recognition using SOM block diagram 54
4.9 SOM Clustering 55
4.10 Gesture recognition using HMM block Diagram 57
4.11 Frames distribution over KMean clusters 58
4.12 Proposed hidden markov models 60
4.13 Incremental approach 61
4.14 Final system screen 64

5.1 Nearest Neighbor algorithm accuracy using different distance methods 67
5.2 Self Organizing Map algorithm accuracy using dimensions 69
5.3 K-Mean clusters 70
5.4 Accuracy of HMM with 1-5 states and tolerance 0.01 71
5.5 Accuracy of HMM with 1-5 states and tolerance 0.01 72

A.1 Maping bins 84
A.2 Second expirement 85

List of Tables

5.1 Results for 11 gestures using HMM 66

5.2 Nearest Neighbor results comparison 68

5.3 Results using SOM 68

5.4 Results for 11 gestures using HMM 71

List of Tables

5.1 Results for 11 gestures using HMM 66

5.2 Nearest Neighbor results comparison 68

5.3 Results using SOM 68

5.4 Results for 11 gestures using HMM 71

List of Algorithms

1 The K-Mean clustering algorithm 10

2 The SOM algorithm 16

3 GR using NN algorithm 53

4 GR using SOM algorithm 56

5 GR Using HMM algorithm 60

Chapter 1

Introduction

Gesture recognition is defined as interpreting human gestures using mathematical algorithms. Gestures can originate from any bodily motion or state [33]. Current focuses in this field include emotion recognition from the face, hand gesture recognition or even body gestures. Many approaches have been made using cameras and computer vision algorithms to interpret sign language. However, the identification and recognition of human behaviors is also the subject of gesture recognition techniques.

1.1 Gesture recognition

In order to point out the main reasons for using gesture recognition we need to list some of basic commonly used gestures in our daily life. For example if we need to stop a taxi, we use a specific gesture, and if we want to say "goodbye" or "hi" or

even point to something we can use an awareness gesture. All these gestures as a motion vary between different countries and cultures. These gestures, which we use every day, can be promoted as part of our language.

One of the important benefits of gesture recognition is that it can provide help and assistance for people with disabilities such as the deaf or anyone who cannot use his voice to communicate or make gestures using his/her body. If we can provide them a way to communicate with other people by interpreting their sign language into spoken words, or even those who don't know sign language, gestures can be very important.

Gesture recognition can also be used in some security and monitoring fields. These days cameras and monitoring devices are found everywhere and with low cost, we can use this technology to improve automated detection for some interesting or suspicious behaviors in many firms or even in the street (emotion recognition).

The types of gesture recognition vary based on the applications it is used for, and the human body part it recognizes. These situations and variables determine the device we may use. Some gestures, which are based on hand motion can be detected and recognized using hand gloves, other motions like face gestures and expressions need cameras. When tracking the whole human body gesture we may need two or more stereo 2D cameras, or a 3D camera.

Due to the availability and the low cost of the 3D cameras nowadays in the market like KINECT which is a 3D depth imaging device produced by Microsoft in 2010 [20], we have focused on the human body motion and gestures. This type of

recognition collects information about the human body and translates motion into meaningful words humans can understand and work with.

Generally, if we can recognize and translate the body motions collected by any type of sensor as an input for a computer to useful and meaningful words, it may improve the usability of the technology that helps people who have physical limitation.

The use of gesture recognition as an input device for the computer may eliminate the need of some other ordinary input devices such as keyboard, joystick or mouse specially when there is a physical limitation for the user. It can be used for many applications and may work in a more speedy and accurate manner.

Gesture recognition may be used in many applications such as communication strategy. This can be used by the computer for gestures to recognize, understand, actions to perform and even words to translate and speak. This type of gestures may focus on a whole body motion or facial expressions or even hand movement. Many types of sensors may be used to apply the communication types for this purpose.

Security fields can detect interesting or suspicious motions of person behavior. Also other applications can be in gaming and entertainment fields. So that it can be used with situations which human and machine needs to communicate without physical connection.

In order to make our experiments and explore the performance of the KINECT sensor for gesture recognition goal, we need first to collect the 3D images and make some processing and normalization, then we need to transform the processed infor-

mation into 3D skeleton to understand the gesture in the skeleton using machine learning approach taking in consideration the context of this gesture in order to translate the gesture meaning to the user.

1.1.1 Problem definition of gesture recognition in general

Using KINECT sensor and the corresponding part recognition algorithm, a skeleton of N joints is produced from each frame. Each joint is represented as (x, y, z) point in 3D space. A gesture is to be recognized from a sequence of V frames using a gesture recognition approach L. This approach L will classify each gesture into C classes. The gesture recognition is to find L such that $L(X_i) = c_j$; with maximum recognition rate. Where X_i is the sequence of $N \times V$ points, and $c_j \in C$ gesture class.

1.1.2 Contribution

Our contribution in this thesis is to obtain and achieve a gesture recognition system using the KINECT sensor under real-time constraints ($\approx$ 30fps). We have implemented a gesture recognition system using different algorithms and a KINECT sensor. The algorithms are Nearest Neighbor Algorithm (NN) with different distance measuring methods, Self Organizing Map (SOM) and Hidden Markov Model (HHM).

1.1.3 Thesis structure

Chapter 2 is concerned with the main theoretical background needed for the reader to understand the next Chapters. It is divided into five main sections that should cover all topics as follows: Clustering, KFold cross validation, Temporal Models, KINECT and Infrared technology and Building skeleton.

Chapter 3 reviews some literature for gesture recognition and presents the previous research work. Presenting What researchers used in their study like colored gloves or stereo cameras and others used depth cameras. Many researchers focused on hand recognition, while some studied the human body gestures. Also we show what methodology researchers use, some used Hidden Markov Model to achieve their goal, while others tried to use a simpler algorithm like nearest neighbor classification and the similarity measure.

Chapter 4 presents our work and how data was acquired, collected and processed to reach the recognition of gestures. This describes our work using different machine learning algorithms. We show the process of creating the feature vectors and how we process these features in order to implement the gesture recognition system using KINECT.

We show the results we achieved in our experiments in Chapter 5. We have list the results we obtained from using the Nearest Neighbor, Self Organizing Map and Hidden Markov Model algorithms.

Chapters 6, summarize our research in this chapter. Also we mentioned and suggested future work which could be done to increase the knowledge in this field

of study.

Chapter 2

Theoretical background

This chapter is concerned with the main theoretical background needed for the reader to understand the next chapters. The theoretical background is divided into four main sections that should cover all topics we need. First we mention the clustering algorithms which we used to cluster and label the data into classes. Then we describe the KINECT as a 3D sensor which we have used. After that we present the cross validation method used in our experiments. Finally, we present the technology and algorithms for the body part labeling and building skeleton.

2.1 Review to machine learning approaches

Machine learning approaches are categorized either as clustering or classification. Clustering is defined as unsupervised classification of patterns, observations, measurements, data items, or feature vectors into groups called clusters [15]. It is a

Chapter 2

Theoretical background

This chapter is concerned with the main theoretical background needed for the reader to understand the next chapters. The theoretical background is divided into four main sections that should cover all topics we need. First we mention the clustering algorithms which we used to cluster and label the data into classes. Then we describe the KINECT as a 3D sensor which we have used. After that we present the cross validation method used in our experiments. Finally, we present the technology and algorithms for the body part labeling and building skeleton.

2.1 Review to machine learning approaches

Machine learning approaches are categorized either as clustering or classification. Clustering is defined as unsupervised classification of patterns, observations, measurements, data items, or feature vectors into groups called clusters [15]. It is a

technique that allows objects with similar characteristics to be grouped together in order to facilitate their further processing [30]. The clustering problem has been addressed in many contexts and by researchers in many disciplines. K-Mean and Self Organizing Map (SOM) are examples for clustering techniques used for solving problems.

On the other hand classification is supervised learning approach that simply provided with two sets of data, training and testing sets. The training set is used to learn from labeled samples so that to identify unlabeled examples on testing set using highest accuracy. So the goal is for the learner to identify the elements in the test set and give them the appropriate label. Multiple different approaches are used for classification [24]. The following sections introduce three classification methods: Nearest Neighbor(NN) and Hidden Markov Model (HMM).

2.1.1 K-Mean clustering

Clustering is a widely studied problem, it has been used in many applications. The K-Mean method is very efficient in producing good clustering results for many practical applications among some cluster algorithms. In K-mean algorithm, the number of clusters K is determined before clustering (i.e. means before the implementation), the number of clusters in the existing data is known. In this way, K-means is not a complete solution for data clustering (some extension of K-mean called X-mean can estimate K) [28]. However, some prior knowledge of number of clusters is required, for instance, we can determine the number of clusters needed experimentally.

KMean is a simple clustering unsupervised machine learning algorithm [1], the process is based on partitioning an N-Dimensional data into K sets on the basis of a sample.

The K-Mean starts with K groups each consists of a single random point [1].When adding a new point to the group which mean is nearest to the new point, the mean of this group is modified taking in consideration the new added point. After adding all points to their nearest group based on the mean, we will have K groups with K centers. Every point will be labeled by its nearest group.

Let the k prototypes $(\omega_1, \omega_2, \ldots, \omega_k)$ be initialized to one of the n input examples $(i_1, i_2, \ldots, i_n)$. In this case, n is the number of pixels which has the higher response. Each example i_l is equal to one of the prototypes ω_j. The condition is $k \ll n$. Because the number of prototypes must be greatly less than the number of input examples, so that the clustering algorithm can be effective, but also assuming that well defined clusters exist. The principle of K-mean algorithm is to assign each example to different cluster and give the prototypes that are also the centroids of clusters. In n input examples, there may be K clusters. C_j represents the j^th $(0 < j \leq K)$ cluster whose value is a disjoint subset of input examples. See Algorithm 1.

The error is the sum of all the distance. The strategy which can be made is to keep it as small as possible. So the error also can be used to evaluate the performance of each iteration. The time complexity of the K-Mean algorithm is O(nkl), where n is the number of samples, k is the number of clusters and l is the number of

iterations [29].

Algorithm 1 The K-Mean clustering algorithm

Initialize k prototypes $(\omega_1, \omega_2, \ldots, \omega_k)$, such that each input example is assigned to a certain prototype, which is $\omega_j = i_l \quad j \in \{1, 2, \ldots, k\} \quad l \in \{1, 2, \ldots, n\}$.
Each cluster C_j is represented by the prototype ω_j. examples i_l, where $l \in \{1, 2, \ldots, n\}$.
while The error does not change significantly. **do**
Find the nearest prototype ω_{j*} with certain distance measurements by

$$\|i_l - \omega_{j*}\|_2 \leq \|i_l - \omega_j\|_2 \quad j \in \{1, 2, \ldots, k\} \tag{2.1}$$

The error is calculated by:

$$E = \sum_{j=1}^{k} \sum_{i_l \in C_j} \|i_l - \omega_j\|^2 \tag{2.2}$$

which is the overall sum of the distances between all the examples and its prototypes.
end while

[29]

2.1.2 Nearest Neighbor(NN) classification

The nearest-neighbor is one of the simplest classification algorithms. The training phase is simply to store every training sample with its label. To make a prediction for a test sample, first compute its distance to every training sample data using one of the distance measuring methods. Then, assign the test sample with the closest training samples label [8]. In the case that that there are two or more same distance points with different class labels, then the most numerous class is chosen.

And various distance measurement methods are currently used when implementing this algorithm. These are discussed in the following section. The Nearest Neighbor searching algorithm has a complexity of $O(n^2)$ where n is the number of samples [12].

2.1.2.1 Common distance functions for NN:

Given a source vector s and a target vector t both of size n, the distances between the vector s and t can be calculated using the following methods [27]:

1. Euclidean distance, also known as L_2 distance($\|.\|_2$):

$$d_{st}^2 = (s-t)(s-t)^T \tag{2.3}$$

2. Standardized Euclidean distance:

$$d_{st}^2 = (s-t)V^{-1}(s-t)^T \tag{2.4}$$

   where V is the $n \times n$ diagonal matrix whose j^{th} diagonal element is $S(j)^2$, where S is the vector containing the inverse weights.

3. Mahalanobis distance:

$$d_{st}^2 = (s-t)C^{-1}(s-t)^T \tag{2.5}$$

   where C is the covariance matrix.

4. City block metric, also known as L_1 distance($\|.\|_1$):

$$d_{st} = \sum_{j=1}^{n} |s_j - t_j| \tag{2.6}$$

5. Minkowski metric:

$$d_{st} = \sqrt[p]{\sum_{j=1}^{n} |s_j - t_j|^p} \tag{2.7}$$

6. Chebychev distance:

$$d_{st} = max_j \left\{ \cdot |s_j - t_j| \right\} \tag{2.8}$$

7. Cosine distance:

$$d_{st} = \left(1 - \frac{s\acute{t}}{\sqrt{(s\acute{s})(t\acute{t})}} \right) \tag{2.9}$$

8. Correlation distance:

$$d_{st} = 1 - \frac{(s - \bar{s})(t - \bar{t})^T}{\sqrt{(s - \bar{s})(s - \bar{s})^T} \sqrt{(t - \bar{t})(t - \bar{t})^T}} \tag{2.10}$$

where

$$\bar{s} = \frac{1}{n} \sum_{j}^{N} s_j \tag{2.11}$$

and

4. City block metric, also known as L_1 distance($\|.\|_1$):

$$d_{st} = \sum_{j=1}^{n} |s_j - t_j| \tag{2.6}$$

5. Minkowski metric:

$$d_{st} = \sqrt[p]{\sum_{j=1}^{n} |s_j - t_j|^p} \tag{2.7}$$

6. Chebychev distance:

$$d_{st} = max_j \left\{ |s_j - t_j| \right\} \tag{2.8}$$

7. Cosine distance:

$$d_{st} = \left(1 - \frac{s\acute{t}}{\sqrt{(s\acute{s})(t\acute{t})}} \right) \tag{2.9}$$

8. Correlation distance:

$$d_{st} = 1 - \frac{(s - \bar{s})(t - \bar{t})^T}{\sqrt{(s - \bar{s})(s - \bar{s})^T}\sqrt{(t - \bar{t})(t - \bar{t})^T}} \tag{2.10}$$

where

$$\bar{s} = \frac{1}{n} \sum_{j}^{N} s_j \tag{2.11}$$

and

$$\bar{t} = \frac{1}{n} \sum_{j}^{N} t_j \tag{2.12}$$

9. Hamming:

$$d_{st} = (\#(s_j \neq t_j)/n) \tag{2.13}$$

where # is the count.

10. Jaccard distance:

$$d_{st} = \frac{\#\left[(s_j \neq t_j) \cap ((s_j \neq 0) \cup (t_j \neq 0))\right]}{\#\left[(s_j \neq 0) \cup (t_j \neq 0)\right]} \tag{2.14}$$

where # is the count.

11. Spearman distance:

$$d_{st} = 1 - \frac{(r_s - \bar{r_s})(r_t - \bar{r_t})^T}{\sqrt{(r_s - \bar{r_s})(r_s - \bar{r_s})^T}\sqrt{(r_t - \bar{r_t})(r_t - \bar{r_t})^T}} \tag{2.15}$$

where

r_{sj} is the rank of s_j taken over $s_{1j}, s_{2j}, ...s_{ns}, s$. r_{tj} is the rank of t_j taken over $t_{1j}, t_{2j}, ...t_{nt}, t$. r_s and r_t are the coordinate-wise rank vectors of s and t,

$$\bar{r_s} = \frac{1}{n} \sum_{j} r_{sj} = \frac{(n+1)}{2} \tag{2.16}$$

$$\bar{r}_t = \frac{1}{n}\sum_j r_{tj} = \frac{(n+1)}{2} \tag{2.17}$$

2.1.3 Self Organizing Map (SOM)

Self Organizing Map (SOM) is unsupervised clustering approach that can be used for classification, it is a type of neural network.

The locations of the responses tend to become ordered to some meaningful coordinates system for different input features, where being created over the network, and the location or coordinates of a cell in the network corresponds to a particular domain of the input signal patterns. It had been used for some tasks like robotics, pattern recognition and even processing of semantic information [21].

Like other neural networks, the SOM has to be trained with samples from the input-space. These samples are represented as input vectors. During the training process neurons compete for inputs. With each training step winning neurons are adjusted to better match the input they receive. Feedback between the neurons allows the entire network to eventually converge to a final state [21].

After training, each neuron in the SOM will represent a portion of the input space. To accomplish this representation each neuron is associated with a parametric reference vector, referred to as a model vector. The length of each model vector is equal to the length of the input vectors, such that each element within a model vector represents a dimension of the input-space. The initial values of the elements are most commonly randomized, such that a mapping of the input-space onto the

initial SOM would have no meaning. Other initializations are possible and may reduce the time required for the map to converge [21].

By specifying the dimensionality of the SOM, it can adjust its weights to match the distribution of the input space, generally as unsupervised approach it will cluster the data over the cells of the map. On the other hand a trained SOM can be used for classification, by providing the SOM with a new input vector, it will specify the Best Matching Unit (BMU) which is the nearest node to the new vector and can be computed using distance function like equation distance for example. So the SOM can label each new vector depending on the location of the BMU cluster label.

The following algorithm describes how SOM works, assuming t is the current iteration, δ:limit on time iteration,$\mathbf{Wv}$: current weight vector, $\mathbf{D}$: target input, $\theta(t)$: restraint due to distance from Best Matching Units (BMU), usually called the neighborhood function and $\alpha(t)$: is learning restraint due to time.

[37]

2.1.4 Hidden Markov Model (HMM)

HMM is a statistical approach, which uses previous time series information to estimate output states name. Rabiner [31], Describes in his paper about HMM that it can be considered as a generalization of a mixture model where the hidden variables (or latent variables), which control the mixture component to be selected for each observation, are related through a Markov process rather than independent of each other.

initial SOM would have no meaning. Other initializations are possible and may reduce the time required for the map to converge [21].

By specifying the dimensionality of the SOM, it can adjust its weights to match the distribution of the input space, generally as unsupervised approach it will cluster the data over the cells of the map. On the other hand a trained SOM can be used for classification, by providing the SOM with a new input vector, it will specify the Best Matching Unit (BMU) which is the nearest node to the new vector and can be computed using distance function like equation distance for example. So the SOM can label each new vector depending on the location of the BMU cluster label.

The following algorithm describes how SOM works, assuming t is the current iteration, δ:limit on time iteration,$\mathbf{Wv}$: current weight vector, $\mathbf{D}$: target input, $\theta(t)$: restraint due to distance from Best Matching Units (BMU), usually called the neighborhood function and $\alpha(t)$: is learning restraint due to time.

[37]

2.1.4 Hidden Markov Model (HMM)

HMM is a statistical approach, which uses previous time series information to estimate output states name. Rabiner [31], Describes in his paper about HMM that it can be considered as a generalization of a mixture model where the hidden variables (or latent variables), which control the mixture component to be selected for each observation, are related through a Markov process rather than independent of each other.

Algorithm 2 The SOM algorithm

Randomize the map nodes weight vectors.
Grab an input vector.
Traverse each node in the map.
Use Euclidean distance to find similarity between the input vector and the map's node's weight vector.
while $t < \delta$ **do**
Track the node that produces the smallest (BMU)).
Update the nodes in the neighborhood of BMU by pulling them closer to the input vector.

$$\mathbf{Wv}(t+1) = \mathbf{Wv}(t) + \theta(t)\alpha(t)(\mathbf{D}(t) - \mathbf{Wv}(t)) \text{ [21]} \tag{2.18}$$

Increase t
end while

Two reasons why HMM became popular. First, the model is very rich with mathematical structures and can form the theoretical basis for use in a wide range of important applications. Second, when the model is applied properly, it works very well in practice for applications [4].

HMMs is used in many applications which interested in predection systems and signal processing. Also it is used with success to low level natural language processing tasks like part-of-speech tagging and capturing information from documents.

The theory is named by the name of Andrei Markov [26] in the early twentieth century, after that Baum and his colleagues developed the theory of HMM in the 1960s [2].

Each state in the model emitted all observation symbols with a finite probability. That makes the model much more expressive statistical tool for modeling

generative sequences that can be characterized by an underlying process generating an observable sequence.

2.1.4.1 Formal Definition:

Given S, the state alphabet set, and V the observation set:

$$S = (s_1, s_2, ..., s_N) \tag{2.19}$$

$$V = (v_1, v_2, ..., v_M) \tag{2.20}$$

Q is defined as a fixed state sequence of length T, and corresponding observations O:

$$Q = q_1, q_2, ..., q_t \tag{2.21}$$

$$O = o_1, o_2, ..., o_t \tag{2.22}$$

The Hidden Markov Model is defined as $\lambda = (A, B, \pi)$ when: A is a transition array which store the probability of state j following state i. State transition probabilities are time independent:

$$A = [a_{ij}], a_{ij} = P(q_t = s_i \mid q_{t-1} = s_i) \tag{2.23}$$

B is the observation array which store the probability of observation k being produced from the state j, Also it is independent of t:

$$B = [b_i(k)], b_i(k) = P(x_t = v_k \mid q_t = s_i) \tag{2.24}$$

and π is the initial probability array for the states:

$$\pi = [\pi_i], \pi_i = P(q_1 = s_i) \tag{2.25}$$

Two assumptions are made by the model. The first one called the Markov assumption. So states that the current state is dependent only on the previous state which represents the memory of the model:

$$P(q_t \mid q_1^{t-1}) = P(q_t \mid q_{t-1}) \tag{2.26}$$

The independence assumption states that the output observation at time t is depends only on the current state. So it doesn't independs on previous observations and states:

$$P(o_t \mid o_1^{t-1}, q_1^t) = P(o_t \mid q \mid t) \tag{2.27}$$

Following are the three problems related to HMM:

- The learning problem which is solved by the Baum-Welch algorithm:

Given some training observation sequences $O = o_1, o_2...o_K$ and general structure of HMM (numbers of hidden and visible states), determine HMM parameters $M = (A, B, \pi)$ that best fit training data [31].

- The evaluation problem which is solved by forward-backward algorithm: Given the HMM $M = (A, B, \pi)$ and the observation sequence $O = o_1, o_2...o_K$, calculate the probability that model M has generated sequence O [31].

- The decoding problem which solved by Viterbi algorithm: Given the HMM $M = (A, B, \pi)$ and the observation sequence $O = o_1, o_2...o_K$, calculate the most likely sequence of hidden states s_i that produced this observation sequence O [31].

Following are the three problems in details:

2.1.4.2 Learning

Learning is used to build the model and be able to estimate the model parameters $\lambda = (A, B, \pi)$ that best describe that process, we need a set of training data from a process to be given. Two standard approaches were made to this task. Those approaches are dependent on the form of the examples and will be referred to as supervised and unsupervised training. Supervised training can be performed if the training examples contain both the inputs and outputs of a process. Also supervised training can be perform by equating inputs to observations, and outputs to states, but if only the inputs are provided in the training data then unsupervised training must be used to guess a model that may have produced those observations [31].

To create a model λ is to have a large amount of training examples, each annotated with the correct classification. We define two sets:

- $t_1 \cdots t_N$ is the set of tags, which we equate to the HMM state set $s_1 \cdots s_N$
- $w_1 \cdots w_M$ is the set of words, which we equate to the HMM observation set $v_1 \cdots v_M$

So with this model decoding the most probable hidden state sequence any state given an observation sequence of words. In order to determine the model parameters λ, the Maximum Likelihood Estimates (MLE) from a sequence can be used. For the transition matrix we use:

$$a_{ij} = P(ti \mid t_j) = \frac{Count(t_i \mid t_j)}{Count(t_i)} \tag{2.28}$$

where $Count(t_i, t_j)$ is the number of times t_j followed t_i in the training data. For the observation matrix:

$$b_j(k) = P(w_k \mid t_j) = \frac{Count(w_k, t_j)}{Count(t_j)} \tag{2.29}$$

where $Count(w_k, t_j)$ is the number of times w_k was appears t_j in the training data. And lastly the initial probability distribution:

$$\pi_i = P(q_1 = t_i) = \frac{Count(q_1 = t_i)}{Count(q_1)} \tag{2.30}$$

In practice, when estimating a HMM from counts, it is necessary to apply smoothing in order to avoid zero counts and improve the performance of the model on data not appearing in the training set.

A similar coefficient can be computed for$\hat{\beta}_t = (i)$.

2.1.4.3 Evaluation

By using a given HMM, and a sequence of observations, to be used for computing $P(O \mid \lambda)$, the probability of the observation sequence given a model.

The probability of the observations O for a specific state sequence Q is:

$$p(O \mid Q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) = b_{q_1}(o_1) \times b_{q_2}(o_2) \cdots \times b_{q_T}(o_T) \tag{2.31}$$

and the probability of the state sequence is:

$$P(Q \mid \lambda) = \pi_{q1} a_{q_1 q_2} a_{q_2 q_3 q T-1 q T} \tag{2.32}$$

the probability of the observations given the model could be calculated as:

$$P(O \mid \lambda) = \sum_{Q} P(O \mid Q, \lambda) P(O \mid \lambda) = \sum_{q_1 \ldots . q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \ldots a_{q_T - 1 q_t} b_{q_t}(o_T) \tag{2.33}$$

This result allows the evaluation of the probability of O, but to evaluate it

directly would be exponential in T. Recognizing many redundant calculations would be made by directly evaluating Equation 2.33 could get a better approach, and therefore caching calculations can lead to reduced complexity.

At each time step the cache as a trellis of states is implemented, calculating the cached valued (called α) for each state as a sum over all states at the previous time step. α is the probability of the partial observation sequence $o_1, o_2 \cdots o_t$ and state s_i at time t. The forward probability variable is defined:

$$\alpha_t(i) = P(o_1 o_2 \cdots o_t, q_t = s_i \mid \lambda) \tag{2.34}$$

By working through the trellis filling in the values of α the sum of the final column of the trellis will equal the probability of the observation sequence. The algorithm for this process is called the forward algorithm and is as follows:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N \tag{2.35}$$

2. Induction:

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)\alpha_{ij}]b_j(o_{t+1}), 1 \leq t \leq T-1, 1 \leq j \leq N \tag{2.36}$$

3. Termination:

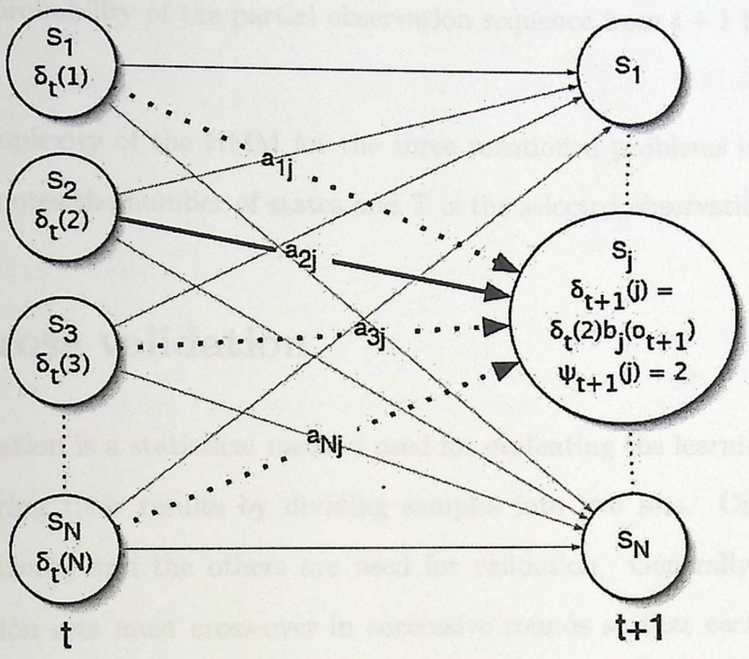


Figure 2.1: The induction step of the forward algorithm [4]

$$p(O \mid \lambda) = \sum_{i=1}^{N} \alpha T(i) \tag{2.37}$$

The key to the forward algorithm is the induction step and is depicted in Figure 2.1. For each state $s_j, \alpha_j(t)$ stores the probability of arriving in that state having observed the observation sequence up until time t.

The backwards algorithm is defined as exact reverse of the forword algorithm with the backwards variable:

$$\beta_t(i) = P(o_t + {}_1o_t + {}_2 ... o_t \mid q_t = s_i, \lambda \tag{2.38}$$

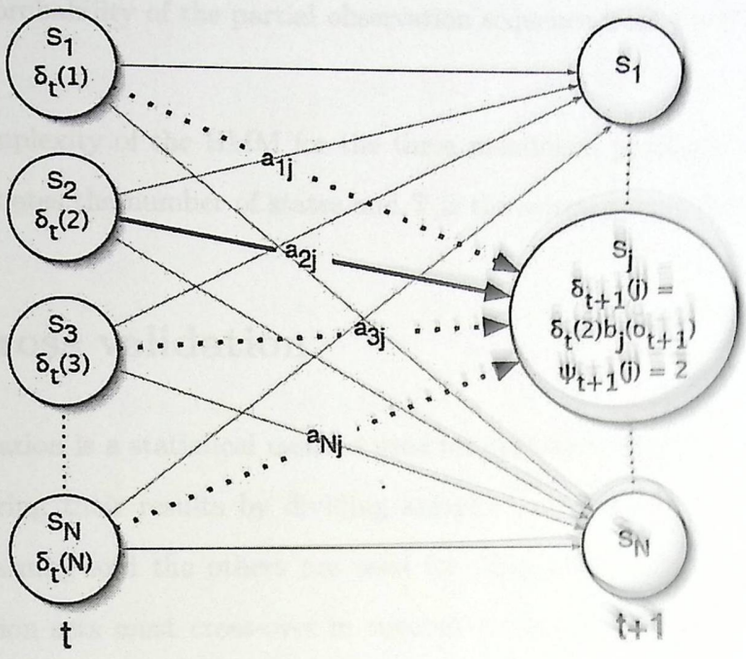


Figure 2.1: The induction step of the forward algorithm

$$p(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The key to the forward algorithm is the induction step and is depicted in Figure 2.1. For each state s_j, $\alpha_j(t)$ stores the probability of arriving in that state having observed the observation sequence up until time t.

The backwards algorithm is defined as exact reverse of the forward algorithm with the backwards variable:

As the probability of the partial observation sequence from $t+1$ to T, starting in state s_i.

The complexity of the HMM for the three mentioned problems is $O(TN^2)$ [6] where N denotes the number of states and T is the selected observation time.

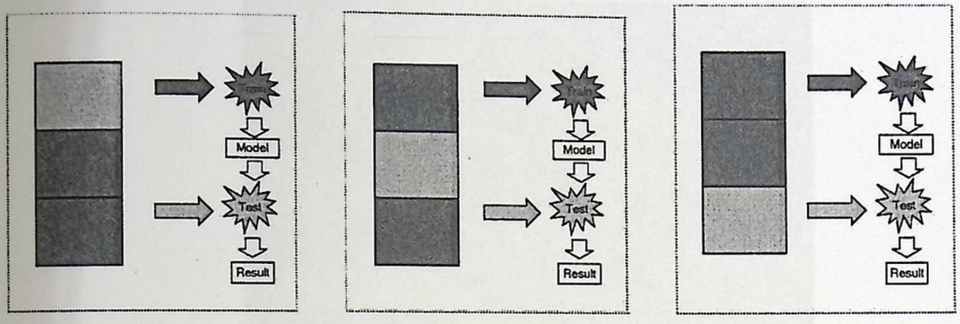
2.2 Cross validation

Cross-Validation is a statistical method used for evaluating the learning algorithms and comparing their results by dividing samples into two sets. One of them is used for learning and the others are used for validation. Generally the learning and validation sets must cross-over in successive rounds so that each point has a chance to be validated [32]. The basic algorithm of cross-validation is K-fold cross validation.

In K-fold cross-validation the data partitioned into K sets, the size of these sets is nearly equal. Then subsequently K iterations of training and validation are performed so that within each iteration a different fold of the data is used for validation while the remaining $K-1$ folds are used for training [32].

Figure 2.2a; demonstrates a first iteration of an example with $K = 3$. The darker sections of the data are used for training while the lighter sections are used for validation, Figure 2.2b and Figure 2.2c show the next iterations. In data mining and machine learning 10-fold cross-validation ($K = 10$) is the most common.

the process of the cross validation runs as follows:



(a) First fold selection. (b) Second fold selection. (c) Last fold selection.

Figure 2.2: KFold algorithm where $K = 3$ [32]

In each iteration, one or more learning algorithms use $K - 1$ folds of data for training and the learned models are asked to make predictions about the data in the validation fold [32].

Then a performance for the used learning algorithm is used on each fold calculated using a performance function like accuracy. Finally when the iterations are completed, K samples with their accuacy will be available for each algorithm.

Also Different ways can be used to calculate an aggregate measure from these samples such as averaging [32].

2.3 KINECT and Infra-red technology

2.3.1 KINECT sensor overview

KINECT is a "controller-free gaming and entertainment experience" produced by Microsoft for Xbox360 video game platform which allow user to interact and control

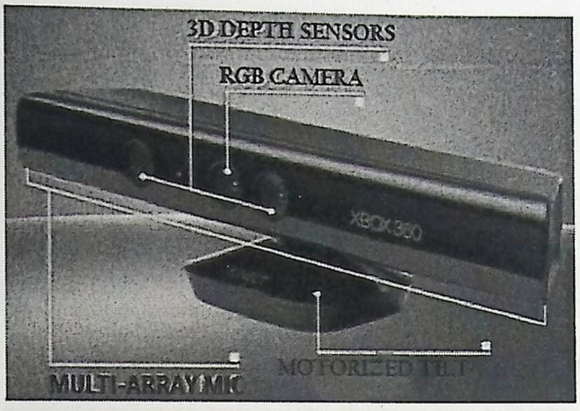


Figure 2.3: KINECT sensor

the Xbox 360 without need to touch the controller [20].The hardware of the KINECT contains:

- Color and depth (RGB-D) sensing camera.
- Accelerometer.
- Microphones.
- Control the rotating and tilting motor.

The RGB camera sends images of 640×480 pixels at 30 images per second, depth frames also of 640×480 pixels 30 times a second. The depth sensor range between 0.7 meters to 4.8 meters. A motor tiles the sensor with a range of 27 degrees. Figure 2.3 shows the KINECT sensor and its parts.

The depth image constructed from the infrared laser projector combines monochrome CMOS sensor under any light conditions.

Figure 2.4: Camera with infrared sensor sends the infrared beams

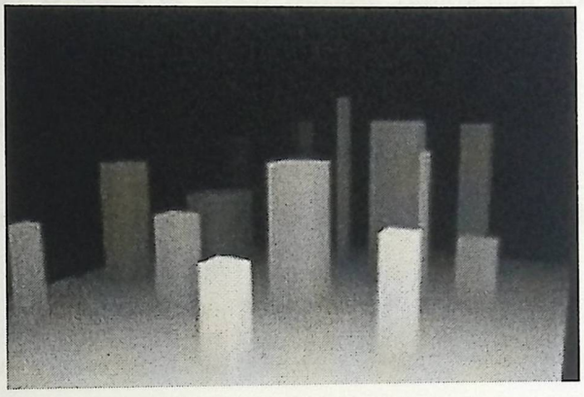


Figure 2.5: 3D image

More than one driver for KINECT presented, gives the ability to interact and communicate with KINECT from ordinary PC's by acting as a translator between the operating system and the hardware device.

The first driver published for KINECT is OpenKinect or libfreenect, first developed by Hector Martin. This driver works for Windows, Linux and OpenKinect drivers include all code necessary to:

- Activate

- Initialize
- Communicate data with the KINECT hardware

OpenKinect is cross-platform and run on Windows, Linux, and OS X. The drivers expose an API for C/C++ and managed code (C#),although drivers published by NUI Group (AlexP) known as CL NUI Platform which is Windows KINECT driver and OpenNI or Open Natural Interaction driver with API's to communicate with KINECT [9].

Microsoft released an SDK for KINECT sensor includes:

- Drivers, for using KINECT sensor devices on a computer that is running Windows 7.
- APIs and device interfaces.
- Source code samples.

This library contains a skeleton tracking methods called NUI Skeleton API. It provides information about the location of up to two players standing in front of the KINECT sensor array, with position and orientation information.

The data provides a set of points called skeleton positions that compose a skeleton, as shown in Figure 4.2. This skeleton represents a user's current position and pose.

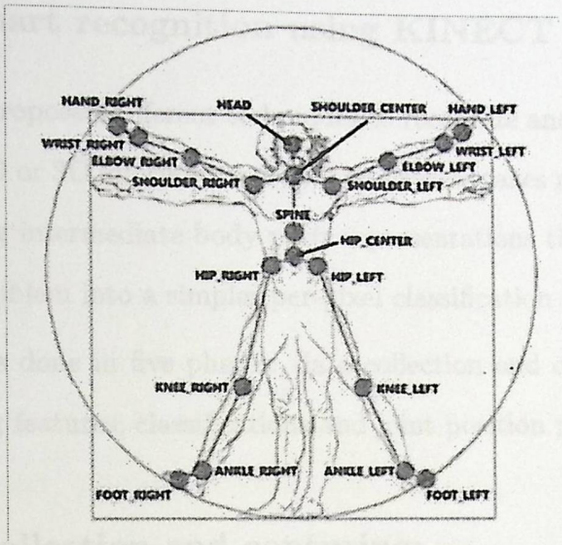


Figure 2.6: Skeleton joints relative to the human body. [20]

2.3.2 KINECT limitation

The KINECT sensor has some hardware and software limitations. Based on some experiments we made, we have measured the maximum distance captured as 3.8 meters; the minimum distance it captures is 0.7 meters. Also there was an error fraction with the accuracy of the distance it provides. We have measure the actual distance for the object placed in front of the KINECT and compare the results with the output of the sensor and notice that the error starts with almost 0 on 0.7 meter distance and increases when the object getting far, the error we calculate was up to 4 cm when the object reaches 3.8 meters far from the KINECT (0.01 for each cm).

2.3.3 Body part recognition using KINECT

Many researchers proposed different techniques to recognize and build human skeleton, either using 2D or 3D information [36]. The process takes an object recognition approach, designing intermediate body parts representations that map the difficult pose estimation problem into a simpler per-pixel classification problem.

Their work was done in five phases: data collection and capturing, body part labeling, extracting features, classifications and joint position proposals.

2.3.4 Data collection and capturing:

As there are various range of poses which are difficult to simulate, a large database of motion captured for human actions. The database consist of approximately 500k frames in a few hundred sequences of driving, dancing, kicking, running, navigating menus, etc [36].

Often, changes in pose from one motion capture frame to the next are so small. So similar redundant poses are discard from the initial motion capture data using furthest neighbor clustering where the distance between poses p_1 and p_2 is defined as $\max_j \left\| p_1^j - p_2^j \right\|_2$ the maximum Euclidean distance (see Equation 2.3) over body joints j. The result is a subset of 100k poses such that no two poses are closer than 5cm [36].

2.3.4.1 Body part labeling:

In this phase we need to define several localized body part labels that cover the body. Some labels are defined to directly localize particular skeletal joints of interest, while others fill the gaps or could be used in combination to predict other joints [36].

The pairs of depth information and body part images are used as fully labeled data to learn the classifier. For the experiments they use 31 body parts: {LU/RU/LW/RW head, neck, L/R shoulder, LU/RU/LW/RW arm, L/R elbow, L/R wrist, L/R hand, LU/RU/LW/RW torso, LU/RU/LW/RW leg, L/R knee, L/R ankle, L/R foot}, where L: Left, R: Right, U: Upper, W: lower [36]. The left and right labels allow the classifier to disambiguate the left and right sides of the body [9].

2.3.4.2 Extracting features:

The feature extraction phase was by employing simple depth comparison features:

$$f_\theta = (I, x) = d_I(x + \frac{u}{d_I(x)} - d_I(x + \frac{v}{d_I(x)}) \tag{2.39}$$

where $d_I(x)$ is the depth value at pixel x in image I, and parameters $\theta = (u, v))$ describe offsets u and v. The normalization of the offsets by $\frac{1}{d_I(x)}$ ensures the features are depth invariant. So at a given point on the body, a fixed world space offset will result whether the pixel is close or far from the camera.

The features become 3D translation invariant. If an offset pixel lies outside the

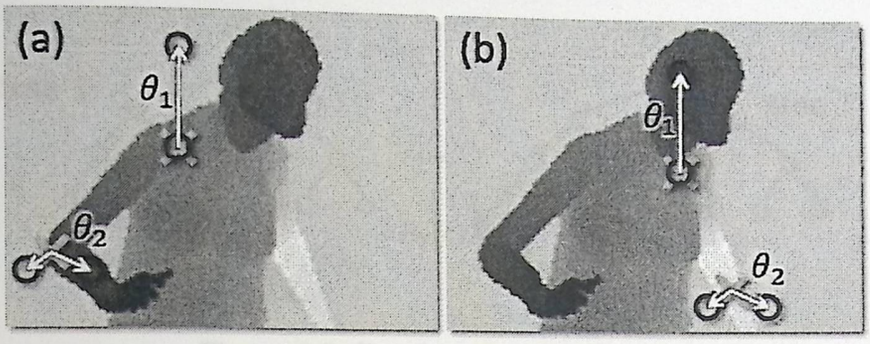


Figure 2.7: 3D Image features
3D Image features [36].(a), the two features produce large depth difference response. In (b), the same features at new locations produce much smaller response.

bounds or in the background of the image, the depth $d_I(\acute{x})$ has a large positive constant value.

As we can see in Figure 2.7(a), feature θ_1 looks upwards; so it will give a large positive response for pixels x near the top of the body, on the other hand the value will be close to zero for pixels x lower down the body which may also instead help find thin vertical structures such as the arm. See 2.7(b).

These features may only give a weak signal about which part of the body the pixel belongs to, further it is combined to a decision forest which will take care to accurately disambiguate all trained parts.

2.3.4.3 Joint position proposals:

In this phase, 3 randomized decision trees was used. As shown in Figure 2.8, a forest is an ensemble of T decision trees, each consisting of split and leaf nodes.

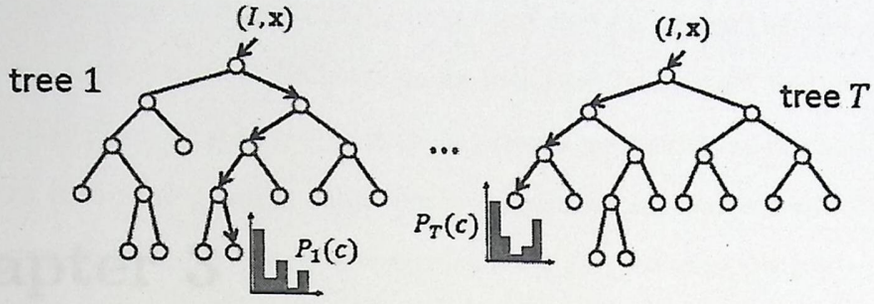


Figure 2.8: Random decision forests.
Random decision forests [36].

Each split node consists of a feature f_θ and a threshold.

In order to determine class of pixel x in image I as we can notice in Figure 2.8, one starts at the root and repeatedly evaluates, branching left or right according to the comparison to threshold τ. At the leaf node reached in tree t, we store a learned distribution $P_t(c|I,x)$ over all labels c. The distributions are averaged together for all trees in the forest in order to give final classification.

$$P(c|I,x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I,x) \tag{2.40}$$

Chapter 3

Literature review

In this chapter, we will review some literature for gesture recognition and present the previous research work. Due to the variant hardware devices used for gesture recognition and its impact to the recognition features and strategy as we use the KINECT sensor, we are presenting the previous studies and papers categorized by which hardware device sensor was used in the research like gloves, stereo, 3D and KINECT sensor.

Some researchers used special hardware devices like colored gloves or stereo cameras and others used depth cameras. Many researchers focused on hand recognition, while some studied the human body gestures. The methodology researcher's use varies. Some used Hidden Markov Model (HMM) to archive their goal, while others tried to use a simpler algorithm like nearest neighbor classification and the similarity measure.

As possible applications for human interaction with computers has evolved, in-

terest in advancing and perfecting gesture and pattern recognition has increased. There are a large number of reports to be found on the subject, and hand-gesture research comprises the majority of gesture pattern recognition research. This is due to the fact that hand gestures share nearly a universal meaning across cultures, and they are critically important as a communication tool for most people.

With data processing stage some studies use the HMM like [43, 18, 46] who used both HMM and neural networks. [7] uses fuzzy and neural network algorithms and uses nearest neighbor and furthers neighbor classifiers to predict 3D positions of body joints from single depth image and no temporal data.

Kendons gesture continuum suggests that as the quantity of information transmitted by the human voice decreases, the amount of information conveyed by gesturing directly increases [16]. In some cases, they are the only communication method available (e.g. - some disabled persons). The approach to perfecting gesture recognition varies. Data collection methods include 3D camera images, datâ gloves, and stereo cameras.

Some researchers used 3D camera image to gain better recognition as [25]. Others try to use different data collection technique like [7] who use Pair of data gloves and [46, 18, 43] use stereo cameras for data acquisition.

3.1 Gloves:

The use of pairs of data gloves in these studies is common. Lamberti and Camastra [23] used a wool 3-color glove and Learning Vector Quantization to result in a

real-time recognition rate of nearly 98 percent on 907 different hand gestures. In this case, one color was used to dye the palm, and each set of adjacent fingers were colored differently using the other two colors. This particular glove was especially inexpensive. The system was tested by being asked to classify gestures into one of 13 classes. The total number of gestures included in these 13 classes was 1541. Gestures were performed by people of varying physique and from both genders.

Wang and Popovic made use of a multi-colored cloth glove bearing a specific pattern [40], the idea being that a distinct color set simplifies pose inference to the point that the actual pose of a hand can be determined with a single frame. This particular glove had twenty patches made up of 10 individual colors, which were determined by the capability of the camera to distinguish these specific 10 colors and no more. Using a nearest-neighbor algorithm, the result of this particular study was the introduction of a hand-tracking user-input device for computers consisting of a single cloth glove and one camera. They suggested add-ons to their system in the form of design props, refined calibration of their camera, multi-touch input devise, and additional cameras for more accuracy or bimanual input (as long as the hands do not occlude or interlock).

Keskin, Aran, and Akuran utilized HMM and colored gloves in an attempt to develop a gestural interface [17, 19]. They used basic web cameras and applied 3D reconstruction from stereo images, then trained HMMs to recognize specific gestures. The product of this study was a system capable of tracking two hands and 3D motion in real-time. The results of their application testing were impressive in that they successfully identified 98.75 percent of gestures in 160 trials. Such a success rate

suggests real-world use of gestures for inputs into computer applications.

Bedregal, Costa, and Dimuro used a data glove to collect information and introduced fuzzy rules-based logic for the recognition of hand gestures from Brazilian sign language (LIBRAS) [3]. Hand classifications were based on angles of finger joints, and classifications of segments of hand gestures for identifying gestures. Gesture segmentation is based on monotonic gesture segment and a given gesture is comprised of a set of these segments. This method is accurate as long as a detailed analysis of gesture features is done and that information is maintained in this system.

3.2 Stereo camera:

Guo [14] and Wilson [44] have utilized stereo cameras in studies that suggest feasibility for future human-computer interface technology that will not require precise articulation tracking. In the case of Guo, a Microsoft Kinect device was used, and Wilsons work utilized the parametric Hidden Markov Model.

Elmezain, proposed an automatic system that recognizes both isolated and continuous gestures for Arabic numbers (0 - 9) from stereo color image sequences by the motion trajectory of a single hand using HMM [13]. There are three basic features: location, orientation and velocity used to recognize the continuous gestures; and they relied upon the orientation feature in their system. Their database contained 30 video sequences for each isolated gesture number from 0 to 9 and 70 video sequences for continuous gestures. The LRB topology with 5 states presented the best performance. Results show that; an average recognition rate was 98.94 percent

and 95.7 percent for isolated and continuous gestures, respectively. The practical applications of a robot understanding human direction are of course, endless.

Regarding the data processing phase, many studies utilize the Hidden Markov Model. For example, Wilson and Bobick [41] pioneered work in this area by presenting a parametric approach able to learn an HMM from a set of demonstrations, where recognition is achieved by the EM algorithm are parameterizations of movement are taken as latent variables.

Wilson and Bobick then built on that work by extending parametric HMM to consider non-linear smooth output distributions [42]. The result was the ability to model a far larger class of gestures and movements than had previously been achieved by the linear parametric HMM, and suggest that parameterization can be customized to a specific gestures and movements performed by the subject.

Some researchers approach the challenge from a different angle. Rather than directly following the hand. Researchers Ye, Corso, and Hager took an object-centered approach to track a single hand and computed the 3D appearance using a region-based coarse stereo matching algorithm in a volume around the hand. Noting the differences in appearance feature gave the motion cue, and an unsupervised learning scheme captured the cluster structure. Utilizing both HMM and neural networks to model gesture dynamics, the researchers modeled gestures and were able to achieve over 96 percent real-time gesture recognition accuracy in testing that involved a large set of combinations of motion [45]. They basically implemented a K-means algorithm to learn the centroid of each of the clusters of the feature sets. The

choice of the number of clusters is empirical. Then they represent each vector using a symbol that indicates its cluster identity. In their data sets comprised of 6 gestures, they used less than 20 clusters to describe each of the feature set, including the appearance set, the motion set and the appearance-motion combination set. Given a pair of rectified stereo images of the scene, a disparity map can be computed using a standard correspondence search algorithm.

3.3 3D camera:

3D camera has some benefits over other traditional intensity sensors [7]. Since it works in low light levels, giving a calibrated scale estimate, being color and texture invariant, and resolving silhouette ambiguities in pose. It also greatly simplifies the task of background subtraction.

In regard to the approach, it is straightforward to build realistic depth images of people, therefore to build a large, inexpensive training dataset.

Malassiotis, Aifanti, and Strintzis utilized a 3D sensor to generate a more dense range image of the scene, and this lead to the recognition of a wider range of complex hand gestures [25]. Their results were largely independent of illumination conditions and scene content, and it depended upon several 3D image analysis algorithms that were new to this study. This system was tested in regard to sign-language movements and postures, with 20 hand postures from Greek sign language being chosen to attempt to recognize. Initial tests showed a success rate of 97 percent accurate recognition was reached. Also, when 4 different postures with 50 images for each

choice of the number of clusters is empirical. Then they represent each vector using a symbol that indicates its cluster identity. In their data sets comprised of 6 gestures, they used less than 20 clusters to describe each of the feature set, including the appearance set, the motion set and the appearance-motion combination set. Given a pair of rectified stereo images of the scene, a disparity map can be computed using a standard correspondence search algorithm.

3.3 3D camera:

3D camera has some benefits over other traditional intensity sensors [7]. Since it works in low light levels, giving a calibrated scale estimate, being color and texture invariant, and resolving silhouette ambiguities in pose. It also greatly simplifies the task of background subtraction.

In regard to the approach, it is straightforward to build realistic depth images of people, therefore to build a large, inexpensive training dataset.

Malassiotis, Aifanti, and Strintzis utilized a 3D sensor to generate a more dense range image of the scene, and this lead to the recognition of a wider range of complex hand gestures [25]. Their results were largely independent of illumination conditions and scene content, and it depended upon several 3D image analysis algorithms that were new to this study. This system was tested in regard to sign-language movements and postures, with 20 hand postures from Greek sign language being chosen to attempt to recognize. Initial tests showed a success rate of 97 percent accurate recognition was reached. Also, when 4 different postures with 50 images for each

posture were tested, the resulting recognition rate was 95 percent. The researchers suggested more extensive testing with larger image numbers to determine system limits, and they're currently investigating real-time application of this approach.

Lahamy and Litchi utilized a range camera capable of capturing a full 3D point cloud with an array sensor at video rates [22]. The camera was a Swiss Ranger SR4000, a time-of-flight camera which produces images at a rate up to 54 frames-per-second. The goal of this study was to create a human-machine interface which would recognize hand gestures. There were two applications tested for this system; one was identifying the number of raised fingers, and the second was for manipulating a moving object in a virtual environment in accordance with a hand gesture. The motivation for this second application is for oil and gas reservoir modeling. While there were some of the not-unexpected issues with placement of lighting and the issues with an over-abundance of direct light, they were able to demonstrate reasonable optimism for the use of range cameras for real-time applications. They suggest some of the issues found would be resolved by adding cameras.

Breuer, Exckes and Mller used an infrared time-of-flight range camera (Swiss Ranger SR2 miniature) with up to a 30 Hz frame rate to capture 3D surface points from the subjects hand. This particular camera delivers real-time data and requires no specific background or skin color [5]. They were able to accurately fit a hand model into the point cloud and to follow its movement. The result was the ability to estimate the first 7 degrees of freedom (DoF) of the hand within 200 ms of a human hand with 2-3 Hz frame rate. Principle Component Analysis (PCA) is used to obtain a first crude estimate on the location and orientation of the hand. An articulated

posture were tested, the resulting recognition rate was 95 percent. The researchers suggested more extensive testing with larger image numbers to determine system limits, and they're currently investigating real-time application of this approach.

Lahamy and Litchi utilized a range camera capable of capturing a full 3D point cloud with an array sensor at video rates [22]. The camera was a Swiss Ranger SR4000, a time-of-flight camera which produces images at a rate up to 54 frames-per-second. The goal of this study was to create a human-machine interface which would recognize hand gestures. There were two applications tested for this system; one was identifying the number of raised fingers, and the second was for manipulating a moving object in a virtual environment in accordance with a hand gesture. The motivation for this second application is for oil and gas reservoir modeling. While there were some of the not-unexpected issues with placement of lighting and the issues with an over-abundance of direct light, they were able to demonstrate reasonable optimism for the use of range cameras for real-time applications. They suggest some of the issues found would be resolved by adding cameras.

Breuer, Exckes and Mller used an infrared time-of-flight range camera (Swiss Ranger SR2 miniature) with up to a 30 Hz frame rate to capture 3D surface points from the subjects hand. This particular camera delivers real-time data and requires no specific background or skin color [5]. They were able to accurately fit a hand model into the point cloud and to follow its movement. The result was the ability to estimate the first 7 degrees of freedom (DoF) of the hand within 200 ms of a human hand with 2-3 Hz frame rate. Principle Component Analysis (PCA) is used to obtain a first crude estimate on the location and orientation of the hand. An articulated

hand model is fitted to the data to refine the first estimate. They determined that intensive background light (e.g. bright sunlight) led to higher photon shot noise, and thus degenerated precision of measurements. Furthermore the study determined this approach was promising, especially as 3D camera technology advances. These advances will lead to advantages such as smaller distances between the camera and the subject without photo integrator over-saturation. This would mean a denser sample and more detail of the visibility of the hand, which should in turn lead to an increase in (DoF) recognition.

3.4 KINECT:

More research is now being done using Microsoft's Kinect motion sensing device, which was the first widespread release of a 3D camera. Originally built for Microsofts Xbox 360 video game console, Kinect for Windows became available in 2011, and research into its uses for gesture recognition has taken off.

While there had been some success using Kinect for face recognition and body tracking, Tang attempted to collect data using a Kinect sensor for the purposes of recognizing movement on a smaller scale (e.g. specific hand gestures) [38]. The system first used a full body tracker to locate the subjects hand, then to recognize movements of the hand over a time period. They integrating RGB and depth data collected using the Kinect and they were able to demonstrate hand gesture recognition. They assumed that the subject was standing approximately 3 meters away from the Kinect sensor, providing imperfect but acceptable scale invariance.

hand model is fitted to the data to refine the first estimate. They determined that intensive background light (e.g. bright sunlight) led to higher photon shot noise, and thus degenerated precision of measurements. Furthermore the study determined this approach was promising, especially as 3D camera technology advances. These advances will lead to advantages such as smaller distances between the camera and the subject without photo integrator over-saturation. This would mean a denser sample and more detail of the visibility of the hand, which should in turn lead to an increase in (DoF) recognition.

3.4 KINECT:

More research is now being done using Microsoft's Kinect motion sensing device, which was the first widespread release of a 3D camera. Originally built for Microsofts Xbox 360 video game console, Kinect for Windows became available in 2011, and research into its uses for gesture recognition has taken off.

While there had been some success using Kinect for face recognition and body tracking, Tang attempted to collect data using a Kinect sensor for the purposes of recognizing movement on a smaller scale (e.g. specific hand gestures) [38]. The system first used a full body tracker to locate the subjects hand, then to recognize movements of the hand over a time period. They integrating RGB and depth data collected using the Kinect and they were able to demonstrate hand gesture recognition. They assumed that the subject was standing approximately 3 meters away from the Kinect sensor, providing imperfect but acceptable scale invariance.

They used histograms for an open and closed hand and the open hand has a histogram with more spikes and overall variation. In this particular case, grasp and drop gestures were accurately identified over 90 percent of the time. This, of course, translates to the ability to drag and drop items in a virtual setting, which is ability with numerous possible applications.

Ren, also used a Kinect sensor to collect data, and then created FEMD (Finger-Earth Movers Distance), a distance metric utilized to track fingers rather than the whole hand [35]. This approach allowed them to recognize more slight movements. Extensive testing proved the FEMD-based approach to be quite accurate. These same researchers also demonstrated the accuracy of their system by teaching it to play a traditional Rock-Paper-Scissors game. The computer would choose its weapon, recognize the human player's weapon. And determine the winner. On a more useful note, they also demonstrated the systems ability to input gestures instructing the system to perform mathematical computations [34].

An important challenge that should be resolved through gesture recognition technology is that of "wet lab" personnel using a computer keyboard in an environment in which they're required to wear gloves. A system of accurate recognition of finger movements could allow the lab technician the ability to work on a computer without ever actually touching the keyboard. Du and Hang, addressed this situation using a Kinect [11]. This study was promising in that it achieved a correct classification rate of 94 percent, but there is much work to do on this particular application. Du and To's system was only developed to recognize a single hand and specific positioning of the fingers (spread apart, pointing up) was required. Still, it's

They used histograms for an open and closed hand and the open hand has a histogram with more spikes and overall variation. In this particular case, grasp and drop gestures were accurately identified over 90 percent of the time. This, of course, translates to the ability to drag and drop items in a virtual setting, which is ability with numerous possible applications.

Ren, also used a Kinect sensor to collect data, and then created FEMD (Finger-Earth Movers Distance), a distance metric utilized to track fingers rather than the whole hand [35]. This approach allowed them to recognize more slight movements. Extensive testing proved the FEMD-based approach to be quite accurate. These same researchers also demonstrated the accuracy of their system by teaching it to play a traditional Rock-Paper-Scissors game. The computer would choose its weapon, recognize the human player's weapon. And determine the winner. On a more useful note, they also demonstrated the systems ability to input gestures instructing the system to perform mathematical computations [34].

An important challenge that should be resolved through gesture recognition technology is that of "wet lab" personnel using a computer keyboard in an environment in which they're required to wear gloves. A system of accurate recognition of finger movements could allow the lab technician the ability to work on a computer without ever actually touching the keyboard. Du and Hang, addressed this situation using a Kinect [11]. This study was promising in that it achieved a correct classification rate of 94 percent, but there is much work to do on this particular application. Du and To's system was only developed to recognize a single hand and specific positioning of the fingers (spread apart, pointing up) was required. Still, it's

a worthwhile application and further research is warranted.

Van and Bergh, also used a Kinect in their work to create a real-time gesture interaction system with a robot [39]. Using the depth sensor, they were able to recognize 3D gestures (e.g. pointing), and to communicate and give directions to the robot. Specifically, the robot, equipped with a navigation system of its immediate environment, would seek out and find the person to interact with, detect a pointing gesture telling it where to go, and then explore the new vicinity for the next person to interact with. The researchers found their work promising enough to suggest further work involving combining a Kinect sensor with a camera stereo to address some of the issues that arise with unpredictable lighting or an overexposure of lighting,(e.g. outdoors).

Doliotas attempted gesture recognition in particularly demanding circumstances such as unpredictable light and moving objects in background [10]. They achieved 85 percent accuracy employing a Kinect sensor and using the colored image to detect hand position. For depth image, they used a hand detector based on motion from frame differencing, which combined with depth segmentation according to the depth information for each pixel. They defined 10 classes representing 10 digits from 0 to 9, each represented by a gesture. For each class, they defined several training video sequence samples using a colored glove. They then used 10 different users, each performs 3 times each gesture digit.

Recognizing the input gesture was done using the nearest neighbor classification framework, and the similarity measure that used for the 1NN scheme was the score

returned by the Dynamic Time Warping algorithm. The DTW algorithm temporally aligns two sequences, a query sequence and a model sequence. Then computes a matching score, which is used for classifying the query sequence.

We are presenting our study and the goal of this work as to examine and test how far we can reach to implement a gesture recognition technology using low cost and low accuracy sensor called KINECT produced by Microsoft. And using various machine learning algorithms.

Chapter 4

Gesture Recognition using KINECT

This chapter presents our work and how data was acquired, collected and processed to reach the recognition of gestures. Working with gesture recognition consists of many phases. Here in this work, we need six phases to build the proposed system.

The phases we used in our work starts with data collection using KINECT as a 3D imaging device, then we process and normalize this data and transform the processed information into 3d skeleton. After that we learn and label the gesture in the skeleton using machine learning approach taking in consideration the context of this gesture. Finally we output the gesture meaning to the user.

Figure 4.1 shows the general block diagram for the proposed system we are presenting here.

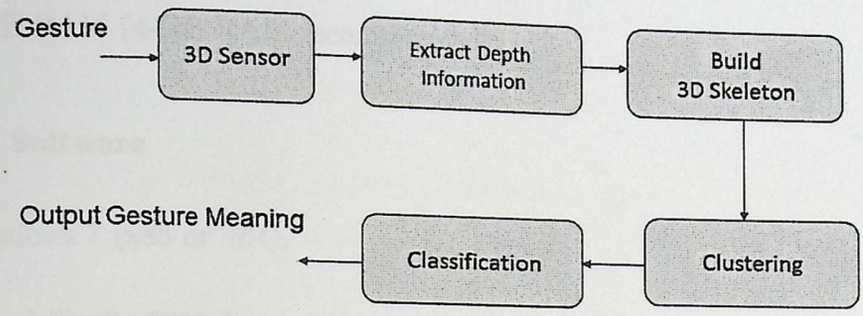


Figure 4.1: General block diagram

4.1 Data acquisition using KINECT as a 3D imaging device.

After connecting and preparing the KINECT sensor as discussed in Appendix A, we have used Microsoft driver and SDK, which was published by Microsoft Corp, on July-29th- 2011. It is a beta version of the SDK for KINECT sensor and it is easy to install. It also includes skeleton tracker with the ability to track up to two users at a time.

This driver has some major software and hardware requirements in order to run on a personal computer. Those requirements are as follows:

4.1.0.4 Hardware

- Kinect for Xbox 360 sensor.
- Computer with a dual-core, 2.66-GHz or faster processor.
- Windows 7-compatible graphics card that supports DirectX 9.0c capabilities.

- 2-GB RAM (4-GB RAM recommended).

4.1.0.5 Software

- Windows 7 (x86 or x64).
- Visual Studio 2010 Express (or other 2010 edition).
- Microsoft .NET Framework 4.0.

This driver provides a real time tracking process with 27 to 31 frames per second, and recognizes 20 joints to build the Skeleton of the human body. The methodology of user detection and body part recognition is described in details in Chapter 2. Figure 4.2 shows the human body joints detected which are Hip Center, Spine, Shoulder Center, Head, Shoulder Left, Elbow Left, Wrist Left, Hand Left, Shoulder Right, Elbow Right, Wrist Right, Hand Right, Hip Left, Knee Left, Ankle Left, Foot Left, Hip Right, Knee Right, Ankle Right, Foot Right.

The data acquired from the KINECT sensor contains three streams; 2D image (Figure 4.4), depth Image (Figure 4.3) and sound stream, in addition to the skeleton joints (Figure 4.5), which we need for the feature extraction phase. The 20 skeleton joints are presented as the 3D special point (X, Y, Z), where X = horizontal axis, Y= vertical axis, Z = distance between joint and the sensor (depth).

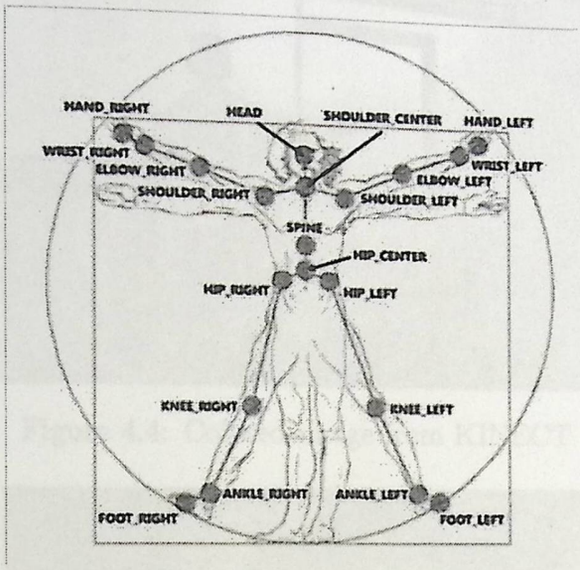


Figure 4.2: Skeleton joints relative to the human body. [20]

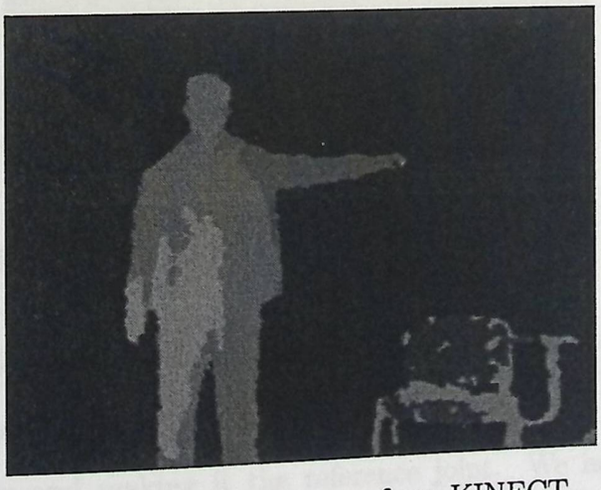


Figure 4.3: Depth image from KINECT

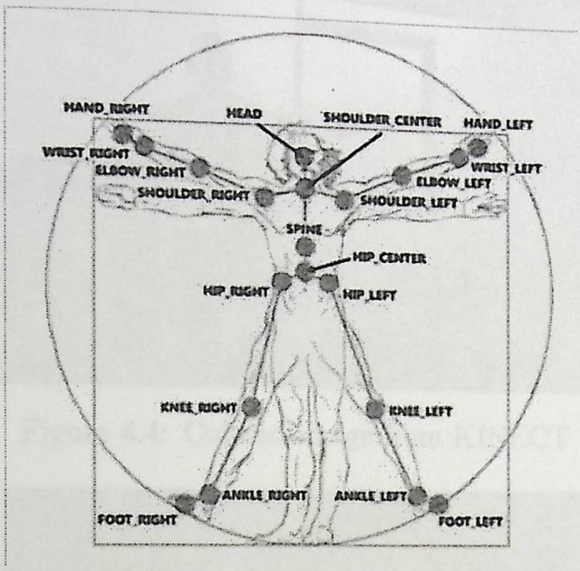


Figure 4.2: Skeleton joints relative to the human body. [20]

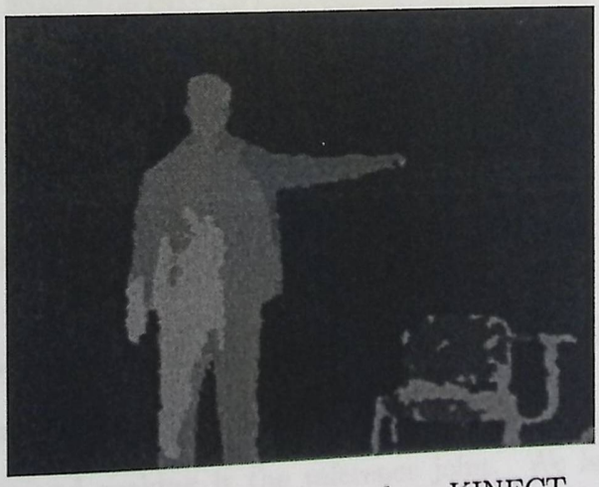


Figure 4.3: Depth image from KINECT

Figure 4.4: Colored image from KINECT

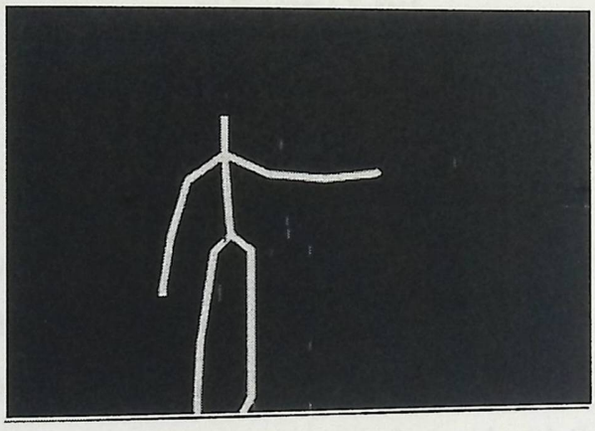


Figure 4.5: Detected skeleton

4.2 Scale and transition invariant:

We have made some data enhancement and normalization for each joint. This data was collected to make joints invariant for user position from the KINECT, starting from the Hip joints and making it the reference joint. We have also made two adjustments, scaling and translation.

All joints X and Y values centered in the original with reference to the reference joint position by subtracting its position from the Hip reference joint position, and after that, scaling is done by multiplying each joint X and Y positions with the Z joint position. So, no matter how far the user is from the KINECT sensor, it will be invariant with distances between joints as we notice in Equation4.1 and Equation 4.2 which are used for this task and valid for all depth values in range of 0.7 meters to 4.8 meters.

$$x'_i = (x_i z_i) + y_{ref} \tag{4.1}$$

$$y'_i = (y_i z_i) + y_{ref} \tag{4.2}$$

Where x' and y' are the new normalized x and y positions after scale and transition, and y_{ref} is the y value of the Hip joint. By obtaining this information, we get a vector V of length 60 values, formed as $V'_1 = X'_1, Y'_1, Z'_1, X'_2, Y'_2, Z'_2, X'_3, \cdots etc$, this vector contains all normalized joints positions referenced with the Hip joint position.

After that, we start recording data for training and testing. The recording process is done by capturing five frames for every gesture with interval of eight frames per second; so we will get five frames; 1.29 seconds (frame rate is 31 frames per second), in each frame we will have V vector describing the joints positions. Based to previous experiments we find that the difference in motion between frames with the high frame rate will not be large, because it is based on the human body

movement and joint positions and five frames with eight frames gap will form a discriminant gesture.

In our test we have capture eleven different gestures (see Figure 4.6), each was captured ten times. For the eleven captured gestures we will have 2200 feature vector to be used for training and testing. Every gesture has been recorded ten times, each time five frames are captured, and each frame forms a vector of 60 values.

As a result we will have five sequenced vectors $S = (V_1, V_2, V_3, V_4, V_5)$ denoted by S, which describes the gesture, or movement done by the user.

Listed below some of the gestures:

We stored the 2200 vector into a comma delimited file, and by using Matlab, we have used different machine learning techniques to build the gesture recognition phase. These techniques are : Nearest Neighbor (NN), Self Organizing Map (SOM) and Hidden Markov Model (HMM) with K-Mean clustering.

4.3 Gesture recognition using Nearest Neighbor (NN):

To apply the Nearest Neighbor algorithm for predicting the class and recognizing gestures we have prepared the data to be entered to the algorithm and tested the predicted gestures labels. Figure 4.7 shows the proposed gesture recognition using the Nearest Neighbor algorithm. For validation we have used the ten fold cross validation.

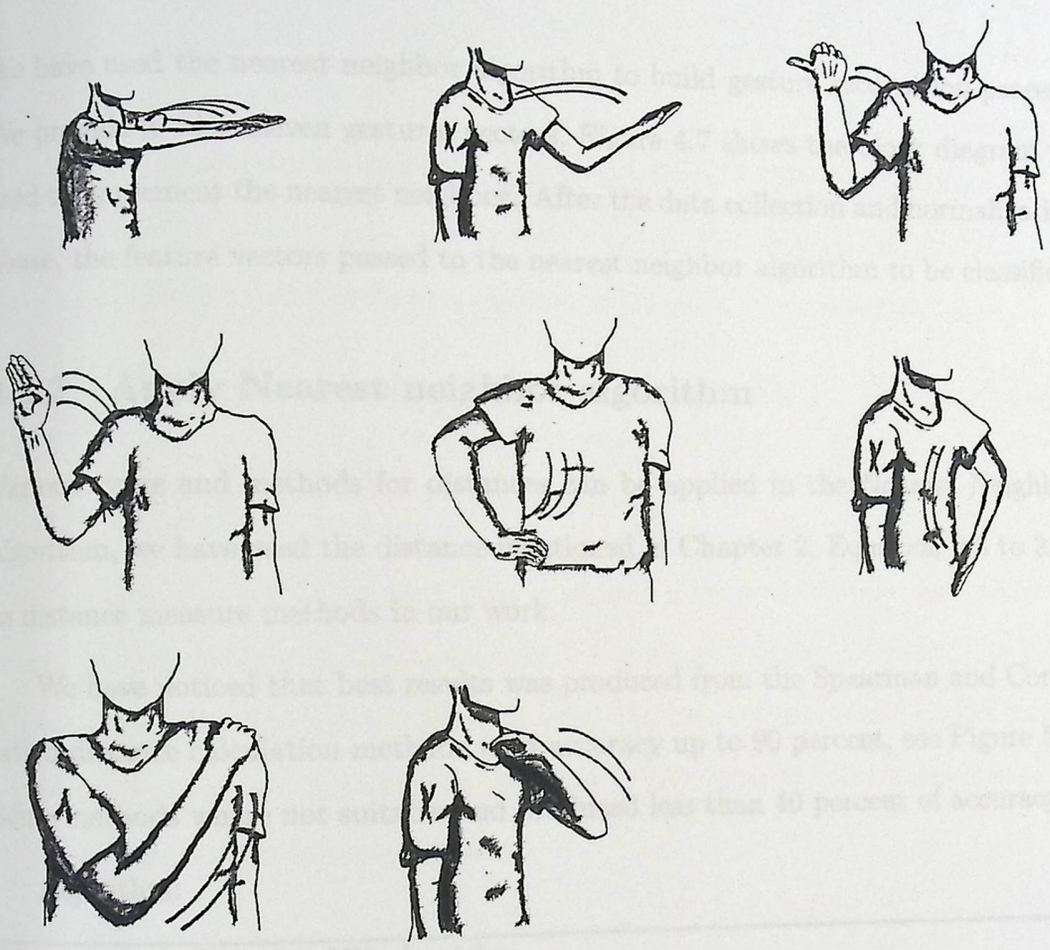


Figure 4.6: Gesture samples

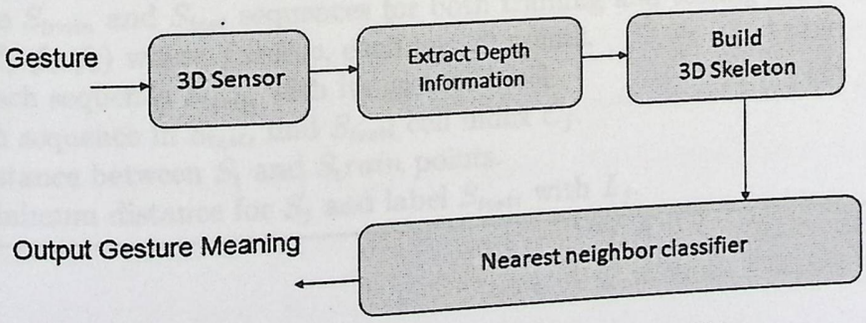


Figure 4.7: Gesture recognition using NN block Diagram

4.3.1 Data preparation

We have used the nearest neighbor algorithm to build gesture recognition process. We provide it with eleven gestures vectors, Figure 4.7 shows the block diagram we used to implement the nearest neighbor. After the data collection and normalization phase, the feature vectors passed to the nearest neighbor algorithm to be classified.

4.3.2 Apply Nearest neighbor algorithm

Various types and methods for distances can be applied in the Nearest Neighbor algorithm, we have used the distance mentioned in Chapter 2, Equation 2.3 to 2.15 as distance measure methods in our work.

We have noticed that best results was produced from the Spearman and Correlation distance calculation methods with accuracy up to 90 percent, see Figure 5.1. Some methods where not suitable and produced less than 40 percent of accuracy.

Algorithm:

Algorithm 3 GR using NN algorithm

Split the data into training set and testing set.
Initialize S_{train} and S_{test} sequences for both training and testing sets, where $S = (f_1, f_2, f_3, f_4, f_5)$ where f:frame, each has 20 points.
Label each sequence S_{train} with its gesture label.
For each sequence in S_{test}, find S_{testi} cell index C_j.
Find distance between S_i and S_train points.
Find minimum distance for S_j and label S_{testi} with L_j.

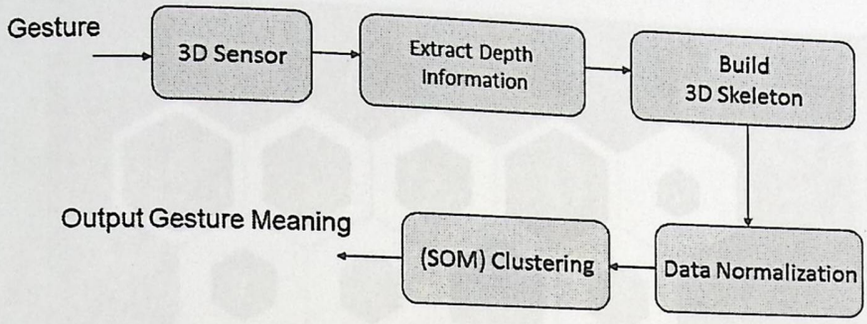


Figure 4.8: Gesture recognition using SOM block diagram

4.4 Gesture recognition using Self Organizing Map (SOM):

We have used the Self Organizing Map clustering technique to make gesture recognition, we provide it with the eleven gestures vectors, Figure 4.8 shows the block diagram we used to implement the SOM, after the data collection and normalization phase, the feature vectors passed to the SOM to be clustered and classified in order to output the gesture meaning or class.

4.4.1 Data preparation:

The normalized data which was collected by the KINECT sensor was prepared to be used for the Self Organizing Map (SOM). Each frame represented by 60 values contains the 20 joints positions in the 3D space. Furthermore, each five frames where reshaped together in sequential order to present the feature vector for the SOM with length 1200 value. We have split the data into two sets: training set and testing set.

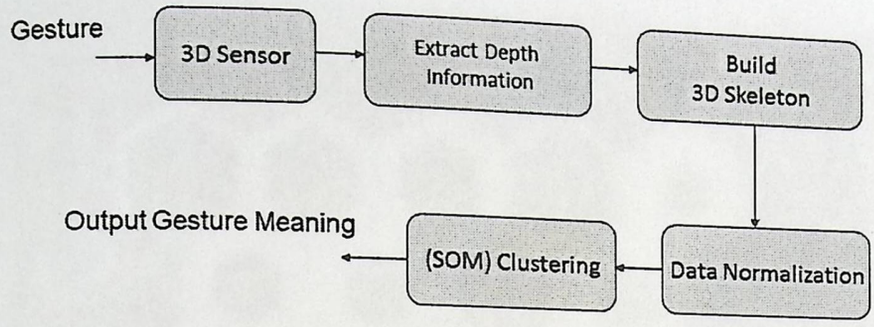


Figure 4.8: Gesture recognition using SOM block diagram

4.4 Gesture recognition using Self Organizing Map (SOM):

We have used the Self Organizing Map clustering technique to make gesture recognition, we provide it with the eleven gestures vectors, Figure 4.8 shows the block diagram we used to implement the SOM, after the data collection and normalization phase, the feature vectors passed to the SOM to be clustered and classified in order to output the gesture meaning or class.

4.4.1 Data preparation:

The normalized data which was collected by the KINECT sensor was prepared to be used for the Self Organizing Map (SOM). Each frame represented by 60 values contains the 20 joints positions in the 3D space. Furthermore, each five frames where reshaped together in sequential order to present the feature vector for the SOM with length 1200 value. We have split the data into two sets: training set and testing set.

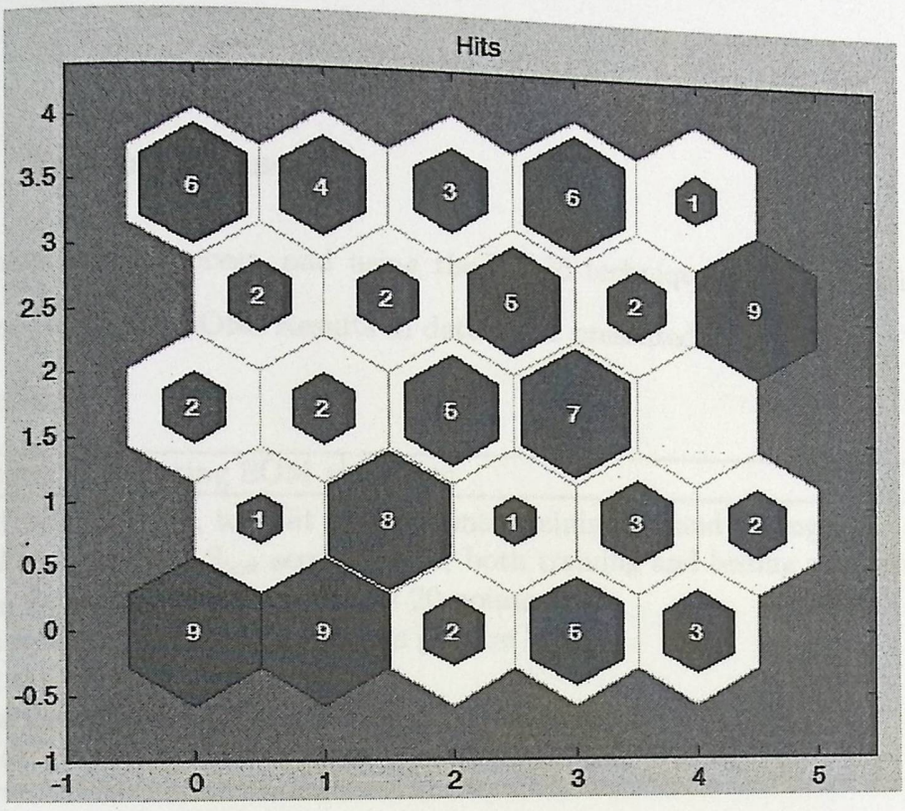


Figure 4.9: SOM Clustering

4.4.2 Training phase:

Two dimensional SOM was created and trained for this training data so each vector represents one gesture of five sequenced frames. Figure 4.9 shows the clustering results and how the gestures were clustered. We can notice that some clusters have more than one gesture, nearby gestures and far gestures positions in the map.

The decision of which cluster contains which gesture is based on majority function, we have applied a search technique with voting and counting clusters gesture, so the cluster is labeled based on the majority of gestures classified to belong to that

cluster.

4.4.3 Testing Phase:

After the training process, and using the KFold technique we have classified the testing set using the SOM. Results in details are presented in Chapter 5.

Algorithm:

Algorithm 4 GR using SOM algorithm

KFold with $K = 10$, to split the data into training set and testing set.
Initialise S_{train} and S_{test} sequences for both training and testing sets, where $S = (f_1, f_2, f_3, f_4, f_5)$, f:frame contains 20 points.
Label each sequence S_{train} with its gesture label L_j.
Train the SOM on S_{train}.
For each sequence in S_{test}, find S_{testi} cell index C_j.
Find C_j majority gestures labels L_j and label S_{testi} with L_j.

4.5 Gesture recognition using hidden markov model (HMM):

We have used the Hidden Markov Model and built 11 models for each gesture. Figure 4.10 shows the block diagram we used to implement the HMM's.

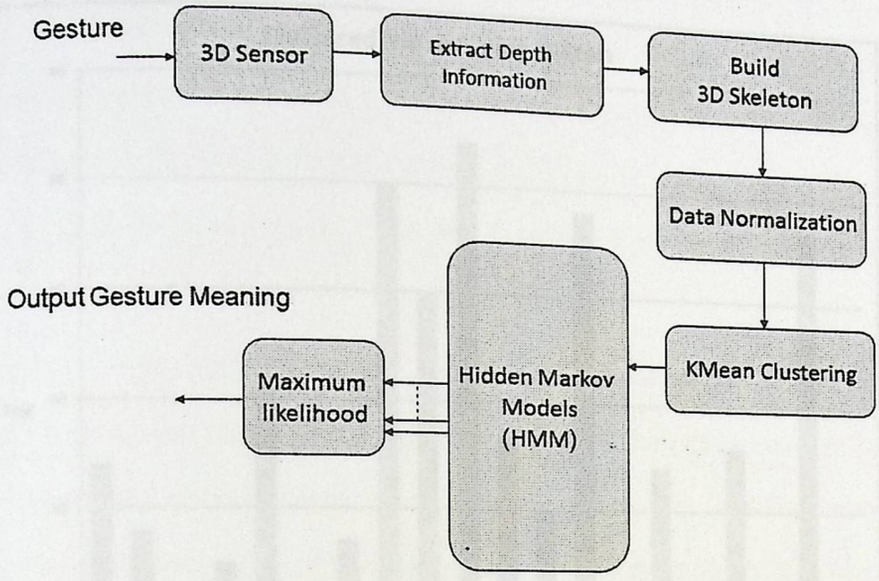


Figure 4.10: Gesture recognition using HMM block Diagram

4.5.1 Data preparation:

In order to use the HMM, we have use a clustering technique to reduce the dimensionality of the feature vector. We have tried to use the KMean and the Fuzzy CMean clustering algorithms. The number of clusters was selected experimentally.

Each frame represented by 60 values contains the 20 joints positions were passed to the KMean clustering algorithm to find the frames cluster as a value of 1-20 based on the number of clusters we have. Figure 4.11 shows the distribution of the frames over the K-Mean clusters. Each five sequenced frames provide one vector for the HMM. The data was split randomly to training and testing sets and we must mention here that the data which was used for clustering is the training set and doesn't include the testing set.

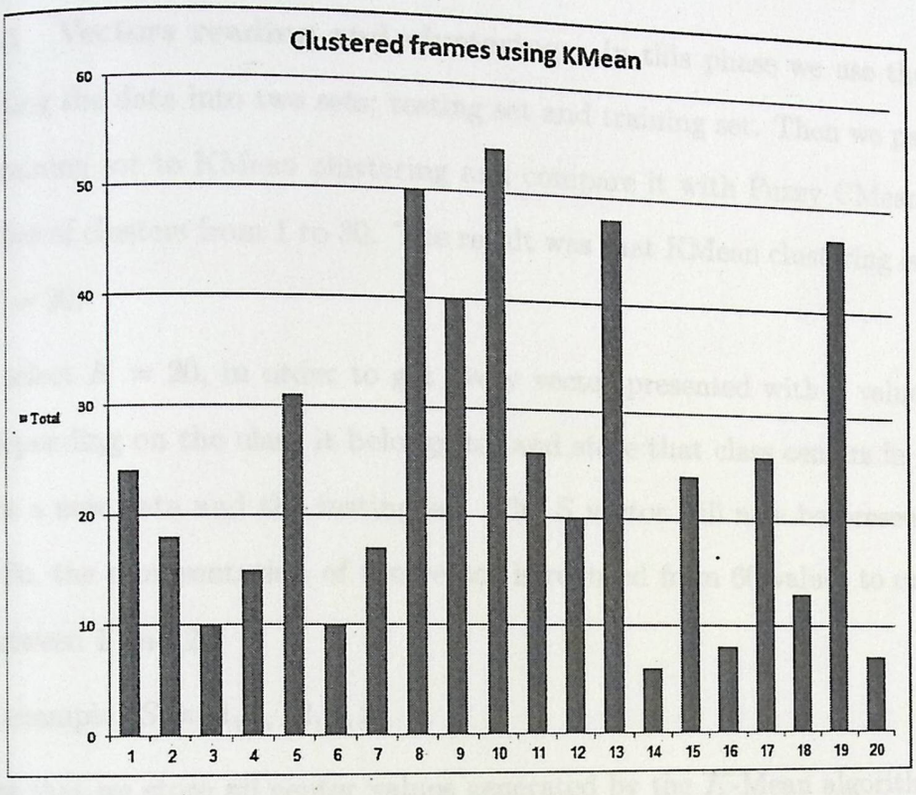


Figure 4.11: Frames distribution over KMean clusters

4.5.2 Training Phase:

We have created 11 Hidden Markov Models. Each model to be trained for one gesture. Each training gesture data were inputted for its model as a sequence of five values. Each value represents the frame cluster. The HMM emission and transition values were initiated randomly. We have used the Baum-Welch algorithm with tolerance 1e-6.

As a result, we had 11 trained hidden markov models, each has emotion and transition values to be used in the featured testing phase as we will describe in Chapter 5.

4.5.2.0.1 **Vectors reading and clustering:** In this phase we use the KFold for splitting the data into two sets; testing set and training set. Then we pass all V of the training set to KMean clustering and compare it with Fuzzy CMean results for number of clusters from 1 to 30. The result was that KMean clustering is diverge when $K = 20$.

We select $K = 20$, in order to get every vector presented with a value from 1 to 20, depending on the class it belongs to, and store that class centers in order to use it for a new data and the testing set. The S vector will now be presented as 5 values. So, the representation of the vector is reduced from 60 values to one single value between 1 and 20.

For example: $S = (1, 4, 12, 3, 2)$

After that we store all center values generated by the K-Mean algorithm to be used to find new and test data clusters.

Next we start the HMM process by building 11 models. Each model will lead to one single gesture. All training S, which belongs to each HMM, are passed to be trained on.

Determining the best number of states using KFold results, was chosen experimentally as two states for each HMM as mentioned in Chapter (5).

After the learning process for the hidden Markov models using Matlab, we were able to save the output Emission and Transition matrices to be loaded into an online system, which rebuilds the hidden Markov models depending on the values emission, transition and labels as they were exported from Matlab models.

4.5.2.0.1 **Vectors reading and clustering:** In this phase we use the KFold for splitting the data into two sets; testing set and training set. Then we pass all V of the training set to KMean clustering and compare it with Fuzzy CMean results for number of clusters from 1 to 30. The result was that KMean clustering is diverge when $K = 20$.

We select $K = 20$, in order to get every vector presented with a value from 1 to 20, depending on the class it belongs to, and store that class centers in order to use it for a new data and the testing set. The S vector will now be presented as 5 values. So, the representation of the vector is reduced from 60 values to one single value between 1 and 20.

For example: $S = (1, 4, 12, 3, 2)$

After that we store all center values generated by the K-Mean algorithm to be used to find new and test data clusters.

Next we start the HMM process by building 11 models. Each model will lead to one single gesture. All training S, which belongs to each HMM, are passed to be trained on.

Determining the best number of states using KFold results, was chosen experimentally as two states for each HMM as mentioned in Chapter (5).

After the learning process for the hidden Markov models using Matlab, we were able to save the output Emission and Transition matrices to be loaded into an online system, which rebuilds the hidden Markov models depending on the values emission, transition and labels as they were exported from Matlab models.

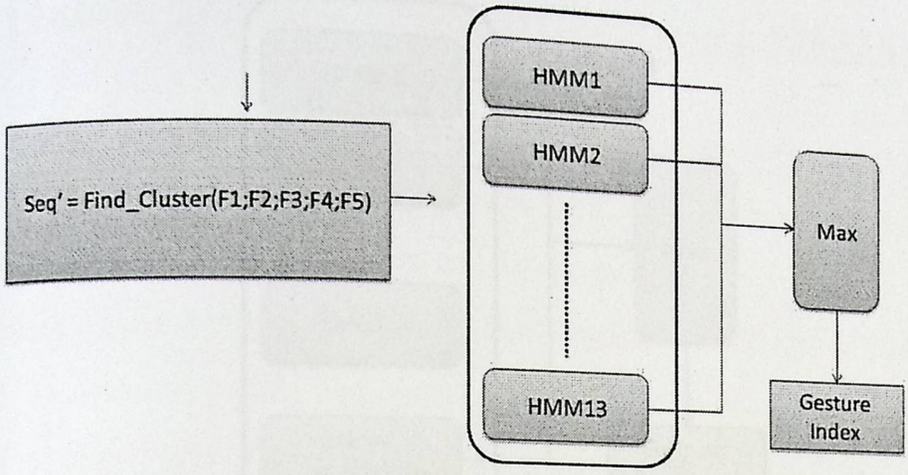


Figure 4.12: Proposed hidden markov models

4.5.3 Testing Phase:

After the training process we have clustered the testing set using the HMM. Results are presented in Chapter 5.

Algorithm:

Algorithm 5 GR Using HMM algorithm

Split the data into training set and testing set.
Initialize Raw_{train} and Raw_{test} sequences for both training and testing sets.
Apply KMean Clustering $Raw_{train} \rightarrow S_{train}$.
Find S_{test} labels using closest cluster center from the KMean.
Reshape sequance $S = (f_1, f_2, f_3, f_4, f_5)$, where f is a frame.
Label each sequence S_{train} with its gesture label L.
Train each HMM on its gestures from S_{train}.
for all S_{test} **do**
  Find HMM maximum likelihood. $S_{test} = L$
end for

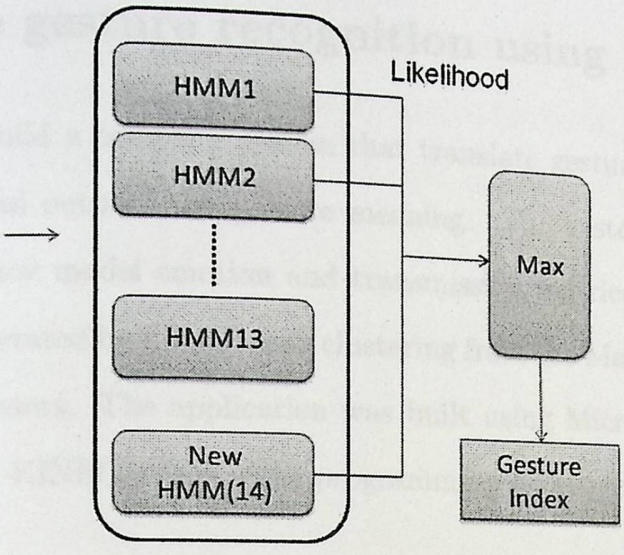


Figure 4.13: Incremental approach

4.5.4 Incremental approach

Our approach is based on learning each HMM for one gesture, sampled from a sequence of frames. This approach is incremental because when we need to add a gesture for the system to be identified, we need to add a new HMM and pass the sequence of that gesture. This will not affect previous learned models or the clustering phase because it will have different sequences to be identified with. Figure 4.13 shows the process of adding new trained model to the system.

To apply new gesture, we need to provide the system with new sample sequence by recording frames in the same way described previously. Then we find the collected data sequences by finding each frame cluster from the cluster centers we have. After this process we add the new HMM to the system and learn this new model with new sequences to identify.

4.6 Online gesture recognition using HMM:

In this phase we build a complete system that translate gestures captured by the KINECT sensor and outputs the gesture meaning. The system uses the output of the hidden markov model emotion and transmission matrices. It also uses the clusters centers generated by the K-Mean clustering from the Matlab to find the new captured frame clusters. The application was built using Microsoft Visual Studio 2010 and Microsoft KINECT SDK. The programming language is C #.

4.6.1 Building the models:

In this phase, the system reads the trained models matrices and builds a model for each gesture. These models are already trained and ready to be used.

4.6.2 Capturing data:

Captured data from the KINECT sensor is transferred and translated into skeleton joints, each joint represented by a point in a 3D space. For each frame joints we calculate that joint's position with reference to the Hip joint value. Also we apply the scaling and transformation update (as we mentioned before) on each captured joint to form a feature vector of the frame.

Then each frame is subtracted from the already defined centers generated by the K-Mean to find the distance between the captured frame and all other cluster centers. The min distance will be the cluster which this frame belongs to.

After we label the frame, the system adds this frame to an array and when we have 5 labels it forwards the sequence to be recognized by the 11 HMMs and compare the likelihood values each HMM provides. The maximum likelihood determines the gesture.

4.6.3 Two enhancements:

First enhancement is defining a threshold ω for sequences to be accepted or rejected. If the maximum likelihood generated from the 11 HMMs is less than ω, it will be rejected. In other words if the gesture made by the user does not belong to the set of gestures it knows then the HMMs will produce a low value of likelihood and it will be rejected.

The model with the highest likelihood is considered as the class this sequence belongs to, the difficulty we faced here was finding the start and end points of the sequence because of the continuity of the recognition process.

We have solved this issue by overlapping frames between sequences passed to the model because the point we started the sequence in is not deterministic for the user when he behaves in continues recognition mode. Each clustered frame is added to the list, then the sequence of FIVE clustered frame values are passed to the Hidden Markov Models to figure out which model provides us with maximum likelihood.

As an example, Figure 4.14 presents a screenshot of the final system. We can notice the RGB image, depth image, human skeleton and the result generated for the current gesture which is: Right hand moving side up.

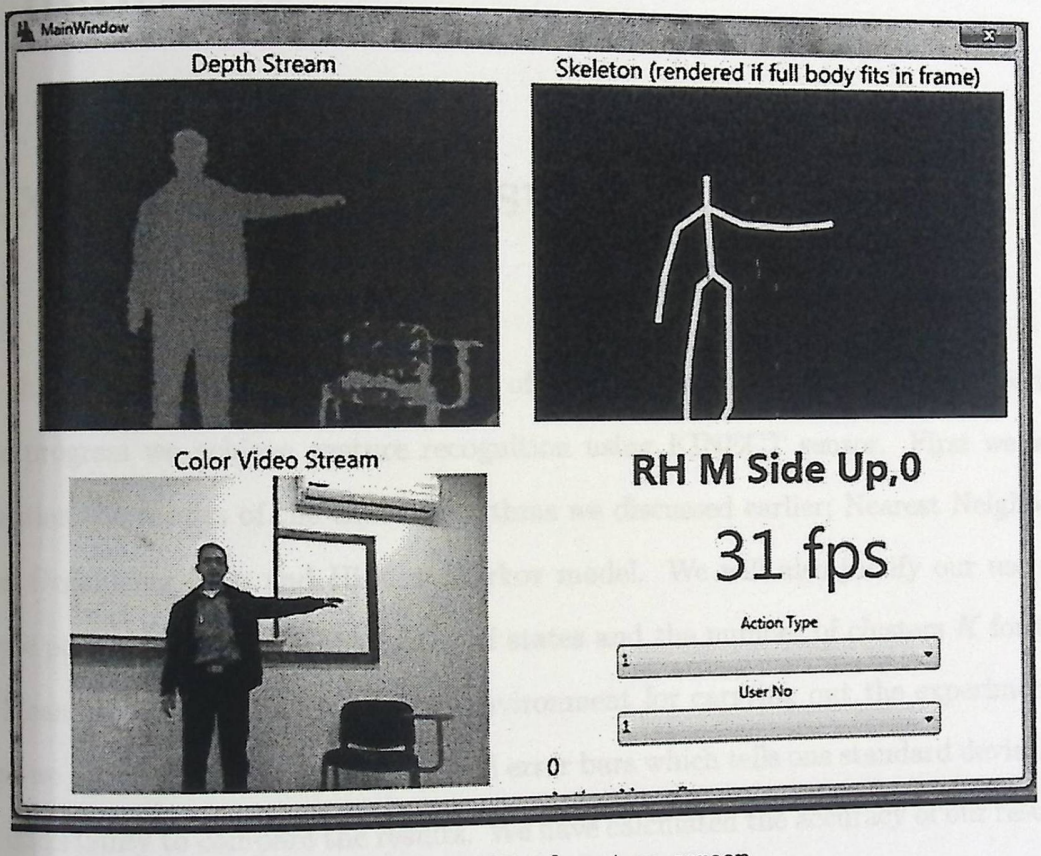


Figure 4.14: Final system screen

Chapter 5

Experimental results

In this chapter we will list the results of our experiments. These results describe the progress we achieve gesture recognition using KINECT sensor. First we will mention the results of the three algorithms we discussed earlier; Nearest Neighbor, Self Organizing Map and Hidden Markov model. We will also justify our use for some parameters like HMM number of states and the number of clusters K for the KMean algorithm. We use Matlab environment for carrying out the experiments, also we use 10 Fold cross validation and error bars which tells one standard deviation of uncertainty to compare the results. We have calculated the accuracy of our results as the number of correctly classified gestures over the total number of gestures.

Table 5.1: Results for 11 gestures using HMM

K	Euclidean	Seuclidean	Cityblock	Minkowski	Chebychev	Mahalanobis	Cosine	Cor
1	80.80	80.80	72.72	80.80	79.79	86.86	85.85	9
2	71.71	75.75	63.63	71.71	79.79	77.77	77.77	8
3	73.73	72.72	62.62	73.73	75.75	81.81	84.84	9
4	69.69	70.70	53.53	69.69	83.83	83.83	83.83	8
5	65.65	63.63	59.59	65.65	67.67	77.77	89.89	8
6	81.81	78.78	70.70	81.81	85.85	82.82	84.84	8
7	59.59	60.60	46.46	59.59	67.67	76.76	89.89	9
8	76.76	77.77	62.62	76.76	81.81	79.79	93.93	8
9	72.72	70.70	57.57	72.72	79.79	75.757	88.88	8
10	80.80	80.80	69.69	80.80	82.82	85.85	93.93	9
Average	73.33	73.23	61.91	73.33	78.48	80.90	87.37	9

5.1 Gesture recognition using Nearest Neighbor algorithm

Listed below the results for using Nearest Neighbor algorithm with different distance methods mentioned in Chapter 2, equation 2.3 to 2.15, sorted using best results obtained.

We can notice for Table 5.1 that Spearman and Correlation formulas obtained best values over other methods. Spearman (see Equation 2.15 works on two phases, first it computes the rank of the vector values and then use it to find the similarity measure between the sequences we provide, so similar sequences will have similar rank order for the moving joints. Correlation distance depends on the angle between the two sequence vectors, in this case when the two vectors which belongs to the same gestures has any value of shift or scale, then this value will be omitted because

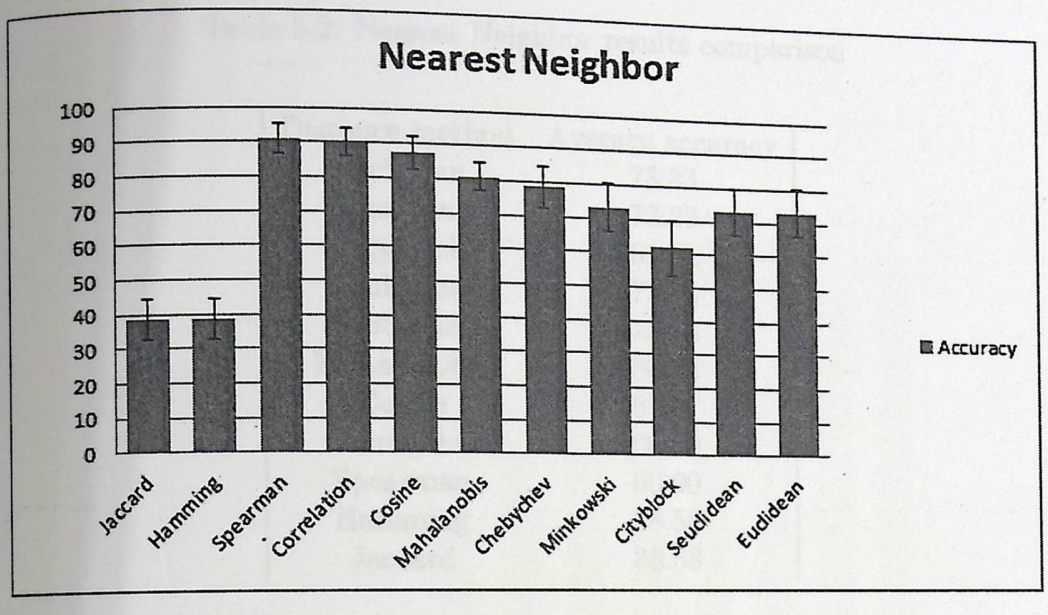


Figure 5.1: Nearest Neighbor algorithm accuracy using different distance methods

the angle produced will be same.

Hamming and Jaccard are not suitable for this task. Hamming can be used for binary data while Jaccard method based on counting equal points over the set (see Equation 2.14).

Table 5.2: Nearest Neighbor results comparison

Distance method	Average accuracy
Euclidean	73.33
Seuclidean	73.23
Cityblock	61.91
Minkowski	73.33
Chebychev	78.48
Mahalanobis	80.90
Cosine	87.37
Correlation	90.30
Spearman	90.90
Hamming	38.58
Jaccard	38.58

Table 5.3: Results using SOM

K	1	2	3	4	5	6	7	8	9	10	Avg
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	3	7	11	12	17	21	24	27	31	33	18.6
4	35	38	38	39	43	43	46	49	51	54	43.6
5	56	56	59	63	63	67	67	67	68	68	63.4
6	69	69	69	68	67	67	69	71	71	73	69.3
7	74	74	74	75	77	78	77	76	75	74	75.4
8	75	77	76	74	74	73	73	73	74	74	74.3
9	71	69	67	68	66	67	68	68	67	67	67.8
10	69	71	72	72	72	70	69	69	69	69	70.2

5.2 Gesture recognition using Self Organizing Map(SOM) algorithm

We can see the results for using SOM with $n \times n$ dimensionality as shown in Table 5.3.

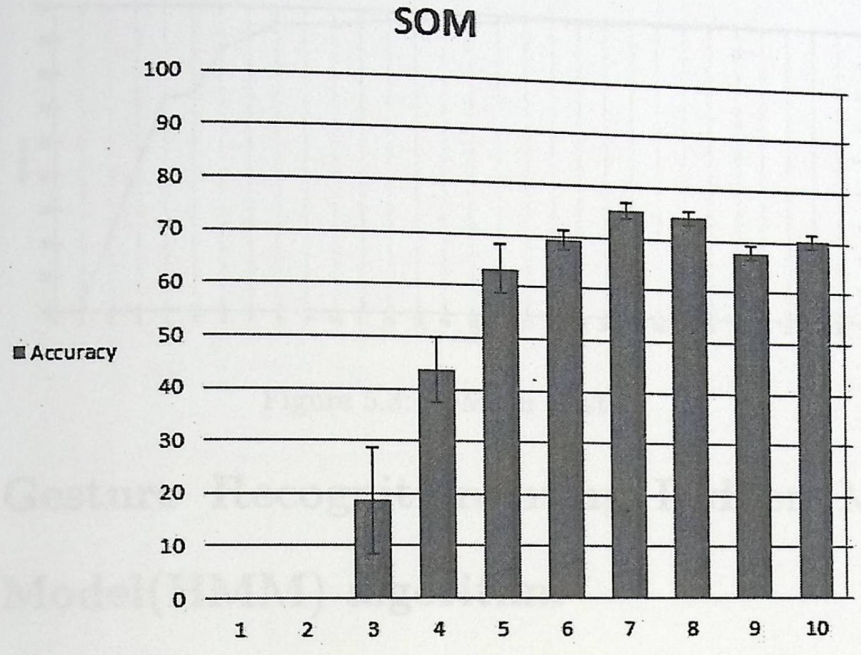


Figure 5.2: Self Organizing Map algorithm accuracy using dimensions

We can notice form Table 5.3 that best results reached when we used 7×7 SOM with accuracy of 75 percent. we can justify the reason of this result because when the dimensionality of the SOM is very low, multiple gestures with different classes will be placed in the same class due to the similarity between them. On the other hand, if the number of clusters is large it will cluster the gestures into small classes where we need them to be grouped based on their type. This number is so related to the number of gestures we use.

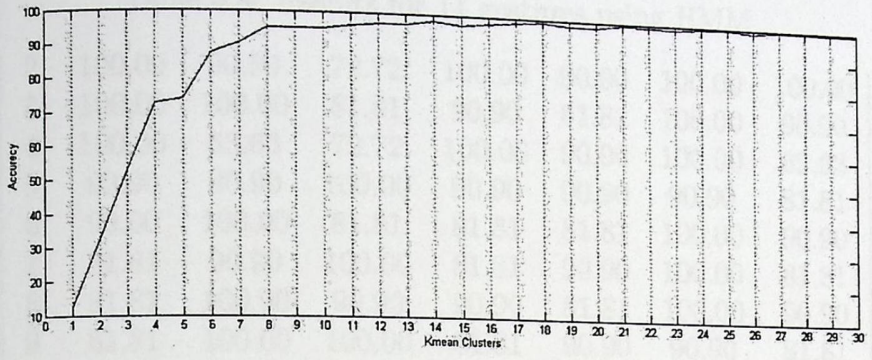


Figure 5.3: K-Mean clusters

5.3 Gesture Recognition using Hidden Markov Model(HMM) algorithm

The results we obtain from HMM algorithm was better than we obtain from NN and SOM, below the results from using HMM algorithm, we have applied the HMM multiple times with different number of states.

The first step in this phase was applying KMean clustering for each frame vector with 20 clusters. That provides a sequence of 5 frames for each gesture. Each gesture was recorded 10 times and values passed to the HMM related to that gesture. Using cross validation with $K = 10$ we have achieved an accuracy of 98 Percent.

We need to cluster the frames before the HMM training phase, we have experimentally chosen $K = 20$ based on the results we notice in Figure 5.3.

As we notice, the number of states has no major affect on the obtained results. We made another two experiments to insure this idea by the following results which are the result of the KFold on the HMM states number between 1-5 and tolerance

Table 5.4: Results for 11 gestures using HMM

2	100.00	90.90	72.72	100.00	90.90	100.00	100.00
3	100.00	100.00	81.81	90.90	81.81	100.00	90.90
4	100.00	63.63	72.72	100.00	90.90	100.00	63.63
5	90.90	90.90	100.00	90.90	90.90	90.90	81.81
6	90.90	100.00	81.81	81.81	81.81	100.00	90.90
7	81.81	90.90	100.00	81.81	90.90	100.00	81.81
8	81.81	100.00	90.90	90.90	81.81	100.00	90.90
9	81.81	100.00	100.00	81.81	90.90	90.90	81.81
10	90.90	90.90	100.00	90.90	90.90	90.90	100.00

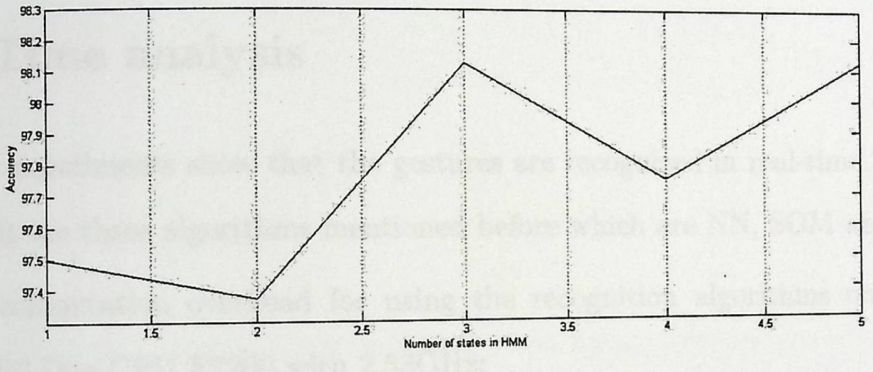


Figure 5.4: Accuracy of HMM with 1-5 states and tolerance 0.01

0.01:

Using Matlab, we have implemented the previous test for the 11 gestures (see Figure 4.6) and cross validation with $K = 10$ which achieves an accuracy between 96 to 98 percent. We can notice that gestures we choose has similarity manner. The gestures were chosen from the American sign language library to make the system close to being used in reality.

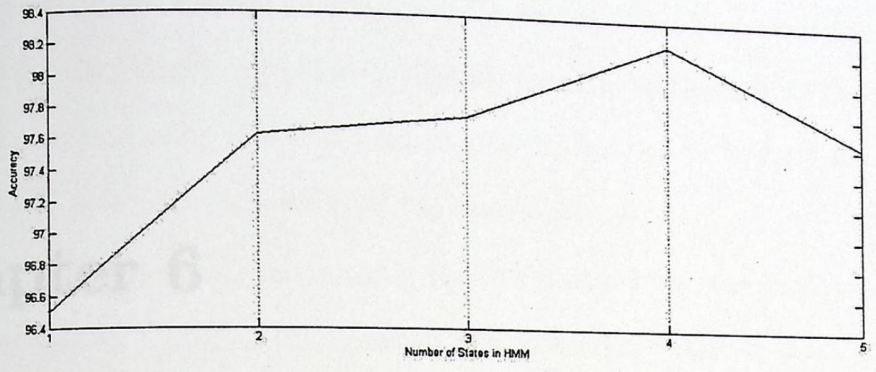


Figure 5.5: Accuracy of HMM with 1-5 states and tolerance 0.01

5.4 Time analysis

The time experiments show that the gestures are recognized in real-time. We have implement the three algorithms mentioned before which are NN, SOM and HMM.

The computation overhead for using the recognition algorithms on Intel(R) Core(TM)2 Duo CPU E7200 with 2.53GHz:

Two states HMM needs 1.67 milliseconds/frame, while SOM with dimensionality 7×7 took 8 milliseconds/frame, and NN using Spearman distance took 9.21 milliseconds/frame which keeps the recognition system work in a real-time manner.

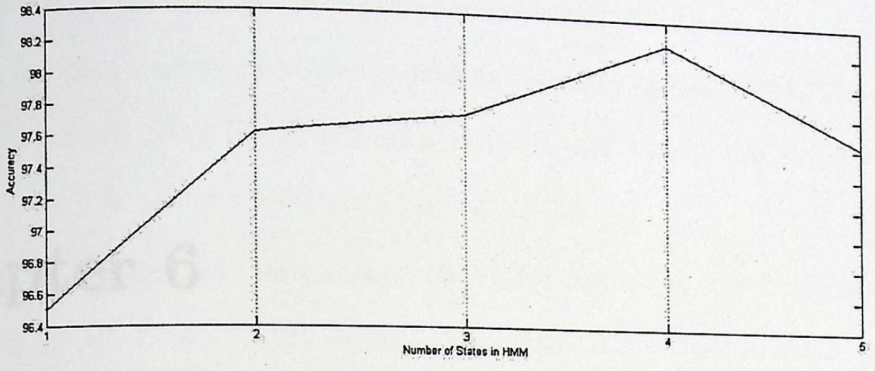


Figure 5.5: Accuracy of HMM with 1-5 states and tolerance 0.01

5.4 Time analysis

The time experiments show that the gestures are recognized in real-time. We have implement the three algorithms mentioned before which are NN, SOM and HMM.

The computation overhead for using the recognition algorithms on Intel(R) Core(TM)2 Duo CPU E7200 with 2.53GHz:

Two states HMM needs 1.67 milliseconds/frame, while SOM with dimensionality 7×7 took 8 milliseconds/frame, and NN using Spearman distance took 9.21 milliseconds/frame which keeps the recognition system work in a real-time manner.

Chapter 6

Conclusion and future work

6.1 Conclusions

We have presented a gesture recognition system using different machine learning algorithms for the KINECT sensor.

First we capture data from the KINECT sensor which was translated into human body skeleton. Then we normalize the joints positions based on reverence selected joint by Appling transformation and rotating over each recorded joint.

The normalization process makes the algorithm invariant for changing in position with reverence to the sensor position.

The recognition algorithms we have used were Nearest Neighbor (NN) Algorithm, Self Organizing Map (SOM) and Hidden Markov Model (HMM). Through our experiments we have noticed that the HMM algorithm archives best recognition

for gestures with an average of 95 percent accuracy. We have used the K-Mean algorithm for data clustering before providing the data to the HMM. The output of gesture recognition using HMM is a set of models that can be learned and added to the system to gain better usability of the algorithm.

We built a recognition system using the HMM algorithm, the final system builds the trained models based on the emission and transition matrices. Each gesture has an already trained model and ready to be used. The model with the highest likelihood is considered as the class this sequence belongs to, we also made a threshold for the sequences to be accepted or rejected based on the likelihoods they provide.

Each clustered frame by the K-Mean is recorded, then the sequence of 5 clustered frame values are passed to the HMM to figure out which model provides us with maximum likelihood.

We have using different distance methods for NN algorithm and we obtain about 80 percent using Spearman distance. Other distances did not provide good results. Furthermore SOM algorithm provides about 68 percent of recognition accuracy using a map of 7×7 dimensions. All three algorithms works in real-time, we have used 10 fold cross validation.

The recognition process runs in real-time, where two states HMM took 1.67 milliseconds, SOM with dimensionality 7×7 took 8 milliseconds and NN using Spearman distance took 9.21 milliseconds. We used error bars and cross validation method to verify which results are significantly better than others.

6.2 Future work

We encourage to continue the work in this study by taking advantage of the RGB image captured from the KINECT for recognize hand parts and fingers which depth data accuracy doesn't provide. There are many applications for adding this feature to the system so it will recognize many accurate and small gestures made by hand.

Another method we can suggest here where adding the face gestures and emotions from the 2D captured image and use the impression as a feature for the gestures recognized. Also the use of two Kinect sensors with different viewing angles may increase recognition accuracy.

Bibliography

[1] K. Alsabti, S. Ranka, and V. Singh. An efficient k-means clustering algorithm. In *Proceedings of International Parallel Processing Symposium*, 1998.

[2] L E Baum, T Petrie, G Soules, and N Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[3] Benjamin C Bedregal, Antonio C R Costa, and Gragaliz P Dimuro. Fuzzy rule-based hand gesture recognition. *Artificial Intelligence*, 217:1–10, 2006.

[4] Phil Blunsom. Hidden markov models, 2004.

[5] Pia Breuer, Christian Eckes, and Stefan Müller. Hand gesture recognition with a novel ir time-of-flight range camera: a pilot study. In *Proceedings of the 3rd international conference on Computer vision/computer graphics collaboration techniques*, MIRAGE'07, pages 247–260, Berlin, Heidelberg, 2007. Springer-Verlag.

[6] Hung-Ching Chen, Mark Goldberg, Malik Magdon-Ismail, and William A. Wallace. Reverse engineering an agent-based hidden markov model for complex social systems. In *Proceedings of the 8th international conference on Intelligent data engineering and automated learning*, IDEAL'07, pages 940–949, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] S. C. Chen, K. H. Chang, C. K. Liang, S. W. Lin, T. H. Huang, M. C. Hsieh, C. H. Yang, C. M. Wu, and C. H. Lo. 3d gesture language recognition system. In *4th Kuala Lumpur International Conference on Biomedical Engineering 2008*, volume 21 of *IFMBE Proceedings*, pages 773–777. Springer Berlin Heidelberg, 2008. 10.1007/978-3-540-69139-6-192.

[8] D. Coomans and D.L. Massart. Alternative k-nearest neighbour rules in supervised pattern recognition : Part 1. k-nearest neighbour classification by using alternative voting rules. *Analytica Chimica Acta*, 136(0):15 – 27, 1982.

[9] Microsoft Corp. *programming guide for kinect sdk.* Microsoft, 2011.

[10] Paul Doliotis, Alexandra Stefan, Chris Mcmurrough, David Eckhard, and Vassilis Athitsos. Comparing gesture recognition accuracy using color and depth information. *Hand The*, 2011.

[11] Heng Du and To TszHang. Hand gesture recognition using kinect. *Computer Engineering*, 2011.

[12] Charles Elkan. Nearest neighbor classification. 2011.

[13] Mahmoud Elmezain, Ayoub Al-Hamadi, Jorg Appenrodt, and Bernd Michaelis. A hidden markov model-based continuous gesture recognition system for hand motion trajectory. *2008 19th International Conference on Pattern Recognition*, pages 1–4, 2008.

[14] Jianming Guo. Hand gesture recognition and interaction with 3d stereo camera supervise by dr . hongdong li. *Hand The*, (November):1–34, 2011.

[15] A K Jain, M N Murty, and P. J. Flynn. Data clustering: A review, 1999.

[16] Adam Kendon. Some relationships between body motion and speech. In Aron Seigman and Benjamin Pope, editors, *Studies in Dyadic Communication*, pages 177–216. Pergamon Press, Elmsford, NY, June 1972.

[17] C Keskin, O Aran, and L Akarun. *Real time gestural interface for generic applications.* Citeseer, 2005.

[18] C. Keskin, A. Erkan, and L. Akarun. Real time hand tracking and 3d gesture recognition for interactive interfaces using hmm. In *In Proceedings of International Conference on Artificial Neural Networks*, 2003.

[19] C Keskin, A Erkan, and L Akarun. Real time hand tracking and 3d gesture recognition for interactive interfaces using hmm. *ICANNICONIPP*, page 2629, 2003.

[20] Microsoft Corp. Redmond WA. Kinect. Kinect for xbox 360, 2011.

[21] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. Som pak: The self-organizing map program package, 1996.

[22] Herve Lahamy and Derek Litchi. Real-time hand gesture recognition using range cameras. *The International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences on CDROM*, page 38, 2010.

[23] Luigi Lamberti and Francesco Camastra. Real-time hand gesture recognition using a color glove. In *Proceedings of the 16th international conference on Image analysis and processing: Part I*, ICIAP'11, pages 365–373, Berlin, Heidelberg, 2011. Springer-Verlag.

[24] Erik G. Learned-Miller. *Supervised Learning and Bayesian Classification.* PhD thesis, 2011.

[25] S. Malassiotis, N. Aifanti, and M. G. Strintzis. A gesture recognition system using 3D data. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pages 190–193, 2002.

[26] A A Markov. An example of statistical investigation of the text eugene onegin concerning the connection of samples in chains. *Science in Context*, 19(04):591, 2006.

[27] MATLAB. *version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts, 2010.

[28] D. Pelleg and A. Moore. X-means: Extending K-means with efficient estimation of the number of the clusters. In *Proceedings of International Conference on Machine Learning*, 2000.

[29] O. Joaquín Pérez, R. Rodolfo Pazos, R. Laura Cruz, S. Gerardo Reyes, T. Rosy Basave, and H. Héctor Fraire. Improving the efficiency and efficacy of the k-means clustering algorithm through a new convergence condition. In *Proceedings of the 2007 international conference on Computational science and its applications - Volume Part III*, ICCSA'07, pages 674–682, Berlin, Heidelberg, 2007. Springer-Verlag.

[30] D. T. Pham, S. S. Dimov, and C. D. Nguyen. Selection of k in k -means clustering. *Proceedings of the I MECH E Part C Journal of Mechanical Engineering Science*, 219(1):103–119, January 2005.

[31] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.

[32] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 532–538. Springer US, 2009.

[33] Matthias Rehm, Nikolaus Bee, and Elisabeth André. Wave like an egyptian: accelerometer based gesture recognition for culture specific interactions. In *Proceedings of the 22nd British HCI Group Annual Conference on People and*

Computers: Culture, Creativity, Interaction - Volume 1, BCS-HCI '08, pages 13–22, Swinton, UK, UK, 2008. British Computer Society.

[34] Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition with kinect sensor. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 759–760, New York, NY, USA, 2011. ACM.

[35] Zhou Ren, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 1093–1096, New York, NY, USA, 2011. ACM.

[36] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-Time Human Pose Recognition in Parts from a Single Depth Image. In *CVPR*, 2011.

[37] Mu-Chun Su and Hsiao-Te Chang. Fast self-organizing feature map algorithm. *Neural Networks, IEEE Transactions on*, 11(3):721–733, May 2000.

[38] Matthew Tang. Recognizing hand gestures with microsoft s kinect. *stanfordedu*, 14(4):303–313, 2011.

[39] Michael Van Den Bergh, Daniel Carton, Roderick De Nijs, Nikos Mitsou, Christian Landsiedel, Kolja Kuehnlenz, Dirk Wollherr, Luc Van Gool, and Martin Buss. Real-time 3d hand gesture interaction with a robot for understanding directions from humans. *2011 ROMAN*, pages 357–362, 2011.

[40] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Trans. Graph.*, 28(3):63:1–63:8, July 2009.

[41] A D Wilson and A F Bobick. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.

[42] A D Wilson and A F Bobick. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.

[43] Andrew D. Wilson, Student Member, Ieee Computer Society, Aaron F. Bobick, and Ieee Computer Society. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:884–900, 1999.

[44] Andrew David Wilson, Aaron F. Bobick, Bruce M. Blumberg, Stephen A. Benton, and Andrew David Wilson. Recognition of human gesture, 2000.

[45] Guangqi Ye, J J Corso, and G D Hager. Gesture recognition using 3d appearance and motion features. *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 160–160, 2004.

[46] Guangqi Ye, Jason J. Corso, and Gregory D. Hager. Gesture recognition using 3d appearance and motion features. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 10 - Volume 10*, pages 160–, Washington, DC, USA, 2004. IEEE Computer Society.

Appendix A

Appendix

A.1 Preparing the KINECT

We have prepare the KINECT sensor to make the data acquisition phase, In this phase we have changed the interface of the KINECT cable which is a special 9 pin interface produced by Microsoft for this device to the XBOX gaming station [20] to USB interface with power support, the device cable contains 8 connected pins as follows: 2 data bins, 3 power pins and 3 Grounded pins.

We have replaced this interface with the USB interface by connecting the data pins, one +5V power line, grounded line and supply the other +12v power line with external power supply.

After testing and comparing the four drivers and SDK's mentioned before, we conclude that the Microsoft SDK has the best performance and the most accurate

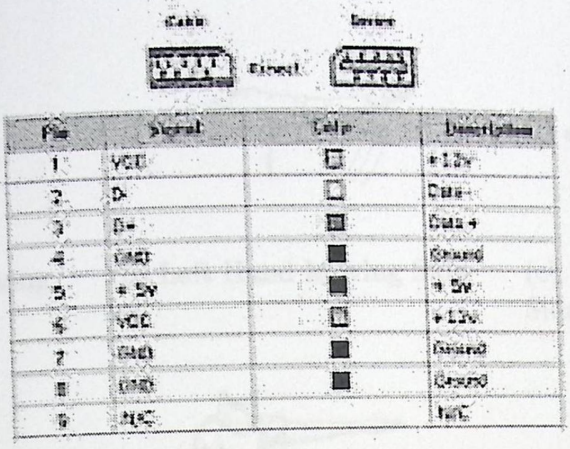


Figure A.1: Maping bins

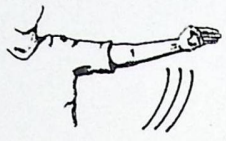
output.

After that we have tried more than one driver software for the KINECT sensor, first we have OpenKINECT driver which was the first online Open source driver supporting 2D images and depth images acquisition and C sharp, C++ applications to display the data collected from the sensor. The disadvantage of this driver was complicated installation process.

After that we tried CL NUI Platform driver for KINECT, the production of this driver was on October-12th-2010. The installation and setup process was easier than the OpenKinect, and it uses a WPF/C sharp application with source code in order to display both 2D and depth gray scale images.

The third drive was provided from Open Natural Interaction (OpenNI), OpenNI which provides an open source library, and the applications could be used for KINECT sensor, such as Skeleton Tracker. It uses the depth image and builds a histogram to detect human body and track it. It also provides up to two users at

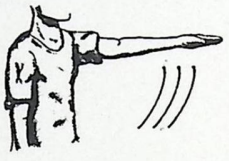
(a) Right Hand Moving Side Up



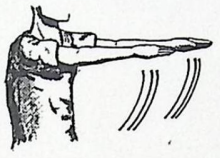
(b) Left Hand Moving Side up

(c) Both Hands Moving Side Up

(d) Right Hand Moving Front Up

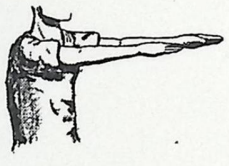


(e) Left Hand Moving Front Up



(f) Both Hands Moving Front Up

(g) Both Hands Steady Side Up



(h) Both Hands Steady Front Up

Figure A.2: Second expirement

a time.

The fourth driver is SDK, which was published from Microsoft Corp, on July-29th- 2011. It is a beta version of the SDK for KINECT sensor and it is easy to install. It also includes Skeleton Tracker with the ability to track up to two users at a time.