1-1-2012

Trajectory Analysis on Spherical Self-Organizing Maps With Application to Gesture Recognition

Artur Oliva Gonsales
Ryerson University

Trajectory Analysis on Spherical Self-Organizing Maps with Application to Gesture Recognition

by

Artur Oliva Gonsales

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Masters of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2012

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

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

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

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

Trajectory Analysis on Spherical Self-Organizing Maps with application to gesture recognition

Masters of Applied Science 2012

Artur Oliva Gonsales

Electrical and Computer Engineering

Ryerson University

Abstract

In this work, a new approach to gesture recognition using the properties of Spherical Self-Organizing Map (SSOM)is investigated. Bounded mapping of data onto a SSOM creates not only a powerful tool for visualization but also for modeling spatiotemporal information of gesture data. The SSOM allows for the automated decomposition of a variety of gestures into a set of distinct postures. The decomposition naturally organizes this set into a spatial map that preserves associations between postures, upon which we formalize the notion of a gesture as a trajectory through learned posture space. Trajectories from different gestures may share postures. However, the path traversed through posture space is relatively unique. Different variations of posture transitions occurring within a gesture trajectory are used to classify new unknown gestures. Four mechanisms for detecting the occurrence of a trajectory of an unknown gesture are proposed and evaluated on two data sets involving both hand gestures (public sign language database) and full body gestures (Microsoft Kinect database collected in-house) showing the effectiveness of the proposed approach.

Acknowledgement

I would like to express my sincere gratitude to Prof. Matthew Kyan, who has guided me through my master's degree and have provided a lot of support and help during my studies. To all my family and friends for their support and understanding. To all my colleagues and friends from Ryerson Multimedia Lab who have helped me by sharing their knowledge with me. Especially to Adrian Bulzacki for collecting and providing the Microsoft Kinect data set used in this thesis. To Naimul Mefraz Khan, for his support in the topic of Spherical Self-Organizing Maps. Finally, to Chun-Hao Wang for providing network access to the Microsoft Kinect data set.

Contents

1 Introduction **1**

1.1 Introduction to Gestures and Gesture Recognition 1

1.1.1 Recent developments in acquisition technology 6

1.1.2 Traditional approaches to gesture recognition 7

1.1.3 Domain independent approach to gesture recognition 11

1.1.4 Introduction to gesture data sets used in this thesis 12

1.1.5 Mathematical definition of a gesture 15

1.2 A brief introduction to Supervised and Unsupervised Learning 17

1.3 Thesis Overview 19

1.4 Thesis Contributions 21

1.4.1 Publications 21

2 Self organizing approaches to gesture recognition **22**

2.1 Introduction 22

2.2 Neural Networks 23

2.3 Self-Organization 26

2.3.1 Principles of Self-Organization 26

2.3.2 The Kohonen Self-Organizing Feature Map (SOFM) 28

2.4 Gesture recognition using Self-Organizing Maps and Trajectories 31

2.5 Summary 41

3 Trajectory analysis of temporal data using Self-Organizing Maps **43**

3.1 Introduction 43

3.2 Spherical SOM (SSOM) 43

3.2.1 Topology of the SOFM lattice 43

3.2.2 Weight adaptation strategy in the spherical SOM 45

3.2.3 Significance of the Spherical Topology 49
3.3 Temporal data analysis using SOM and SSOM 50
3.3.1 SOM based approaches for video analysis 50
3.3.2 Video information analysis based on SSOM 56
3.4 Four approaches to gesture recognition through trajectory analysis 63
3.4.1 Approach #1: gesture recognition using all postures 64
3.4.2 Approach #2: gesture recognition using weighted aggregation of all postures 65
3.4.3 Approach #3: gesture recognition using posture transitions 67
3.4.4 Approach #4: gesture recognition using weighted aggregation of all posture transitions 68
3.5 Summary 69

4 Experimental Results **70**
4.1 Experimental setup 70
4.2 Results 72
4.2.1 Approach #1: gesture recognition using all postures 72
4.2.2 Approach #2: gesture recognition using weighted aggregation of all postures 86
4.2.3 Approach #3: gesture recognition using posture transitions 86
4.2.4 Approach #4: gesture recognition using weighted aggregation of all posture transitions 90
4.3 Comparison of results 90
4.4 Summary 93

5 Conclusions and Future Work **96**
5.1 Conclusions 96
5.2 Thesis Contributions 97
5.3 Future Work 98

List of Tables

1.1 Australian Sign Language Variable description 14
1.2 Australian Sign Language gesture data 14
1.3 Microsoft Kinect Full body gesture data 15
1.4 Microsoft Kinect Full body gesture variables 18

4.1 Recognition rate for Kinect dataset: Approach #1 83
4.2 Recognition rate for PowerGlove dataset: All postures 86
4.3 Recognition rate for Kinect dataset: Weighted aggregation of all postures . . 86
4.4 Recognition rate for PowerGlove dataset: Weighted aggregation of all postures 87
4.5 Recognition rate for Kinect dataset: Using posture transitions 87
4.6 Recognition rate for PowerGlove dataset: Using posture transitions 90
4.7 Recognition rate for Kinect dataset: Weighted aggregation of all posture transitions 90
4.8 Recognition rate for PowerGlove dataset: Weighted aggregation of all posture transitions 91
4.9 Results comparison for Microsoft Kinect dataset 92
4.10 Results comparison for Australian Sign Language (ASL) dataset 92

List of Figures

1.1 Kendon's Gesture Continuum 3
1.2 Examples of human postures 4
1.3 Pattern Recognition System (ideal for static gesture recognition) 5
1.4 Recent touch technology 7
1.5 Examples of body motion sensors and controllers. From top to bottom: Sony Playstation Move, Nintendo Wii, Microsoft Kinect 8
1.6 Recurrent Neural Network 10
1.7 Dynamic Time Warping 11
1.8 Nintendo PowerGlove 13
1.9 Microsoft Kinect skeleton 16
1.10 Representation of a single feature 17
1.11 The problem of defining a cluster. The clusters are formed based on the coordinates of the data points 20

2.1 A Simple Neuron [1] 24
2.2 Example of a basic Neural Network 25
2.3 Self-Amplification, Competition and Co-operation in learning: w_{k*} represents the synaptic vector of the winning neuron (competition), which adapts (self amplifies) toward the input thereby strengthening its correlation with future inputs in this region. At the same time, associative memory is imparted to neighbouring neurons (co-operation): related to the winner through some defined or inferred topology 28
2.4 Mapping samples from an input space onto a SOM lattice of prototypes; showing the winning node (best matching unit: BMU of the current input x_i (top), Gaussian neighbourhood function (bottom) 30

2.5 Flow chart and data structures for the learning process based on Oshita and Matsunaga method [2] 32
2.6 Winner neurons for a gesture element [3] 34
2.7 Time invariant recognition with sparse code [3] 35
2.8 Sparse code for a backward gesture element [3] 36
2.9 Sparse code with time transition [3] 36
2.10 Sparse code for gesture elements and gesture [3] 37
2.11 Correspondence of gesture trajectory points to their respective BMUs on the SOM. The BMUs constitute the states of the Markov models [4] 40
2.12 Markov model for a gesture's optical flow [4] 40

3.1 One-dimensional SOM ($\Re^1$) 44
3.2 Data association ($\Re^2$ space (3.2a)) and ($\Re^3$ space (3.2b) 45
3.3 Spherical SOM with quadrilateral elements 46
3.4 Data association in a spherical SOM 47
3.5 The spherical self-organizing map 48
3.6 The feature vector is calculated from the raw data. Then the vector is mapped on the SOM by finding the best-matching map unit [5] 51
3.7 Segments of trajectory at three different time steps. The trajectory is illustrated with a black dotted line [5] 51
3.8 Representative frames and SOM signatures of three video shots [6] 53
3.9 Growing Self-Organizing map after training [7] 55
3.10 Sample frames used for training the Spherical SOFM (taxi scene) 58
3.11 Spherical SOM after training. The color green, red, and blue show the different edges belonging to the three different shots. 60
3.12 Spherical SOM after training. The color green, red, and blue show the different edges belonging to the three different shots. 60
3.13 Video scene/shot separation using Spherical SOM 61
3.14 Spherical SOM after training with individual features 62
3.15 Spherical SOM after training with individual features 62
3.16 Linkage between keyframes in the video shots 63
3.17 Temporal sequence of postures representing an arbitrary gesture. In this example a simple gesture consisting of 5 postures is displayed. The mapping of the gesture is shown as a trajectory on the SSOM in red 65

3.18 Showing the gesture recognition process: the classification occurs by creating a counter for the unknown gesture (blue trajectory), which counts the common instances or postures between the unknown gesture trajectory and the already learned by the SSOM trajectories (trajectories 1 & 2, red and green respectively). A histogram is then built based on these counters and a gesture is classified according to the highest value obtained in this histogram 66
3.19 Showing the gesture recognition process: Assume the SSOM learned the path of two trajectories from gesture 1 class (pink & red). These two trajectories have common postures in their path. In this method we look at the frequency information obtained from trajectories within one gesture class. Specifically, we add a weight factor representing how frequent a specific posture was found in the path of a trajectory from a specific gesture class. The unknown blue trajectory is classified using this frequency information 67
3.20 In this approach we use temporal information obtained from trajectories, specifically we use posture transitions represented as R_i. A histogram is built based on how many common posture transitions occur between the unknown gesture trajectory and all the gesture trajectories learned by the SSOM . . . 68

4.1 PowerGlove Gesture trajectories 1. First row: Alive; Second row: All; Third row: Answer; Fourth row: Boy 73
4.2 PowerGlove Gesture trajectories 2. First row: Building; Second row: Buy; Third row: Change; Fourth row: Cold 74
4.3 PowerGlove Gesture trajectories 3. First row: Come; Second row: Computer; Third row: Cost; Fourth row: Crazy 75
4.4 PowerGlove Gesture trajectories 4. First row: Danger; Second row: Deaf; Third row: Different; Fourth row: Draw 76
4.5 PowerGlove Gesture trajectories 5. First row: Drink; Second row: Eat; Third row: Exit; Fourth row: Forget 77
4.6 Microsoft Kinect Gesture trajectories 1. First row: Air Guitar; Second row: Archery; Third row: Baseball; Fourth row: Boxing 78
4.7 Microsoft Kinect Gesture trajectories 2. First row: Celebration; Second row: Chicken; Third row: Clapping; Fourth row: Crying 79
4.8 Microsoft Kinect Gesture trajectories 3. First row: Driving; Second row: Elephant; Third row: Football; Fourth row: Heart Attack 80

4.9 Microsoft Kinect Gesture trajectories 4. First row: Laughing; Second row: Monkey; Third row: Skip Rope; Fourth row: Sleeping 81
4.10 Microsoft Kinect Gesture trajectories 5. First row: Swimming; Second row: Titanic; Third row: Zombie 82
4.11 Gesture Recognition: Using all postures - Microsoft Kinect Dataset 84
4.12 Gesture Recognition: Using all postures - Nintendo PowerGlove Dataset . . . 85
4.13 Gesture Recognition: Using posture transitions - Microsoft Kinect Dataset . 88
4.14 Gesture Recognition: Using posture transitions - Nintendo PowerGlove Dataset 89
4.15 Microsoft Kinect gesture recognition comparison chart 94
4.16 Nintendo PowerGlove gesture recognition comparison chart 95

List of Abbreviations

Abbreviation	Description
2D	Two Dimensional
3D	Three Dimensional
ANN	Artificial Neural Network
SOM	Self-Organizing Map
SOFM	Self-Organizing Feature Map
SSOM	Spherical Self-Organizing Map
PCA	Principal Component Analysis
MDS	Multi-Dimensional Scaling
VQ	Vector Quantization
SBD	Shot Boundary Detection
BMU	Best Matching Unit
CCD	Compact Composite Descriptor
CEDD	Color and Edge Directivity Descriptor
FCTH	Fuzzy Color and Texture Histogram
BTDH	Brightness and Texture Directionality Histogram
SpCD	Spatial Color Distribution Descriptor
SGONG	Self-Growing and Self-Organized Neural Gas network
SVM	Support Vector Machine
HMM	Hidden Markov Model
DP	Dynamic Programming
HSOM	Hierarchical Self-Organizing Map

Chapter 1

Introduction

1.1 Introduction to Gestures and Gesture Recognition

As computer hardware becomes more and more powerful, their processing power makes it possible to create new ways for humans to interact with computers, and to be able to perform tasks more intuitively, creatively and productively. We need to no longer be constrained by what the keyboard and mouse allow us to do. Nowadays, efficient human computer interaction frameworks carry a big role in our daily lives. Gestures could be considered as a tool that facilitates the exchange of information between humans and computers. Being able to identify gestures through gesture recognition approaches is a challenge that researchers have to handle. In simple words, *gesture recognition* is the process by which the *gestures* made by the user are *recognized* by the receiver.

In this thesis, we attempt to develop a gesture recognition framework, which would be able to accurately classify complex body and hand gestures. This thesis also investigates new Self-Organizing architectures that would allow us to express these gestures as a set of unified features.

According to [8], "Gestures are expressive, meaningful body motions involving physical movements of the fingers, hands, arms, head, face, or body with the intent of: 1) conveying

meaningful information or 2) interacting with the environment. Human gestures simply incorporate a small subspace of possible human motion." Interestingly, a gesture may also be perceived by the environment as a compression technique for the information to be transmitted elsewhere and subsequently reconstructed by the receiver [8]. Some of the applications of gesture recognition are [8]:

- developing aids for the hearing impaired;
- enabling very young children to interact with computers;
- designing techniques for forensic identification;
- recognizing sign language;
- medically monitoring patients' emotional states or stress levels;
- lie detection
- navigating and/or manipulating in virtual environments;
- communicating in video conferencing;
- distance learning/tele-teaching assistance;
- monitoring automobile drivers' alertness/drowsiness levels, etc.

Figure 1.1 shows Kendon's Gesture Continuum, which was an early step to understanding and describing the nature of human gesture [9]. Kendon defined five types of gestures as follows:

1. Gesticulation represents the unprompted actions of the hands and arms which accompany speech.
2. Language-like gestures are movements that are part of speech, and replace spoken words or phrases.

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

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

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

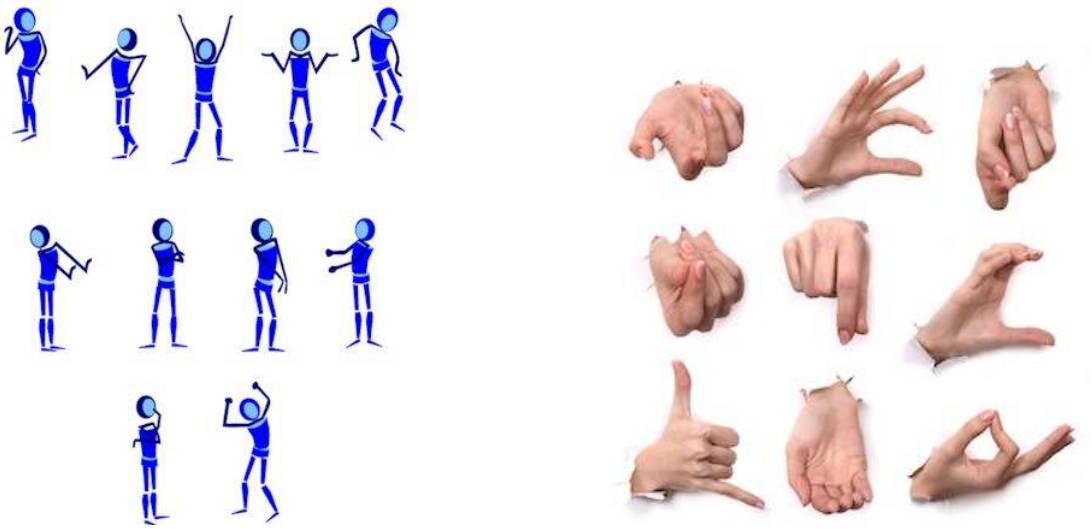
Figure 1.1: Kendon's Gesture Continuum

As one moves to the right in Figure 1.1, the accompaniment of speech with gesture is reduced, impulsiveness decreases, language properties increase, and social guidelines increase.

In computer interfaces, two types of gestures are distinguished [10] - online and offline gestures. Online gestures involve direct manipulation of digital content, for e.g. The use of hands and movements to scale or rotate a tangible object. Offline gestures are those gestures that are processed after the user's interaction with an object has been acquired, for e.g. the activation of a menu, or the recognition of an event.

Breaking gestures down into their component parts, or segmenting them, is required to correctly identify natural and uninterrupted gestures. Segmenting a gesture automatically is very difficult, and this step is often skipped in gesture recognition systems. As a workaround, a start position in time and/or space is required. Often, *postures* are used as a unit to gestures. A posture here is referred to a body (or body part) position at a given point in time.

It is similar to breaking a continuous signal (gesture) into a set of discrete points (postures) describing the nature of the signal. Figure 1.2 provides some examples of human body and hand postures. More general that this, we define a posture in this work to be the state (configuration of a set of variables describing the motion, position or visual properties) of the body at a particular instance in time during a gesture, where a gesture is a temporal sequence of postures whose order conveys specific meaning.



(a) Body postures (b) Hand postures

Figure 1.2: Examples of human postures

In order to capture the gesture fully, one has to be aware of the gesturer's position, movement and configuration. Usually this is done with a data suit, sensors, or special gloves being worn by the gesturer, or using a computer vision system to capture and describe configurations based on image properties.

Gestures tend to vary from person to person, so it is critical to capture the invariant

gesture properties to keep a system universal. When it comes to static gestures, the recognition system can be accomplished through a process illustrated in Figure 1.3, by a way of pattern recognition methods. In general, such techniques for pattern recognition fall into two categories: rule based or exemplar based systems; the latter of which implies the use of a learning mechanism that builds models of gestures from real example data; while the former imposes a model based on domain knowledge. This thesis investigates the exemplar based approaches to tackle gesture recognition.

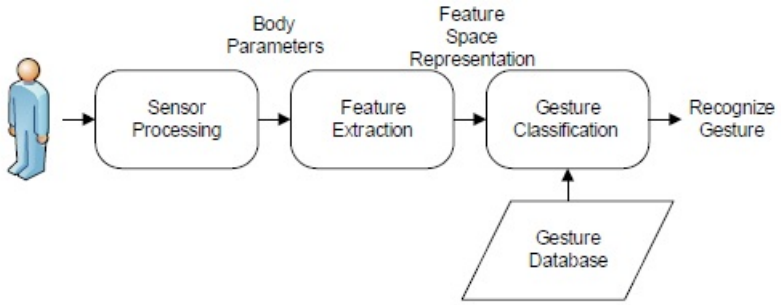


Figure 1.3: Pattern Recognition System (ideal for static gesture recognition)

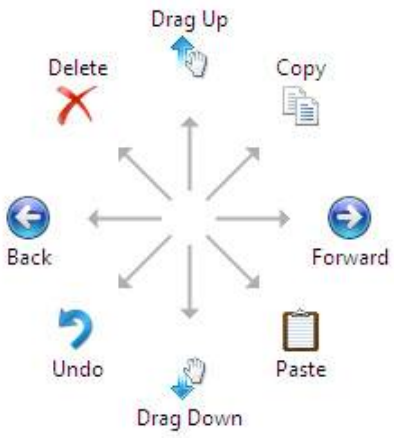
Exemplar based approaches can be further subdivided into supervised (learning with a teacher) and unsupervised (learning without a teacher). In supervised approaches, we have a training set of examples that are labeled, i.e. it is known upfront which example belongs to which class. In unsupervised techniques, some learning takes place without such labels, in which characteristic patterns in the data are sought independently of any class membership. In this work, a hybrid mechanism is proposed that leverages an unsupervised phase of learning known as self organization to extract key characteristics of gestures, in addition to a supervised phase to formalize the description of gestures and aid in their detection.

1.1.1 Recent developments in acquisition technology

Gestures are a very powerful tool for interacting with environment, therefore the discussion about gestures cannot be complete without mentioning recent technologies in gesture acquisition and recognition. Touch interfaces have become common in the present with the introduction of touch displays in mobile phones, tablets, computers, monitors, etc. User can interact with the device via direct touch of the screen, interacting with the data directly rather than using buttons or keyboards as it was in the past. Some examples include Flicks for Microsoft Windows[1](Fig.1.4a) and finger gestures in Apple MacBook laptops[2](Fig.1.4b). The first one involves gestures that can be made with a tablet pen to quickly navigate or perform shortcuts. The same tasks can be done in the second example but using fingers gestures traced on a track pad.

The technology has gone further by introducing gaming consoles such as Microsoft Kinect, Nintendo Wii and Sony Playstation Move (Fig.1.5), where the user plays games by just using his or her body and various gestures understood by the console. In this case the body gestures serve as a language between the computer and human and facilitate the interaction between them for a rich and full experience for the user. Visualization tools have been created based on this technology. One example of such visualization tool is *immersive virtual reality* which was created due to recent advances in research and can provide a rich visualization and interactive modeling and analysis tool [11]. Augmented reality combines real world objects with virtual reality. By overlaying additional information on a real scene, a person, for example, can walk through the streets of New York, without being there physically. It is very interesting to see what the development of this technology will bring us in the future.

[1]http://windows.microsoft.com/en-US/windows7/What-are-flicks
[2]http://www.apple.com/osx/what-is/gestures.html



(a) Windows Flicks

(b) MacBook finger gestures

Figure 1.4: Recent touch technology

1.1.2 Traditional approaches to gesture recognition

Before gestures are analysed they are normally divided into states - it may be a set of postures, or sequence of postures. All these states are typically used as a feature space, on top of which a recognition model is built. Traditionally, techniques such as Hidden Markov Model(HMM), Recurrent Neural Networks(RNN), Dynamic Time Warping(DTW) and Metafeatures have been used for gesture recognition:

1. **HMM**. A HMM [12] is a variant of a finite state machine. However, unlike finite state machines, they are not deterministic but rather probabilistic. HMM consists of the following parts:

Figure 1.5: Examples of body motion sensors and controllers. From top to bottom: Sony Playstation Move, Nintendo Wii, Microsoft Kinect

- A set of states $S = \{1, ..., n\}$.
- A set of output symbols Y.
- Two special subsets of S, the starting states and the ending states. Typically the HMM starts in state 1 and end in state n (although multiple start and end points are also possible).
- A set of allowed transitions T between states. T is a subset of SXS, in other words, a transition goes from one state to another. Self-transitions (eg. from state 1 to state 1) are allowed.
- For each transition, from state i to state j, a probability that the transition is take. This probability is usually represented as a_{ij}. For disallowed transitions $a_{ij} = 0$. These are known as *transition probabilities.*
- For each state j, and for each possible output, a probability that a particular output symbol o is observed in that state. This is represented by the function $b_j(o)$, which gives the probability that o is emitted in state j. These are called

the *emission probabilities*.

HMMs can be employed for classification in the following way: given several HMM models, it is possible to determine the model which will produce a given sequence of observations (postures) with the highest probability. Thus, for each class there is a model with the states, transitions and probabilities set appropriately the *Viterbi*[3]algorithm can be used to calculate the model that most probably generated the sequence of observations.

The problem with this approach is that it is hard to choose the correct model, i.e. what are the appropriate states and transitions. Since there is no definitive answer, experts use domain knowledge and trial and error.

2. **RNN**. Another tool that has been used for temporal classification problems is recurrent neural networks [13]. It is a type of neural network (refer to section 2.2 for more information about neural networks), which is modified to allow for temporal classification (Fig.1.6). A *context* layer is added to the structure, which retains information between observations. The previous contents of the hidden layer are passed into the context layer. These are then fed back into the hidden layer in the next time step. To do classification, post-processing of the outputs form the RNN is performed: when a threshold on the output from one of the nodes is observed, a particular class (gesture) has been registered.

   The disadvantage of this approach is that involves many parameters such as the number of units in the hidden layer, the appropriate structure, the learning rate, etc. Also, if we deal with complex gestures involving many variables and states, a large network is required.

[3]An algorithm for calculating the sum of the probabilities over all possible paths

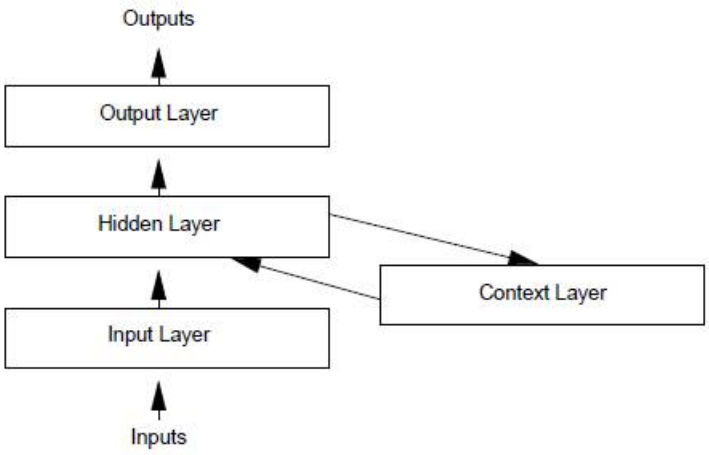


Figure 1.6: Recurrent Neural Network

3. **DTW**. Dynamic time warping [14] is an instance-based learning approach. Instances of postures and posture sequences are warped together to form a template (Fig.1.7) of the same length and size. Unlabeled gesture instances are then compared against those templates based on the distance between the input and the templates, and the unknown gesture is attributed to the nearest template. Although this is a stateless approach, it is difficult to use this approach when dealing with multivariate data, and complex gestures are recognized.

4. **Metafeatures**. Instead of looking at a sequence of postures, in this approach, we look for a set of features in sub-events from the training data, which are either important, typical or distinctive [15]. A learner or classifier is then applied to look for these features in all the gesture instances. The disadvantage of this method is that it involves domain knowledge and different rules need to be applied for the learner to correctly classify unknown gestures. It is domain dependant and rule-based approach, where rules are build based on the nature of the data.

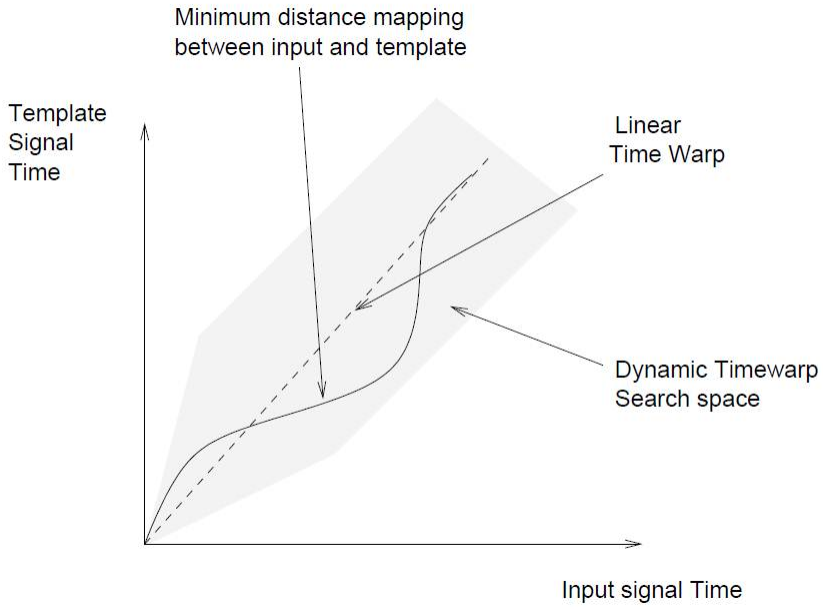


Figure 1.7: Dynamic Time Warping

1.1.3 Domain independent approach to gesture recognition

To overcome the problems and limitations of the approaches described earlier, some recent works have been done using Self-Organizing Maps(SOM). The purpose of SOM is to decompose temporal input samples into commonly occurring states and build a map to know how the states relate to each other. The map (Fig.2.4) works by considering individual vectors from the input space (assume a vector represents a set of input variables describing the configuration of a posture). Training samples are mapped to their closest representation on the lattice: the winning node. This node and surrounding nodes on the lattice are proportionally tuned toward that input vector. Over time, by presenting many different input vectors, all nodes on the lattice tune themselves to samples that occur frequently in the input space. A natural *organization* results which sees similar postures fall onto similar spatial locations in

the map.

Ultimately, we can think of the mapping as a topology preserved projection of postures, into a low dimensional representation. If we map temporal gesture data onto the lattice, the smooth changes happening in high dimensional space should move relatively smoothly also in low-dimensional lattice. This allows us to model some kind of transitions or path that is traced on the map.

The idea behind SOM is that we do not want to use domain knowledge but instead we want to automatically learn or partition the input space into a set of commonly occurring states (postures), and this is achieved in an unsupervised manner. This thesis exploits the properties of the SOM to model gesture trajectories.

1.1.4 Introduction to gesture data sets used in this thesis

For the experiments involved in this thesis it was used two different datasets involving full body gestures and hand gestures. The first data set that was used is the *Australian Sign Language*. The reason why this data set was chosen is because it is an established data set, which is used for sign language. This data set is used as base evaluation in this thesis, since there have been some prior work done on it involving gesture recognition. This thesis also uses a much more newer and complex data set obtained in house with the help of Microsoft Kinect IR camera. Arbitrary gestures classes are used. This data set was used an an exploratory phase for the framework developed in this work. These two datasets were used separately but with a similar experimental setup. The description of the datasets is given below:

1. *Australian Sign Language signs* [16]. This dataset consists of 20 different signs, which were collected from five signers with a total of 6650 samples. The source of data is the raw measurements from a Nintendo PowerGlove, see Fig. 1.8. It was interfaced through a PowerGlove Serial Interface to a Silicon Graphics 4D/35G workstation. Po-

sition information is calculated on the basis of ultrasound emissions from emitters the glove to a 3-microphone "L-Bar" sits atop a monitor. There are two emitters on the glove; and three receivers. This allows the calculation of four pieces of information: x(left/right), y(up/down), z(backward/forward), and roll(is the palm pointing up or down?). x, y and z are measured with 8 bit accuracy. These x, y, z positions are relative to a calibration point which is when the palm is resting on the seated signer's thigh. Roll is four bits. Finger bend is generated by conductive bend sensors on the first four fingers. Values vary between zero(straight) and three(fully bent). Accuracy is 2 bits. The gloves automatically apply a hysteresis filter on these bend sensors. Table 1.1 describes the variables from the dataset.

Figure 1.8: Nintendo PowerGlove

In total there are 70 samples per gesture class and each feature vector contains 8 entries depicted in Table 1.1. Since each gesture takes a different amount of time to be performed, the gestures' length is variable but usually ranges between 40 and 60 samples. Table 1.2 shows the gestures involved in the experiments.

2. *Microsoft Kinect full body gesture database.* This dataset was collected using a sensor equipment in the Microsoft Kinect camera, see Fig. 1.5. A virtual version of the game Charades was used to collect full body gesture data. Nineteen gestures were selected randomly out of a classic commercial version of Charades. Table 1.3 alphabetically

Variable	Data Type	Description
x	continuous	x position between -1 and 1. Units are in meters
y	continuous	y position between -1 and 1. Units are in meters
z	continuous	z position between -1 and 1. Units are NOT meters. This space should not be treated as linear, although it is safe to treat it as monotonically increasing
roll	continuous	roll with 0 meaning 'palm down', rotating clockwise through to a maximum of 1 (not included), which is also 'palm down'
thumb	continuous	thumb bend; has a value of 0 (straight) to 1 (fully bent)
fore	continuous	forefinger bend; has a value of 0 (straight) to 1 (fully bent)
index	continuous	index finger bend; has a value of 0 (straight) to 1 (fully bent)
ring	continuous	ring finger bend; has a value of 0 (straight) to 1 (fully bent)

Table 1.1: Australian Sign Language Variable description

Alive	All	Answer
Boy	Building	Buy
Change	Cold	Come
Computer	Cost	Crazy
Danger	Deaf	Different
Draw	Drink	Eat
Exit	Forget	Visual

Table 1.2: Australian Sign Language gesture data

lists the 19 different gestures that were used in the database. It is easy to see how these gestures are very open to interpretation. Of the 19 gestures (classes), 50 full samples of each gesture were sampled.

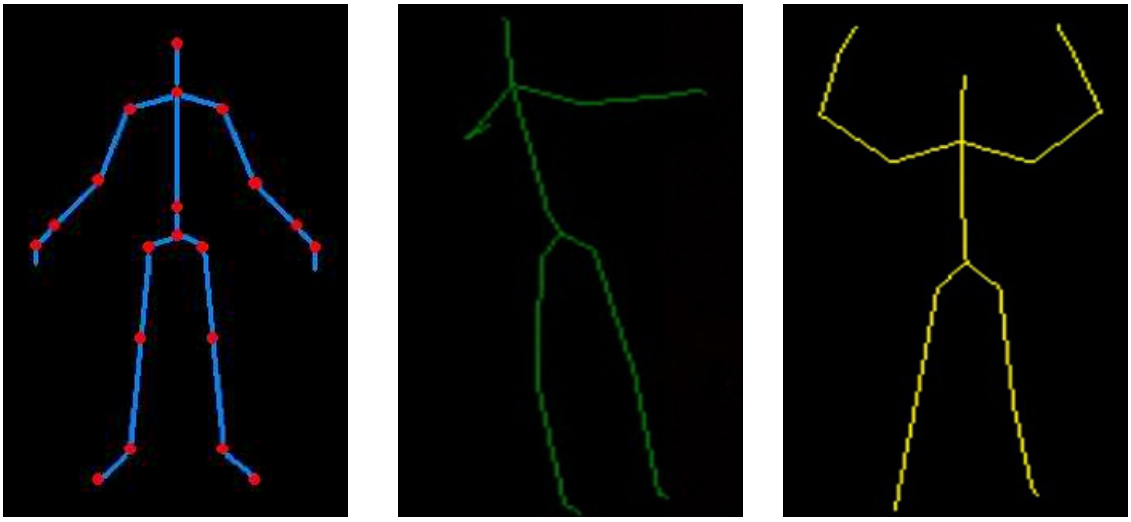
The Kinect primarily samples user 'gesture' information from the IR depth camera. The data coming from the camera is oriented relative to its distance from the Kinect. This becomes problematic when searching for the solution to universal truths in gestures. A normalization technique was developed that converted all depth and position data into vectors relative to a single joint presumed most neutral. In this case the torso was considered as the neutral position of the body. Figure 1.9a shows the skeleton model with the points (body parts) used in the dataset and Table 1.4 shows all the body parts being tracked. The result includes positive and negative x, y, and z-axis values. Figure 1.10 demonstrates a sample feature vector obtained from the Kinect sensors. The feature vector consists of 60 features (three displacement vectors -x,y,z multiplied by 20 body points). The average temporal length of each gesture in the database is 200-300 frames.

1.1.5 Mathematical definition of a gesture

We define a gesture x_i, where $x_i = \{t_0, t_1, ..., t_j, ...t_n\}$ is an ordered time series of multivariate input vectors t_j, where $t_j = \{a_1, a_2, ..., a_M\}$ a set of M real valued attributes describing the raw sensor data from input gesture device.

Air Guitar	Clapping	Laughing
Archery	Crying	Monkey
Baseball	Driving	Skip Rope
Boxing	Elephant	Sleeping
Celebration	Football	Swimming
Chicken	Heart Attack	Titanic
		Zombie

Table 1.3: Microsoft Kinect Full body gesture data



(a) Showing body points being tracked

(b) "Zombie" gesture instance

(c) "Celebration" gesture instance

Figure 1.9: Microsoft Kinect skeleton

In the case of the ASL data set, $t_j = \{x_j, y_j, z_j, roll_j, thumb_j, fore_j, index_j, ring_j\}$ representing the state of the input sensors in the PowerGlove at time j.

In the Kinect data set, $t_j = \{p_1, p_2, ..., p_k, ..., p_{60}\}$, where $p_k = [x_k, y_k, z_k]$ is the x, y, z position of the k^{th} joint in the skeletal data detected by the Microsoft Kinect SDK.

Let $X = \{(x_1, L_1), (x_2, L_2), ..., (x_p, L_p), ..., (x_n, L_n)\}$ be the labeled set of samples, where L_p is an element of L - the set of possible labels: $L = \{w_1, w_2, ..., w_n\}$.

In the ASL data set, examples of w_i's are $w_1 =$ "Alive", $w_2 =$ "Boy", ..., $w_{20} =$ "Visual" (See Fig. 1.2b).

In the Kinect data set, $w_1 =$ "Air Guitar", $w_2 =$ "Archery", ..., $w_{19} =$ "Zombie" (Examples of gesture instances are shown in fig.1.9c and 1.9b).

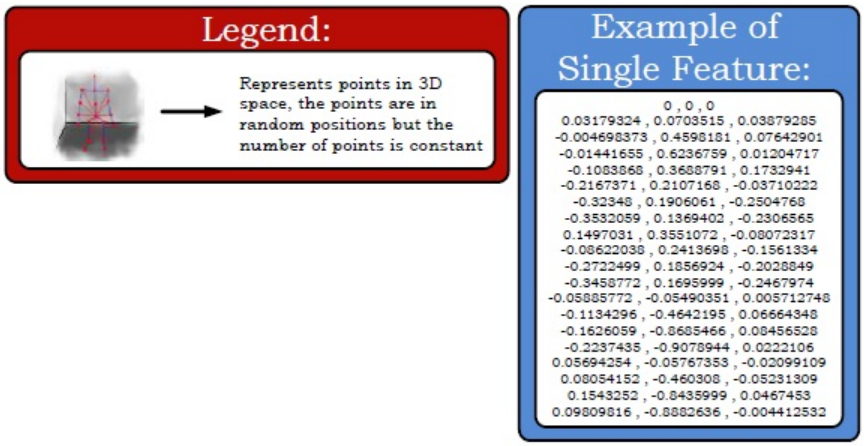


Figure 1.10: Representation of a single feature

1.2 A brief introduction to Supervised and Unsupervised Learning

In an effort to provide some background without delving into the vast landscape of pattern recognition, this section, will highlight the main features of supervised versus unsupervised approaches in a general sense. For a more complete survey, the reader is referred to [17]. In supervised learning, as alluded to earlier, examples relating to known classes or categories are used in "direct" or "teach" system. This typically proceeds by learning a weighted mapping (e.g. linear or nonlinear) transform of the input pattern (x_i) onto a decision space (w), i.e. the class labels referred to in the previous section. In general, the approach is to start with an arbitrary mapping and an example input (training sample), and consider the error of mapping to the desired output. The mapping is then adjusted to reduce this error. By considering many training samples, error is reduced and an appropriate mapping is "learned". Neural networks and support vector machines are examples of supervised learning ([18], [19]).

Torso
Head
Neck
Left Shoulder
Left Elbow
Left Hand
Right Shoulder
Right Elbow
Right Hand
Left Hip
Left Knee
Left Foot
Right Hip
Right Knee
Right Foot

Table 1.4: Microsoft Kinect Full body gesture variables

Unsupervised Learning by contrast, implies no access to class labels, and involves a more subtle goal: to essentially formulate or discover significant patterns or features in a given set of data, without the guidance of a teacher. The patterns are usually stored as a set of *exemplars* (prototypes): representations of natural groupings of similar data or *clusters*, present within the input space. The piece we have no labels or definitions for are the actual postures. We want to use unsupervised learning to identify such postures. This is a kind of interim mapping between x and w. We go from $x- > p- > w$, where p is a posture space. This will be explained further in chapter 3.

The concept of clustering is invariably tied to an adequate definition of what constitutes a *cluster*. This is a subject of much debate, and often depends on problem domain. Never the less, the concept of a cluster is inherently related to the concept of *similarity* between samples from an input data space. In the majority of cases, the distinct lack of a *priori* information in an unsupervised learning problem warrants the need for a generalized framework. As such, the process of *clustering* can be broadly defined as one that seeks to

group together similar or related entities. Each entity or sample is often taken to be some n-dimensional vector (defined on an input space $\Re^n$): this is represented as a 2D space in Figure 1.11. For instance, an input gesture may be represented as a set of x, y, z positions representing joint locations as in the Kinect data mentioned previously. Generally, the ability of an algorithm to extract natural groupings from the data is linked to a choice of distance metric used to assess similarity (for e.g., a standard Euclidean metric might result in the set of colored groupings shown in Fig.1.11, each reflecting a similar type of pattern in the input).

A popular method for identifying groupings in the literature is the k-means method [20], where the groupings are often considered as classes unto themselves. Hybrid approaches may perform such groupings and then associate groups with a label in a later supervised step. This is often employed when we have partially labeled data available for training. In these configurations down to a set of characteristic postures typically encountered in gestures expressed for a given application. Working with this reduced set, we then apply a supervised approach to describe a sequence of these key postures that is relevant for a particular gesture.

Specifically this thesis explores the use of a Spherical Self-Organizing Map (SSOM) for this purpose, which performs unsupervised clustering of the gesture data. The ability to map the gesture data from higher dimension to lower along while preserving the topological association (described in Chapter 2 and 3) between postures belonging to a particular gesture makes the SSOM a suitable tool to explore and model smoothly changing data such as gesture data.

1.3 Thesis Overview

The objective of this thesis is to investigate the potential of SSOM's to facilitate the modeling and recognition of high dimensional temporal information in gesture recognition applications. Alternative Self-Organizing Map (SOM) structures are studied for their ability to represent data. Following is the structure of the remainder of this thesis:

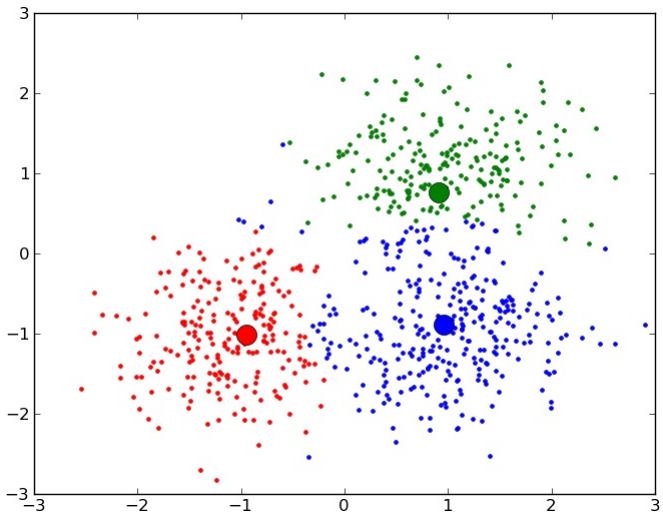


Figure 1.11: The problem of defining a cluster. The clusters are formed based on the coordinates of the data points

- Chapter 2 (Gesture Recognition) discusses self-organization, supervised and unsupervised learning. It brings examples from literature where 2D self-organizing maps are used to tackle gesture recognition. It introduces how trajectories are used for analyzing gesture data.

- Chapter 3 (Trajectory analysis based on video and gesture data using Self-Organizing Maps) gives an introduction into spherical self-organizing map and discusses some benefits over the conventional 2D SOM. An experiment is performed using a 3D SOM to show some of the properties of the Self-Organizing Map to help map smoothly (slowly) varying data such as a video clip containing several scenes. Some conclusions are drawn, which are later used for next chapter, where the main experiments are discussed. Trajectory methods, which are used in this thesis are introduced: using all postures, weighted aggregation of all postures, using posture transitions, weighted aggregation of all posture transitions.

- Chapter 4 (Experiments and Results) explains the experimental setup. A thorough explanation is given on each of the four methods implemented for gesture recognition and results are discussed.

- Chapter 5 (Conclusion) summarizes the results of this thesis's contributions. It gives an evaluation to the experiments and suggests some future work and suggestions.

1.4 Thesis Contributions

The main contributions to this thesis are twofold:

1. Using a Spherical Self-Organizing Map to decompose gestures into a well separated sparse set of postures.
   - Constrained trajectories on the sphere formed from mapping the gesture data onto the spherical lattice gives an ability to analyse the data without worrying about the size of the sphere.
   - SSOM has more resolution comparing to 2D (flat) SOM (no border effect), and it is ideal for reducing high dimensional sequence of data into trajectories on the sphere.

2. Proposal and investigation of four different approaches to model gesture trajectories extracted from the SSOM mapping with application to gesture recognition.

1.4.1 Publications

"Trajectory analysis on Spherical Self-Organizing Maps with application to gesture recognition". Submitted to *WSOM 2012, 9th workshop on Self-Organizing Maps*.

Chapter 2

Self organizing approaches to gesture recognition

2.1 Introduction

The gesture recognition challenge lies largely in mathematical computation and modeling. Systems need to be created that use more complex data sets and models than a speech recognition system, which is a two-dimensional problem (2D), or a handwriting recognition system, which is a three-dimensional problem (3D). Gestures are complex, because they are represented as high dimensional data and the challenge lies in both to recognize and interpret the current gesture, and to understand its meaning by analyzing its context as indicated earlier.

Rule based approach to gesture recognition refers to gesture recognition systems that do not perform learning on the gesture data. But rather, impose a set of rules that must be satisfied in order for a gesture to be recognized. Often such systems cannot generalize well however, and if a gesture is performed poorly by a person, for example, then the system will be unable to recognize it. An example of this type of gesture recognition systems may be seen in modern touch-screen mobile phone or tablets, where a user has to use a specific set of touch gestures with his or her fingers in a certain order. These gestures are recognized

easily because they are programmed and hard coded onto the device itself, and in case if the user performs the gesture poorly or in a wrong way, no interaction between the device and the person will occur. Some examples include Flicks for Microsoft Windows, and finger gestures on MacBook Apple laptops, which were mentioned earlier in the introductory chapter. These kinds of gestures make the interaction between the person and the device more natural, although there is also a limitation in the complexity of the gesture: complex patterns of movement cannot be easily understood.

Exemplar based learning approaches on the other hand, typically have the ability to generalize, unlike rule based approaches, and thus are the directions explored in this work. As mentioned, neural networks represent a common family of such approaches in pattern recognition. In fact, the architecture investigated in this work is based on a special type of neural network known as a self-organizing map (SOM). In this Chapter, we give a foundation and an overview of neural networks with focus on self-organization and unsupervised learning, and review some works from the literature that have used such principles in gesture recognition.

2.2 Neural Networks

A neural network in the context of this thesis represents a data modeling tool that can capture and represent complex input/output relationships [1]. It is thought to be inspired by the way biological nervous systems, such as the brain, process information. The network is built from a large number of highly interconnected processing elements (neurons) all working in parallel, see Figure 2.1. In contrast to conventional computers, where problems are being solved by a specific algorithm or a step by step instruction, Neural networks learn by example and by the data that is being provided to the network. A neural network is usually configured for a specific application, such as pattern recognition or data classification, through a learning process.

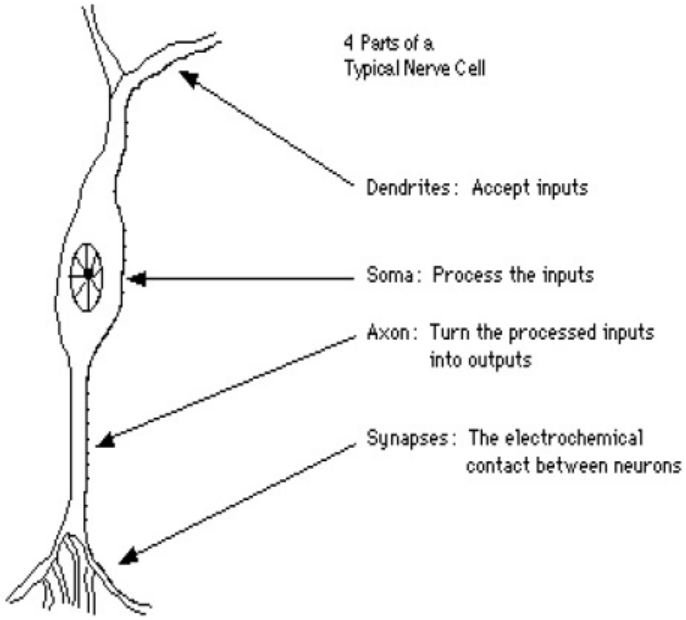


Figure 2.1: A Simple Neuron [1]

In general a Neural Network consists of three types of layers which have input neurons, hidden neurons and output neurons respectively. Figure 2.2 shows the basic principle of Neural Networks.

In this case $X_1, X_2, ..., X_N$ represents the inputs which might be from other neurons; Y represents the outputs which might be inputs to other neurons or the final outputs, and $W_1, W_2, ..., W_N$ representing the strength of the connections between the inputs and the neuron. Finally, the neuron collects all the adaptive inputs, and uses the activation function to generate the output. The activation function could be non-linear function or Gaussian. The formula is shown below:

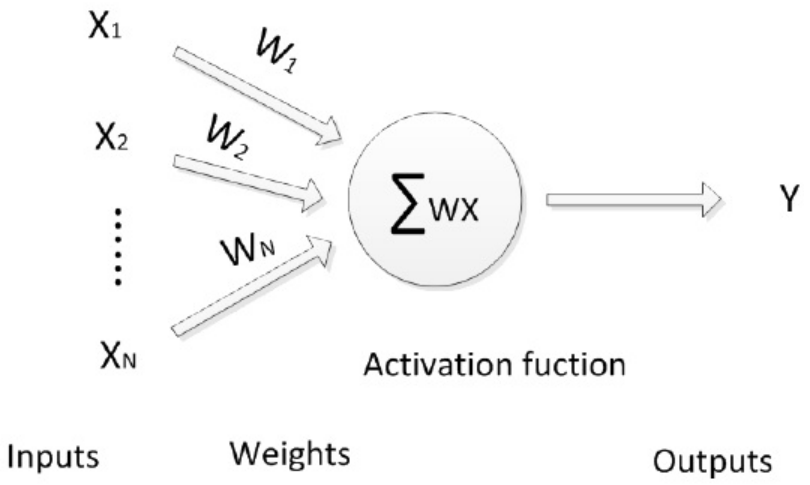


Figure 2.2: Example of a basic Neural Network

$$y = f(\sum_{i=0}^{n} W_i X_i) \tag{2.1}$$

where y is the output from the neuron, n is the number of inputs, W_i is the weight of the input, X_i is the input to the neuron, and f is the activation function, e.g. for sigmoid, $f(x) = \frac{1}{1+e^{-x}}$.

The majority of neural networks learn in a supervised manner, i.e. the actual output of a neural network is compared to the desired output. The weights learned by the network, which initially are random, are adjusted so that with the next cycles or iterations will closely resemble the desired output. In general, this learning method tries to minimize the error of all the processing elements by constantly modifying the input weights until they reach some acceptable accuracy. The data used to train a neural network has to be numeric, therefore often the raw data has to be converted from its environment (i.e. extracting color features

from an image). It is important to emphasize how the training of the network is important. After a successful training, a supervised network should be tested on an unknown data to see how it performs. This step is to ensure that the network has not just simply memorized a given set of data but has also learned the general patterns in the data.

Unsupervised learning is another category of a neural network learning process. This method is used by the SOM, also commonly known as **Kohonen** network [21]. These networks use no external influences (e.g. guiding "error" signals) to adjust their weights. They use their input data to look for patterns or trends and make according adaptations. The network has all the necessary information to organize itself. As such competition between the processing elements in this type of network is used as a the basis for learning. The next section will introduce the topic of Self-Organization and discuss the *Kohonen's Self-Organizing Feature Map (SOFM).*

2.3 Self-Organization

2.3.1 Principles of Self-Organization

Unsupervised Learning and Self-Organization are inherently related. In general, self organizing systems are typified by the union of *local interactions* and *competition over some limited resource.* In his book [22] identifies four major principles of self-organization:

1. **Synaptic Self-Amplification and Competition:** This principle is expressed through Hebb's postulate of learning [23]. This states that, when two neurons are within a significant proximity enabling one to excite another, and furthermore, do so persistently, some form of physiological/metabolic growth process results. A synaptic path evolves and strengthens between the two neurons, so that future associations occur much more readily. This action of strengthening the association between two nodes, functions as a correlation between their two states. These properties combined, have led to the

modified Hebbian *adaptive* rule, as proposed by Kohonen [24]:

$$\Delta w_{k*} = \alpha\varphi(k*, x_i)[x_i - w_{k*}] \tag{2.2}$$

where w_{k*} is the synaptic weight vector of the *winning* neuron $k*$, α is the learning rate, and some scalar response $\varphi(.)$ to the firing neuron $k*$ (*activation*).

The activation function $\varphi(.)$ is the result of the second principle (synaptic competition). Neurons generally compete to see which is most representative of a given input pattern presented to the network. Some form of discriminative function oversees this process (for example: choosing a neuron as a *winner* if its synaptic vector minimizes Euclidean distance over the set of all neurons). This process is often termed *Competitive Learning*.

2. **Co-operation:** Hebb's postulate is also suggestive of the lateral or associative aspect to the way knowledge is then captured in the network: i.e. not just through the *winner*, but also through nearby neurons in the output layer. Often the strength of connection is not considered, but rather a simple link function as an indicator of which other nodes in the network will more readily be associated with a a winning node. Thus a local neighbourhood is defined and it is through this that knowledge may be imparted. Local adaptation usually follows the simple Kohonen update rule 2.2, whereby a portion of information is learnt by the winning node, with neighbouring nodes extracting lesser portions from the same input.

3. **Knowledge through Redundancy** When exposed, any order or structure inherent within a series of activation patterns represents redundant information that is ultimately encoded by the network as knowledge. In other words, the network will evolve such that similar patterns will be captured and encoded by similar output nodes, whilst neighbouring nodes organize themselves around these dominant redundancies, each in turn focusing on and encoding lesser redundant pattern across the input space.

2.3.2 The Kohonen Self-Organizing Feature Map (SOFM)

Architectures such as Kohonen's *Self-Organizing Feature Map* (SOFM) [24] represent one of the most fundamental realizations of the principles outlined in Section 2.3.1, and as such have been the foundation for much neural network based research in data mining.

In the basic SOM, shown in Figure 2.4 (top), samples (vectors from an input space defined on $\Re^d$), are essentially mapped onto a lower dimensional grid or lattice (typically 2D or 3D). Nodes on the grid (neurons) act as memory elements: storing prototype vectors that ultimately describe commonly occurring vector patterns from the input space. The mapping takes place such that nodes nearby to one another on the lattice, map to patterns that are nearby one another in the input space.

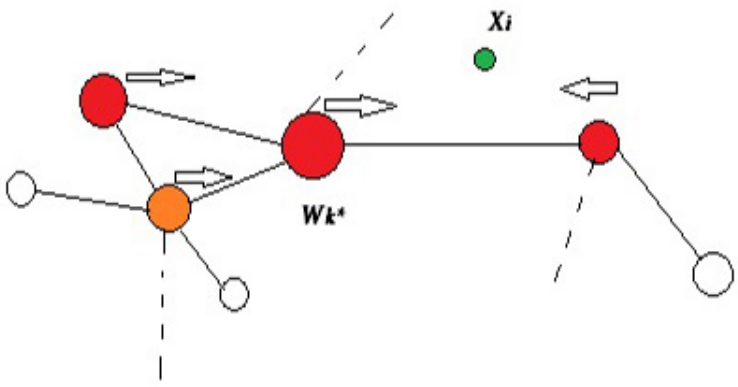


Figure 2.3: Self-Amplification, Competition and Co-operation in learning: w_{k*} represents the synaptic vector of the winning neuron (competition), which adapts (self amplifies) toward the input thereby strengthening its correlation with future inputs in this region. At the same time, associative memory is imparted to neighbouring neurons (co-operation): related to the winner through some defined or inferred topology

The lattice on which a SOM is defined, thus serves as a set of *associative connections* linking prototypes to one another, through which information is shared laterally throughout learning. This is a basic expression of the principle depicted in Figure 2.3. The SOM

algorithm is presented as follows:

1. Initialize the map lattice as an MxN matrix neuron vectors $W = [\mathbf{w}_{i,j}] : \mathbf{w}_{i,j} \in \Re^d; 0 < i < M-1; 0 < j < N-1$. Choose random values for each vector on the grid (or initialize with random samples selected from X)

2. Randomly select a sample vector $\mathbf{x}_i$ from X, and present to the network

3. Choose a *winning* node $\mathbf{w}_{i*,j*}$ such that: $d(\mathbf{w}_{i,j}, \mathbf{x}_i) \forall i \neq i^* j \neq j^*$

4. Update the neurons on the lattice according to the Kohonen learning rule:

$$\mathbf{w'}_{i,j} + \alpha(t) H(r_{i*,j*}, r_{i,j}, \sigma(t))[\mathbf{x}_i - \mathbf{w}_k *] \tag{2.3}$$

$$H(r_{i*,j*}, r_{i,j}, \sigma(t)) = \exp(\frac{-\|r_{i*,j*} - r_{i,j}\|}{2\sigma(t)} \tag{2.4}$$

5. Update $\alpha(t)$ and $\sigma(t)$: which are relaxed over time

6. Repeat step 2, until all $\mathbf{w}_{i,j}$ have converged

In the above algorithm, $r_{i,j}$ denotes the position vector of the node at (i, j) on the map lattice, α represents a learning rate which decays from a small initial value (say 0.1) allowing small portions of information from the input vector to be learnt by the neurons in the lattice. The neighbourhood function, in this case a Gaussian neighbourhood Eq.2.4 is depicted in Figure 2.4 (bottom), and is adaptively controlled by a decaying radius $\sigma(t)$ (initially larger than that of the map: e.g. sqrt$(M^2 + N^2)$). This allows information from the input vector to be further modulated and shared across the map relative to the winner $(w_{i*,j*})$. With time, as the radius over which information is shared reduces, the map switches from a *locating* phase where nodes reorganize themselves over the input space, to more of a *tuning* phase where only the winners are adjusted.

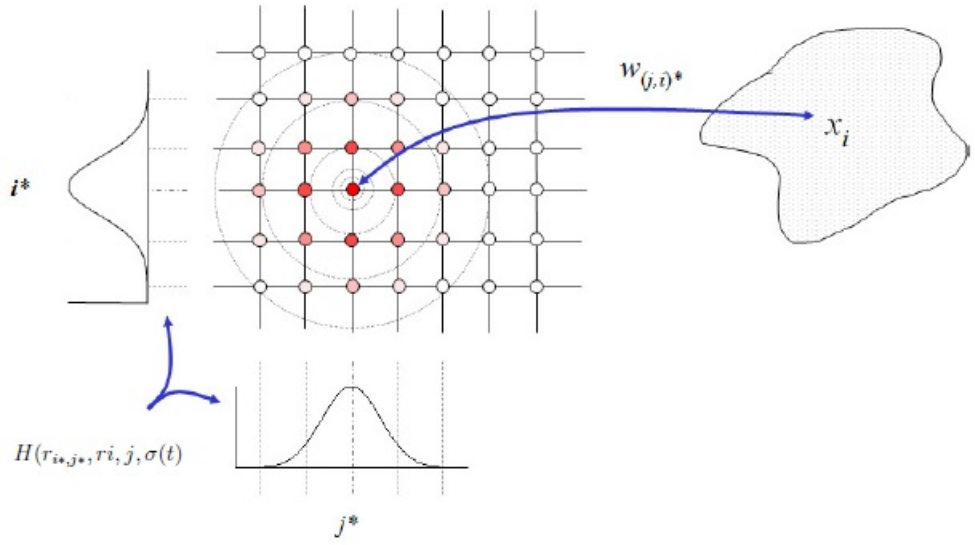


Figure 2.4: Mapping samples from an input space onto a SOM lattice of prototypes; showing the winning node (best matching unit: BMU of the current input x_i (top), Gaussian neighbourhood function (bottom)

In general, it is this feature that leads to a SOM's innate ability to infer an *ordered* or *topologically preserved* mapping of the underlying data space. Associations between nodes are advantageous as they help guide the evolution of such networks, and may also assist in formulating post-processing strategies or for extracting higher-level properties of any clusters discovered (e.g. inter-cluster relationships).

Now that SOM has been introduced, this thesis will continue with a section related to gesture recognition using SOMs, which inspired to investigate the way gesture data maps onto a SOM's lattice. Because of its temporal nature, gestures are ideal for studying *trajectories* (by *trajectories* we refer to the spatio-temporal trace that the gesture data maps on the SOFM.

2.4 Gesture recognition using Self-Organizing Maps and Trajectories

The use of SOM in the area of gesture recognition has been relatively recent. Some methods which will be discussed shortly in this section have used SOM in various ways to divide the sample data into clusters of phases, which are further processed with the help of other tools and techniques. The use of *trajectories* in these works has been limited used just as a transitional step towards gesture recognition, whereas this thesis focuses on the use of trajectories as a feature itself, from which one can perform the recognition process.

The first work discussed is called *automatic learning of gesture recognition model using SOM and SVM* [2] by M. Oshita and T. Matsunaga. The authors proposed a method in which they first use SOM to process the gesture data and then apply a Support Vector Machine (SVM) to partition the feature space into regions belonging to separate classes. They emphasize that the gesture data, in this case they use two Nintendo Wii Remote controller with three-dimensional acceleration sensors as input, is multi-dimensional and time-varying. Their approach is interesting because they divide each gesture into short phases and then apply a pattern recognition technique for multi-dimensional data to recognize each phase.

The authors mention that the main issue lies in the ability to deal with multi-dimensional and time-varying input data. They bring as an example the Hidden Markov Model (HMM) [2], which is a popular method for recognizing time-varying data. It simply represents a recognition model as a network of nodes that produce some symbols and transition probabilities between nodes. The problem with this approach is that it handles input data as a series of discrete symbols, which causes difficulty in handling multi-dimensional signals and prevents accurate recognition.

Dynamic Programming (DP) is another method that authors mention in their work. In

this technique they match the trajectory from the input signals and the sample trajectory from a gesture. In this manner the system can determine whether the gesture has been executed. The disadvantage of this is that the trajectories must be projected onto a low-dimensional feature space. Furthermore, a valid threshold must be specified to measure the similarity of two trajectories.

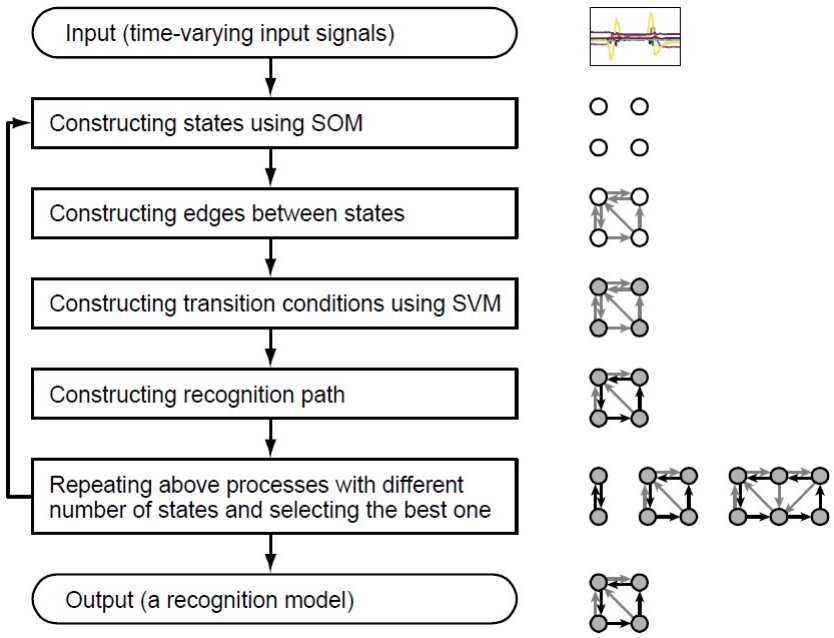


Figure 2.5: Flow chart and data structures for the learning process based on Oshita and Matsunaga method [2]

Figure 2.5 shows the flow chart for the learning process based on Oshita and Matsunaga method. The system is explained here:

1. Input signals are taken from the sample data performed by a user.

2. SOFM is applied to all feature vectors from the sample data and states are constructed

for the state machine. Each of the states contains feature vectors belonging to the corresponding unit.

3. The system determines possible transitions for each state: if state A is followed by a feature vector of state B in the input sample data, an edge from state A to state B is created.

4. SVM is applied to learn the transition conditions for each state. In order to construct an SVM for each state, the sample data belonging to the state and those belonging to adjacent - possible transition - states are used as training data.

5. A state machine is constructed from the above process

The system determines the correct path for a gesture by storing the path from the sample data as a series of states. Then, depending on the transition in the state machine from new incoming data, the gesture following the most states in a particular gesture is chosen.

Another interesting approach to gesture recognition using Self-Organizing Maps was published by A. Shimada and R. Taniguchi in [3]. Their method is based on *Sparse Code of Hierarchical SOM*(HSOM). First, we shall describe how HSOM is different from a regular SOFM. Hierarchical Self-Organizing Map [25] is a two layer SOM network, where the lower layer has a connection with an input layer. In this case, the second layer receives an input vector from the first layer directly.

The method proposed here uses the property of HSOM to first learn postures (which are the minimum unit of a gesture) in the first layer, and then learn short gestures consisting of some time-series postures in the second layer. Authors argue that the time length of a human gesture is not always the same even if same gestures are compared. They highlight that the key issue in their method is to absorb the time variant appropriately in order to make clusters which include the same gesture class.

The term *Sparse Code* is used by the authors to represent the activated pattern of neurons in the first layer of the SOM. The first layer of the SOM learns human posture information represented as a vector I^1 and consisting of the positional information of the head, hands, feet, etc. The output of the SOFM from first layer denoted as O^1 is then used as input to the second layer. Here a posture is shown as $I^1(t)$, where t simply represents the time length of the gesture and $c(t)$ is the winning node or neuron of the input $I^1(t)$. In Figure 2.6 all the $c(t)$ for all of the postures which make up a gesture element are shown. In the figure, each circle shows a neuron of the SOM and the bottom part show a gesture element. The grey-coloured neurons in the figure are the winners for the posture sequence $(I(1), I(2), I(3))$.

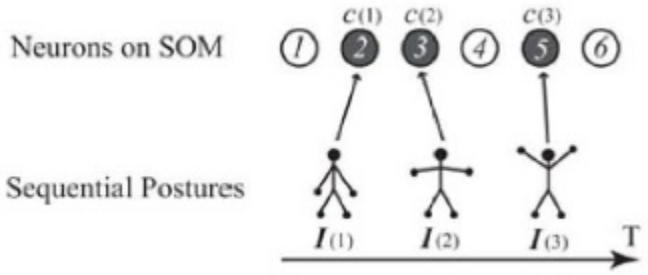


Figure 2.6: Winner neurons for a gesture element [3]

The output regarded as a *sparse code* and consisting of an activated pattern of winner neurons for a gesture element has the possibility to have same neurons selected more than once for different gestures. In this case the sparse code helps the upper layer of the SOM to reach time invariant recognition, because the sparse code is similar between gesture elements which have different time lengths from each other, see Fig. 2.7. As seen in the Figure 2.7, although the time length of the gesture element is longer than the one in Figure 2.6, the sparse code is the same.

A problem may occur in the case where a gesture is forward and backward similar. The SOM will treat this as one gesture, although they may belong to different ones. To solve

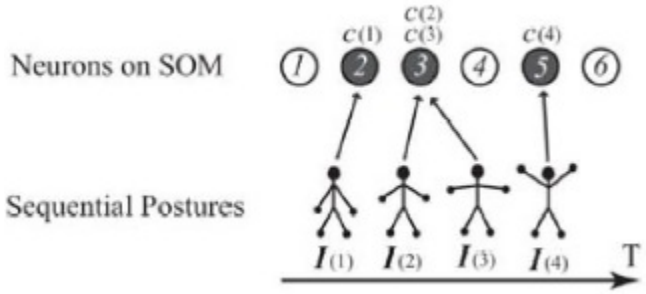


Figure 2.7: Time invariant recognition with sparse code [3]

this problem authors use an ordering mechanism to keep track of the sequence of activated neurons. An example of a backward gesture element is shown in Figure 2.8.

Figure 2.9 shows how the order of the activated neurons is expressed in the color strength. It is clear that the color pattern of the forward gesture is different from the one of the backward gesture.

To eliminate the slight difference between sparse codes for the same gesture category authors apply a Gaussian filter to the code. This can be seen in Figure 2.10, where element 2 and 3 seem to look similar because they have a backward relation. In Shimada's and Taniguchi's approach the sparse codes are distinguished by the second layer's SOM, because transition information is considered in the sparse code.

The interesting part of this technique for gesture recognition is how the authors tackle the problem of time invariance or length invariance if one talks in terms of trajectories that gestures leave on the SOM lattice. The use of multi-layer Self-Organizing network allows them to obtain a more general gesture path on the lattice without worrying about its length.

Another novel approach to *Video-Based Gesture Recognition Using Self-Organizing Feature Maps* was also created by G. Caridakis, C. Pateritas and A. Drosopoulos in [4] and [26]. Their work introduces a probabilistic recognition scheme for hand gestures, where SOMs

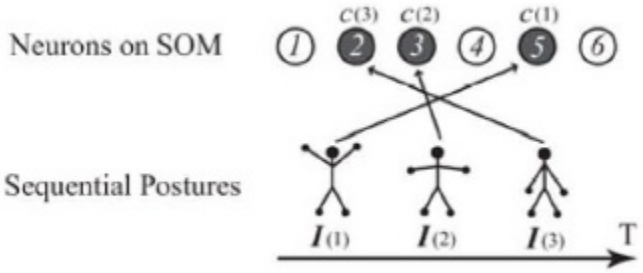


Figure 2.8: Sparse code for a backward gesture element [3]

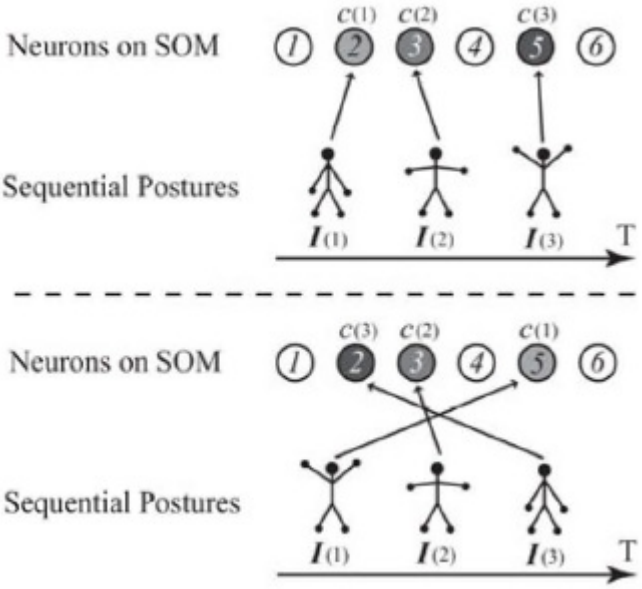


Figure 2.9: Sparse code with time transition [3]

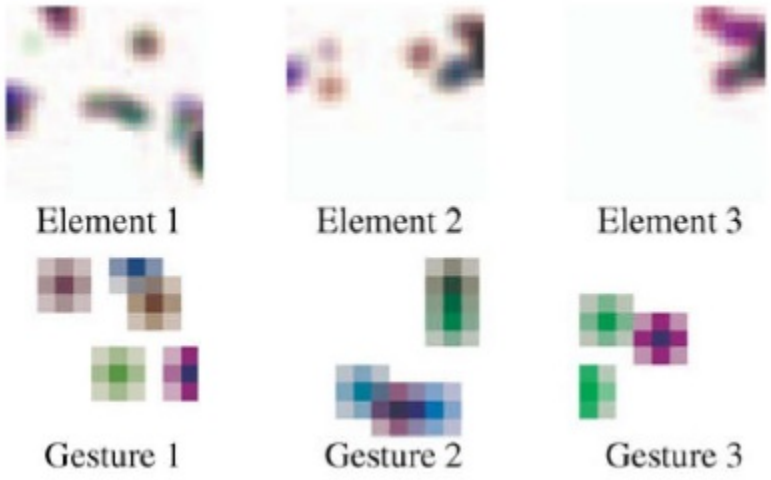


Figure 2.10: Sparse code for gesture elements and gesture [3]

are used to model spatiotemporal information extracted from images. It uses a combination of SOM and Markov models for gesture classification. The classification scheme consists of tracking the transformation of gesture representations from a series of coordinate movements.

First, authors use a near real-time skin detection and tracking module [27] to create moving skin masks, and by tracking their centroids they produce an estimate of user's movements. Each gesture here, is represented by a time series of points, representing the hand's location with respect to the head of the person performing the gesture [4]. The probabilistic model is built from the transformation of a gesture representation from a series of coordinates and movements to a symbolic form. The first transformation is achieved by a Self-Organizing map and is based on the relative position of the hand during a gesture. The argument here is that the SOFM's neighbourhood function provides a distance metric between the gesture symbols used during the classification of an unlabeled gesture.

Another transformation used in this work is based on the optical flow of the gesture, whose main goal is to describe the direction changes in the gesture. These direction changes describe a set of angles of the gesture's trajectory. This set is then used in a quantized form as symbols for the creation of an additional set of Markov models. The system works as follows:

1. Coordinates from all the points from all the gestures are used to train a hexagonal, 2D grid SOM. A gesture G_i can then be transformed from a series of points to a series of map units based on their best matching units (BMUs)

$$T(G_i) = (u_1, u_2, ..., u_l), u_i = BMU(x_i, y_i). \tag{2.5}$$

   Given that u_i is the index of a map unit the function $BMU(x_i, y_i)$ creates S - set of indices of all map units treated as a set of symbols. As explained by the authors in [4], because the u_i value of consequent points of a gesture remains the same - this is because, although continuous hand movement is described by distinct points, consequent points are generally close in the input data space. (Note: although we are talking about 2D SOM in this case, we can once more see the property of the SOM for smoothly varying data and even though we are projecting data from high-dimensional space into lower-dimensional space, we still see a correlation between the similarity distance between the input data and its mapping onto the SOM lattice).

2. Consequent equal values of u_i are replaced with single value which result in the following definition:

$$G_i = N(T(G_i)) = \{u_1, u_2, ..., u_m\} : m \leq l, \forall t \in [2, l] u_t \neq u_{t-1}, \tag{2.6}$$

   where N is a function that removes consecutive equal u_i value and G_i is the transformed gesture instance.

3. The transformation of the gestures with the help of SOM is equivalent to a transfor-

mation of the continuous trail to a sequence of m discrete symbols and which define the finite states for the building of first order Markov chain models. One model for each gesture data is created and the sequence of u_i values is used for the calculation of transition probabilities. These models are used to evaluate new unlabeled gesture. This transformation can be seen in Figure 2.11.

4. An optical flow of each gesture is used to provide a more descriptive representation of each gesture. This descriptive information provides a direction information instead of just the spatial position of gesture points. Authors apply quantization to obtain eight different symbolic values representing direction vectors from consecutive gesture trajectory points. This process is illustrated in Figure 2.12.

   Figures 2.11 and 2.12 depict a set of coordinates that belong to the same cluster - BMU and Quantized Angle for Fig. 2.11 and Fig. 2.12 respectively.

5. The classification of an unlabeled gesture is based on the two sets of Markov models discussed previously. Authors provide the following equation to find the probability of a gesture to belong to category j where MM^{som} represents a model describing a specific category of gesture:

$$P(G_k|MM_j^{som}) = \frac{\sum_{i=1}^{m} S_i^{som}}{m} \tag{2.7}$$

   According to the authors of this work, the above equations averages the values of S_i^{som}, which represent an evaluation factor for each u_i value of the G_k transformed gesture with respect to the MM_j^{som} Markov model, and are calculated as follows:

$$S_i^{som} = max_z(NF_{u_i}^{som}(z)P(z|u_i, MM_j^{som})) \tag{2.8}$$

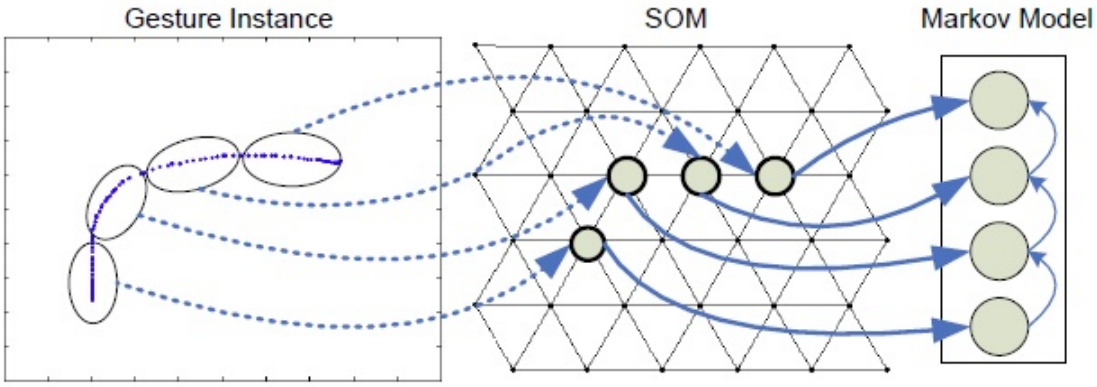


Figure 2.11: Correspondence of gesture trajectory points to their respective BMUs on the SOM. The BMUs constitute the states of the Markov models [4]

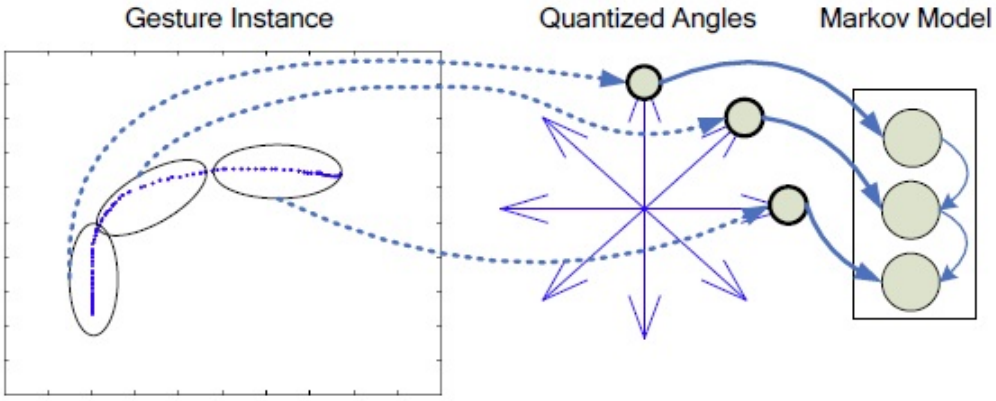


Figure 2.12: Markov model for a gesture's optical flow [4]

$$u_i = argmax_z(S_i^{som}), \tag{2.9}$$

where z is a variable that indexes the units of the trained map and $NF_{u_i}^{som}(z)$ is the distance of the unit z defined by the self-organizing map Gaussian neighbourhood function with the u_i unit as its center. For comparing the length of the unknown gesture with the length of the known gestures authors use a distance metric - *Generalized Median* gesture, which is described by the following equation:

$$\sum_{s_i} L(s_i, m), \forall s_i \in S, \tag{2.10}$$

where S is a set of symbol strings s_i and m is a string that consists of a combination of all or some of the symbols used in the set. $L(,)$ the Levenshtein distance - one of the most widely used string distance metric and calculated as $L_{kj} = L(G_k|M(D_j))$ representing the distance between G_k and the *Generalized media* $M(D_j)$ of each D_j set.

6. Finally the category of the unknown gesture is decided using the MM^{som} set of models expressed in the following form:

$$argmax_j P(G_k|MM_j^{som}) \tag{2.11}$$

Caridakis, Pateritsas and Drosopoulos claim to reach a 93% average accuracy in gesture recognition by using their technique.

2.5 Summary

This chapter has mentioned non-learning gesture recognition approach with an example of modern touch screen devices. It has given an introductory overview of neural networks and supervised and unsupervised learning. Principles of self-organization were discussed along

with the Kohonen's Self-Organizing Map. Finally, some works were presented which related to the topic of gesture recognition with emphasis on using SOMs and trajectories.

Chapter 3

Trajectory analysis of temporal data using Self-Organizing Maps

3.1 Introduction

This chapter will discuss the spatio-temporal nature of SOM. The spherical version of the SOM will be introduced and some properties will be studied initially with the use of video data. Advantages of using a spherical version as opposed to the conventional 2D SOM will be highlighted. Finally, four approaches based on trajectories will be discussed for addressing the problem of gesture recognition.

3.2 Spherical SOM (SSOM)

3.2.1 Topology of the SOFM lattice

The map in the SOM space is a lattice with predefined connections between the nodes. Each node represents a data vector in a one-to-one mapping or a cluster of data vectors in a many-to-one mapping. The topology of the SOM refers to the inter-node connections in the lattice. Based on the topology of the SOM lattice the network may suffer from limitations due to a restricted neighbourhood along the boundaries of the map. A one-dimensional string of connected nodes is the simplest form of the SOM lattice. It may be of either open-ended

topology or closed-loop topology, see Figure 3.1.

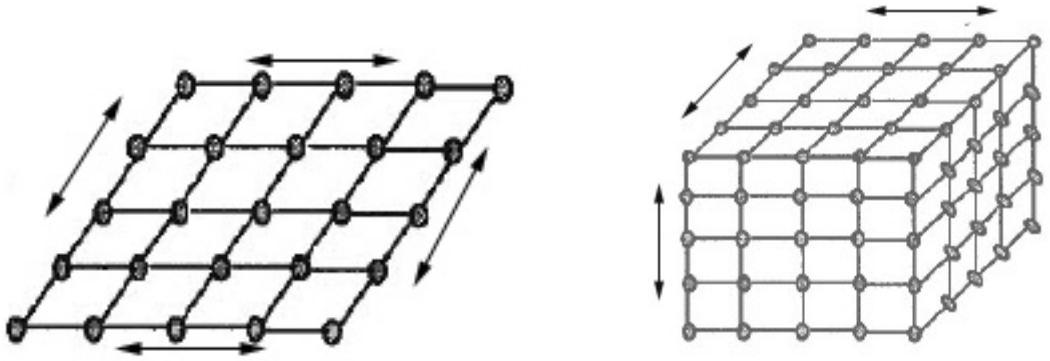
In the latter, the effect of a restricted neighbourhood at the boundaries of the map is minimized as compared to that in the former since every node in the string will have the same number of neighbouring nodes. A two-dimensional SOM lattice is defined as a sheet of connected nodes in Fig. 3.2a, while a cubic structure [28], is the more commonly used three-dimensional SOM lattice, Figure 3.2b. These structures have data association as indicated, as well as in the diagonal direction. A three-dimensional lattice offers a high degree of data correlation or grouping as compared to a one and two-dimensional lattice.

Although a 2D and cubic 3D lattice offer a high degree of association as compared to a 1D string of nodes, due to the additional dimension for lateral interaction among the cluster units, they still have a restricted neighbourhood at the boundaries of map. An alternative 3D lattice is a tessellated sphere with quadrilateral elements shown in Figure 3.3.

Its inter-node structure is equivalent to the latitude-longitude arrangement on the globe. However, its lattice consists of an irregular or non-uniform distribution of nodes, thereby resulting in a quadrilateral facets of varying sizes. There is also a limitation on the number of nodes, because of the possibility of having several nodes overlapping at the poles. Fur-

(a) Open ended topology (b) Closed-loop topology

Figure 3.1: One-dimensional SOM ($\Re^1$)



(a) Two-dimensional SOM (b) Three-dimensional SOM

Figure 3.2: Data association ($\Re^2$ space (3.2a)) and ($\Re^3$ space (3.2b)

thermore, the neighbourhood configuration at the poles is different from that in the rest of the map.

The motivation came to Ritter in [29], who introduced a hyperbolic SOM which employs a predefined grid of spherical topology thereby taking advantage of the symmetry and overall continuity found in the spherical shape. The structure consists of a regular tessellated unit sphere, each node or vertex representing a cluster unit of the SOM. The cluster units are uniformly distributed on the sphere and the faceted structure of the map is defined by triangular elements, Figure 3.4. In contrast to the previous case, the neighbourhood configuration in the lattice of a tessellated sphere with a triangular elements is the same throughout.

3.2.2 Weight adaptation strategy in the spherical SOM

The tessellated lattice in the spherical SOM is created by sub-dividing a regular Icosahedron to a desired level. The structure of the SOM therefore consists of nodes uniformly arranged on a tessellated unit sphere with a uniform triangular elements, each node representing a

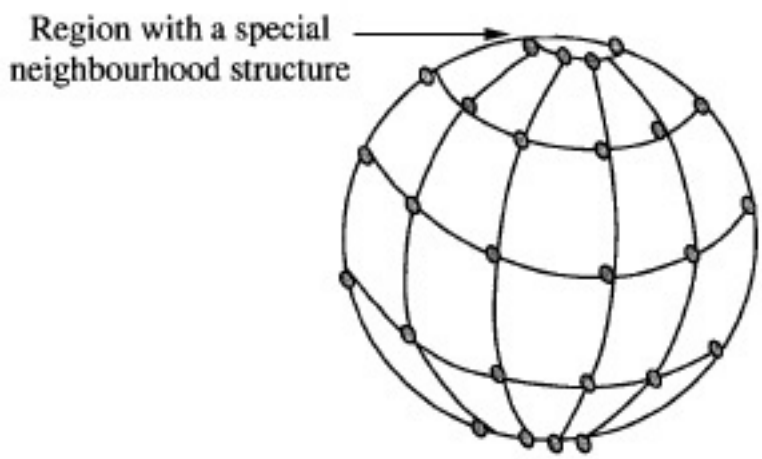


Figure 3.3: Spherical SOM with quadrilateral elements

cluster unit. Each training pattern X (defined in Chapter 2) in the input data space is connected to every cluster unit in the SOM space (ϖ) by a weight vector $w_{i,j,k}$. Every node or cluster unit located at (i, j, k) has a variable neighbourhood ($NE_{i,j,k}$) associated with it and all the nodes that fall within the area defined by $NE_{i,j,k}$ constitute the region-of-influence of (i, j, k). The spherical SOM is shown in Figure 3.5.

Training proceeds much the same way as in the standard SOM, where a weight vector is associated with each node in the lattice and adapts to represent exemplars (clusters) in the input space.

The following are steps of the self-organizing algorithm used to generate the spherical map:

► *Step 0.* Initialize the weight vectors to small random values and set the desired number of cycles, N_{cycle}.

► *Step 1.* Randomly select an input vector x^i from the dataset. For each selected input vectors do *Steps 2-4.*

► *Step 2.* Compute the error or difference $E^i_{i,j,k}$ between the input vector and the weight vectors for all the cluster units (i.e. nodes) in the network using

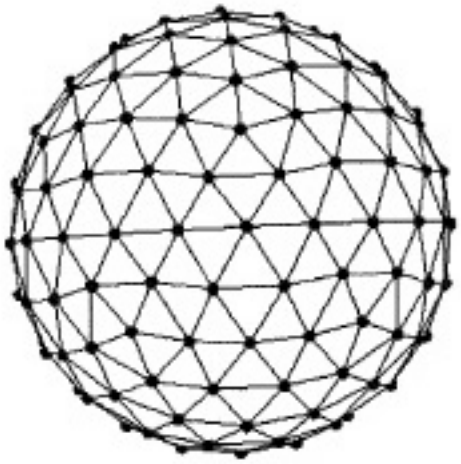


Figure 3.4: Data association in a spherical SOM

$$E^i_{i,j,k} = \varphi(u_{i,j,k}) \sum_{n=1}^{N} \|x^i_n - w_{n,i,j,k}\| \tag{3.1}$$

where $\varphi(u_{i,j,k})$ is a count-dependent non-decreasing function used to prevent cluster under-utilization [30], and $w_{n,i,j,k}$ is the weight from the n^{th} input to the $(i,j,k)^{th}$ cluster unit given $i = 1, 2, ..., Ij = 1, 2, ..., J$, and $k = 1, 2, ..., K$.

► *Step 3.* Select the winning cluster unit $(i,j,k)*$ to be the one with the minimum error, $E^i_{i,j,k}$. This determines the mapping Φ between an input vector and the weight vector of a node in the SOM lattice.

$$(i,j,k)* = min\{E^i_{i,j,k}\} \tag{3.2}$$

► *Step 4.* Update the weights associated with the winning cluster unit $(i,j,k)*$ and all the units residing within the specified neighbourhood $NE_{i,j,k*}$ using

$$w_{i,j,k*}(new) = w_{i,j,k*}(old) + \alpha[x^i - w_{i,j,k*}(old)] \tag{3.3}$$

where

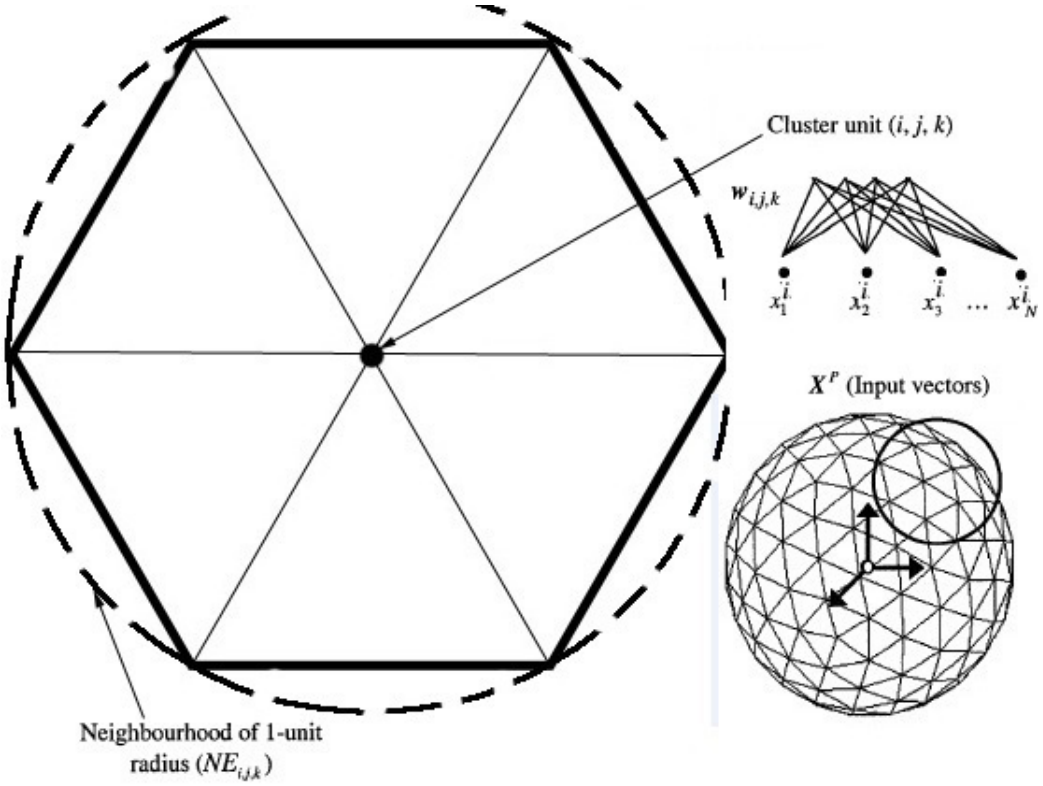


Figure 3.5: The spherical self-organizing map

$$\alpha = \mu(\frac{NE_{i,j,k*}}{NE_{initial}}), \tag{3.4}$$

and

$$NE_{i,j,k*} = f(N_{cycle}), \tag{3.5}$$

μ is a predefined learning rate, and $NE_{initial}$ is the initial neighbourhood size in terms of the number of units. The neighbourhood size is gradually reduced in discrete steps during training such that only the winning unit is updated at the end of the training period. This is achieved using heuristic rules that are dependent on the initial neighbourhood size.

► *Step 5.* Check the stopping criterion. If the stopping criterion is not satisfied then go to *Step 1* and repeat the process.

3.2.3 Significance of the Spherical Topology

One of the advantages of the Spherical Self-Organizing feature map over the conventional 2D SOM topology is its closed structure, which makes it possible to reflect inter-node associations in the map in the form of a closed 3D shape. In [31] Brennan and Hulle conducted a case study in which they demonstrated the advantages of Spherical SOM, where, in particular, they looked at the *border effect* (border neurons are surrounded by fewer neurons) - a common feature of a 2D SOM. The case study involved a real world data set consisting of 144 students from a university-level school for Science and Arts in Brussels, Belgium. 14 different scores in different skills were used such as general literacy, skills using operating systems, skills with email, text processing, presentation software and other computer or language related skills. First they trained a 2D SOM of size 13x5 neurons using the given data, followed by a training of a Spherical SOM. By using the collected Best Matching Units (BMU's), authors were able to calculate the quantization error and in the case of the flat (2D) SOM the values were higher: 3.036 vs. 2.051 of the Spherical SOM. Brennan and Hulle concluded by mentioning that the Spherical SOM distinguished itself from the flat SOM in their case study, showing their representations to be more suitable to modeling the data set [31].

As it will be revealed later in this thesis, a SSOM is a good tool for visualizing temporal data that slowly changes with time. Because of its *bounded* nature, such data has the freedom to map anywhere in its lattice, interacting with the neighbour data in manner which is not possible in a conventional 2D SOM. Trajectories forming from such mapping can be used as features itself when analysis new data (in this thesis trajectories are used to classify new gestures).

3.3 Temporal data analysis using SOM and SSOM

Similar as to gestures, video contains gradually changing data. As one can decompose gestures into postures, the same can be said about video - it can be decomposed into a series of keyframes, each representing a still image (snapshot) of the video. A set of postures or keyframes form a gesture or a video respectively. This Section begins by introducing some literature review on how the SOM has been used mainly for video scene/shot detection and video summarization. It continues by introducing an experiment conducted to demonstrate how the use of SSOM can decompose a video clip into their respective scenes and shots.

3.3.1 SOM based approaches for video analysis

One of the recent studies has been trying to analyse video data and how it maps onto a SOM. This way in [5] authors proposed a method for *Video Segmentation and Shot Boundary Detection Using Self-Organizing Maps*. Their video shot boundary detection (SBD) algorithm spots discontinuities in the visual stream by monitoring video frame trajectories on Self-Organizing Maps. Their argument is that the SOM space resembles the probability density differences in the feature space and that distances between SOM coordinates give more information than distances between plain features taken from the video stream.

SOM can be very helpful in video segmentation. The reason being is because consecutive frames in a continuous video segment that has been filmed by a single camera are normally visually similar. In this manner any transition effects can be categorized into *abrupt cuts* and *gradual transitions* based on how fast these changes happen in the video stream. For instance, in cut transitions the change from one shot to another is instantaneous, whereas in the gradual scene change the transition has some duration.

In the proposed technique in [5] authors call the path that the data maps onto the SOM as the time advances a *temporal trajectory*. At each moment in time a feature vector is first calculated from the raw video and then a vector is mapped to the best matching unit

(BMU). This process is illustrated in Figure 3.6.

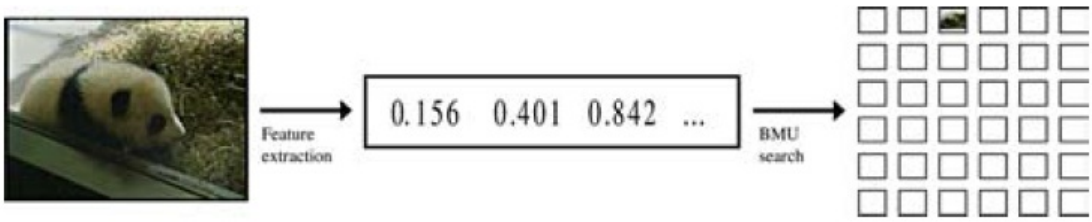


Figure 3.6: The feature vector is calculated from the raw data. Then the vector is mapped on the SOM by finding the best-matching map unit [5]

In Figure 3.7 one can observe the gradual and abrupt changes in the trajectory as frames are being mapped onto their BMUs on the SOM. Similar frames are mapped close to each other and the distances between consecutive frames can be used to find discontinuities in the video segment.

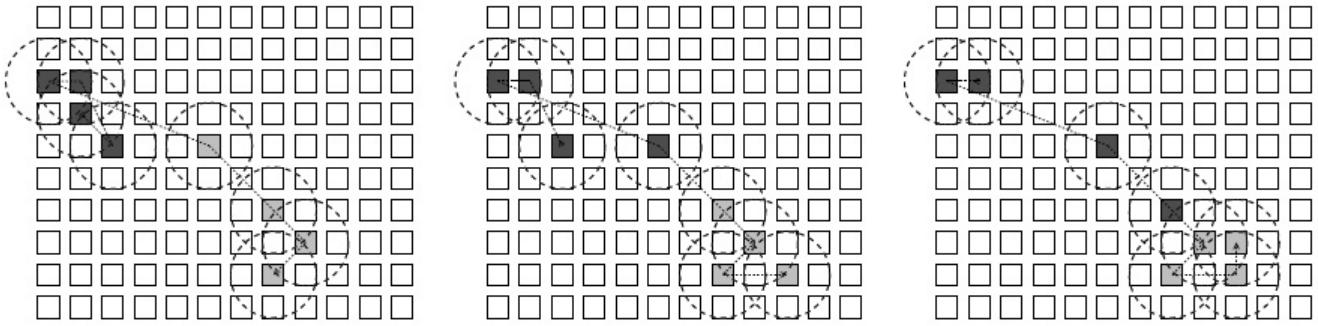


Figure 3.7: Segments of trajectory at three different time steps. The trajectory is illustrated with a black dotted line [5]

For their work authors in [5] use a *sliding frame windows* technique to detect a shot boundary. In simple words, they use distances between multiple frames before and after a current point of interest, instead of using map distances between two consecutive BMUs.

If the lengths of the preceding and following frame windows are l_p and l_f, the sets of the preceding and the following frames, $S_p(t)$ and $S_f(t)$, at time instants $t \in \{l_p, l_p + 1, ...\}$ can be expressed mathematically as:

$$\begin{aligned} S_p(t) &= \{f_{t-l_p}, f_{t-l_p+1}, ..., f_{t-1}\} \\ S_f(t) &= \{f_t, f_{t+1}, ..., f_{t+l_f-1}\}, \end{aligned} \tag{3.6}$$

where f_n denotes frame n of the video. The distance measured at time t on SOM k can be formulated as:

$$d(k,t) = d(k; S_p(t), S_f(t)) = min_{f_i,f_j} \|C_k(f_i) - C_k(f_j)\|; f_i \in S_p(t), f_j \in S_f(t). \tag{3.7}$$

The function $C_k(f_n)$ maps frame f_n on SOM k by calculating the corresponding map-specific feature vector value and finding the BMU. The function returns the discrete BMU coordinates (x, y). The decision of whether there is a shot boundary between the frame windows $S_p(t)$ and $S_f(t)$, i.e. between frames $f_t - 1$ and f_t is done by comparing the distance to a fixed SOM specific threshold T_k such that $d(k,t) > T_k$. For the training authors have used standard MPEG-7 descriptors: *MPEG-7 Colour Structure,MPEG-7 Dominant Colour, MPEG-7 Scalable Colour, MPEG-7 Region Shape and MPEG-7 Edge Histogram.* Other features used were *Average Colour, Colour Moments, Texture Neighbourhood, Edge Histogram, Edge Co-occurrence and Edge Fourier* [32].

Similarly in [6] Koskela and Laaksonen used the above method to do *rushes summarization using self-organizing maps* for the BBC rushes summarization task [33] of TRECVID[1]2007 [34]. After applying the shot boundary detection algorithm (SBD) to the rushes videos, the shots obtained are analysed further for their summarization. A new idea introduced here is the construction of shot-wise signatures of video segments. As authors explain in their work, this is needed to analyse the trajectory of the consecutive BMUs during the shot and

obtain a temporal signature representing the dynamic structure of the shot. This approach might be needed if for instance, one wants to distinguish a scene where a man walks into a room from another where a man walks out of the room since this method takes into account the temporal or causal aspect of the video. Without going into much detail, representative frames and SOM signatures of video segments are built and compared to each other to find unique segments for building the video summary, see Figure 3.8.

Figure 3.8: Representative frames and SOM signatures of three video shots [6]

Below of each video frame, as it appears as red spots are the convolved (low-pass filtered) hit distribution on the SOM surface which form the basis of the signature (due to the topology preservation property of the SOM, authors decided to smooth the representation by forcing the neighbouring SOM units to interact with each other). The hit frequencies give a discrete histogram representing the visual contents of the shot. Three different SOMs were used with different features such as color layout, edge co-occurrence and edge histogram to obtain the three signatures below each representative frame.

[1]TREC Video Retrieval Evaluation - a conference series by the National Institute of Standards and Technology, to encourage research in information retrieval by providing a large test collection and uniform scoring procedures.

Thomas Barecke in [7] also uses a SOM to do a *summarization of video information*. In his approach Berecke divides a video sequence into (still-) image, audio, and motion information, but only uses still image features. After applying a shot detection algorithm on the video he uses a representative keyframe from each shot, from which the author extracts a color histogram using a specified color space. His system supports IHS, HSV, and RGB color models[2]. Every histogram is described by a numerical feature vector and the resulting description for a video sequence is a set of vectors. These sets of vectors are then used to train the self-organizing map. As Berecke mentions the main difficulty is defining an appropriate size for the map. This occurs because a 2D SOM has to have big enough grid size to create as many clusters as it is needed to separate the keyframes. An extension of self-organizing maps that overcomes this problem is the growing SOM [35]. Its main idea is to start with a small map and then add during training iteratively new units to the map, until the overall measured error is sufficiently small.

Figure 3.9 illustrates the Growing self-organizing map after its training. The brightness of a cell indicates the number of shots assigned to each node. Also, on each node the keyframe of the shot with the smallest difference to the cluster center is displayed. The main role of a SOM here is to separate the keyframes into clusters which then can be used to summarize a video.

Dim P. Papadopoulos in his paper *Video Summarization Using a Self-Growing and Self-Organized Neural Gas Network* [36] demonstrates another video summarization technique using a SOM. The novelty of his method consists in using *Compact Composite Descriptors* (CCDs) to describe each frame on the video. The family of CCDs includes the following four descriptors:

[2]RGB color model - an additive color model in which red, green and blue light are added together in various ways to reproduce a broad array of colors; HSV color model - a blend of three color components: Hue, Saturation and Value; IHS color model - a blend of three color components: Intensity, Hue and Saturation

Figure 3.9: Growing Self-Organizing map after training [7]

1. the Color and Edge Directivity Descriptor (CEDD) [37]

2. the Fuzzy Color and Texture Histogram (FCTH) [37]

3. the Brightness and Texture Directionality Histogram (BTDH) descriptor [38] and

4. the Spatial Color Distribution Descriptor (SpCD) [39]

In order to classify the frames into clusters, the method utilizes a Self-Growing and Self-Organized Neural Gas (SGONG) network. Its main advantage is that it adjusts the number of created neurons and their topology in an automatic way [36].

3.3.2 Video information analysis based on SSOM

As seen from the literature review in the previous section, it is a common practice to use a regular flat (2D) SOM for clustering or separating multi-dimensional video data. All the experiments seen here use a SOM in an attempt to decompose the video into small entities such as keyframes or so called *phases* (e.g. scenes, shots). Normally, by studying how this entities map on the SOM lattice, one can guess the similarity between the consecutive frames or *phases*. Due to SOM's topology preservation, it is evident that semantically similar shots and scene from the video will map in a nearby neighbourhood on the SOMs surface.

The attempt in this thesis is to research alternative SOM structures to represent slowly changing data such as video or gestures. In this thesis, the ability of Spherical Self-Organizing Maps to describe and represent the data in such a manner that it can be compared among other similar data of the same nature is studied. This thesis builds upon the believe that the Spherical SOM, because of its nature, can uniformly distribute a smoothly varying data on its lattice. This work studies the path that the data traces in a form of a *trajectory signature*. Because of its Spherical nature, SSOM does not have the limitations of having boundaries as the 2D version and the trajectories can freely be created on the whole sphere.

The following experiment was conducted to demonstrate the smoothness property of a Spherical SOFM on a slowly varying video data. The video clip used for this experiment came from a movie scene of a Hollywood movie taken from *HollyWood2 - Human Actions and Scenes Dataset*[3] [40]. The goal was to visualize the separation of the video clip into different scenes or shots. In this case, the video clip consisted of three semantically and visually different parts. The video clip was separated into its keyframes using FFmpeg[4]software and the beginning and end of each shot was known beforehand.

[3]www.di.ens.fr/ laptev/actions/hollywood2

[4]FFmpeg is an open source, complete cross-platform solution to record, convert and stream audio and video with various audio/video codec libraries: http://ffmpeg.org

Figure 3.10 shows some of the sample keyframes that were used during the training process. From the figure it is clear that the video clip consists of three visually different shots. In order to train the network and get a good separation results the following steps were taken:

1. Separated the video clip into its keyframes
2. Extracted Features from each keyframe
3. Trained the Spherical Self-Organizing Feature map with the extracted features
4. Visualized the various shots by mapping individual frames onto the SOFM and colouring them according to their shot

As mentioned before, the video clip had to be split into its keyframes in order to extract some features that would represent the video. Some experimentation was needed at this point, since it was required to choose a set of features that would represent the video in the best manner. It was chosen to use the following features: *Color Histogram [41], Color Layout [42], Color Moment [43], and Edge Histogram [44]. Color Histogram* was chosen because it gave the ability to trace the color changes throughout the keyframes and detect where sudden color changes happened. *Color Layout* was chosen because of its ability to capture the spatial distribution of the colors in the keyframes.*Color Moment* was chosen because it is a good texture descriptor, especially in the area of *content-based image retrieval* [45]. And finally the standard MPEG-7[5] *Edge Histogram Descriptor* was used because of its compactness in describing edge distribution with a histogram based on local edge distributions in images [46].

The next step was to adjust our Spherical SOM parameters for its training. The SOM consisted of 2562 nodes uniformly distributed on the sphere. This number of nodes allowed to have a uniform number of neighbours between each node of five or six neighbours and was

[5]Multimedia content description standard or formally Multimedia Content Description Interface

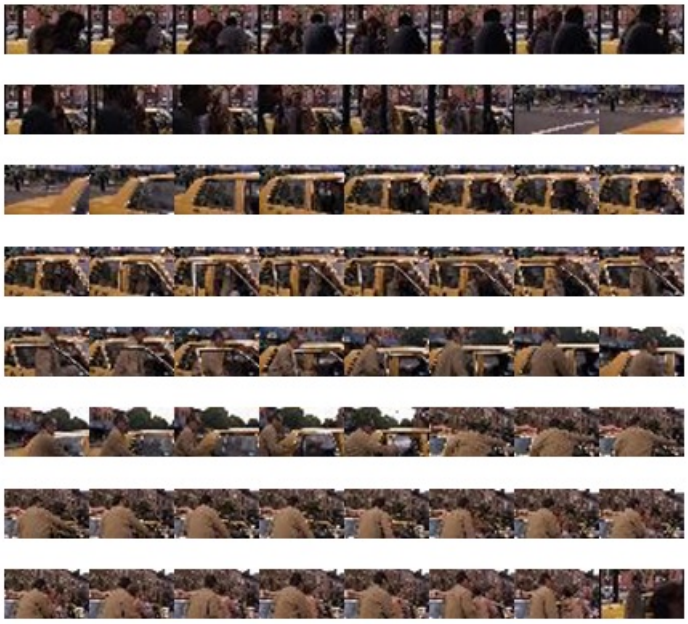


Figure 3.10: Sample frames used for training the Spherical SOFM (taxi scene)

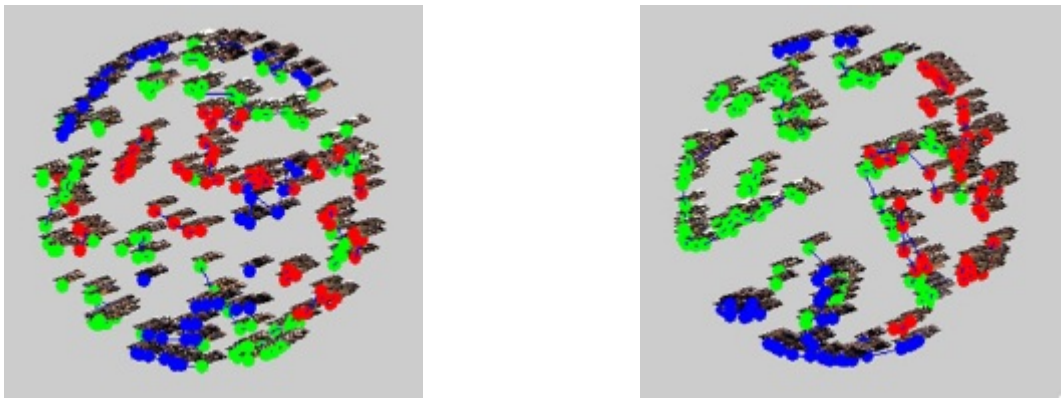
big enough to cluster a big dataset containing over a thousand data samples. Please, refer to Fig. 3.4 for a topological view of the Spherical SOM. The other two important parameters for the experiment were the *neighbourhood operator* and the number of epochs or training cycles through which the network had to go through to obtain satisfactory results. These two parameters were experimentally adjusted through several trials. For instance, it was found that for the neighbourhood operator value of four the SOM gave a good separation between the shots (this process can be seen in Fig. 3.11-3.12. Also, after twenty epochs of training only small changes were noticed in the distribution of data on sphere, so 20 epochs were used for the experiment.

Following the algorithm of weight adaptation given in Section 3.2.2 all the features for

each keyframe were converted to a feature vector and concatenated into 375x165 vector containing all the features, where 375 represented the number of frames used for training and 165 represented the length of all the features concatenated together (64 for Color Histogram, 12 for Color Layout, 9 for Color Moment, and 80 for Edge Histogram).

Figures 3.11 and 3.12 show the result after the training of the network. All the keyframes were mapped onto their corresponding best matching units on the sphere. The different colors represent a different shot or scene in the video. On this particular video there are three different scenes. One can see how the value of the neighbourhood operator influences on the separation between the keyframes. The higher the value the better the separation between individual keyframes belonging to different scenes. After a few trials of training, it was determined that the best result was given by a neighbourhood operator value of 4 depicted in Fig. 3.12b. A zoomed version of the SOM is shown in Fig. 3.13, where it is clearly seen that there is practically no overlapping between the three scenes.

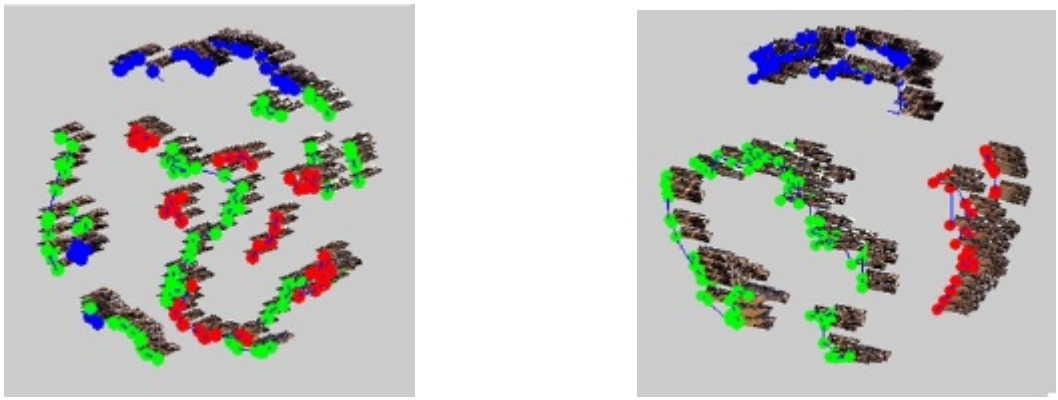
Also, to see the benefits of using all the four features (Color Histogram, Color Layout, Color Moment, Edge Histogram) as opposed to using them separately for training, a few more experiments were conducted. Figures 3.14 and 3.15 show the result of the SOM after training with each single feature separately. As one may notice, none of the maps gave a perfect separation between the different scenes as some keyframes mapped onto the wrong cluster. The linkage between the keyframes on the map was artificially created based on the distance between the keyframes. That is the reason why the keyframes appear connected to each other. This way all the neighbouring keyframes were linked together as seen in Fig. 3.13. Whenever bigger jumps occurred the linkage between the keyframes was broken. This can be seen in the video shot shown in green color in Fig. 3.16. The frames were removed on purpose so that the linkages could be seen more clearly. Only the nodes on the lattice were left for visualization. The discontinuities have been shown with black circles. What these jumps tell us is that the change in the consecutive keyframes has been big enough in



(a) trained with a neighbourhood operator value of 0.5

(b) trained with a neighbourhood operator value of 1

Figure 3.11: Spherical SOM after training. The color green, red, and blue show the different edges belonging to the three different shots.



(a) trained with a neighbourhood operator value of 2

(b) trained with a neighbourhood operator value of 4

Figure 3.12: Spherical SOM after training. The color green, red, and blue show the different edges belonging to the three different shots.

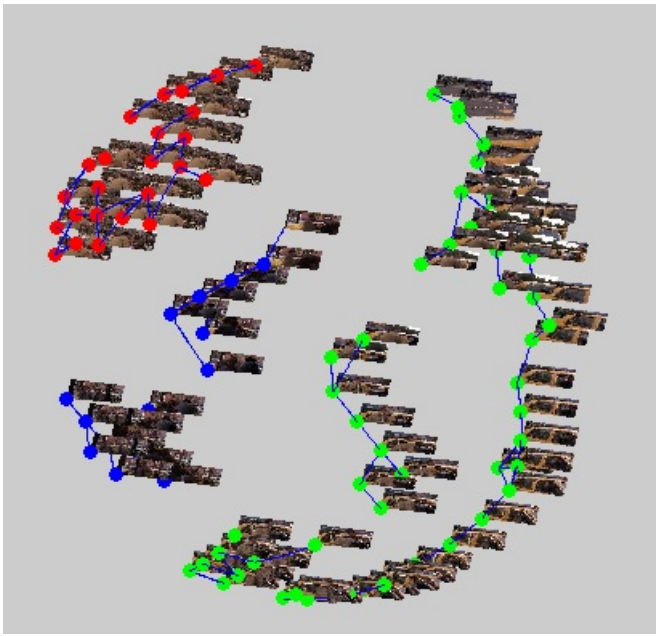
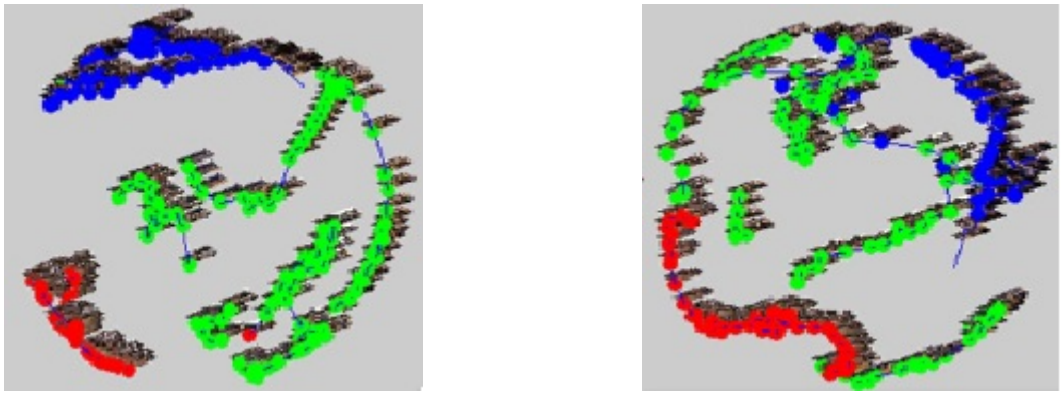


Figure 3.13: Video scene/shot separation using Spherical SOM

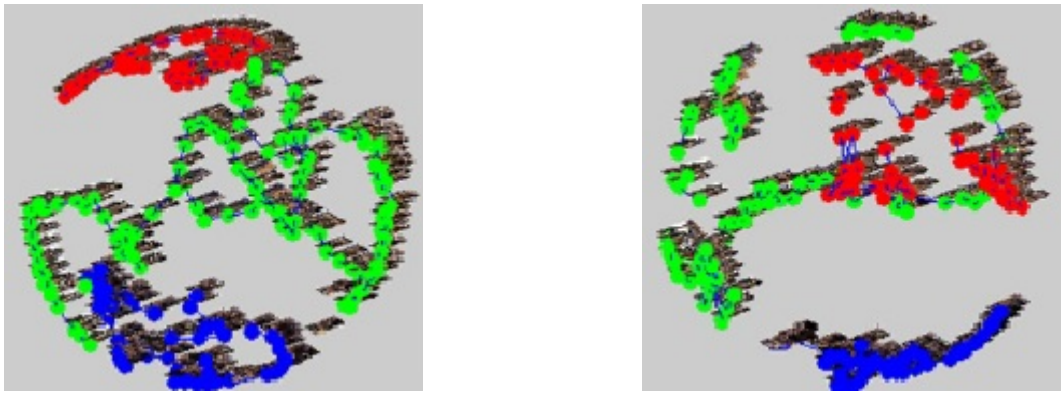
the video shot such that the trajectory had to be continued somewhere further out from its immediate neighbours.

The experiments that were conducted with the video scenes gave a new thought on how the Spherical SOM could be used the problem domain of interest for this work: namely gesture recognition. It is understood the difficulty in representing a video clip and its various scenes and shots with a trajectory because of the complexity of the video. For instance, video does not necessarily have to change smoothly from frame to frame, and may experience abrupt jumps on its trajectory. It is difficult to analyze such instances because visually similar frames can overlap in clusters that they do not belong to and may introduce some errors in defining a trajectory for a scene. Instead, it was thought that by treating all the elements



(a) trained with Color Histogram (b) trained with Color Layout

Figure 3.14: Spherical SOM after training with individual features



(a) trained with Color Moment (b) trained with Edge Histogram

Figure 3.15: Spherical SOM after training with individual features

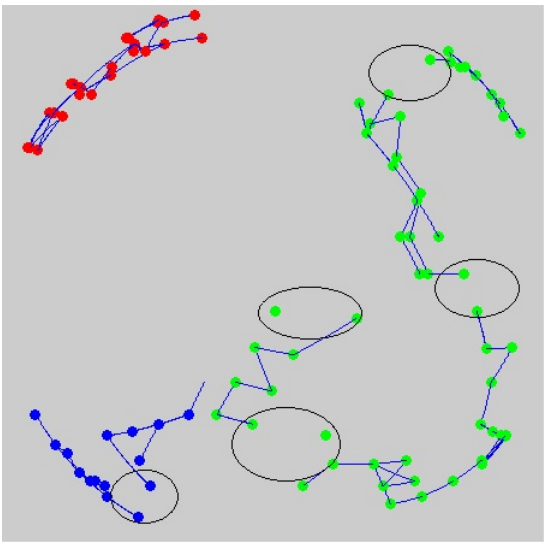


Figure 3.16: Linkage between keyframes in the video shots

falling in the trajectory as snapshots or postures that form the path of the trajectory, These trajectories themselves can carry useful information that could be analyzed directly. With this thought in mind, the next section introduces four different approaches based on SSOM trajectories aiming to tackle gesture recognition.

3.4 Four approaches to gesture recognition through trajectory analysis

This section introduces four gesture recognition approaches through the use of trajectories traced on the SSOM lattice. One of the advantages of the SSOM for this kind of application is that regions of density found in the feature space will map to equally spaced and well separated locations on the sphere due to the wrap-around effect of the lattice. In this thesis, this property is leveraged to build trajectory based features to distinguish between human full body motions and hand gestures.

3.4.1 Approach #1: gesture recognition using all postures

The gesture recognition and classification initially starts with a simple approach. As mentioned earlier, all the BMUs that trace a trajectory for a specific gesture are considered as postures. A *trajectory* in this case, is referred to the temporal path that the data maps into on the SSOM lattice based on a set of consecutive BMUs, Fig.3.17. All the BMUs from each of the gesture classes are used as a collection of nodes or postures for the purpose of classification of unknown data (Note: the datasets for these experiments were introduced in Section 1.1.1). This is done in the following manner:

1. All the BMUs falling into trajectories belonging to a specific gesture are recorded into a set G_i as follow:

$$G_i = \{Tr_{(i,1)}, ..., Tr_{(i,m-1)}, Tr_{(i,m)}\}, \tag{3.8}$$

   where $Tr_{(i,j)} for j = 1, 2, ...m$ is a trajectory forming a gesture G_i and i is the gesture index which represents a gesture class, also

$$Tr_{(i,j)} = \{P_k, ..., P_{n-1}, P_n\}, \tag{3.9}$$

   where P_k is the k^{th} node in the Spherical SOFM lattice (i.e. posture) and n is the number of nodes or postures in the trajectory $Tr_{(i,j)}$.

2. Feature vectors (consisting of the coordinates of the body parts and sensor data from Microsoft Kinect dataset and Nintendo PowerGlove accordingly) of an unknown gesture coming from the testing portion of the dataset are then compared against the weights of the SSOM and the BMUs from the new trajectory of an unknown gesture are collected into a new set T_p similarly as in Eq. 3.9, where p is the index of the new unknown gesture.

3. A frequency posture counter K_i assists in determining the class of the unknown gesture,

where i represents an index of a known gesture. The counter K_i for a gesture i is incremented if a posture from an unknown gesture belongs to a gesture being compared against. This way T_p is compared against all the G_i in the database. So, if $K_i \geq K_1, K_2, ..., K_n$, where K_n is a counter belonging to gesture with index n, then K_i is chosen as the winning counter and the unknown gesture is classified as gesture G_i, see Fig.3.18.

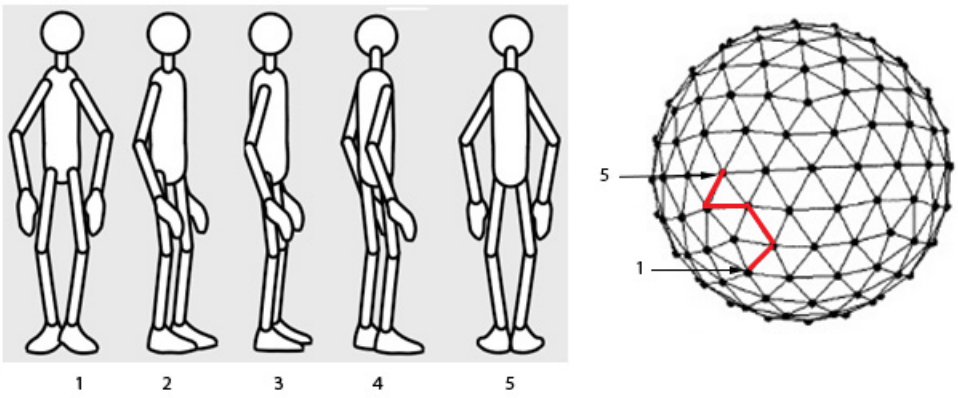


Figure 3.17: Temporal sequence of postures representing an arbitrary gesture. In this example a simple gesture consisting of 5 postures is displayed. The mapping of the gesture is shown as a trajectory on the SSOM in red

3.4.2 Approach #2: gesture recognition using weighted aggregation of all postures

The approach taken for gesture recognition in the last section only takes into consideration a set of postures as a primary data to classify an unknown gesture. This set does not take into account the frequency with which a specific posture is encountered in a gesture's trajectories. For this reason a frequency factor is introduced in this approach. All postures are aggregated into a set, but at the same time each posture is associated with a weight. The more a posture appears in a gesture path while training the network, the more weight it has towards that

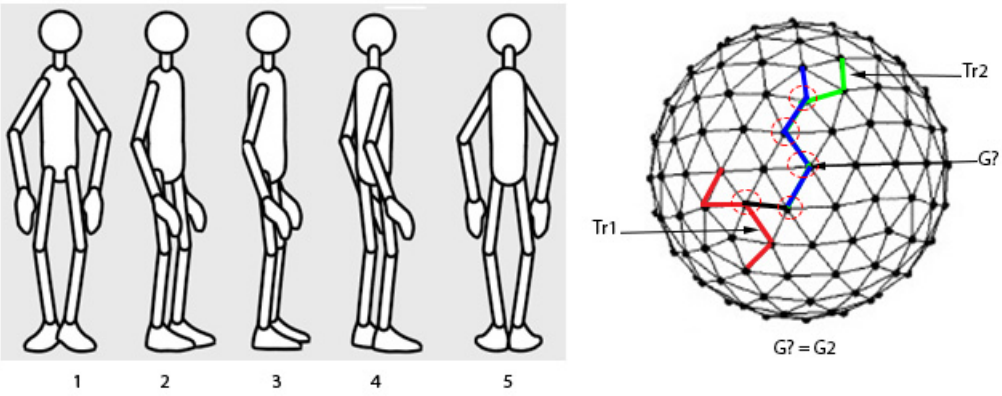


Figure 3.18: Showing the gesture recognition process: the classification occurs by creating a counter for the unknown gesture (blue trajectory), which counts the common instances or postures between the unknown gesture trajectory and the already learned by the SSOM trajectories (trajectories 1 & 2, red and green respectively). A histogram is then built based on these counters and a gesture is classified according to the highest value obtained in this histogram

gesture. The reason behind the weight factor is that the path that a trajectory, representing a specific gesture maps, on the lattice of the SSOM tends to activate the same neurons (postures), giving it a higher probability to appear again if the same gesture is traced, see Fig.3.19

Mathematically this can be expressed as P_k (See Eq. 3.9) having a weight factor w_i associated with it. Simply, when making a decision about a new unknown gesture w_i is multiplied by K_i rather than just using the frequency posture counter all by itself as in the previous Section. So, if $K_i * w_i \geq K_1 * w_1, K_2 * w_2, ..., K_n$, where K_n is a counter belonging to gesture with index n, then K_i is chosen as the winning counter and the unknown gesture is classified as gesture G_i.

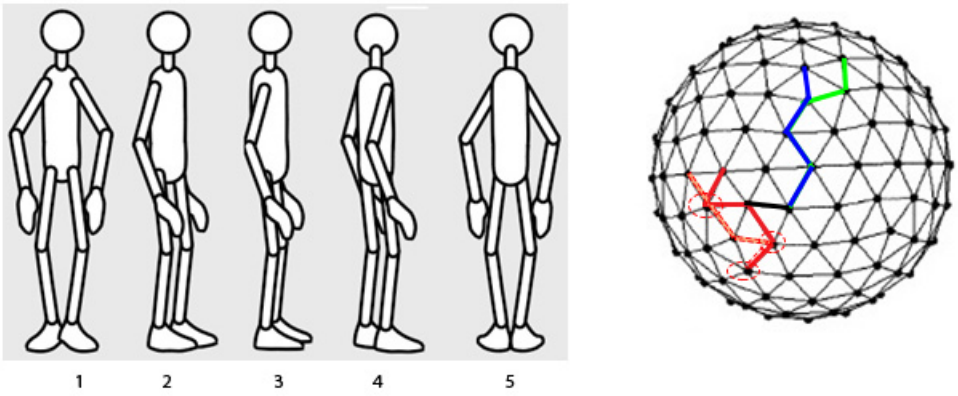


Figure 3.19: Showing the gesture recognition process: Assume the SSOM learned the path of two trajectories from gesture 1 class (pink & red). These two trajectories have common postures in their path. In this method we look at the frequency information obtained from trajectories within one gesture class. Specifically, we add a weight factor representing how frequent a specific posture was found in the path of a trajectory from a specific gesture class. The unknown blue trajectory is classified using this frequency information

3.4.3 Approach #3: gesture recognition using posture transitions

Previous approaches, treated each BMU as a posture and no dynamic information was used. Dynamic information in this case refers to the posture transitions that occur during the tracing of a gesture onto the SSOM. The main argument here is that trajectories belonging to the same gesture should follow not only the same path on the Spherical lattice but also have similar transitions in terms of postures. For example, when a person performs a "Driving" gesture, he or she will follow the same posture transition as he or she moves the hands in a 3D space. For this specific reason the classification of an unknown gesture is evaluated based on similar posture transitions during the formation of its trajectory. G_i still consists of a set of trajectories as shown in Eq. 3.8, but instead:

$$Tr_{(i,j)} = \{R_k, ...R_{n-1}, R_n\}, \tag{3.10}$$

where R_k is the k^{th} posture transition and n is the number of posture transitions in the trajectory $Tr_{(i,j)}$.

The classification of an unknown gesture is done similarly as in the method using all postures in approach #1, but instead of having a K_i being a posture counter, now it is replaced by a posture transition counter. So, if $K_i \geq K_1, K_2, ..., K_n$, where K_n is a gesture transition counter belonging to a gesture with index n, then K_i is chosen as the winning counter and the unknown gesture is classified as gesture G_i, where i is the index of a known gesture, see Fig.3.20

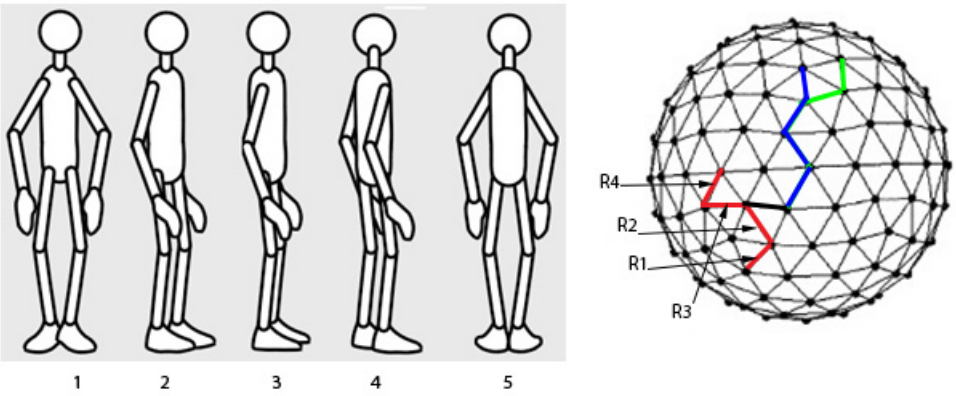


Figure 3.20: In this approach we use temporal information obtained from trajectories, specifically we use posture transitions represented as R_i. A histogram is built based on how many common posture transitions occur between the unknown gesture trajectory and all the gesture trajectories learned by the SSOM

3.4.4 Approach #4: gesture recognition using weighted aggregation of all posture transitions

The last approach in this thesis uses a weighted aggregation of all posture transitions. Similarly as in the method with weighted aggregation of all postures a weight is introduced.

This approach not only takes in consideration the posture transitions happening in a gesture trajectory but also the frequency with which these transitions occur. For this approach Equation 3.10 is still used but the classification process is based on the following statement: if $K_i * w_i \geq K_1 * w_1, K_2 * w_2, ..., K_n$, where K_n is a counter belonging to gesture with index n, then K_i is chosen as the winning counter and the unknown gesture is classified as gesture G_i. Here w is the weight of a particular posture transition.

3.5 Summary

This chapter has introduced the Spherical SOM. Details about the SSOM configuration, important parameters and benefits over a SOM were discussed. It was evident from the discussion that the SSOM is an optimal choice for applications such as analyzing video clips and gesture data, because of its ability to minimize topological discontinuity. Some examples from literature were brought up, where SOM was used as a tool for analyzing smooth and abrupt changes in the video stream, highlighting the advantages of SOM. An experiment was shown with a video clip from a movie, where regions of density found in the feature space were mapped into equally spaced and well separated locations on the sphere of SSOM clearly showing the wrap-around effect. Finally, four different approaches to gesture recognition were presented based on the use of trajectories obtained from the mapping of gesture data onto the SSOM. Here, the trajectories were used as features themselves to extract useful information and create a baseline to compare against unknown gesture data samples.

The next chapter will present the experimental results based on the approaches described above.

Chapter 4

Experimental Results

4.1 Experimental setup

In all the experiments performed in this thesis the setup was identical. A Spherical SOM was used with specific settings and size which will be discussed shortly. All the experiments were performed on stand alone PC with Windows 7, 4GB of RAM and Intel Core i7 CPU (2.67GHz). MATLAB R2011b environment was used for all the experiments and the visualization part. On average it took several minutes to train the network with one gesture depending on the feature vector size of a specific gesture.

Similarly as with the SSOM experiment with the video clip, where some of the Spherical SOM properties were explored, the same topological structure was used. The SOM consisted of 2562 neurons (nodes) uniformly distributed on the sphere. Similarly as in the video scene experiment performed in the previous chapter, this number of nodes gave a uniform number of neighbours between the nodes and was big enough to cluster the gesture datasets containing thousands of samples each. The *neighbourhood operator* was also left intact and was set to a value of 4, as it was already tested to give a fairly good separation while exploring the video scenes. Finally, the number of epochs was set to 20.

The network was trained once for every gesture dataset. For both datasets, half was

used for training purposes and the other half was used for gesture recognition (25 gesture samples from the Microsoft Kinect dataset, and 35 gesture samples from Australian Sign Language dataset (Nintendo PowerGlove)). The data from the Microsoft Kinect sensors and the Nintendo PowerGlove was directly used as a feature space for training the SOM. No data normalization was necessary in this case. Every feature vector from the feature space represented a specific state for a given gesture. These states are treated as postures, similarly as keyframes in a video stream.

The main motivation behind these experiments was to obtain a trajectory representation of the gestures datasets and use this information for gesture recognition. The fact that gesture information is a smoothly varying data was used as the main argument to use a Spherical SOM. Because of its ability to reduce dimensionality and represent dynamic structures well without boundaries such as video or gesture information a SSOM gives the best option in terms of data visualization.

Below is shown the training process of the SSOM with all the variables and feature space given:

1. Weight vectors w are initialized to small random values and the desired number of cycles N_{cycle} is set to 20.

2. Input vector x^i is randomly selected from the dataset (since two trainings were performed, the two datasets were used separately one at a time), See Figure 1.10 for a feature vector example from Microsoft Kinect gesture dataset.

3. The error or difference $E^i_{i,j,k}$ between the input vector and the weight vectors for all the cluster units (i.e. nodes) in the network is computed using Equation 3.1.

4. The winning cluster unit (i, j, k) to be the one with the minimum error, $E^i_{i,j,k}$ is selected.

5. The weights associated with the winning cluster unit (i, j, k) and all the units residing

within the specified neighbourhood $NE_{i,j,k}$ are updated using Equations 3.3, 3.4 and 3.5.

6. The steps above are repeated until all N_{cycle} are used.

The next few pages show some sample trajectories that were obtained during the mapping process (See Figures 4.1 - 4.10). These gesture trajectories are a representation of the BMUs hit sequence that each gesture class mapped onto the SSOM. Information from these trajectories was used for gesture recognition and classification. The data that is being used in the trajectory mapping comes from the training portion of the datasets. All the BMUs are in 3D space although they appear as a 2D images. The lattice of the Spherical SOM was removed on purpose so that the trajectories could be seen more clearly. Three trajectories from each gesture class for both Microsoft Kinect and Nintendo PowerGlove were chosen randomly.

Each gesture class was displayed at a specific and different angle from the rest in order to show the data path more clearly. From Figures 4.1 - 4.10 it is evident that the trajectories for each gesture class trace a similar if not identical path on the spherical lattice of the SSOM. It is also clear that each gesture leaves a path which is unique if comparing to other gestures, although it is important to point that many gestures might and will have common BMUs since they may contain similar postures that trace a specific gesture. Every BMU hit of a gesture is considered a posture belonging to a given gesture class. A collection of these postures forms a gesture.

4.2 Results

4.2.1 Approach #1: gesture recognition using all postures

Figures 4.11 and 4.12 show some of the results obtained from the registration process. The results have been normalized between zero and one. There are 25 gestures tested from Mi-

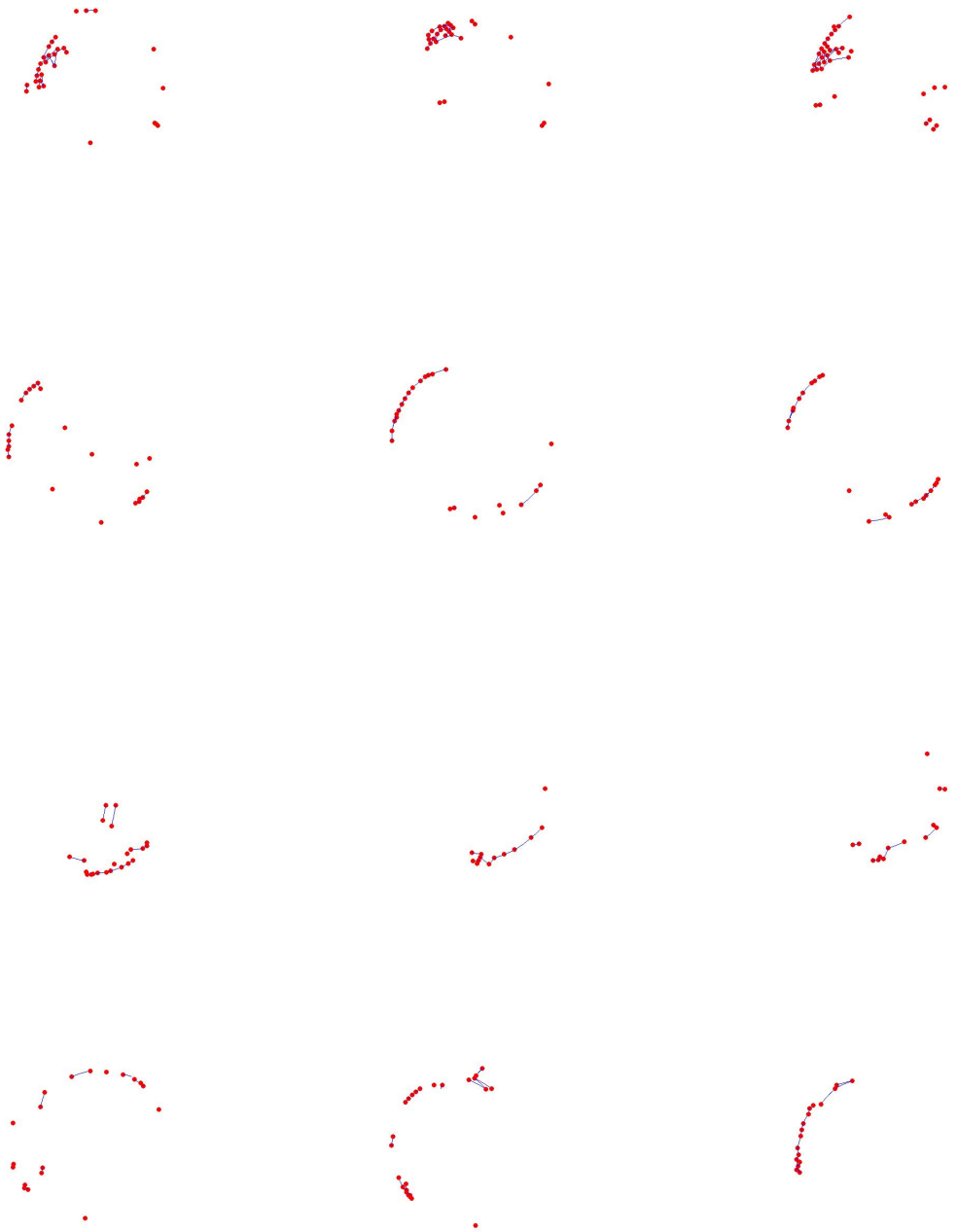


Figure 4.1: PowerGlove Gesture trajectories 1. First row: Alive; Second row: All; Third row: Answer; Fourth row: Boy

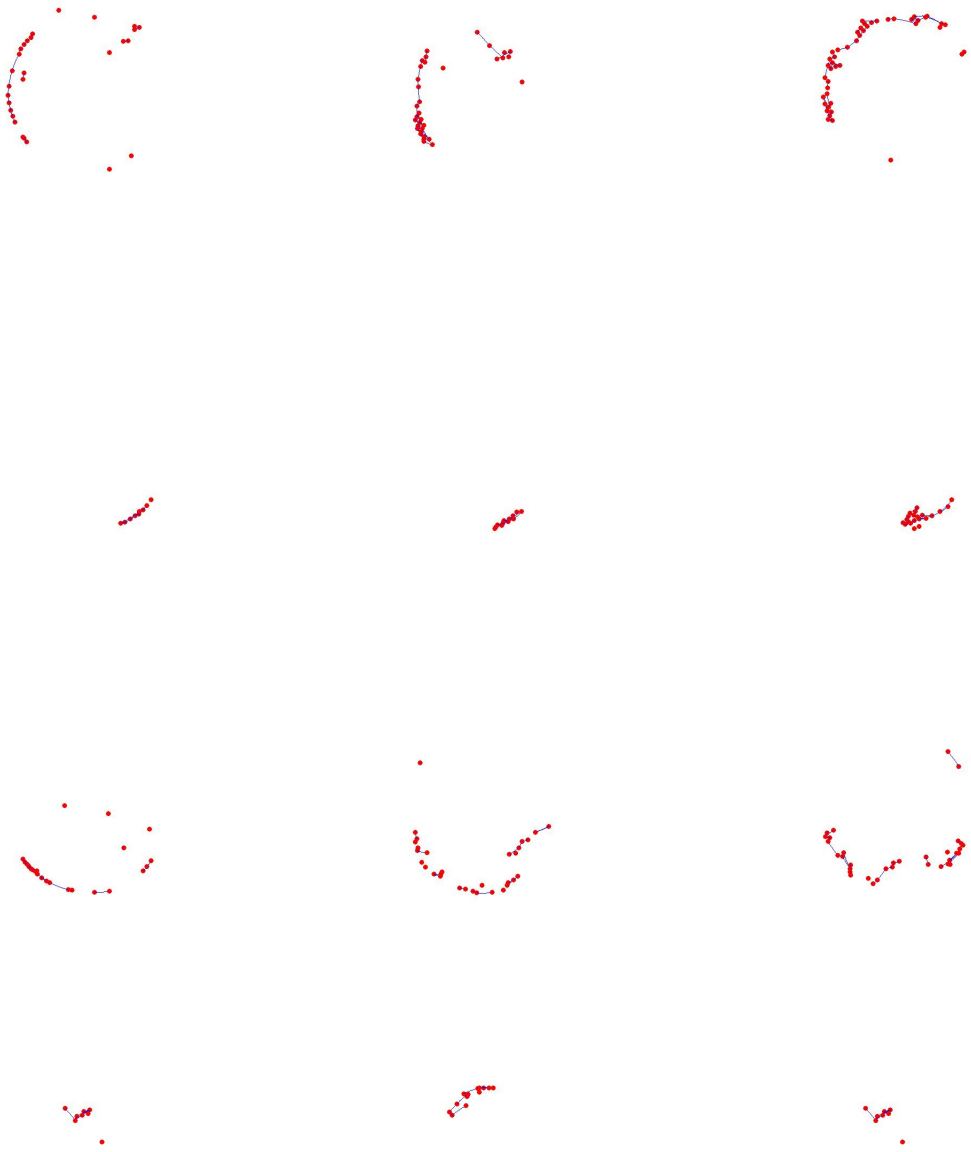


Figure 4.2: PowerGlove Gesture trajectories 2. First row: Building; Second row: Buy; Third row: Change; Fourth row: Cold

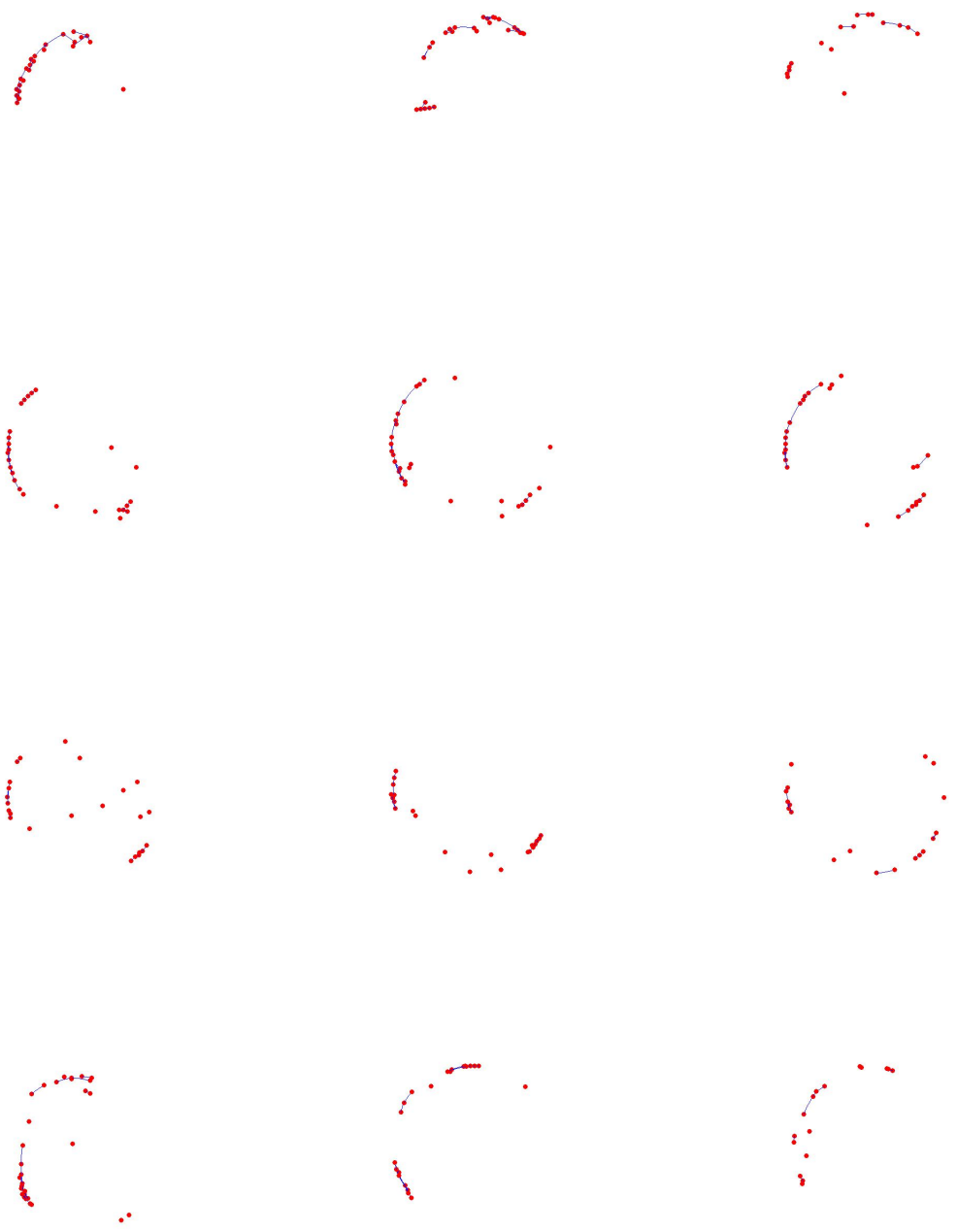


Figure 4.3: PowerGlove Gesture trajectories 3. First row: Come; Second row: Computer; Third row: Cost; Fourth row: Crazy

Figure 4.4: PowerGlove Gesture trajectories 4. First row: Danger; Second row: Deaf; Third row: Different; Fourth row: Draw

Figure 4.5: PowerGlove Gesture trajectories 5. First row: Drink; Second row: Eat; Third row: Exit; Fourth row: Forget

Figure 4.6: Microsoft Kinect Gesture trajectories 1. First row: Air Guitar; Second row: Archery; Third row: Baseball; Fourth row: Boxing

Figure 4.7: Microsoft Kinect Gesture trajectories 2. First row: Celebration; Second row: Chicken; Third row: Clapping; Fourth row: Crying

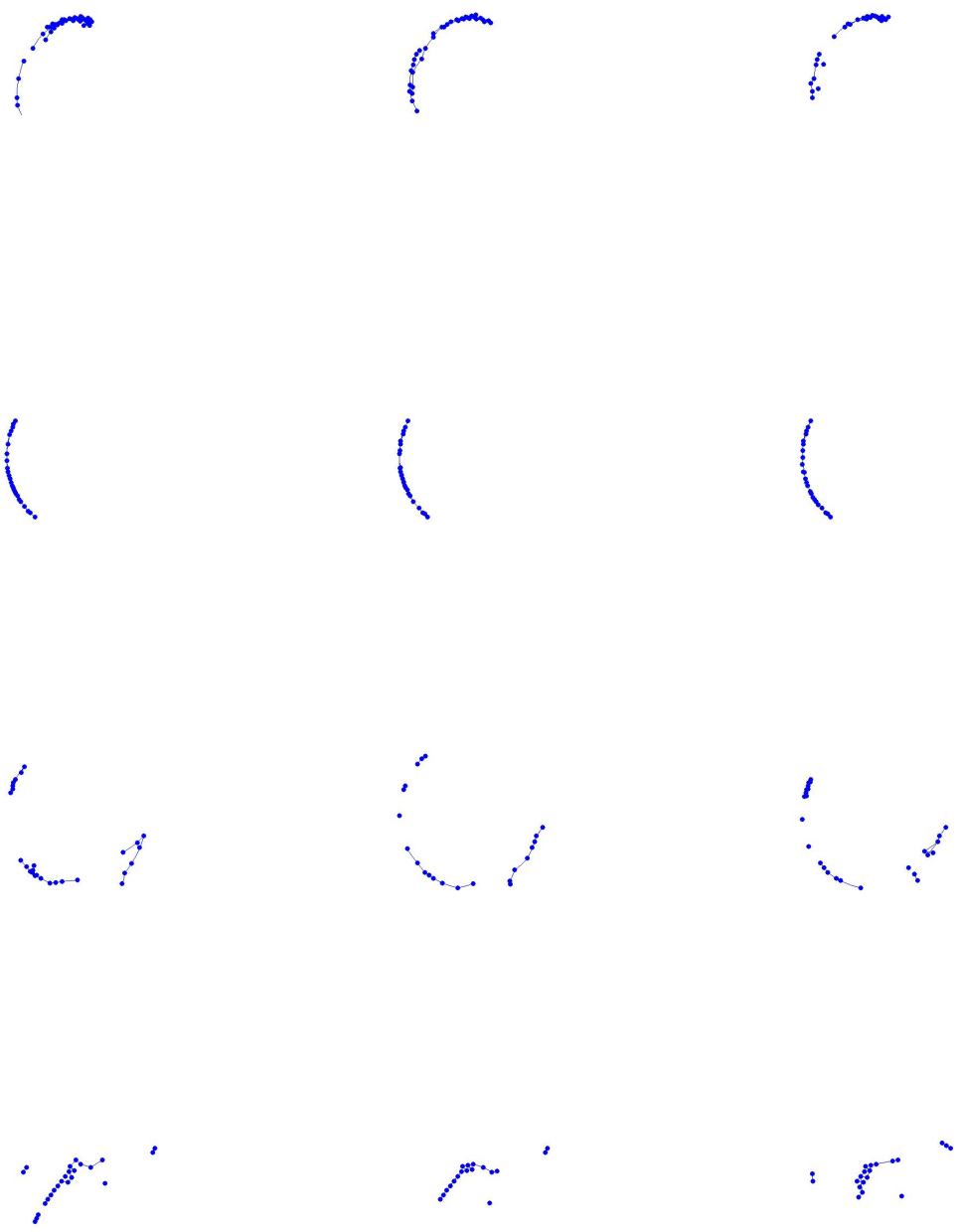


Figure 4.8: Microsoft Kinect Gesture trajectories 3. First row: Driving; Second row: Elephant; Third row: Football; Fourth row: Heart Attack

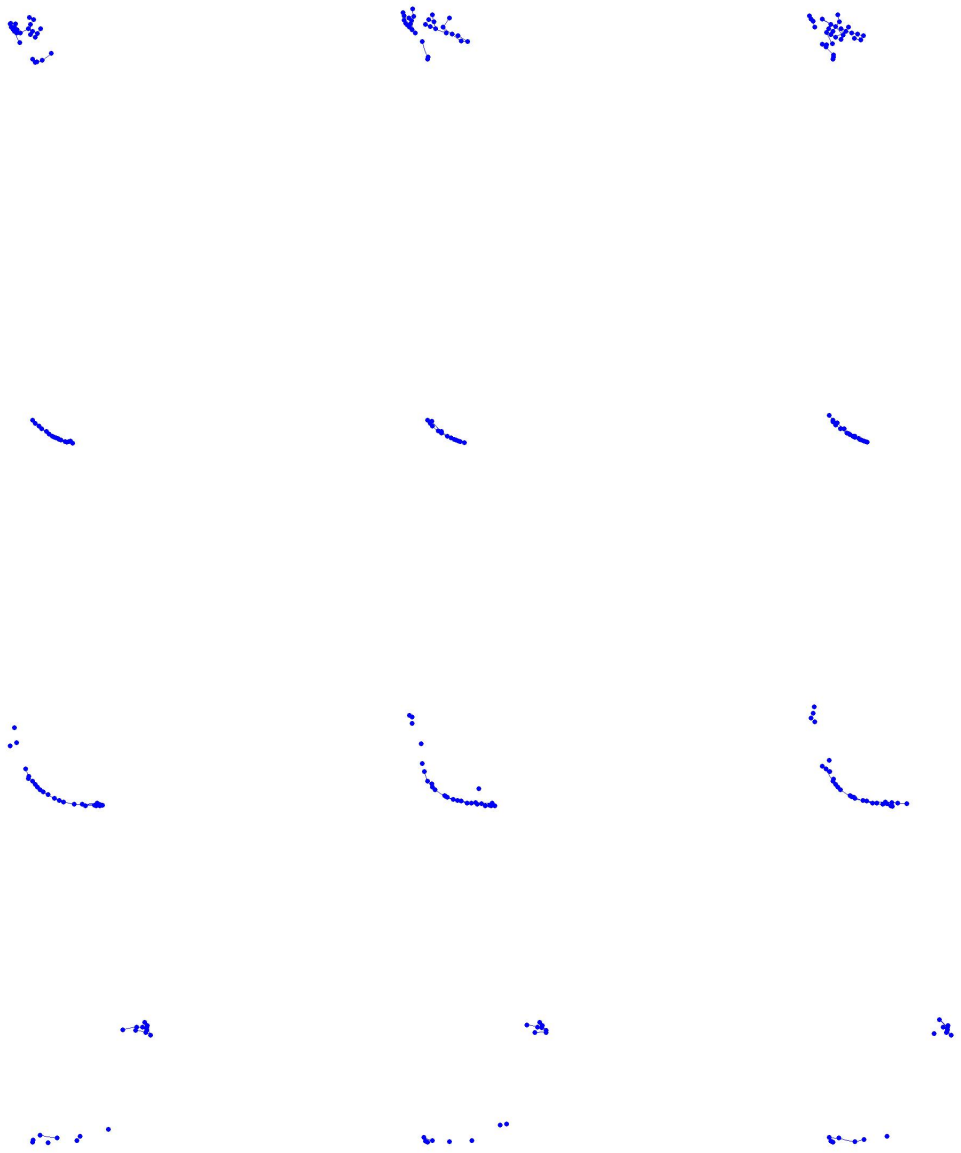


Figure 4.9: Microsoft Kinect Gesture trajectories 4. First row: Laughing; Second row: Monkey; Third row: Skip Rope; Fourth row: Sleeping

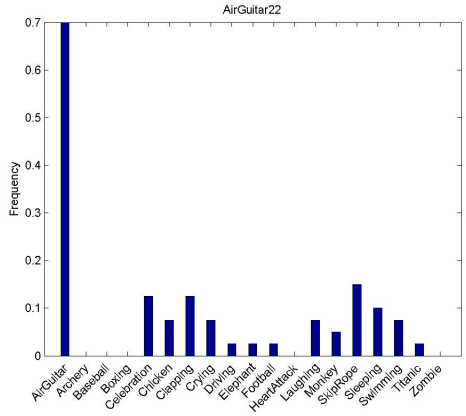
Figure 4.10: Microsoft Kinect Gesture trajectories 5. First row: Swimming; Second row: Titanic; Third row: Zombie

crosoft Kinect dataset and 35 from Nintendo PowerGlove, but only a few graphs are shown as an example. The figures represent a histogram of the BMUs hit (postures) frequency from various gestures. The gestures with higher values are the winning gestures. The label of each graph represents the true gesture classification, so if that specific gesture has a higher value among other gestures in the graph, than the gesture is considered to be recognized correctly. Otherwise the gesture is not registered correctly. For instance in Fig. 4.14a the correct gesture is Air Guitar and it is evident from the graph that the gesture "Air Guitar" has the highest value, therefore is registered correctly. As mentioned earlier, some postures are repeated in different gestures, for that reason misclassification occurs. An example of this is Fig. 4.14f, where the gesture has been misclassified to a gesture "Crying" although the true gesture is "Celebration".

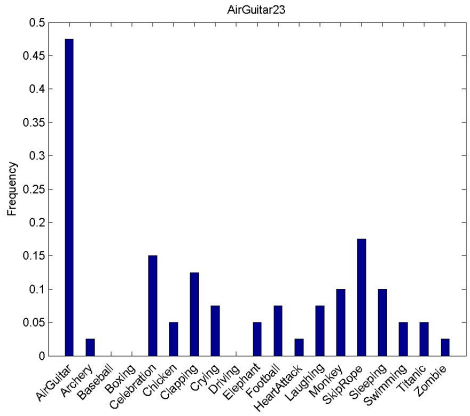
In Fig. 4.12, representing the Nintendo PowerGlove dataset, it is evident that the graph contains more noise than in Kinect data, therefore more misclassifications occur and the correct gesture recognition rate is lower. The gesture recognition rate is shown in Tables 4.1 and 4.2. In brackets is shown the number of correctly classified gestures: the values are out of 25 and 35 for Microsoft Kinect data and Nintendo PowerGlove data accordingly. It is evident that the recognition rate of the Nintendo Powerglove is lower if comparing to Kinect rate for reasons explained earlier. The next section explains another gesture recognition method which somewhat increases the accuracy of classified gestures.

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Air Guitar	80(20)	Clapping	52(13)	Laughing	80(20)
Archery	72(18)	Crying	56(14)	Monkey	84(21)
Baseball	76(19)	Driving	68(17)	Skip Rope	64(16)
Boxing	88(22)	Elephant	64(16)	Sleeping	72(18)
Celebration	84(21)	Football	68(17)	Swimming	96(24)
Chicken	48(12)	Heart Attack	80(20)	Titanic	52(13)
Zombie	60(15)				

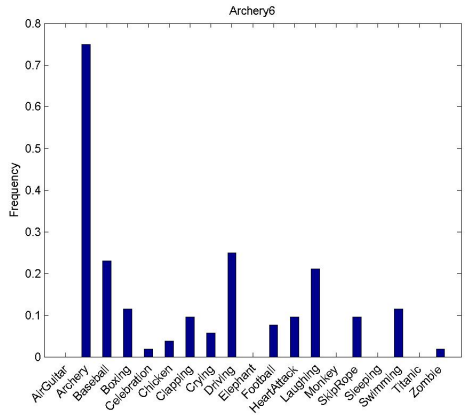
Table 4.1: Recognition rate for Kinect dataset: Approach #1



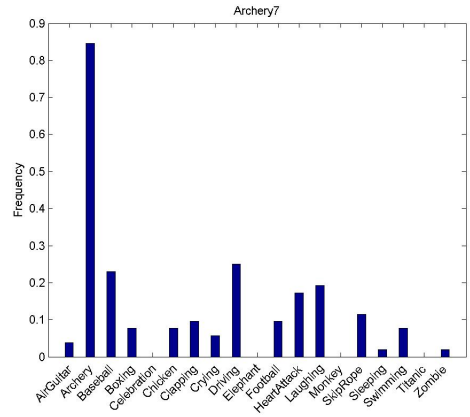
(a) AirGuitar 22



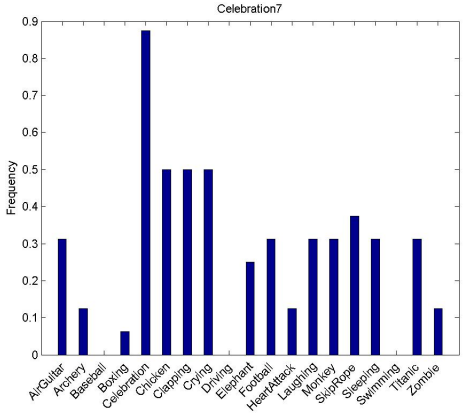
(b) AirGuitar 23



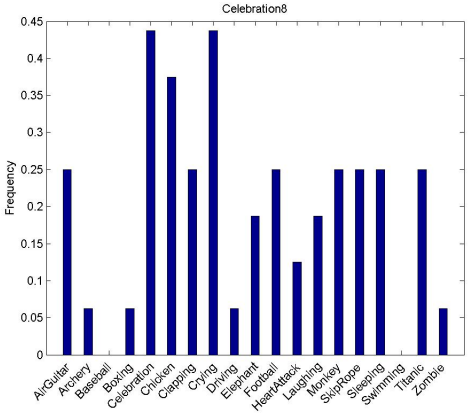
(c) Archery 6



(d) Archery 7

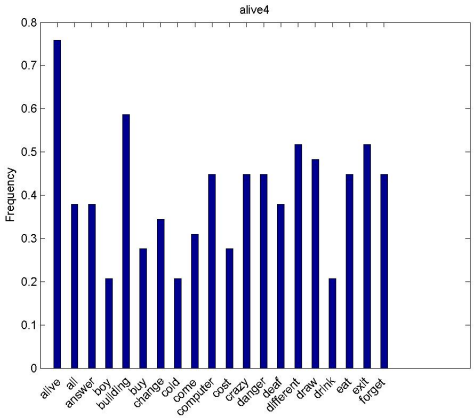


(e) Celebration 7

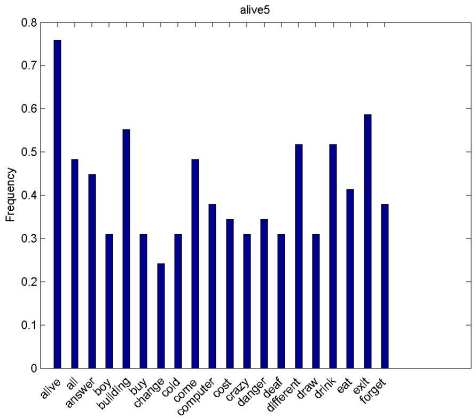


(f) Celebration 8

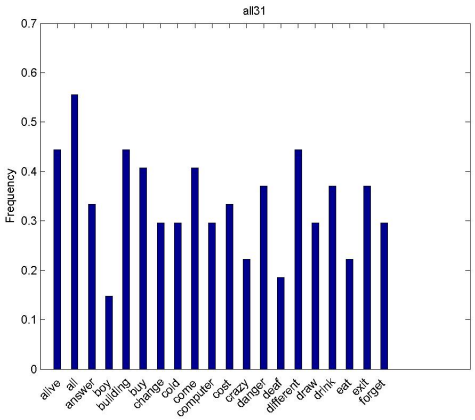
Figure 4.11: Gesture Recognition: Using all postures - Microsoft Kinect Dataset



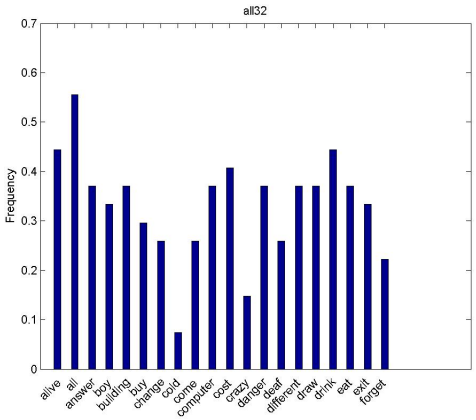
(a) Alive 4



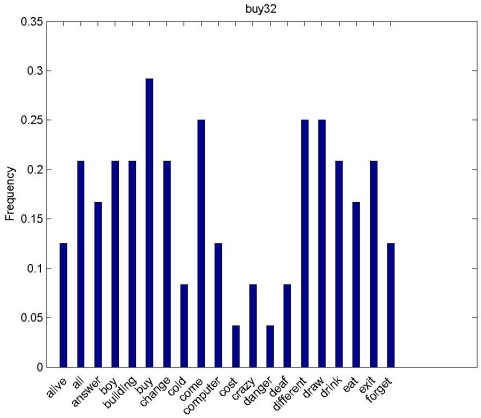
(b) Alive 5



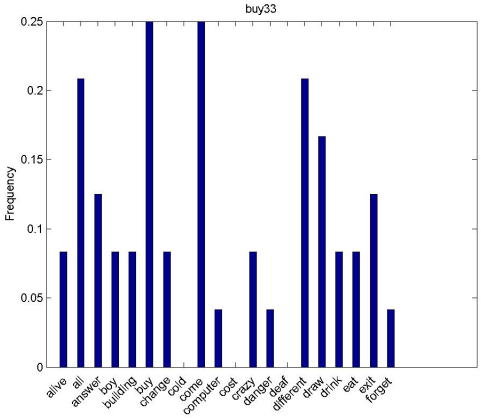
(c) All 31



(d) All 32



(e) Buy 32



(f) Buy 33

Figure 4.12: Gesture Recognition: Using all postures - Nintendo PowerGlove Dataset

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Alive	48.6(17)	Cold	34.3(12)	Different	45.7(16)
All	65.7(23)	Come	20(7)	Draw	60(21)
Answer	20(7)	Computer	14.3(5)	Drink	25.7(9)
Boy	42.9(15)	Cost	22.9(8)	Eat	14.3(5)
Building	54.3(19)	Crazy	54.3(19)	Exit	20(7)
Buy	22.9(8)	Danger	28.6(10)	Forget	28.6(10)
Change	60(21)	Deaf	37.1(13)		

Table 4.2: Recognition rate for PowerGlove dataset: All postures

4.2.2 Approach #2: gesture recognition using weighted aggregation of all postures

The results of this approach are seen in Tables 4.3 and 4.4. The overall results for correct gesture classification have gone up if comparing to the first approach where no weights were used. Another approach to gesture recognition will follow, which focuses on the dynamic nature of the gesture trajectories.

4.2.3 Approach #3: gesture recognition using posture transitions

Figures 4.13 and 4.14 show some classification results based on the given approach. If compared to Figures 4.11 and 4.12 the winning gestures appear to be more dominant. For instance in Figure 4.14a and 4.14b for gestures "Alive" and "Buy" the value of the cor-

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Air Guitar	100(25)	Clapping	92(23)	Laughing	100(25)
Archery	100(25)	Crying	88(22)	Monkey	92(23)
Baseball	96(24)	Driving	100(25)	Skip Rope	80(20)
Boxing	88(22)	Elephant	96(24)	Sleeping	28(7)
Celebration	100(25)	Football	100(25)	Swimming	100(25)
Chicken	48(12)	Heart Attack	84(21)	Titanic	96(24)
Zombie	100(25)				

Table 4.3: Recognition rate for Kinect dataset: Weighted aggregation of all postures

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Alive	54.3(19)	Cold	60(21)	Different	57.1(20)
All	74.3(26)	Come	34.3(12)	Draw	37.1(13)
Answer	42.9(15)	Computer	51.4(18)	Drink	11.4(4)
Boy	48.6(17)	Cost	40(14)	Eat	40(14)
Building	25.7(9)	Crazy	54.3(19)	Exit	17.1(6)
Buy	65.7(23)	Danger	22.9(8)	Forget	88.6(31)
Change	45.7(16)	Deaf	25.7(9)		

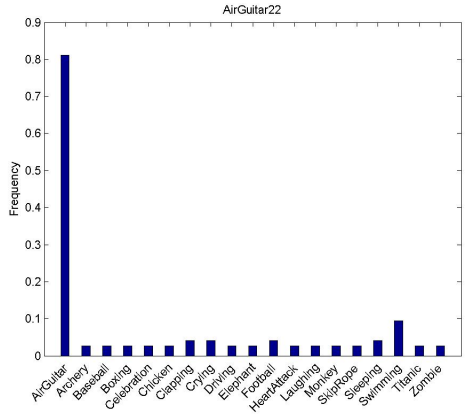
Table 4.4: Recognition rate for PowerGlove dataset: Weighted aggregation of all postures

responding gesture are much higher when comparing to other gestures. However, some misclassification still occurs as seen in Fig. 4.14c for the gesture "All". These results will be improved by taking into account the frequency of the posture transitions discussed in the next Section.

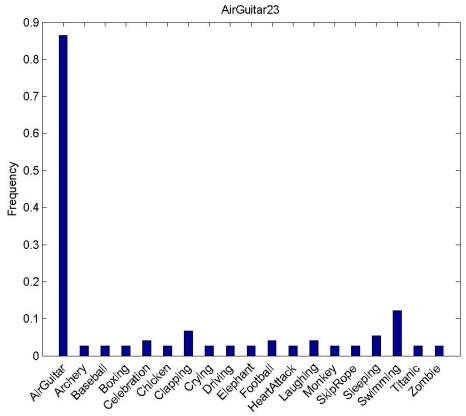
Table 4.5 and 4.6 depict the results of the gesture recognition process. It is clear the advantage of this method over the previous two. Most of the correctly classified values for the Microsoft Kinect data have gone up, only a few have gone down or stayed unchanged. For the Nintendo PowerGlove data there is a significant boost in correct gesture classification as seen in Table 4.6.

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Air Guitar	100(25)	Clapping	100(25)	Laughing	100(25)
Archery	100(25)	Crying	80(20)	Monkey	100(25)
Baseball	88(22)	Driving	100(25)	Skip Rope	80(20)
Boxing	60(15)	Elephant	44(11)	Sleeping	100(25)
Celebration	96(24)	Football	100(25)	Swimming	100(25)
Chicken	76(19)	Heart Attack	100(25)	Titanic	100(25)
Zombie	100(25)				

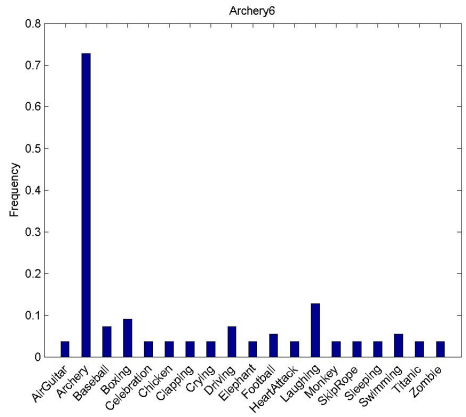
Table 4.5: Recognition rate for Kinect dataset: Using posture transitions



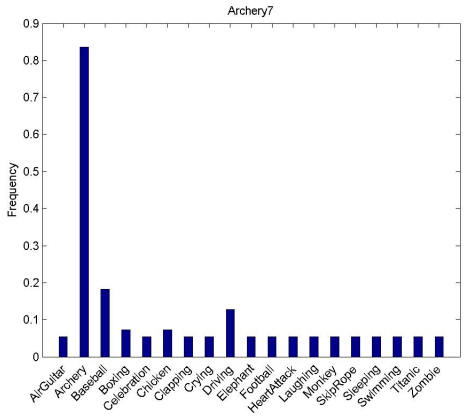
(a) AirGuitar 22



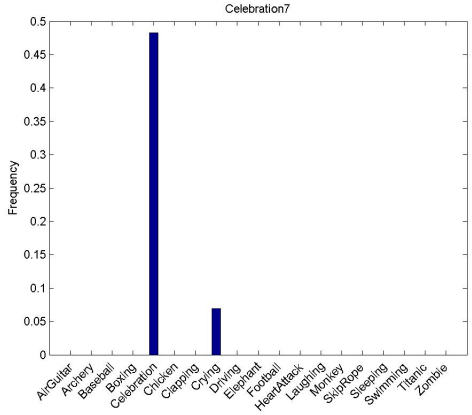
(b) AirGuitar 23



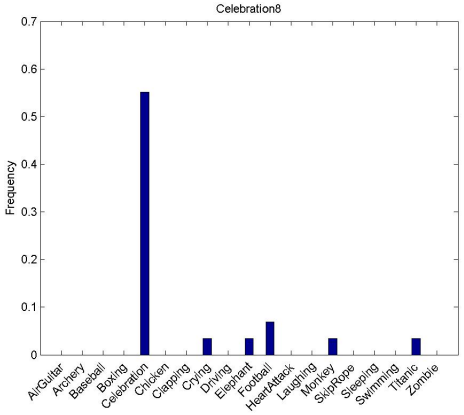
(c) Archery 6



(d) Archery 7

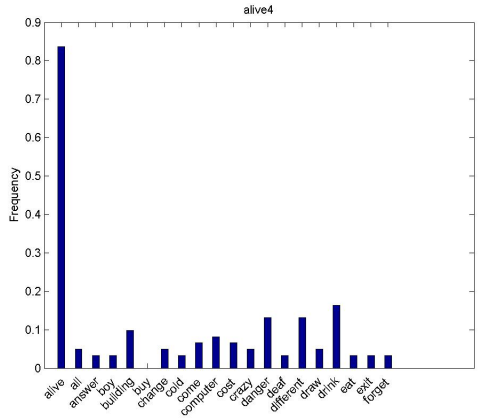


(e) Celebration 7

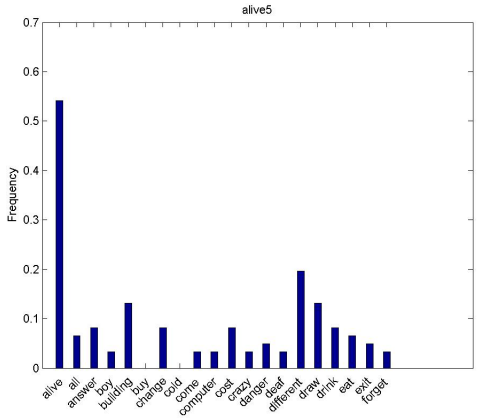


(f) Celebration 8

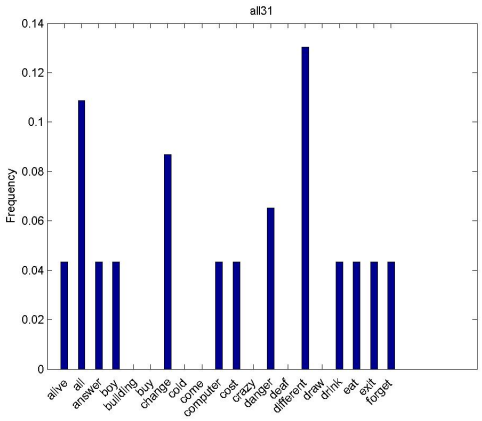
Figure 4.13: Gesture Recognition: Using posture transitions - Microsoft Kinect Dataset



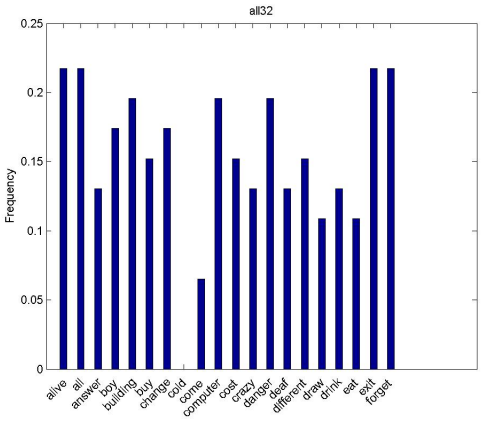
(a) Alive 4



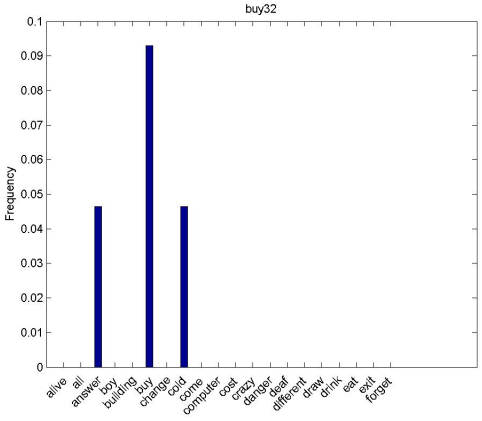
(b) Alive 5



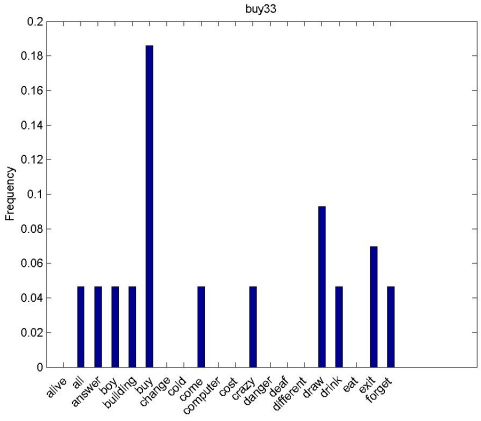
(c) All 31



(d) All 32



(e) Buy 32



(f) Buy 33

Figure 4.14: Gesture Recognition: Using posture transitions - Nintendo PowerGlove Dataset

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Alive	88.6(31)	Cold	94.3(33)	Different	88.6(31)
All	91.4(32)	Come	74.3(26)	Draw	80(28)
Answer	85.7(30)	Computer	91.4(32)	Drink	85.7(30)
Boy	88.6(31)	Cost	82.9(29)	Eat	85.7(30)
Building	91.4(32)	Crazy	82.9(29)	Exit	77.1(27)
Buy	85.7(30)	Danger	85.7(30)	Forget	80(28)
Change	85.7(30)	Deaf	80(28)		

Table 4.6: Recognition rate for PowerGlove dataset: Using posture transitions

4.2.4 Approach #4: gesture recognition using weighted aggregation of all posture transitions

Tables 4.7 and 4.8 show the classification results. Clearly this approach helped obtain the best gesture recognition result out of all approaches implemented in this thesis. The results of this approach clearly support the argument that gestures of the same class tend to trace the same path and have the same posture transitions when mapped onto the SSOM.

4.3 Comparison of results

When testing a solution to some problem it is always necessary to have a baseline for comparison of results to see the benefits of using one method against another one. In this thesis

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Air Guitar	100(25)	Clapping	100(25)	Laughing	100(25)
Archery	92(23)	Crying	100(25)	Monkey	100(25)
Baseball	100(25)	Driving	100(25)	Skip Rope	92(23)
Boxing	100(25)	Elephant	100(25)	Sleeping	100(25)
Celebration	100(25)	Football	100(25)	Swimming	96(24)
Chicken	100(25)	Heart Attack	100(25)	Titanic	88(22)
Zombie	92(23)				

Table 4.7: Recognition rate for Kinect dataset: Weighted aggregation of all posture transitions

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Alive	94.3(33)	Cold	97.1(34)	Different	88.6(31)
All	97.1(34)	Come	85.3(30)	Draw	88.6(31)
Answer	97.1(34)	Computer	94.3(33)	Drink	91.4(32)
Boy	94.3(33)	Cost	88.6(31)	Eat	88.6(31)
Building	97.1(34)	Crazy	91.4(32)	Exit	85.3(30)
Buy	85.7(30)	Danger	94.3(33)	Forget	91.4(32)
Change	94.3(33)	Deaf	91.4(32)		

Table 4.8: Recognition rate for PowerGlove dataset: Weighted aggregation of all posture transitions

two different datasets were used: Australian Sign Language (Nintendo PowerGlove) and Microsoft Kinect dataset. The Kinect data set is an artificial dataset obtained in Ryerson's University Multimedia Lab, therefore there are simply no other experiments in literature, which have used this dataset. Also, because of the use of new technologies used to obtain the Kinect gesture data, it is difficult to find a similar dataset that would have used a similar device setup. Having said that, the recognition result for the Microsoft Kinect dataset is merely evaluated based on the percentage of correct categorization of gestures. Although direct comparison of results would not be suitable in this case, Table 4.9 shows the results from this thesis related to Kinect dataset along with results of other works discussed in Section 2.4. All the values in the Table are given as an average of correct gesture classifications.

As for the Australian Sign Language, the dataset has been used previously for gesture recognition in [47], see Table 4.10. As a baseline for comparison we use a metafeatures and HMM approach. In the first one, author relies on domain knowledge and learners and classifiers, which are based on the nature of the gesture data. The disadvantage of using metafeatures, as pointed in section 1.1.2, is that domain specific knowledge is required. Rules have to be created based on the properties of the gesture data. This requires a lot of observations to be made from the gesture data. On the other hand, it is a trial and error method, where experiments need to be performed based on the set of attributes or features chosen to

Method	Recognition rate[%]
Kinect data: Approach #1	70.7
Kinect data: Approach #2	88.8
Kinect data: Approach #3	90.7
Kinect data: Approach #4	97.8
Approach in [2]	90.5
Approach in [3]	98.25
Approach in [4]	93

Table 4.9: Results comparison for Microsoft Kinect dataset

Method	Recognition rate[%]
ASL data: Approach #1	36
ASL data: Approach #2	44.9
ASL data: Approach #3	85.3
ASL data: Approach #4	92.3
Metafeatures (rule-based) [47]	94
HMM [47]	86.5

Table 4.10: Results comparison for Australian Sign Language (ASL) dataset

represent the gesture data in order to compare what features work better. Approach 4 in this thesis for example, does not impose domain specific knowledge reaching similar performance in terms of correctly classified gestures. In terms of complexity, approach 4 is much simpler with nearly the same recognition result as in [47]. Another advantage that is worth mentioning is that only 50% of the gesture data is used for training the SSOM in the four proposed approaches. Instead, author in [47] uses a 80%-20% ratio for training and testing respectively.

Similarly, HMM performance is also being used for comparison (How HMM works is discussed in section 1.1.2.). Falling behind in correct classification rate (Table 4.10), not only it performs poorly, but also possesses a number of disadvantages such as: making assumption about the data (the transition probabilities depend only on the current state), requiring a large number of parameters to be set, only positive data can be used to train, to name a few. The approaches mentioned here do not suffer from these limitations.

4.4 Summary

This chapter has introduced the results of the four approaches to gesture recognition using Spherical Self-Organizing Map given in the previous chapter. Two different body and hand gesture datasets were used to prove the benefits of using trajectories on Spherical SOMs to classify unknown gestures. The concept of unbounded mapping of data was clearly shown with an example of experiments conducted in this chapter yielding good classification result. Figures 4.15 and 4.16 show the final classification results in a chart view. The results for all the approaches discussed in this chapter are given. As expected, *Weighted aggregation of all posture transitions* performed the best among others for both datasets. Interestingly, from these figures one may notice that some gestures are classified better than other for various approaches. After some thoughts of why this is the case, it was apparent that some gestures are very similar to others in terms of the postures that they consisted of. For instance, gesture "Driving" was mistakenly classified with gesture "Zombie" in approaches 1,2, and 3, because these two gestures are very alike in their postures. Other examples may include "Chicken" vs. "Heart Attack" all for the same reasons. However, most misclassification was eliminated with the introduction of the weight factor, specifically in approach #4. Because no other experiments involving Spherical Self-Organizing Maps with application to gesture recognition similar to the ones performed here with the same datasets were found in literature, the results were evaluated merely on the percentage of correct classification. It was obtained an average result of 97.8% and 92.3% for correct classifications based Microsoft Kinect and Nintendo PowerGlove respectively. This result is considered satisfactory, although some improvements could be made, which will be discussed in the final chapter of this thesis.

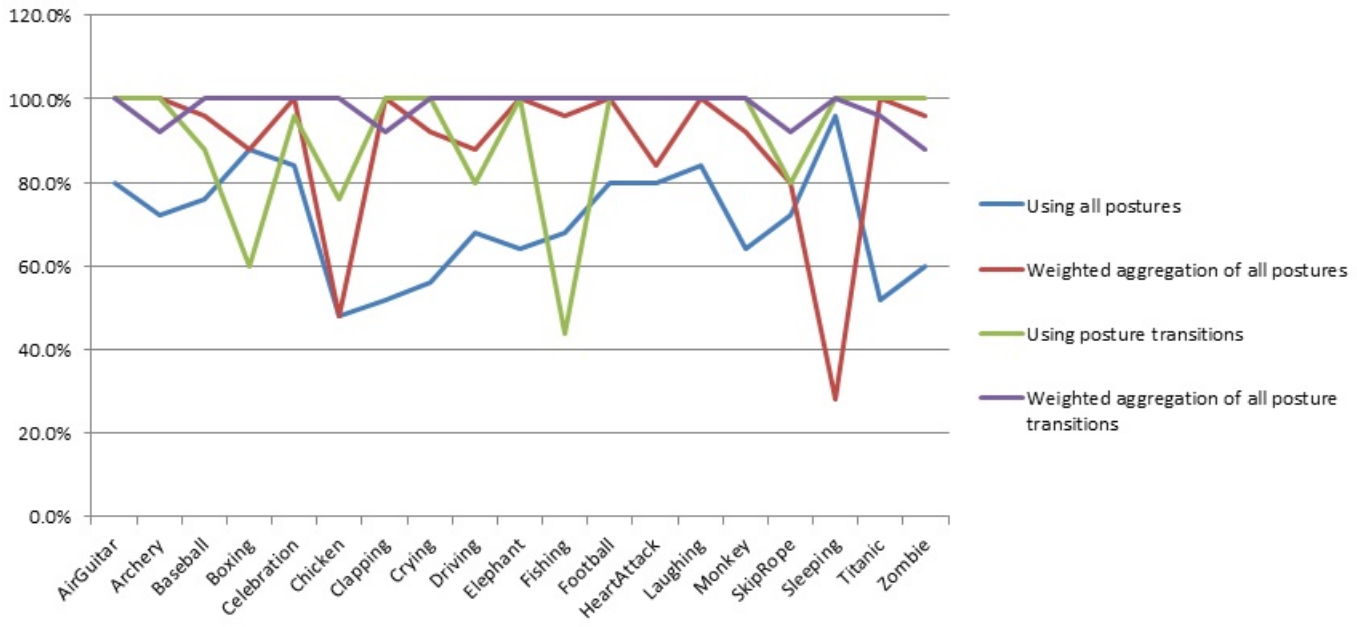


Figure 4.15: Microsoft Kinect gesture recognition comparison chart

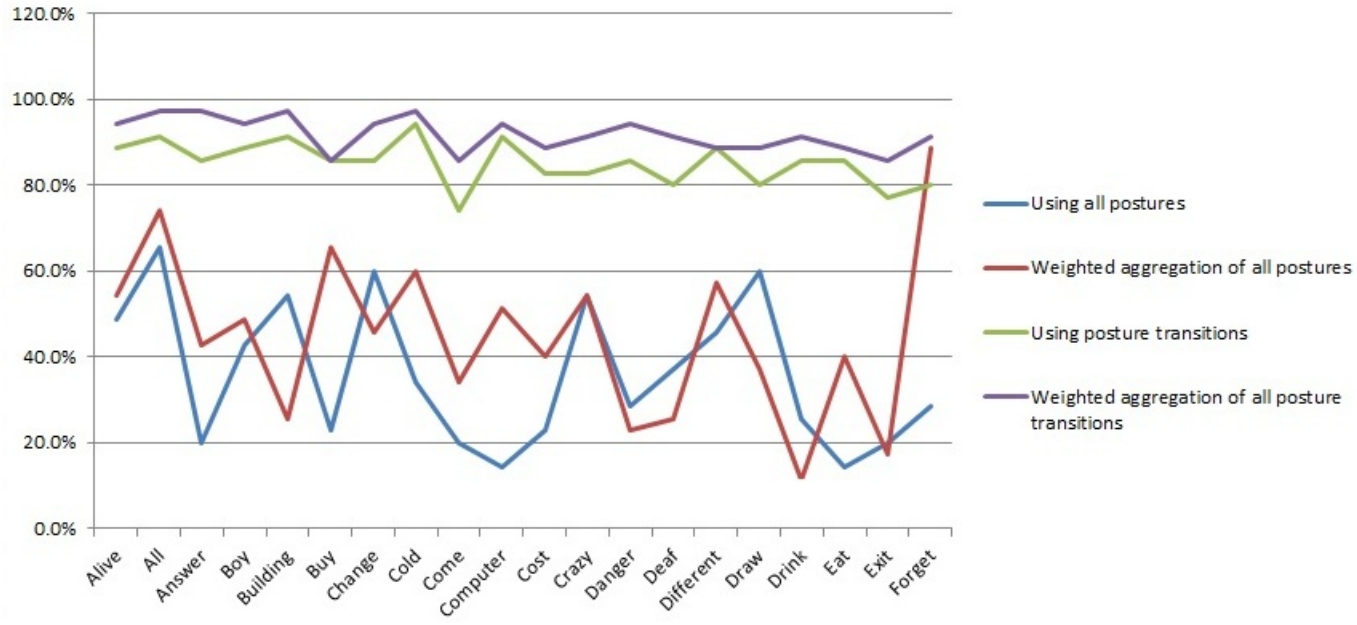


Figure 4.16: Nintendo PowerGlove gesture recognition comparison chart

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, gesture recognition was approached from the perspective of time-varying and multi-dimensional data. The differences between a conventional 2D Self-Organizing Map and a Spherical 3D Self-Organizing Map were introduced, highlighting the benefits of using the latter. Some examples of studies involving Self-Organizing Maps were brought into reader's attention with application to video scene/shot detection and summarization, and gesture recognition. In these studies, temporal trajectories were introduced as a product of data being mapped onto the SOM. Spatio-temporal data was tested such as video keyframes and gesture data represented mostly by space coordinates. An experiment was conducted to demonstrate not only the clustering properties of the Spherical SOM, but also the ability to retain the dynamic structure of the data. Examples of gesture recognition were given that used various tools along with the Self-Organizing Maps such as Dynamic Programming, Markov Models, and different variations of SOFMs such as the Hierarchical SOM. Finally, full body and hand gesture datasets were used to form temporal trajectories based on a SSOM, where posture transitions were studied and used as a classification mean for gesture recognition.

Approach 4 - *gesture recognition using weighted aggregation of all posture transitions* -

proved to be the best performing approach. There are several reasons to this:

1. Considering that the Spherical lattice of the SSOM is divided into thousands of neurons (or better say *postures*), after the SSOM training, different paths corresponding to various gestures are formed. These paths that the SSOM learns are all unique to each other in terms of postures that the path traverses. Some transitions on the lattice occur more often for some gestures than for others. If performing a given gesture (body or hand gesture), the gesturer tends to describe the same path in space, following a similar sequence of postures. This idea is taken and used on the SSOM lattice based on the ability to map the data from higher dimension to lower, while preserving the topological order. This allows the SSOM, while in the training process, to learn the common posture transitions that occur when a gesture is executed.

2. The weight introduced in this approach also boosts the correct classification rate. The weight increases the probability of an unknown gesture having similar posture transitions to a given gesture mapped on the SSOM to be classified correctly.

3. Finally, temporal trajectory information is used. As opposed to approaches 1, and 2, where just a set of postures are used to distinguish between gestures the last approach 4 uses a spatio-temporal model of classification, benefiting from the ability of the SSOM to map the feature space into equally spaced and separated locations on the sphere.

5.2 Thesis Contributions

The main contributions to this thesis are as follow:

1. Using a Spherical Self-Organizing Map to decompose gestures into well separated sparse set of postures.

   - Constrained trajectories on the sphere formed from mapping the gesture data onto the spherical lattice gives an ability to analyse the data without worrying about the size of the sphere.

- SSOM has more resolution comparing to 2D (flat) SOM (no border effect) and it is ideal for reducing high dimensional sequence of data into trajectories on the sphere.

2. Investigation of four different approaches to model gesture trajectories with application to gesture recognition.

5.3 Future Work

Considering that the main thesis contributions involve the use of SSOM trajectories in the study of gesture data, it is intuitive and natural to assume that the next step would be to look further into the dynamic connections that the trajectories have. Different approaches can still be implemented for example, by using more than two consecutive posture transitions for the classification of new gesture data. Also, more accurate and meaningful features could be used, as it is obvious from the results that the Australian Sign Language dataset yields lower classification rate compared to Microsoft Kinect dataset, therefore giving somewhat worse trajectory representation. Similar gestures also would tend to provide poorer result in classification (e.g. "Driving" vs "Zombie"), because they normally contain similar set of postures in a similar sequence. A way for distinguishing such gestures with similar set of postures needs to be addressed for future work.

When dealing with temporal trajectories the problem of non-uniform data arises. A lot of thought was put toward finding a method to represent trajectories of the same gesture class as a feature having the same length. This is because every trial of a gesture may contain a variable number of postures traced on the SSOM lattice. Although they all describe a similar path, the number of postures would always be variable. The approaches for gesture recognition here do not deal with this issue as only posture transitions are being used. However it should be possible to use a Fourier Descriptor (FD) to represent such a trajectory. Every trajectory on the SSOM consisted of some kind of a shape and FDs are one immediate way of describing a shape mathematically. Since, all the postures on the Spherical SOM are

represented as a set of coordinates with x,y,z and all the points are equally located from the radius of the sphere, these coordinates could easily be converted from Cartesian coordinate system into Spherical coordinate system by dropping the radius value r and just leaving the angles in (r, θ, ϕ). A Fourier Descriptor could be obtained and used for new gesture classification. The only issue is the variable length of the FDs for the trajectories which would give different FD values for the same gesture class. This problem with length variance in the trajectory was discussed in this thesis by A. Shimada and R. Taniguchi in [3] in Chapter 2. The authors deal with this problem by using a Hierarchical SOFM consisting of several network layers. However, the use of Fourier Descriptor to describe a SSOM trajectory would form a good feature in applications like gesture recognition and video analysis. In this case, the advantage of using a spherical SOM is that it offers a constrained spherical coordinate system on which such a descriptor can be based.

Bibliography

[1] A. P. Kshirsagar and M. N. Rathod, "Article: Artificial neural network," *IJCA Proceedings on National Conference on Recent Trends in Computing*, vol. NCRTC, pp. 12–16, May 2012.

[2] M. Oshita and T. Matsunaga, "Automatic learning of gesture recognition model using som and svm," in *Proceedings of the 6th international conference on Advances in visual computing*, vol. 6453, pp. 751-759, Springer, 2010.

[3] A. Shimada and R. I. Taniguchi, "Gesture recognition using sparse code of hierarchical SOM," in *19th International Conference on Pattern Recognition*, pp. 1–4, December 2008.

[4] G. Caridakis, C. Pateritsas, A. I. Drosopoulos, A. Stafylopatis, and S. D. Kollias, "Probabilistic video-based gesture recognition using self-organizing feature maps," in *International Conference on Artificial Neural Networks*, vol. 4669, pp. 261–270, Springer, 2007.

[5] H. Muurinen and J. T. Laaksonen, "Video segmentation and shot boundary detection using self-organizing maps," in *Scandinavian Conference on Image Analysis*, pp. 770–779, 2007.

[6] M. Koskela, M. Sjöberg, J. Laaksonen, V. Viitaniemi, and H. Muurinen, "Rushes summarization with self-organizing maps," in *Proceedings of the 1st workshop on TRECVID on Video Summarization*, pp. 45–49, ACM, 2007.

[7] T. Bärecke, E. Kijak, A. Nürnberger, and M. Detyniecki, "Summarizing video information using self-organizing maps," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'2006)*, pp. 540–546, July 2006.

[8] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Trans. Systems, Man and Cybernetics*, vol. 37, pp. 311–324, May 2007.

[9] A. Kendon, "Some relationships between body motion and speech," *A. Siegman and B. Pope, Eds., Studies in Dyadic Communication*, pp. 177–210, 1972.

[10] D. Kammer, G. Freitag, M. Keck, and M. Wacker, "Taxonomy and overview of multitouch frameworks: Architecture, scope and features," in *Workshop on Engineering Patterns for Multitouch Interfaces*, (Berlin), 2010.

[11] A. van Dam, A. S. Forsberg, D. H. Laidlaw, J. J. L. Jr., and R. M. Simpson, "Immersive VR for scientific visualization: A progress report," *IEEE Computer Graphics and Applications*, vol. 20, no. 6, pp. 26–52, 2000.

[12] A. Meng, "An introduction to markov and hidden markov models." Internet: http://www2.imm.dtu.dk/pubdb/p.php?3313, Oct. 2003. [July 25, 2012].

[13] Y. Bengio, *Neural networks for speech and sequence recognition*. International Thomson Computer Press, 1996.

[14] W.-D. Chang and J. Shin, "Dynamic positional warping: Dynamic time warping for online handwriting," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 5, pp. 967–986, 2009.

[15] M. W. Kadous and C. Sammut, "Classification of multivariate time series and structured data using constructive induction," *Machine Learning*, vol. 58, no. 2-3, pp. 179–216, 2005.

[16] T. U. K. A. Information and I. Computer Sciene University of California, "Recordings of a subset of australia sign language signs." http://www.cse.unsw.edu.au/ waleed/tml/data/, June 1999. [Feb. 2012].

[17] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs: Prentice Hall, 1988.

[18] A. E. Maren, "The need-to-know of neural networks," *Journal of Neural Network Computing*, no. Summer, pp. 57–65, 1989.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] Y. M. Cheung, "K*-means: A new generalized k-means clustering algorithm," *Pattern Recognition Letters*, vol. 24, pp. 2883–2893, Nov. 2003.

[21] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, pp. 1464–1480, Sept. 1990.

[22] S. Haykin, *Neural Networks: A Comprehensive Introduction*. Prentice Hall, 1999.

[23] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley, 1949.

[24] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, pp. 59–69, 1982.

[25] J. Lampinen and E. Oja, "Clustering properties of hierarchical self-organizing maps," *J. Mathematical Imaging and Vision*, vol. 2, pp. 261–272, Nov. 1992.

[26] G. Caridakis, K. Karpouzis, C. Pateritsas, A. I. Drosopoulos, A. Stafylopatis, and S. D. Kollias, "Hand trajectory based gesture recognition using self-organizing feature maps and markov models," in *IEEE International Conference on Multimedia and Expo*, pp. 1105–1108, 2008.

[27] J.-C. Martin, G. Caridakis, L. Devillers, K. Karpouzis, and S. Abrilian, "Manual annotation and automatic image processing of multimodal emotional behaviors: validating the annotation of TV interviews," *Personal and Ubiquitous Computing*, vol. 13, no. 1, pp. 69–76, 2009.

[28] M. H. Gross and F. Seibert, "Visualization of multidimensional image data sets using a neural network," *The Visual Computer*, vol. 10, pp. 145–159, Dec. 1993.

[29] H. Ritter, "Self-organizing maps on non-euclidean spaces," in *Kohonen Maps. E. Oja and S. Kaski eds.*, Amsterdam: Elsevier, 1999, pp. 97-110.

[30] A. K. Krishnamurthy, S. C. Ahalt, D. E. Melton, and P. Chen, "Neural networks for vector quantization of speech and images," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 8, pp. 1449–1457, 1997.

[31] D. B. M. M. V. Hulle, "Comparison of flat SOM with spherical SOM: A Case Study," H. Tokutaka, M. Ohkita, K. Fujimura Eds., Japan Springer, 2007, pp. 31-41.

[32] M. Sjberg, H. Muurinen, J. Laaksonen, and M. Koskela, "Picsom experiments in trecvid 2006," in *In Proceedings of the TRECVID 2006 Workshop*, (Gaithersburg, MD, USA), Nov. 2006.

[33] P. Over, A. F. Smeaton, and P. Kelly, "The TRECVID 2007 BBC rushes summarization evaluation pilot," in *Proceedings of the international workshop on TRECVID video summarization*, (New York, NY, USA), pp. 1–15, ACM, 2007.

[34] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and trecvid," in *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, (New York, NY, USA), pp. 321–330, ACM, 2006.

[35] A. Nurnberger and M. Detyniecki, "Weighted self-organizing maps: Incorporating user feedback," in *International Conference on Artificial Neural Networks and Neural In-*

formation Processing - ICAN/ICONIP 2003, in Lecture Notes in Computer Science, pp. 883–890, 2003.

[36] D. P. Papadopoulos, S. A. Chatzichristofis, and N. Papamarkos, "Video summarization using a self-growing and self-organized neural gas network," in *MIRAGE*, vol. 6930 of *Lecture Notes in Computer Science*, pp. 216–226, Springer, 2011.

[37] S. A. Chatzichristofis, K. Zagoris, Y. S. Boutalis, and N. Papamarkos, "Accurate image retrieval based on compact composite descriptors and relevance feedback information," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 24, no. 2, pp. 207–244, 2010.

[38] S. A. Chatzichristofis and Y. S. Boutalis, "Content based radiology image retrieval using a fuzzy rule based scalable composite descriptor," *Multimedia Tools Appl*, vol. 46, no. 2-3, pp. 493–519, 2010.

[39] S. A. Chatzichristofis, Y. S. Boutalis, and M. Lux, "SpCD - spatial color distribution descriptor - A fuzzy rule based compact composite descriptor appropriate for hand drawn color sketches retrieval," in *ICAART 2010 - Proceedings of the International Conference on Agents and Artificial Intelligence*, pp. 58–63, INSTICC Press, 2010.

[40] M. Marszałek, I. Laptev, and C. Schmid, "Actions in context," in *IEEE Conference on Computer Vision & Pattern Recognition*, 2009.

[41] L. G. Shapiro and G. C. Stockman, *Computer Vision*. Prentice Hall, 2001.

[42] L. Cieplinski, "MPEG-7 color descriptors and their applications," in *Proceedings of the 9th International Conference on Computer Analysis of Images and Patterns*, (London, UK, UK), pp. 11–20, Springer-Verlag, 2001.

[43] Z.-C. Huang, P. P. K. Chan, W. W. Y. Ng, and D. S. Yeung, "Content-based image retrieval using color moment and gabor texture feature," in *IEEE International Conference on Machine Learning and Computing*, pp. 719–724, 2010.

[44] P. Wu, Y. Choi, Y. M. Ro, and C. S. Won, "Mpeg-7 texture descriptors," *International Journal of Image and Graphics*, vol. 1, pp. 547–563, Oct. 2001.

[45] H. Yu, M. Li, H. Zhang, and J. Feng, "Color texture moments for content-based image retrieval," in *IEEE International Conference on Image Processing*, vol. 3, pp. 929–932, 2002.

[46] C. S. Won, D. K. Park, and S.-J. Park, "Efficient use of MPEG-7 edge histogram descriptor," *ETRI Journal*, vol. 24, pp. 23–30, Feb. 2002.

[47] M. W. Kadous, *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. PhD thesis, The University of New South Wales, 2002.