3D CONTINUOUS REAL-TIME ARABIC SIGN LANGUAGE RECOGNITION

A Thesis Submitted to the Department of Scientific Computing, Faculty of Computer & Information sciences Ain Shams University, in the Partial Fulfillment of the requirements for PHD Degree of Computer and Information Sciences

BY

AHMED SAMIR ELONS HUSSEIN
Master. Degree in Scientific Computing Department,
Faculty of Computer & Information sciences
Ain Shams University.

SUPERVISED BY

Prof. Dr. MOHAMMED FAHMY TOLBA
Professor, Scientific Computing Department,
and former dean of Faculty of Computer & Information sciences,
Ain Shams University.

Prof. Dr. MAGDY ABOUL-ELA
Professor, Sadat Academy
for Management Sciences Maadi, Cairo, Egypt

(2012)
Abstract

This thesis aims to research and develop recognition engine for Arabic Sign Language (ASL) at a level of detail necessary for recognizing signing. The translation model depends on a video tracking system which contains Arabic signs which are translated to text in real-time response.

Automated translation systems for sign languages are important in a world that is showing a continuously increasing interest in removing barriers faced by physically challenged individuals in communicating and contributing to the society and the workforce. These systems can greatly facilitate the communication between the vocal and the non-vocal communities. For the hearing-impaired, such systems can serve as the equivalent of speech-recognition systems used by speaking people to interact with machines in a more natural way. The Arabic sign language (ASL) has some characteristics which makes the translating system is very complex. First the (ASL) is a descriptive language, that the signer should describe the word to express the meaning. Secondly the right and left hands can be used interchangeably to express the same meaning. Thirdly a lot of gestures depend on the face expressions which are out of the scope of this work. ASL has about 160 postures do not depend on the face. The first problem is how to acquire a good hand posture and motion description from a video signal. In this work, the videos are acquired using a camera, with 24 bits/pixel color resolution and 160 X 120 pixel image size 5frames/sec.

The research begins with exploring the facilities of different feature extraction methods. Many feature generation methods have been developed using Pulse-Coupled Neural Network (PCNN). Most of these methods succeeded to achieve the invariance against object translation, rotation and scaling but could not neutralize the bright background effect and non-uniform light on the quality of the generated features. To overcome the shortcomings, the research proposes a new method to enhance the features quality. The "Continuity Factor" is defined and considered as a weight factor of the current pulse in signature generation process. This factor measures the simultaneous firing strength for connected pixels. Some signs could not be classified because a single view is not enough so, 3D model was constructed using multiple views was suggested and implemented. A novel technique to deal with pose variations in 3D object recognition is proposed. This technique uses Pulse-Coupled Neural Network (PCNN) for
image features generation from two different viewing angles. These signatures qualities are then evaluated, using a proposed fitness function. The features evaluation step is taken before any classification steps are performed. The evaluation results dynamic weighting factors for each camera express the features quality from the current viewing angles. The proposed technique uses the two 2D image features and their corresponding calculated weighting factors to construct optimized quality 3D features.

Then, Gesture- Reconstruction Module is applied. This module recognizes the continuous gestures and translates them to Arabic language. Graph matching technique was applied. A decision tree-based sub-graph isomorphism algorithm was customized and implemented.

Finally, a post processing module based on Natural Language Processing (NLP) rules is proposed to detect and correct expected errors resulting from recognition system. Popular applications for example, Optical Character Recognition (OCR), handwritten recognition, speech recognition, etc… have been researched to increase the accuracy of the recognition using NLP rules. But previous sign language recognition researchers have never explored this concept. We suggest a new hybrid semantic-oriented approach which can correct semantic level errors as well as lexical errors, and is more accurate for especially domain-specific sign language recognition error detection and correction. Through extensive experiments, it will be demonstrated the better performance of the proposed post processing approach.
Acknowledgment

I am deeply indebted to my main supervisor and the ex-dean of our faculty Professor. Dr Mohamed Fahmy Tolba for his great care, valuable advices, helpful guidance and providing different facilities to carry out this work.
I sincerely acknowledge Dr. Magdy Aboul-Ela for the supervision of this work, continuous advices, guidance and the great help in the interpretation of the results.
Thanks also are extended to all my colleagues in the scientific department. Finally I would like to express my deepest gratitude and appreciation to all my family members specially my parents for their help and support.

Ahmed Samir El-ons.
Table of Contents

1- Introduction
   1.1 Definitions 12
   1.2 Problem Statement 12
   1.3 Aim 15
   1.4 Motivations & Objective 16
   1.5 Contribution 16
   1.6 The Proposed Scenario and Scope 17
   1.7 Thesis Outline 18

2- Literature Review
   2.1 Gesture Modeling 22
      2.1.1 Two Dimensional Modeling 22
      2.1.2 Three Dimensional Modeling 23
   2.2 Data Specification and Analysis. 24
   2.3 New Trends in Sign Language Recognition 25
      2.3.1 Sign Language Recognition and recent Sensors 25
      2.3.2 Sign Language Recognition and Real-Time Response 27
      2.3.3 Previous System Problems 28
   2.4 System Specification and Analysis 28
   2.5 Research Methodology and Techniques 31

3- Proposed 2D Hand Postures Recognition 34
   3.1 Pulse-Coupled Neural Network (PCNN) 34
      3.1.1 Neural Networks for Image Processing 34
      3.1.2 PCNN Models 37
      3.1.3 Feature Extraction Methods 39
   3.2 The Proposed Feature Generation Approach 40
      3.2.1 Continuity Factor and the Proposed Approach 40
      3.2.2 The Proposed Approach 41
   3.3 Proposed ASL Recognition Architecture 43
   3.4 Implementation and Experimental Results 44

4- Proposed 3D Hand Signs Recognition 47
   4.1 3D Object Recognition 47
      4.1.1 Viewer-Centered Approach 48
      4.1.2 Object-centered Approach 49
   4.2 Hands 3D Model Construction 50
   4.3 Proposed 3D Signature Generation Approach 52
      4.3.1 PCNN Signature Quality Measurement 53
      4.3.2 PCNN Signature Quality Optimization 52
   4.4 Arabic Postures and Gestures Recognition Architecture 58
      4.4.1 3D Posture Classification 59
      4.4.2 Dynamic gesture Classification 59
4.5-Implementation and Experimental Results

5- Graph Matching & Gesture Re-Construction
5.1 Graph Matching Problem
5.1.1 Basic Notation and Terminology
5.1.2 Complexity of Graph Matching
5.2 Graph & Sub-graph Isomorphism Algorithms
5.3 A Brief Review of Ullman's Algorithm
5.4 The Proposed Sub-graph Isomorphism by Means of Decision Trees
5.5 A More Efficient Representation of Decision Trees
5.6 Traversal of Decision Tree
5.7 ASL Gestures and Graph Matching
5.8 Graph Construction & Tree Traversal Algorithm Customization
5.8.1 Graph Construction
5.8.2 Traversal Algorithm Customization
5.9 Experimental Results

6- Proposed Recognition Error Detection and Correction
6.1 NLP in Recognition Systems
6.2 A Proposed Error detection Semantic Oriented Approach
6.2.1 Lexico-Semantic Pattern
6.2.2 Domain Knowledge Construction
6.3 A Proposed Statistical Error Correction Approach
6.4 Case study and Experimental Results

7- Conclusion

8- Bibliography

List of Figures

1.1 Signs of government from three different sign languages [32] 14
1.2 The proposed work scenario 19
1.3 The waterfall model used in the proposed system 20
1.4 the proposed system use-case diagram 20

2.1 Datagloves a-Virtual Technologies Cybergloves
b-5th wireless dataglove 23

2.2 An example of an image containing depth information produced by the
3DV system’s camera. Lighter areas are closer to the camera acquisition
23

2.3 Human detection and tracking in uncontrolled environments. 26
2.4 Shows 2 different postures a-front view. b- side view. 28
2.5 colored images sample for PCNN based model 29
2.6 the number of postures (1 hand, 2 hands) against the number of gestures 29
2.7 No of samples against SNR 30
2.8 No of samples against Object Percentage 30

3.1 Ono-to-one pixel mapping of the PCNN 34
| 3.2 | One PCNN Neuron and its feeding radius | 35 |
| 3.3 | Original PCNN Neuron | 35 |
| 3.4 | A modified model for PCNN neuron Optimized PCNN | 37 |
| 3.5 | A sparse pulse, b-shows a dense pulse | 39 |
| 3.6 | Two images of the same posture, left is clean and right is distorted | 41 |
| 3.7 | Image signatures of 2 images using eq(12), eq(14) and proposed approach | 41 |
| 3.8 | the recognition system schematic diagram | 41 |
| 3.9 | The recognition accuracy against the number of features from the 3 methods | 44 |
| 3.10 | Shows 2 different postures a-front view. b- Side view. | 45 |

| 4.1 | Cameras positions in A and B | 49 |
| 4.2 | The hands 3D features generation | 50 |
| 4.3 | The same hand posture in different 2 poses | 50 |
| 4.4 | a- 2 poses for the same posture which Sum fails. b- The corresponding signatures. | 51 |
| 4.5 | a- 2 poses of the same posture which (19) fails. b- The corresponding signatures | 54 |
| 4.6 | The optimization problem for 3D pose invariant features | 55 |
| 4.7 | Common optimization methods traditionally used in computer vision | 57 |
| 4.8 | The signer and capturing cameras positions | 57 |
| 4.9 | Hand posture classification algorithm steps | 58 |
| 4.10 | one gesture represents counting from 1 to 2 and the corresponding postures | 58 |
| 4.11 | The Corresponding NFA for counting from 1 to 2 | 59 |
| 4.12 | ROC curve of the three proposed objective functions | 62 |
| 4.13 | the recognition accuracy against the object percentage | 62 |
| 4.14 | Recognition accuracy against the number of gestures | 64 |

| 5.1 | Ullman’s Algorithm | 74 |
| 5.2 | The row-column representation of the adjacency matrix | 76 |
| 5.3 | the set A(g1)of permuted adjacency matrices of g1 is listed | 79 |
| 5.4 | Decision tree for the graphs g1 in Fig and the graph g2 | 80 |
| 5.5 | the (Optimize Tree) procedure | 82 |
| 5.6 | Compact Decision tree. | 83 |
| 5.7 | The decision tree procedure | 86 |
| 5.8 | the main steps to recognize the connected gestures | 89 |
| 5.9 | illustrates a gesture expresses counting from 1 to 2 and its associated graph | 90 |
| 5.10 | the main 4 regions of the frame | 91 |
| 5.11 | 4 examples for 2 hand postures with different 2 hand posture classes and different regions and their corresponding graphs in order | 92 |
| 5.12 | the possible cases of the modified procedure output | 94 |
| 5.13 | the Modified Decision Tree Procedure | 95 |
5.14 the tree size against model graphs database size
5.15 a) Execution time in sec against the database size.
           b) Computational steps against number of tree nodes

6.1 The error detection diagram  
6.2 Schematic diagram for error correction module  
6.3 the schematic diagram of error detection and correction modules interaction

List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>comparison between conventional ANN &amp; PCNN</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>Examples of previous 3D recognition systems</td>
<td>48</td>
</tr>
<tr>
<td>4.2</td>
<td>values of w1 and w2 with different capturing angles</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>the possible combinations of w1 and w2</td>
<td>56</td>
</tr>
<tr>
<td>4.4</td>
<td>The recognition accuracy using the 3 objective functions</td>
<td>61</td>
</tr>
<tr>
<td>4.5</td>
<td>the gestures of the dataset</td>
<td>63</td>
</tr>
<tr>
<td>4.6</td>
<td>Measured times across each stage for different datasets sizes</td>
<td>65</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparative study with existing systems</td>
<td>66</td>
</tr>
<tr>
<td>5.1</td>
<td>the recognition accuracy</td>
<td>99</td>
</tr>
<tr>
<td>6.1</td>
<td>Template database sample</td>
<td>107</td>
</tr>
<tr>
<td>6.2</td>
<td>The correction detection Module Performance</td>
<td>107</td>
</tr>
<tr>
<td>6.3</td>
<td>Sample of ill-LSP correction</td>
<td>108</td>
</tr>
<tr>
<td>6.4</td>
<td>The correction detection Module Performance</td>
<td>109</td>
</tr>
<tr>
<td>6.5</td>
<td>the recognition accuracy before and after enhancement</td>
<td>109</td>
</tr>
</tbody>
</table>

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>Two Dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>Three Dimensional</td>
</tr>
<tr>
<td>ASL</td>
<td>Arabic Sign Language</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Marcov Model</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>PCNN</td>
<td>Pulse-Coupled Neural Network</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-Layer Perceptron</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>LSP</td>
<td>Lexico Semantic Pattern</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver Operating Characteristics</td>
</tr>
<tr>
<td>NFA</td>
<td>Non-deterministic Finite Automaton</td>
</tr>
</tbody>
</table>
symbols

<table>
<thead>
<tr>
<th>symbol</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L(i) )</td>
<td>input linking potential</td>
</tr>
<tr>
<td>( F(i) )</td>
<td>feeding potential</td>
</tr>
<tr>
<td>( S )</td>
<td>intensity of given image element</td>
</tr>
<tr>
<td>( U(i) )</td>
<td>the activation potential of neuron</td>
</tr>
<tr>
<td>( \theta(i) )</td>
<td>threshold potential of neuron</td>
</tr>
<tr>
<td>( \alpha_L, \alpha_F \text{ and } \alpha_q )</td>
<td>decay coefficients</td>
</tr>
<tr>
<td>( \beta )</td>
<td>linking coefficient</td>
</tr>
<tr>
<td>( \mathbf{VL and VF} )</td>
<td>coefficients of the linking and threshold potential</td>
</tr>
<tr>
<td>( Y_{\text{sur}} )</td>
<td>firing information</td>
</tr>
<tr>
<td>( Y_{\text{out}} )</td>
<td>Neuron output</td>
</tr>
<tr>
<td>( * )</td>
<td>convolution operator</td>
</tr>
<tr>
<td>( R )</td>
<td>matrix of weight coefficients</td>
</tr>
<tr>
<td>( X(i) )</td>
<td>Output quantity based on sigmoid function.</td>
</tr>
<tr>
<td>( Y(i) )</td>
<td>Output quantity based on step-function</td>
</tr>
</tbody>
</table>
List of Publications


• Ahmed Samir ,M. Aboul-Ela and M.F Tolba." Neutralizing Lighting non-homogeneity and Background Size in PCNN Image Signature for Arabic sign language recognition" in "Neural Computing and Applications international journal" DOI : 10.1007/s00521-012-0818-4. with impact factor 0.7 in 2012.

• Ahmed Samir ,M. Aboul-Ela and M.F Tolba." A Proposed PCNN Features Quality Optimization Technique for Cameras Weighting in Pose-Invariant 3D Arabic Sign Language Recognition" has been accepted and awaiting for publication in "Applied Soft Computing" journal with impact factor 2.6 in 2012

• Ahmed Samir ,M. Aboul-Ela and M.F Tolba." 3D Object Recognition Using Multiple 2D Views for Arabic Sign Language" has been published in "Journal of Experimental & Theoretical Artificial Intelligence" DOI:10.1080/0952813X.2012.680073 with impact factor 0.7 in 2012.

• Ahmed Samir ,M. Aboul-Ela and M.F Tolba " Arabic sign language continuous sentences recognition using PCNN and graph matching " Neural Computing and Applications international journal" DOI : 10.1007/s00521-012-1024-0 with impact factor 0.7 in 2012.

• Ahmed Samir ,M. Aboul-Ela and M.F Tolba "3D Arabic sign language recognition using linear combination of multiple 2D views" Informatics and Systems (INFOS), 2012 8th International Conference on Page(s):6 -13 May 2012
• Ahmed Samir, M. Aboul-Ela and M.F Tolba "A proposed graph matching technique for Arabic sign language continuous sentences recognition" Informatics and Systems (INFOS), 2012 8th International Conference on Page(s):14 -20 May 2012


• Ahmed Samir, M. Aboul-Ela and M.F Tolba "Light and Background-Independent PCNN Feature Generation Model for Arabic Sign Language Recognition" submitted in International Conference on Advanced Machine Learning Technologies and Applications (AMLTA12)

Chapter 1
Introduction

The most natural way human being communicates with other is by using voice and gestures. However, the human-machine interfaces are still very primitive, thereby forcing us to adapt to the machine requirements. The use of keyboards and/or mice is non-natural communication devices for us humans. Many researchers have dedicated efforts for years to create a more natural human-machine interface. The speech recognition is being widely researched for decades. Its fundamentals and theoretical background is well studied and many commercial products have been developed. However, the gesture recognition is still much unexplored field. This lack of research is due mainly to the very high computational load needed to process efficiently video signals. Only nowadays enough powerful processors have being developed, able to process video signals in real time and/or handle with the huge amount of information necessary to store these kinds of signals. With the advancing of the computer’s hardware, more sophisticated applications can be though, and more natural interfaces can be built. Nowadays, Virtual Reality, Remote Operation, Robot Command, and Sign Language Translation are the applications, which take more benefit of the development of gesture recognition techniques.

We can find in the Oxford - Advanced Learner’s dictionary [1] the following definition for gesture.
“Gesture - a movement of a part of the body, esp. the hand or head intended to suggest a certain meaning”[1].

From this definition we can conclude that the objective of the gesture is to allow a more complete communication, not only using voice, or when voice use becomes impossible.

1.1 Definitions

We use gestures naturally during a conversation in order to clarify or better express ideas or actions. These kinds of gestures only help the communication, being non essential to the understanding. When we think in gesture, a hand gesture is the first thing that appears in our minds; however a gesture can be performed using any part of our body. There are gestures that use mainly the head, like”yes”, ”no”, ”doubt”, although similar
ones can be done using the hands also. However, the hands still are undoubtedly the main tools used for gestural expression. The term “gesture” is usually related to motion [2] as we can see by the dictionary’s definition. Although, from a scientific point of view, the gestures can be divided into two distinctive categories:

A-Static

In this thesis, we will call the static gestures as hand postures, adopting the posture definition used by Liang:

"Posture is a specific combination of hand position, orientation, and flexion observed at some time instance” [1]

Posture or static gestures are not time varying signals, so they can be completely analyzed using only one or a set of images of the hand took in a specific time. Good examples of postures are the facial information like a smile or an angry face, and the hand postures for 'OK' or 'stop' hand signs, which a simple picture is enough for complete understanding.

B-Dynamic

We will reserve the word “gesture” to describe dynamic gestures, according Liang.

"Gesture is a sequence of postures connected by motions over a short time span.”[1]

A gesture can be thought as a sequence of postures. In a video signal the individual frames define the postures and the video sequence defines the gesture. The head 'No' and 'Yes', and hand 'goodbye' or 'come here' gestures can only be recognized taking the temporal context information, being good examples of dynamic gestures.

The gestures usually help us in the communication; however there are cases where the gestures are the only way possible to a person communicates with other. This is the case of the hear-impaired people.

The sign language is undoubtedly the most grammatically structured and complex set of human gestures. Once it is a very challenging problem, besides it has a strong social appeal, recently, many researchers have dedicated efforts to design automatic translators from sign language to text or speech [3] and vice-versa [4]. However, reliable, fast, and vocabulary complete system are not reached yet with the current technology.
Sign Languages are genuine languages, with their own grammar, and can be very different from spoken language of the country. Moreover, a Japanese signer would have so much trouble to understand a Brazilian signer as a Japanese to understand spoken Portuguese [4], once the words and grammar are completely different, see Fig 1.1.

The American Sign Language (ASL) has more than 6000 gestures and uses 26 hand postures to represent the American alphabet [5]. The Japanese Sign Language is composed by more than 8000 gestures and contains a finger spelling composed by 76 hand postures and gestures [6]. The finger spelling in sign languages is used mainly to describe names and places when a correspondent sign is not available. Not only the hands take an important place in SL recognition, head movements and facial information are very important to the understanding the sign languages as we can. Besides the facial information, the emotional expression can be added to sign by varying sign parameters such as [7]:

- speed of gesture;
- size of gesture space;
- number of repetitions or duration;
- tension of gesturing; and
- Hold-time of a posture while signing.

Fig. 1.1 signs of government from three different sign languages [7].
1.2 Problem Statement

Any 3-D object (hands in our case) may be represented as one or more images taken from different viewpoints. In most object recognition scenarios the object of interest is at a viewing distance that gives a clear view of the object as a whole with sufficient detail visible to render it distinctive. In such a scenario, the depth variation across the object of interest is usually sufficiently small in comparison to its distance from the camera that the perspective projection may be well-approximated by an affine projection. In a view based object recognition approach, or in other words, the problem of recognizing an object from a single 2-D image may then be formulated as follows:

Suppose we are given a template function \( F_0 \), a target scene image function \( I \) and Transformation \( T \) that transforms the template into \( F \).

The goal of object recognition is to minimize

\[
P = g(I,F)
\]

With respect to the transformation \( T \). \( g(\ldots) \) is a matching metric giving rise either to a dissimilarity or similarity score. If the minimum of \( P \) is smaller than or equal to some threshold \( Th \), then we can say we have a match.

The transformation \( T \) can be a combination of:

Translation-Rotation-Scaling

In order to accomplish the invariance against object pose, each single view output features are evaluated by a proposed Quality Optimization Fitness Function. The 2D features are then linearly combined to construct 3D features with maximal fitness function value. The calculated linear coefficients values change dynamically according to the object pose. The
proposed model is easy to construct and use, and is general enough to be applied across a variety of recognition problems.

1.3 Aim

The main aim of this research has been to carry out a new study on the area of object recognition via model-based approach. More specifically, we focused on examination of the linear combination of views theory and its extension to more complicated objects and, in particular hand signs in Arabic sign language.

1.4 Motivations & Objective

Since the Sign Language is the most complex and structured set of human gestures, it is the base and main subject of this project. The Arabic Sign Language has more than 9000 gestures and uses 26 hand postures and 5 dynamic gestures to represent the Arabic alphabet. The organization of Arabic Sign Language in Egypt has started in 1983. Some Arabic countries have their own sign language such as Tunis. (ASL) has some characteristics differs it from other sign languages:

- Arabic Sign Language is not so well defined until now, because the research in this field has started in the last two decades.
- In Arabic Sign Language, the two hands can be used interchangeably to express the same gesture. This property makes the same meaning may be represented with two gestures.
- Arabic Sign language is a “descriptive” language, such that, the gesture tries to describe the meaning, for instance, the gesture of “JUDGE” tries to describe the actions, which the judge does. This property not only makes the gesture too long.
- Facial expressions or human body play an important role for sign understanding.

This research mainly aims to develop techniques that can be applied to build a real-time Arabic Sign Language automatic recognition system based on multiple simultaneous views. This system recognizes continuous sentence of gestures. In this context, a gesture acquisition method based on
"Computer Vision" and an information processing stage based on Artificial Neural Networks were chosen because they are the natural way that we human beings do the task.

1.5 Contribution

The main contributions made in this thesis are that encapsulated in the following items:

- Initially examining the 2-dimensional image-based object (hand posture) recognition problem in detail, we explore different feature extraction techniques that accomplish the "transformation invariance" property. We concluded that "Pulse-Coupled Neural Network" (PCNN) can achieve most of transformations invariance like translation, rotation and scaling. But it failed to tolerate the large background size or lighting non-homogeneity because of PCNN "interference" phenomena. A new technique has been proposed and applied to PCNN feature generation process; this technique enhanced the generated feature quality in large-size background and non-uniform lighting conditions.

- A 3-D model is proposed and developed for multiple-views hand postures recognition. Pulse Coupled Neural Network is used to generate features vector for single view. Two views with different view angles are used; each view generates its features vector. The two 2D-vectors then are linearly combined into one 3D vector.

- A novel technique is developed to deal with pose variations. This technique uses Pulse-Coupled Neural Network (PCNN) for image features generation from two different viewing angles. These signatures qualities are then evaluated using a proposed fitness function. The features evaluation step is done before any classification steps are performed. The evaluation results dynamic weighting factors for each camera which express the features clearness from the current viewing angles. The proposed technique uses the two 2D image features and their corresponding calculated weighting factors to construct optimal quality 3D features.

- Proposing and developing Arabic Sign Language recognition post processing approach based on sign language knowledge and
candidate distance information given by recognition system engine. Given an output sentence from a sign language recognition system to be verified and corrected, the system retrieves word candidates from the lexicon. The retrieved candidates are ranked by the conditional probability of matches given word confusion probability.

1.6 The Proposed Scenario and Scope
The Arabic Sign Language has more than 9000 gestures. Most of gestures are a combination of 158 postures, 88 single hands (44 left hands, 44 right hands) and 70 postures use 2 hands.

The scope of this work is:
- Classification for all 158 postures with a single view.
- Classification for all 158 postures with multiple synchronized views.
- Translation for 150 hand gestures.

Fig 1.2 illustrates the proposed scenario for recognition of sentence-level gestures flow. Starting from the input video that comes from the deaf and dumb signer until an Arabic sentence is output to the normal person.
In order to build our system and achieve the above mentioned technical objectives, we use an adapted version of the waterfall model used to design and implement software. The waterfall model is shown in Fig 1.3
Figure 1.4 illustrates the main use-cases for the developed system.

Fig 1.4: the proposed system use-case diagram
1.7 Thesis Outline
The thesis is organized into seven chapters and one appendix. Chapter 1 is an introductory chapter that defines the problem of sign language recognition and introduces the motivation and contribution of this work. Chapter 2 intends to give a brief idea of the history of gesture recognition and sign language recognition, reviewing the most important techniques, used input devices and published related literature. Also this chapter presents the scientific background needed to understand the proposed approaches. The proposed model for static 2D hand postures classification is presented in chapter 3. Chapter 4 describes the 3D hand model construction from multiple 2D views, also describes the proposed feature optimization technique and how it can be applied to accomplish pose-invariance property. Chapter 5 illustrates the dynamic gesture reconstruction module using best match algorithm, chapter 6 describes the post-processing module that can be used to enhance the translation accuracy using Arabic sign language knowledge and candidate distance information given by recognition system engine. Chapter 7 concludes the thesis, and proposes problems for future investigation.
Chapter 2

Literature Review

In this chapter we review the literature of gesture recognition systems. We discuss the taxonomy of gestures to set the context of gestures in Arabic Sign Language. The previous research trials in gesture recognition will be illustrated for the purpose of comparative study conducting. A comparison between three-dimensional (3D) and two-dimensional (2D) modeling will be given using different input devices.

2.1 Gesture Modeling

Gestures encapsulate a large family of body movements that express ideas or meaning [7, 8]. Gestures can be categorized as follows:

- **Manipulative gestures** (e.g. picking up a ball) are used to move objects and convey no direct meaning in conversation [9], and
- **Communicative gestures** (e.g. waving goodbye) are used to express meaning and intent in conversation [9, 10].

2.1.1. Two Dimensional Modeling

2D modeling is perspective based, that is, where 2D image data is captured from a single camera’s point of view. Image segmentation and manipulation algorithms are used to extract information from the image. This information is used to classify the gesture. Grobel and Assam achieve a classification rate of 91.3% by extracting features from a video of signers wearing colored gloves [11]. The colored gloves made segmenting and extracting the hands’ position and shape more robust. 2D modeling techniques rely on computer vision algorithms to extract information of a gesture, rather than using specialized equipment. Computer vision based techniques use a camera and image manipulation algorithms to interpret gestures. This provides a more natural way of interactions between the system and user [12]. The process of finding and analyzing hand postures in cluttered images is extremely complex and troublesome. This has led to the development of methods involving wearing of colored markers or colored gloves on the hands and restricting the background of the video. The wearing of extra equipment and
restricting the background of the video are widely acknowledged limitations of computer vision based techniques.

2.1.2 Three-Dimensional Modeling

3D modeling entails capturing a gesture or sign in a three-dimensional space [13]. Data gloves have been widely used to track the 3D movement of hands, Fig 1.2. It uses an array of sensors fitted to gloves that the user wears. The sensors are able to record information such as hand shape, hand orientation, hand global position and hand velocity. A pinch glove was used by Kim and Waldron to obtain a sequence of 3D positions of a hand’s trajectory [14]. The sequential data obtained from the data gloves is used to classify the gesture. Kim and Waldron were able to achieve a GR accuracy of 86% using the pinch glove. 3DV systems have developed an image sensor which is capable of producing RGBD signals, where R stands for red, G for green, B for blue and D stands for the distance of each pixel relative to the camera’s position [14]. This makes it possible to track hand movements in 3D without the signer having to wear data-gloves.

![Datagloves](a-Virtual Technologies Cybergloves b-5Th wirless datagloes)

While still under development 3DV systems let Fujimura and Liu develop a GR system for Japanese Sign Language (JSL) [15]. Fujimura and Liu use the information captured by 3DV system’s image sensor to classify JSL. Depth information is displayed in an image produced by the camera. Dark regions denote objects that are far away from the camera with lighter regions describing objects that are closer. An example of an image produced by the 3DV system’s camera is shown in Figure 2.2.
2.2 Related Work

Sign language as a kind of gestures is one of the most natural means of exchanging information for most deaf people. The aim of sign language recognition is to provide an efficient and accurate mechanism to transcribe sign language into text or speech. Earlier researchers investigate different sign language recognition mechanisms, B. Bauer and H. Hienz [16] in 2000 developed a GSL (German Sign Language) recognition system that uses colored cloth gloves in both hands. The system is based on Hidden Markov Models (HMM) with one model of each sign. A lexicon of 52 signs was collected from one signer both for training and classification. A 94% recognition percentage was achieved. N. Tanibata et al. [17] -in 2001- proposed a method of extraction of hand features and recognition of JSL (Japanese Sign Language) words. For tracking the face and hand, they could recognize 64 out of 65 words successfully by 98.4%. Chen et al [18] introduced in 2003 a system for recognizing dynamic gestures (word signs) for TSL (Taiwanese Sign Language). They used frequency domain features (Fourier Transform) plus some information from motion analysis for recognizing 20 words. The data set was collected from 20 signers but the system is person dependent. HMMs were used as the classifier. An average of 92.5% recognition rate was achieved. In 2004 and 2005, J. Zieren et al. [19, 20] presented two systems for isolated recognition: the first is for recognizing GSL, on a vocabulary of 152 signs achieving a rate of 97%, using HMM, but the rate decreases for the group of signs that contain overlaps in either hands or face and hands. Unlike previous works that concentrated on manual features only, a more recent approach in 2009 by Kelly et al. [21] also incorporates a non-manual feature, namely head movement. The system relies on a single webcam and the user wearing colored gloves for continuous sign recognition. Testing of the framework consisted of 160 video clips of un-segmented sign language sentences and a
small vocabulary of eight manual signs and three head movement gestures. A detection ratio of 95.7% could be achieved. A larger ongoing project called SignSpeak is EU-funded and was introduced by Dreuw et al. [22]. It is being built on previous work, uses an ordinary 2D camera and aims at translating continuous sign language to text, supporting a large vocabulary and recognizing both manual and non-manual features [22, 23].

Compared to other sign languages, not much has been done in the automation of the Arabic sign language, except few individual attempts. M. Al-Rousan et al. [24] and O. Aljarrah et al. [25] in 2000 and 2001 respectively, developed two systems for recognizing 30 static gestures of Arabic sign language, using a collection of Adaptive Neuro-Fuzzy Inference System (ANFIS) networks for training and classification depending on spatial domain features. In 2003 Assaleh et al. [26] used colored gloves for collecting a varying size data samples for 30 manual alphabet of Arabic sign language. Polynomial classifiers were used as a new approach for classification. In a recent (2005) work in Arabic Sign Language, Mohandes et al. [27] developed a system that recognized 50 signs of words performed by one person having 10 samples per sign. They achieved a 92% recognition accuracy.

2.3 New trends in Sign Language Recognition
Recent researchers in sign language translation have many trials to make it applicable in real-world and to accomplish that; the system must enjoy the following properties:

- A relatively low cost hardware and technology should be available.
- The real-time response property.
- Guaranteeing the system translation accuracy under different environmental factors.

2.3.1 Sign Language Recognition and recent Sensors
In recent years, many different interaction types and sophisticated hardware input devices have been developed. Traditional single camera sensors have drawbacks on environmental conditions, for example the exposure latitude, lens distortion etc. There are many other vision based sensors used for hand gesture.

A stereo camera is a type of camera with two or more lenses and image sensors, allowing the camera to simulate human binocular vision which known as the stereo photography. A depth image can be calculated by the
disparity attribute of a stereo camera (Klaus, Sormann & Karner 2006 [28]). Some hand recognition methods (Lee & Hong 2010 [29]; Li et al. 2011 [30]) used the depth image from stereo camera. However the depth map from the stereo camera is difficult to achieve good quality which is inappropriate to extract good features. Multi-camera based method involves calibration of the camera. A multi-camera system is used to solve the problem of occlusion on hand tracking (Kato & Xu, 2006 [31]). The additional information from multi sensors revises the feature extracted from monocular sensor. The result tends to be more accurate with more sensors used, whereas this brings the computation burden. And the calibration usually needs to be done by the user since the multi-camera positions are uncertain, which is impractical. A GPU based parallel computing framework (Tzevanidis et al., 2010 [32]; Oikonomidis, Kyriazis, and Argyros, 2011 [33]) proposed a solution with multi cameras calibration. Single depth sensor such as the time-of-flight (TOF) camera was used on many researches (Ghobadi et al., 2007 [34]; Bergh & Gool 2010 [35]; Plagemann et al., 2010 [36]). Such sensors are very expensive and have low resolution. Combining the RGB sensors, a kind of new invented RGB-D sensor is made, for example, the Microsoft’s Kinect Fig 2.3, the PrimeSense’s PS1080, etc. The depth image from this kind of sensor has general better quality than previous sensors, and very cheap compare with the TOF camera. The core algorithm of full body tracking on the Kinect as mentioned above was proposed (Shotton et al., 2011 [37]).
When Microsoft released Kinect in November 2010, it was mainly targeted at consumers owning a Microsoft Xbox 360 console. With the slogan "You are the controller", a controller-free experience for games and entertainment was advertised, allowing the user to interact with the system using gestures and speech [38, 39].

The device itself features an RGB camera, a depth sensor and a multi-array microphone, and is capable of tracking users' body movement [40]. The interest in it was high among developers and thus, shortly after its release an unofficial open source driver was introduced, followed by many Kinect-based projects and technical demos. Since Kinect is able to track the user's full body, it seems natural to build a framework for sign language recognition. In sign languages, manual features are used along with facial expressions and different body postures in order to express words and grammatical features. Additionally, people and objects can be placed in front of a signing person and referred to during a conversation. Even though Microsoft stated that "Kinect that is shipping this [2010's] holiday will not support sign language", a demo video by the Center for Accessible Technology in Sign (CATS) shows how a few sign language sentences are recognized correctly on a system with limited vocabulary [41, 42]. The demo, however, does not support hand shape recognition, and since sign language generally features different hand-shapes, similar signs cannot be distinguished.
2.3.2 Sign Language Recognition and Real-Time Response

In sign language recognition, the real-time response of the system for the interaction task is most important providing an immediate feedback to the user. Consequently, tracking and recognition must be handled as fast as possible, preferably at the same rate as images are displayed or obtained. Otherwise, the user could be easily confused or irritated and would not know whether he or she has already initiated an action. As a result, it is important to handle all necessary calculations within a given time frame. Although modern CPUs are becoming increasingly faster, it is sometimes difficult to process all incoming and outgoing data in real-time. Many researches now explore the potentials of the current graphic processing units (GPUs). Furthermore, by now the GPU is not only utilized for graphical purposes, but also for performing highly parallel tasks efficiently. The usage of graphic cards for such a purpose is defined as general purpose computation on the GPU. Textures or in this case image data, as obtained from digital cameras or webcams, are well-suited for the computation and execution of multiple operations in parallel on the GPU.

Park et al. [43] implemented a real-time embedded FPGA-based gesture recognition system using 5DT data glove. This approach is used in order to reduce the problems of space limitations, movement limitations and lighting limitations. The architecture of the system consists of three main modules that are input module, recognition module and display module. The system recognizes the hand gesture by performing data calculations with a checksum function on the input data and compares the result to the header byte before proceeds to the matching process. The matching process compares the input hand gesture with the pre-defined hand gesture. Then, the result is displayed on the LCD screen.

Yu et al. [44] introduced an FPGA-based smart camera which is called "GestureCam". This smart camera is capable to perform simple vision-based hand gesture recognition. GestureCam comprises an image capture unit (ICU), FPGA-based gesture recognition unit (GRU), and a host and display unit (HDU). A low-pass filter is applied in order to obtain a clearer skin image from the extracted skin color image. GestureCam applies contour tracing algorithm, which is based on inner boundary tracing algorithm (IBTA) with several medications, to extract hand contour for gesture classification. GestureCam classification module is using ANN and trajectory-based methods.
2.3.3 Previous System Problems

Earlier researches in Sign Language Recognition in general suffered from 2 main problems. First, the recognition accuracy is sensitive to any changes like the signer pose, background or geometrical transformations. Second, all the existing models find difficulties in identifying some postures. This is caused by the fact that a single view for the hand does not distinguish between two different postures Fig 2.4.

Figure 2.4 shows 2 different postures a-front view. b- side view

The earlier researchers in Arabic sign language recognition faced more complicated problem concerning Arabic sign language. These issues can be listed in:

- Arabic sign language has not been standardized till very soon. This issue poorly reflects on using the Arabic sign language resources; such that they have poor credibility.
- ASL is a descriptive language by nature; the signer acts the sign behavior other than using shortened sign.
- The signer can use the 2 hands interchangeable to express the same sign. This issue doubles the data size in the recognition system.

The previous mentioned problems in previous sign language recognition in general and Arabic sign language specifically, have the great effect in choosing the proposed approach.

2.4 System Specification and Analysis

With the help from the educational ministry, 15 video films have been captured for 15 different persons. Each film contains 250 different gestures. Figure 2.5 shows a sample for the captured RGB image scenes contain the static hand postures. The experiments are done using Intel® Core™ i5-2300 (6MB Cache) and 8GB Dual Channel DDR3 at 1333MHz. also 2 cameras 5 megapixel each.
The first study is done, is to collect ASL static postures which form most of ASL words. Figure 2.6 illustrates that most of gestures are a combination of 158 postures, 88 single hands (44 left hands, 44 right hands) and 70 postures use 2 hands. Each single video contains the dynamic gestures with different signer poses, background size and signer-camera distance.

![Fig 2.5 colored images sample for PCNN based model](image1)

![Fig 2.6 the number of postures (1 hand, 2 hands) against the number of gestures](image2)

The second step is to collect images for both training and testing. The images have been captured in an uncontrolled environment which is affected by noise, non-uniform lighting and background, different capturing distance and traditional object transformations. Using two fixed cameras, two simultaneous images are captured for the same postures. It means that there exist 10 image pairs for each single posture each 10 degrees. We have 1580 image pairs for a single posture. For each image
pair, 15 exemplars were captured. It means that: we have total 23700 image pairs. Figure 2.7 illustrates the number of image pairs against Signal to Noise Ratio (SNR) assuming white Gaussian noise.

![Figure 2.7: No of samples against SNR](image)

Moreover, the object size relative to the background size inside the image varies; the term "Object Percentage" is defined as the ratio of object pixels of the whole image. This term expresses the distance between the camera and the signer assuming a fixed resolution, Fig 2.8 shows the object percentage distribution inside samples dataset.

The images used in the testing set were not included in the training set. The capturing angles represented in the training set were not included in the testing set of images, thus ensuring a pose-independent classification of postures.

![Figure 2.8: No of samples against Object Percentage](image)
2.5 Research Methodology and Techniques
The research methodology is based on dividing the dynamic sentences of gestures to elementary static postures, classify these limited number postures and then recollect the classified postures to construct dynamic sentences.
The proposed methodology offers the extendibility privilege; the words (gestures) can be appended offline to the database without re-training to the model each time.
To develop such person, distance and environment independent system that works for continuous sign language in real-time, some techniques were picked and modified to achieve the required criteria.

- **Pulse-Coupled Neural Network (PCNN)**
  This work emphasizes the usage of PCNN as image features generation method.

  Why PCNN for feature generation
  ✓ Contrary to other neural networks, it does not require training.
  ✓ It has the ability to extract image features without preprocessing steps, such as filtering or segmentation.
  ✓ It generates 1-D image signature which can identify the image.
  This signature is invariant to standard image transformations, such as rotation and scaling.
  ✓ The generated signature is easy to compute and adds no performance cost to the classification process.
  ✓ The image signature is invariant to background even in an uncontrolled environment.

- **Multi-Layer Perceptron Neural Network (MLP)**
The most popular network used in classification problems is "Multi-Layer Perceptron" (MLP). This network has many benefits: it is a general purpose model with various applications; it is capable of modeling nonlinear complex functions, good performance in noisy input and it can be adaptable for environmental changes by changing weights and/or topology. There are several training algorithms for MLP; Back propagation is used to train the network.

- **Graph Matching**

Graphs are a general and powerful data structure useful for the representation of various objects and concepts. In pattern recognition and machine vision, for example, graphs are often used to represent object models, which are known a priori and store in a database, and unknown objects, which are to be recognized. Using graphs as representation formalism, the recognition problem turns into a graph matching problem. An input graph representing an unknown object is compared to the database in order to find the most similar model graph.

*Why Graphs?*

Graphs have some interesting invariance properties. For example, if a graph, which is drawn on paper, is translated, rotated or transformed into mirror image, it still the same graph in the mathematical sense. These invariance properties as well as the fact that graphs are very well suited to model complex objects in terms of parts and their relations make them very attractive for various applications. Graph representation and graph matching have been successfully applied to a large number of problems in computer vision and pattern recognition. Examples include character recognition, schematic diagram interpretation, shape analysis, image registration, and 3-D object recognition.
Chapter 3
Proposed 2D Hand Postures Recognition

Computers can perform many operations considerably faster than a human being. Yet there are many tasks where the computer falls considerably short of its human counterpart. There are numerous examples of this. Given two pictures a preschool child could easily tell the difference between a cat and a dog. Yet this same simple problem would confound computers. As with any technology, it is just as important to learn when to use a specific recognition model and explain why to choose this specific model other than other models. This chapter begins with an overview of neural network architecture, and how a Pulse Coupled Neural Network (PCNN) is constructed. Next you will be show how PCNN is customized to recognize hands postures. Ultimately the trained neural network training must be validated.

This chapter also discusses the mathematical model of PCNN and its architecture. Also, it gives a broad overview of PCNN application in image filtering, smoothing, segmentation and features extraction. Then it shows the proposed 2D hands postures recognition model and its experimental results.

3.1 Pulse-Coupled Neural Network (PCNN)
The PCNN has similarities to the biological vision system, which extracts various features in parallel by different centre and then fuses them at the stage of conscience [19]. PCNN is an image transform that removes “unimportant” details while improving the overall quality of the image. It also provides substantial noise smoothing without losing image pattern and shape information [45]. The applications then include image “cleanup” for further processing by an algorithm of one’s choice, and also image compression since there are far fewer intensity levels to deal with than in the original image. Other applications involve feature fusion and image decomposition [46].

3.1.1 Neural Networks for Image Processing
Contrary to popular conception, neural networks do not mimic the operation of the human brain. The best that can be said is that a simplistic analogy is possible: like the brain, neural nets can be thought of as interconnected neurons linked together by synapses [46]. When enough of
the input synapses send a signal into a neuron, it ‘fires’, causing signals to be sent down its output synapses, which in turn cause other neurons to fire, and so on. However, what can a neural net do? Very simply - it can map data. When a pattern of signals is sent down its input links, the neurons fire in complex and inscrutable ways, resulting in a pattern of signals on the output links. The nature of this mapping depends partly on the network’s structure - the number of neurons, their organization, and so on - and partly on the strengths, of weights, on each of the links [47]. The larger the weight, the greater the effect that the link has in determining whether or not the neuron it connects into will fire. These weights could be set up by a learning process, which trains the network to behave in a required way. This is one of the essential characteristics of the technology. One approach to training is to present the untrained network with patterns of typical input data and adjust the weights according to how much the resulting output patterns differ from what they should be [45]. This adjustment is performed repeatedly, and for many input patterns, until the network operates satisfactorily.

The following properties characterize the PCNN among the neural network in general:

**There is a one-to-one match between each input data and each neuron.** That is, each neuron receives intensity data from exactly one pixel in the input image. In Fig 3.1 It could also be seen that the PCNN only consists of one single layer.

![Fig 3.1: Ono-to-one pixel mapping of the PCNN](image)

From this, it follows that all interconnections between neurons are within this layer. In the PCNN, each neuron only has contact with its closest neighbors; the feeding radius. Figure 3.2[47] shows a neuron that has eight...
neurons within feeding radius. Each of these neurons contributes to the input for the neuron in the middle.

Each neuron contains two input compartments: the feeding and the linking. The feeding receives an external stimulus as well as local stimulus while the linking only receives local stimulus. The local stimulus comes from the neurons within feeding radius. This local stimulus is hereafter called the firing information. The external stimulus is the intensity from the corresponding pixel in the picture. The feeding and linking are combined in a second order fashion to create the potential. The potential together with the output then decides whether the neuron should fire or not [48].

The PCNN does not require training. Instead, all the neurons in the net have their parameters set. However, the parameters should be able to be changed during operation as seen in the introduction, but not by the network itself [20]. In Table 3.1 [48] features of the conventional ANNs are compared to the PCNN features.

Table 3.1 a comparison between conventional ANN & PCNN
### 3.1.2 PCNN Models

A pulse-coupled neural network (PCNN) is a model of a biological network, specifically, a model of fragment of cat's sight network. It is a single-layer network [48] composed of neurons. Each of them is linked to one pixel of the input image. Each neuron contains two input compartments: the feeding and the linking. The feeding receives an external stimulus as well as local stimulus while the linking only receives local stimulus [48]. The local stimulus comes from the neurons within feeding radius. This local stimulus is hereafter called the firing information. The external stimulus is the intensity from the corresponding pixel in the picture. The feeding and linking are combined in a second order fashion to create the potential which then decides together with the output whether the neuron should fire or not [47]. In 2000, Bressloff and Coombes [49] made an intensive study of the dynamic behavior of PCNN with strong coupling and pointed out how the phase locking state enters into an unstable state with the enhancement of the coupling strength in the weak coupling limit.

PCNN model with some characteristics, such as strong adaptive capturing ignition, internal coupling, dynamic alignment threshold to control impulse-firing, has been widely applied to image de-noising [50], image smoothing [51], image segmentation [52, 53] and image fusion [54]. It is also partly used in shortest path optimization [55], Structural layout optimization [56], etc.

Johnson [51] has modified the initial PCNN model, which was much easier to use in computer calculations. The improved model of PCNN has the suitable feature of transmitting burst, which is widely used in such fields as image processing, pattern recognition and so on.

### Modified PCNN

<table>
<thead>
<tr>
<th>conventional ANNs</th>
<th>PCNNs for vision purposes</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple model neurons</td>
<td>complex model neurons</td>
</tr>
<tr>
<td>no considerations of timing effects</td>
<td>modelling of neural timing</td>
</tr>
<tr>
<td>continuous activation</td>
<td>activity conversion into spike trains</td>
</tr>
<tr>
<td>almost full connectivity</td>
<td>sparse connectivity</td>
</tr>
<tr>
<td>all neurons involved</td>
<td>few neurons involved</td>
</tr>
<tr>
<td>mainly supervised learning</td>
<td>unsupervised learning</td>
</tr>
</tbody>
</table>
There are several differences between the algorithms for the modified PCNN neuron and the exact physiological pulse coupled neuron. The differences are due to several simplifications made to the calculations, while still keeping the main features of the general theory. Each neuron in the modified PCNN could be described by the following set of equations [57]:

\[
L(i) = L(i-1) \cdot e^{-\alpha_L} + VL \cdot (R \ast Y_{sur}(i-1)) \quad (1)
\]

\[
F(i) = S + F(i-1) \cdot e^{-\alpha_F} + VF \cdot (R \ast Y_{sur}(i-1)) \quad (2)
\]

\[
U(i) = F(i) \cdot [1 + \beta \cdot L(i)] \quad (3)
\]

\[
\theta(i) = \theta(i-1) e^{-\alpha_q} + V\theta Y_{out}(i-1) \quad (4)
\]

\[
U > \theta(i) \Rightarrow Y_{out} = 1 \quad \text{(Firing Condition)}
\]

otherwise \(\Rightarrow Y_{out} = 0\) \quad (5)

Where \(L(i)\) is input linking potential, \(F(i)\) is input feeding potential and \(S\) represents the intensity of given image element. \(U(i)\) is the activation potential of neuron, \(\theta(i)\) is threshold potential of neuron and \(i\) is iteration step. Parameters \(\alpha_L\), \(\alpha_F\) and \(\alpha_q\) decay coefficients, \(\beta\) is linking coefficient and parameters \(VL\) and \(VF\) are coefficients of the linking and threshold potential. \(Y_{sur}\) is the firing information that indicate whether the surrounding neurons have fired or not and \(Y_{out}\) indicates whether this neuron fires or not. \(R\) is the matrix of weight coefficients and \(*\) is convolution operator.

An example of the modified PCNN neuron architecture is shown in Fig 3.4 as a schematic block diagram of the modified PCNN neuron as described through (1) - (5).

![Figure 3.4: A modified model for PCNN neuron Optimized PCNN](image)

Figure 3.4: A modified model for PCNN neuron Optimized PCNN
The main aim of Optimized PCNN was to reduce the number of generated features to reach high image recognition performance. The optimization was based on the PCNN with modified primary input (MPCNN), where the following disadvantages were eliminated [57]:

- High number of parameters and problems with their optimization.
- Optimal number of iteration steps determination.
- Value of the most significant feature equals to 1. After standardization so this feature loses its information value.

The feeding potential $F(i)$ is defined by the intensity pixel $S_{ij}$ only as in case of Modified-PCNN. The linking potential $L(i)$ is defined only by the convolution matrix $K(i)$ that is calculated by term:

$$K(i) = R * (X_{sur}(i-1).Y_{sur}(i-1))$$

$$L(i) = K(i)$$

$$U(i) = F(i) \times [1 + \beta \cdot L(i)]$$

$$X(i) = \frac{1}{1+e^{-\theta(i) - U(i)}}$$

$$X(i) > 0.5 \Rightarrow Y_{out} = 1 \quad (\text{Firing Condition})$$

$$X(i) \leq 0.5 \Rightarrow Y_{out} = 0$$

Where $Y(i)$ is output quantity based on step-function and $X(i)$ is output quantity based on sigmoid function.

### 3.1.3 Feature Extraction Methods

The standard approach of feature generation $G(n)$ for specific iteration $(i)$ is based on series of virtual binary images generation. It is calculated as sum of output quantities $Y_i$ of activated neurons in the given iteration step [58].

$$G(n) = \Sigma_{i=1}^{n} Y_i$$

Through influence of geometrical transforms it is very important to standardization of the generated features by standard equation [57]

$$g(n) = \frac{G(n)}{\max (G)}$$

Where $\max (G)$ is function that returns the maximal value in the feature space for the first impulse of function $G(n)$. The feature with maximal
value is a feature with maximal information value for image recognition process. Equation (12) results one at this feature which makes this value irrelevant. To overcome this problem, a modification can be applied to (12), such that [58]:

\[
g(n) = \frac{g(n)}{\sum_{n} S_{nj}} \tag{13}
\]

Where \( z_{ij} \) is the intensity of a given image pixel \((i,j)\), the previous equation achieves the standardization but it does not satisfy the condition of \( 0 \leq g(n) \leq 1 \). \( 0 \leq g(n) \) is always guaranteed, but \( g(n) \leq 1 \) is not valid in the all cases. This is because the sum of values \( Y(i) \) in the given iteration step \( i \) may be higher than sum of values \( S_{ij} \). Froge [59] proposed a new form of feature generation by introducing a new equation for feature value calculation \( g(n) \):

\[
g(n) = \frac{\sum_{i} X(i) \times Y(i)}{\sum_{n} S_{nj}} \tag{14}
\]

Also, he checked the validity of the standardization condition \( 0.5 < (X(i) \times Y(i)) < 1 \) and he showed the invariance against non-standard cases. He did not study the effect of lighting conditions and background brightness on his method.

### 3.2 The Proposed Feature Generation Approach

The aim of the proposed method is to allow sufficient long time-series generation with minimum interference effect. Additional iterations are important to extract more detailed features in large background or non-uniform light distribution. The idea of the proposed method is to weight each pulse by "Continuity factor". This factor mainly depends on how the surrounding pixels are fired in the same iteration. This factor has its maximum value when the pulse shows dense scene, and has its minimum value when the pulse shows holes and sparse scene. Figure 3-5 showed 2 pulses, the first showed a sparse pulse which will take a small weighting factor while the second contains a dense one for the same image which will have a relatively higher value.
3.2.1 Continuity Factor and the Proposed Approach

First we define an operator which is sensitive to the pixel intensity change, the operator happens to be the gradient. If the image is regarded as a function of two variables $A(x, y)$, then the gradient is defined as:

$$\nabla x^2 = Y_{out}(x + 1, y) - Y_{out}(x - 1, y) \quad (15)$$
$$\nabla y^2 = Y_{out}(x, y + 1) - Y_{out}(x, y - 1) \quad (16)$$

This operator is symmetrical with respect to the pixel $(x, y)$, although it does not consider the value of the pixel at $(x, y)$. Regardless the operator used to compute the gradient, the resulting vector contains information about the degree of strength the surrounding pixels are connected. The magnitude of the gradient vector is the length of the hypotenuse of the right triangle having sides $(\nabla x$ and $\nabla y$), and this reflects the strength of the continuity, or continuity response, at any given pixel.

$$c_r = \sqrt{(\nabla x^2)^2 + (\nabla y^2)^2} \quad (17)$$

$\nabla x^2$ and $\nabla y^2$ takes either 0 or 1, the continuity response values are 0, 1 and 1.414. The value of the response can be summed then normalized and producing the Continuity Factor, at a given pulse $(i)$ by computing the following formula:

$$CF(i) = \frac{\sum_{(x,y)} c_r}{(1/14) \times N} \quad (18)$$

Where $N$ is the number of pixels in the input image, the value 1.414 is the maximum value $Cr$ can take. This concludes the correctness of normalization condition $0 \leq CF \leq 1$. The new method modifies (14) by adding the continuity factor defined in (18) and produces:

$$g(n) = \frac{\sum_{(x,y)} (x \times y \times c_r)}{\sum_{x,y} s_x s_y} \quad (19)$$

Since it has been proved that $CF$ at iteration $(i)$ is less than 1, the condition of standardization is still correct, $0 \leq g(n) \leq 1$ and invariance against non-standard cases is achieved.

3.2.2 The Proposed Approach
The first approach of feature generation by (11) suffers from standardization problem, because it is affected hugely by object transformation. The second approach of feature generation by (12) is not accepted because it suffers from invalidity of normalization condition. The third approach of feature generation by (14) is suitable for OM-PCNN also is invariant against non-standard cases. The disadvantage of this approach is the necessity of eliminating the image background effect and ensuring the homogeneity of lighting distribution. This method permits the calculation of a sufficiently long image signature with a minimal lighting effect. He emphasized the standardization against geometrical transformation. Feature extraction method using eq(15) showed a superiority to other methods, Fig 3.6 illustrates 2 images of the same posture; one of them is a clean image and the other is a scaled and non homogeneous light image.

Figure 3.6 Two images of the same posture, left is clean and right is distorted

Figure 3-7 illustrates the corresponding image features generated by eq(12), eq(14) and the proposed approach for both images. Using eq(19) gave a more accurate signature for the distorted image.
3.3 Proposed ASL Recognition Architecture
The proposed recognition system is a hybrid system mainly is composed of 2 basic modules:
1- Enhanced feature extraction module using PCNN.
2- An invariant classifier using Multi-layer perceptron (MLP).
Neural networks are applicable in virtually every situation in which a relationship between the predictor variables (independents, inputs) and predicted variables (dependents, outputs) exists, even when that relationship is very complex and not easy to articulate in the usual terms of
"correlations" or "differences between groups". The most popular network used in classification problems is "Multi-Layer Perceptron" (MLP). This network has many benefits: it is a general purpose model with various applications; it is capable of modeling nonlinear complex functions, good performance in noisy input and it can be adaptable for environmental changes by changing weights and/or topology. There are several training algorithms for MLP [60]. Back propagation is used to train the network. Figure 3.8 illustrates the proposed system for ASL static postures recognition system.

3.4 Implementation and Experimental Results

This research applies the proposed method to recognize ASL 158 static postures through exposing the image to OM-PCNN and evaluates the image signature. The image signature values are then classified using Multi-Layer Perceptron (MLP) network. Each posture is represented by 10 exemplars:

- 6 of them used as training set, the object pixels percentage inside the image varies from 40% to 80%.

Figure 3.8 the recognition system schematic diagram
• 2 as clean test set, object pixels percentage varies from 60 % to 70% and uniform light distribution.
• 1 contains the same posture but the object pixels percentage does not exceed 25% of the image pixels and shifted from its original position.
• 1 contains non-uniform lighting conditions.
The implementation details of the PCNN are: $\beta=0.1, \alpha_L=1.0, \alpha_q=0.8$, $V_L=0.25$, $V_F=0.6$ and $V_\theta=20$.

$$R = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$ (20)

The implementation details of the MLP network are: Step size = 0.001, Number of iterations = 50, 1 hidden layer has been used and Sigmoid Activation function was used. This study aims to enhance the feature calculation and compare previous methods results using the sum method (equation (11)), Froge [59] (equation (14)) method and the proposed method result using equation (19). The comparison is presented according to the same recognition environment; Fig.3.9 illustrates the superiority of the proposed method (19).

Figure 3.9: The recognition accuracy against the number of features from the 3 methods

The first approach does not neutralize rotation and shifting. The second approach as expected showed enhanced results in both clean and scaled
rotated data sets due to standardization capabilities and ability to save the feature with the maximal information value. It fails in images with non-uniform light distribution and large background percentage. The proposed method achieves 92% recognition accuracy which was obtained when the maximum 11 features are used which mainly is a problem field dependent factor. When the features increase, the recognition ratio starts to degrade. This degradation happens because the effect of the interference starts to increase as the number of iterations increases. Although the interference effect still exists, the superiority of the proposed method has been proved. It permits the calculation for relatively larger number of iterations which can be sufficient to recognize the images with non-uniform lighting or large background. The proposed model can be generalized and applied to other sign languages [5-7].

The recognition finds some problems in identifying some postures. These problems come from the fact that a single view of the hand does not distinguish between two different postures. Fig 3.10 shows two different postures the frontal views do not differ which cannot be resolved using the single frontal view.

![Figure 3.10: 2 different postures a-front view. b- side view](image)

An enhancement (discussed in next chapter) can be made by constructing 3D model for the posture containing multiple 2D images that contain both frontal and side views. This enhancement requires capturing the posture with more than one camera with different perspectives. Two synchronized cameras (associated with different workstations) can be used; one in the frontal view can recognize 90% of the postures, and another one in the side view can recognize the remaining undistinguishable 10%.
Chapter 4
Proposed 3D Hand Signs Recognition

Earlier researches in Sign Language Recognition suffered from a main problem that they find difficulties in identifying some postures. This is caused by the fact that a single view of the hand does not distinguish two different postures. This chapter proposes a novel technique to deal with pose variations in 3D object recognition. This technique uses Pulse-Coupled Neural Network (PCNN) for image features generation from two different viewing angles. These signatures qualities are then evaluated, using a proposed fitness function. The features evaluation step is taken before any classification steps are performed. The evaluation results dynamic weighting factors for each camera express the features quality from the current viewing angles. The proposed technique uses the two 2D image features and their corresponding calculated weighting factors to construct optimized quality 3D features. An experiment was conducted in Arabic Sign Language recognition application which multiple views are necessary to distinguish some signs. The proposed technique obtained 90% recognition accuracy for pose-invariant restrictions with a degree of freedom from 0 to 90.

4.1 3D Object Recognition

Earlier researches in 3D object recognition attempt to recover full 3D shape information before performing the recognition task. This method is known as object based representation. Another method is known as view-based technique, in which 3D object is described, using set of 2D characteristic views. Paggio and Edelman [61] showed that 3D objects can be recognized from the raw intensity values in 2D images, using a generalized radial basis functions. They demonstrated that a full 3D structure of an object can be estimated if enough 2D views of the object are provided. The main disadvantage of view-based technique is the inherent loss of information in the projection from 3D object into 2D image. Moreover, the 2D image of a 3D object depends on factors, such as the camera viewpoint and the viewing geometry. The first step in an object recognition system is to define an appropriate coordinate system. There are two ways to define this coordinate system of a three-dimensional shape, the viewer-centered approach and the object-centered approach.

4.1.1 Viewer-Centered Approach
If objects usually appear in a relatively few stable positions with respect to the camera then they can be represented efficiently in a viewer-centered, viewing angle dependent, coordinate system, which describes the 3-D object using a set of 2-dimensional characteristic views or aspects. Each characteristic view describes how the object appears from a single viewpoint. Typical examples of object recognition using viewer-centred representations are the aspect graphs by Koenderink [62] Paggio and Edelman [61] B¨ulthoff and Edelman [63] and Ullman and Basri [64]

4.1.2 Object-centered Approach

The alternative to the viewer-centered approach is the object-centered approach, which describes objects usually as a three-dimensional entity within a coordinate system attached to the object. For example, it specified the object’s parts relatively to the object’s main axis. Object-centered representations are independent of the camera parameters and location, and yield the most concise and usually most accurate shape descriptions. There are also two main choices for the object recognition strategy: the feature-based strategy, which is based on shape information (Huttenlocher and Ullman [64, 65] ,Lamdan et al.[66] and Jacobs [67])) and the image-based strategy, which is based on a direct representation of image intensity [68-70] or on a filtered version of the image [71, 72]. Table 4.1 illustrates some of the previous developed systems and their characteristics.
Table 4.1 Examples of previous 3D recognition systems

<table>
<thead>
<tr>
<th>Authors</th>
<th>Approach</th>
<th>Camera-as</th>
<th>Degree space</th>
<th>Pose Invariant</th>
<th>Object Distance Consideration-on</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vasilios Zografos and Brand F. Buxton[73]</td>
<td>View-Based Approach</td>
<td>1</td>
<td>22.5</td>
<td>Yes</td>
<td>No</td>
<td>85% on 20 objects</td>
</tr>
<tr>
<td>Jan Weighhardt and Cristoph von der Malsburg[74]</td>
<td>Object-Based approach</td>
<td>2</td>
<td>20</td>
<td>Yes</td>
<td>No</td>
<td>87% on clean 22 faces</td>
</tr>
<tr>
<td>Gary Bradeskl [75]</td>
<td>View-Based Approach</td>
<td>2</td>
<td>60</td>
<td>Yes</td>
<td>No</td>
<td>90% on 60 objects</td>
</tr>
<tr>
<td>Horst Eidenberger [76]</td>
<td>View-Based Approach</td>
<td>1</td>
<td>-</td>
<td>Yes</td>
<td>No</td>
<td>60% on 50 faces</td>
</tr>
</tbody>
</table>

Tangruamsub [77] presented a 3D object recognition method. The proposed objectives are to overcome shortcoming of the appearance-based method, which lacks a spatial relationship between the parts of an object, and those of other 3D model methods, which require a complicated computation. The proposed method is based on a voting process. Appearance estimation is introduced in this work in order to deal with the faulty detection problem. He tested the method of object detection and pose estimation.

4.2 Hands 3D Model Construction

This chapter proposes and implements a model of the problem of object recognition particularly the detection of 3-D objects in 2-D intensity images which may be viewed from a variety of angles. A technique is proposed and developed to address this problem, which falls into the category of view-based approaches, that is, a method in which an object is represented as a collection of a small number of 2-D views, as opposed to a generation of a full 3-D model. This technique is based on the theoretical observation that the geometry of the set of possible images of an object undergoing 3-D rigid transformations and scaling may, under most imaging
conditions, be represented by a linear combination of a small number of 2-D views of that object. The system needs two synchronized cameras which are positioned in fixed points (A, B) which are located on the same horizontal but differ $90^\circ$ from each other. Figure 4.1 illustrates the cameras position points A and B.

![Figure 4.1 Cameras positions in A and B.](image)

Each signer hands to be recognized must be placed in its stable condition at the centre of circular turntable, which can be rotated in space of only 180 degrees. The view angles outside this range are irrelevant; the signer back will block the vision. After the signer's hands are placed at the centre of the turntable, the 2D images of the object are acquired while point A is located at $0^\circ$ and point B at $90^\circ$. Then, the two cameras will be rotated $5^\circ$ at a time and the two 2D images will be acquired again. Each time the cameras will be rotated at $5^\circ$ until point a reaches $90^\circ$ and point B reaches $180^\circ$. Hence, for each object 19 2D image sets are obtained from each camera. These images are divided into two groups, 10 image sets for training data and 9 image sets for testing data. The acquired images at $(0^\circ, 90^\circ), (10^\circ, 100^\circ)\ldots$ and $(90^\circ, 180^\circ)$ were used as the training set and the rest of the images (images at $(5^\circ, 95^\circ), (15^\circ, 105^\circ)\ldots$ and $(85^\circ, 175^\circ)$) were used as the testing set. The training data set is used to build the 3D object model in the recognition stage. Fig 4.2 illustrates the 3D feature generation operation.
The 2D captured images are then digitized and sent to the \textit{PCNN Feature Generation} module. The two output signatures are then weighted and combined linearly to produce 3D image signature. The PCNN 2D feature generation module uses (15) as image signature generation technique. Let \( P_1(i) \) represent the generated signature at point A and \( P_2(i) \) at point B, we define \( F(i) \) as 3D image signature. It is computed as the weighted linear combination of them:

\[
F(t) = \frac{w_1 P_1(t) + w_2 P_2(t)}{2}
\]  

(4.1)

The weights \( w_1 \) and \( w_2 \) values vary each viewing angle for each camera. At start camera B is more important such that it has the frontal view; the weight \( w_1 \) contains a larger value. On the contrary, at last capturing, camera A will have the frontal vision; the weight \( w_2 \) contains a larger value. Table 4.2 shows the values of \( w_1 \) and \( w_2 \) with each capturing angles. Note that, the sum value of \( w_1 \) and \( w_2 \) must be equal to 1.

<table>
<thead>
<tr>
<th>Camera A angle</th>
<th>Camera B angle</th>
<th>( W1 )</th>
<th>( W2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>90</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Table 4.2 values of \( w_1 \) and \( w_2 \) with different capturing angles.
The proposed weights in the previous table are calculated based on an experimental application of the proposed "Feature Quality Optimization" technique (which will be discussed in the next section) to training dataset. The model construction can be time and effort costly but this cost can be ignored since it is offline and done once. Since the available dataset are labeled, the supervised learning technique can be used to construct "3D Model" for each posture.

4.3 Proposed 3D Signature Generation Approach

This section describes the proposed technique that can be used to combine multiple 2D image signatures to generate a 3D image signature. This technique mainly depends on a novel concept in PCNN research works which we call "Image Signature Quality Optimization".

The proposed concept tries to find the best weights for multiple signatures to be linearly combined based on optimizing an objective function. This objective function tries to measure the clarity of image signature and its validity in truly representing the image. Unfortunately, there is no previous work that supposed such concept to compare with, so one proposed 3 objective functions and compared them. The proposed method succeeded in accomplishing the pose-invariance in real-time environment; the signer has the flexibility to move in XY space. According to this result, there is no frontal or side view. The weight of each camera is dynamically determined according to the signer view to the camera.

Figure 4.3 illustrates the same posture but in different poses. In the first pose, camera A is the frontal and camera B is the side. The second pose, the 2 cameras are swapped, camera A becomes side and camera B becomes frontal.
The question is: how to calculate \((w1)\) and \((w2)\) for the 2 views PCNN signatures to optimize the features quality of the generated 3D signature.

### 4.3.1 PCNN Signature Quality Measurement

The main objective of this paper is to propose an objective function that can present a credible indicator of the PCNN signature quality.

**A- Sum Function**

The first proposed objective function is the signature value sum. The basis of this function is, as larger as the values of signature points, the clearer features are obtained.

\[
\text{Sum (signature)} = \sum_{j=1}^{n} g(j)
\]  \hspace{1cm} (4.2)

Where \((g)\) is the feature generation function described by \((15)\) and \((j)\) is the iteration number. However, in some situations \((18)\) gives a wrong indicator, Fig 4.4 shows 2 different views to the same posture, view 1 carries clearer features than view 2. The result of \((18)\) does not indicate that reality:

- \(\text{Sum (view1)} = 5.9\)
- \(\text{Sum (view2)} = 7.1\)

Contrary to the reality, the Sum Function results that view 2 contains clearer features than view 1.
Another objective function is defined; this function basis is that the interference makes the signatures points tend to saturate in the middle value 0.5. The second function sums the difference between each point and the middle point. It is supposed that when the sum value is maximal, the interference effect is minimal.

\[
\text{Central Diff Sum} \quad \text{(signature)} = \sum_{j=1}^{n} [x(j) - 0.5] \tag{4.3}
\]

From conducted experiments (results section), also in some situations (4.3) indicates wrongly the clearer view. Figure 10 shows 2 different poses of the same posture such that F2 gives a wrong indicator.

\[
\text{Central Diff Sum (view1)} = 2.4
\]

\[
\text{Central Diff Sum (view2)} = 2.6
\]

The same problem happened but with a lower error; the Central Difference Sum function indicates wrongly that view2 is clearer than view1.
C-Gradient Sum Function
Another objective function is proposed. It gives an indicator to the signature sharpness. The signature sharpness is inversely proportional to the signature interference effect; as much the signature sharpness increases, the interference effect decreases. The proposed function measures the sharpness by calculating the sum of differences between each point and its predecessor.

\[ \text{GradientSum} \left( \text{signature} \right) = \sum_{j=2}^{n} \left[ g(j) - g(j-1) \right] \] (4.4)

Gradient Sum Function shows superiority to the previous equation such that it succeeded in giving a right indicator while the previous equations fail. For the views in Fig. 4.4 and Fig. 4.5, the results of (4.4) were:
- GradientSum (view1)=3.2
- GradientSum (view2)=1.5 (Fig 4.4)
- GradientSum (view1)=4
- GradientSum (view2)=1.9 (Fig 4.5)

In the experimental results section, the 3 proposed objective functions were applied and a comparative study on them is conducted.

4.3.2- PCNN Signature Quality Optimization
The second step is to apply an optimization algorithm to maximize GradientSum described in (4.4). The input of this algorithm is the two image signatures from 2 different poses, the viewing angles for both poses are unknown. The main step needed is re-formulate the 3D features generation problem into an optimization form, Fig 4.6.

<table>
<thead>
<tr>
<th>Input: g1(t) and g2(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let G [ x ]</td>
</tr>
<tr>
<td>Find w1 and w2 that:</td>
</tr>
<tr>
<td>Maximize GradientSum = [ x ] (objective)</td>
</tr>
<tr>
<td>Such that:</td>
</tr>
<tr>
<td>W1+w2=1 (Constraint)</td>
</tr>
</tbody>
</table>

Figure 4.6: The optimization problem for 3D pose invariant features
After formulating the optimization problem as a nonlinear objective function and problem domain constraints, an optimization algorithm is applied. Figure 4.7 shows common optimization methods are traditionally used in computer vision [78, 79]. The study and comparison of these algorithms are out of this paper scope.

Figure 4.7: Common optimization methods traditionally used in computer vision

The possible combinations of w1 and w2 can be listed in Table 3, assuming 2 digits accuracy.

<table>
<thead>
<tr>
<th>W1</th>
<th>W2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.95</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>0.15</td>
<td>0.85</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>0.95</td>
<td>0.05</td>
</tr>
</tbody>
</table>

"Exhaustive Search" algorithm is used to find the optimal combinations of w1 and w2 that maximizes the fitness function. The main reason for the
selection of "Exhaustive Search" is that the combinations are few and limited. The proposed technique shows a promising form to real-time recognition for sign language recognition problem. The time performance of the real-time recognition system is studied in detail in experimental section. The following reasons guarantee the real-time performance:

- The search technique "Exhaustive Search" was employed to find the optimal combinations between limited combinations.
- The independence nature between weights substitution in objective function permits that calculations are made simultaneously using any parallel processing paradigm.
- The feature quality objective function computation does not offer an additional time cost to the system.

Since the proposed technique does not use any special property of Arabic sign language, it can be used to recognize any sign language. Moreover, it can be applied to different 3D object recognition and tracking applications.

**4.4 Arabic Postures and Gestures Recognition Architecture**

To perform a sign, the hearing-impaired individual is positioned in front of main camera. The secondary camera is 90° apart from the main camera. Figure 13 illustrates the position of the signer and the two cameras. The perpendicular distance between the signer and camera A is equal to the distance between him and camera B.

![Figure 4.8: The signer and capturing cameras positions](image)

The signer performs a complete gesture which represents a word. The gesture is a movement which is composed of a sequence of postures.
4.4.1 3D Posture Classification
Each static image is considered as a posture, which is classified by "Posture Classification Algorithm". The algorithm steps are illustrated in Fig 4.9.

**Posture_Classification_Algorithm**

**Input:** 2 input images from 2 perpendicular views. The user pose has a degree of freedom between 0 and 180.

**Output:** The posture output class.

**Steps:**
- The PCNN feature generator computes for each image its signature using (14).
- Generates all possible combinations between $w1$ and $w2$ such that $w1+w2=1$.
- Substitutes all these combinations in the proposed objective function in GradientSum function.
- Uses "Exhaustive Search" to find the combination that maximizes the objective function.
- The output 3D signature is calculated by:

$$w1.P1+w2.P2$$

- 3D features are then classified using MLP neural network.

Figure 4.9: Hand posture classification algorithm steps.

4.4.2 Dynamic Gesture Classification
Each gesture is represented as a sequence of postures classes. This sequence may contain unclassified postures which are considered "transient state". Figure 4.10 shows a signer that counts from 1 to 2. Each frame represents a separate posture; all the transient movements are "transient postures".

![One Gesture](image)

Figure 4.10: one gesture represents counting from 1 to 2 and the corresponding postures
The previous gesture contains 3 posture classes: one class for counting 1, another class for counting 2 and transient class represents the transition. Figure 4.11 illustrates the representation of the previous gesture as Non-deterministic Finite Automaton (NFA).

![Diagram of NFA for counting from 1 to 2]

Figure 4.11. The Corresponding NFA for counting from 1 to 2

After the gesture is being represented by NFA, the "Best Match" algorithm is used to find the most probable meaning from gestures database. This part can be considered as an "Error Tolerant" module for posture classification. Any posture which is unclassified or falsely classified from posture classification module can be corrected in gesture classification.

4.5-Implementation and Experimental Results

The first step is to collect images for both training and testing. The images have been captured in an uncontrolled environment which is affected by noise, non-uniform lighting and background, different capturing distance and traditional object transformations. Using two fixed cameras, two simultaneous images are captured for the same postures. It means that there exist 10 image pairs for each single posture each 10 degrees. We have 1580 image pairs for the whole dataset. For each image pair, 15 exemplars were captured. It means that: we have total 23700 image pairs.

This research applies the proposed architecture of constructing and recognizing 3D hands postures model. The second study tried to classify 158 postures using Optimized PCNN as feature generator technique followed by MLP as a classifier. Equation (15) was used as signature generation technique. Each posture is represented by 19 2D image sets which are obtained from each camera. These images are divided into two groups: 10 image sets for training data and 9 image sets for testing data are captured in different angles. The training sets are used to construct the 3D model. The implementation details of the PCNN for 2D features generation are: $\beta=0.1, \alpha_L=1.0, \alpha_q=0.8, V_L=0.25, VF=0.6$ and $V\theta=20$. 
The implementation details of the MLP network are: 1 hidden layer has been used and Sigmoid Activation function was used. The network has been trained using the Back-Propagation algorithm and was trained until mean square error between the network output and desired output falls below 0.05. The weights were updated after each pattern presentation. The learning rate and momentum were 0.2 and 0.1 respectively. To determine the suitable input features for training the MLP network, three sets of analyses were carried out using 5, 11 and 18 features as given in Table 4. In each set, 50 simulated images. As previously mentioned in section 4, the acquired images at (0°, 90°), (10°, 100°)...and (90°, 180°) were used as the training set and the rest of the images (images at (5°, 95°), (15°, 105°)... and (85°, 175°)) were used as the testing set. The results of comparison of the classification accuracies of the three objective functions (at training epoch 20, 40, 60, 80 and 100) are shown in Table 4.4. The 3D model is constructed, using all these sets with substituting the values of w1 and w2 with values from Table 2. These images are then exposed to the posture classification algorithm to be recognized. Table 4 illustrates the recognition accuracy using the 3 proposed fitness functions of the complete dataset. It shows that the fitness function described by (20) achieves the best results. The recognition accuracy does not get below 85%. The 5 features set gave the lowest overall accuracy compared to the other two. For the set containing 11 features, the accuracy is generally higher than that of the 18 features set. The comparison between the 11 features set and 18 features set shows that increase in the number of features will not necessarily increase the classification accuracy at the same number of hidden nodes. In addition, a larger features set will increase the processing time while resulting in similar results. Thus, in the current research 11 features set were selected as the optimized set of input features to train the network and classify the postures.

Table 4.4: The recognition accuracy using the 3 objective functions
The maximum accuracy (95%) was found to occur when 14 hidden nodes were used. The comparison of the ROC (Receiver Operating Characteristics) curves obtained by the 3 proposed objective functions is given by Fig 4.12. The area under the ROC curve of the Gradient Sum Function is significantly larger than the other two functions. This difference indicates that the Gradient Sum Function has the greatest feature quality optimization effect.

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>Accuracy/%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5 features</td>
</tr>
<tr>
<td>Sum Function</td>
<td>Gradient Sum Function</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>76.3</td>
</tr>
<tr>
<td>40</td>
<td>79.1</td>
</tr>
<tr>
<td>60</td>
<td>79.1</td>
</tr>
<tr>
<td>80</td>
<td>79.1</td>
</tr>
<tr>
<td>100</td>
<td>79.1</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>82.3</td>
</tr>
<tr>
<td>40</td>
<td>80.2</td>
</tr>
<tr>
<td>60</td>
<td>80.2</td>
</tr>
<tr>
<td>80</td>
<td>80.5</td>
</tr>
<tr>
<td>100</td>
<td>81.3</td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>81.3</td>
</tr>
<tr>
<td>40</td>
<td>84.4</td>
</tr>
<tr>
<td>60</td>
<td>81.3</td>
</tr>
<tr>
<td>80</td>
<td>81.3</td>
</tr>
<tr>
<td>100</td>
<td>81.3</td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>80.2</td>
</tr>
<tr>
<td>40</td>
<td>81.5</td>
</tr>
<tr>
<td>60</td>
<td>81.3</td>
</tr>
<tr>
<td>80</td>
<td>81.3</td>
</tr>
<tr>
<td>100</td>
<td>82</td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>81.3</td>
</tr>
<tr>
<td>40</td>
<td>81.3</td>
</tr>
<tr>
<td>60</td>
<td>81.3</td>
</tr>
<tr>
<td>80</td>
<td>81.3</td>
</tr>
<tr>
<td>100</td>
<td>81.3</td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>77.9</td>
</tr>
<tr>
<td>40</td>
<td>81.3</td>
</tr>
<tr>
<td>60</td>
<td>82.5</td>
</tr>
<tr>
<td>80</td>
<td>82.5</td>
</tr>
<tr>
<td>100</td>
<td>82.5</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>86</td>
</tr>
<tr>
<td>40</td>
<td>81.3</td>
</tr>
<tr>
<td>60</td>
<td>81.3</td>
</tr>
<tr>
<td>80</td>
<td>81.3</td>
</tr>
<tr>
<td>100</td>
<td>81.3</td>
</tr>
</tbody>
</table>
The system fails to distinguish some postures because the signer was at a relatively large distance from the cameras system. This issue comes from the fact that some samples are captured at a relatively long distance which makes the object size very small and indistinguishable.

After deciding picking the fitness function in (4.4), Fig 4.13 illustrates the recognition accuracy against the object percentage of the frontal and side views. It shows that as the object percentage increases, the features are clearer and the recognition accuracy increases. When object percentage got lower than 40%, the recognition accuracy decreases dramatically. This comes from that background portion which affects the image signature which is known as "interference effect".
This study scope contains the recognition of 50 dynamic word gestures. Table 4.5 shows the words that have been included. The dataset can be easily extended so that it can be extended offline without re-training of postures models.

Table 4.5: the gestures of the dataset

<table>
<thead>
<tr>
<th>Sign Number</th>
<th>Arabic word</th>
<th>English Translation</th>
<th>Sign Number</th>
<th>Arabic word</th>
<th>English Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>جديد</td>
<td>New</td>
<td>26</td>
<td>عمل</td>
<td>Work</td>
</tr>
<tr>
<td>2</td>
<td>ذكي</td>
<td>Smart</td>
<td>27</td>
<td>ياخذ</td>
<td>Take</td>
</tr>
<tr>
<td>3</td>
<td>طويل</td>
<td>Tall</td>
<td>28</td>
<td>يعطي</td>
<td>Give</td>
</tr>
<tr>
<td>4</td>
<td>قصير</td>
<td>Short</td>
<td>29</td>
<td>نقلن</td>
<td>Discuss</td>
</tr>
<tr>
<td>5</td>
<td>وسيم</td>
<td>Handsome</td>
<td>30</td>
<td>يتكلم</td>
<td>Speak</td>
</tr>
<tr>
<td>6</td>
<td>مهم</td>
<td>Important</td>
<td>31</td>
<td>يدرس</td>
<td>Teach</td>
</tr>
<tr>
<td>7</td>
<td>مهم</td>
<td>Careless</td>
<td>32</td>
<td>يعلم</td>
<td>Know</td>
</tr>
<tr>
<td>8</td>
<td>عصبي</td>
<td>Nervous</td>
<td>33</td>
<td>يصح</td>
<td>Correct</td>
</tr>
<tr>
<td>9</td>
<td>هادي</td>
<td>quiet</td>
<td>34</td>
<td>يقطع</td>
<td>Cut</td>
</tr>
<tr>
<td>10</td>
<td>مميز</td>
<td>Special</td>
<td>35</td>
<td>يمارس</td>
<td>Practice</td>
</tr>
<tr>
<td>11</td>
<td>من</td>
<td>From</td>
<td>36</td>
<td>أسد</td>
<td>Lion</td>
</tr>
<tr>
<td>12</td>
<td>الي</td>
<td>To</td>
<td>37</td>
<td>تعرف</td>
<td>Tiger</td>
</tr>
<tr>
<td>13</td>
<td>على</td>
<td>on</td>
<td>38</td>
<td>فيل</td>
<td>Elephant</td>
</tr>
<tr>
<td>14</td>
<td>عن</td>
<td>Off</td>
<td>39</td>
<td>قطة</td>
<td>Cat</td>
</tr>
<tr>
<td>15</td>
<td>في</td>
<td>In</td>
<td>40</td>
<td>كلب</td>
<td>Dog</td>
</tr>
<tr>
<td>16</td>
<td>مهندس</td>
<td>Engineer</td>
<td>41</td>
<td>سماء</td>
<td>Sky</td>
</tr>
<tr>
<td>17</td>
<td>طبيب</td>
<td>Doctor</td>
<td>42</td>
<td>أرض</td>
<td>Land</td>
</tr>
<tr>
<td>18</td>
<td>مدرس</td>
<td>Teacher</td>
<td>43</td>
<td>ماء</td>
<td>Water</td>
</tr>
<tr>
<td>19</td>
<td>عامل</td>
<td>Worker</td>
<td>44</td>
<td>رمل</td>
<td>Sand</td>
</tr>
<tr>
<td>20</td>
<td>تاجر</td>
<td>Merchant</td>
<td>45</td>
<td>البحر</td>
<td>Sea</td>
</tr>
<tr>
<td>21</td>
<td>يضرب</td>
<td>Hit</td>
<td>46</td>
<td>دبابة</td>
<td>Tank</td>
</tr>
<tr>
<td>22</td>
<td>يمشي</td>
<td>Walk</td>
<td>47</td>
<td>طائرة</td>
<td>Plane</td>
</tr>
<tr>
<td>23</td>
<td>يأكل</td>
<td>Eat</td>
<td>48</td>
<td>صاروخ</td>
<td>Rocket</td>
</tr>
<tr>
<td>24</td>
<td>يشرب</td>
<td>Drink</td>
<td>49</td>
<td>مسند</td>
<td>Gun</td>
</tr>
<tr>
<td>25</td>
<td>ينام</td>
<td>Sleep</td>
<td>50</td>
<td>سفينة</td>
<td>Ship</td>
</tr>
</tbody>
</table>

The results obtained on isolated gestures levels are as follows: one issue must be addressed, system performance behavior against dataset size. Figure 4.14 illustrates the accumulative effect of the dataset size on system recognition accuracy. The recognition accuracy never got beneath 94%.
The misclassification in some situations comes from the fact that some gestures NFA may be totally included in another gesture NFA. For example, the gesture "مدرس" is part of the gesture "مدرس". This inclusion makes the best match misclassify the gesture NFA. Testing times are across the various stages are measured for variations in postures and gestures datasets sizes. Table 4.6 illustrates the measured times (in ms) across each system stage. The experiments are done using Intel® Core™ i5-2300 (6MB Cache) and 8GB Dual Channel DDR3 at 1333MHz. also 2 cameras 5 megapixel each.
The features extraction phase time is independent on dataset size, also features optimization phase. The posture classification phase increases as much as the postures dataset size increases. The gesture composition time is affected by gestures database size. For full dataset size, recognition for a single gesture (word) does not exceed 0.7 second. Finally we present a comparative study (Table 4.7) between the proposed model and other systems; it includes the dataset size, recognition algorithm used and environment constraints.
Table 4.7: Comparative study with existing systems

<table>
<thead>
<tr>
<th>Author</th>
<th>Sign language</th>
<th>Dataset size</th>
<th>Recognition model</th>
<th>Recognition accuracy</th>
<th>Pose invariant</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>B. Bauer and H. Hienz [16]</td>
<td>German</td>
<td>52 signs from a single signer</td>
<td>Hidden Markov Models</td>
<td>94%</td>
<td>No</td>
<td>Recognition accuracy is guaranteed for one signer. The dataset is not extendible, when we need add a new sign, it needs re-training to the whole system.</td>
</tr>
<tr>
<td>Chen et al. [11]</td>
<td>Taiwanese</td>
<td>20 words from 20 signer</td>
<td>Frequency domain features (Fourier Transform) plus some information from motion analysis HMMs was used as the classifier.</td>
<td>92.5%</td>
<td>No</td>
<td>Recognition accuracy is guaranteed for one signer. Isolated signs only.</td>
</tr>
<tr>
<td>O. Aljarrah et al. [25]</td>
<td>Arabic</td>
<td>30 static gestures</td>
<td>collection of Adaptive Neuro-Fuzzy Inference System (ANFIS) networks for training and classification depending on spatial domain</td>
<td>96%</td>
<td>No</td>
<td>Static signs only.</td>
</tr>
<tr>
<td>Mohandes et al. [27]</td>
<td>Arabic</td>
<td>50 signs of words performed by one person having 10 samples per sign.</td>
<td>Histograms analysis and MLP as classifier.</td>
<td>They achieved 92% recognition accuracy</td>
<td>No</td>
<td>Static postures only.</td>
</tr>
<tr>
<td>Tolba et al. [80]</td>
<td>Arabic</td>
<td>30 alphabets postures</td>
<td>PCNN followed by MLP</td>
<td><em>gained 93% recognition accuracy</em></td>
<td>No</td>
<td>The system is signer-independent and achieved system invariance against rotation, scaling and color. Found difficulties in identifying some postures. This is caused by the fact that a single view for the hand does not distinguish between two different postures.</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>Arabic</td>
<td>158 static postures and 50 dynamic gestures</td>
<td>Multiple PCNN signatures, MLP and best match algorithm</td>
<td>Gained 95% for static postures. 96% for dynamic gestures</td>
<td>yes</td>
<td>The system is insensitive to signer position, view angle and background effects</td>
</tr>
</tbody>
</table>
Chapter 5
Graph Matching & Gesture Reconstruction

This chapter consists of two basic sections:
First it illustrates the scientific background for "Graph Matching" problem and its applications. It illustrates one of the standard algorithms for sub-graph isomorphism.
Also, it illustrates a decision tree-based algorithm which was proposed by B.T. Messmer and H. Bunke [81].
Secondly, it illustrates the proposed technique for gesture re-construction (gesture level). Then it proposes an application for sub-graph matching for gestures re-construction (sentence level). And shows the decision tree-based algorithm customization for dynamic gesture recognition.

5.1 Graph Matching Problem

Graphs are a general and powerful data structure useful for the representation of various objects and concepts. In pattern recognition and machine vision [82], for example, graphs are often used to represent object models, which are known a priori and store in a database, and unknown objects, which are to be recognized. Using graphs as representation formalism, the recognition problem turns into a graph matching problem. An input graph representing an unknown object is compared to the database in order to find the most similar model graph.
In a general representation, the nodes typically represent objects or parts of objects, while the edges describe relations between objects or object parts. For example, a node may represent a line segment, a closed region, or a surface patch of a 3-D object, while an edge describes relationships such as parallelism for straight lines, or spatial adjacency for image regions and surface patches.
Why Graphs?

Graphs have some interesting invariance properties [83] for example, if a graph, which is drawn on paper, is translated, rotated or transformed into mirror image, it still the same graph in the mathematical sense. These invariance properties as well as the fact that graphs are very well suited to
model complex objects in terms of parts and their relations make them very attractive for various applications. Graph representation and graph matching have been successfully applied to a large number of problems in computer vision and pattern recognition. Examples include character recognition, schematic diagram interpretation, shape analysis, image registration, and 3-D object recognition.

5.1.1 Basic Notation and Terminology

In this section we give the basic definition and notations that will be used throughout this chapter [84].

**Definition 5.1**
A labeled graph $G$ is a 6-tuple, $G = (V, E, \mu, \nu, L_v, L_e)$ where:

- $V$ is the set of vertices,
- $E \subseteq V \times V$ is the set of edges,
- $\mu : V \rightarrow L_v$ is a function assigning labels to the vertices,
- $\nu : E \rightarrow L_e$ is a function assigning labels to the edges.

As usual, we assume that $L_v$ and $L_e$ are finite sets of symbolic labels. Note that the above definition corresponds to the case of directed graphs. Undirected graphs are obtained if we require for each edge $(v_1, v_2)$ an edge $(v_2, v_1)$ in the opposite direction with the same label.

**Definition 5.2**
Given graph $G = (V, E, \mu, \nu, L_v, L_e)$, a sub-graph of $G$ is a graph $S = (V_s, E_s, \mu_s, \nu_s, L_v, L_e)$ such that:

1. $V_s \subseteq V$
2. $E_s = E \cap (V_s \times V_s)$
3. $\mu_s(v) = \begin{cases} \mu(v) & \text{if } v \in V_s \\ \text{undefined} & \text{otherwise} \end{cases}$
4. $\nu_s(e) = \begin{cases} \nu(e) & \text{if } e \in E_s \\ \text{undefined} & \text{otherwise} \end{cases}$

Let $G = (V, E, \mu, \nu, L_v, L_e)$ be a graph with $V = \{v_1, v_2, \ldots, v_n\}$. Then $G$ can also be represented by its adjacency Matrix $M=\{m_{ij}\}_{i,j=1,\ldots,n}$, where $m_{ii} = \mu(v_i)$ and $m_{ij} = \nu((v_i, v_j))$ for $i \neq j$. Apparently, the adjacency
matrix representation of a graph does not take into account loops at a vertex. However, this is not a real restriction as loops can be represented by means of an extended set of vertex labels.

Clearly, the matrix \( M \) is not unique for a graph \( G \). If \( M \) represents \( G \), then any permutation of \( M \) is also a valid representation of \( G \).

**Definition 5.3:**

A \( n \times n \) matrix \( P = (p_{ij}) \) is called a **permutation matrix** if:

1. \( p_{ij} \in \{0, 1\} \) for \( i, j = 1, \ldots, n \)
2. \( \sum_{i=1}^{n} p_{ij} = 1 \) for \( j = 1, \ldots, n \)
3. \( \sum_{j=1}^{n} p_{ij} = 1 \) for \( i = 1, \ldots, n \)

If a graph \( G \) is represented by \( n \times n \) adjacency matrix \( M \) and \( P \) is an \( n \times n \) permutation matrix, then the \( n \times n \) matrix

\[
M' = PM^TP
\]

where \( P^T \) denotes the transpose of \( P \) is also an adjacency matrix of \( G \). If \( p_{ij} = 1 \) then the \( j \)-th vertex in \( M \) becomes the \( i \)-th in \( M' \).

**Definition 5.4:**

Let \( G_1 \) and \( G_2 \) be two graphs and \( M_1 \) and \( M_2 \) their corresponding adjacency matrices. \( G_1, G_2 \) are **isomorphic** if there exists a permutation matrix \( P \) such that:

\[
M_2 = PM_1P^T
\]

Notice that the matrix \( P \) can be understood as a bijective function \( f \) that maps the vertices of \( G_1, G_2 \) and vice versa. That is, \( f(v_j) = v_i \) if \( m_{ij} = 1 \).

We will call both \( P \) and \( f \) a **graph isomorphism** between \( G_1 \) and \( G_2 \) is equivalent to finding a permutation matrix \( P \) for which (2) holds true.

**Definition 5.5:**

Given two graphs \( G_1 \) and \( G_2 \) there is a sub-graph isomorphism from \( G_1 \) to \( G_2 \) if there exists a sub-graph \( S \subset G_2 \) such that \( G_1 \) and \( S \) isomorphic

**Definition 5.6:**

Let \( M = (m_{ij}) \) be an \( n \times n \) matrix. Then \( S_{k \times n}(M) \) denotes the \( k \times m \) matrix that is obtained from \( M \) by deleting rows \( k+1, \ldots, n \) and columns \( m+1, \ldots, n \), where \( k, m, = n \). That is, \( S_{k \times n}(M) = (m_{ij}) i=1, \ldots, k \) and \( j=1, \ldots, m \).
Using the notation introduced in the last definition, not only the concept of graph isomorphism but also sub-graph isomorphism can be described in terms of adjacency matrices. Let $G_1$ and $G_2$ be graphs with adjacency matrices $M_1$ and $M_2$ of dimension $m \times m$ and $n \times n$ respectively, where $m \leq n$. There is a sub-graph isomorphism from $G_1$ to $G_2$ iff there is a $n \times n$ permutation matrix $P$ such that:

$$M_1 = S_{m,n}(PM_2P^T)$$

Thus, the problem of finding a sub-graph isomorphism from $G_1$ to $G_2$ is equivalent to finding the permutation matrix $P$ for which (3) holds [85]. Notice that:

$$S_{m,n}(PM_2P^T)=S_{m,n}(P)M_2\{S_{m,n}(P)\}^T$$

### 5.1.2 Complexity of Graph Matching

Graph matching is considered to be one of the most complex problems in object recognition in computer vision [86]. Its complexity is due to its combinatorial nature. Following the classification of graph matching problems explained in the previous sub-section, as the nature of each of them is different; we will analyze their complexity separately.

**A-Exact graph matching: graph isomorphism**

This whole category of graph matching problems has not yet been classified within a particular type of complexity such as P or NP-complete. Some papers in the literature tried to prove its NP-completeness when the two graphs to be matched are of particular types or satisfy some particular constraints [87], but it still remains to be proved that the complexity of the whole type remains within the NP-completeness at most. On the other hand, for some types of graphs the complexity of the graph isomorphism problem has been proved to be of polynomial type. An example is the graph isomorphism of planar graphs, which has been proven in Hopcroft and Wong [88] to be of polynomial complexity, although the cost of the leading constant also appears to be quite large. As a result, it can be said that this issue remains as an interesting open theoretical problem, although it also encourages researchers to try to find polynomial time solutions for this type of graph matching problems.
**B-Exact sub-graph matching: sub-graph isomorphism**

This particular type of graph matching problems has been proven to be NP-complete in Garey and Johnson [89]. However, some specific types of graphs can also have a lower complexity.

**C-Inexact graph matching: graph and sub-graph homomorphism**

In inexact graph matching, where we have |VM| < |VD|, the complexity is proved in Abdulkader [90] to be NP-complete. Similarly, the complexity of the inexact sub-graph problem is equivalent in complexity to the largest common sub-graph problem, which is known to be also NP-complete.

### 5.2 Graph & Sub-graph Isomorphism Algorithms

There are two basic approaches that past research has taken towards the problem of graph isomorphism. The first approach is based on group-theoretic concepts and the study of permutation groups. In Abdulkader [90], it was shown that there exists a moderately exponential bound for the general graph isomorphism problem [87]. Furthermore, by imposing certain restrictions on the properties of the graphs, it was possible to derive algorithms that have a polynomially bounded complexity. For example, Luks [91] describe a polynomially bounded method for the isomorphism detection of graphs with bounded valence. A method for the computation of the isomorphism of planar graphs is proposed by Köbler [92] that has only a linear time complexity. However, the major drawback of algorithms based on group-theoretic concepts is the fact that there is usually a large overhead and consequently a large constant factor associated with the theoretical complexity.

The second approach to graph and sub-graph isomorphism is more practically oriented and aims directly at developing an algorithmic procedure for isomorphism detection. Most of these algorithms are based on state-space search with backtracking. One of the first publications in this field is the one by Cornil and Gotlieb [93]. A major improvement of the backtracking method was then presented by Ullman [94], who introduced a refinement method which reduces the search space of the backtracking procedure remarkably.
So far, we have only considered the problem of finding a graph or sub-graph isomorphism between two graphs at a time. However, in practical applications there is often a database of graphs, so-called *model graphs*, and a single unknown input graph that must be tested. If the number of graphs in the database is large then the sequential searching testing of each model graph becomes computationally very costly. As a consequence, several systems have been proposed in the past which combine graph or sub-graph isomorphism algorithms with indexing methods. The basic idea of indexing is to use specific and easily computable features of an input graph in order to select a small set of model graphs out of a large database. Some indexing approaches are proposed in [90]. Instead of using an indexing mechanism as a preprocessor to some conventional sub-graph isomorphism algorithm, the database of graphs is transformed into a decision tree. The decision tree is then used to directly and simultaneously index and match the model graphs with the input graph. However, all of the decision tree approaches that have been presented so far are strongly connected to 3D-object recognition and offer no solution to the general graph isomorphism problem.

In this chapter, the method proposed by Messmer and Bunke [95] for graph and sub-graph isomorphism detection, was applied. It has two important features.

1) First, its run time is only quadratic in the number of vertices in the input graph if we neglect the time needed for preprocessing.

2) Secondly, the time complexity is independent of the number of graphs in the database. This method is based on the following idea. We generate the set of all permutations of the adjacency matrix of a model graph and organize them in a decision tree. Different model graphs can be combined into the same decision tree. The decision tree is built from the model graphs in an off-line preprocessing step. At run time, it is used to efficiently determine if there is a sub-graph isomorphism from an unknown input graph to one of the model graphs. The main advantage of the new method is that it is guaranteed to terminate in quadratic time. However, the trade-off for the efficient run time is the size of the decision tree. It contains, in the worst case, an exponential number of nodes.
5.3 A Brief Review of Ullman's Algorithm

For comparison reasons, we briefly review Ullman's method [94]. It can be applied to both graph and sub-graph isomorphism detection. The method is based on backtracking and a refinement procedure.

The input to the algorithm consists of a model graph $G = (V, E, \mu, v, L_e)$ and an input graph $G_1 = (V_1, E_1, \mu_1, v_1, L_1)$. Let $M$ denote the $n \times n$ adjacency matrix of $G$ and $M_1$ the $n \times n$ adjacency matrix of $G_1$. We intend to find all permutation matrices $P$ such that $M = S_{m,m}(P M_1 P^T)$. Note that $S_{i,n}(P)$ is an $i \times n$ permutation matrix that represents a partial matching from the first $i$ vertices of $G$ onto some vertices of $G_1$. If: $S_{i,n}(M_1) = S_{i,n}(P) M(S_{i,n}(P))^T$ then clearly $S_{i,n}(P)$ represents a graph isomorphism from the sub-graph of $G_1$ that consists of the vertices of 1 to $i$ to some sub-graph of $G$.

Ullman's algorithm (see Fig. 5.1 [94]) is based on the idea of finding all sub-graph isomorphism by gradually setting the permutation matrix $(P)$ row by row. From Def. 6.3 in the previous section, we know that each row $K$ in $P$ contains exactly one non-zero entry $P_{k,i} = 1$, while all other elements $P_{k,j}$ of the row $k$ with $j \neq i$ are set 0. The recursive procedure “Backtrack” begins by setting the first element $P_{i1}$ of the top row of $(P)$ to 1 and all other elements in the top row of $P$ to 0. If: $S_{1,n}(P)$ is a partial matching that represents a sub-graph isomorphism, then the procedure “Backtrack” is recursively called again and the second row of $P$ is tentatively set. This process is continued until either $m$ rows of $P$ have been successfully set and a sub-graph isomorphism is found or the condition in step (3.b.ii) is not satisfied. In both cases, the procedure backtracks to the previous level and tries another setting of $P_{k,i}$.

It is important to note that the backtracking procedure can only be applied to two graphs at a time. If More than one model graphs are involved, it has to be called for each of them individually. Therefore, the complexity of the sub-graph isomorphism detection algorithm based on backtracking is linearly dependent on the number of model graphs.
Fig 5.1. Ullman’s Algorithm [94]

\[ Ullman(G = (V, E, \mu, v, L_v, L_e), G_1 = (V_1, E_1, \mu_1, v_1, L_v, L_e)) \]

1. Let \( P = (p_{ij}) \) be a \( n \times n \) permutation matrix, \( n = |V| \), \( m = |V_1| \) and \( M \) and \( M_1 \) denote the adjacency matrices of \( G \) and \( G_1 \), respectively.

2. call Backtrack\((M, M_1, P, 1)\)

3. procedure Backtrack(adjacency matrix \( M \), adjacency matrix \( M_1 \), permutation matrix \( P \), counter \( k \))

   (a) if \( k > m \) then \( P \) represents a subgraph isomorphism from \( G_1 \) to \( G \). Output \( P \) and return.

   (b) for all \( i = 1 \) to \( n \)

      i. set \( p_{ii} = 1 \) and for all \( j \neq i \) set \( p_{ij} = 0 \)

      ii. if \( S_{h, i}(M_1) = S_{h, i}(P)M(S_{h, i}(P))^T \) then

          call Backtrack\((M, M_1, P, k + 1)\)

5.4 The Proposed Sub-graph Isomorphism by Means of Decision Trees

The main problems of the algorithm described in the previous section are:

1) The fact that all permutation matrices which represent a sub-graph isomorphism are calculated and generated at run-time.

2) Furthermore, the algorithm may run into dead ends and backtracking becomes necessary.

In order to overcome these problems and to avoid backtracking at run time, a decision tree based approach is proposed. We assume that there is a set of model graphs that are known a priori, while the input graph becomes accessible at run time only. For each model graph we compute all possible
permutations of its adjacency matrix and transform these adjacency matrices into decision tree. At run time, the matrix of the input graph is then used to find those adjacency matrices in the decision tree that are identical to it. These permutation matrices that correspond to these adjacency matrices represent the graph or sub-graph isomorphism that we are looking for. Let: 

\[ G = (V, E, \mu, v, I_v, I_e) \] 

be a model graph and \( M \) its corresponding \( n \times n \) adjacency matrix. Furthermore, let \( A(G) \) denotes the set of all permuted adjacency matrices of \( G \), 

\[ A(G) = \{ M_P | M_P = PMP^T \text{ where } P \text{ is a } n \times n \text{ permutation matrix} \} \]

The total number of permuted adjacency matrices is \( |A(G)| = n! \) as there are \( n! \) different permutation matrices of dimension \( n \). We are now ready to restate the sub-graph isomorphism problem in terms of the set introduced above. For a model graph \( G \) with corresponding \( n \times n \) adjacency matrix \( M \) and an input graph \( G_I \) with an \( m \times m \) adjacency matrix \( M_I \) and \( m \leq n \), determine whether there exists a matrix \( M_P \in A(G) \) such that 

\[ M_I = S_{m \times m}(M_P) \]

If such a matrix \( M_P \) exists, the permutation matrix \( P \) corresponding to \( M_P \) describes a sub-graph isomorphism from \( G_I \) to \( G \), i.e. 

\[ M_I = S_{m \times m}(M_P) = S_{m \times m}(PMP^T). \]

If \( G_I \) and \( G \) are of equal size, the permutation matrix \( P \) represents a graph isomorphism between \( G_I \) and \( G \), i.e. \( M_I = PMP^T \).

We propose to organize the set \( A(G) \) in a decision tree that each matrix in \( A(G) \) is classified by the tree. It is important to note that the purpose of the decision tree will be to classify adjacency matrices of input graphs.

We introduce a new notation for an \( n \times n \) adjacency matrix \( M = (m_{ij}) \). We say that the matrix consists of an array of so-called row-column elements \( a_i \), where each \( a_i \) is a vector of the form: 

\[ a_i = (m_{1i}, m_{2i}, \ldots, m_{ni}, m_{i(i-1)}, \ldots, m_{ii}) \]

The matrix can then be written as: 

\[ M = (a_1, a_2, \ldots, a_n); \quad i = 1, \ldots, n \]

Fig 5.2 illustrates the structure of an adjacency matrix \( M \) with regard to its row-column elements.
The decision tree is now built according to the row-column elements of each adjacency matrix $M_P \in A(G)$. At the top of the decision tree there is a single root node. The direct successor nodes of the root node constitute the first level of the decision tree. On the first level, the classification of the matrices in $A(G)$ is done according to the first row-column element $a_1$ of each matrix $M_P \in A(G)$. Next, on the second level of the decision tree, the second row-column element $a_2$ of each matrix is used for the classification, and so on. In general, the matrices that are represented by some node on the level $k$ are divided into classes according to the element $a_k$. With each matrix $M_P$ that is represented by some node $N$ on the level $k$, the corresponding permutation matrix $P$ is also given. As $M_P$ has been classified up to the $k$-th vertex, $P$ describes a sub-graph isomorphism for the sub-graph with adjacency matrix $S_{k,k}(M_P)$ to $G$. At run time, $P$ will describe a sub-graph isomorphism for any input graph that has been classified into the node $N$. Finally, at the bottom of the decision tree, there are the leaf nodes. Each leaf node represents a class of identical matrices $M_P \in A(G)$. For each of these matrices, the corresponding permutation matrix is stored in the leaf node. The number of these permutation matrices in each leaf node is equal to the number of automorphisms (isomorphism of a graph to itself).
In Fig. a graph, g1, and its corresponding decision tree is shown. The nodes of the decision tree are represented by shaded circles. Each directed branch from one node to another has associated with it a row-column element. At the top of Fig 5.3, the set A(g1) of permuted adjacency matrices of g1 is listed.

**Indexing**

An important requirement for a decision tree is that the classification on each level must be easily computable. Therefore, if a matrix \( M_P \) is to be classified according to the k-th row–column element \( a_k \), the successor which is reached via an element \( a_{ki} = a_k \) must be easily computable. For this purpose, all row-column elements that are associated to the branches pointing from a node on level \( k \) to a node on level \( k+1 \) are collected in a dictionary of strings. This dictionary is organized as an index structure with \( 2^{k-1} \) indices. There are exactly \( 2^{k-1} \) elements \( mi_j \) in a row-column element in this dictionary can be done in \( 2^{k-1} \) steps. Thus finding the successor node in the decision tree at level \( k \) can be done in \( O(2k - 1) = O(k) \) steps.

**Decision tree for more than one model:**

So far we have only discussed the structure of the decision tree with regard to a single model graph. If there are several model graphs in a database then the most trivial solution would be to build a decision tree individually for each model graph. However, it is possible to represent several graphs by the same decision tree. On each level, the classification of the adjacency matrices for a model graph is done solely on the basis of the current row-column element. In Fig 5.4 [94], the decision tree for the graph g1 of Fig 5.3. and another graph, g2, is displayed. In order to classify each of the adjacency matrices in A(g2). Only two nodes have to be added to the decision tree that corresponds to the graph g1. As there are 3 automorphisms of g2, each of the nodes 13 and 15 in fig. represents 3 adjacency matrices.

At run time, the decision tree is directly used in order to classify the \( m \times m \) adjacency matrix MI of an unknown input graph GI. The matrix
M1 is classified on the first level according to its row-column element $a_{1i}$ matches $a_{1j}$. If there is some branch $i$ from root node to a successor node whose associated element $a_{1i}$ matches $a_{1j}$, the algorithm continues with the successor node to the second level and so on. If at some point no classification is possible, then the input $G_j$ is not isomorphic to any subgraph of the model graphs or any of the model graphs in the database. If each row-column element of $G_j$ has been used in the classification process and some node $N$ in the decision tree has been reached, then each permutation matrix that is associated with $N$ represents a sub-graph isomorphism from the input graph to one of the model graphs. If node $N$ is a leaf node and the input graph and the model graph are equal size then each permutation matrix associated with $N$ describes a graph isomorphism between the input graph and one of the model graphs.
Let $N_1$ and $N_2$ be nodes of the decision tree that both represent the same sub-graph $S$ of a model graph $G$ and let $S$ be given by its adjacency matrix $M_S$. Furthermore, let $M_1$ and $M_2$ be the adjacency matrices represented by $N_1$ and $N_2$, and $P_1$ and $P_2$ the corresponding permutation matrices such that $P_1 M_S P_1^T = M_1$ and $P_2 M_S P_2^T = M_2$. Then, there exists a permutation matrix $R$ such that:

$$M_1 = RM_2 R^T$$ (1)
R can be simply obtained by
\[ R = P_1P_2^T \]  
(2)
because substituting R in (1) by \( P_1P_2^T \) and also substituting M2 by \( P_2M_2P_2^T \) yields
\[ P_1P_2^T(P_2M_2P_2^T)(P_1P_2^T)^T = P_1M_2P_1^T = M_1. \]
Therefore, any adjacency matrix represented in the node N2 can be transformed, by means of the matrix R, into a matrix that is represented in N1.

### 5.5 A More Efficient Representation of Decision Trees

The decision trees described in the last section are unnecessarily large. In this section we introduce a more compact representation. It is based on the observation that at level (k) in the decision tree all sub-graphs consisting of (k) vertices are represented, k=1… n. Notice that each of these sub-graphs (S) is represented (k! /h) times, where (h) denotes the number of different automorphisms of (S). Clearly, all these representations are equivalent to each other, and the information they contain is largely redundant. We now show how this kind of redundancy can be avoided.

Let N1 and N2 be nodes of the decision tree that both represent the same sub-graph S of a model graph G and let S be given by its adjacency matrix \( M_S \). Furthermore, let M1 and M2 be the adjacency matrices represented by N1 and N2, and P1 and P2 the corresponding permutation matrices such that
\[ P_1M_2P_1^T = M_1 \] and
\[ P_2M_2P_2^T = M_2. \]
Then, there exists a permutation matrix R such that:
\[ M_1 = RM_2R^T \]
R can be simply obtained by
\[ R = P_1P_2^T \]
because substituting R by \( P_1P_2^T \) and also substituting M2 by \( P_2M_2P_2^T \) yields
\[ P_1P_2^T(P_2M_2P_2^T)(P_1P_2^T)^T = P_1M_2P_1^T = M_1. \]
Therefore, any adjacency matrix represented in the node N2 can be transformed, by means of the matrix R, into a matrix that is represented in N1.

The most important conclusion from this observation is that for the decision tree node N2 it is not necessary to classify the represented matrices further and create successor node N2. Instead, it is sufficient to
classify the matrices represented in N1 and simply refer or redirect the
node N2 to N1. For this purpose, a "redirecting branch" concept was
introduced in the decision tree that originates at N2 and ends in N1.
Associated with the redirecting branch is the permutation matrix (R).
We now describe a procedure for decision tree size optimization
(Optimize_Tree) see Fig 5.5.

<table>
<thead>
<tr>
<th>Procedure Optimize_Tree (tree T with k levels)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Begin</strong></td>
</tr>
<tr>
<td>for each level i:1 to k</td>
</tr>
<tr>
<td>If there is a tree node (N1) and a tree node (N2) at the same level such that:</td>
</tr>
<tr>
<td>$M_1 = R M_2 R^T$ where R is a permutation matrix.</td>
</tr>
<tr>
<td>Then</td>
</tr>
<tr>
<td>1.1 Create a &quot;redirecting branch&quot; that originates at N2 and ends at N1.</td>
</tr>
<tr>
<td>1.2 save R in an index.</td>
</tr>
<tr>
<td><strong>End</strong></td>
</tr>
</tbody>
</table>

Fig 5.5: the (Optimize_Tree) procedure.
In Fig 5.6 the decision tree for the graphs g1 and g2 resulting from the procedure Optimize_Tree is displayed. There are two redirecting branches in this decision tree (denoted by dotted lines). Associated with branches is the permutation matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. When applied to the 2x2 submatrices $S_{2,2}(A)$, $S_{2,2}(B)$ and $S_{2,2}(D)$ of g1, the matrices $S_{2,2}(C)$, $S_{2,2}(E)$ and $S_{2,2}(F')$ result. For the graph g2, the redirecting branch transforms the matrices $S_{2,2}(A')$, $S_{2,2}(D')$ and $S_{2,2}(E')$ into the matrices $S_{2,2}(C')$, $S_{2,2}(F')$ and $S_{2,2}(B')$. Notice that by introducing redirecting branches into the decision tree it is possible to save three decision tree nodes.
5.6 Traversal of Decision Tree

The decision tree structure that was previously described can now be used to get a very efficient graph and sub-graph isomorphism algorithm. Let $G_1$, $G_2$… $G_I$ be a set of model graphs represented by a decision tree and $G_I$ an unknown input graph. We assume that the input graph is represented by its adjacency matrix $M_I = (a_1, a_2, \ldots, a_m)$ given in row-column format. Fig 5.7, illustrates the (decision tree) procedure. This procedure tries to find out whether there exists a matrix $M \in A(G_i)$ such that $M_I = S_{m \times n}(M)$ by classifying $M_I$ according to its row-column elements. The algorithm starts at the root node of the decision tree and first classifies $M_I$ according to its first element $a_1$. If this step is successful, the classification is continued on the next level. In general, if the process is on level $K$ and $N$ is the current node of the decision tree, then the successor node of $N$ which represents perfectly, the process follows the branch from $N$ to the successor node $N_s$ that represents the row-column element $a_{k \times n}$. If no such element can be found in the dictionary then $M_I$ can't be classified by the decision tree and it follows that $G_I$ is not isomorphic to any sub-graph of the model graphs $G_I$.

In step (2-d) of the algorithm it is checked whether the current node $N_s$ has an outgoing redirecting branch. If this is the case then we follow this redirecting branch. Accordingly, the matrix $M_I$ of the input graph must be permuted by applying the permutation matrix $R$ that is attached to the redirecting branch. Note, however, that if $N$ is on level $K$ than $R$ is $k \times k$ permutation matrix because it was created at compilation time for a sub-graph of size $k$. $M_I$ an $m \times m$ adjacency matrix of the input graph $G_I$ with $m \geq k$. In order to apply $R$ to $M_I$, it is necessary to extend $R$ to an $m \times m$ matrix ($R^m$) by copying rows and column from $m \times m$ identity matrix.

Termination conditions:

1) The algorithm terminates in step (2-b) when it is detected for the first time that there is no sub-graph isomorphism from the input graph to any of the model graphs.

2) Also, in step (3) when the last row-column element $a_{m}$ of $M_I$ has been processed and some node $N$ has been reached.
In the latter case, the matrix $M_i$ is identical to all matrices $M_i$ of the model $G_i$ that are represented in $N$.

If $N$ is not a leaf node then the set of permutation matrices that are stored in $N$ represents all sub-graph isomorphism from the graph $G_f$ to $G_i$.

If, on the other hand, $N$ is a leaf node and $G_f$ and $G_i$ are of equal size then the set of permutation matrices in $N$ represents all graph isomorphism between $G_f$ and $G_i$.

Important Notes
1) It is easy to see that the new algorithm for graph isomorphism traverses the decision tree without the need for backtracking and therefore has a time complexity which is only polynomial in the number of vertices of the input graph.
2) The algorithm is clearly independent of the number of model graphs that are represented in the decision tree.
**Decision Tree** (Node Root, Graph $G_f$

1. Let $G_f$ be given by its adjacency matrix $M_f = (a_{1}, \ldots, a_{m})$ and let $m = |V|$ and $N = \text{Root}$.

2. For $k = 1$ to $m$ do

   (a) Look up the dictionary of row-column elements that is attached to node $N$ and find an entry $a_{x_k}$ such that $a_{x_k} = a_k$.

   (b) If no such element in the dictionary is found, the graph $G$ is not isomorphic to any subgraph of the model graphs or any of the model graphs themselves represented by the decision tree. Exit with failure.

   (c) If an element $a_{x_k} = a_k$ is found in the dictionary, then follow the branch marked by $a_{x_k}$ to the node $N_x$.

   (d) If there is a redirecting branch from $N_x$ to some node $N'$ then set $M_f = R' M_f R'^T$, where $R'$ is the extended $m \times m$ permutation matrix associated to the redirecting branch (see text). Set $N = N'$.

3. For each matrix $M_f$ of a model $G_f$ (with corresponding $n \times n$ adjacency matrix $M_f$) that is represented by the node $N$

   (a) If $m < n$ then the associated permutation matrix $P$ describes a subgraph isomorphism from $G_f$ to $G_i$, i.e. $M_f = S_{m,n}(M_f) = S_{m,n}(P M_f P^T)$.

   (b) If $m = n$ then the associated permutation matrix $P$ describes a graph isomorphism from the input $G_f$ to the graph $G_i$, i.e. $M_f = P M_i P^T$.

---

**5.7 ASL Gestures and Graph Matching**
The main objective of this research is the recognition of connected gestures level. Such that, a whole sentence represents the input for the designated system. While the scope of this work is not to create a user independent, full lexicon system for recognizing ASL, but it should be extendible. After the classification of static postures, the next stage is how to collect these discrete postures to define a word meaning. Then it follows translating a whole sentence.

Work in continuous sign recognition has used Hidden Markov Models (HMMs) and Artificial Neural Network (ANN) [97, 98]. In Table 5.1 we categorize the prior work in recognition of continuous words, embedded in short sentences, in terms of the type of input data, size of experimental data set, technique used, and reported recognition rates. Starner and Pentland [99] were the first to seriously use HMMs for continuous sign recognition. Their HMMs had 4 states with one skip transition and multidimensional, independent, Gaussian observations. With these they achieved near perfect recognition with sentences of fixed structure, i.e. containing personal pronoun, verb, noun, adjective, personal pronoun, in that order. Vogler and Metaxas [100] have been instrumental in significantly pushing the state-of-the-art in automated ASL recognition using HMMs. In terms of the basic HMM formalism, they have explored many variations, such as context dependent HMMs, HMMs coupled with partially segmented sign streams, and parallel HMMs. One of the very exciting lines of work suggested by them to tackle the scalability problem of HMMs is to design systems to recognize "cheremes", the `phonemes' of American Sign Language, instead of the words [100]. Cheremes differ with respect to hand shapes, hand orientation, wrist orientation, location, and movement. They extracted these features using 3D magnetic tracking systems. To control the combinatorial explosion of possible states, they assume independence of the attributes characterizing the cheremes and using separate HMMs for each channel, i.e. parallel HMMs. Christian Vogler and Dimitris Metaxas [100] present a novel approach to ASL recognition that aspires to being a solution to the scalability problems. It is based on parallel HMMs (PaHMMs), which model the parallel processes independently. Thus, they can also be trained independently, and do not require consideration of the different combinations at training time. They run several experiments with a 22 sign vocabulary and demonstrate that
PaHMMs can improve the robustness of HMM-based recognition even on a small scale.
For ANN, m.v.Lamar [101] suggested The T-CombNET model which is inspired by Iwata’s CombNET-II model [102]. The use of Stemend Branch networks in a layered structure allows fast convergence in problems with large number of categories. The system was applied to an user independent, 34 classes, Japanese finger spelling recognition problem. The obtained results show recognition rate of 91.2%. Murakami and Taguchi [103] describe a Dataglove system using recurrent neural network. The system achieves up to 98% recognition for trainers of the system and 77% for users not seen before.

**Notices**
1) It can be noticed that most of the work in continuous sign language recognition has avoided the very basic problem of segmentation/tracking of hands by using wearable devices, such as colored gloves, data gloves, or magnetic markers, to directly get the location features.
2) It is noticeable that, most of these systems are not extendible, such that when it is required to extend the database, the whole system needs to be re-trained.
3) The features used in each system can't be generalized to all languages. The used features can only be applied to proposed sign language.

### 5.8 Graph Construction & Tree Traversal Algorithm Customization

The main idea is to use the graph matching problem and algorithm as suggested solution for connected gestures classification. The gestures which represent alphabets or words are stored in database as models graphs. Each graph consists of a group of vertices and edges. These graphs are attributed & directed graphs. The connected flow of input gestures is represented by the input graph. Fig 5.8 illustrates the main steps.
We now face two main challenges:

- Firstly, is how to construct the graphs and to discard the transitional movements inherent to the finger spelling or dynamic gesture?
- Secondly, look for a customization methodology to the Tree_Traversal procedure to suit the dynamic gesture re-construction problem.

### 5.8.1 Graph Construction

Three approaches were studied in handling this problem. The first approach ignores this problem and analyzes every input frame; the second one filters the input postures after posture classification module. The last one was suggested by Messmer [94] who mainly filters the input frames before posture classification module.

**A- First Approach:**

This approach mainly ignores this problem. It gives each image (posture) a distinct vertex in the constructed graph. In this approach, unknown" posture class appears. This class represents the undetermined postures.

**Graph properties:**

1) Each vertex contains a single attribute which is the posture class.
2) The graph is directed and unlabeled.
3) The movement postures (Transitions) represent loops on vertices, or represent an unknown posture class vertex. Fig 5.9 illustrates a gesture which expresses a person who counts from 1 to 2 and the constructed graph.

![Diagram of a gesture expressing counting from 1 to 2 and its associated graph.]

**B- Second Approach**

In this approach the time segmentation is performed using a simple approach; we discard the transitional movements before posture classification. An analysis of the amount of movement of the scene based on the energy of the 2 consecutive frames difference is performed in order to classify the incoming frame as transitional frame or final static frame. This approach was suggested by Lamar [45][46]. We define the frame difference energy function as

\[ E_d(f_i, f_{i-1}) = \sum_{l=1}^{nl} \sum_{c=1}^{nc} (f_i(l, c) - f_{i-1}(l, c))^2 \]

where \( f_i(l, c) \) is the pixel value for the i-th frame on the position (line,column). Given the condition

\[ E_d(f_i, f_{i-1}) \geq E_{th1}. \]

Where \( E_{th1} \) is a fixed threshold value. If the energy of frame difference satisfies the condition, then it is selected as a dynamic frame, so it is not analyzed. If the condition fails, it is considered a static frame, and then must be processed to recognize the hand posture.

**c- Third Approach**
This approach mainly depends on filtering the hand postures after hand classification module. The filtering criterion is based on:

1) The output class of the posture.
   This feature represents the output class from the previously applied hand classification module. Also the unknown class exists.

2) The relative position of the hand.
   This feature represents the relative hand position. The frame is pre-segmented to 4 regions as illustrated in fig. These regions are labeled as \{A, B, C, and D\}. Any static posture lies into one of these regions as illustrated in Fig 5.10.

\[
\begin{array}{c|c|c|c}
\text{A} & \text{D} \\
\text{B} & \text{C} \\
(0,0) & (x, y) \\
\end{array}
\]

Fig 5.10 : the main 4 regions of the frame

**Graph Construction**

The graph consists of a set of vertices. The input posture sequence is filtered according to the previously mentioned criteria. The given posture is considered a new vertex in the following cases:

1) The new posture (Pnew) and the previous posture (Pold) have a different posture classes.

2) The new posture (Pnew) and the previous posture (Pold) are in different region.
   See Fig 5.11.
Fig 5.11: 4 examples for 2 hand postures with different 2 hand posture classes and different regions and their corresponding graphs in order.

The three methods have been applied for the dataset. The third technique shows better time results. The results and comparison of the three methods are shown in details in the experimental results chapter.

After graph construction is performed. The decision-tree based algorithm is applied to detect the graph and sub-graph isomorphism.
5.8.2 Traversal Algorithm Customization

The algorithm proposed by B.T Messmer and H. Bunke described in section 5.6 should be customized to suit the given application. To accomplish this, we have two points to cover:

1) The algorithm assumes that, the input graph is smaller than or equal to any model graph in the database. In other words, the procedure looks for sub-graph isomorphism from input graph to models graphs. While in the current application, the input graph represents an input sentence and the model graphs are isolated gestures. In other words, we need to discover the contrary, sub-graph isomorphism from models graphs to input graph.

2) The algorithm stops after finding one sub-graph isomorphism. While the application needs to discover all sub-graphs isomorphism with all models graphs.

The customization idea depends on changing the termination condition of the algorithm. Such that, it stops when the input sentence is recognized. Fig 5-13 illustrates the modified algorithm. The modification should cover 3 different cases as illustrated in Fig. 5-12.

1) No gestures were recognized.
2) The input sentence is equivalent to one sentence (Graph Isomorphism).
3) The input sentence corresponds to a sequence of gestures. (Multiple Sub-Graphs Isomorphism).
Fig 5.12: The possible cases of the modified procedure output

- No graph or subgraph isomorphism were found: the input sentence could not be recognized
- Graph Isomorphism was found: the input sentence is one gesture
- Sub-graphs Isomorphisms were found: the input sentence is a sequence of gestures
Modified Decision Tree Procedure

1. Let \( \mathcal{G}_I \) be given by its adjacency matrix \( M_I = (a_{ij}, \ldots, a_{mn}) \) and \( \mathcal{N} = \text{Root} \). Let No Of Subgraphs=0.

2. For k=1 to m
   (2.a) Look up the dictionary of row-column elements that is attached to Node \( \mathcal{N} \) and find an entry \( a_{k,n} \) such that \( a_{k,n} = a_{k} \).
   (2.b) While the current node has a successor branch
      2.b.1 If no such element in the dictionary is found, go to (2.a).
      2.b.2 If an element \( a_{k,n} = a_{k} \) is found and \( \mathcal{N} \) is not a leaf node, then follow the branch marked \( a_{k,n} \) to the node \( \mathcal{N} \), k=k+1.
      2.b.3 If an element \( a_{k,n} = a_{k} \) is found and \( \mathcal{N} \) is a leaf node (sub-graph isomorphism is found), then:
         No Of Subgraphs= No Of Subgraphs+1
      2.b.4 If there is a redirecting branch from \( \mathcal{N} \) to some node \( \mathcal{N}' \), then set \( M' = R'MR'^T \).
   End while
   (2.c) k=k+1, and mark the previous part of the input graph as recognized.

End for.

3. If No Of Subgraphs=0, then no sub-graph isomorphism found.
4. Else if No Of Subgraphs=1, then a graph isomorphism found.
5. Else if No Of Subgraphs >1, then sub-graphs isomorphisms are found.

In step (2-b) an inner while loop was added. This loop finds out if there exists one sub-graph isomorphism. In step (2-b.3) a leaf node was reached, then a sub-graph isomorphism exists, increment the No Of Subgraphs by 1.
In step (2-c), go to the next input graph node, and mark the previous input graph part as recognized. After the end of the external for loop (end of input graph), check for the value of No_Of_Subgpaphs. Such that, if No_Of_Subgpaphs still takes 0, then either graph or sub-graph isomorphism were found. Else if No_Of_Subgpaphs takes 1 then a graph isomorphism was found. It means that the input sentence is equivalent to one gesture. Else if No_Of_Subgpaphs takes a value more than 1, then sub-graphs isomorphisms were found. It means that the input sentence corresponds to a sequence of gestures.
5.9 Experimental Results

In this section, one illustrates the results of the gesture construction module; a new decision tree-based sub-graph isomorphism algorithm is customized and implemented. The experiment contains 30 connected sentences of total 100 words. First, a study is conducted to measure the effect of graph construction approaches on the decision tree size in terms of nodes count using the 3 approaches (graph construction section). Figure 5.14 illustrates the size of the offline built decision tree against the size of the graph models database size.

![Figure 5.14: the tree size against model graphs database size](image)

It is clear that applying the 3rd approach leads to a minimum tree size relative to the other approaches. The 1st approach gives the maximum tree size because it ignores the transitional movements; it causes that each frame represents a distinct vertex in the gesture graph. The equation used in the 2nd approach decreases the tree size because it omits transition frames. Meanwhile, it does not accomplish the minimum size because of its relative nature of the threshold values. The performance is then measured by: computing the execution time in seconds and counting the number of basic computation steps that are performed while searching for all graph and sub-graph isomorphism. A basic computation step is defined as the comparison of one model graph vertex and its incident edges to one input graph vertex and its incident edges. Figure 5.15 illustrates the system performance measures using execution time against model graphs database size. Moreover, the computational steps needed against the decision tree nodes count.
Figure 5.15- a) Execution time in sec against the database size. B) Computational steps against number of tree nodes.

The performance measurements emphasizes the polynomial nature of the proposed customized sub-graph isomorphism algorithm; it concludes that as much as the gestures graphs database size increases, the running time will be still bounded by a polynomial. As mentioned before, 30 connected sentences are tested using a graph database containing 100 dynamic gestures (words). The recognition accuracy of the proposed system is measured using the number of 3 words-sentences that have been successfully translated and the number of 4 words- sentences that have been successfully recognized. Table 5.1 illustrates the recognition accuracy of both 3 and 4 words sentences.
Table 5.1 illustrates that the recognition accuracy of 30 sentences does not lay down 19 complete successfully translated sentences. Besides, it reveals the reality that as much as the database size increases, the recognition accuracy decreases. This decrease is a result of adding models graphs that are very similar, which causes misclassification of the input sentences.

<table>
<thead>
<tr>
<th>Graph database size</th>
<th>Sentences composed of 3 words</th>
<th>Sentences composed of 4 words</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Totally successfully translated</td>
<td>Sentences with single error</td>
</tr>
<tr>
<td></td>
<td>misclassified</td>
<td>unclassified</td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>24</td>
<td>3</td>
</tr>
<tr>
<td>80</td>
<td>21</td>
<td>5</td>
</tr>
<tr>
<td>100</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>26</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>40</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>80</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>19</td>
<td>3</td>
</tr>
</tbody>
</table>
Chapter 6

Proposed Recognition Error Detection and Correction

Previous developed systems suffer from "controlled-environment" constraint, if one of the control assumptions is violated, the recognition accuracy dramatically decreases. In this paper, a post-processing module based on Natural Language Processing (NLP) rules is proposed to detect and correct expected errors resulting from recognition system. Popular applications for example, Optical Character Recognition (OCR), handwritten recognition, speech recognition, etc… have been researched to increase the accuracy of the recognition using NLP rules. But previous sign language recognition researchers have never explored this concept. We suggest a new hybrid semantic-oriented approach which can correct semantic level errors as well as lexical errors, and is more accurate for especially domain-specific sign language recognition error detection and correction. Through extensive experiments, it will be demonstrated the better performance of the proposed post-processing approach. The experiments were done on Arabic sign language recognition and it can be generalized to other sign languages.

6.1 NLP in Recognition Systems

There are increasing practical applications that use natural language processing (such as, Machine Translation, Semantic Web applications, cross-lingual information retrieval, etc.)[104, 105]. Also natural language knowledge has been exploited to enhance the recognition accuracy of some applications (such as, Optical Character Recognition, Speech Recognition, Natural Language Based Querying, etc.) [106, 107].

Ringger and Allen [108] suggested the noisy channel model for speech error correction. They simplified a statistical machine translation (MT) model called IBM statistical MT model [109]. They tried to construct a general post-processor that can correct errors generated by any speech recognizer. The model consists of two parts: a channel model, which accounts for errors made by the ASR, and the language model, which
accounts for the likelihood of a sequence of words being uttered in the first place. They trained the channel model and the language model both using some transcriptions from TRAINS-95 dialogue system which is a train traveling planning system [110]. Here, the channel model has the distribution that an original word may be recognized as an erroneous word. Kaki et al. [111] suggested another approach, that is, a straightforward and intuitive method to robustly handle many kinds of recognition errors. They collected many error patterns that occurred in a speech translation system, and constructed also a corpus consisting of a general word string from that domain. They could correct any type of errors by matching the strings in the transcription with error patterns in the database. However, their approach has a disadvantage in that they are only feasible to the trained (or collected) error patterns, hence if the domain of the application is changed, the system must be trained again from the start, which is time and money consuming. In some similar areas such as spelling error correction or optical character recognition (OCR) error correction, NLP researchers traditionally identified five levels of errors in text: (1) a lexical level, (2) a syntactic level, (3) a semantic level, (4) a discourse structure level, and (5) a pragmatic level [112]. In spelling correction and OCR error correction problem, correction schemes mainly focused on non-word errors in lexical level. However, errors of speech recognition are real-word errors which should be classified into syntactic and semantic level errors, because a recognizer produces word sequences existing in lexicon. The previous works focused on lexical level error correction, thus may not be appropriate to be applied to different domains. None of these approaches has been applied to sign language recognition due to some problems:

- The sign language depends in some meanings on face expressions and body language [5].
- The sign language has no grammatical rules with its sense in NLP, in natural languages, the words must meet a specific order in the sentence, and meanwhile in Arabic sign language for example the word order is irrelevant.

### 6.2 A Proposed Error detection Semantic Oriented Approach
This section presents a semantic-oriented approach to correct erroneous outputs of a Arabic sign language recognizer using domain knowledge. We focus on real-word error detection and correction, so our approach is based on the domain knowledge for performing semantic level error detection.

6.2.1 Lexico-Semantic Pattern

A lexico-semantic pattern (LSP) is a structure where linguistic entries and semantic types are used in combination to abstract certain sequences of the words in a text. It has been used in the area of natural language interface for database (NLIDB) [113] and a TREC QA system for the purpose of matching the user query with the appropriate answer types [114, 115]. In an LSP, linguistic entries consist of words and phrases. Semantic types consist of common semantic classes and domain specific (or user-defined) semantic classes. In domain-specific application, well defined semantic concepts are required, and the domain-specific semantic classes represent these requirements. The words in a query sentence are converted into the LSP through two steps. First, a morphological analysis is performed, which segments a sentence of words into morphemes. Next, each morpheme of the sentence is converted into a suitable semantic symbol by searching several types in the semantic dictionaries.

6.2.2 Domain Knowledge Construction

The domain information is reflected with template database: LSP-converted sentences of the source statements which are used for the actual error detection task after signs recognition. The template entries are automatically acquired by the Sentence-to-LSP translation from the source statements using two semantic category dictionaries: domain dictionary and ontology dictionary. The domain dictionary is a subset of the general semantic category dictionary, and focuses only on the narrow extent of the knowledge it concerns, since it is impossible to cover all the knowledge of the world in implementing an application. On the other hand, the ontology dictionary reflects the pure general knowledge of the world; hence it performs a supplementary role in extracting semantic information. The domain dictionary provides the specific vocabulary which is used at semantic representation task of a user sentences and the template database. Assuming that some sign statements for a specific target domain is
predefined; a record of the template database is made up of a fixed number of LSP elements, such as semantic tags, and domain-specific semantic classes. Sentence-to-LSP transforms given sentence into corresponding LSP, and the LSP’s enhance the coverage of extraction by information abstraction through many to-one mapping between queries and an LSP. The transformation consists of two phases: Named entity (NE) recognition and NE tagging [116]. NE recognition discovers all the possible semantic types for each word by consulting a domain dictionary. Figure 6.1 is a schematic diagram of the post error detection process.

Figure 6.1: The error detection diagram

### 6.3 A Proposed Statistical Error Correction Approach

The problem of word correction can be stated as follows:
Let L={W1,W2,…,Wm} be the set of all the words in a given, for an input sentence, S=S1, S2, … Sn produced as the output of sign language recognition. Find the best word sequence Ŵ=W1, W2… Wn, for Wi ∈ L, that maximizes the probability Pr (Ŵ|S):
\( \hat{W} = \arg\max_w (\Pr(W|S)) \)

Using Bayesian formula, we can rewrite it as:

\[
\hat{W} = \arg\max_w (\Pr(W|S)) = \arg\max_w \left( \frac{\Pr(W) \times \Pr(S|W)}{\Pr(S)} \right) \text{ since the evidence probability is constant for all input sentences, it can be ignored.}
\]

The prior probability \( \Pr(W) \) is given by the language model and can be computed as

\[
\Pr(W) = \prod_{i=1}^{n} \Pr(W_i|W_{i-1})
\]

Where \( \Pr(W_i|W_{i-1}) \) is the probability that the word \( W_i \) appears given that \( W_1, W_2, \ldots, W_{i-1} \) appeared. We assume that a word \( W_i \) will appear is affected only by the immediate preceding word. Thus:

\[
\Pr(W_i|W_{i-1}) = \Pr(W_i|W_{i-1}) \quad \Pr(W) = \prod_{i=1}^{n} \Pr(W_i|W_{i-1})
\]

The likelihood probability \( \Pr(S|W) \) reflects the characteristics of the preceding classifier results, we assume that the words produced under the recognition step are independent, Thus:

\[
\Pr(S|W) = \prod_{i=1}^{m} \Pr(S_i|W_i)
\]

We conclude that:

\[
\hat{W} = \arg\max_w \left( \prod_{i=1}^{m} \Pr(S_i|W_i)^* \Pr(W_i|W_{i-1}) \right)
\]

The problem of calculation \( \hat{W} \) is reduced to calculate the word-bigram probability \( \Pr(S_i|W_i) \) and the word confusion probability \( \Pr(W_i|W_{i-1}) \); the word-bigram probability can be computed from a specific domain lookup table.
We estimate the word confusion probability by computing the distance between the already classified word and the second candidate word in the classification result. Figure 6.2 illustrates the schematic diagram for the error correction steps.

After the error detection module outputs an ill-LSP result, the error correction module starts with the classifier results and finds another candidate sequence of words to be checked again. This loop continues till the error detection results a correct sequence of words, Fig 6.3.

![Figure 6.3: the schematic diagram of error detection and correction modules interaction](image)

The question becomes now, in the back iteration step, using the original sequence and choosing the second maximum combination is better than using the updated sequence? Both solutions have been implemented and the results are discussed in the experimental results section.

### 6.4 Case study and Experimental Results

The experiment was conducted on the domain of clothing point-of-sale, starting with collecting 100 sentences that are used as queries and responses, 70 sentences were used as training and build the template database and 30 were used for testing. We used the Arabic sign language recognition system described in chapter 5. The first step is to create the
template database for this specific domain; this database consists of valid LSP in this domain and was constructed by Sentence-to-LSP module. Table 6.1 illustrates a sample for template database.

Table 6.1: Template database sample

<table>
<thead>
<tr>
<th>@ Color % Degree % What_Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>% What_Question % Degree @ Color</td>
</tr>
<tr>
<td>% How_Much_Question % Cost</td>
</tr>
<tr>
<td>% Cost % How_Much_Question</td>
</tr>
</tbody>
</table>

Note that we inserted in database all possible combinations of words order to overcome the sign language non-grammatical nature issue. The second step is to evaluate the performance of error detection approach. Two main indicators can be computed to measure the error detection performance, the true positive percentage and false positive percentage. The true positive indicates the success of the system in discovering an ill LSP

\[
TP\% = \frac{\text{No of true discovered ill LSP}}{\text{No of Sentences containing ill LSP}}
\]

Meanwhile the false positive indicates the response of error occurrence while the LSP is correct.

\[
FP\% = \frac{\text{No of false discovered ill LSP}}{\text{No of Sentences does not contain ill LSP}}
\]

Table 6.2 illustrates the TP and FP percentages results with different data sample sizes.

Table 6.2: the error detection performance

<table>
<thead>
<tr>
<th>No of Sentences</th>
<th>No of Sentences contains ill-LSP</th>
<th>No of Sentences does not contain ill-LSP</th>
<th>TP %</th>
<th>FP %</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>5</td>
<td>15</td>
<td>100%</td>
<td>7%</td>
</tr>
<tr>
<td>50</td>
<td>15</td>
<td>35</td>
<td>87%</td>
<td>8%</td>
</tr>
<tr>
<td>75</td>
<td>25</td>
<td>50</td>
<td>88%</td>
<td>10%</td>
</tr>
<tr>
<td>100</td>
<td>40</td>
<td>60</td>
<td>87.5%</td>
<td>10%</td>
</tr>
</tbody>
</table>
The results show that the TP percentage does not get beneath 87% and FP percentage does not get higher than 10%. The FP and TP occur because of the unknown patterns that result from the classifier step.

Next step is to evaluate the performance of the correction module using 2 approaches: using the original sentence in every round and using the updated sentence in each round. The results will be evaluated using the Correction Percentage and the average number of iterations between error detection and correction modules.

\[
\text{Correction Percentage} = \frac{\text{Number of successfully corrected sequence}}{\text{Number of all sequences with ill LSP}}
\]

\[
\text{Average Number of Iterations} = \frac{\sum \text{All sequences} \times \text{Number of iterations}}{\text{Number of all sequences with ill LSP}}
\]

Table 6.3 illustrates a sample for the correction for an ill-LSP.

<table>
<thead>
<tr>
<th>Input Sentence from classifier (Arabic)</th>
<th>كيف سعر هذا القميص؟</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Sentence from classifier (English)</td>
<td>How is the Price of this T-Shirt?</td>
</tr>
<tr>
<td>Converted LSP</td>
<td>%How %Price % T-Shirt</td>
</tr>
<tr>
<td>Corrected Statement (Arabic)</td>
<td>كم سعر هذا القميص؟</td>
</tr>
<tr>
<td>Corrected Statement (English)</td>
<td>HowMuch is this T-Shirt?</td>
</tr>
</tbody>
</table>

Table 6.4 illustrates the Correction Percentage and Average Number of Iterations for input ill-LSP for different data sample sizes.
Table 6.4: The correction Module Performance

<table>
<thead>
<tr>
<th>No of ill-sentences</th>
<th>Approach (1) Using the original sentences</th>
<th>Approach (2) using the updated sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correction Percentage</td>
<td>Average Number of Iterations</td>
</tr>
<tr>
<td>10</td>
<td>100%</td>
<td>1.8</td>
</tr>
<tr>
<td>20</td>
<td>95%</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>86%</td>
<td>2.2</td>
</tr>
<tr>
<td>40</td>
<td>82.5%</td>
<td>2.4</td>
</tr>
<tr>
<td>50</td>
<td>86%</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Table 4 illustrates that using the updates sentence in every iteration, gives better correction ratio and lower average number of iterations. This comes from the fact that using the updated sentence every time contains an accumulated knowledge from previous non-successful corrections.

The last evaluation step is to evaluate the hybrid model (error detection and error correction) as one unit and measure its added value to the recognition system. Due to the non-existence of previously developed sign language recognition that uses semantics to enhance the recognition, a comparison was accomplished between the recognition system only and the recognition system after adding this enhancement. The Arabic sign language recognition system that was proposed in chapter 5 is used to establish this unbiased comparison. The dataset used in this comparison is the same 100 collected sentences. Table 6.5 illustrates the recognition accuracy obtained before and after the semantic enhancement.

Table 6.5: the recognition accuracy before and after enhancement

<table>
<thead>
<tr>
<th>System</th>
<th>Before enhancement</th>
<th>After enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognition accuracy</td>
<td>77.5%</td>
<td>96%</td>
</tr>
</tbody>
</table>

The proposed model as mentioned was developed and tested for Arabic sign language for clothes point of sale specific domain, but it can be adapted to other domains by replacing the specific domain Ontology and language model look up table. Also, its main architecture can be used for other sign languages by replacing the Ontology domain. The main problem of the proposed model is that it assumes the same number of words in the
uncorrected and corrected sequence so it allows only replacement, in future it can be extended to allow addition or deletion of words.
Chapter (7)
Conclusion

The hand gesture recognition is a very promising field of study. It can be used to the development of more user friendly human-machine interface. Its use in automatic Sign Language recognition has a strong humanitarian appeal, and can contribute to increase the interaction between people.

This work, initially explored different features generation methods. Different approaches have been developed in feature generation using PCNN. They could not achieve the invariance against light conditions or bright background. The results of experiments of the proposed approach confirmed that the usage of continuity factor as an enhanced form of feature generation. The proposed feature generation approach permits the calculation of sufficiently long image signature with minimal interference effect. While the long output image signature keeps the interference in its minimum, it gives more detailed features needed in large background or non-uniform lighting cases. It was applied to ASL system which recognized the Arabic sign static Alphabets and achieved 90% recognition accuracy, and showed invariance against geometrical transforms, bright background and light conditions. The system failed to differentiate between some postures that have undistinguishable frontal view. This research finally concludes that this recognition accuracy can't be enhanced more than the previously mentioned percentage and this is because a single view is not enough to distinguish the remaining 10% of the postures. This problem was the main motivation to propose a 3D recognition model based on multiple synchronized views from different source cameras. The developed system proposed and used two perpendicular cameras which give the frontal and side views. The PCNN calculates the image signature for the two simultaneously captured images, each of which represents 2D features. The two signatures are weighted and summed to produce 3D image features. Posture classification produced recognition accuracy more than 90% of 158 postures which build ASL one and two hands postures dictionary. The posture classification module requires that object percentage in the frontal view is more than 40%.

But an additional step was needed to enable the pose-invariant capability. This step requires a feature quality evaluation to determine the best combination between the views. In much previous work, the features
quality is only determined after the classification step. A novel method was proposed that can evaluate the feature quality before the recognition step. This method is used to assign weights to each camera pose which indicates features clarity from this pose. The two 2D images from 2 views are then weighted using the calculated weights and linearly combined producing single 3D features. This technique was implemented and tested on 3D Arabic sign language. This model can be used to recognize any other sign languages also it can be used in 3D object recognition problems in general.

Different sign language recognition systems in general and Arabic specifically were developed. Many of these systems lack extendibility and suffer from environmental factors sensitivity, such as signer position and view angle. This research developed and implemented ASL translation system that can be extended offline and gives the freedom of position and viewing angles. Posture classification produced a recognition accuracy of 95% of 158 postures which build ASL one and two hands postures dictionary.

Till now, all the proposed models can handle single posture or gesture. This next step proposes and implements ASLR system of it focuses on recognizing the continuous gestures using graph-matching technique. Gestures have been divided into elementary elements, static postures. Gesture recognition is performed by "graph matching" algorithm. The gestures, which represent alphabets or words, are stored in database as models graphs. Each graph consists of a group of vertices and edges. The algorithm used for graph and sub-graph isomorphism detection is based on the decision tree paradigm. In the computational complexity analysis, it is shown that the new algorithm has a worst-case run time complexity that is only quadratic in the size of the graphs that are to be compared. The recognition rate does not lay down 70% for 100 gestures composing 30 continuous sentences.

Due to free recognition environment, errors are unavoidable. We proposed a post-processing model based on NLP semantics. A hybrid semantic-oriented approach is proposed in Arabic sign recognition error detection and correction which shows better performance in domain-specific applications. The proposed approach has the following advantages: First, it is fast and easy to develop, and leads to computationally simple implementation. The background knowledge ontology dictionaries are independent of the sign recognition lexicon, and open-vocabulary, and are constructed only once, except for the domain dictionary which depends on
a specific application domain, but is very small compared to the ontology dictionary. Second, because the LSP scheme transforms pure lexical entries into abstract semantic categories, the size of the error pattern database can be reduced remarkably. Third, with all these facts, the LSP correction has a high possibility of generating semantically correct detection due to the massive use of semantic contexts. A proposed statistical approach is shown to correct the errors that were detected in the semantic oriented detection module. The experimental results showed that using the updated sequence every time gives more correction percentage and lower average number of iterations. The hybrid model has increased the recognition accuracy of already existing sign recognition system from 88% to more than 95%. In future, a modification is proposed to permit unequal number of words in input and corrected sequence.

**Future Work**

We can think of many future works:

- Develop the opposite direction translation, from Arabic text to Arabic sign language.
- Investigate in more depth in NLP semantics and make the post-processing model subject-oriented instead of semantics.
- Explore the feasibility of publishing the both translation systems as services in public cloud.
- Explore more features quality optimization techniques other than best match.
Appendix A
Static Hands Postures Samples.
Bibliography


Sign Language Technologies, Language Resources and Evaluation Conference (LREC), 2010


[42] "CopyCat and Kinect", overview of the CopyCat Kinect demo on the website of the Center for Accessible Technology in Sign (CATS), http://cats.gatech.edu/content/copycat-and-kinect.


[73] Vasileios Zografos and Bernard F. Buxton Pose-Invariant 3D Object Recognition Using Linear Combination of 2D Views and Evolutionary Optimisation International Conference on Computing: Theory and Applications (ICCTA’07) 2007


ملخص الرسالة

أن الهدف من هذا العمل البحثي هو تطوير نظام ثلاثي الأبعاد للتعرف على لغة الأشارات العربية
معتمداً على نظام التصوير بحيث أنه تم الترجمة في رد فعل وقتي. و بدون أن نبقى تكثي أنظمة
الترجمة من و الي لغة الأشارات من المتطلبات التي ينادي بها العالم للمساعدة على كسر الحواجز
بين الإنسان الطبيعي ومن يعاني من إعاقة الصمم. من أجل ذلك تم تركيز الهدف من البحث على
تطوير التقنيات الخاصة بنظام "الترجمة الآلية للاشارات الصم و البكم العربية".

ان لغة الأشارات العربية من أكثر اللغات تعقيداً وذلك لبعض الخصائص التي تميّزها. أولها أنها
لغة وصفية، بالنسبة للشخص الذي يؤدي الأشارة يقوم بوصفها. ثانياً نفس المعنى من الممكن أن يؤدي
باليد اليمنى أو اليسرى على حد سواء. ثالثاً الكثير من الأشارات تعتبّر اعتماداً أساسياً على
التعابير الخاصة بالوجه. وهذا خارج نطاق هذا العمل البحثي. أن لغة الأشارات العربية تتضمن ما
يقرب من (9000) أشارة وأيضاً (26) حركة ساكنة (5) أشارة حركية للتعبير عن الحروف
الأجدية. 160 حركة وقتيّة لا تعتمد على الوجه.

أولى الخطوات الأساسية هو الحصول على صور جيدة للإشارات الفيديو.. في
هذا العمل البحثي. أشارات الفيديو يتم استخراجها باستخدام كاميرا (24 bits/pixel) (جهة لو،
حوصل الصورة (160 X 120 pixel) في 5frames/sec). أيضاً من المطلوب أن يتم توسيع
مجردة لإضفاء عليه. أيضاً أن يكون اتجاهاً لتطوير جيد للتنقيف ما بين الأشارات
المختلفة.

بعد ذلك تم دراسة و تطوير نظام باستخدام كاميرا وأحد باستخدام مصنف هو هجين من
PCNN/MLP (القدرة على النبات في الشبكة العصبية) و تتمتع الشبكة العصبية (PCNN/MLP)
الأداة في أحوال مختلفة (الراحة، دوران و التحريك). هذا بالإضافة على تعديل تأثير الإضاءة و
حجم اليد في الصورة. في البحث، تم استكشاف الدقة والدقة في النموذج الرياضي لاستخراج
"صمت الصورة" عن طريق إضافة "معالج الإتصال" الذي يمكن عن مدى التماسك و التأثير
المباشر للنقط المضيئة في الصورة. ولكن انا البحث مشكلة تمييز بين بعض الأشارات التي لا
يمكن تمييزها من زاوية تصوير واحدة. و من البدء تطوير نظام ثلاثي الأبعاد باستخدام 2 كاميرا
من زاويتين متعددة. تم تطوير نظام متكرر للاستخدام في حركة وضعية. نظام التصور يستخدم هذا النظام
على تحضير النماذج الصناعية المستخرجة من للزوايا المتزامنة للفم الحركة. يتم التحميل
بين النماذج المستخرجة في أداة النماذج ثلاثية الأبعاد واعضاء دالة فإنماهي و أتراكما و
أثبات نجاحها من قبل فريق البحث.

وبعد التعرف على الأشارات مفردة يتم إعادة تجميع الأشارة الحركية والعمل و يتم ترجمتها
للعربية. يتم تطبيق تقنية توافق الأشكال. الخوارزم المستخدم يعتمد على شجرة القرار. و قد تم
تأهيله و تطبيقه.

و أخيرا تم تطوير نظام متكرر لتصحيح أخطاء التعرف متمهذا على التوافر اللغوي و المعاني
المستخرجة من اللغة و معالجتها. قد تم تطوير أنظمة مشابهة لأمثلة مختلفة مثل التعرف الضوئي
OCR (التعامل الضوئي على الكتابة و التعرف على الكلام المنطوق). و لكن
لا يتم تطبيق هذه الفكرة على أنظمة مشابهة لترجمة لغة الإشارات. في هذا البحث، تم تطوير نظام
اكتشاف و تحسين الخطأ و تم تطبيقه على نطاق معين و قد تم أيضاً دورة هذا النظام و
مساهمته في تحسين دقة الترجمة.
التعريف الوقتي ثلاثي الأبعاد على لغة الاشارات العربية المستمرة

هذه الرسالة مقدمة إلى قسم الحسابات العلمية، كلية الحاسبات و المعلومات، جامعة عين شمس للحصول على درجة الدكتوراه في علوم الحاسبات و المعلومات.

مقدمة من
أحمد سمير الأسح حسين
حاصل على درجة الماجستير قسم الحسابات العلمية، كلية الحاسبات و المعلومات، جامعة عين شمس

المشرفون
أ.د. محمد فهمي طلبة
استاذ قسم الحسابات العلمية، كلية الحاسبات و المعلومات، جامعة عين شمس

أ.د. مجدى أبو العلا
استاذ بآكاديمية السادات، المعادي، القاهرة