Faculty of Computers and Information
Information Technology department

Hand Gesture Recognition Mobile Application

Wireless devices play an important role in our life . Mobile phones can be used to help people to overcome their problems that face them every day, every where. Here it intended to enable deaf mutes to communicate smoothly and easily with their communities.

This project is considered a mobile application that can get benefit from mobile cameras to serve deaf–mute people to translate the sign language into voice. Mobile takes a video for signs and send it to server and the server sends voice to other mobile.

This application will give the deaf-mute people the opportunity to overcome their problem and become more flexible to communicate with people without any difficulties.
This project is targeted to serve over than 10 million people in Egypt that have this handicap and will serve any communications company to enlarge its customer number.

During the last half of the century, sign languages are now accepted as minority languages which coexist with majority languages, and they are the native languages for many deaf-mutes. Various sign language systems have been developed by manufacturers around the globe but they are neither flexible nor cost-effective for the end users. With the development of software operated hand held devices, hand gesture recognition became a new hope for this group of people to communicate among themselves or with other people. However device based measurement which measures hand gestures with equipment such as data gloves which can archive the accurate positions of hand gestures as its positions are directly measured. Secondly, vision-based technique which can cover hands signer in which signer does not need to wear data gloves device.

مقدمه

تلعب الهواتف المحمولة دورا هاما في حياتنا. حيث انها يمكن أن تستخدم لمساعدة مستخدميها للتغلب على المشاكل التي تواجههم يوميا للتواصل مع الآخرين.
هذاالمشروع يعتبر تطبيق على الهواتف المحمولة التي يمكن أن تستخدم كاميرا الهاتف لخدمة الصم والبكم لترجمة لغة الإشارة إلى صوت. تأخذ كاميرا الهاتف الاشارات وترسلها لتتم معالجتها وارسال الصوت المقابل للأشارة إلى هاتف محمول آخر.

هذا التطبيق سيعطي الصم والبكم فرصة للتغلب على مشكلاتهم وأصبحت أكثر مرونة على التواصل مع الآخرين من دون أي صعوبات.

ويستهدف هذا المشروع خدمة أكثر من 10 مليون شخص في مصر لكى يتغلب على هذه العوائق ، وسوف يخدم هذا المشروع أي شركة اتصالات عن طريق زيادةعدد العملاء.

Acknowledgment

Firstly we would like to thank ***Allah*** for giving us talent and ability to complete our project and reaching quite satisfaction of it.

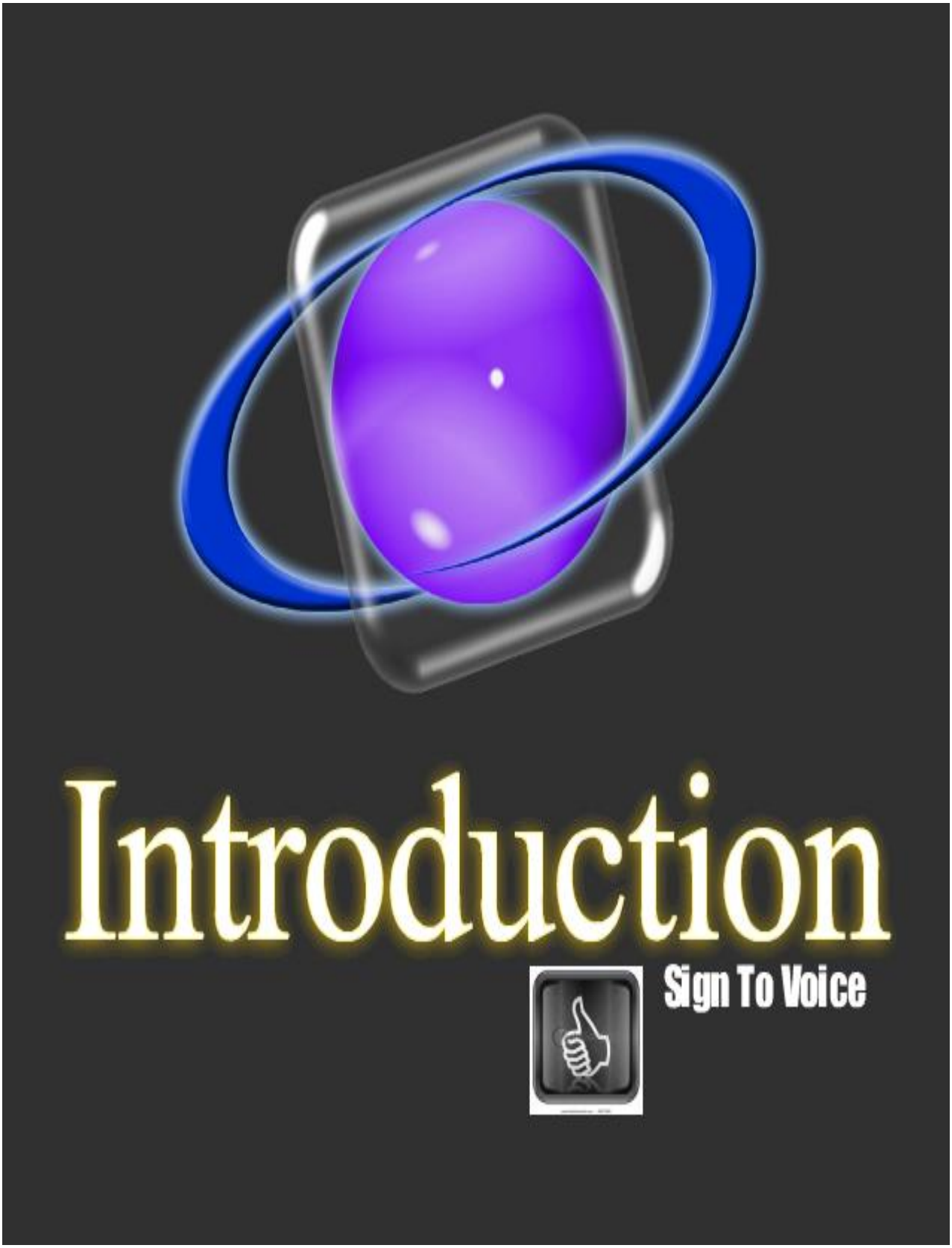
Thanks to our supervisors Dr. Osama Abd El-raoof for his efforts with us & for having the confidence in our abilities to allow us to pursue the topics which excite us. Dr. Osama we want to thank him for the opportunities he has provided us to work with him & for his apt comments and his time.

We would like to thank particularly the following people for their guidance and many valued discussions related to this work: Eng.Tamer Fathy, Eng.Walaa Ali, Eng.Mahmoud Khatab, and . And to our friends Nader Mohmoud, Sherif and Mohamed Elmenshawy and helwan group more than thanks, for their effort this year and their work which made difference in the project and their time that they spent all of it working with us. We won't be able to thank them enough for their help.

Lastly we would like to thank our Families, for being supportive and for the encouragement, understanding, and helping they have provided us all over the year to achieve our dreams.

Table of Contents

List of Figures


Introduction
Sign To Voice

Ch 1 Introduction

1.1. Motivation

Deaf-mute people have been physically and mentally challenged with various difficulties that impede their advancement on all economical, political, and social levels. Statistics around the world show a relatively high rate of deaf-mute people. Rates extend to an average of approximately 6 million deaf-mute people in Egypt, Taking individual age groups into consideration; it is observed that the speaking-impairment rate increases as the age increase. For it is observed that the elderly people (65 years and above) show a 25% rate and people whose ages range between 55 and 64 have a rate of 15% [1].

It would be unjust for deaf-mute people to be excluded from society just because they can not treat with other people. They are mentally capable individuals who deserve and are even demanded to play an effective role in society. Society cannot dismiss whatever potential they have, simply because it needs it; for it is known that society advances only with the collective effort of all of its members. To this end, technology is utilized to aid humans in their collective effort to overcome such impediments.

A technology in our times is that has helped us to engage in this project. New mobile technology and wireless technology play an important rule in this project.

1.2. Introduction about gesture recognition

1.2.1. What is Gesture Recognition?

Interface with computers using gestures of the human body, typically hand movements. In gesture recognition technology, a camera reads the movements of the human body and communicates the data to a computer that uses the gestures as input to control devices or applications. For example, a person moves his hands in front of a camera to open a document in a computer.

One way gesture recognition is being used is to help the physically impaired to interact with computers, such as interpreting (recognize) sign language. The technology also has the potential to change the way users interact with computers by eliminating input devices such as mice and keyboards and allowing the unencumbered body to give signals to the computer through gestures such as finger pointing.

Gesture recognition does not require the user to wear any special equipment or attach any devices to the body. The gestures of the body are read by a camera instead of sensors attached to a device such as a data glove.

In addition to hand and body movement, gesture recognition technology also can be used to read facial and speech expressions (i.e., lip reading), and eye movements.

1.2.2. Body Gesture Recognition Application

1.2.2.1. Sign Language

Sign languages serve the same functions as a spoken language. They are natural languages which are used by many deaf people all over the world. The majority of those mute people have only a limited vocabulary of the spoken language of the community in which they live.

As a consequence their reading and writing ability in that language is limited as well. Therefore, sign language users may want to read and write documents in their own language. The developed sign writing system allows a user to perform this task. Since sign language is a visual spatial language a video-based input system is developed as the most natural 3D input device.

American Sign Language or ASL is an example of a sign language. These simple gesture alphabets are used to spell names or words (letter by letter), for which there is no signing either known or present in the vocabulary. ASL provides a more suitable problem domain over British Sign Language as the BSL finger spelt alphabet is a two-handed system. Although this two handed system in reality provides a method of signing which is far easier to understand, it presents added difficulty for computer vision tasks due to the problems associated with occlusion.

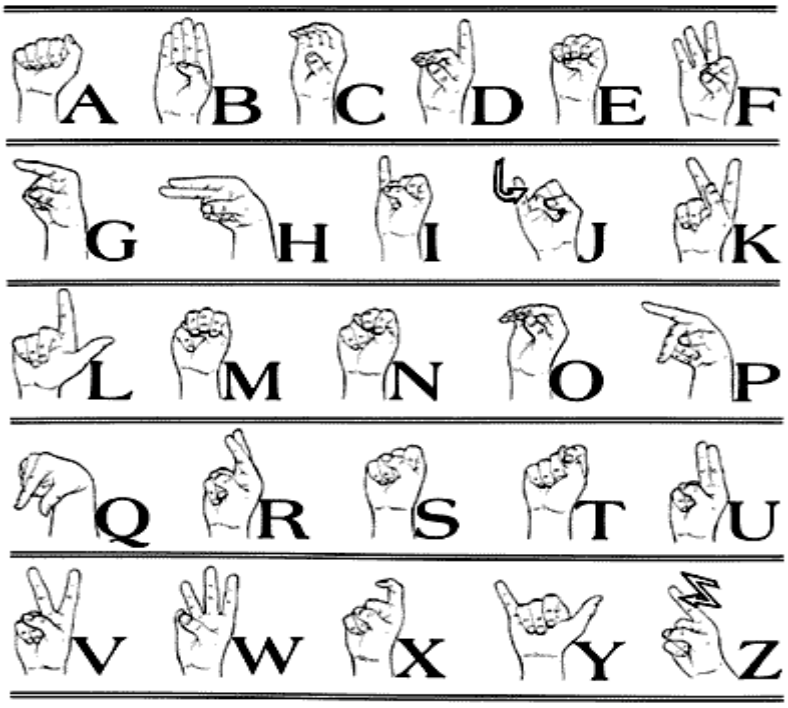


Figure (1.1) Sign Language Gestures

1.2.2.2. Virtual environments

The main goal of Virtual environment is to increase the usability of the computer and solve more complex cases by simulating it graphically.

Virtual environments provide the sensory experience of being in a computer generated, simulated space. They have potential uses in applications ranging from education and training to design and prototyping.

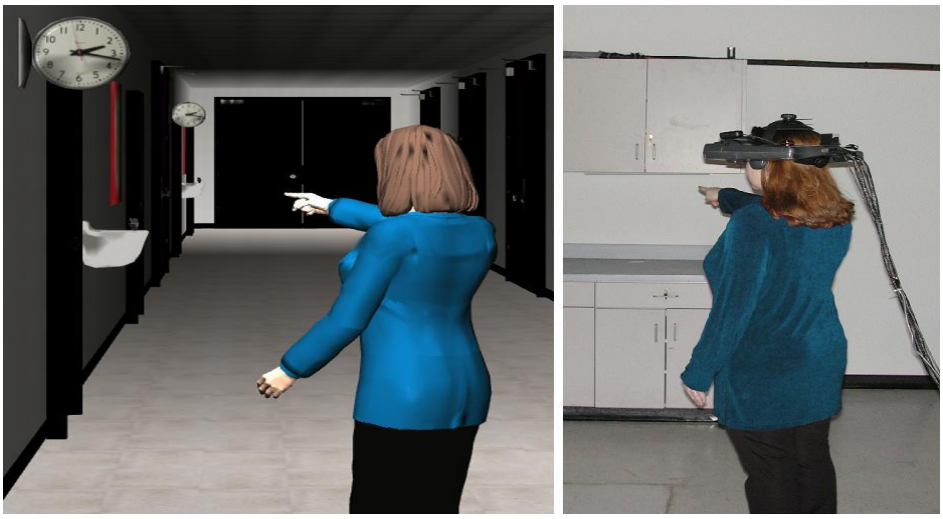


Figure (1.2) Virtual environment

Virtual environments provide a whole new way of viewing and manipulating data. Current technology moves the images out of desktop monitors and into the space immediately surrounding the user. Users can literally put their hands on the virtual objects. Unfortunately techniques for interacting with such environments have yet to mature. Gloves and sensor based trackers are unwieldy, constraining and uncomfortable to use. A natural, more intuitive method of interaction would be to allow the user to move objects with their hands and manipulate them as if they were real physical objects. In addition, a simple hand gesture recognition mechanism allows free-hand input to be interpreted similar to a mouse pointer.

1.3. Solution Approach

Computer vision is essential for the body gesture recognition system. There is an extensive body of related computer vision research which could fill many books. Here, we summarize the major works that could fit for real-time user interface operation through body gesture recognition in a fairly unconstrained environment. The computer vision processing includes three major tasks.

1.3.1. Detection

In computer vision and image processing the concept of **feature detection** refers to methods that aim at computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not. The resulting features will be subsets of the image domain, often in the form of isolated points, continuous curves or connected regions.

1.3.2. Tracking

Body tracking is difficult since it has a variety in its position, orientation and movements' freedom .Therefore tracking method needs fast approach to operate at image acquisition frame rate. In some ways detection method can be suitable and fast enough.

On the other hand, mostly solid objects can be tracked with very limited shape modeling efforts. Since tracking with a rigid (solid) appearance model is not possible for human body in general, most approach resort to shape-free color information or background differencing.

Figure (1.3) object tracking

1.3.3. Recognition

Recognizing or distinguishing different gesture configurations is a very difficult and largely unsolved problem in its generality. However, to achieve the more strict requirements of user interface quality, robustness and a low false positive rate are more important than recognition of the complete gesture configuration space from the entire view sphere. The posture recognition task becomes more tractable for a few select postures from fixed views.

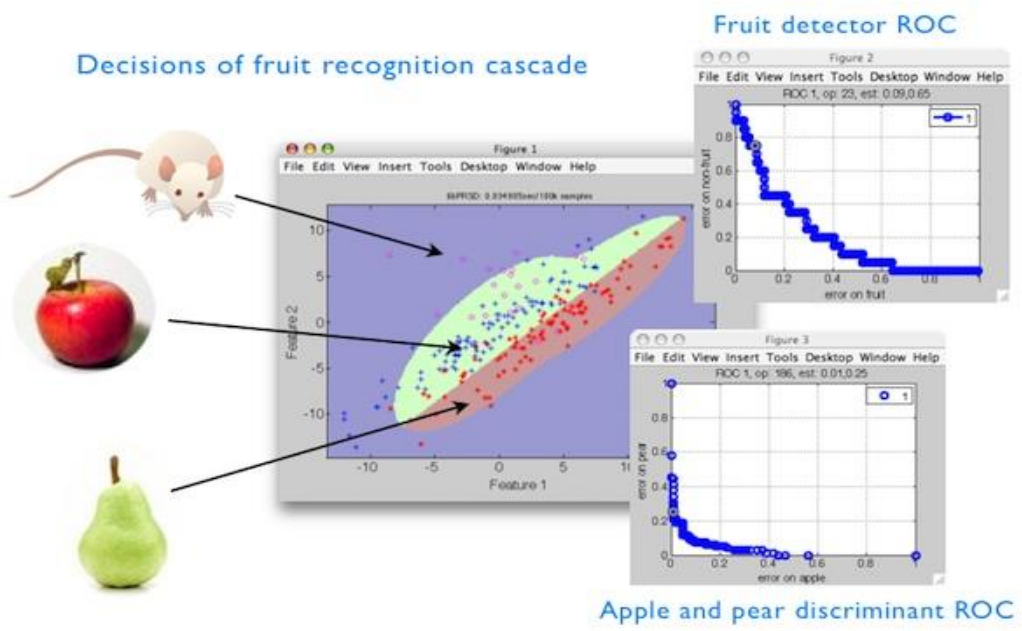


Figure (1.4) Object Recognition

1.3. Overview of the Document

The documentation is organized as follows; Chapter 2 introduces the proposed project analysis, discussing the System Development Life Cycle (SDLC).

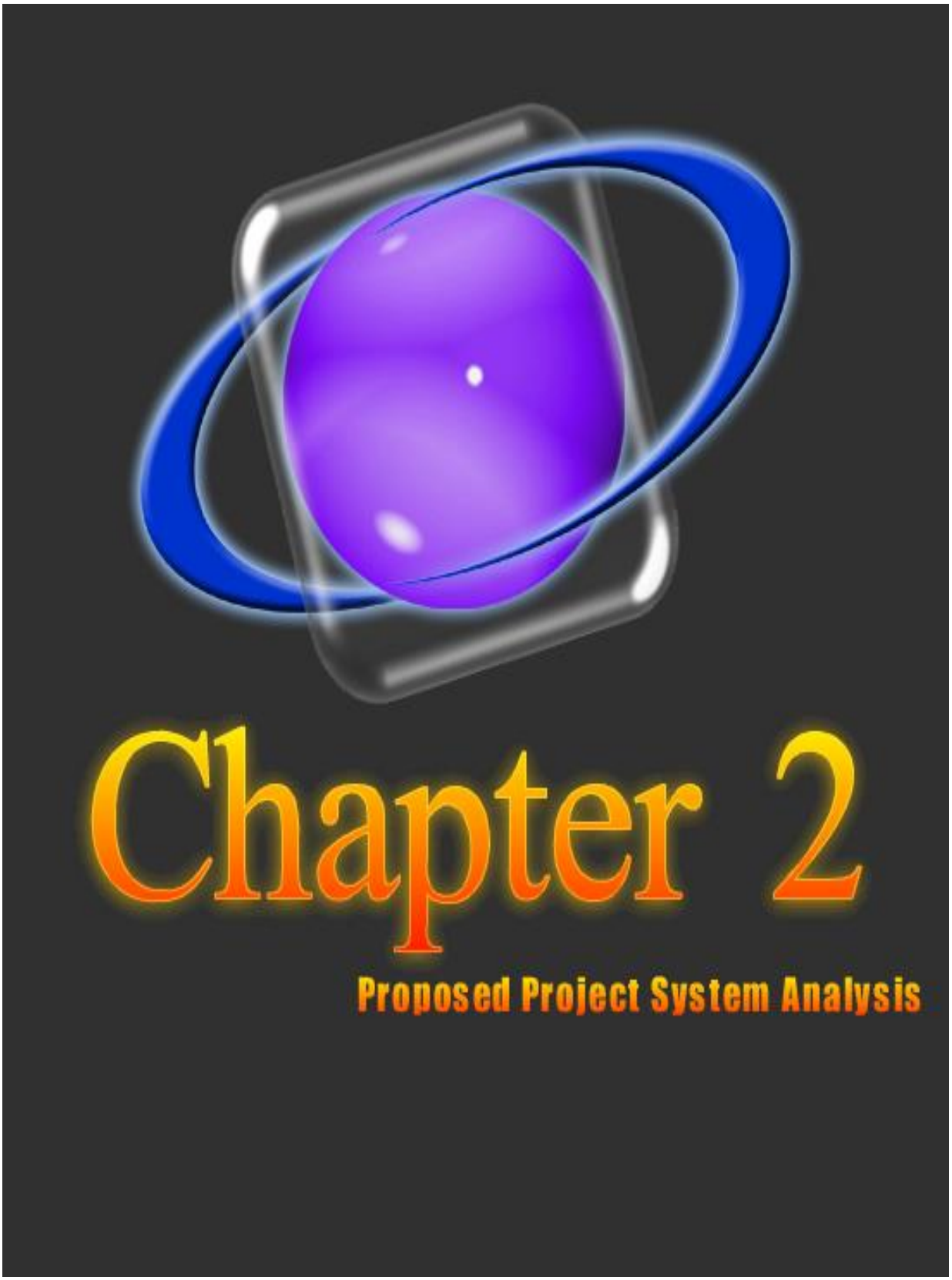
Concerning chapter 3 discusses the Mobile Technology, Operating System of Mobiles and how can we use J2ME.

To go to chapter 4 that describes Wireless Connection and how UDP Socket opened between Mobile and Server.

To go to chapter 5 that discusses how to recognize hand gesture, extract hand from background and produce text.

To go to chapter 6 that discusses how sound produced and how to send voice from server to Mobile.

To go to chapter 7 that describes the design and analysis method of the system by applying the Unified Modeling Language (UML).


Chapter 2
Proposed Project System Analysis

Ch 2
Proposed Project Analysis

2.1- Introduction:

In the world of the sign language and its difficulty, being successful requires careful planning and taking the right steps. The SDLC "System Development Life Cycle" (Figure 2.1) will be very helpful for taking the right step in building a successful Hand Gesture Recognition System or any other kind of Software.

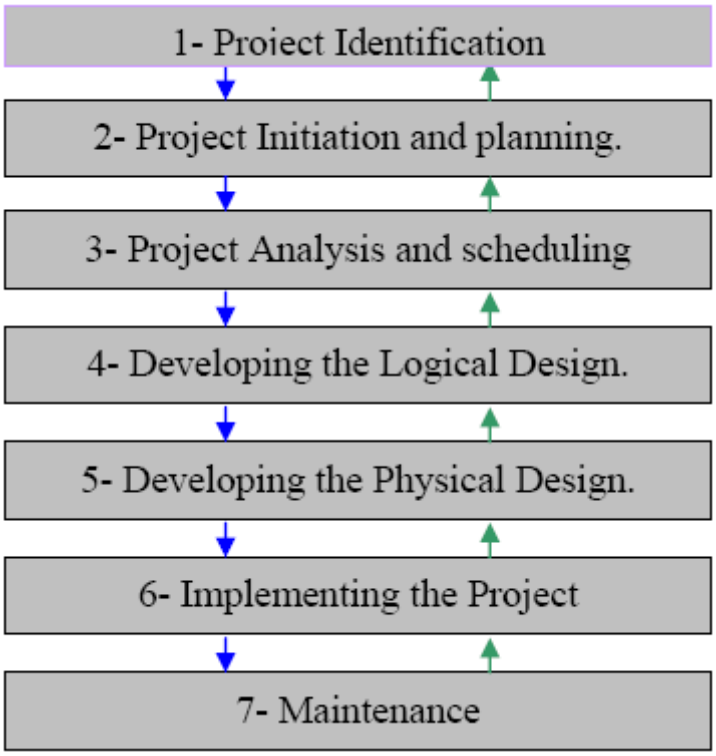


Figure (2.1) SDLC

2.2- Project Phases:

SDLC has defined a several standard phases for developing any system. These phases if followed carefully, the system designing and analyzing and planning and implementation will give a good results.

2.2.1- Project Identification:

Hand Gesture Recognition system is consists of two parts. The first Part is running on Server, this part is responsible for Image Processing and Hand Gesture Recognition. The second part is operating on Mobile Phones that are connected to the Server via wireless network and they are acting as client.

Mobile is being hold by Deaf-Mute and the other generates the sound of the Deaf-Mute gesture.

2.2.2- Project Planning:

-*Project Scope:* This Project is aim to able to communicate between Deaf-Mute and human community via mobile communication.

-*Feasibility Study:*

- **Economic:** This project needs high performance devices (server, mobiles) and wireless network.
- **Technical:** The Tools and Development software used for building the project are one of the most important factors that affect the project performance and building time.
- **Operational:** Recognition of Sign Language is very important for completing this project.

2.3- Project Analysis & Scheduling:

2.3.1- Project Scheduling:

Number	Task	Start	End	Duration	Q2 - 2009		
					April	May	June
1	Creating Image&Sound DB	4/1/2009	4/2/2009	1			
2	Open wireless Connection Between Server and Cleints.	4/1/2009	4/30/2009	21			
3	Image Theshold.	4/1/2009	4/4/2009	3			
4	Image Segmentation.	4/3/2009	4/21/2009	12			
5	Matching With DB.	4/21/2009	4/29/2009	6			
6	Execute the image Processoing part with the wireless connection.	5/1/2009	6/2/2009	22			
7	Generate Sound at server and send it to other client	6/2/2009	6/6/2009	4			

Figure (2.2) project Scheduling

2.3.2- Physical Design of the System:

2.3.2.1- Context Diagram (level 0 diagram):

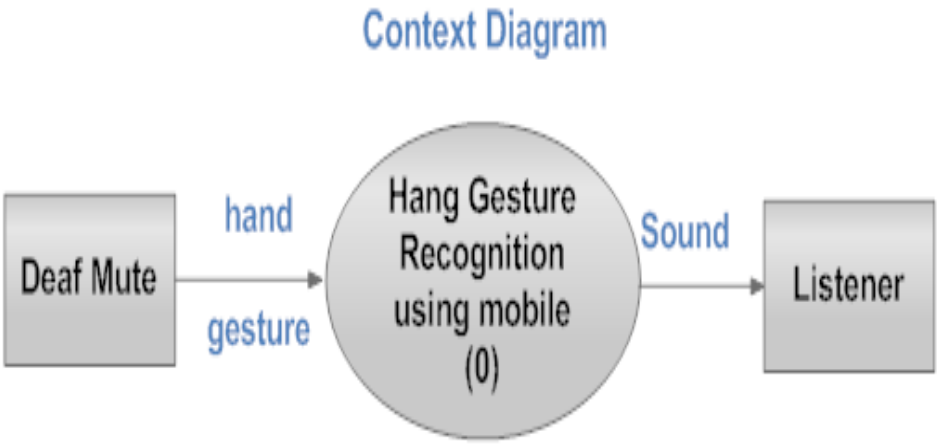


Figure (2.3) Context Diagram of Project

This is an overall view of the all system:

1- **Deaf-Mute*:*** This external entity is a Deaf-Mute has a mobile with camera.

2-**Listener***:* This one is with the other person who talking with the Deaf-Mute.

3-Hand Gesture Recognition using Mobile*:* This Process is what happens in the system (acquiring hand gesture, and generating sound).

2.3.2.2- DFD Level 1:

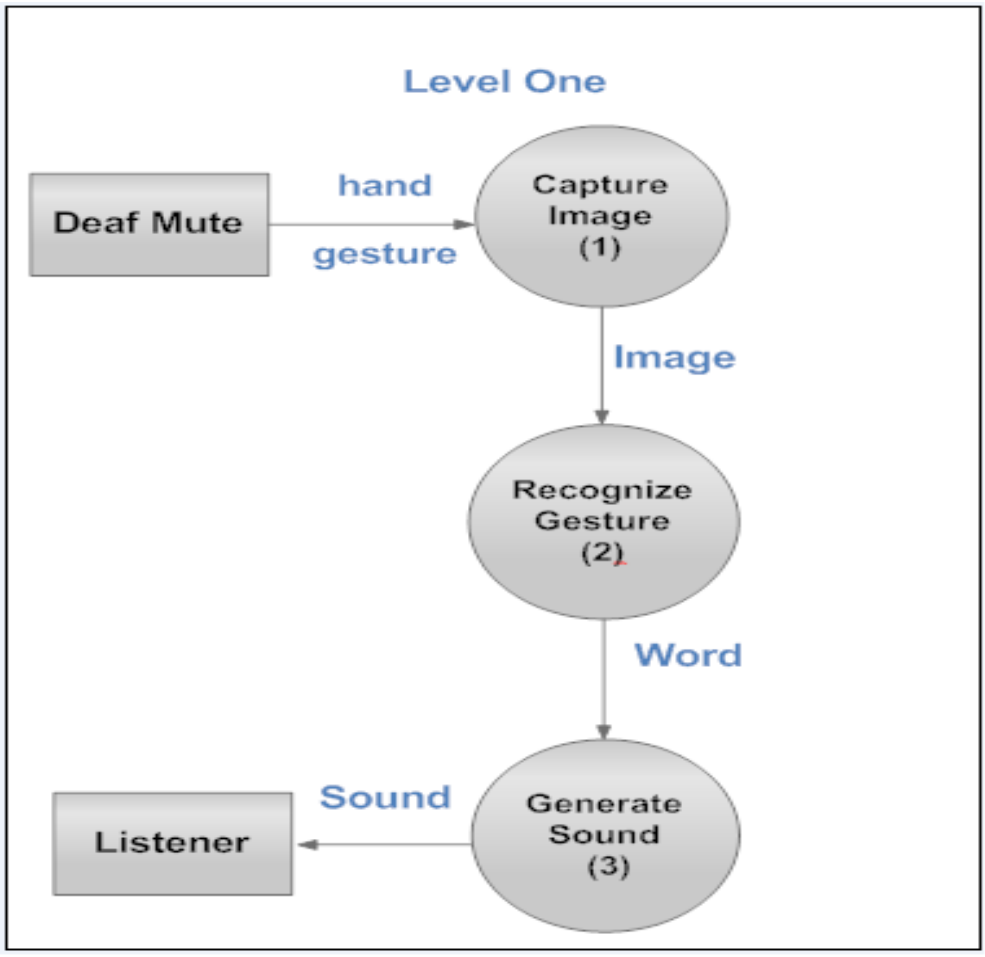


Figure (2.4) Level 1 diagram.

1-Capture Image: The process of acquiring the Hand Gesture via mobile camera and sending it throw the network.

2-**Recognize Gesture**: This process receiving image from Capture Image process and do some operations to generate text word.

3-Generate Sound: getting the word from the Recognize Gesture process and generate the sound of this word.

2.3.2.3- DFD Level 2 (Image Processing Phase):

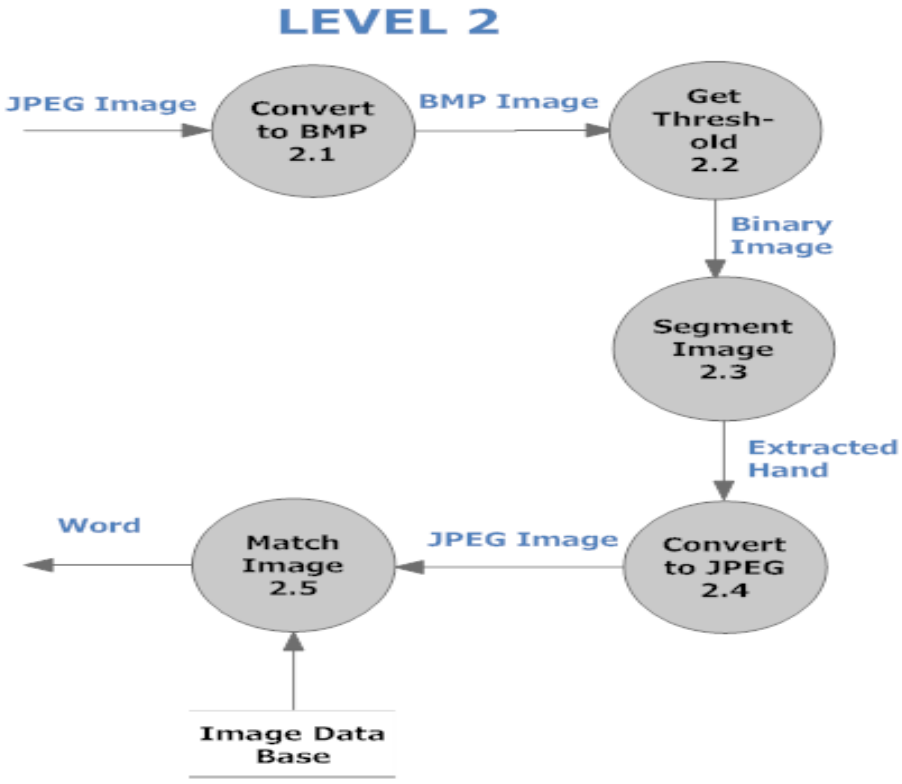


Figure (2.5) Level 2 diagram.

This diagram is what happens in the Recognize Gesture Process in Figure (3.3).

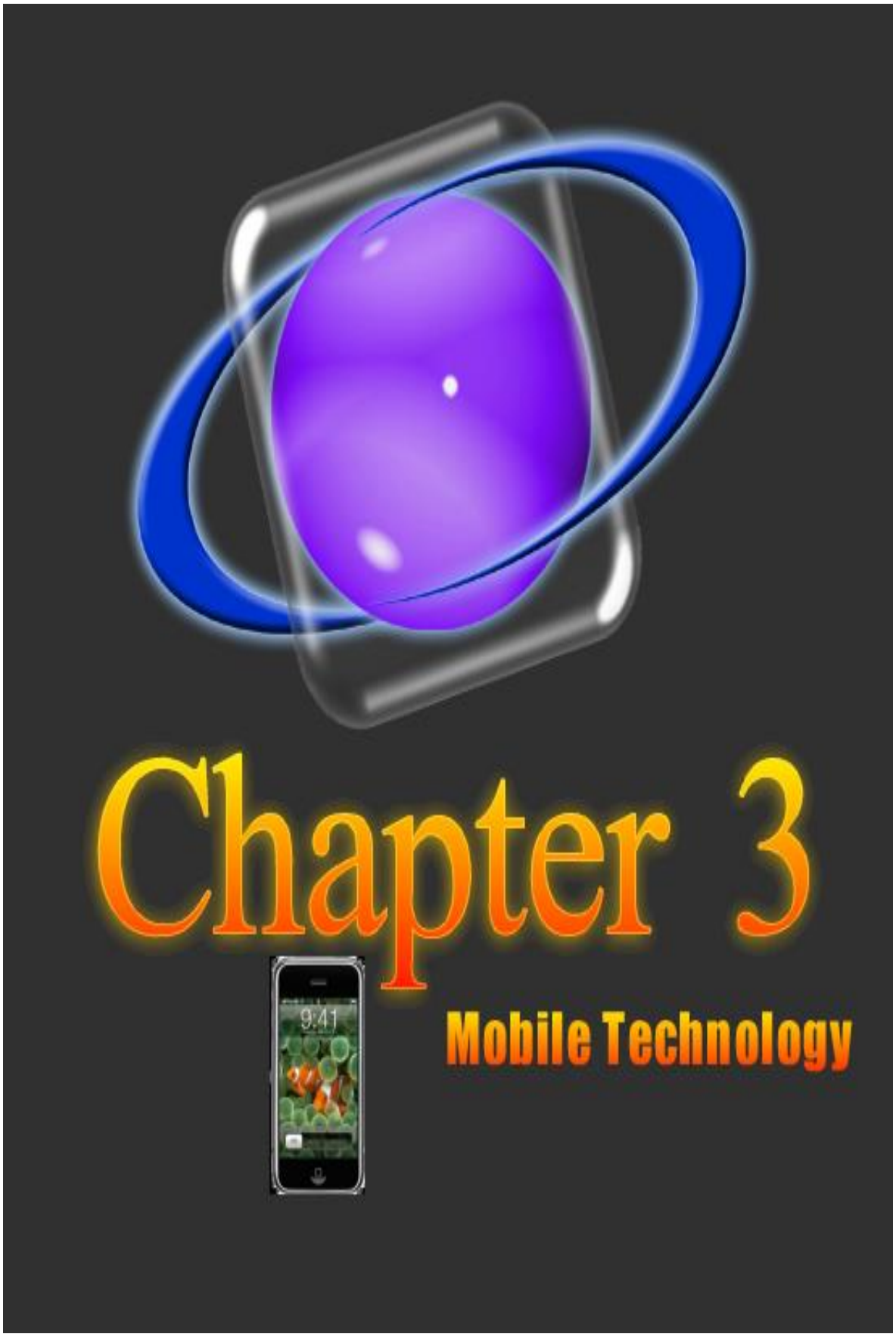
1-**Convert to BMP***:* The image received by Recognize Gesture is in the format JPEG so it should be converted to BMP for the ease of use.

2-**Get Threshold:** Convert the BMP image to Binary Image (Blake White Image).

3-**Segment Image:** *Extracting the hand from the background of the image.*

4- **Convert to JPEG***:* Make the output image JPEG.

5-**Match Image:** Match the segmented image with the opposite word.


Chapter 3
Mobile Technology
9:41

Ch 3
Mobile Technology

3.1- Mobile Operating System:

***Definition*:** The operating system that controls a mobile device similar in principle to an operating system such as Linux or Windows that controls a desktop computer. However, they are currently somewhat simpler, and deal more with the wireless versions of broadband and local connectivity, mobile multimedia formats, and different input methods.

-Live Operating System:

Operating systems that can be found on mobile devices include Palm WebOS ,Symbian OS, RIM's BlackBerry, Windows Mobile, Familiar Linux, Palm OS, The Ångström Distribution, Maemo and the iPhone OS. The Open Handset Alliance's Android is a recent smart phone addition touted by Google and T-Mobile (which launched the G1 phone on October 22, 2008). The OHA hopes Android will gain 4% market share by year's end[].

The most common operating systems (OS) used in Smartphones are in Q3 2008:

3.1.1-Symbian OS:

Figure (3.1) Symbian OS GUI

Symbian has the largest share in most markets worldwide, but lags behind other companies in the relatively small but highly visible North American market. This

matches the success of its owner and largest customer, Nokia, in all markets except Japan. Nokia itself enjoys 52.9% of the smartphone market. In Japan Symbian is strong due to a relationship with NTT DoCoMo, with only one of the 44 Symbian handsets released in Japan coming from Nokia. It is used by many major handset manufacturers, including BenQ, LG, Motorola, Samsung, and Sony Ericsson. Various implementations of user interfaces on top of Symbian (most notable being UIQ and Nokia's own S60) are incompatible, which along with the requirement that applications running on mobile phones be signed is hindering the potential for a truly widely accepted mobile application platform. It has received some adverse press attention due to virus threats (namely Trojan horses).

3.1.2-iPhone OS:

Figure (3.2) iPhone OS GUI

The iPhone and iPod Touch use an operating system called iPhone OS, which is derived from Mac OS X. Third party applications were not officially supported until the release of iPhone OS 2.0 on July 11th 2008. Before this,"jailbreaking" allowed third party applications to be installed, and this method is still available.

3.1.3-BlackBerry OS:

Figure (3.3) BlackBerry OS GUI.

This OS is focused on easy operation and was originally designed for business. Recently it has seen a surge in third-party applications and has been improved to offer full multimedia support.

3.1.4-Windows OS for Mobile:

Figure (3.4) WinOS For Mobile GUI.

The Windows CE operating system and Windows Mobile middleware are widely spread in Asia. Improved variants of this operating system, Windows Mobile 6 Professional (for touch screen devices) and Windows Mobile 6 Standard, were unveiled in February 2007. Current release is Windows Mobile version 6.5 (2009Q2).

3.1.5-Linux:

Figure (3.5) Linux For Mobile GUI.

Linux is strongest in China where it is used by Motorola, and in Japan, used by DoCoMo. Rather than being a platform in its own right, Linux is used as a basis for a number of different platforms developed by several vendors, including Motorola and TrollTech, which are mostly incompatible. PalmSource (now Access) is moving

towards an interface running on Linux. Another platform based on Linux is being developed by Motorola, NEC, NTT DoCoMo, Panasonic, Samsung, and Vodafone.

3.1.6-Palm OS:

Figure (3.6) Palm OS GUI.

Palm Source traditionally used its own platform developed by Palm Inc. Access Linux Platform (ALP) is an improvement that was planned to be launched in the first half of 2007. It will use technical specifications from the Linux Phone Standards Forum. The Access Linux Platform will include an emulation layer to support applications developed for Palm-based devices.

3.1.7-Binary Runtime Environment for Wireless:

BREW was developed in the USA by Qualcomm, Inc and is popular in North America and Japan (au). BREW is a mobile application development platform and end-to-end content delivery ecosystem.

3.2- Mobile Programming:

Sun provides programming language for mobile devices (J2ME).

3.2.1-What is J2ME:

Short for ***J**ava **2** Platform **M**icro **E**dition*. J2ME is Sun Microsystems' answer to a consumer wireless device platform. J2ME allows developers to use Java and the J2ME wireless toolkit to create applications and programs for wireless and mobile devices.

3.2.2-Historical Evolution:

3.2.2.1- The Green Project:

Sun came to the conclusion that the next evolutionary step would be the merger Personal Digital Assistants(PDAs) .The Sun engineers developed a new wireless handheld PDA called Star7 which need small OS that would fit in 1MB RAM.A new programming language called **Oak** has been developed for this purpose. Thus, Oak had to be very small, efficient, and easily portable to other hardware devices. Star7 was finished and officially presented on September 3, 1992.

3.2.2.2- The Spotless System:

Implementers of Java focused on increasing the speed of the JVM, leading to memory-consuming technologies. No effort was made to keep those systems small because for desktop systems, the size was not relevant. Newly created virtual machine of the Spotless System was especially designed to fit the constraints of embedded systems. The result of the Spotless System project was a small JVM that occupies less than 300 kilobytes of static memory on a PC system.

3.2.2.3- The JavaOne99 KVM Preview Version

At the JavaOne99 conference, some results of the Spotless project got their official place in the Java family. The Java Technology was split into three categories:

-Java 2 Enterprise Edition (J2EE),

-Java 2 Standard Edition (J2SE)

-The new Java 2 Micro Edition (J2ME).

Because of its **low memory footprint** of only a few kilobytes, the new virtual machine was named **Kilobyte Virtual Machine (KVM).** The new KVM was directly derived from the Spotless System project. However, there are some changes in the supported package **names**, some classes are **canceled**, and other classes are added.

3.2.3-How J2ME is Organized:

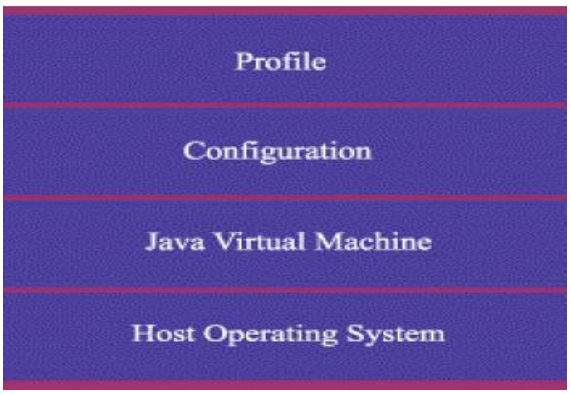


Figure (3.7) J2ME Organization

3.2.3.1-Configurations:

Defines Java Platform for different device classes:

Connected Device Configuration (CDC)

-512 kilobytes (minimum) memory for running Java

-256 kilobytes (minimum) for runtime memory allocation

-Network connectivity, possibly persistent and high bandwidth

Connected, Limited Device Configuration (CLDC)

-128 kilobytes memory for running Java

-32 kilobytes memory for runtime memory allocation

-Restricted user interface

-Low power, typically battery powered

-Network connectivity, typically wireless, with low bandwidth and intermittent access.

3.2.3.2- Profile:

A Profile is an extension to a Configuration. It provides the libraries for a developer to write applications for a particular type of device.

1-Mobile Information Device Profile (MIDP):

This profile adds networking, user interface components, and local storage to CLDC. This profile is primarily aimed at the limited display and storage facilities of mobile phones.

2-PDA Profile (PDAP):

Similar to MIDP, but it is aimed at PDAs that have better screens and more memory than cell phones.

3-Foundation Profile:

It is intended to be used as the basis for most of the other CDC profiles.

4-Personal Basis and Personal Profiles:

The Personal Basis Profile adds basic user interface functionality to the Foundation Profile.

4-RMI Profile:

The RMI Profile adds the J2SE Remote Method Invocation libraries to the Foundation Profile. Only the client side of this API is supported.

5-Game Profile:

The Game Profile, which is still in the process of being defined, will provide a platform for writing games software on CDC devices.

3.2.3.3-Java Editions:

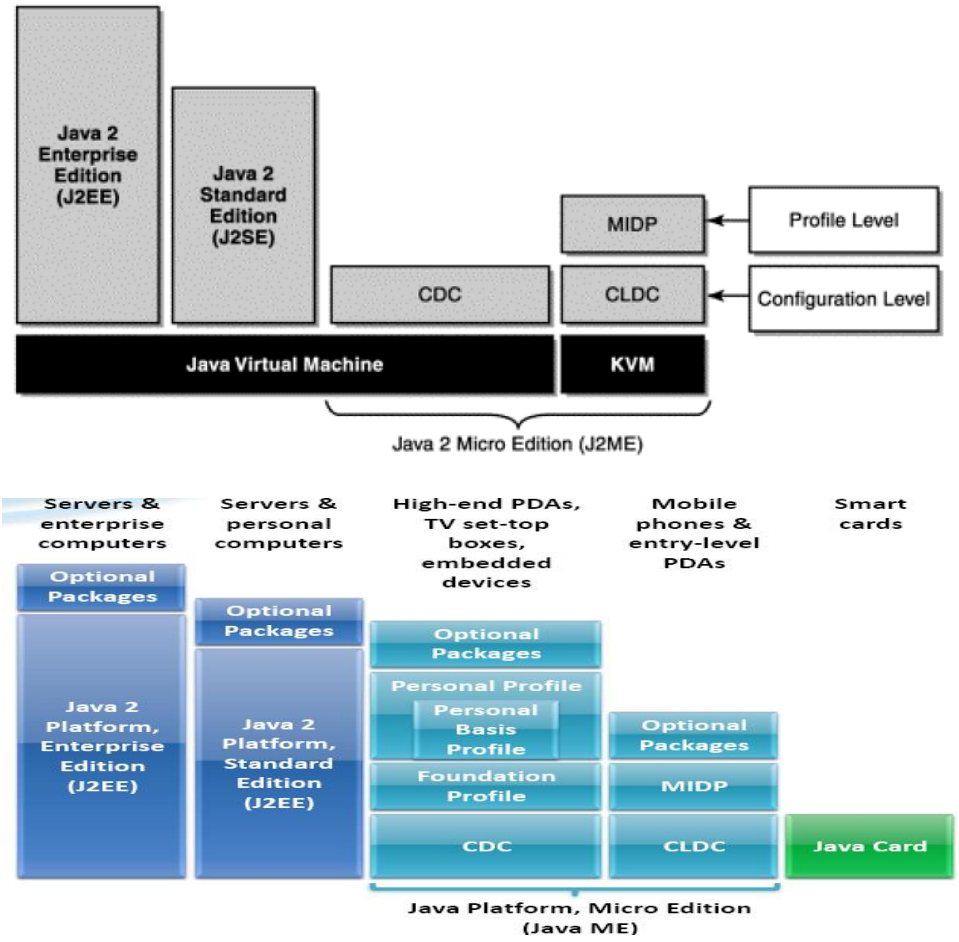


Figure (3.8) J2ME Edition

3.2.3.4- Sample Architecture of a Phone:

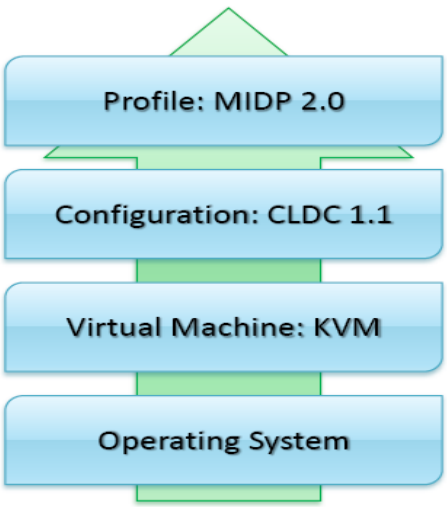
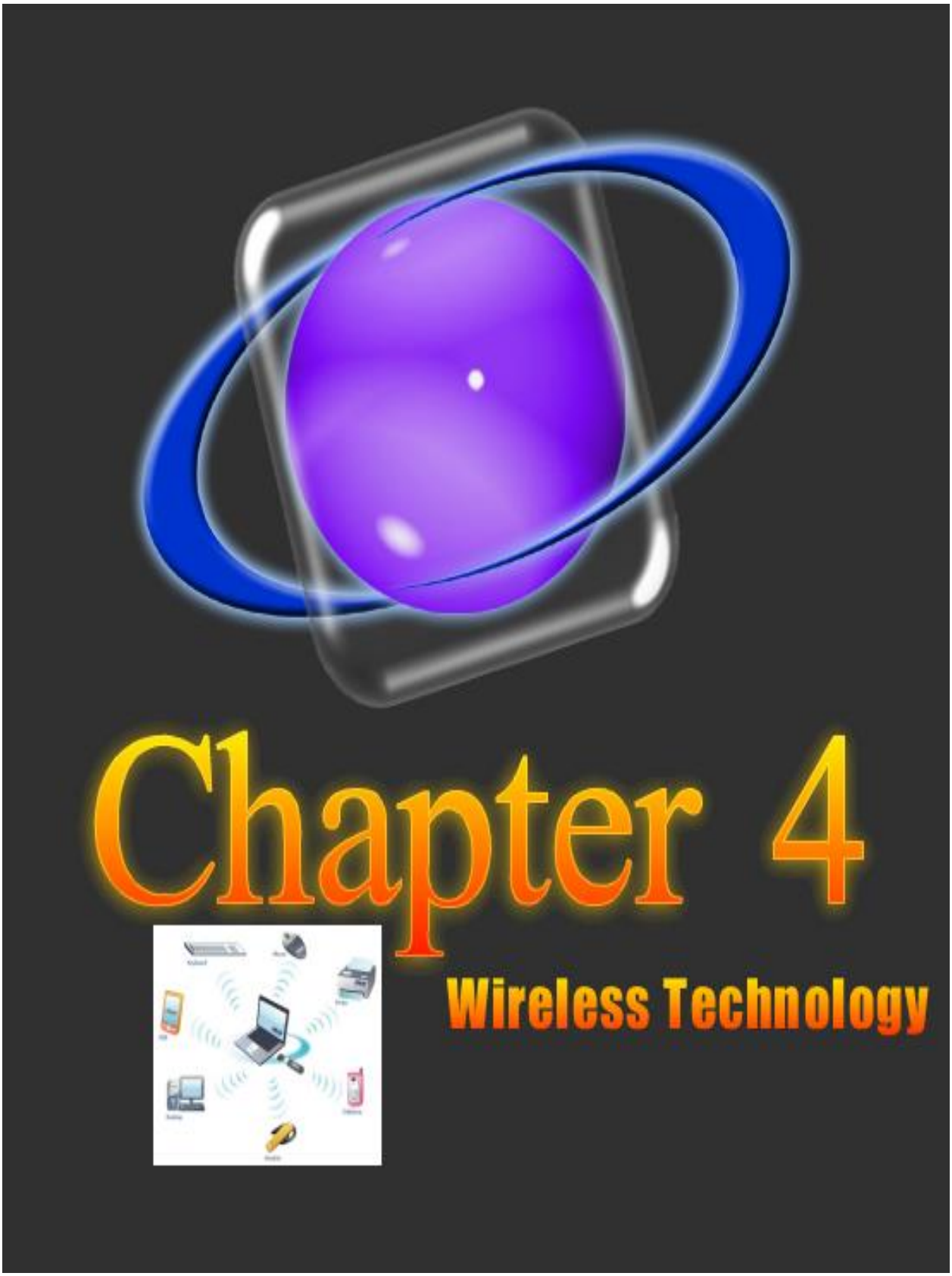


Figure (3.9) Architecture of Mobile Phone.


Chapter 4
Wireless Technology

Ch 4
Mobile Technology

4.1- Wireless Technologies:

A wireless network, or wireless networking, is defined as technology that allows two or more computers, to communicate, (enabling file sharing, printer sharing, internet connection, etc), using standard protocol ?but without the use of network cabling as shown in figure (4.1).

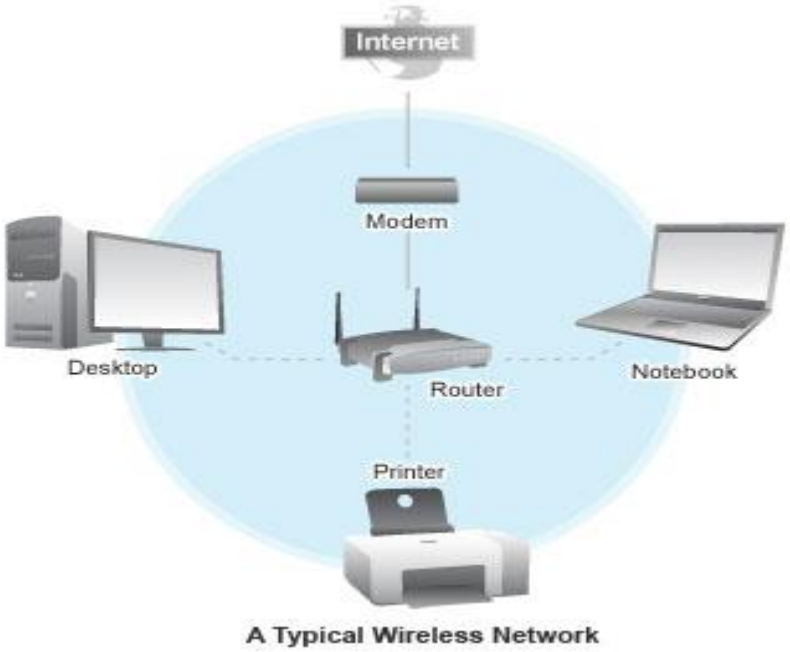


Figure (4.1) Wireless connection

4.1.1-Ad Hoc technology:

A **mobile ad hoc network** (**MANET**), sometimes called a mobile mesh network, is a self-configuring network of mobile devices connected by wireless links. Each device in a MANET is free to move independently in any direction[2] , and will therefore change its links to other devices frequently. Each must forward traffic unrelated to its own use, and therefore be a router. The primary challenge in building a MANET is equipping each device to continuously maintain the information required to properly route traffic. The purpose of the MANET working group is to standardize IP routing protocol functionality suitable for wireless routing application within both static and dynamic topologies with increased dynamics due to node motion and other factors illustrated in figure (4.2).

Figure (4.2) Ad Hoc wireless technology

4.1.2-Infra Structure technology:

Infrastructure mode wireless networking bridges (joins) a wireless network to a wired Ethernet network. Infrastructure mode wireless also supports central connection points for WLAN clients[3]. A wireless access point (AP) is required for infrastructure mode wireless networking. To join the WLAN, the AP and all wireless clients must be configured to use the same SSID. The AP is then cabled to the wired network to allow wireless clients access to, for example, Internet connections or printers. Additional APs can be added to the WLAN to increase the reach of the infrastructure and support any number of wireless clients as shown in figure (4.3).

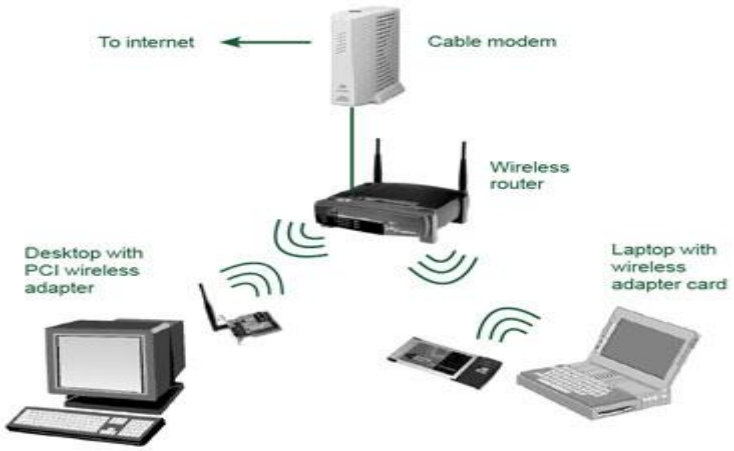


Figure (4.3) Infrastructure wireless network.

4.1.2-Bluetooth technology:

Bluetooth is a Wireless technology that lets your computer, monitor, mouse, keyboard, PDA-in fact anything with a Bluetooth chip communicates by radio instead of cables. The Bluetooth specification aims to allow Bluetooth devices from different Manufacturers to work with one another, so it is not sufficient to specify just a radio system. Because of this, the Bluetooth specification does not only outline a radio system but a complete protocol stack to ensure that Bluetooth devices can discover each other, explore each other's services, and make use of these services.

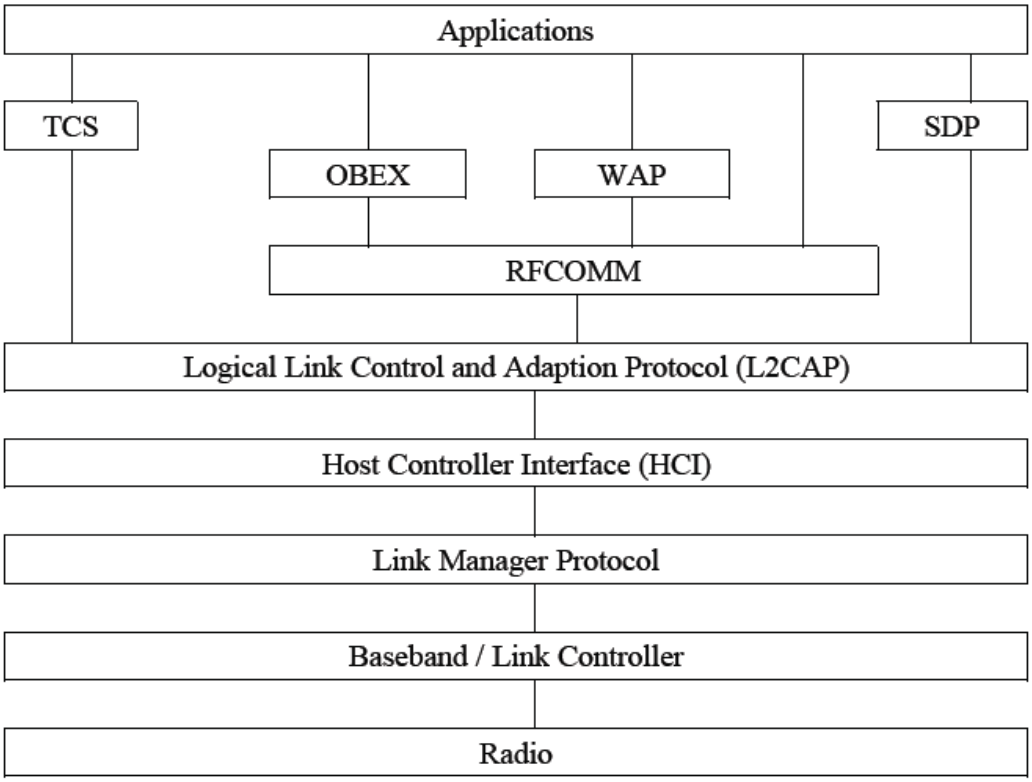


Figure (4.4) Bluetooth protocol stack

The Bluetooth stack is made up of many layers, as shown in Figure 4.4. The HCI is usually the layer separating hardware from software and is implemented partially in Software and hardware/firmware. The layers below the HCI are usually implemented in hardware and the layers above the HCI are usually implemented in software. Note that resource constrained devices such as Bluetooth headsets may have all functionality implemented in hardware/firmware.

Figure (4.5) Bluetooth connection

4.1.3- GSM Wireless technology:

One of the most important conclusions from the early tests of the new GSM technology was that the new standard should employ Time Division Multiple Access (TDMA) technology. This ensured the support of major corporate players like Nokia, Ericsson and Siemens, and the flexibility of having access to a broad range of suppliers and the potential to get product faster into the marketplace. After a series of tests, the GSM digital standard was proven to work in 1988.

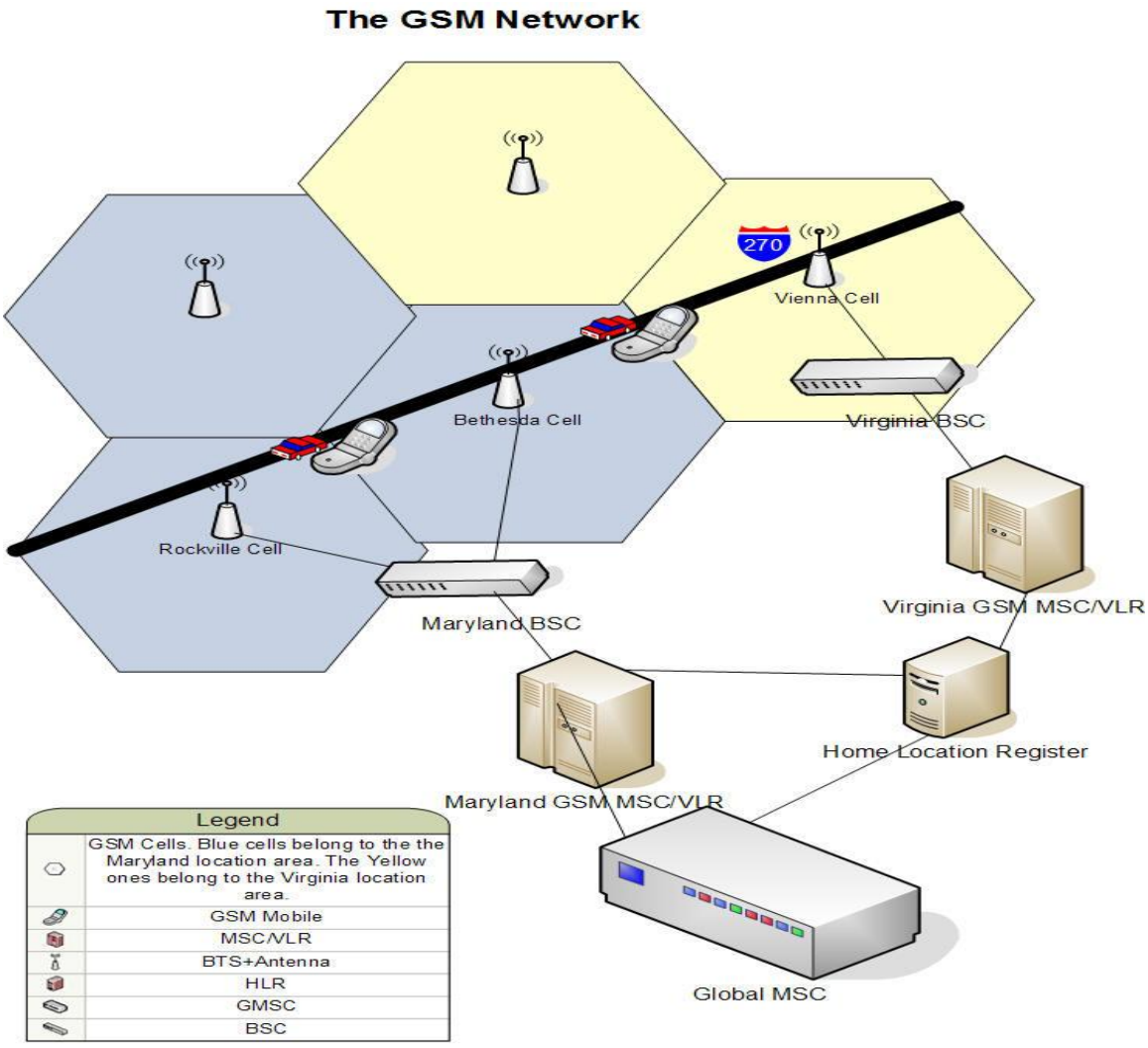


Figure (4.6) GSM Network

4.1.4- WIMAX:

WiMAX is a wireless digital communications system, also known as IEEE 802.16 that is intended for wireless "metropolitan area networks". WiMAX can provide broadband wireless access (BWA) up to 30 miles (50 km) for fixed stations, and 3 - 10 miles (5 - 15 km) for mobile stations. In contrast, the WiFi/802.11 wireless local area network standard is limited in most cases to only 100 - 300 feet (30 - 100m). WiMAX can be used for wireless networking in much the same way as the more common WiFi protocol. WiMAX is a second-generation protocol that allows for more efficient bandwidth use, interference avoidance, and is intended to allow higher data rates over longer distances as shown in figure (4.7).

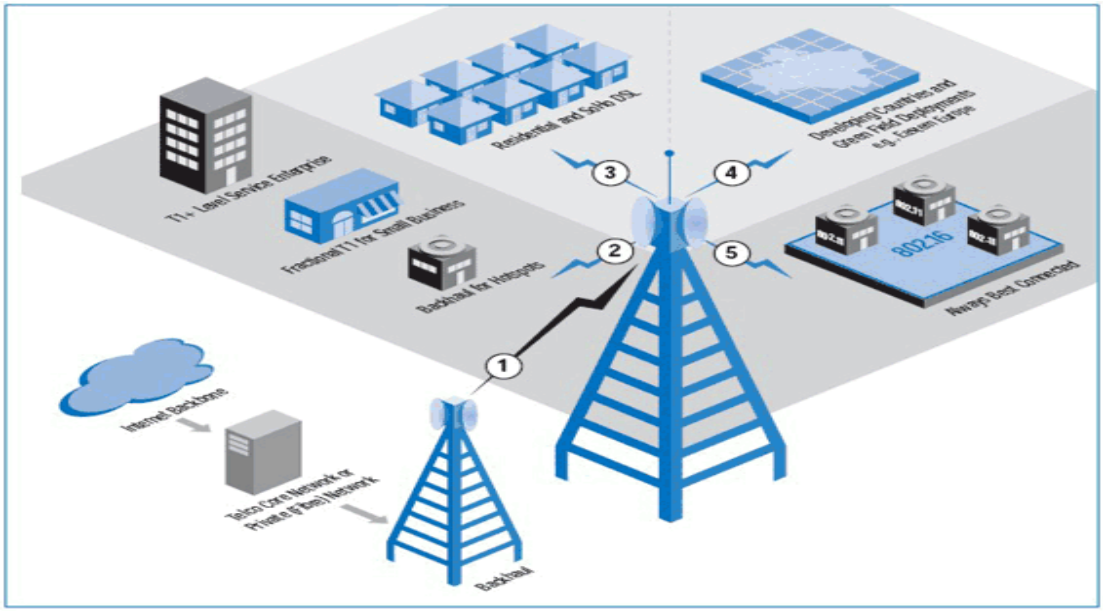


Figure (4.7) WIMAX Network

4.1.5-WI FI:

The name of a popular wireless networking technology that uses radio waves to provide wireless high-speed Internet and network connections. The Wi-Fi Alliance, the organization that owns the Wi-Fi (registered trademark) term specifically defines Wi-Fi as any "wireless local area network (WLAN) products that are based on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards.

Figure (4.8) WI FI Network

4.2 - Transmission Protocols

4.2.2- HTTP protocol:

The wireless devices such as cell phones and two-way pagers keep their owners connected to the outside world at anytime from anywhere. They offer great connectivity that other types of devices couldn't offer. Application development for these wireless devices is going to be in great demand for the next couple years. Network programming plays an important role in wireless application development to take advantage of the connectivity these devices have to offer. Sun's Java 2 Micro Edition (*J2ME*) offers a great development platform for developing applications for the embedded electronics and mobile devices. In Java 2 Micro Edition (J2ME), the Connected Limited Device Configuration (CLDC) defines a generic "configuration" for a broad range of handheld devices. On top of CLDC, the Mobile Information Device Profile (MIDP) is defined specifically for wireless devices such as cell phones and two-way pagers. Wireless device manufacturers need to implement MIDP in order to support Java applications on their devices.

The networking in J2ME has to be very flexible to support a variety of devices and has to be very device specific at the same time. To meet these challenges, the Generic Connection framework is first introduced in the CLDC. The idea of the Generic Connection framework is to define the abstractions of the networking and file I/O as general as possible to support a broad range of handheld devices, and leave the actual implementations of these abstractions to individual device manufacturers. These abstractions are defined as Java interfaces. The device manufacturers choose which one to implement in their MIDP based on the actual device capabilities.

There is 1 class (Connector) and 7 connection interfaces (Connection, ContentConnection, DatagramConnection, InputConnection, OutputConnection, StreamConnection, and StreamConnectionNotifier) defined in the Generic Connection framework. They can be found in the javax.microedition.io package that comes with J2ME CLDC. Please note that there is no implementation of the connection interfaces at the CLDC level. The actual implementation is left to MIDP.

The 7 connection interfaces define the abstractions of 6 basic types of communications: basic serial input, basic serial output, datagrams communications, sockets communications, notification mechanism in a client-server communication, and basic HTTP communication with a Web server.

The relationships between these interfaces are illustrated in the following diagram:

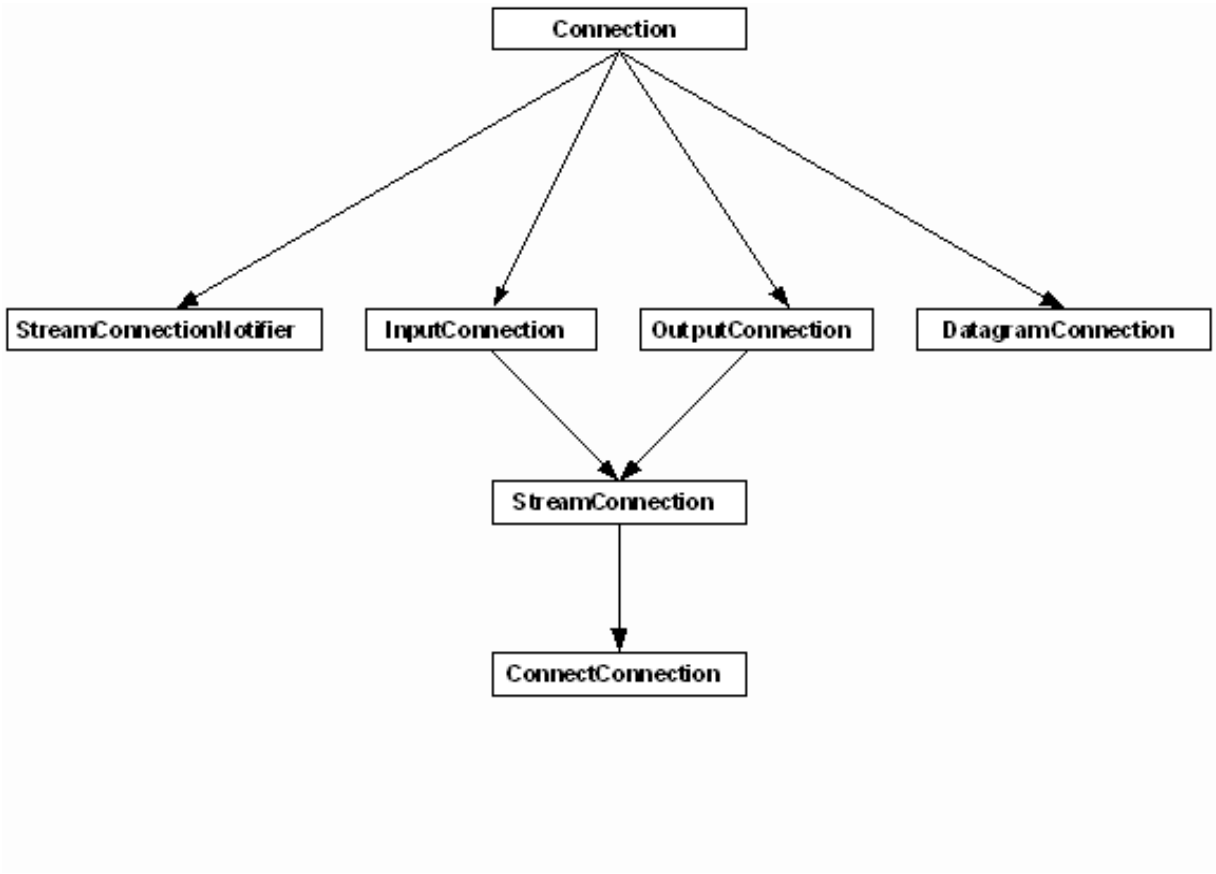


Figure (4.9) HTTP connection

4.2.3- SIP protocol

SIP employs design elements similar to HTTP-like request/response transaction model. Each transaction consists of a client request that invokes a particular method, or function, on the server and at least one response. SIP reuses most of the header fields, encoding rules and status codes of HTTP, providing a readable text-based format. SIP uses the Session Description Protocol (SDP), to exchange the session content.

SIP clients typically use TCP or UDP (typically on port 5060 and/or 5061) to connect to SIP servers and other SIP endpoints. SIP is primarily used in setting up and tearing down voice or video calls. However, it can be used in any application where session initiation is a requirement. These include Event Subscription and Notification, Terminal mobility and so on. All voice/video communications are done over separate session protocols, typically RTP.

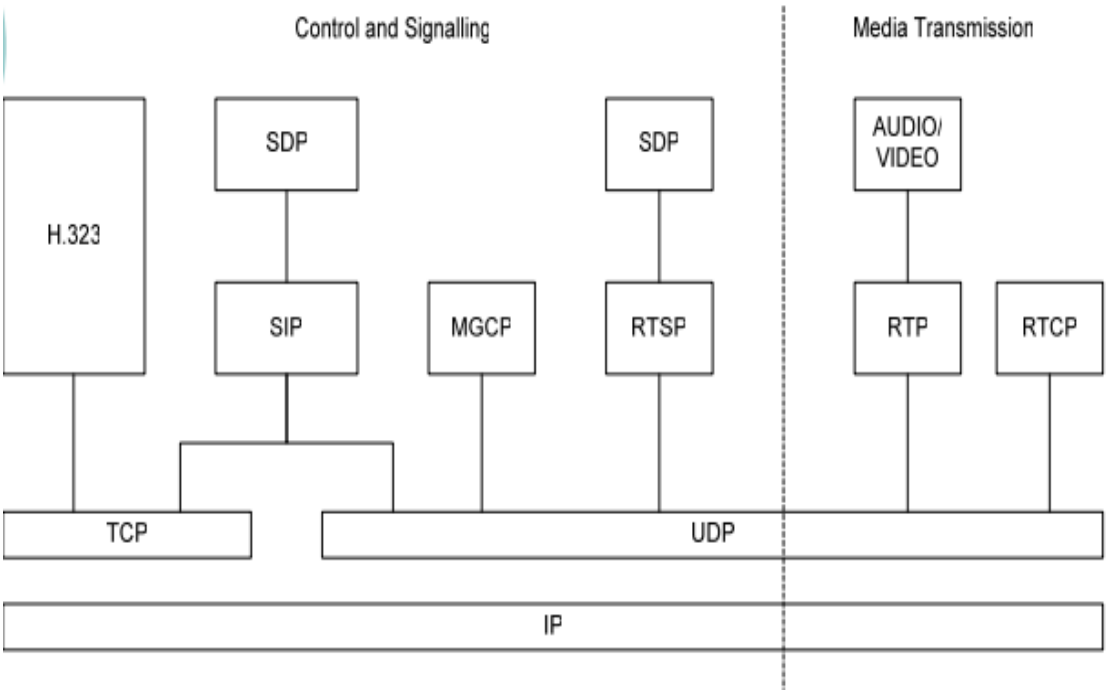


Figure (4.10) SIP Protocol

4.2.4-Socket Connection.

4.2.4.1- Internet socket

In computer networking, an **Internet socket** is the endpoint of a bidirectional communication flow across an Internet Protocol-based computer network, such as the Internet. *Internet sockets* (in plural) are an application programming interface (API) in an operating system, used for inter-process communication. Internet sockets constitute a mechanism for delivering incoming data packets to the appropriate application process or thread, based on a combination of local and remote IP addresses and port numbers. Each socket is mapped by the operational system to a communicating application process or thread [4].

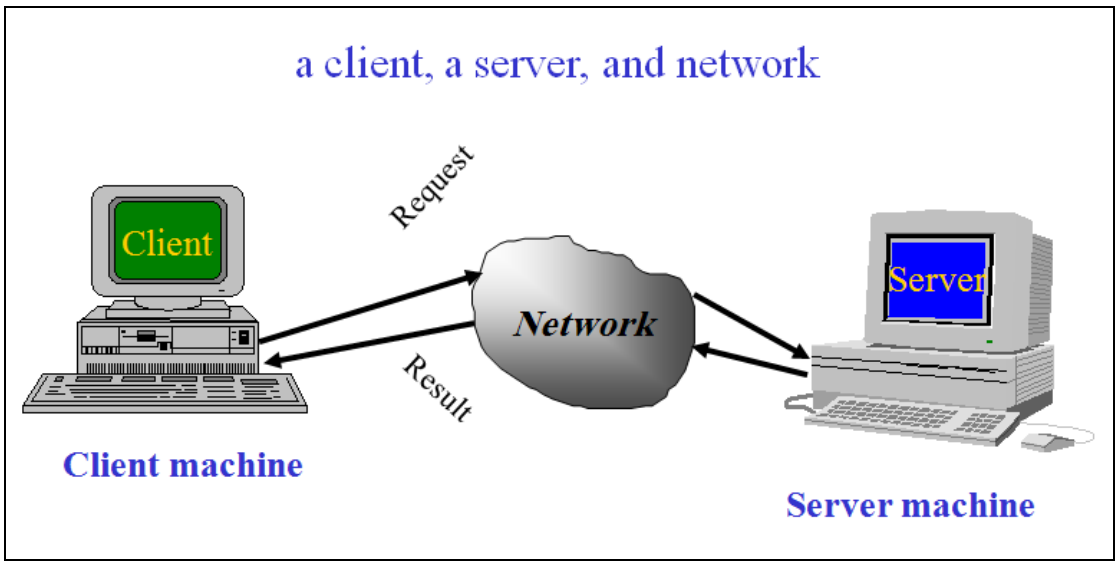


Figure (4.11) Socket Connection

4.2.4.2- Socket Types

- **User Datagram Protocol.**

The **User Datagram Protocol** (**UDP**) is one of the core members of the Internet Protocol Suite, the set of network protocols used for the Internet. With UDP, computer applications can send messages, in this case referred to as *datagram*, to other hosts on an Internet Protocol (IP) network without requiring prior communications to set up special transmission channels or data paths.

- **Transmission Control Protocol**

The **Transmission Control Protocol** (**TCP**) is one of the core protocols of the Internet Protocol Suite. TCP is one of the two original components, with Internet Protocol (IP), of the suite, so that the entire suite is commonly referred to as *TCP/IP*.

4.3 - Connection architecture.

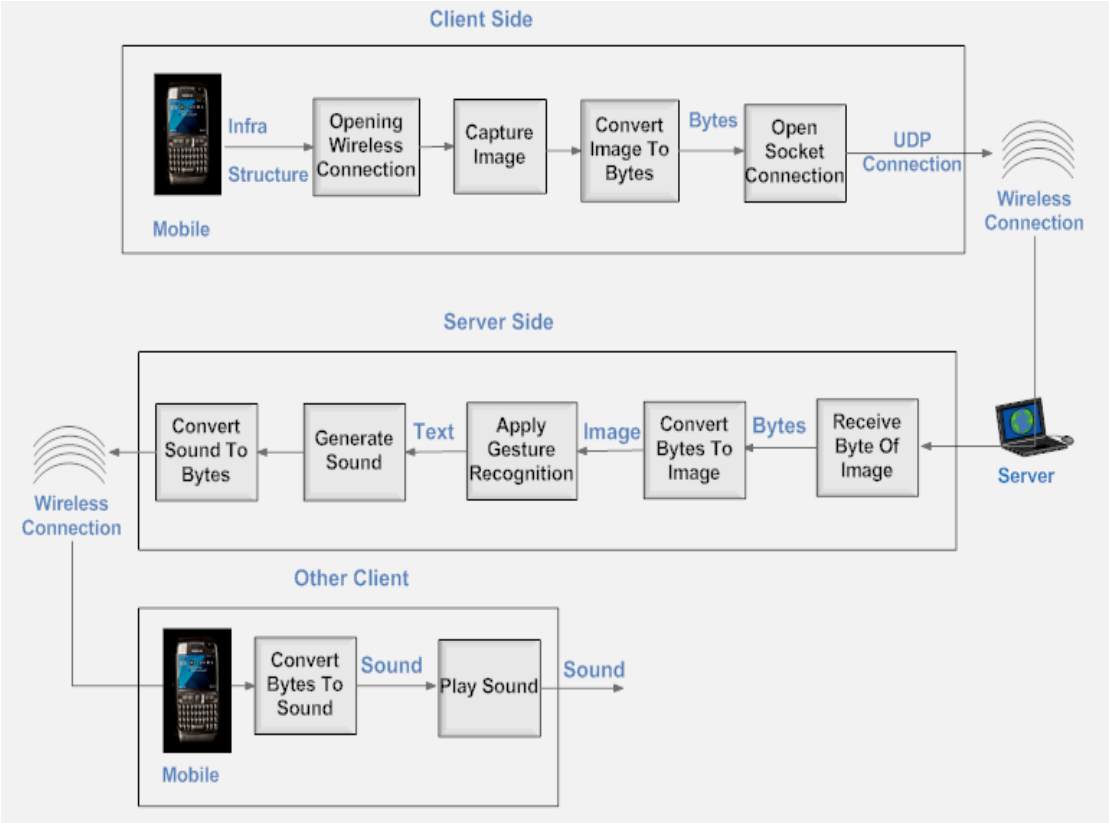


Figure (4.12) **Connection architecture.**

4.3.1 –Opening Wireless Connection.

These projects depend on infrastructure wireless connection between three devices[5][6][7].

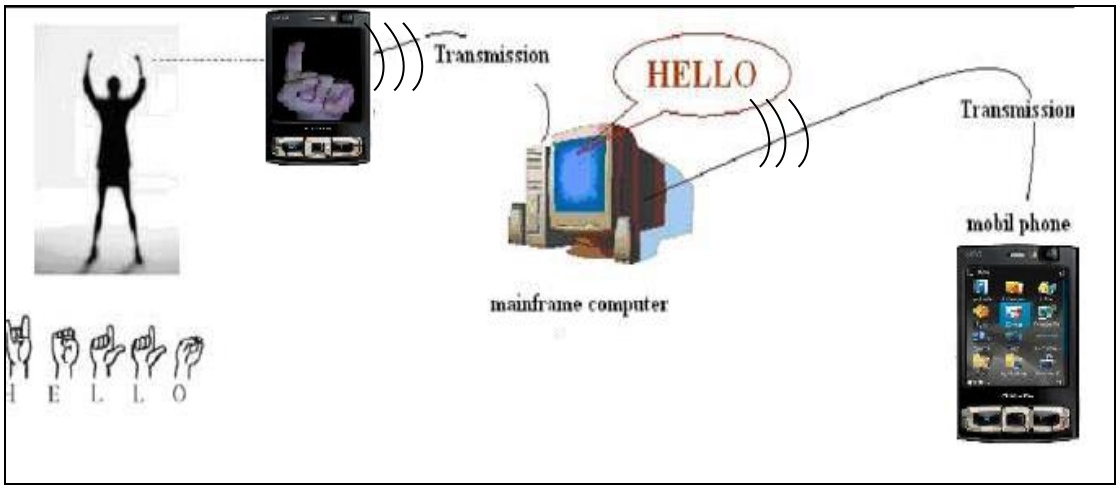
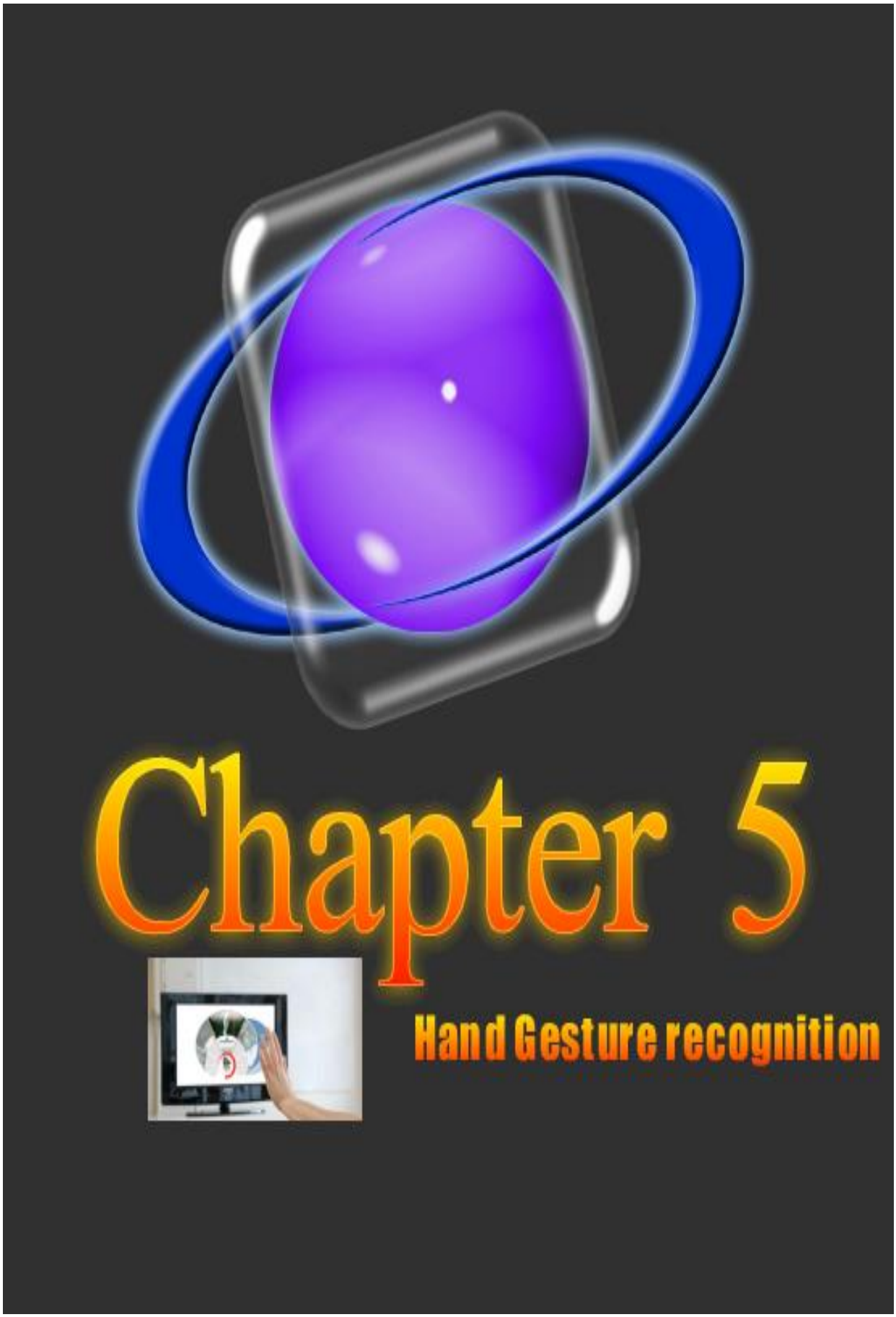


Figure (4.13) **Connection architecture**


Chapter 5
Hand Gesture recognition

Ch 5

Hand Gesture Recognition

5.1 –Introduction

This chapter discusses the most relevant research and publications related to the Project**.** In order to have an idea about the best approach we need to follow to build our system, it was crucial to survey already developed systems that are related to ours. To this end, we researched electronic databases and studied a set of ten papers that were most relevant to our system. We went on with analyzing and comparing the methods of implementation and development applied in each paper; after that, we came-up with an approach most suitable to develop our system. The ten articles surveyed are summarized below.

5.2. Recognition of the “gesture” components of the sign language

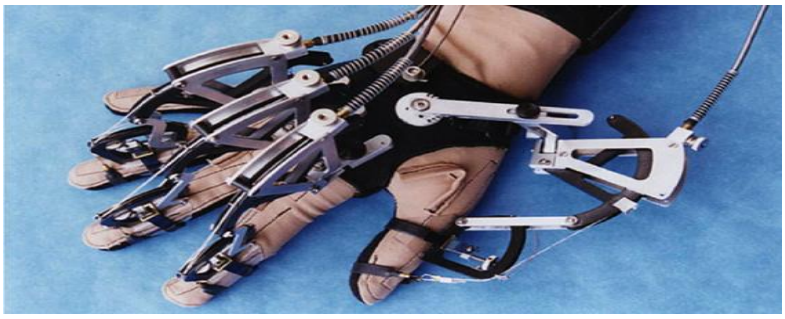
A sensor-based approach [] that can be employed in sign language description and recognition methods is based on recognition of the “gesture” components of the sign language context. The focus of other approaches was on the recognition of finger spellings and the simple linear motions created by the signer. However, such simplicity does not provide an accurate meaningful presentation for the signer. The actual gestures of the sign language are composed of complex motions accompanied by a sequence of continuous sign language words in a sentence. Therefore, a recognition method that is able to recognize complex gestures in a continuous pattern is quite effective. In this (gesture-based) method, the sign language is represented as a combination of gestures or intuitions of the hand (motion).This method uses a sensing glove as a way to input the sign language gestures; however, sensing gloves will cause motion impracticality for the signer, and gesture interpretation will be limited only to the motion of hands (neither face nor body expressions will take part in the interpretation).

5.3. Gesture recognition Alternatives

Gesture recognition represents the natural communication of humans in their daily life. Its importance is also reflected from its ability to be use in many important applications. The methods or alternatives to recognize any gesture is presented in

5.3.1. Sensor-based Approach

One technique for detecting the motion of or the gestures made by the hands is based on using sensory detectors. In most of the sensory-based approaches [8], a cyber-glove is used as a way of interfacing with the processor. The tiny Sensors integrated in each glove detect the position and can in some cases Measure the velocity of corresponding segments of the hand. During processing, each sensor translates the coordinates of its point into a certain Eigen space relative to a certain reference. The input data from the Sensors are manipulated by a certain algorithm that synthesizes it either into Recognized or unrecognized gestures.



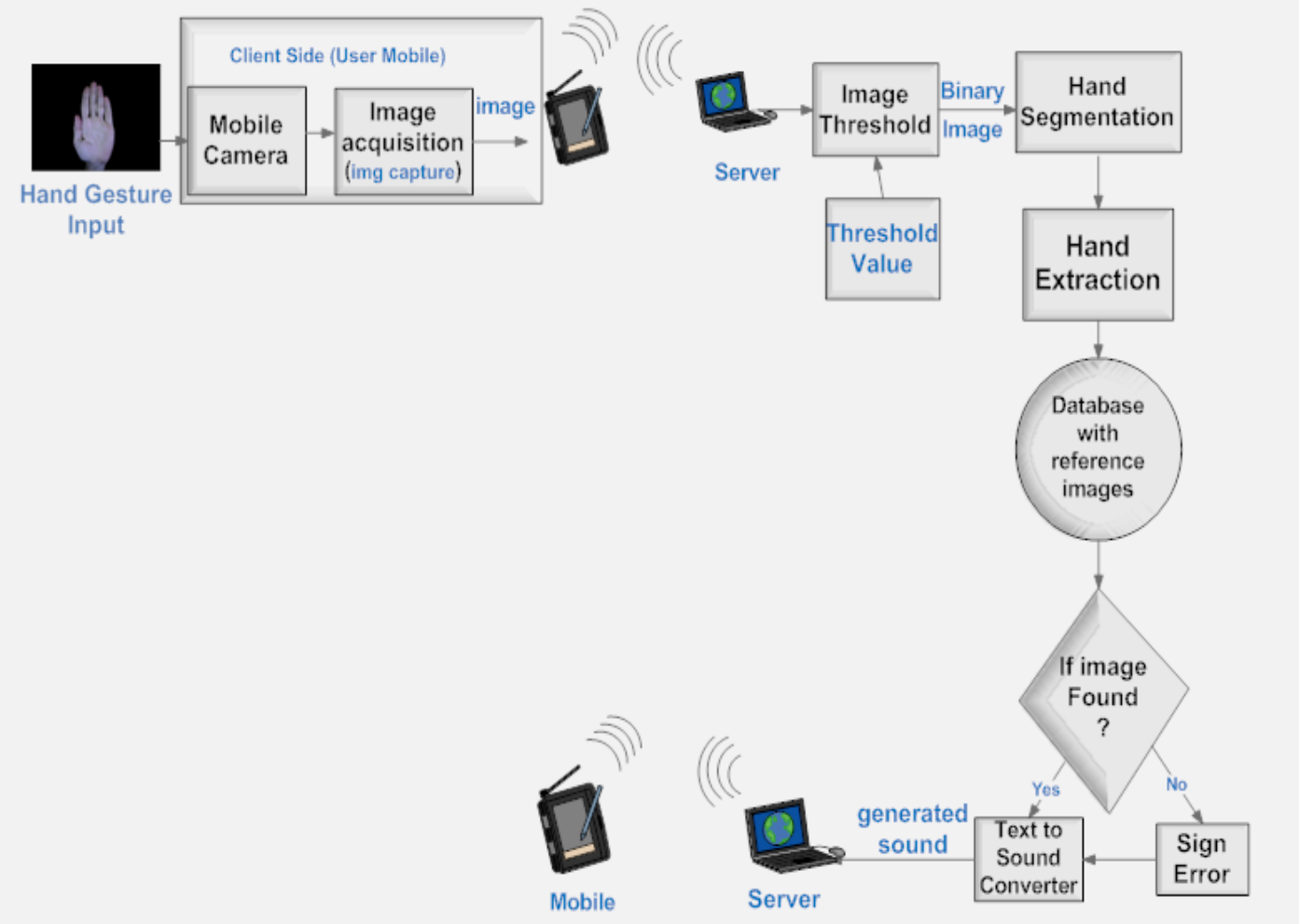
5.3.2. Image processing Approach

This second approach to recognition of gestures is based on image processing Where frames are captured using a video camera. After data is acquired, the Frames are then analyzed in order to detect the hands and face. The use of skin Color facilitates the differentiation of the hand and face from the background Colors using a specific color space. In some of the approaches, distinctly Colored gloves are used in order to have faster detection of the hands. The Motion of the hands or the face is tracked using parameters such as the centroid, area, and vertical and horizontal axis of the hands and face.

5.2.3. Sensors and Image Processing Approach

A third alternative is one that uses both sensors and image processing, each in a specific area. The sensors would be used to detect motion while image recognition method would be employed to detect still images. And since the sensors detect the exact angle in which the different body parts are positioned, a large number of sensors needs to be spread-out on the entire body; the fact that fact makes a system based on such an approach less portable. Adding to this, a system performing image recognition based on both image processing and sensors is more complex and costs more than each approach implemented alone. This is because it combines two technologies based on different principles that require distinct hardware and software components in order to function successfully.

5.4- System Architecture.



(Figure 5.1) Block diagram of whole System

5.5-Face and Hand Detection:

Dealing with detecting object from arbitrary background in most of situation become no longer difficult as the computer vision researches play magnificent role in this approach. Several alternatives exist to achieve this purpose as **background subtraction technique, skin segmentation, and modeling**. Talking about first alternative which includes four major steps: ***Preprocessing, background modeling, foreground detection, and data validation***. This alternative is used for separating potential hand pixels from non hand pixels [9]. The **skin segmentation technique** is based on locating skin regions in an unconstrained input image with a help from look up table (LUT) that may include most of possible skin color distribution. It includes two major approaches: parametric that works with either unimodel Gaussian density function [10] or multi model Gaussian mixture and nonparametric that is used for applying (LUTs) and Bayes classifier [11][12]. Finally the **Modeling technique** discusses the Model-Based detection which is known by active contours that have been proven as an efficient way to incorporate application-specific a prior knowledge into image or into computer vision algorithms. This approach includes two major types in modeling. The first Method is ***segmentation and contour extraction*** which concerned with find margins of each object in an image that are simply represented as a set of connected pixels. This Method is categorized in four classes (Edge-based approach, clustering-based approach, region-based approach, and split/merge approach). The second method is *object classification* **[13][14]** which is used to identify objects in the segmented area.

5.6-**Image Threshold**:

Is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images.

During the thresholding process, individual pixels in an image are marked as "object" pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as "background" pixels otherwise. This convention is known as *threshold above*. Variants include threshold below, which is opposite of threshold above; threshold inside, where a pixel is labeled "object" if its value is between two thresholds; and threshold outside, which is the opposite of threshold inside . Typically, an object pixel is given a value of "1" while a background pixel is given a value of "0." Finally, a binary image is created by coloring each pixel white or black, depending on a pixel's label.

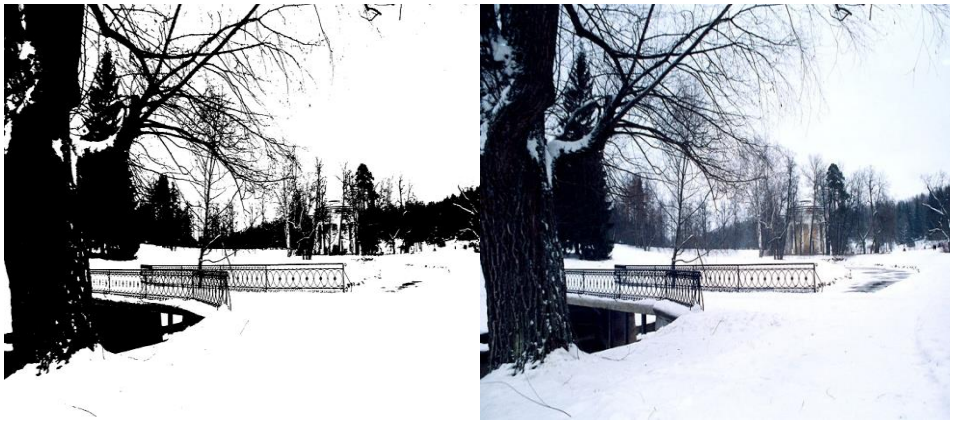


Figure (5.3) [a]-Black & White Inage [b]-colored image

5.6.1-Adaptive Thresholding:

Thresholding is called **adaptive thresholding** when a different threshold is used for different regions in the image. This may also be known as *local* or *dynamic* thresholding. The key parameter in the thresholding process is the choice of the threshold value. Several different methods for choosing a threshold exist; users can manually choose a threshold value, or a thresholding algorithm can compute a value automatically, which is known as *automatic thresholding.* A simple method would be to choose the mean or median value, the rationale being that if the object pixels are brighter than the background, they should also be brighter than the average. In a

noiseless image with uniform background and object values, the mean or median will work well as the threshold, however, this will generally not be the case.

A more sophisticated approach might be to create a histogram of the image pixel intensities and use the valley point as the threshold. The histogram approach assumes that there is some average value for the background and object pixels, but that the actual pixel values have some variation around these average values. However, this may be **computationally expensive**, and image histograms may not have clearly defined valley points, so adaptive threshold don't achieve required speed .

5.6.2-Histogram:

A **histogram** is a graphical display of tabulated frequencies, shown as bars. It shows what proportion of cases fall into each of several categories: it is a form of data binning. The categories are usually specified as non-overlapping intervals of some variable. The categories (bars) must be adjacent. The intervals (or bands, or bins) are generally of the same size.

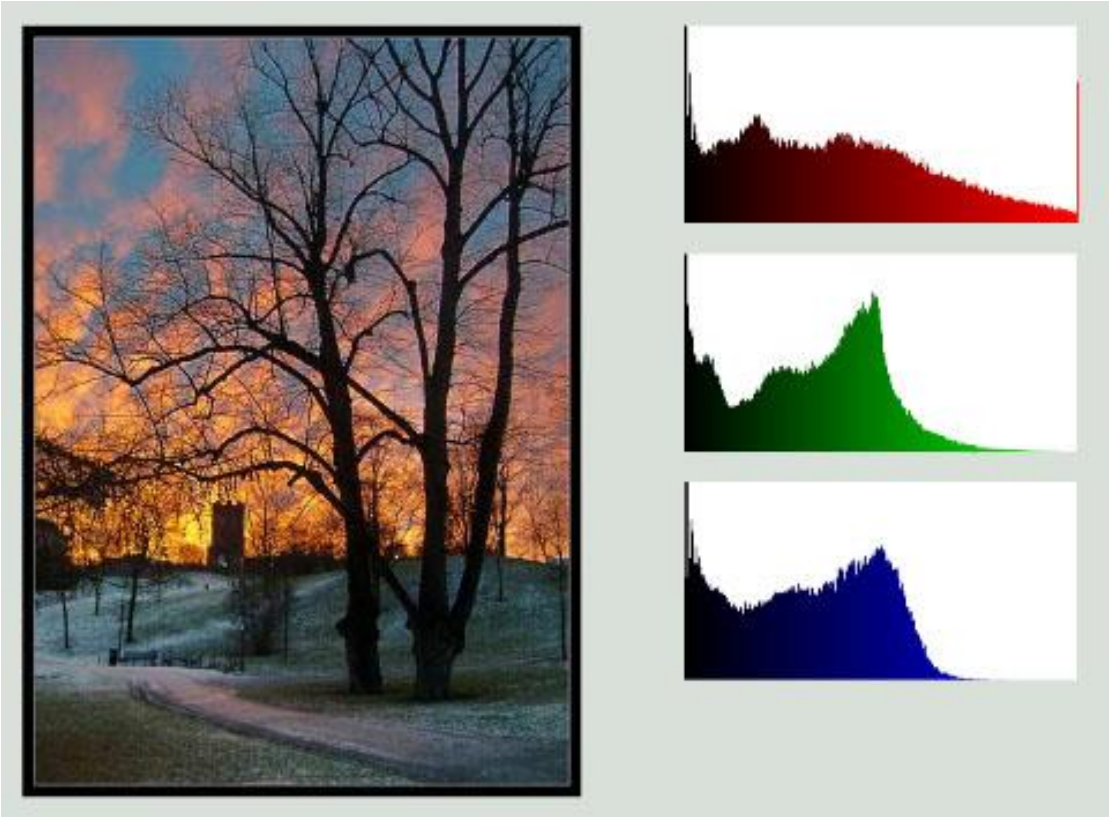


Figure (5.4) image and histogram of RGB channels

5.6.3-Clustering-Based Methods:

In this class of algorithms, the gray-level data undergoes a clustering analysis, with the number of clusters being set always to two. Since the two clusters correspond to the two lobes of a histogram, some authors search for the midpoint of the peaks the algorithm is based on the fitting of the mixture of Gaussians.

5.6.4- Entropy-Based Thresholding Methods:

This class of algorithms exploits the entropy of the distribution of the gray levels in a scene. The maximization of the entropy of the thresholded image is interpreted as indicative of maximum information transfer. Other ways try to minimize the cross-entropy between the input gray-level image and the output binary image as indicative of preservation of information, or a measure of fuzzy entropy .

5.7-Image Segmentation:

In computer vision, **segmentation** refers to the process of partitioning a digital image into multiple segments (sets of pixels) (Also known as super pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s).

Applications of Image Segmentation:

1-Locate objects in satellite images (roads, forests, etc.).

2-Fingerprint recognition.

3-Traffic control systems.

4-Hand recognition.

5.7.1-Edge Detection Method:

Edge detection is a well-developed field on its own within image processing. Region boundaries and edges are closely related, since there is often a sharp adjustment in intensity at the region boundaries. Edge detection techniques have therefore been used as the base of another segmentation technique. The edges identified by edge detection are often disconnected. To segment an object from an image however, **one needs closed region boundaries.** Discontinuities are bridged if the distance between the two edges is within some predetermined threshold. One such method is the edge linking method.

Figure (5.5) edge detection methods

5.7.2 Region growing methods:

The first region growing method was the seeded region growing method. This method takes a set of seeds as input along with the image. The seeds mark each of the objects to be segmented. The regions are iteratively grown by comparing all allocated neighboring pixels to the regions. The difference between a pixel's intensity value and the region's mean, δ, is used as a measure of similarity. The pixel with the smallest difference measured this way is allocated to the respective region. This process continues until all pixels are allocated to a region. Seeded region growing requires seeds as additional input. The segmentation results are dependent on the choice of seeds. Noise in the image can cause the seeds to be poorly placed. Unseeded region growing is a modified algorithm that doesn't require explicit seeds. It starts off with a single region A_1 – the pixel chosen here does not significantly influence final

segmentation. Pixels in the same way as seeded region growing. It differs from seeded region growing in that if the minimum δ is less than a predefined threshold T then it is added to the respective region A_j. If not, then the pixel is considered significantly different from all current regions A_i and a new region A_{n+1} is created with this pixel.

Figure (5.6) region growing.

5.7.3- Neural networks segmentation:

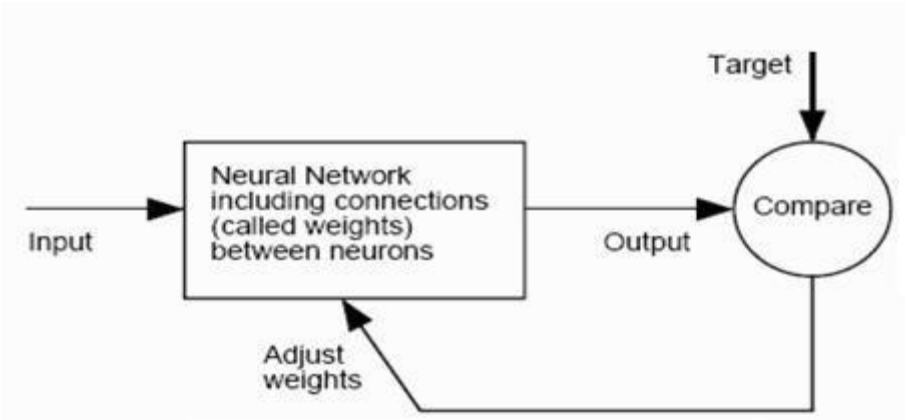


Figure (5.7) Simple Network

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. You can train a neural network to perform a particular function by adjusting the values of the

connections (weights) between elements. Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown below. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are needed to train a network.

Neural networks have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, and speech, vision, and control systems. Today neural networks can be trained to solve problems that are difficult for conventional computers or human beings. Throughout the toolbox emphasis is placed on neural network paradigms that build up to or are used in engineering, financial, and other practical applications.

[a] [b]

Figure (5.8) [a]- Black & White Image [b]- colored image

Image threshold is to convert colored image captured by Mobile camera and sent to server to Black & White image, database also contains Black & White images.

In the algorithm of threshold there are some constraints, the background should be Black to be able to segment hand from background and hand should be centroid in the center of image in front of Mobile camera to get good result.

5.7.4-Hand Segmentation Algorithms:

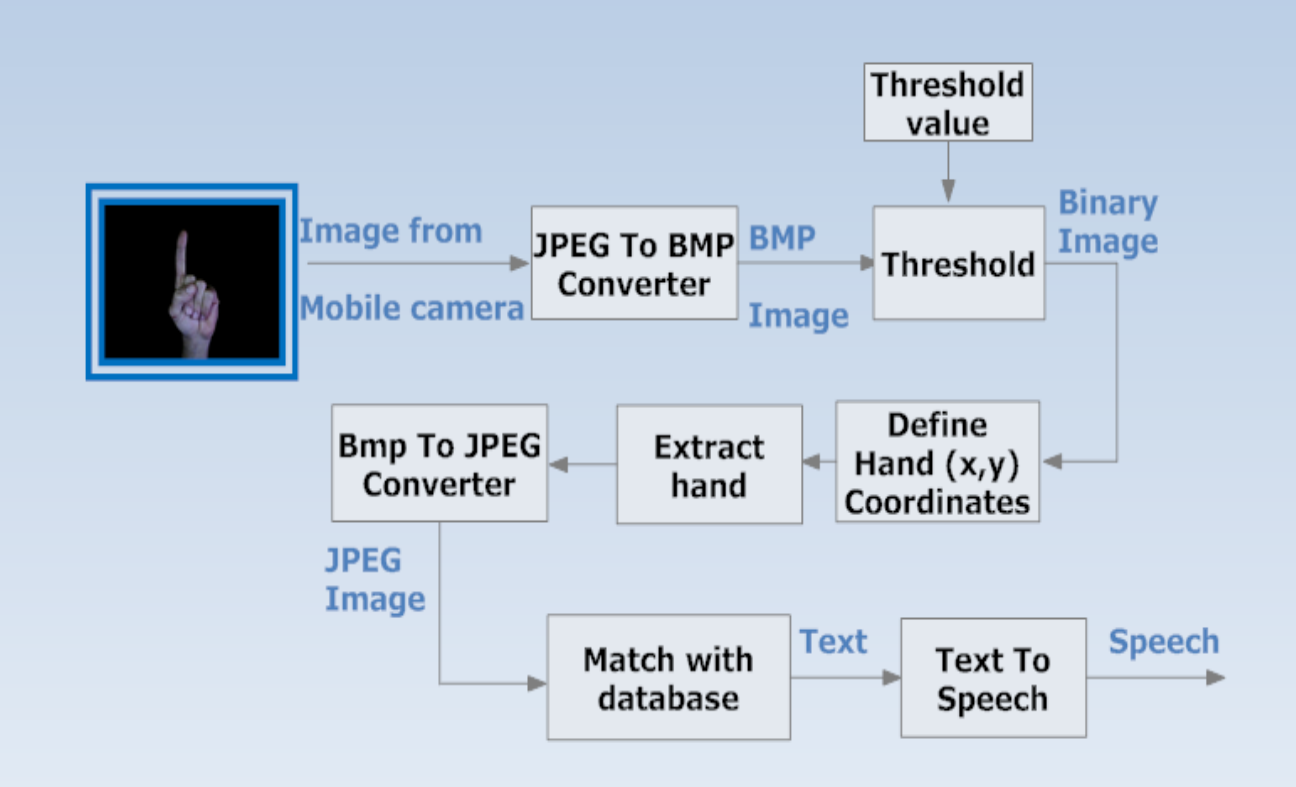


Figure (5.9) Hand Segmentation & Extraction

1-Image from mobile camera:

Image captured by mobile camera and send through wireless connection to the server as JPEG image.

2- JPEG to BMP Converter:

Convert image from JPEG format to BMP format.

3-Threshold:

After Converting the image into BMP it will be converted into Binary image (Black & White) for segmentation.

Firstly the background is black so the threshold value is as follow:

If the value greater than zero, it equal to one (Figure 5-9).

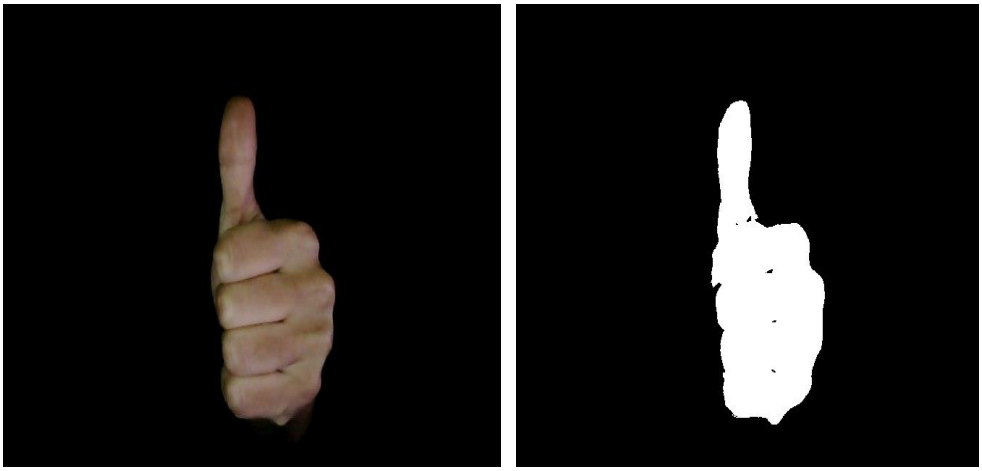


Figure (5-10) Threshold.

4-Hand Extraction:

Calculate the co-ordinates of the hand (white pixels) the top y co-ordinates, bottom y co-ordinates, left x co-ordinates and right x co-ordinates and make a border around it.

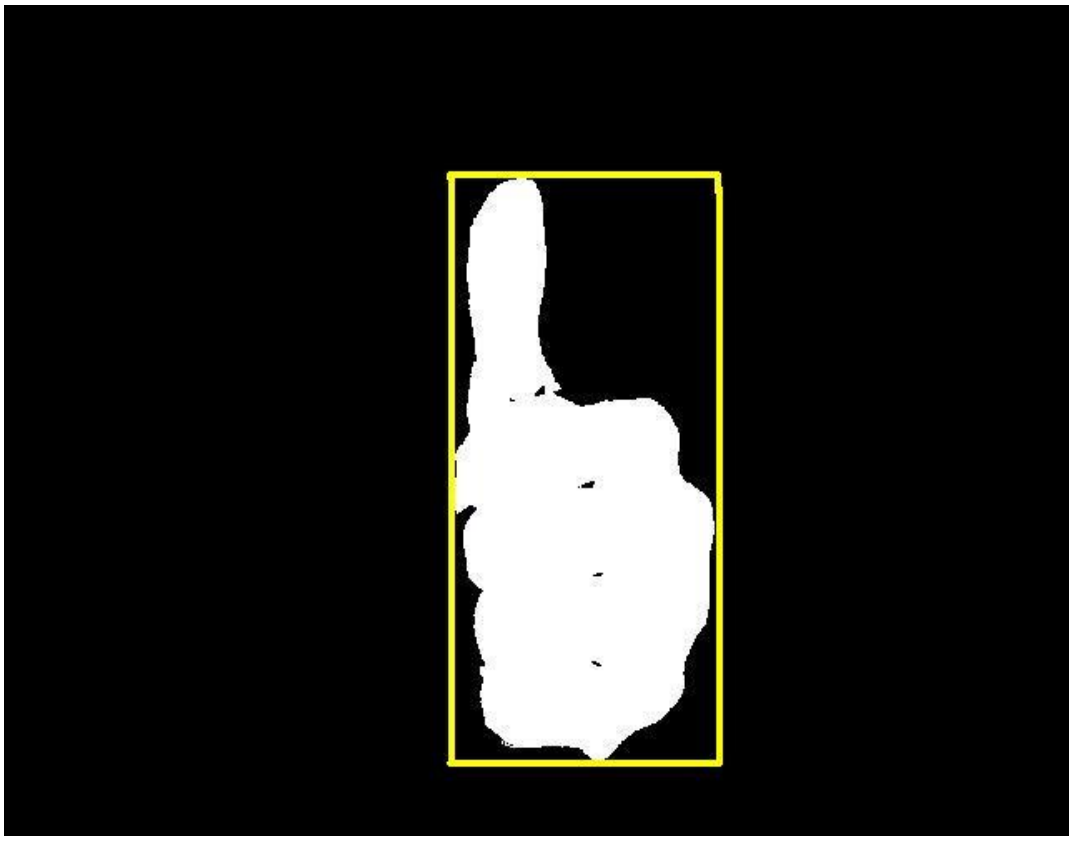


Figure (5-11) object extraction.

Object extraction is based on the background color Figure (5-10). The color of background is static (black) so any color not equal to (000=black) is from the object which will be extracted.

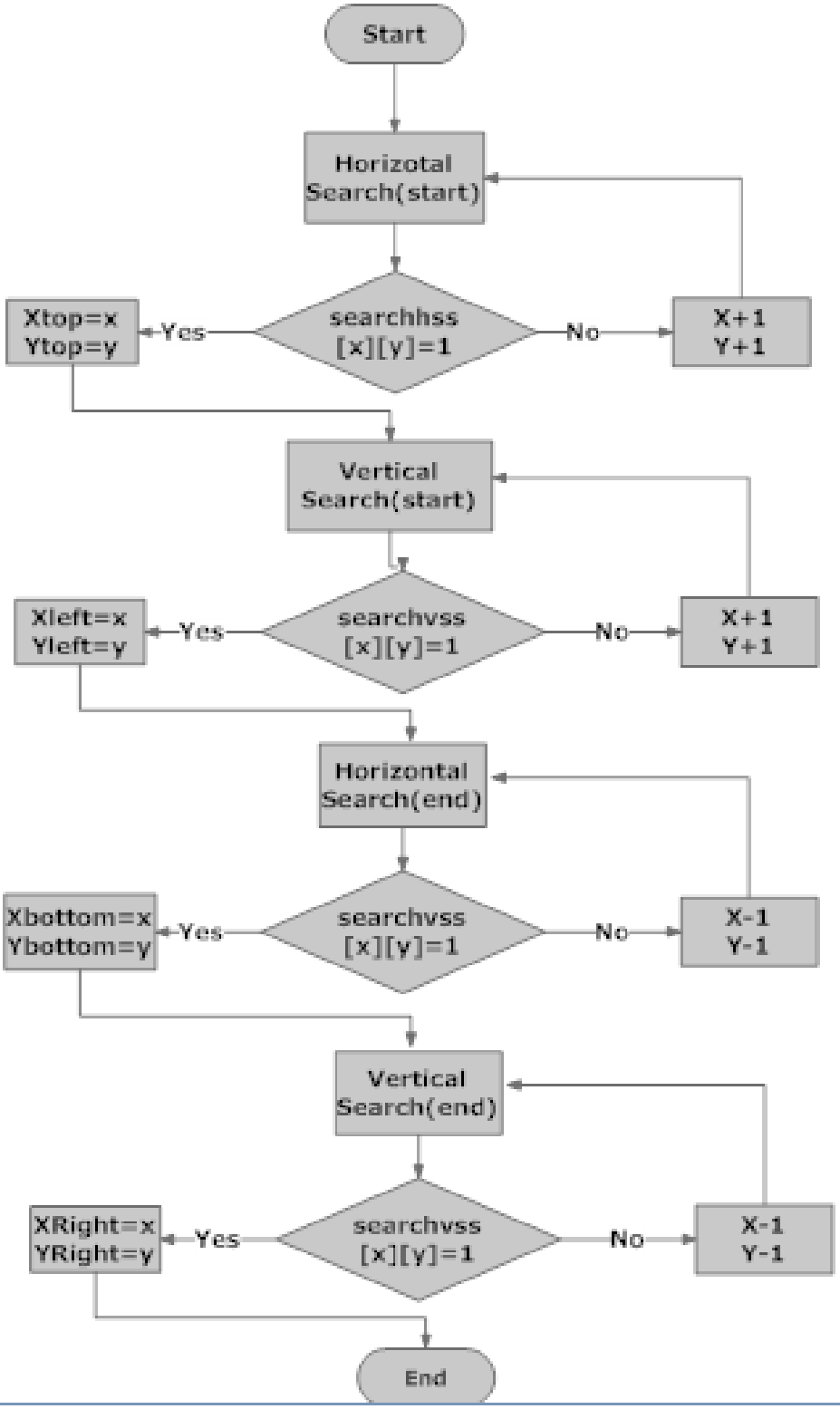


Figure (5-12) Hand Extraction Algorithm.

Four types of search are being made in a system may be called raster scan algorithm. In this algorithm there are four types of search for finding some attributes that helps in hand extraction .

1-Horizontal Search (start):

Begin from the start of the image and search horizontally till finding the first value of the hand (value=1) and then assign the coordinates of this value to Xtop and Ytop.

2-Vertical Search (start):

Begin from the start of the image also but searching vertically for finding Xleft and Yleft.

3-Horizontal Search (end):

Begin from the last corner of the image and decreasing horizontally until finding the Xbottom and Ybottom values.

4-Vertical Search (end):

Begin from the last corner of the image and decreasing vertically until finding the values of Xright and Yright.

After getting the values of (Xtop, Ytop, Xleft, Yleft, Xbottom, Ybottom, Xright, Yright) a new image frame is created using these coordinate for the extracted hand.

Figure (5.13) extracted hand from different input images

5-BMP to JPEG Converter:

Convert BMP format (hand that extracted after threshold) to JPEG format. The output of this stage is as follows.

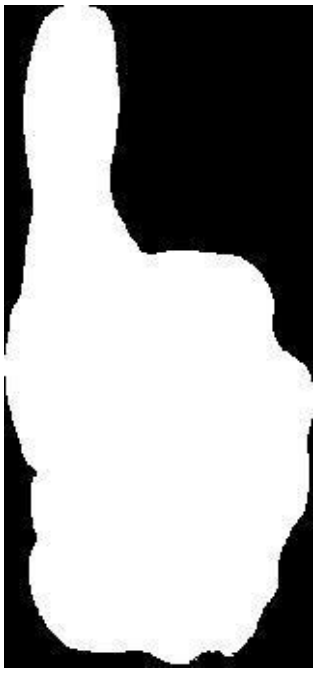


Figure (5.14) extracted hand from image

6- Matching extracted image with database:

If you want to compare images semantically, you will face a hard task that cannot be solved with simple algorithms. You will possibly have to process the image to extract a set of features that can be mapped to semantic objects even if those features are different in regard to some aspects (e.g. position, scale) and compare the objects' features instead of the pixel or image features.

The features for our test will be 25 RGB triples, corresponding to the average of the RGB values on the 25 regions marked in the figure on the left. The image will be normalized to 300x300 pixels. No texture or variance feature will be stored, only the color averages. Each region has 30x30 pixels.

Each image will be represented, then, a 25x3 feature vector. To calculate the similarity measure between two images A and B we will take each of the 25 regions, calculate the **Euclidean distance** between the regions and accumulate.

The distance from A to A will be, by definition, zero. The upper bound (maximum possible distance between two images, using this similarity measure method) is calculated as 25*(Math.sqrt ((255-0)*(255-0) + (255-0)*(255-0) + (255-0)*(255-0))) or a little bit over 11041.

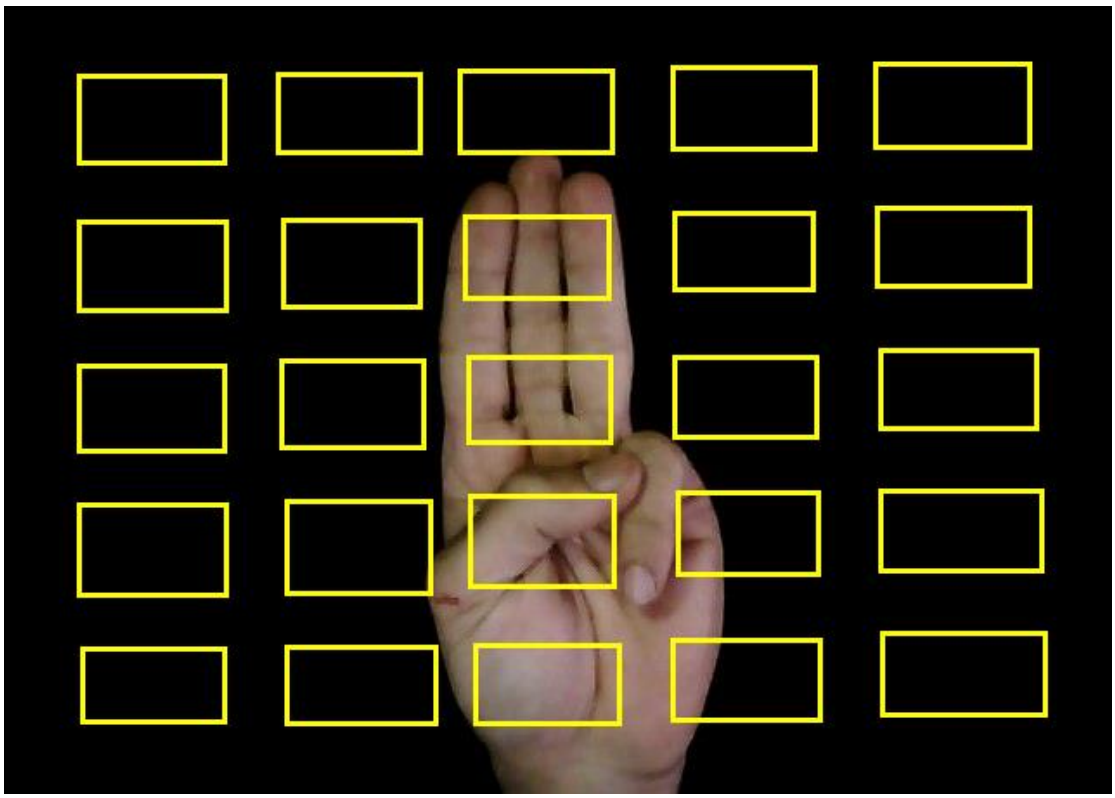
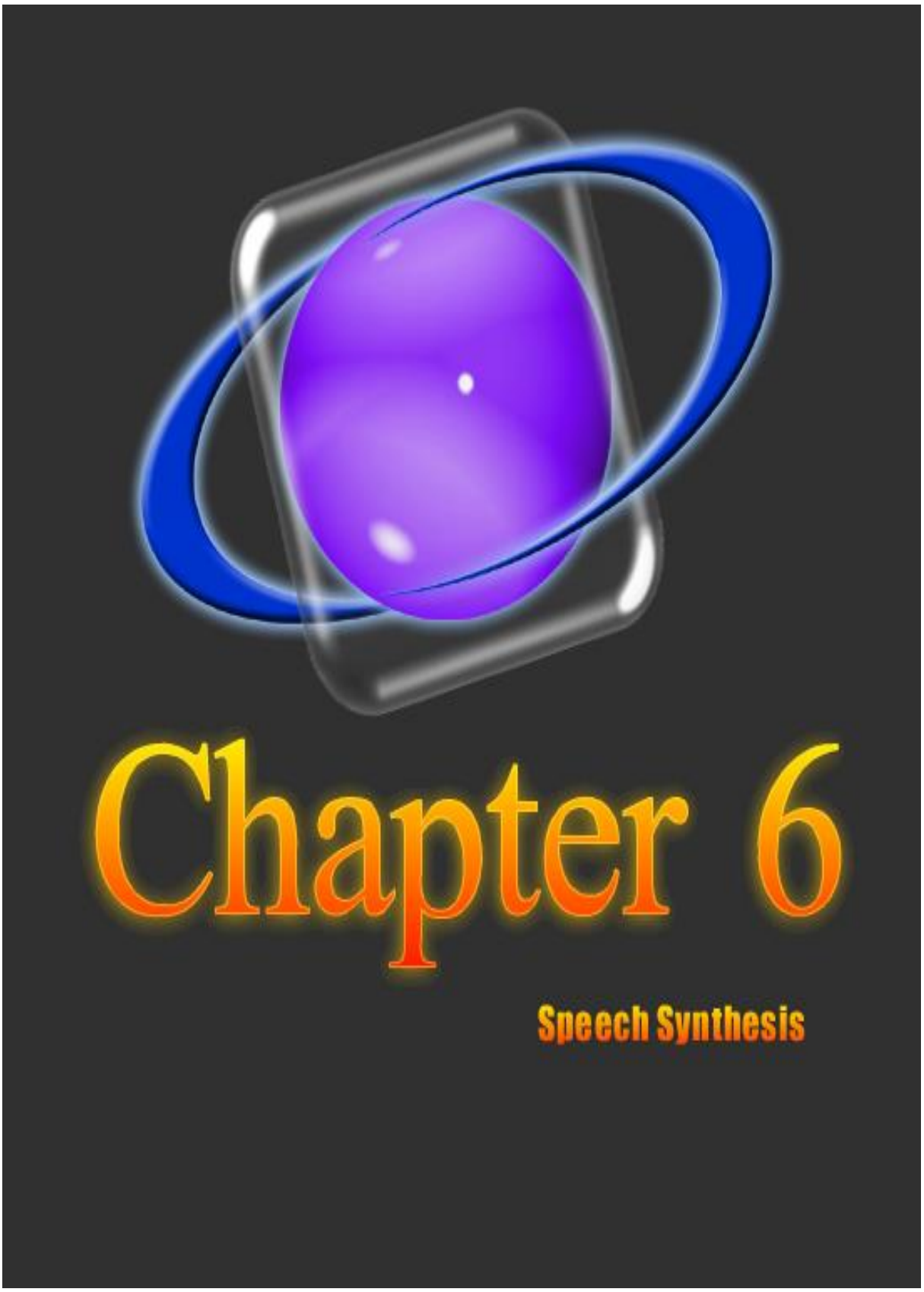


Figure (5.15) region dividing and feature extraction.

This method was chosen because it is simple to understand and implement and can be easily modified by the reader. It combines color (spectral) information with spatial (position/distribution) information, and is expected to be more robust (i.e. tolerant to differences) than comparing pixel by pixel or the average of the whole image, but, again, it is very simple and cannot be expected for perform well in any circumstances, being shown **only as an example**.


Chapter 6
Speech Synthesis

Ch 6
Speech Synthesis

6-1 Introduction:

This chapter introduces the main steps to generate the sound that specified for the whole project. These sound passes through different stages to reach the other mobile from server in a good form that other person understand it.

6-2 Java sound API:

The Java Sound API is a low-level API for effecting and controlling the input and output of sound media, including both audio and Musical Instrument Digital Interface (MIDI) data. The Java Sound API provides explicit control over the capabilities normally required for sound input and output, in a framework that promotes extensibility and flexibility. The Java Sound API provides the lowest level of sound support on the Java platform. It provides application programs with a great amount of control over sound operations, and it is extensible. For example, the Java Sound API supplies mechanisms for installing, accessing, and manipulating system resources such as audio mixers, MIDI synthesizers, other audio or MIDI devices, file readers and writers, and sound format converters. The Java Sound API does not include sophisticated sound editors or graphical tools, but it provides capabilities upon which such programs can be built. It emphasizes low-level control beyond that commonly expected by the end user.

6-2-1 Sampled Audio:

The `javax.sound.sampled` package handles digital audio data, which the Java Sound API refers to as sampled audio. *Samples* are successive snapshots of a signal. In the case of audio, the signal is a sound wave. A microphone converts the acoustic signal into a corresponding analog electrical signal, and an analog-to-digital converter transforms that analog signal into a sampled digital form. The following figure shows a brief moment in a sound recording.

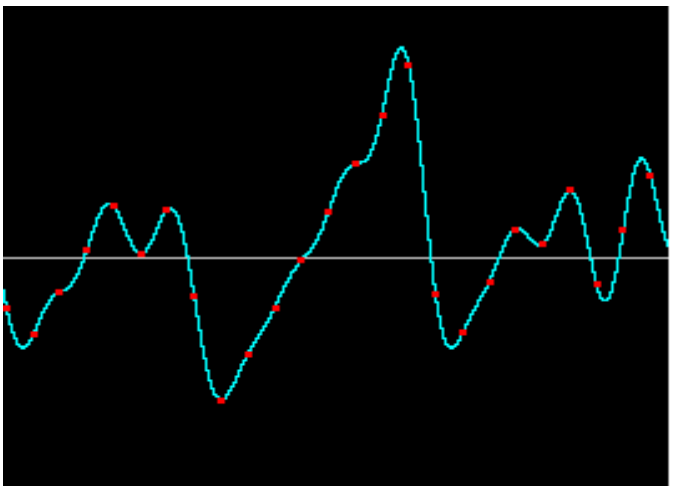


Figure (6-1) sampled sound wave

This graph plots sound pressure (amplitude) on the vertical axis, and time on the horizontal axis. The amplitude of the analog sound wave is measured periodically at a certain rate, resulting in the discrete samples (the red data points in the figure) that comprise the digital audio signal. The center horizontal line indicates zero amplitude; points above the line are positive-valued samples, and points below are negative. The accuracy of the digital approximation of the analog signal depends on its resolution in time (the *sampling rate*) and its *quantization*, or resolution in amplitude (the number of bits used to represent each sample). As a point of reference, the audio recorded for storage on compact discs is sampled 44,100 times per second and represented with 16 bits per sample.

The term "sampled audio" is used here slightly loosely. A sound wave could be sampled at discrete intervals while being left in an analog form. For purposes of the Java Sound API, however, "sampled audio" is equivalent to "digital audio."

Typically, sampled audio on a computer comes from a sound recording, but the sound could instead be synthetically generated (for example, to create the sounds of a touch-tone telephone). The term "sampled audio" refers to the type of data, not its origin.

The Java Sound API does not assume a specific audio hardware configuration; it is designed to allow different sorts of audio components to be installed on a system and accessed by the API. The Java Sound API supports common functionality such as input and output from a sound card (for example, for recording and playback of sound

files) as well as mixing of multiple streams of audio. Here is one example of a typical audio architecture:

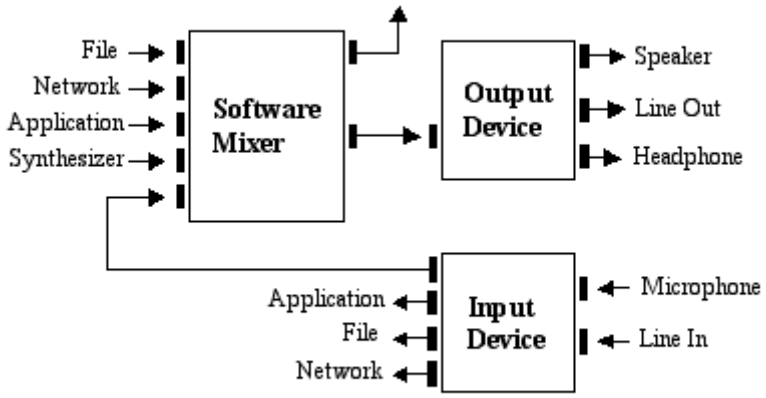


Figure (6-2) Audio architecture

In this example, a device such as a sound card has various input and output ports, and mixing is provided in the software. The mixer might receive data that has been read from a file, streamed from a network, generated on the fly by an application program, or produced by a MIDI synthesizer. The mixer combines all its audio inputs into a single stream, which can be sent to an output device for rendering.

6.2.2 - Musical Instrument Digital Interface (MIDI):

The javax.sound.midi package contains APIs for transporting and sequencing MIDI events, and for synthesizing sound from those events.

Whereas sampled audio is a direct representation of a sound itself, *MIDI data* can be thought of as a recipe for creating a sound, especially a musical sound. MIDI data, unlike audio data, does not describe sound directly. Instead, it describes events that affect the sounds (or actions) performed by a MIDI-enabled device or instrument, such as a synthesizer. MIDI data is analogous to a graphical user interface's keyboard and mouse events. In the case of MIDI, the events can be thought of as actions upon a musical keyboard, along with actions on various pedals, sliders, switches, and knobs on that musical instrument. These events need not actually originate with a hardware

musical instrument; they can be simulated in software, and they can be stored in MIDI files. A program that can create, edit, and perform these files is called a sequencer. Many computer sound cards include MIDI-controllable music synthesizer chips to which sequencers can send their MIDI events. Synthesizers can also be implemented entirely in software. The synthesizers interpret the MIDI events that they receive and produce audio output. Usually the sound synthesized from MIDI data is musical sound (as opposed to speech, for example). MIDI synthesizers are also capable of generating various kinds of sound effects.

Some sound cards include MIDI input and output ports to which external MIDI hardware devices (such as keyboard synthesizers or other instruments) can be connected. From a MIDI input port, an application program can receive events generated by an external MIDI-equipped musical instrument. The program might play the musical performance using the computer's internal synthesizer, save it to disk as a MIDI file, or render it into musical notation. A program might use a MIDI output port to play an external instrument, or to control other external devices such as recording equipment.

The following diagram illustrates the functional relationships between the major components in a possible MIDI configuration based on the Java Sound API. (As with audio, the Java Sound API permits a variety of MIDI software devices to be installed and interconnected. The system shown here is just one potential scenario.) The flow of data between components is indicated by arrows. The data can be in a standard file format, or (as indicated by the key in the lower right corner of the diagram), it can be audio, raw MIDI bytes, or time-tagged MIDI messages.

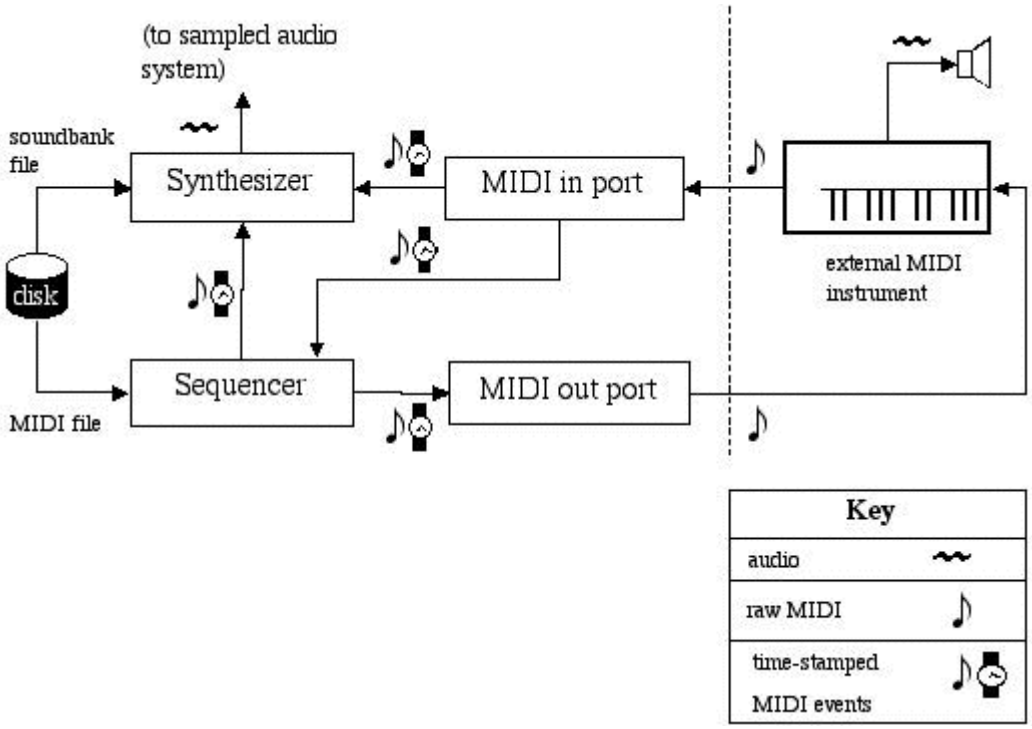


Figure (6-3) A possible MIDI Configuration

In this example, the application program prepares a musical performance by loading a musical score that's stored as a standard MIDI file on a disk (left side of the diagram). Standard MIDI files contain tracks, each of which is a list of time-tagged MIDI events. Most of the events represent musical notes (pitches and rhythms). This MIDI file is read and then "performed" by a software sequencer. A sequencer performs its music by sending MIDI messages to some other device, such as an internal or external synthesizer. The synthesizer itself may read a soundbank file containing instructions for emulating the sounds of certain musical instruments. If not, the synthesizer will play the notes stored in the MIDI file using whatever instrument sounds are already loaded into it.

As illustrated, the MIDI events must be translated into raw (non-time-tagged) MIDI before being sent through a MIDI output port to an external MIDI instrument. Similarly, raw MIDI data coming into the computer from an external MIDI source (a keyboard instrument, in the diagram) is translated into time-tagged MIDI messages that can control a synthesizer, or that a sequencer can store for later use.

6.3 - Formatted Audio Data:

Formatted audio data refers to sound in any of a number of standard formats. The Java Sound API distinguishes between *data formats* and *file formats*.

6.3.1 – Data Formats:

A data format tells you how to interpret a series of bytes of "raw" sampled audio data, such as samples that have already been read from a sound file, or samples that have been captured from the microphone input. You might need to know, for example, how many bits constitute one sample (the representation of the shortest instant of sound), and similarly you might need to know the sound's sample rate (how fast the samples are supposed to follow one another). When setting up for playback or capture, you specify the data format of the sound you are capturing or playing.

In the Java Sound API, a data format is represented by an `AudioFormat` object, which includes the following attributes:

- Encoding technique, usually pulse code modulation (PCM)
- Number of channels (1 for mono, 2 for stereo, etc.)
- Sample rate (number of samples per second, per channel)
- Number of bits per sample (per channel)
- Frame rate
- Frame size in bytes
- Byte order (big-endian or little-endian)

PCM is one kind of encoding of the sound waveform. The Java Sound API includes two PCM encodings that use linear quantization of amplitude, and signed or unsigned integer values. Linear quantization means that the number stored in each sample is directly proportional (except for any distortion) to the original sound pressure at that instant—and similarly proportional to the displacement of a loudspeaker or eardrum that is vibrating with the sound at that instant. Compact discs, for example, use linear PCM-encoded sound. Mu-law encoding and a-law encoding are common nonlinear encodings that provide a more compressed version of the audio data; these encodings are typically used for telephony or recordings of speech. A nonlinear encoding maps the original sound's amplitude to the stored value using a nonlinear function, which

can be designed to give more amplitude resolution to quiet sounds than to loud sounds.

A frame contains the data for all channels at a particular time. For PCM-encoded data, the frame is simply the set of simultaneous samples in all channels, for a given instant in time, without any additional information. In this case, the frame rate is equal to the sample rate, and the frame size in bytes is the number of channels multiplied by the sample size in bits, divided by the number of bits in a byte.

For other kinds of encodings, a frame might contain additional information besides the samples, and the frame rate might be completely different from the sample rate. For example, consider the MP3 (MPEG-1 Audio Layer 3) encoding, which is not explicitly mentioned in the current version of the Java Sound API, but which could be supported by an implementation of the Java Sound API or by a third-party service provider. In MP3, each frame contains a bundle of compressed data for a series of samples, not just one sample per channel. Because each frame encapsulates a whole series of samples, the frame rate is slower than the sample rate. The frame also contains a header. Despite the header, the frame size in bytes is less than the size in bytes of the equivalent number of PCM frames. (After all, the purpose of MP3 is to be more compact than PCM data.) For such an encoding, the sample rate and sample size refer to the PCM data that the encoded sound will eventually be converted into before being delivered to a digital-to-analog converter (DAC).

6.3.2- File formats:

A file format specifies the structure of a sound file, including not only the format of the raw audio data in the file, but also other information that can be stored in the file. Sound files come in various standard varieties, such as WAVE (also known as WAV, and often associated with PCs), AIFF (often associated with Macintoshes), and AU (often associated with UNIX systems). The different types of sound file have different structures. For example, they might have a different arrangement of data in the file's "header." A header contains descriptive information that typically precedes the file's actual audio samples, although some file formats allow successive "chunks"

of descriptive and audio data. The header includes a specification of the data format that was used for storing the audio in the sound file. Any of these types of sound file can contain various data formats (although usually there is only one data format within a given file), and the same data format can be used in files that have different file formats.

In the Java Sound API, a file format is represented by an `AudioFileFormat` object, which contains:

- The file type (WAVE, AIFF, etc.)
- The file's length in bytes
- The length, in frames, of the audio data contained in the file
- An AudioFormat object that specifies the data format of the audio data contained in the file

The `AudioSystem` class provides methods for reading and writing sounds in different file formats, and for converting between different data formats. Some of the methods let you access a file's contents through a kind of stream called an `AudioInputStream` . An `AudioInputStream` is a subclass of the `InputStream` class, which encapsulates a series of bytes that can be read sequentially. To its superclass, the `AudioInputStream` class adds knowledge of the bytes' audio data format (represented by an `AudioFormat` object). By reading a sound file as an `AudioInputStream`, you get immediate access to the samples, without having to worry about the sound file's structure (its header, chunks, etc.). A single method invocation gives you all the information you need about the data format and the file type.

6.4- Accessing Audio System Resources:

The Java Sound API takes a flexible approach to system configuration. Different sorts of audio devices (mixers) can be installed on a computer. The API makes few assumptions about what devices have been installed and what their capabilities are. Instead, it provides ways for the system to report about the available audio components, and ways for your program to access them.

The following sections show how your program can learn what sampled-audio resources have been installed on the computer, and how it can gain access to the

available resources. Among other things, the resources include mixers and the various types of lines owned by the mixers.

6.4.2 - Permission to Use Audio Resources:

The Java Sound API includes an `AudioPermission` class that indicates what kinds of access an applet (or an application running with a security manager) can have to the sampled-audio system. Permission to record sound is controlled separately. This permission should be granted with care, to help prevent security risks such as unauthorized eavesdropping. By default, applets and applications are granted permissions as follows:

- An *applet* running with the applet security manager can play, but not record, audio.
- An *application* running with no security manager can both play and record audio.
- An application running with the default security manager can play, but not record, audio.

In general, applets are run under the scrutiny of a security manager and aren't permitted to record sound. Applications, on the other hand, don't automatically install a security manager, and are able to record sound. (However, if the default security manager is invoked explicitly for an application, the application isn't permitted to record sound.)

Both applets and applications can record sound even when running with a security manager if they have been granted explicit permission to do so.

If your program doesn't have permission to record (or play) sound, an exception will be thrown when it attempts to open a line. There is nothing you can do about this in your program, other than to catch the exception and report the problem to the user, because permissions can't be changed through the API. (If they could, they would be pointless, because nothing would be secure!) Generally, permissions are set in one or more policy configuration files, which a user or system administrator can edit using a text editor or the Policy Tool program.

6.5 - Synthesizing Sound:

Most programs that avail themselves of the Java Sound API's MIDI package do so to synthesize sound. The entire apparatus of MIDI files, events, sequences, and sequencers, which was previously discussed, nearly always has the goal of eventually sending musical data to a synthesizer to convert into audio. (Possible exceptions include programs that convert MIDI into musical notation that can be read by a musician, and programs that send messages to external MIDI-controlled devices such as mixing consoles.)

The `Synthesizer` interface is therefore fundamental to the MIDI package. This page shows how to manipulate a synthesizer to play sound. Many programs will simply use a sequencer to send MIDI file data to the synthesizer, and won't need to invoke many `Synthesizer` methods directly. However, it's possible to control a synthesizer directly, without using sequencers or even `MidiMessage` objects, as explained near the end of this page.

The synthesis architecture might seem complex for readers who are unfamiliar with MIDI. Its API includes three interfaces:

- **Synthesizer**
- **MidiChannel**
- **Soundbank**

and four classes:

- **Instrument**
- **Patch**
- **SoundbankResource**
- **VoiceStatus**

As orientation for all this API, the next section explains some of the basics of MIDI synthesis and how they're reflected in the Java Sound API. Subsequent sections give a more detailed look at the API.

6.5.1- MIDI Synthesis.

How does a synthesizer generate sound? Depending on its implementation, it may use one or more sound-synthesis techniques. For example, many synthesizers use wavetable synthesis. A wavetable synthesizer reads stored snippets of audio from memory, playing them at different sample rates and looping them to create notes of different pitches and durations. For example, to synthesize the sound of a saxophone playing the note C#4 (MIDI note number 61), the synthesizer might access a very short snippet from a recording of a saxophone playing the note Middle C (MIDI note number 60), and then cycle repeatedly through this snippet at a slightly faster sample rate than it was recorded at, which creates a long note of a slightly higher pitch. Other synthesizers use techniques such as frequency modulation (FM), additive synthesis, or physical modeling, which don't make use of stored audio but instead generate audio from scratch using different algorithms.

Instruments

What all synthesis techniques have in common is the ability to create many sorts of sounds. Different algorithms, or different settings of parameters within the same algorithm, create different-sounding results. An *instrument* is a specification for synthesizing a certain type of sound. That sound may emulate a traditional musical instrument, such as a piano or violin; it may emulate some other kind of sound source, for instance, a telephone or helicopter; or it may emulate no "real-world" sound at all. A specification called General MIDI defines a standard list of 128 instruments, but most synthesizers allow other instruments as well. Many synthesizers provide a collection of built-in instruments that are always available for use; some synthesizers also support mechanisms for loading additional instruments.

An instrument may be vendor-specific—in other words, applicable to only one synthesizer or several models from the same vendor. This incompatibility results when two different synthesizers use different sound-synthesis techniques, or different internal algorithms and parameters even if the fundamental technique is the same. Because the details of the synthesis technique are often proprietary, incompatibility is common. The Java Sound API includes ways to detect whether a given synthesizer supports a given instrument.

An instrument can usually be considered a preset; you don't have to know anything about the details of the synthesis technique that produces its sound. However, you can still vary aspects of its sound. Each Note On message specifies the pitch and volume of an individual note. You can also alter the sound through other MIDI commands such as controller messages or system-exclusive messages.

Channels

Many synthesizers are *multimbral* (sometimes called *polytimbral*), meaning that they can play the notes of different instruments simultaneously. (*Timbre* is the characteristic sound quality that enables a listener to distinguish one kind of musical instrument from other kinds.) Multimbral synthesizers can emulate an entire ensemble of real-world instruments, instead of only one instrument at a time. MIDI synthesizers normally implement this feature by taking advantage of the different MIDI channels on which the MIDI specification allows data to be transmitted. In this case, the synthesizer is actually a collection of sound-generating units, each emulating a different instrument and responding independently to messages that are received on a different MIDI channel. Since the MIDI specification provides only 16 channels, a typical MIDI synthesizer can play up to 16 different instruments at once. The synthesizer receives a stream of MIDI commands, many of which are channel commands. (Channel commands are targeted to a particular MIDI channel; for more information, see the MIDI specification.) If the synthesizer is multitimbral, it routes each channel command to the correct sound-generating unit, according to the channel number indicated in the command.

In the Java Sound API, these sound-generating units are instances of classes that implement the `MidiChannel` interface. A `synthesizer` object has at least one `MidiChannel` object. If the synthesizer is multimbral, it has more than one, normally 16. Each `MidiChannel` represents an independent sound-generating unit.

Because a synthesizer's `MidiChannel` objects are more or less independent, the assignment of instruments to channels doesn't have to be unique. For example, all 16 channels could be playing a piano timbre, as though there were an ensemble of 16 pianos. Any grouping is possible—for instance, channels 1, 5, and 8 could be playing guitar sounds, while channels 2 and 3 play percussion and channel 12 has a bass

timbre. The instrument being played on a given MIDI channel can be changed dynamically; this is known as a *program change*.

Even though most synthesizers allow only 16 or fewer instruments to be active at a given time, these instruments can generally be chosen from a much larger selection and assigned to particular channels as required.

Soundbanks and Patches

Instruments are organized hierarchically in a synthesizer, by bank number and program number. Banks and programs can be thought of as rows and columns in a two-dimensional table of instruments. A bank is a collection of programs. The MIDI specification allows up to 128 programs in a bank, and up to 128 banks. However, a particular synthesizer might support only one bank, or a few banks, and might support fewer than 128 programs per bank.

In the Java Sound API, there's a higher level to the hierarchy: a soundbank. Soundbanks can contain up to 128 banks, each containing up to 128 instruments. Some synthesizers can load an entire soundbank into memory.

To select an instrument from the current soundbank, you specify a bank number and a program number. The MIDI specification accomplishes this with two MIDI commands: bank select and program change. In the Java Sound API, the combination of a bank number and program number is encapsulated in a `Patch` object. You change a MIDI channel's current instrument by specifying a new patch. The patch can be considered the two-dimensional index of the instruments in the current soundbank.

You might be wondering if soundbanks, too, are indexed numerically. The answer is no; the MIDI specification does not provide for this. In the Java Sound API, a `Soundbank` object can be obtained by reading a soundbank file. If the soundbank is supported by the synthesizer, its instruments can be loaded into the synthesizer individually as desired, or all at once. Many synthesizers have a built-in or default soundbank; the instruments contained in this soundbank are always available to the synthesizer.

Voices

It's important to distinguish between the number of *timbres* a synthesizer can play simultaneously and the number of *notes* it can play simultaneously. The former was described above under "Channels." The ability to play multiple notes at once is referred to as *polyphony*. Even a synthesizer that isn't multitimbral can generally play more than one note at a time (all having the same timbre, but different pitches). For example, playing any chord, such as a G major triad or a B minor seventh chord, requires polyphony. Any synthesizer that generates sound in real time has a limitation on the number of notes it can synthesize at once. In the Java Sound API, the synthesizer reports this limitation through the `getMaxPolyphony` method.

A *voice* is a succession of single notes, such as a melody that a person can sing. Polyphony consists of multiple voices, such as the parts sung by a choir. A 32-voice synthesizer, for example, can play 32 notes simultaneously. (However, some MIDI literature uses the word "voice" in a different sense, similar to the meaning of "instrument" or "timbre.")

The process of assigning incoming MIDI notes to specific voices is known as *voice allocation*. A synthesizer maintains a list of voices, keeping track of which ones are active (meaning that they currently have notes sounding). When a note stops sounding, the voice becomes inactive, meaning that it's now free to accept the next note-on request that the synthesizer receives. An incoming stream of MIDI commands can easily request more simultaneous notes than the synthesizer is capable of generating. When all the synthesizer's voices are active, how should the next Note On request be handled? Synthesizers can implement different strategies: the most recently requested note can be ignored; or it can be played by discontinuing another note, such as the least recently started one.

Although the MIDI specification does not require it, a synthesizer can make public the contents of each of its voices. The Java Sound API includes a `VoiceStatus` class for this purpose.

A `VoiceStatus` reports on the voice's current active or inactive status, MIDI channel, bank and program number, MIDI note number, and MIDI volume.

With this background, let's examine the specifics of the Java Sound API for synthesis.

6.5.2- Permission to Play Synthesized Sound:

The audio produced by any installed MIDI synthesizer is typically routed through the sampled-audio system. If your program doesn't have permission to play audio, the synthesizer's sound won't be heard, and a security exception will be thrown.

6.6 -Providing Sampled-Audio Services:

the Java Sound API includes two packages, `javax.sound.sampled.spi` and `javax.sound.midi.spi`, that define abstract classes to be used by developers of sound services. By implementing and installing a subclass of one of these abstract classes, a service provider registers the new service, extending the functionality of the runtime system. This page tells you how to go about using the `javax.sound.sampled.spi` package to provide new services for handling sampled audio.

There are four abstract classes in the `javax.sound.sampled.spi` package, representing four different types of services that you can provide for the sampled-audio system:

- `AudioFileWriter` provides sound file-writing services. These services make it possible for an application program to write a stream of audio data to a file of a particular type.
- `AudioFileReader` provides file-reading services. These services enable an application program to ascertain a sound file's characteristics, and to obtain a stream from which the file's audio data can be read.
- `FormatConversionProvider` provides services for converting audio data formats. These services allow an application program to translate audio streams from one data format to another.
- `MixerProvider` provides management of a particular kind of mixer. This mechanism allows an application program to obtain information about, and access instances of, a given kind of mixer.

To recapitulate earlier discussions, service providers can extend the functionality of the runtime system. A typical SPI class has two types of methods: ones that respond to queries about the types of services available from a particular provider, and ones that either perform the new service directly, or return instances of objects that actually provide the service. The runtime environment's service-provider mechanism provides *registration* of installed services with the audio system, and *management* of the new service provider classes.

In essence there is a double isolation of the service instances from the application developer. An application program never directly creates instances of the service objects, such as mixers or format converters, that it needs for its audio processing tasks. Nor does the program even directly request these objects from the SPI classes that administer them. The application program makes requests to the `AudioSystem` object in the `javax.sound.sampled` package, and `AudioSystem` in turn uses the SPI objects to process these queries and service requests.

The existence of new audio services might be completely transparent to both the user and the application programmer. All application references are through standard objects of the `javax.sound.sampled` package, primarily `AudioSystem`, and the special handling that new services might be providing is often completely hidden.

6.7 – voice production:

Producing sound have several stages that make use of java capabilities that described earlier. We can abstract these stage for producing sound in six stages:

First stage: Read sound file from the data base of sounds.

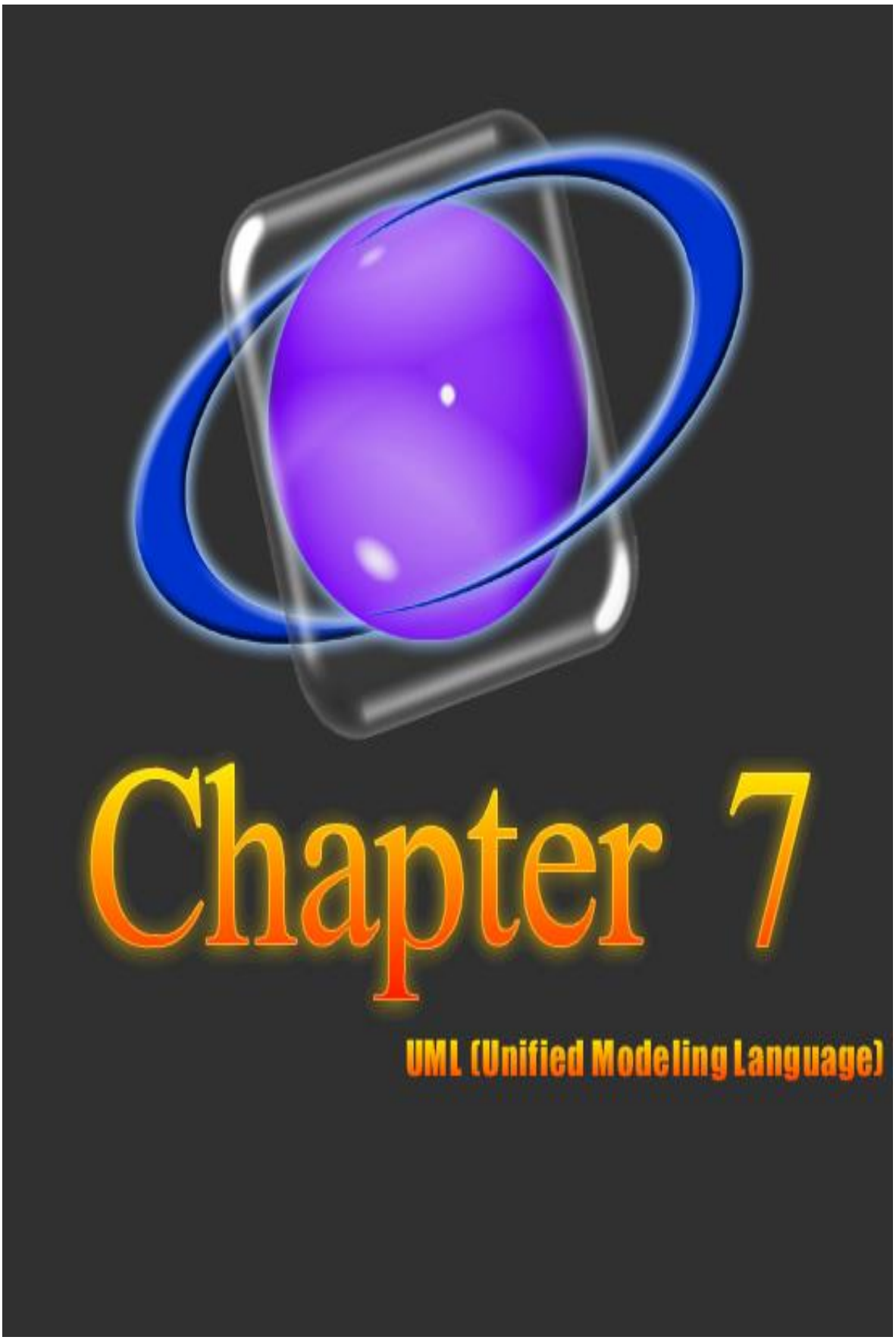
Second stage: Convert this sound into sampled data that converted to bytes(`AudioFileReader`).

Third stage: Transmit these bytes through transmission channel.

Fourth stage: Receive these bytes in the client mobile.

Fifth stage : Collect these bytes in a sound file again and play it to produce sound(`AudioFileWriter`).

Sixth stage: Synthesizing speech produced.


Chapter 7
UML (Unified Modeling Language)

Ch 7
UML
(Unified Modeling Language)

7.1. Overview

This chapter will discuss the design of the system and how different parts of the System work together. This Section shows the UML diagrams of the system.

7.2. Use Case Diagram

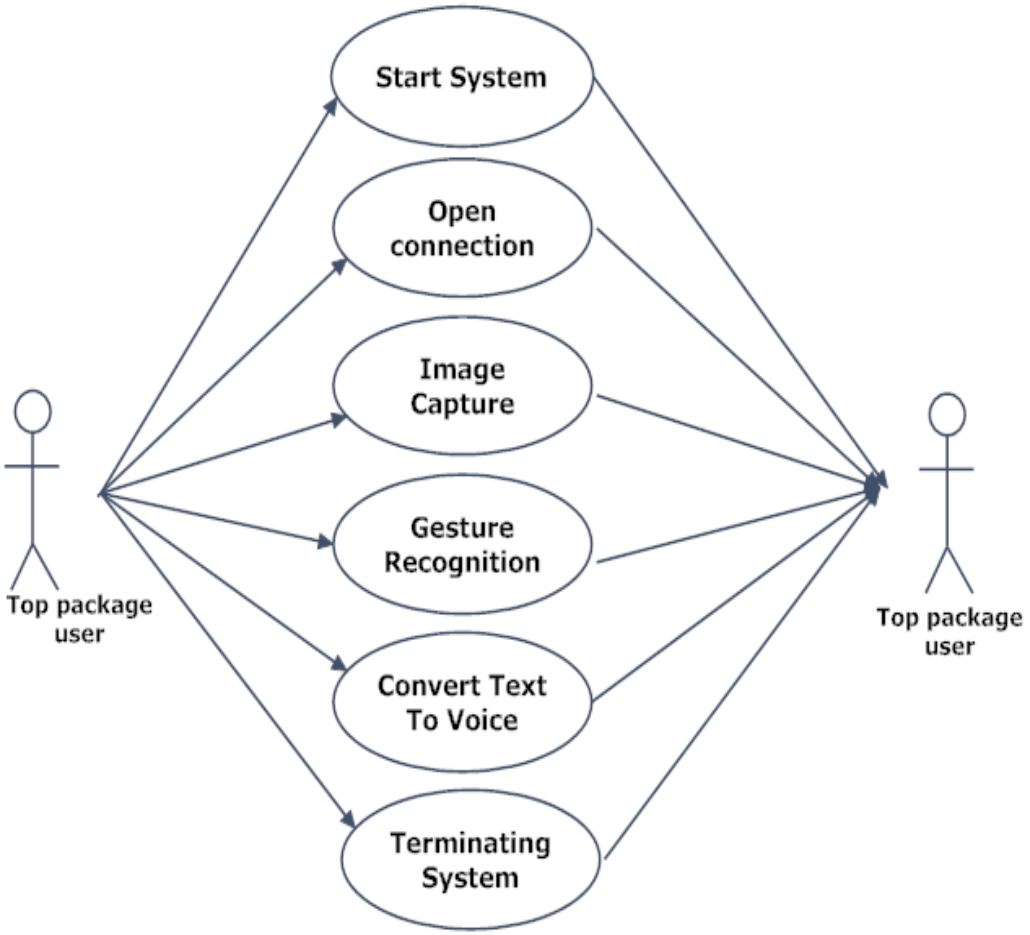


Figure (7.1) Use Case Diagram

7.3. Use Case Scenario

System:	Hand Gesture Recognition
User Case Name:	Open Connection
Actor:	User
Description:	Open UDP connection between mobile and server.
Course of actions in the system:	Step1: Open Wireless Connection between Mobile & Server. Step2:use Socket programming to open connection.

Table (7.1) Scenario

System:	Hand Gesture Recognition
User Case Name:	Image capture
Actor:	User
Description:	Capture hand gesture by mobile camera and send it to server.
Course of actions in the system:	Step1: Open Wireless Connection between Mobile & Server. Step2: Capture image by mobile camera. Step3: Send image to server

Table (7.2) Scenario

System	Hand Gesture Recognition
Actor	Perform a gesture
Description	The system user should perform any available gesture to be detected and recognized by the system.
Course of actions in the system	Step1: the user will stand in front of the camera and do sign. Step2: the system will detect this gesture. Step3: in each image the system will analyze the gesture and recognize its meaning. Step4: finally the system will release the meaning of user gesture in sound.

Table (7.3) Scenario

System :	Hand Gesture Recognition
User Case Name:	Convert Text to Voice
Actor:	User
Description:	Used to convert text to voice.

Table (7.4) Scenario

7.4. **Class Diagram**

7.5. Sequence diagram

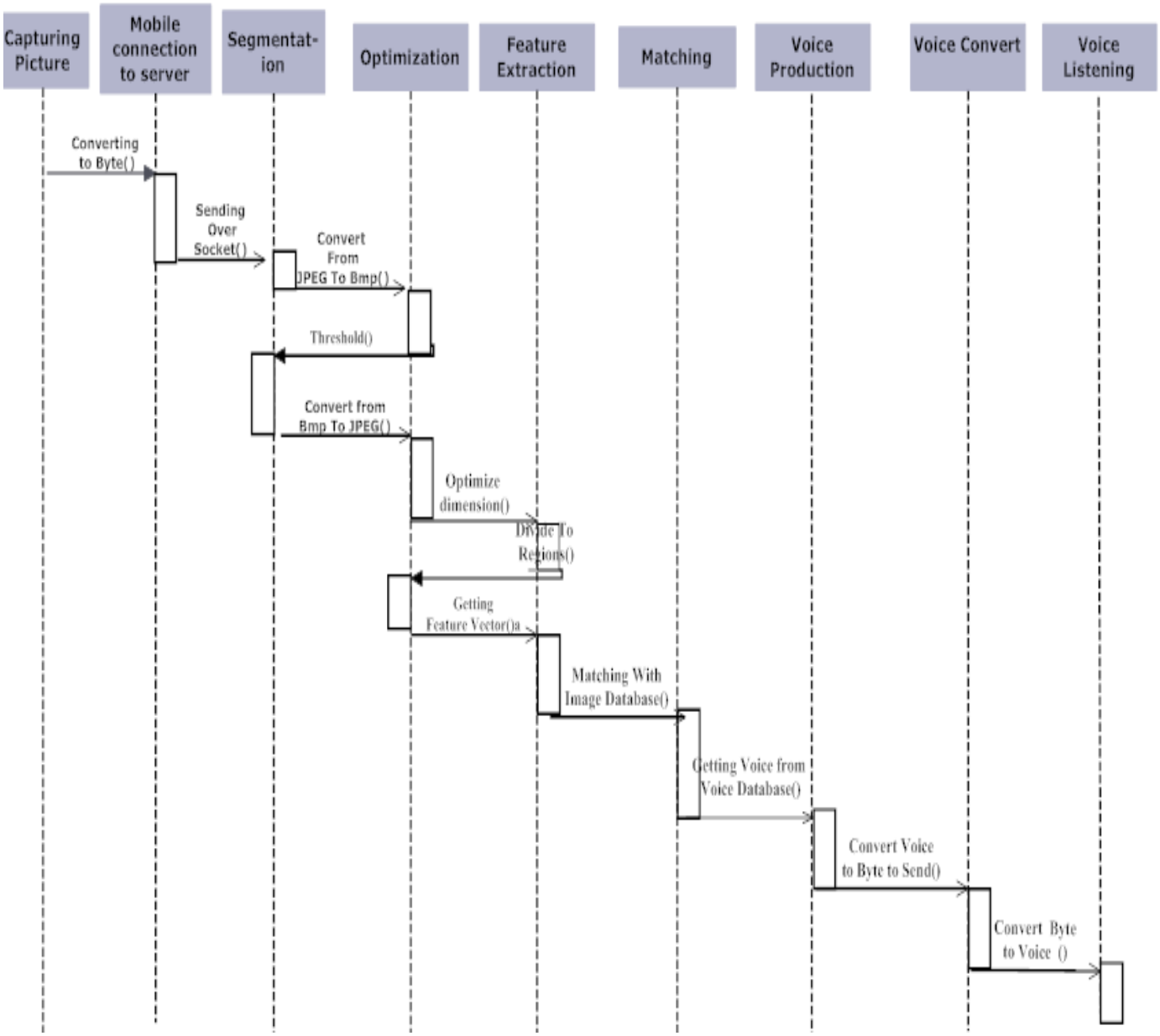


Figure (7.3) Sequence diagram

7.6. Activity Diagram

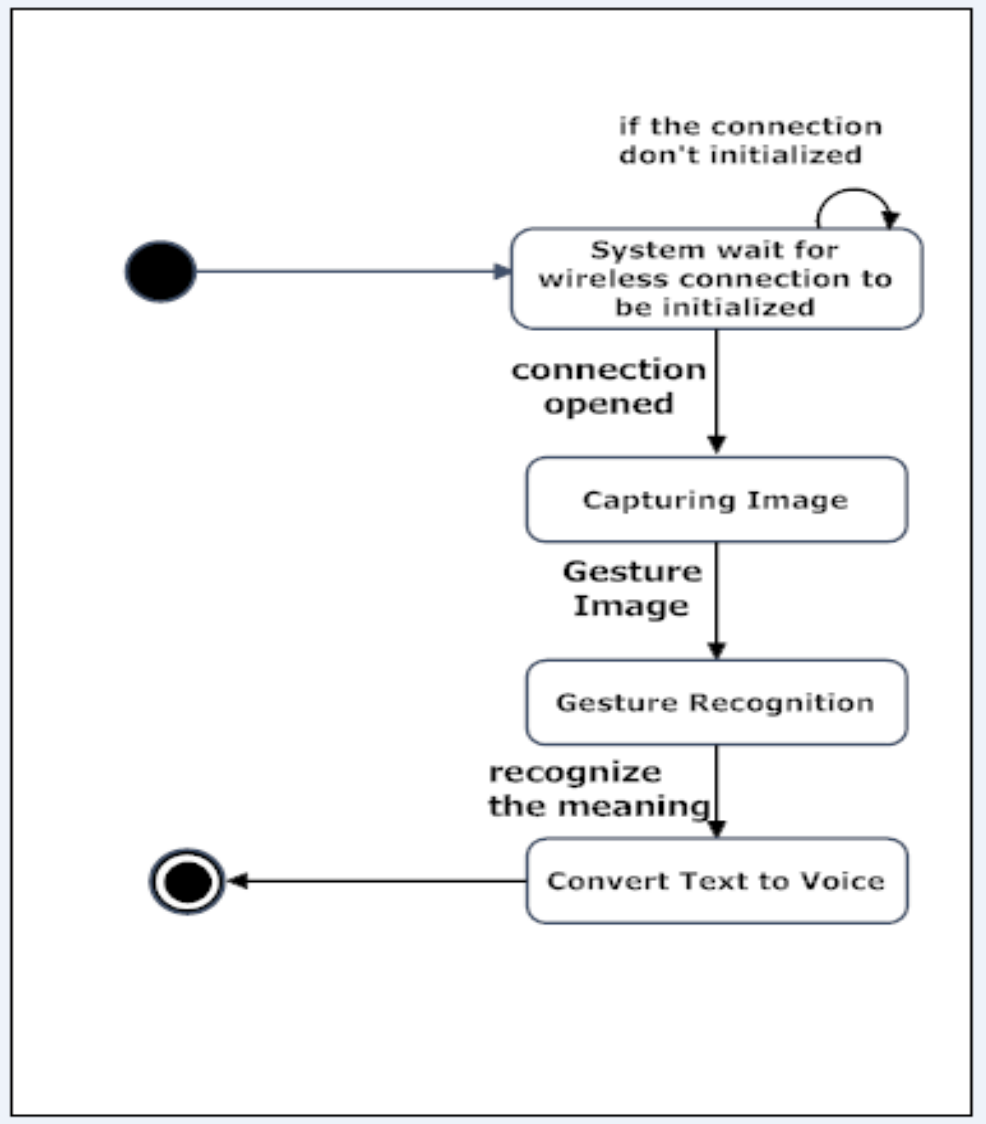


Figure (7.4) activity diagram.

Ch 8 Conclusion

Chapter "8"Conclusion….....

8.1- Conclusion

We presented the implementation of a system that aims to translate sign language gestures into speech. The system is based on an image processing approach where a digital camera is used to capture hand movements. After the hand is detected using algorithms employing Java Library, frames are periodically captured and processed. Threshold is first applied, then hand detect. After hand detection, the image is sent to be compared to a set of images in a reference database. The comparison is based **Euclidean distance**. In case of a match, the sound of the hand gesture. The Sign error massage will appear in case if was not found.

8.2- Limitations.

Our project has some limitations:

1-Mobile camera should be far from hand 30 cm.

2-Background must be Black.

3-Mobile must be has WirelessLan facility.

8.3- Future Work.

Complete software application that can be made available commercially to deaf mute that can recognize hand gesture with face.

The project now is working in samples of letters and words , our future plan will make the system work to translate sentences not only words and numbers by applying AI (Artificial intelligence) to understand the order of the words and return the sentence that is meant.

References

[1] Wikipedia. (2006). The free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Main_Page

[2] **http://en.wikipedia.org/wiki/Mobile_ad-hoc_network**

[3]http://compnetworking.about.com/cs/wireless/f/infrawireless.htm

[4] http://en.wikipedia.org/wiki/Internet_socket

[5]Apress.Pro.Java.ME.MMAPI.Mobile.Media.API.for.Java.Micro.Edition.May. 2006.pdf

[6] J2ME in a Nutshell.pdf

[7] Netbeans j2Me Tutorial.pdf

[8] Ohki, M. The Sign Language Telephone. 7th World Telecommunication Forum, Vol. 1, pp.3.91-395, 1995

[9] Shahzad Malik. 1st-Real-time Hand Tracking and Finger Tracking for Interaction, ***2003.***

[10] Francesca Gasparini, Raimondo Schettini .Skin segmentation using multiple thresholds DISCO (Dipartimento di Informatica, Sistemistica e Comunicazione) Universita degli studi di Milano-Bicocca, Via Bicocca degli Arcimbodli

[11] Lam Phung ET. ADAPTIVE SKIN SEGMENTATION IN COLOR IMAGES**.**

[12] Rasiwasia .Color Space for Skin Detection – A Review.

[13] Y. Y. S. Ju, M. Black, "Cardboard people: A parameterized model of articulated image motion," Proc. Int. Conf. on Automatic Face and Gesture Recognition, pp. 38–44, 1996

[14] D. M. Gavrila and L. S. Davis, "3-d model based tracking of humans in action: A multi- view approach," CVPR'96, pp. 73– 80, 1996.