

Project Report
On
**INTELLIGENT SOFTWARE FOR TEACHING VISUALLY
IMPAIRED STUDENTS**

Submitted in partial fulfilment of the requirement
for the award of the Degree of
Bachelor of Engineering
in
Information Technology
By

Laavanya Ganesh
Vishakha Jagtap
Kunal Javeri

under the guidance of
Prof. Rupali Sawant



Department of Information Technology
Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Munshi Nagar, Andheri-West, Mumbai-400058
University of Mumbai
April 2014

Project approval Certificate

This is to certify that the Project entitled "INTELLIGENT SOFTWARE FOR TEACHING VISUALLY IMPAIRED STUDENTS" has been completed successfully by Ms. Laavanya Ganesh, Ms. Vishakha Jagtap and Mr. Kunal Javeri is approved for the award of Degree of Bachelor of Engineering in Information Technology from University of Mumbai.

External Examiner

Internal Examiner

(signature)

Name:

Date:

(signature)

Name:

Date:

Seal of the Institute

Contents

Description	Page No.
Abstract.....	1
Abbreviations.....	2
1.0 Introduction.....	3
1.1 Existing System	3
1.2 Problem Definition	3
1.3 Proposed System.....	4
1.4 Project Scope	4
1.5 Assumptions and Constraints.....	5
1.6 System Requirements	5
 2.0 Review of Literature.....	 6
 3.0 Methodology	 8
3.1 Algorithm	8
3.1.1 Boundary Extraction	9
3.1.2 Signature analysis.....	9
3.1.2.1 Distance calculation	10
3.1.2.2 Screening based on similarity.....	10
3.1.3 Color	11
3.1.3.1 Object extraction	12
3.1.3.2 RGB to HSV conversion.....	13
3.1.3.3 Distance calculation.....	13

3.1.3.4 Screening based on similarity	14
3.1.4 Text to speech conversion	14
4.0 Analysis.....	20
4.1.1 Use Case Diagram and Documentation	20
4.1.2 ActivityDiagram	23
4.1.3 Sequence Diagram.....	24
4.1.4 CollaborationDiagram	25
4.1.5 Class Diagram.....	26
4.1.6 Component Diagram	27
4.1.7 Deployment Diagram.....	28
4.1.8 System Flowchart	29
4.2 System design	30
4.3 Implementation	36
5.0 Results and discussions	37
5.1 sample code for key scenarios	37
5.2 Code metrics	47
5.3 Test case	49
5.4 Result snapshots.....	50
6.0 Conclusion and future work.....	54
6.1 Conclusion	54
6.2 Future Work	54
Appendix- Algorithm	55
References	56
Acknowledgement.....	57

List of Figures

Figure 3.1.3:Flowchart for color retrieval	12
Figure 4.1.1: Use Case Diagram.....	22
Figure 4.1.2: Activity Diagram.....	23
Figure 4.1.3: Sequence Diagram.....	24
Figure 4.1.4: Collaboration Diagram.....	25
Figure 4.1.5: Class Diagram.....	26
Figure 4.1.6: Component Diagram	27
Figure 4.1.7: Deployment Diagram.....	28
Figure 4.1.8: Block Diagram.....	29
Figure 4.2.1 Architecture Diagram.....	30
Figure 4.2.2 Image Management Module.....	31
Figure 4.2.3:User location.....	32
Figure 4.2.4:Image mapping.....	33
Figure 4.2.5:Data mining.....	34
Figure 4.2.6:Audio interface.....	35
Figure 5.4: CBIR.....	50

Abstract

Today, computers are integrated parts of children's play activities as well as school education. However, computers are often inaccessible to children with visual impairments and therefore their learning process is affected due to lack of equipments and facilities.

Nowadays, many softwares have come up with the upgrading technology to teach the visually impaired. Our project basically aims at allowing the visually impaired children of age group 2-5 yrs to adapt to softwares equipped with rich audio resources, corresponding to hours of recorded messages, music and various sound effects, with coherent educational contents.

Suppose the visually impaired children are to be taught the concept of Physics or Maths, there shall be existing games in the software designed to teach them the concepts through a fun-learning experience. This project deals with object detection and image processing to collaborate educational content with less intensive teaching and easy-go learning. This project aims at delivering retrieved information about the image captured by the PC with the help of content based image proceeding (CBIR). So this project aims at teaching the blind in a fun-learning way rather than in a procedural conceptual way.

Abbreviations

CBIR: Content Based Image Retrieval

OCR: Optical Character Recognition

TIM: Tactile Interactive Multimedia

TBB: Tactile Books and Braille

PC: Personal Computer

GUI: Graphical user Interface

UML: Unified Modified Language

Chapter 1

Introduction:

1.1: Existing System

In today's world , there are systems for the visually impaired like which use tactic language to drive CD-ROMs and audio resources. There is a Tactile Interactive Multimedia(TIM) software which allows the blind children to adapt to computer games.

Most blind computer users have a screen reader combined with synthetic speech and/or braille display. Haptic interfaces use the sense of touch in the user interaction with the haptic interface it is thus possible to feel shapes that are based on digital information. There are now computer programs available that some of the graphical information in a GUI via a haptic device.[1]

1.2 Problem Definition

The present softwares for the visually impaired give access to only the text on screen due to the Braille display, but not the graphics. Also the mainstream computers that are available today are often inaccessible to children with visual impairments. There their learning process is affected due to lack of equipments and facilities. In addition to this, the current systems do not take into consideration the mental ability of the visually impaired children of different age groups. These children are required to be taught in a fun-learning way understanding how much they can grasp and how well they can analyse situations. The fun-learning X-factor is what that is lacking in the current system. This is the drawback existing in the current scenario in the world of visually impaired.[2]

1.3 Proposed System:

Most of the existing systems for the blind children are designed with a purpose to conceive computer games and educational applications in an autonomous way with an assistance of a sighted person. To add a fun learning element to the existing system we need to design a system that includes games for different age groups and education for all the different age groups can be incorporated through the games. Through this attempt the visually impaired children can grasp things taught to them quickly. There can also be in the motion sensor in the webcam of the computer to detect the motion of the visually impaired children and direct them where they are going wrong. There will be a normal braille instrument which will convey what the children want to learn. Understanding the gestures they will be directed through an audio interface. [2]

Interactive learning for the blind children can be achieved through designing games in which the blind child can navigate and listen to the different sound environments along with the explanations through which the child can understand the topics well.

1.4 Scope

The project basically aims at allowing the visually impaired children of age-groups 2-5 years to adapt to softwares equipped with rich audio resources corresponding to recorded information with coherent educational content. The project mainly includes OBJECT DETECTION AND IMAGE PROCESSING based on the CBIR and OCR. There will be an installed webcam attached to the PC which will capture the image kept in front of it and give information about it through an audio interface. Also it can read out captured text.

1.5 Assumptions and Constraints

1.5.1 Assumptions:

- The visually blind know the intricacies of operating the web camera, audio interface and the PC
- Most of the blind children targeted can listen

1.5.2 Constraints:

- The object should be placed at 30-40 cm from the web camera at an appropriate small uphill
- Targeted age group is only 2-5 years for the project
- Text image should only be in capital letters and the text image should be held straight.
- OCR will read only the uppercase alphabets
- Image to be captured has to have a black background

1.6 System Requirements

- Hardware Interfaces
 - PC
 - Croma Full HD webcam
 - JBL speakers(audio interface)
 - JBL headset(for individual learning)
- Software Interfaces
 - Matlab and its inbuilt database

Chapter 2

Review of Literature

2.1

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.5291&rep=rep1&type=pdf>

Paper on “Using haptics in computer interfaces for blind people”

This paper talks about on how to find a way to use haptic technology to provide new computer interaction techniques for visually impaired and those with physical disabilities. It also throws light on finding objects, understanding objects, haptic widgets and physical interaction.

2.2 <http://ui4all.ics.forth.gr/i3SD2000/Archambault.PDF>

Paper on “development and adaption of computer games for young blind children”

This paper concerns of a development of computer games using a tactile equipment connected to the PC. It deals with the designing of a scripting language used to drive the tactile equipment. The scripting language covers a wide variety of software that help the blind in adapting to existing CD-ROMs.

2.3 . http://www.icdvrat.rdg.ac.uk/2004/papers/S03_N1_Eriksson_ICDVRAT2004.pdf

Paper on “ computer games for children with visual impairments”

It talks about two types of games: Image based games for children with partial eye-sight and sound based games so that games can be accessible without relying on vision.

2.4.

<http://www.vipconduit.com/linkurls.php?lnkcat=Computer%20Games%20For%20Blind%20And%20Low%20Vision%20Users>

This website provides a wide variety of software audio games for the visually impaired.

2.5 <http://www.wonderbaby.org/articles/best-accessible-computer-games-blind-kids>

This website provides a list of computer games for young toddlers like Balleyland, Priory Woods, PCS Games, GMA Games and many more.

2.6 <http://inova.snv.jussieu.fr/tim/publis/aaate01/aaate01-arch.html>

This paper talks about the TIM software. TIM is a project whose main objective is to offer to visually impaired children of various levels of psychomotor development the possibility to play

computer games in an autonomous way. TIM proposes to develop an adapting tool allowing to design high quality computer games using a tactile and audio interface from existing contents. TIM includes high level research on cognitive psychology and education sciences in order to ensure a high level of quality allowing blind children in early youth to use a computer, like sighted children. The software gives to the computer a double role: ludic and educational. For some children, having additional disabilities, like cognitive troubles, it can have a third role: a therapeutic tool.

2.7 http://support.perkins.org/site/PageServer?pagename=store_homepage

This website provides the perkins products available for the visually impaired children ,their features ,their prices and where to locate them.

Chapter 3

Theory

The basic idea of the project is to impart a fun element in the learning process of the visually impaired children. The project is considering the children in the age group of 2-5 yrs of age.

The flow of the project can be explained as follows,

Firstly, the child who desires to learn will place the object in front of the web cam of the PC. Then, this web cam will capture the image of the object in his hand. This will be done through the Content based image retrieval (CBIR).

Instead of exact matching of the captured image, content-based image retrieval calculates visual similarities between a query image and images in a database. Accordingly, the retrieval result is not a single image but a list of images ranked by their similarities with the query image. The most ranked image will be chosen. Then the description related to this image, that is available in the form of text will be converted into corresponding speech and this speech is given out to the blind child who is aspiring to learn.

The child can also keep a text image in front of the web camera. The web camera will capture the text image and read it out in the form of speech through the audio interface

3.1 Algorithm

3.1.1 Boundary Extraction:

- First convert images (database and the input) to binary using appropriate threshold. This is done so that the object and the background are clearly distinguished using the following function:

level = graythresh(r)

bw = im2bw(r, level);

- We then obtain the sequence of boundary coordinates for each pixel using the function

b=bwboundaries (bw);

- We require generally the predominant image so from obtained boundaries of a picture image , the maximum boundary is obtained by

d=cellfun ('length', b);

[maxd,k]=max(d);

b=b{k};

3.1.2 Signature analysis:

- Now the signature analysis is carried out for the set of dominant boundary coordinate sequences obtained above of both the input and the database images (i.e, the signatures are calculated for each of them .So a code for Signature analysis function is written and the function is used as follows:

[st,angle,x0,y0]=signature (b);

- Signature is calculated by obtaining the centroid of each set of boundary coordinates and then the distances from the centroid to certain boundary pixels are calculated for each for the same angles.

3.1.2.1 Distance calculation:

- The signature vector of each database image is then compared with that of the query image using the Euclidean distance metric.
- Euclidean distance metric involves finding absolute distance between the signatures and then root mean square of those values is found.

Eg:

$$d\{1\}=\sqrt{\sum(\sum(abs(st\{1\}-st_input).^2))}$$

- The distances are then sorted in increasing order so that the image farthest from the query comes last.

3.1.2.2 Screening Based on Similarity:

- To match the images we assign a threshold value. The threshold value is calculated as mean of maximum and minimum distances as:

Eg.:

$$dst=(ds(1)+ds(l))/2;$$

- An index is provided for the threshold so that threshold can be adjusted as per requirement.
- If the distances are less than or equal to the threshold value, the images fall under matching set (**database2**) ,the rest of the images are segregated as mismatches.

3.1.3 Colour:

Retrieving images based on color similarity is achieved by computing a comparison of the equivalent HSV model of each image with that of query image .In image analysis color of the objects like biscuits; fruits etc. are usually represented as RGB components. This requires processing 3 separate frames by the vision system. For efficient color processing RGB information is converted to HSV. HSV model is closer to human perception. With image representation in the HSV domain, the color analysis is based on primarily the hue value. The 3-D RGB space is reduced to 2-D HS-space. The database screened by the above shape matching (the biscuit images qualifying as good enough based on shape) are the subject to color screening and thereby matches are displayed based on required threshold.



3.1.3.1 Flowchart (Screening the image database based on their color):

Database image Query image

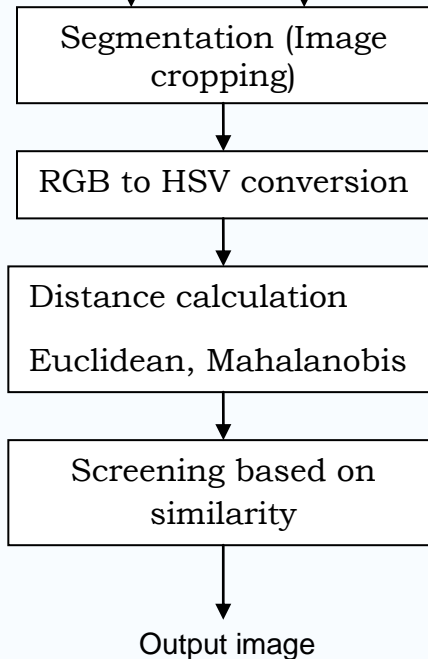


Figure 3.1.3 : flowchart for color retrieval

3.1.3.1 Object Extraction (*segmentation*):

- All the images in the database2 are cropped at the object location using the centroid of the objects such that the obtained images are equal in size (to that of the smallest object in the database2) while keeping the centroid of each object intact (i.e. as the centroid of the cropped image). The same is repeated for query image as well.

recta(i,:)= $[x0u(i)-dix/2 \ y0u(i)-diy/2 \ dix \ diy]$; %x0u,y0u : centroid

crop{i} = imcrop(u{i},recta(i,:));

3.1.3.2 RGB to HSVconversion:

- The cropped database2 images are then converted from RGB model to HSI form using the following functions and operations.

$hsv\{i\}=rgb2hsv(crop\{i\});$

$h\{i\}=hsv\{i\}(:, :, 1);$

$s\{i\}=hsv\{i\}(:, :, 2);$

$hs\{i\}=sqrt((h\{i\}.*h\{i\})+(s\{i\}.*s\{i\}));$

- The same is repeated for the query image.

3.1.3.3 Distance calculation:

- The cropped HSV images thus obtained are compared with the cropped query image (HSV model) using pixel to pixel comparison for the cropped part.
- Comparison is done using either Euclidean distance metric or the mahalanobis distance metric as per our choice. In this project both the comparisons are used to figure out the best distance metric.
- Euclidean distance metric involves absolute distance matrix between the database and the query and then root mean square vale of all the elements in the matrix.

$d(i)=sqrt(sum(sum(abs(hs\{i\}-hs_inputcrop).^2)));$

- Mahalanobis distance metric involves obtaining the mean vector of every row of query(HSV cropped) and repeating it to the whole row, subtracting each database image(HSV cropped) matrix with this(**Yc**), obtaining the covariance matrices(**Cx**) for the query matrix ,obtaining the conjugate for all the difference

matrices and then distance between a database image and the query is obtained by:

$$d=\text{real}(\text{sum}(Yc/Cx.*\text{conj}(Yc),2));$$

- The distances obtained thus are then sorted in ascending order of their magnitudes so as to order the images based on their closeness to the query.

3.1.3.4 Screening Based on Similarity:

- A threshold is then set that divides the database2 into set of close matches to query and otherwise. The threshold value is calculated as mean of maximum and minimum distances as:

$$\text{Eg.: } dst=(ds(1)+ds(l))/2;$$

- An index is provided for the threshold so that threshold can be adjusted as per requirement.
- If the distances are less than or equal to the threshold value, the images fall under matching set (**output database**), the rest of the images are segregated as mismatches. The matching set is the required result.

3.1.4 Text to speech conversion:

- Text-to-speech fundamentally functions as a pipeline that converts text into PCM digital audio. The elements of the pipeline are:
 - 1) Text normalization
 - 2) Homograph disambiguation
 - 3) Prosody
 - 4) Play audio
 - 5) Generating voice

3.1.4.1 Text Normalization

- Text normalization isolates words in the text. For the most part this is as trivial as looking for a sequence of alphabetic characters, allowing for an occasional apostrophe and hyphen.
- Text normalization then searches for numbers, times, dates, and other symbolic representations. These are analyzed and converted to words. (Example: "\$54.32" is converted to "fifty four dollars and thirty two cents.") It is essential to code up the rules for the conversion of these symbols into words, since they differ depending upon the language and context.
- Whatever remains is punctuation. The normalizer will have rules dictating if the punctuation causes a word to be spoken or if it is silent. Once the text has been normalized and simplified into a series of words, it is passed onto the next module, homograph disambiguation.

3.1.4.2 Homograph Disambiguation

- The next stage of text-to-speech is called "homograph disambiguation." Often it's not a stage by itself, but is combined into the text normalization or pronunciation components.
- In English and many other languages, there are hundreds of words that have the same text, but different pronunciations. A "homograph" is a word with the same text as another word, but with a different pronunciation. The concept extends beyond just words, and into abbreviations and numbers.
- Text-to-speech engines figure out the meaning of the text, and more specifically of the sentence, by parsing the sentence and figuring out the part-of-speech for the individual word. This is done by guessing the part-of-speech based on the word endings, or by looking the word up in a lexicon.
- The pronunciation module accepts the text, and outputs a sequence of phonemes, just like it is in a dictionary.

- To get the pronunciation of a word, the text-to-speech engine first looks the word up in its own pronunciation lexicon. If the word is not in the lexicon then the engine reverts to "letter to sound" rules.
- Letter-to-sound rules guess the pronunciation of a word from the text.
- The below stated is the rule for guessing the pronunciation:
- An algorithm is used to segment the word and figure out which letter "produces" which sound. It is clearly seen that "h" in "hello" produces the "h" phoneme, the "e" produces the "eh" phoneme, the first "l" produces the "l" phoneme, the second "l" nothing, and "o" produces the "oe" phoneme. In other words the individual letters produce different phonemes. The "e" in "he" will produce the "ee" phoneme.
- Once the words are segmented by phoneme, another algorithm determines which letter or sequence of letters is likely to produce which phonemes. The first pass figures out the most likely phoneme generated by each letter. "H" almost always generates the "h" sound, while "o" almost always generates the "ow" sound. A secondary list is generated, showing exceptions to the previous rule given the context of the surrounding letters. Hence, an exception rule might specify that an "o" occurring at the end of the word and preceded by an "l" produces an "oe" sound. The list of exceptions can be extended to include even more surrounding characters.
- When the letter-to-sound rules are asked to produce the pronunciation of a word they do the inverse of the training model. To pronounce "hello", the letter-to-sound rules first try to figure out the sound of the "h" phoneme. It looks through the exception table for an "h" beginning the word followed by "e"; Since it can't find one it uses the default sound for "h", which is "h". Next, it looks in the exceptions for how an "e" surrounded by "h" and "l" is pronounced, finding "eh". The rest of the characters are handled in the same way.
- This technique can pronounce any word, even if it wasn't in the training set, and does a very reasonable guess of the pronunciation, sometimes better than

humans. It doesn't work too well for names because most names are not of English origin, and use different pronunciation rules.

3.1.4.3 Prosody

- Prosody is the pitch, speed, and volume that syllables, words, phrases, and sentences are spoken with. Without prosody text-to-speech sounds very robotic, and with bad prosody text-to-speech sounds unclear.
- The technique that engines use to synthesize prosody varies, but there are some general techniques.
- First, the engine identifies the beginning and ending of sentences. In English, the pitch will tend to fall near the end of a statement, and rise for a question. Likewise, volume and speaking speed ramp up when the text-to-speech first starts talking, and fall off on the last word when it stops. Pauses are placed between sentences.
- Engines also identify phrase boundaries, such as noun phrases and verb phrases. These will have similar characteristics to sentences, but will be less pronounced. The engine can determine the phrase boundaries by using the part-of-speech information generated during the homograph disambiguation. Pauses are placed between phrases or where commas occur.
- Algorithms then try to determine which words in the sentence are important to the meaning, and these are emphasized. Emphasized words are louder, longer, and will have more pitch variation. Words that are unimportant, such as those used to make the sentence grammatically correct, are de-emphasized.
- Next, the prosody within a word is determined. Usually the pitch and volume rise on stressed syllables.
- All of the pitch, timing, and volume information from the sentence level, phrase level, and word level are combined together to produce the final output. The output from the prosody module is just a list of phonemes with the pitch, duration, and volume for each phoneme.[3]

3.1.4.4 Play Audio

- The speech synthesis is almost done by this point. All the text-to-speech engine has to do is convert the list of phonemes and their duration, pitch, and volume, into digital audio.
- Methods for generating the digital audio will vary, but many text-to-speech engines generate the audio by concatenating short recordings of phonemes. The recordings come from a real person. In a simplistic form, the engine receives the phoneme to speak, loads the digital audio from a database, does some pitch, time, and volume changes, and sends it out to the sound card.
- Most noticeable is that one recording of a phoneme won't have the same volume, pitch, and sound quality at the end, as the beginning of the next phoneme. This causes a noticeable glitch in the audio. An engine can reduce the glitch by blending the edges of the two segments together so at their intersections they both have the same pitch and volume. Blending the sound quality, which is determined by the harmonics generated by the voice, is more difficult, and can be solved by the next step.
- The sound that a person makes when he/she speaks a phoneme, changes depending upon the surrounding phonemes.
- If you record "cat" in sound recorder, and then reverse it, the reversed audio doesn't sound like "tak", which has the reversed phonemes of cat. Rather than using one recording per phoneme (about 50), the text-to-speech engine maintains thousands of recordings (usually 1000-5000). Ideally it would have all possible phoneme context combinations recorded, $50 * 50 * 50 = 125,000$, but this would be too many. Since many of these combinations sound similar, one recording is used to represent the phoneme within several different contexts.
- Even a database of 1000 phoneme recordings is too large, so the digital audio is compressed into a much smaller size, usually between 8:1 and 32:1 compression. The more compressed the digital audio, the more muted the voice sounds.
- Once the digital audio segments have been concatenated they're sent off to the sound card, making the computer talk.

3.1.4.5 Generating a Voice

- The first step is to select a voice talent. The voice talent then spends several hours in a recording studio reading a wide variety of text. The text is designed so that as many phonemes sequence combinations are recorded as possible. You at least want them to read enough text so there are several occurrences of each of the 1000 to 5000 recording slots.
- After the recording session is finished, the recordings are sent to a speech recognizer which then determines where the phonemes begin and end. Since the tools also know the surrounding phonemes, it's easy to pull out the right recordings from the speech. The only trick is to figure out which recording sounds best. Usually an algorithm makes a guess, but someone must listen to the phoneme recordings just to make sure they're good.
- The selected phoneme recordings are compressed and stored away in the database. The result is a new voice.[4]

Chapter 4

Analysis

UML stands for Unified Modeling Language. It is a third generation method for specifying, visualizing and documenting the artifacts of an object oriented system under development. Object modeling is the process by which the logical objects in the real world (problem space) are represented (mapped) by the actual objects in the program (logical or a mini world). This visual representation of the objects, their relationships and their structures is for the ease of understanding. This is a step while developing any product after analysis.

4.1.1 Use Case Diagram and documentation

Use case diagram consists of use cases and actors and shows the interaction between them. The key points are:

- The main purpose is to show the interaction between the use cases and the actor.
- To represent the system requirement from user's perspective.
- The use cases are the functions that are to be performed in the module.
- An actor could be the end-user of the system or an external system.

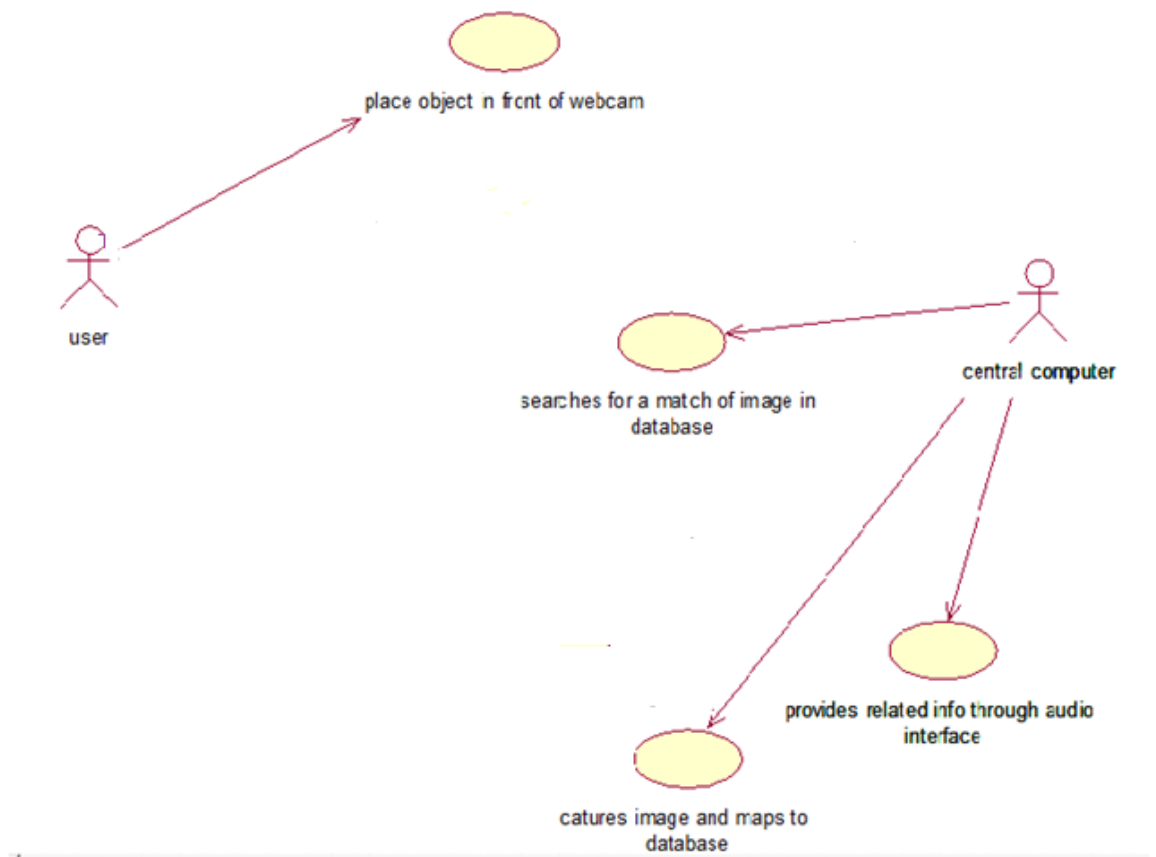


Figure 4.1.1

UseCaseId : 1

Name : Search

Actor : central computer

Goals : To retrieve information regarding the captured image

Precondition : Webcamera should have captured appropriate image

Computer Action	System Response
Webcamera captures image in front of it	System extracts answers from the temporary database or web after finding a map of the template.

Post condition : Extracted information is systematically arranged in order of appropriateness

UseCaseId : 2

Name : place

Actor : User

Goals : To conceive information about a particular object

Precondition : The object should be available with the user

User Action	System Response
Places desired object in front of webcam	System captures the image and maps it to templates in temporary database or web.

UseCaseId : 3

Name : speech

Actor : Admin

Goals : To convert retrieved information to audio and provide it through an audio interface to user

Computer Action	System Response
Mainframe converts extracted information to speech with the help of suited algorithms	The generated audio is provided to user through an audio interface

4.1.2 Activity Diagram

Activity diagrams, which are related to program flow plans (flowcharts), are used to illustrate activities. In the external view, we use activity diagrams for the description of those business processes that describe the functionality of the business system.

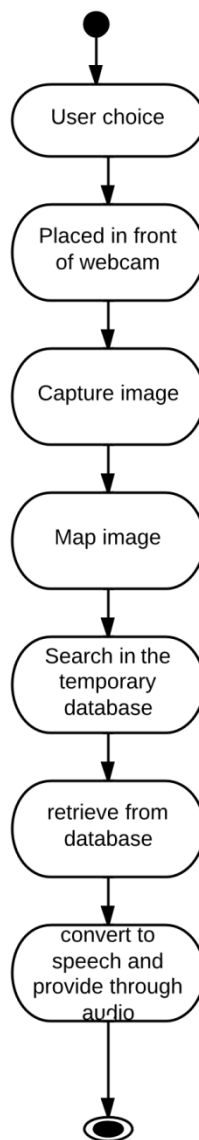


Figure 4.1.2

4.1.3 Sequence Diagram

The purpose of sequence diagram is to show the flow of functionality through a use case. In other words, we call it a mapping process in terms of data transfers from the actor through the corresponding objects.

The key points are:

- The main purpose is to represent the logical flow of data with respect to a process
- A sequence diagram displays the objects and not the classes.

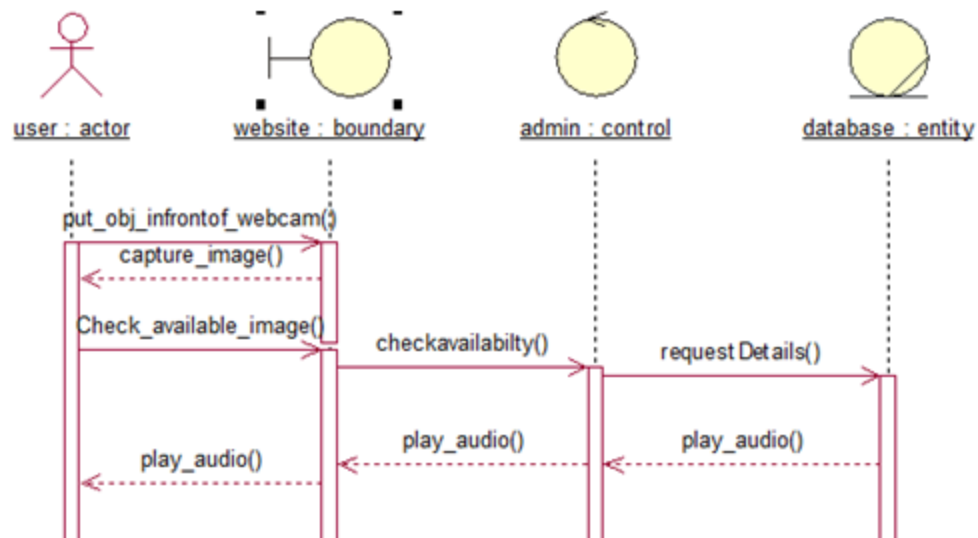


Figure 4.1.3

4.1.4 Collaboration Diagram

In UML, collaboration diagrams describe interactions among objects in terms of sequenced messages. They represent a combination of information taken from class, sequence and use-case diagrams describing both the static and dynamic behaviour of the system

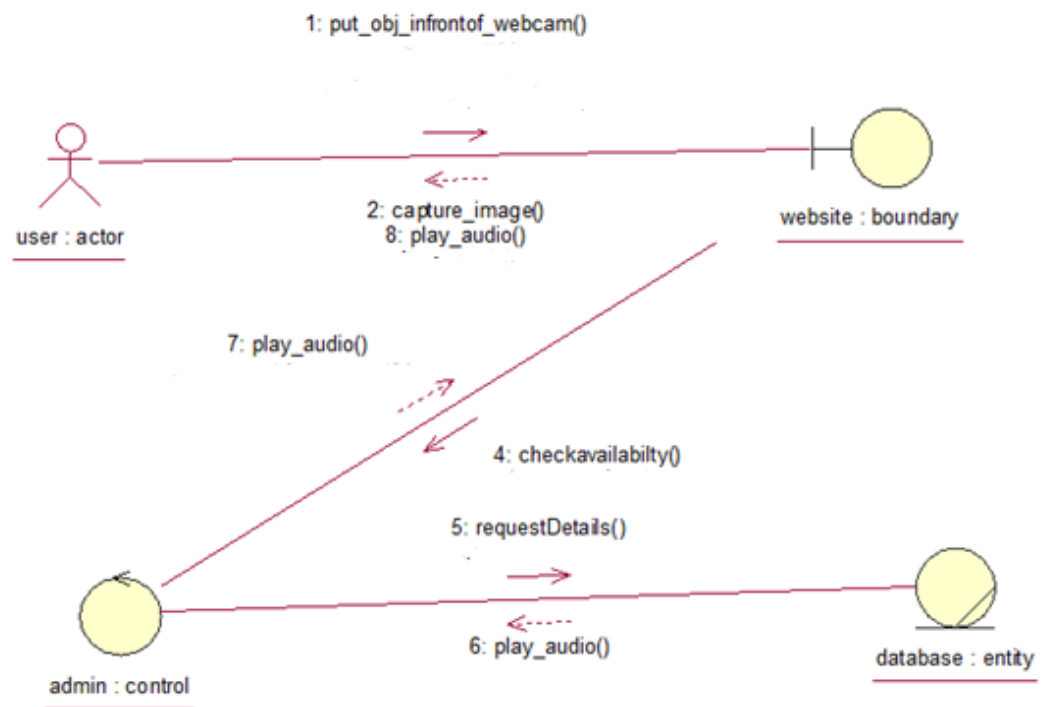


Figure 4.1.4

4.1.5 Class Diagram

Class Diagram consists of the classes and the objects and the interaction between them. It mainly deals with the interaction between classes in the system, their behavior and properties of the system. Apart from classes this also provides inheritance relationships in the project. Class diagrams consist of basically two parts: first one is the member variables and class variables and the second part consists of the total number of methods available in the class.

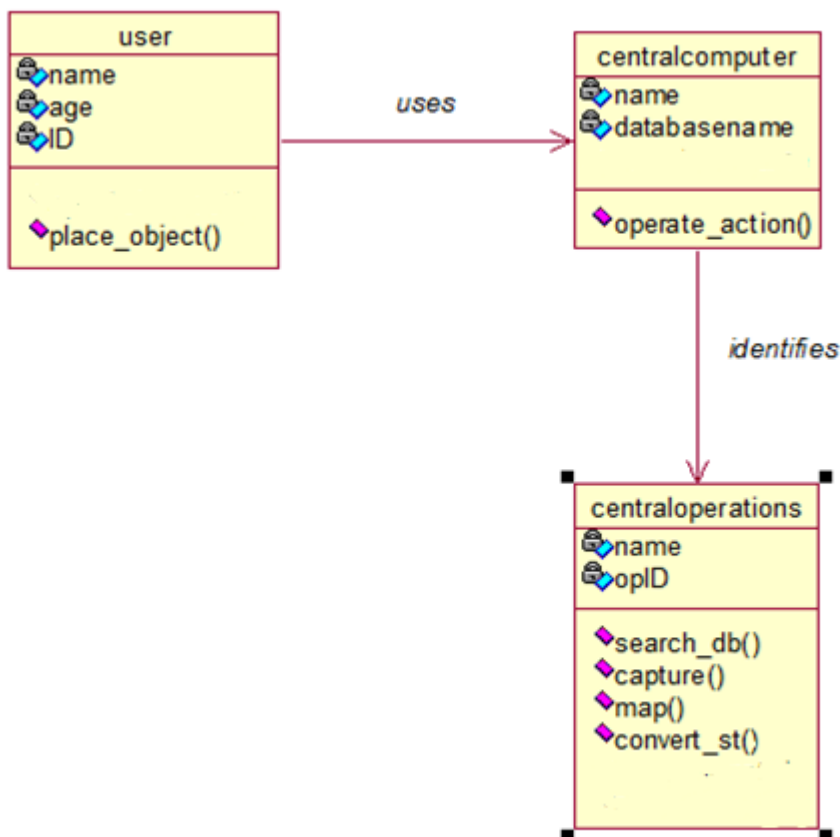


Figure 4.1.5

4.1.6 Component Diagram

Describes how a software system is split up into components and shows the dependencies among these components. This may have a visual stereotype in the top right of the rectangle of a small rectangle with two even smaller rectangles jutting out on the left

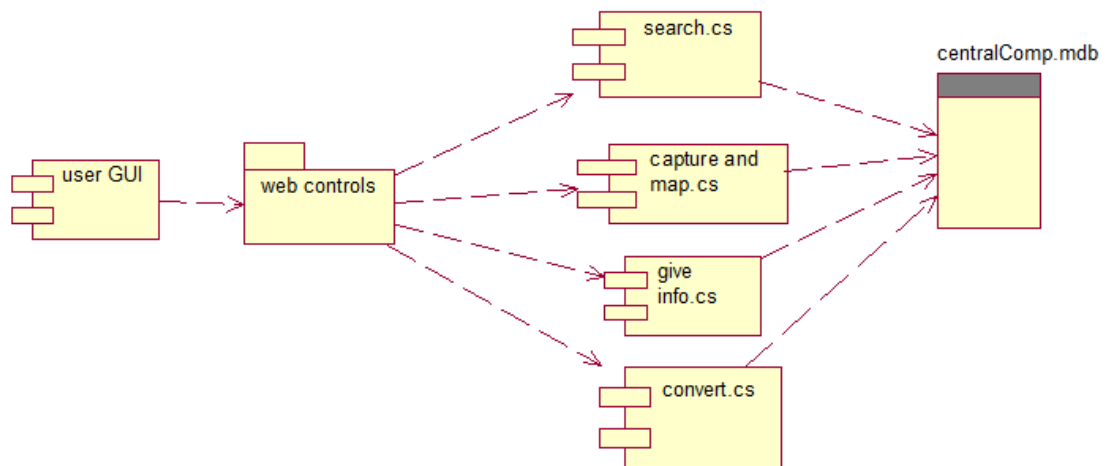


Figure 4.1.6

4.1.7 Deployment Diagram

In UML, deployment diagrams model the physical architecture of the system. They show the relationships between the software and hardware of the system and the physical distribution of the processing. They are used for visualizing, specifying and documenting following systems: embedded systems, client /server systems and distributed systems.

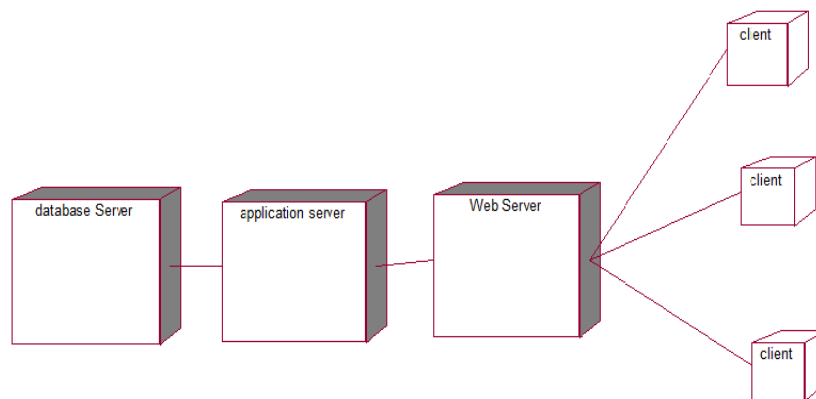


Figure 4.1.7

4.1.8 System flowchart

Block Diagram:

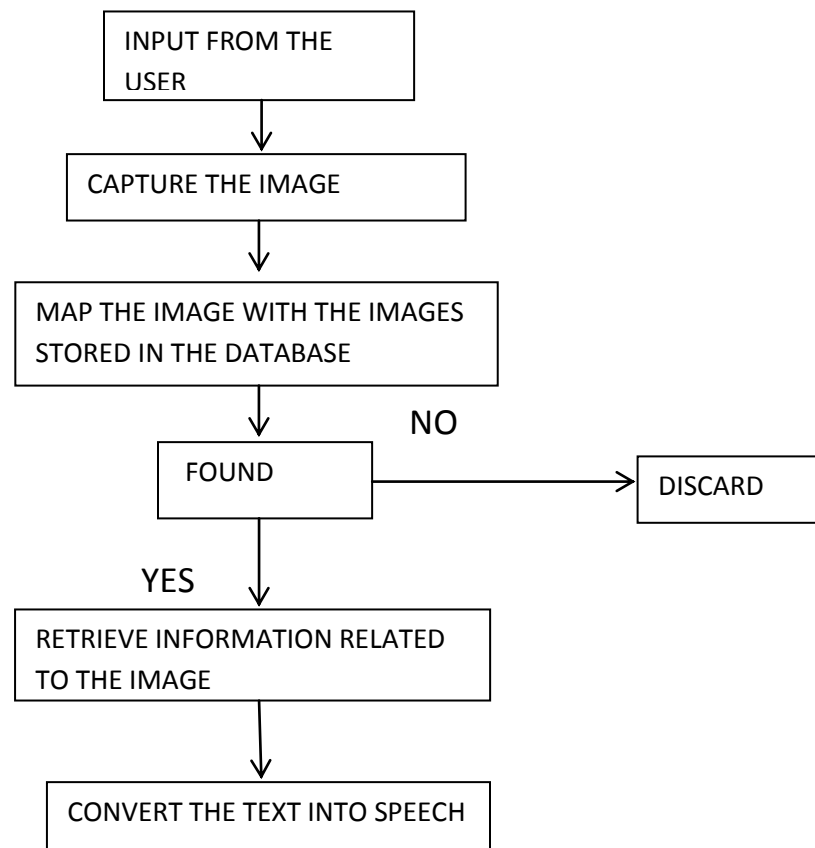


Figure 4.1.8

4.2 System Design

4.2.1 Architectural Design

Architecture Diagram

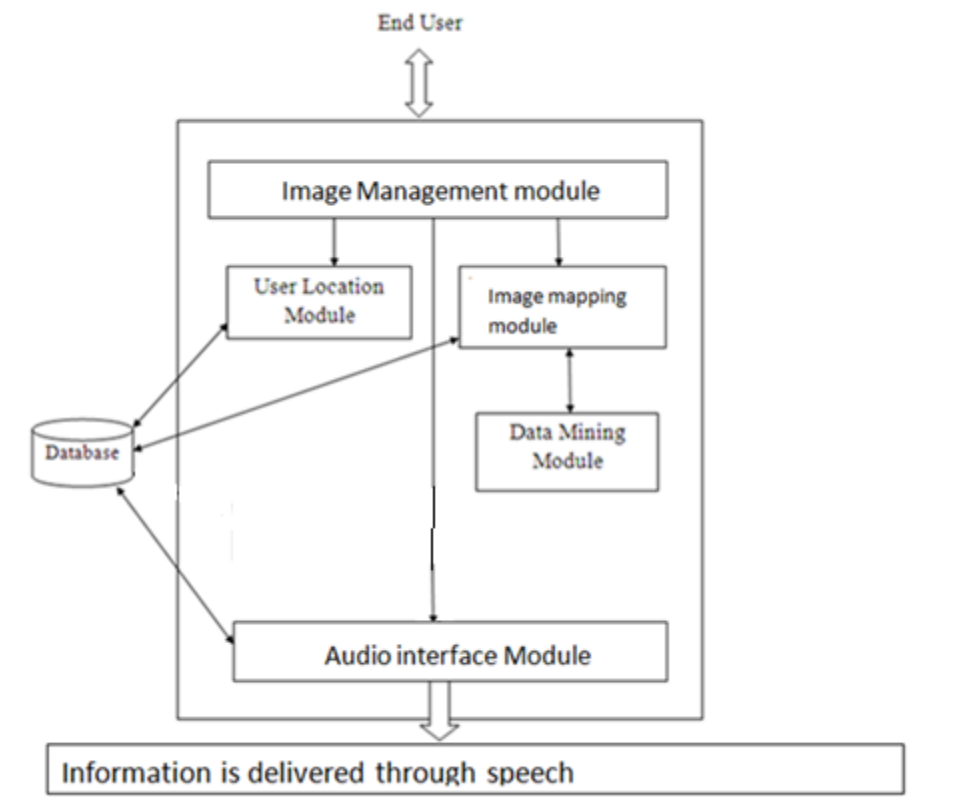


Figure 4.2.1

Modules:

1) Image management module

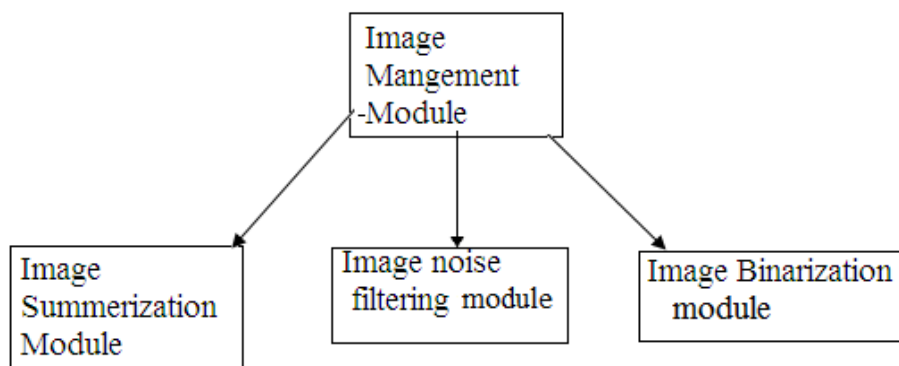


Figure 4.2.2

This module filters noise and smoothenes the captured image and performs binarization on it. It then detects the correct image from the captured image with the help of contour analysis.

Image summarization module: It then detects the correct image from the captured image with the help of boundary extraction

Image filtering module: This module filters noise and smoothenes the captured image

Image binarization module: This module performs binarization on it thereby converting a grayscale or an RGB image to a binary image .

2) User location module

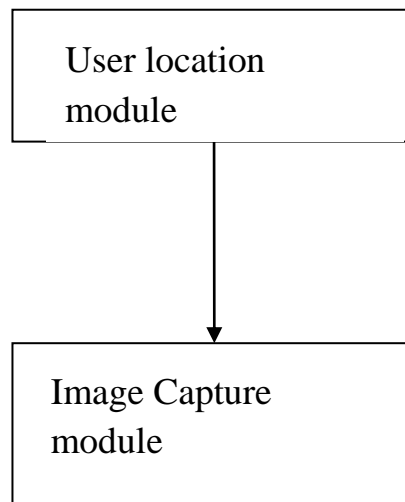


Figure 4.2.3

This module is basically used for locating the user in front of the webcam and then capturing the image of which the user requires information.

3) Image mapping module

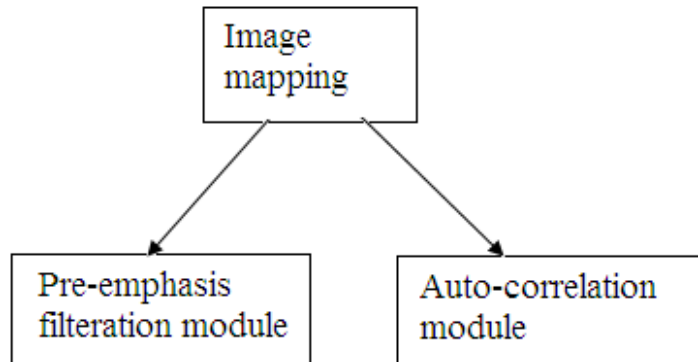


Figure 4.2.4

This module, deals with mapping the captured image to the templates present in the temporary database.

Pre-emphasis filtration module: This module checks for similarities between the query image and the template in database based on colour, shape and texture.

Colour: RGB to HSV conversion

Texture: matching energy levels of the two images

Shape: Through signature analysis

Auto-correlation module: References of the image are present in the database and this module checks whether the captured image matches about 90% to any of the templates present in the database.

4) Data mining module

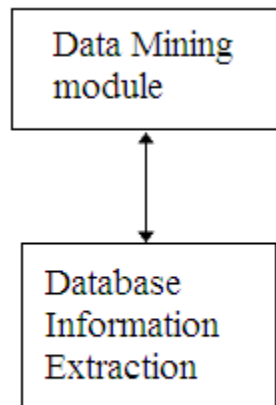


Figure 4.2.5

It performs extraction of data from the temporary database if the templates are matched. It also does the work of retrieving information from the search engines. Data once collected is stored in the database which keeps itself up-to-date with time.

5) Audio interface module

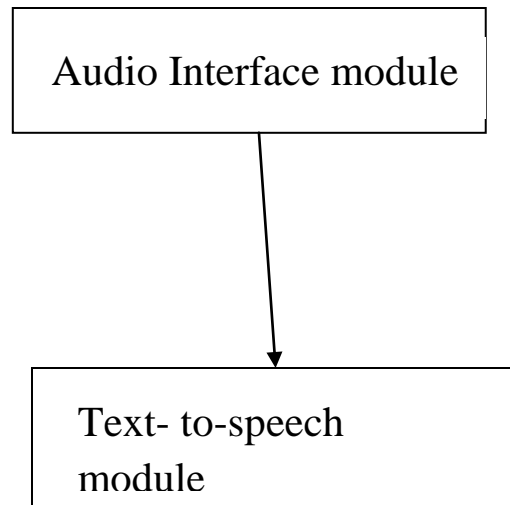


Figure 4.2.6

This module deals with converting the retrieved information to speech and providing the user with the required information with the help of an audio interface.

4.3 Implemetation

4.3.1 Installation of Matlab 2010

This will complete the successful installation.

4.3.2 Installation of drivers for the external devices like webcam and audio interface

Chapter 5

Results and discussions

5.1 Sample code for key scenarios

5.1.1 CBIR

```
clc
clear all
imaqreset
resultValues = [];    % Results matrix...
resultNames = {};
i = 1;                % Indices...
j = 1;
filesList = dir(strcat(pwd, '\data\*.bmp'));
filesListCount = size(filesList,1);

%% Add training and test directories to path
addpath('./data');

%% Input Image
% vid=videoinput('winvideo',2,'YUY2_320x240');
% preview(vid);
% pause(10);
% img1 = getsnapshot(vid);
% closepreview(vid)
[queryx, querymap] = imread('331.bmp');
%% Colour Search
for filesListCounter = 1: filesListCount
    [X, RGBmap] = imread(strcat(pwd, '\data\',filesList(filesListCounter).name));
    HSVmap = rgb2hsv(RGBmap);

    D = quadratic(queryx, querymap, X, HSVmap);
    resultValues(i) = D;
```

```

b = fileList(fileListCounter).name;
len_file = length(b);
len_file = len_file - 4;
for k = 1 : len_file;
    o_file(1,k) = b(1,k);
end
o_file = char(o_file);
o_file = cellstr(o_file);
resultNames(j) = o_file;
o_file = [];
i = i + 1;
j = j + 1;
end
resultValues;
resultNames;

[sortedValues, index] = sort(resultValues);
fid = fopen('colourResults.txt', 'w+');      % Create a file, over-write old ones.
for i = 1:5    % Store top 10 matches...
    tempstr = strcat(char(resultNames(index(i))),'.bmp');
    %    disp(resultNames(index(i)));
    fprintf(fid, '%s\r', tempstr);
    %    disp(sortedValues(i));
    %    disp(' ');
end
fclose(fid);

c_show = strcat(pwd,'\data\',resultNames(index(1)),'.bmp');
c_show = char(c_show);
figure;
imshow(c_show);
c_name = strcat(resultNames(index(1)),'.bmp');

```

```

c_name = char(c_name);
title(strcat('Recognise image through color is (' ,c_name,')'));

displayResults('colourResults.txt', 'Colour Results...');

```

%% Texture part

```

queryEnergies = obtainEnergies(queryx, 6);      % Obtain top 6 energies of the image.

% Open colourResults txt file... for reading...
fid = fopen('colourResults.txt');

fresultValues = [];    % Results matrix...
fresultNames = {};
i = 1;                % Indices...
j = 1;

while 1
    imagename = fgetl(fid);
    if ~ischar(imagename), break, end    % Meaning: End of File...

    [X, RGBmap] = imread(imagename);

    imageEnergies = obtainEnergies(X, 6);

    E = euclideanDistance(queryEnergies, imageEnergies);

    fresultValues(i) = E;
    fresultNames(j) = {imagename};
    i = i + 1;

```

```

    j = j + 1;
end

fclose(fid);

% Sorting final results...

[sortedValues, index] = sort(fresultValues);    % Sorted results... the vector index
                                                % is used to find the resulting files.

fid = fopen('textureResults.txt', 'w+');        % Create a file, over-write old ones.

for i = 1:4    % Store top 5 matches...
    imagename = char(fresultNames(index(i)));
    fprintf(fid, '%s\r', imagename);

    %   disp(imagename);
    %   disp(sortedValues(i));
    %   disp(' ');

end

fclose(fid);
displayResults('textureResults.txt', 'Texture Results...');
t_show = strcat(pwd, '\data\', fresultNames(index(1)));
t_show = char(t_show);
figure;
imshow(t_show);
t_name = fresultNames(index(1));
t_name = char(t_name);
title(strcat('Recognise image through texture is (' , t_name, ')'));

```

%% Shape results

```
inimg = shap_pre(queryx, querymap);
resultValues = [];    % Results matrix...
resultNames = {};
i = 1;                % Indices...
j = 1;
for filesListCounter = 1: filesListCount
    [X, RGBmap] = imread(strcat(pwd, '\data\', filesList(filesListCounter).name));
    dbimg = shap_pre(X, RGBmap);
    D = corr2(inimg, dbimg);

    resultValues(i) = D;
    b = filesList(filesListCounter).name;
    len_file = length(b);
    len_file = len_file - 4;
    for k = 1 : len_file;
        o_file(1,k) = b(1,k);
    end
    o_file = char(o_file);
    o_file = cellstr(o_file);
    resultNames(j) = o_file;
    o_file = [];
    i = i + 1;
    j = j + 1;
end
resultValues;
resultNames;

[sortedValues, index] = sort(resultValues, 'descend');
fid = fopen('shapeResults.txt', 'w+');    % Create a file, over-write old ones.
```

```

for i = 1:5    % Store top 10 matches...
    tempstr = strcat(char(resultNames(index(i))),'.bmp');
%    disp(resultNames(index(i)));
    fprintf(fid, '%s\r', tempstr);
%    disp(sortedValues(i));
%    disp(' ');
end
fclose(fid);

s_show = strcat(pwd,'\data\',resultNames(index(1)),'.bmp');
s_show = char(s_show);
figure;
imshow(s_show);
s_name = strcat(resultNames(index(1)),'.bmp');
s_name = char(s_name);
title(strcat('Recognise image through shape is ('s_name,')'));

displayResults('shapeResults.txt', 'Shape Results...');

o_color = strcat('Color name:',c_name);
disp(o_color);
tts(o_color);

o_texture = strcat('Texture name:',t_name);
disp(o_texture);
tts(o_texture);

o_shape = strcat('Shape name:',s_name);
disp(o_shape);
tts(o_shape);

```

5.1.2 OCR

warning off

clc, close all, clear all

%z=0;

%imagen=imread('untitled.bmp');%Read Binary Image

imaqreset;

vid=videoinput('winvideo',1,'YUY2_320x240');

preview(vid)

pause(15)

a=getsnapshot(vid);

g=a(:,1);%to select red matrix write 1, green -2 , blue -3

imview(g);

log=roicolor(g,0,100); %change the values according to estimate.m yellow(2):210
255, blue(2):55 85, red(1):155,175,

imagen =clip(log);

%IMVIEW(log);

imshow(imagen);%title('INPUT IMAGE WITH NOISE')

%*-*Filter Image Noise*-*-*

if length(size(imagen))==3 %RGB image

 imagen=rgb2gray(imagen);

end

imagen = medfilt2(imagen);

[f c]=size(imagen);

imagen (1,1)=255;

imagen (f,1)=255;

imagen (1,c)=255;

imagen (f,c)=255;


```

%imview(imagen);
%*-*-*END Filter Image Noise*-*-*
word=[];%Storage matrix word from image
re=imagen;
fid = fopen('text.txt', 'wt');%Opens text.txt as file for write
while 1
    [fl re]=lines(re);%Fcn 'lines' separate lines in text
    rq=fl;
    while 1
        [fl1 rq]=columns(rq);
        imgn=~fl1;
        %*-*Uncomment line below to see lines one by one*-*-*
        % imview(fl);pause(1)
        %*-*-*-*-*-*-*
        %*-*-*-*Calculating connected components*-*-*-*
        L = bwlabel(imgn);
        mx=max(max(L));
        BW = edge(double(imgn),'sobel');
        [imx,imy]=size(BW);
        for n=1:mx
            [r,c] = find(L==n);
            rc = [r c];
            [sx sy]=size(rc);
            n1=zeros(imx,imy);
            for i=1:sx
                x1=rc(i,1);
                y1=rc(i,2);
                n1(x1,y1)=255;
            end
            %*-*-*-*END Calculating connected components*-*-*-*
            n1=~n1;

```

```

n1=~clip(n1);
img_r=same_dim(n1);%Transf. to size 42 X 24
%*-!*Uncomment line below to see letters one by one*-!*-!*
    %imshow(img_r);pause(1)
%*-!*-!*-!*-!*-!*
letter=read_letter(img_r);%img to text
word=[word letter];
end
if isempty(rq)
    fprintf(fid,'%s\n',word);
else

%fprintf(fid,'%s\n',lower(word));%Write 'word' in text file (lower)
fprintf(fid,'%s\t',word);%Write 'word' in text file (upper)
end

word=[];%Clear 'word' variable

if isempty(rq) %See variable 're' in Fcn 'lines'
    break
end
end
if isempty(rq)
    if isempty(re)
        break
    end
end
end

%*-!*When the sentences finish, breaks the loop*-!*-!*

%*-!*-!*-!*-!*-!*-!*-!*-!*-!*-!*-!*-!*-!*-!*-!*-!*

```

end

fclose(fid);

winopen('text.txt')%Open 'text.txt' file

fid = fopen('text.txt', 'r');


a = fscanf(fid,'%c');

tts (a);

fclose(fid)


5.2 Code metrics

jsmeter.info




Edit

Visualize

[Share](#) [Tweet](#) 


Line	Function	Statements	Lines	Comment Lines	Comment%	Branches	Depth	Cyclomatic Complexity	Halstead Volume	Halstead Potential	Program Level	MI
1	[[code]]	8	13	0	0%	0	0	1	238	4.75	0.0200	110.50
1	=	2	3	0	0%	0	0	1	24.2	8.00	0.331	142.08

jsmeter.info




Edit

Visualize

[Share](#) [Tweet](#) 


Line	Function	Statements	Lines	Comment Lines	Comment%	Branches	Depth	Cyclomatic Complexity	Halstead Volume	Halstead Potential	Program Level	MI
1	[[code]]	19	19	0	0%	0	0	1	302	4.75	0.0157	103.54

jsmeter.info



Edit

Visualize

[Share](#) [Tweet](#) 

Line	Function	Statements	Lines	Comment Lines	Comment%	Branches	Depth	Cyclomatic Complexity	Halstead Volume	Halstead Potential	Program Level	MI
1	[[code]]	32	26	0	0%	1	1	2	351	4.75	0.0135	97.715
1	displayResults	1	2	0	0%	0	0	1	8.42	11.6	1.38	152.25

5.2.1 Performance

Performance testing can be applied to understand the softwares scalability, or to benchmark the performance in the environment of third party products.

Sr no	Procedure	Success
1	Many images captured one after the other	Retrieving images properly and mapping them correctly with the templates in the

		database
2	Many responses from the database for the image	No Clustering in the system

5.2.2 Usability:

Usability testing is the process by which the human-computer interaction characteristics of a system are measured, and weaknesses are identified for correction.

- Ease of learning
- Navigation
- Subjective user satisfaction
- General appearance

5.2.3 Client Side Compatibility:

The client side compatibility is also tested in various platforms, using various languages etc.

5.2.4 Server Side Interface:

Compatibility of server with software and database should be tested so that retrieval of information is appropriate and satisfactory

5.3 Test cases

Test Case No.1

Test Title : Webcam Functionality

Description: This test is done to map working of the webcam. How image is map into the object and search in database.

Inputs : Object's capture by webcam.

Expected Result:

1. Failure: If image is not in database. It will be discarded
2. Success: Actual object image captured by the webcam and searches in database.

Test case No.2

Test Title : Text to speech.

Description :This test checks whether the user appropriately entered text and converts it into audio format.

Inputs : Enter text in the documents.

Expected Result :

1. Failure :Error message will be displayed that user is pressing wrong word.
2. Success :Entered text is converted into the audio.

.

5.4 Result snapshots

5.4.1 CBIR

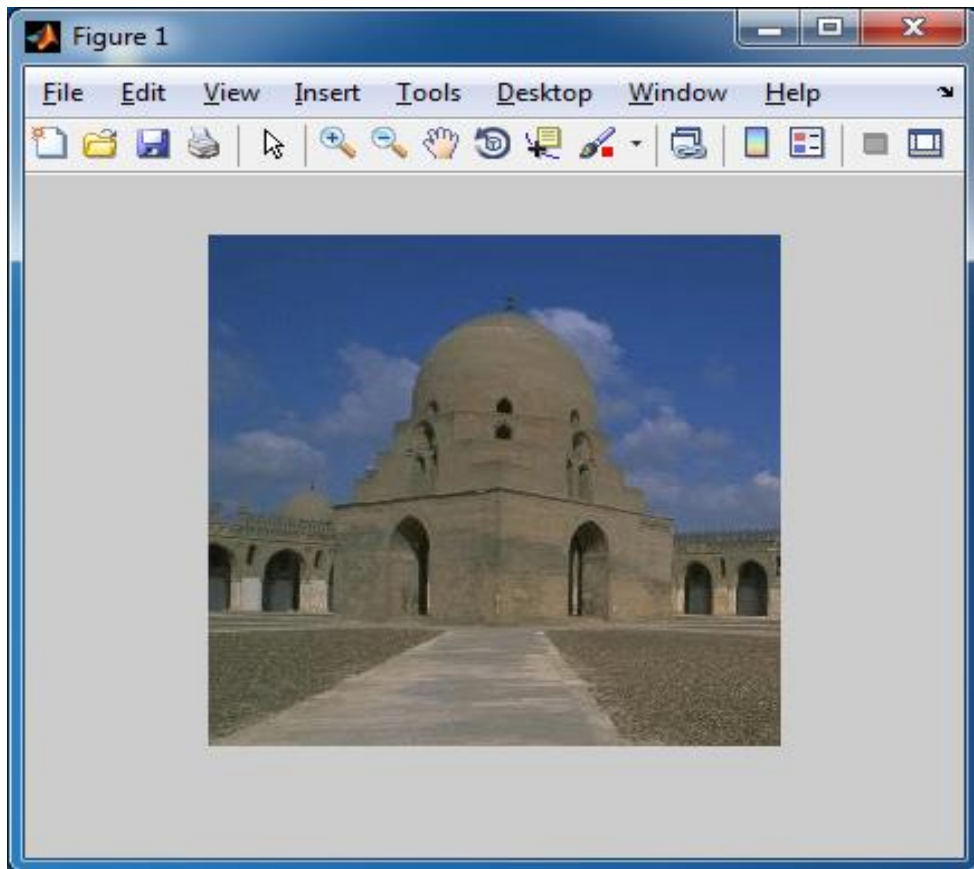


Figure. 5.4.1 image kept in front of camera

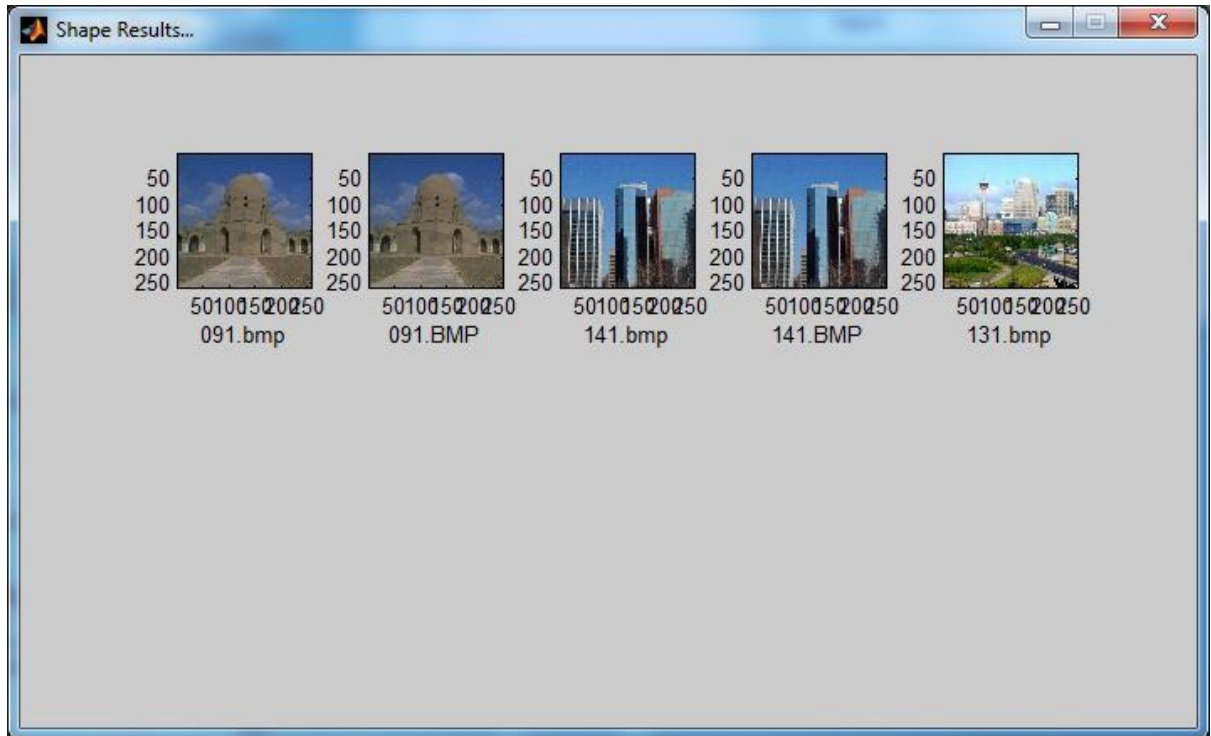


Figure 5.4.2 Results on the basis of comparison with respect to shape

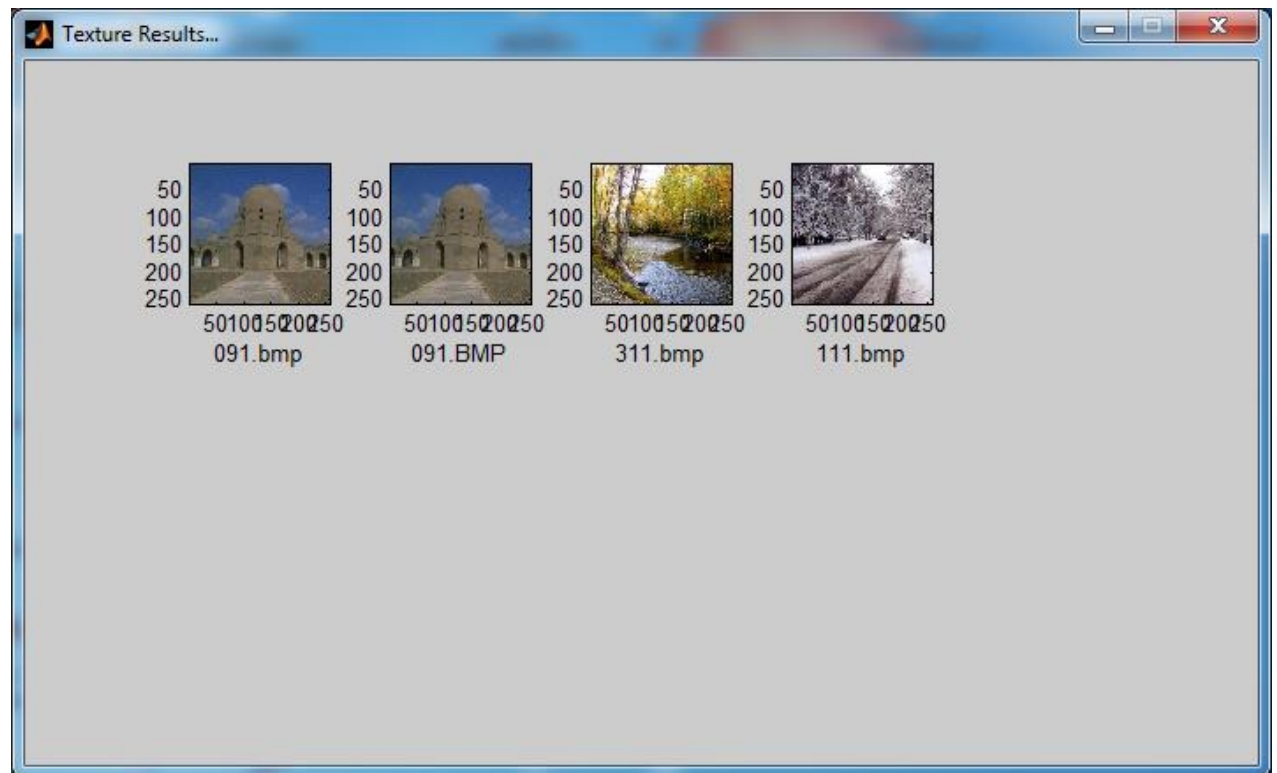


Figure. 5.4.3 Results on the basis of comparison with respect to texture

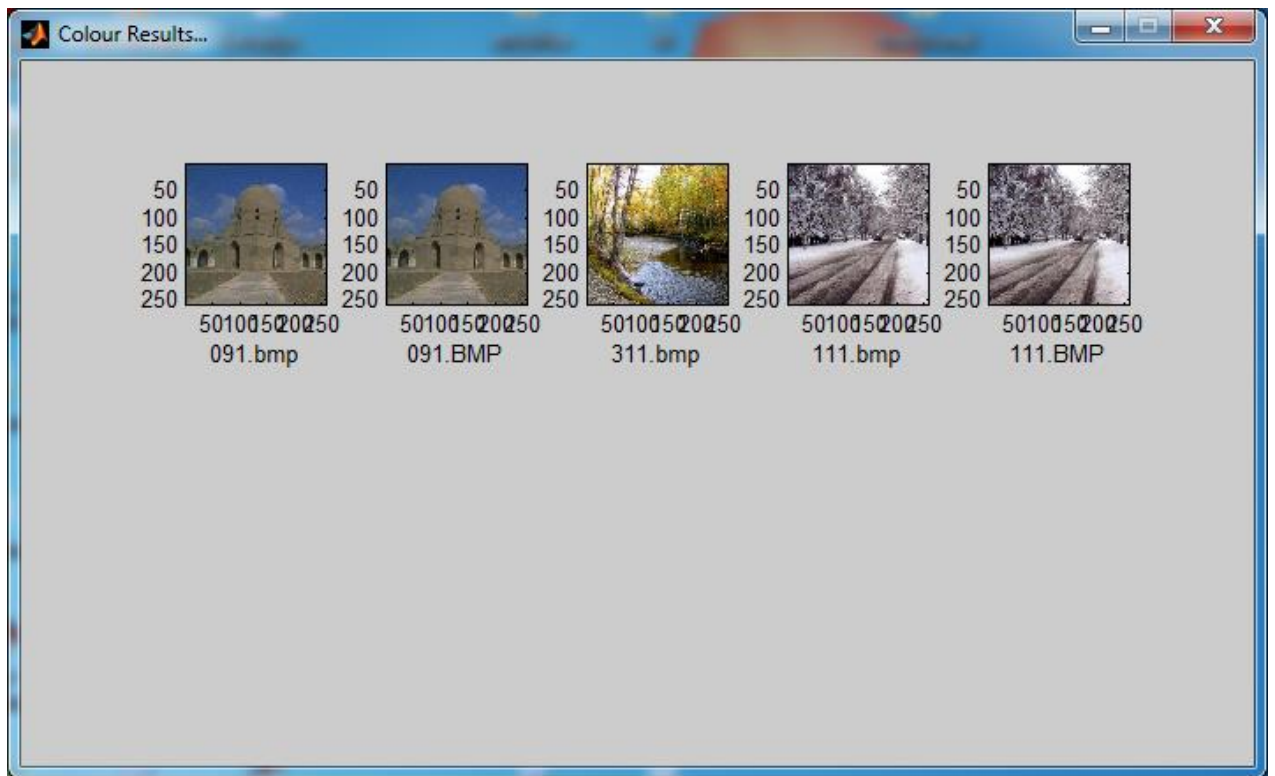


Figure. 5.4.4 Results on the basis of comparison with respect to color

5.4.2 OCR

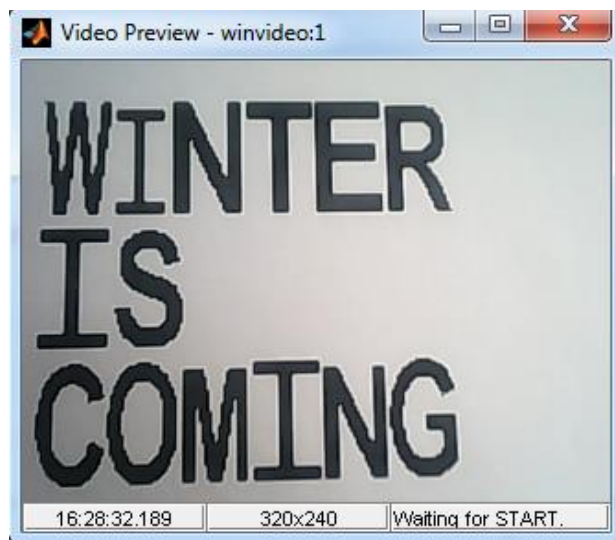


Figure 5.4.5: OCR Text capturing video

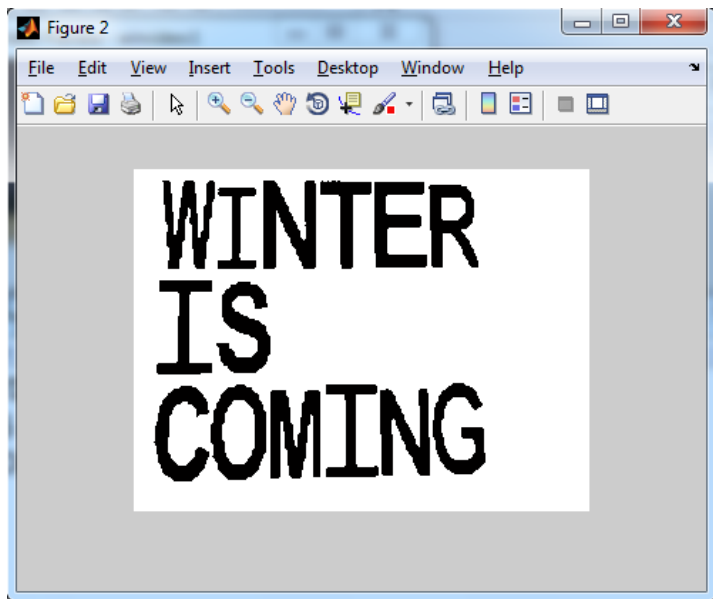


Figure 5.4.6: OCR captured text image

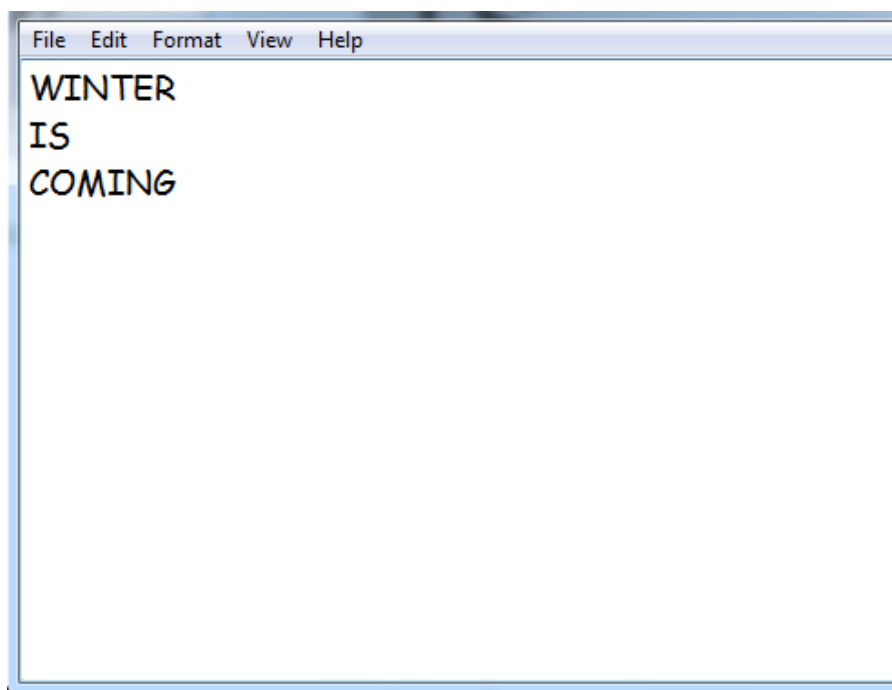


Figure 5.4.7: OCR text converted to speech

Chapter 6

Conclusion and future work

6.1 Conclusion

Existing system provide softwares for teaching the visually impaired children, there is no inclusion of the fun learning element. Thus the proposed system includes the teaching in the way that the visually impaired children would love to learn, i.e. it has an interactive way. . It is an implementation of the Content based image processing along with the Optical content recognition . The proposed system provides the algorithms for Boundary extraction, Object extraction and conversion of text to speech. Thus the final system is implemented using an approach to capture, detect, search in the database, extract information and convert into speech.

6.2 Future work

- Mobile and web applications can be developed for the designed software
- The software can be taken to the next level where motion sensor games can be incorporated to teach the blind students
- Age group can be extended further and more syllabus can be taught
- Multimedia games in collaboration with coherent educational content can be designed using existing haptic devices and braille instruments

Appendix

FUNCTION TEXT_TO_SPEECH

It converts the text derived of an image and even the text captured by the camera into speech

```
function tts( text )
```

```
if nargin<1
```

```
    text = 'Please call this function with text';
```

```
end
```

```
try
```

```
    NET.addAssembly('System.Speech');
```

```
    Speaker = System.Speech.Synthesis.SpeechSynthesizer;
```

```
    if ~isa(text,'cell')
```

```
        text = {text};
```

```
    end
```

```
    for k=1:length(text)
```

```
        Speaker.Speak (text{k});
```

```
    end
```

```
catch
```

```
    warning(['If this is not a Windows system or ' ...
```

```
        'the .Net class exists you will not be able to use this function.' ...
```

```
        'Please let me know what went wrong: wgarn@yahoo.com']);
```

```
end
```

References:

- [1] "Using Haptics in Computer Interfaces for Blind People" by Calle Sjöström
Certec, Lund University
- [2] "TIM(Tactile Interactive Multimedia) Develop[ement and adaptation of computer
games for young blind childrens" by Dominique Archambault and Dominique Burger
- [3] "perkins.com" <http://support.perkins.org/>
- [4] T.K.Patra, www.ijecct.org/v2n5/223_0205M30.pdf
- [5] "Computer games for children with visual impairments" by Y Eriksson1 and D
Gärdenfors
- [6]"Wonderbaby.org": <http://www.wonderbaby.org/articles/best-accessible-computer-games-blind-kids>
- [7] A.Kain:citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.26.2496

Acknowledgement

We have immense pleasure in presenting the synopsis report for our project entitled “Intelligent Software for Teaching Visually Impaired Children”. We would like to take this opportunity to express our gratitude to a number of people who have been sources of help and encouragement during the course of this project.

We are very grateful and indebted to our project guide Prof. Rupali Sawant, for providing her enduring patience, guidance and invaluable suggestions. She was the one who never let our moral down and always supported us through our thick and thin. She is a constant source of inspiration for us and took utmost interest in our project.

We would also like to thank all the Staff Members for their valuable cooperation and permitting us to work in the I.T. Lab.

We are also thankful to all our classmates for giving us their useful advice and immense cooperation. Their support made the working of this project very pleasant. Last but not the least we thank the Almighty for giving us the strength and courage during the development of the project

Laavanya Ganesh
Vishakha Jagtap
Kunal Javeri