# Text-to-Speech Conversion
# Let Matlab Scream

[School]
[Course title]

# PROJECT OVERVIEW

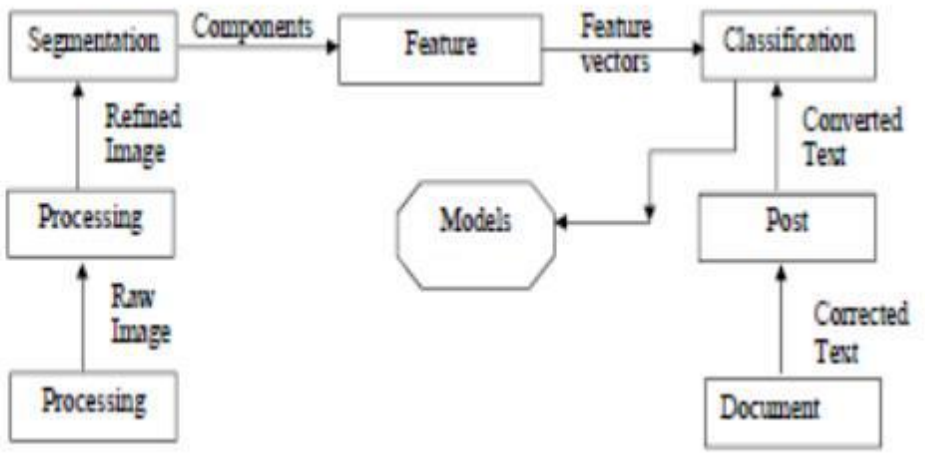
Language is the ability to express one's thoughts by means of a set of signs (text), gestures, and sounds. It is a distinctive feature of human beings, who are the only creatures to use such a system. Speech is the oldest means of communication between people and it is also the most widely used. Technology is always moving towards the betterment for human.

With that goal in mind, we are planning to make a Text to Speech (TTS) system for the handicapped people who have eye problems and also for the people who are still illiterate and cannot read. We have created a MATLAB Project where we have aimed to solve this problem.

## Introduction:

Text-to-speech (TTS) is the generation of synthesized speech from text. Our goal is to make synthesized speech as intelligible, natural and pleasant to listen, as human speech. The quality of a speech synthesizer is judged by its similarity to the human voice and by its ability to be understood. A text-to-speech synthesizer allows people with visual impairments and reading disabilities to listen to written works on a home computer. Many computer operating systems have included speech synthesizers since the early 1990s.

OCR is used to extract characters and symbols from the Text Image without any mistakes, making the software more reliable. It can also extract handwritten as well as printed text into computer-readable text. The performance of OCR can be improved by using proper algorithm.



The above figure shows the internal framework of OCR. Almost all the OCR designs use modification of this basic architecture. Given the text image for recognition, first it is preprocessed for the removal of artifacts and the image is prepared for recognition. It involves binarization, skew correction, normalization, and undergoes image enhancements like filtering out noise and contrast correction. Then, the image is segmented into characters from each other. This process includes two levels. Information from connected component analysis and projection analysis can be used to assist text segmentation. Segmentation is followed by feature extraction, which is concerned with the representation of the object. Feature extraction and classification are the heart of OCR. Feature extraction is expected to make the image invariant to rotation, translation, scaling, line-thickness, etc

## OBJECTIVE

- To understand the speech recognition and its fundamentals.
- Working and applications of 'Image Processing'.
- To gain a real insight into the OCR algorithm and into Speech Synthesis algorithm
- People who are vocally handicapped can communicate with people who do not understand the sign language.
- Children may find this interesting to learn the pronunciation of words.
- To get acquainted with a number of interesting algorithms such as the "Maximally Stable Extremal Regions" , "The Stroke Width Transform" etc.

## BACKGROUND

- The concept of speech recognition started somewhere in 1940s [3], practically the first speech synthesis program was appeared in 1952 at the bell labs, that was about recognition of a digit in a noise free environment.
- 1940s and 1950s consider as the foundational period of the speech recognition technology, in this period work was done on the foundational paradigms of the speech recognition that is automation and information theoretic models.
- In the 1960's we were able to recognize small vocabularies (order of 10-100words) of isolated words, based on simple acoustic-phonetic properties of speech sounds. The key technologies that were developed during this decade were, filter banks and time normalization methods.
- Dennis Klatt's MITalk synthesizer in many senses defined the perception of automatic speech synthesis to the world at large. Later developed into the product DECTalk, it produces somewhat robotic, but very understandable, speech. It is a formant synthesizer, reflecting the state of the art at the time.
- Yoshinori Sagisaka at Advanced Telecommunications Research (ATR) in Japan developed nuu-talk (nuutalk92) in the late 80s and early 90s. It introduced a much larger inventory of concatenative units; thus, instead of one example of each diphone unit, there could be many, and an automatic, acoustically based distance function was used to find the best selection of sub-word units from a fairly broad database of general speech.
- In 2000 ,Prof Keiichi Tokuda's HTS System showed that building generative models of speech, rather than selecting unit instances can generate reliable high quality speech. Its prominence came to the fore front at the first Blizzard Challenge in 2005 which showed that HTS output was reliably understood by listeners. HTS, and so-called HMM synthesis seems to do well on smaller amounts of data, and when then the data is less reliably recorded which offers a significant advantage over the requirement of very large carefully labelled corpora that seem to be required for unit selection work.
- The availability of free and semi-free synthesis systems, such as the Festival Speech Synthesis System and the MBROLA project, makes the cost of entering the field of speech synthesis much lower, and many more groups have now joined in the development.
- After the five decades of research, the speech recognition technology has finally entered marketplace, benefiting the users in variety of ways.
- Our inspiration for this project mainly came from the misery of unprivileged people. We wanted to do something for them. And we felt that, the illiterate faces a lot of problems

when they go to unknown region and cannot understand anything (eg: Train Station). So that is why we wanted to make such a project.

## Methodology:

The entire program is basically divided into 3 main parts.

- Image Processing Phase.
- Calling Matlab's buit in function 'ocr'
- Speech Processing Phase.

Here goes the step-by-step explanation of each of the three phases of the program.

## Image Processing Phase:

### *imread:*

### Syntax:

A = imread(filename)
A = imread(filename, fmt)

### Description:

A = imread(filename) reads the image from the file specified by filename, inferring the format of the file from its contents. If filename is a multi-image file, then imread reads the first image in the file.

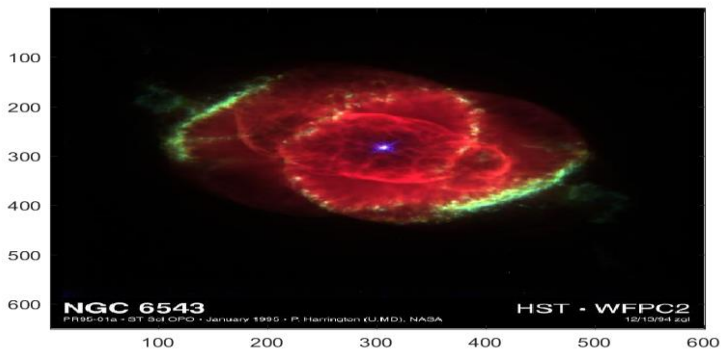
A = imread(filename,fmt) additionally specifies the format of the file with the standard file extension indicated by fmt. If imread cannot find a file with the name specified by filename, it looks for a file named *filename.fmt*.

### Example:

A = imread('ngc6543a.jpg');

imread returns a 650-by-600-by-3 array, A.

image(A);



## Rgb2gray:

This method converts the RGB image to greyscale image.

## Syntax:

I = rgb2gray(RGB);

## Description:

I = rgb2gray(RGB) converts the truecolor image RGB to the grayscale intensity image I. The rgb2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

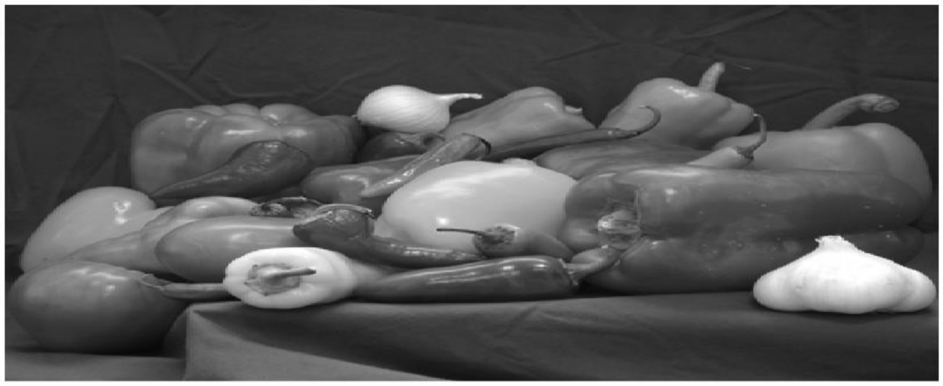
## Examples:

The following programme reads and displays an RGB image, and then converts it to grayscale.Read the sample file, peppers.png, and display the RGB image.

```
RGB = imread('peppers.png');
imshow(RGB);
```

Convert the RGB image to a grayscale image and display it.

```
I = rgb2gray(RGB);
figure
imshow(I)
```



## Algorithm:

Rgb2gray converts RGB values to grayscale values by forming a weighted sum of the *R*, *G*, and *B* components:

0.2989 * R + 0.5870 * G + 0.1140 * B

## *graythresh:*

Global image threshold using Otsu's method.

## Syntax:

```
level = graythresh(I);
```

## Description:

level = graythresh(I) computes a global threshold, level, that can be used to convert an intensity image to a binary image with imbinarize. The graythresh function uses Otsu's method, which chooses the threshold to minimize the intraclass variance of the black and white pixels.

[level ,EM] = graythresh(I) returns the effectiveness metric, EM, as the second output argument. The effectiveness metric is a value in the range [0,1] that indicates the effectiveness of the thresholding of the input image. The lower bound is attainable only by images having a single gray level, and the upper bound is attainable only by two-valued images.

## Examples:

Read a grayscale image into the workspace.

```
I = imread('coins.png');
```

Calculate a threshold using graythresh. The threshold is normalized to the range [0, 1].

```
level = graythresh(I)
```

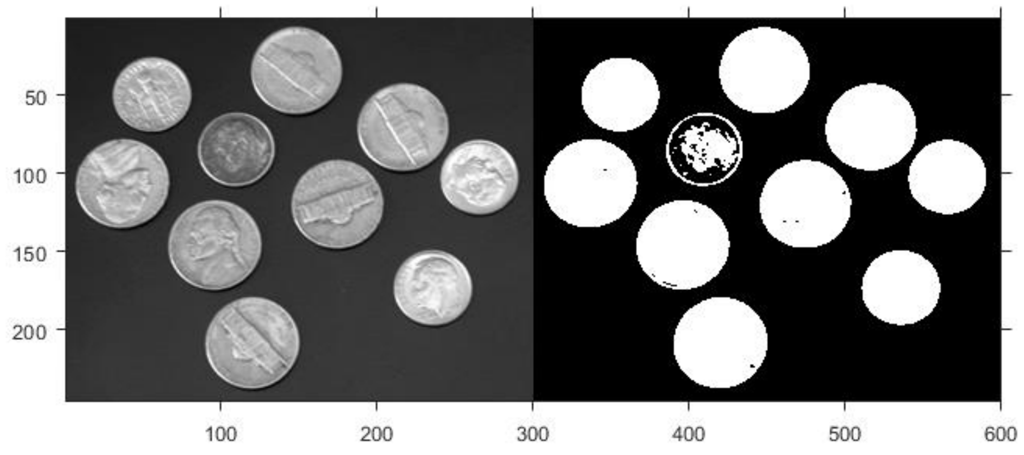
```
level = 0.4941
```

Convert the image into a binary image using the threshold.

```
BW = imbinarize(I,level);
```

Display the original image next to the binary image.

```
imshowpair(I,BW,'montage')
```



## Input Arguments:

Intensity image, specified as a nonsparse N-D array. The graythresh function converts multidimensional arrays to 2-D arrays, using reshape, and ignores any nonzero imaginary part of I.

## Output Arguments:

Global threshold, returned as a positive scalar. level is a normalized intensity value in the range [0, 1].

## *im2bw:*

Convert image to binary image, based on threshold.

### Syntax:

```
BW = im2bw(I,level)
```

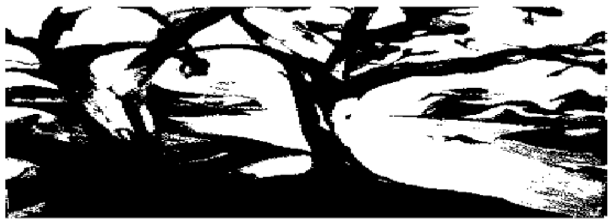
## Description:

BW = im2bw(I,level) converts the grayscale image I to binary image BW, by replacing all pixels in the input image with luminance greater than level with the value 1(white) and replacing all other pixels with the value 0 (black).

This range is relative to the signal levels possible for the image's class. Therefore, a level value of 0.5 corresponds to an intensity value halfway between the minimum and maximum value of the class.

## Example:

```
load trees
BW = im2bw(X,map,0.4);
imshow(X,map), figure, imshow(BW)
```



## Input Arguments:

2-D grayscale image, specified as an *m*-by-*n* numeric matrix.

**Data Types:** single | double | int16 | uint8 | uint16

## Output Arguments: Binary image, returned as an *m*-by-*n* logical matrix.

**Data Types:** logical.

## Ocr:

Recognize text using optical character recognition.

## Syntax:

```
txt = ocr(I)
txt = ocr(I, roi)
```

## Description:

txt = ocr(I) returns an ocrText object containing optical character recognition information from the input image, I. The object contains recognized text, text location, and a metric indicating the confidence of the recognition result.

txt = ocr(I, roi) recognizes text in I within one or more rectangular regions. The roi input contains an *M*-by-4 matrix, with *M* regions of interest.

## Examples:

```
businessCard = imread('businessCard.png');
     ocrResults   = ocr(businessCard)
```

## Output:

```
ocrResults =
  ocrText with properties:

                      Text: '‘ MathWorks®...'
    CharacterBoundingBoxes: [103x4 double]
      CharacterConfidences: [103x1 single]
                     Words: {16x1 cell}
         WordBoundingBoxes: [16x4 double]
           WordConfidences: [16x1 single]
```

## Input Arguments:

Input image, specified in *M*-by-*N*-by-3 truecolor, *M*-by-*N* 2-D grayscale, or binary format. The input image must be a real, nonsparse value. The function converts truecolor or grayscale input images to a binary image, before the recognition process. It uses the Otsu’s thresholding technique for the conversion. For best ocr results, the height of a lowercase ‘x’, or comparable character in the input image, must be greater than 20 pixels. From either the horizontal or vertical axes, remove any text rotations greater than +/- 10 degrees, to improve recognition results.

**Data Types:** single | double | int16 | uint8 | uint16 | logical

**Output Arguments:** Recognized text and metrics, returned as an ocrText object. The object contains the recognized text, the location of the recognized text within the input image, and the metrics indicating the confidence of the results. The confidence values range is [0 1] and represents a percent probability. When you specify an *M*-by-4 roi, the function returns ocrTextas an *M*-by-1 array of ocrText objects.

## Load A Global.NET Assembly:

This example shows how to make .NET classes visible to MATLAB by loading a global assembly using the NET.addAssembly function.

The speech synthesizer class (available in .NET Framework Version 3.0 and above) provides ready-to-use text-to-speech features. For example:

```
NET.addAssembly('System.Speech');
speak = System.Speech.Synthesis.SpeechSynthesizer;
speak.Volume = 100;
Speak(speak,'You can use .NET Libraries in MATLAB')
```

The speech synthesizer class, like any .NET class, is part of an assembly. To work with the class, one needs to call NET.addAssembly to load the assembly into MATLAB. The vendor documentation contains the assembly name. For example, one should search the Microsoft .NET Framework website for the System.SpeechSynthesizer class. The assembly name is System.Speech.

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

The System.Speech assembly is a global assembly.Through this part of the code we invoke the MATLAB speech synthesizer.Here the speak function reads aloud the word with the predefined volume and rate.

### *audiorecorder:*

## Syntax:

```
recorder = audiorecorder
recorder = audiorecorder(Fs,nBits,nChannels)
```

## Description:

recorder = audiorecorder creates an 8000 Hz, 8-bit, 1-channel audiorecorder object.

recorder = audiorecorder(Fs,nBits,nChannels) sets the sample rate Fs (in Hz), the sample size nBits, and the number of channels nChannels.

## Input Arguments:

**Fs**: Sampling rate in Hz. Valid values depend on both the sample rates permitted by MATLAB® and the specific audio hardware on your system. MATLAB has a hard restriction of 1000 Hz <= Fs <= 384000 Hz, although further hardware-dependent restrictions apply. Typical values supported by most sound cards are 8000, 11025, 22050, 44100, 48000, and 96000 Hz.
**Default:** 8000

Nbits:  Bits per sample. Valid values depend on the audio hardware installed: 8, 16, or 24.

**Default:** 8

nChannels: The number of channels 1(mono) or 2 (stereo). Default:1

## ***Record:***

Record audio to audiorecorder object.

### Syntax:

record(*recorderObj*)
record(*recorderObj*, *length*)

### Description:

record(*recorderObj*) records audio from an input device, such as a microphone connected to your system. *recorderObj* is an audiorecorder object that defines the sample rate, bit depth, and other properties of the recording.

record(*recorderObj*, *length*) records for the number of seconds specified by *length*.

## ***getaudiodata:***

Store recorded audio signal in numeric array.

### Syntax:

```
y = getaudiodata(recorder)
y = getaudiodata(recorder, dataType)
```

### Description:

*y* = getaudiodata(*recorder*) returns recorded audio data associated
with audiorecorder object *recorder* to double array *y*.

*y* = getaudiodata(*recorder*, *dataType*) converts the signal data to the specified data
type: 'double', 'single', 'int16', 'int8', or 'uint8'.

## Processing complex/natural input image:

Complex scenes are images that contain undetermined or random scenarios. For example, one can detect and recognize text automatically from captured video to alert a driver about a road sign. This is different than structured scenes, which contain known scenarios where the position of text is known beforehand.

Segmenting text from an complex images greatly helps with additional tasks such as optical character recognition (OCR). The automated text detection algorithm detects a large number of text region candidates and progressively removes those less likely to contain text.

- ❖ Detecting candidate text regions using MSER

The MSER feature detector works well for finding text regions. It works well for text because the consistent color and high contrast of text leads to stable intensity profiles.We have used the detectMSERFeatures function to find all the regions within the image and plot these results.

## ❖ Deleting non-text regions based on basic geometric properties

Although the MSER algorithm picks out most of the text, it also detects many other stable regions in the image that are not text. For example, geometric properties of text can be used to filter out non-text regions using simple thresholds. Alternatively,one can use a machine learning approach to train a text vs. non-text classifier. Typically, a combination of the two approaches produces better results.There are several geometric properties that are good for discriminating between text and non-text regions including

- Aspect ratio
- Eccentricity
- Euler number
- Extent
- Solidity

## ❖ Stroke width variation:

Another common metric used to discriminate between text and non-text is stroke width. *Stroke width* is a measure of the width of the curves and lines that make up a character. Text regions tend to have little stroke width variation, whereas non-text regions tend to have larger variations. One can estimate the stroke width of one of the detected MSER regions. One can do this by using a distance transform and binary thinning operation.

## ❖ Merging the text region for the final detection:

- At this point, all the detection results are composed of individual text characters. To use these results for recognition tasks, such as OCR, the individual text characters must be merged into words or text lines. This enables recognition of the actual words in an image, which carry more meaningful information than just the individual characters. For example, recognizing the string 'EXIT' vs. the set of individual characters {'X','E','T','I'}, where the meaning of the word is lost without the correct ordering.

- One approach for merging individual text regions into words or text lines is to first find neighboring text regions and then form a bounding box around these regions. This makes the bounding boxes of neighboring text regions overlap such that text regions that are part of the same word or text line form a chain of overlapping bounding boxes.

- Now, the overlapping bounding boxes can be merged together to form a single bounding box around individual words or text lines. To do this, one can compute the overlap ratio between all bounding box pairs. This quantifies the distance between all pairs of text regions so that it is possible to find groups of neighboring text regions by looking for non-zero overlap ratios.

- The bboxOverlapRatio function is used to compute the pair-wise overlap ratios for all the expanded bounding boxes, then graph is used to find all the connected regions.
- The output of conncomp are indices to the connected text regions to which each bounding box belongs. These indices are used to merge multiple neighboring bounding boxes into a single bounding box by computing the minimum and maximum of the individual bounding boxes that make up each connected component.
- Finally, before showing the final detection results, false text detections are suppressed by removing bounding boxes made up of just one text region. This removes isolated regions that are unlikely to be actual text given that text is usually found in groups say words and sentences.

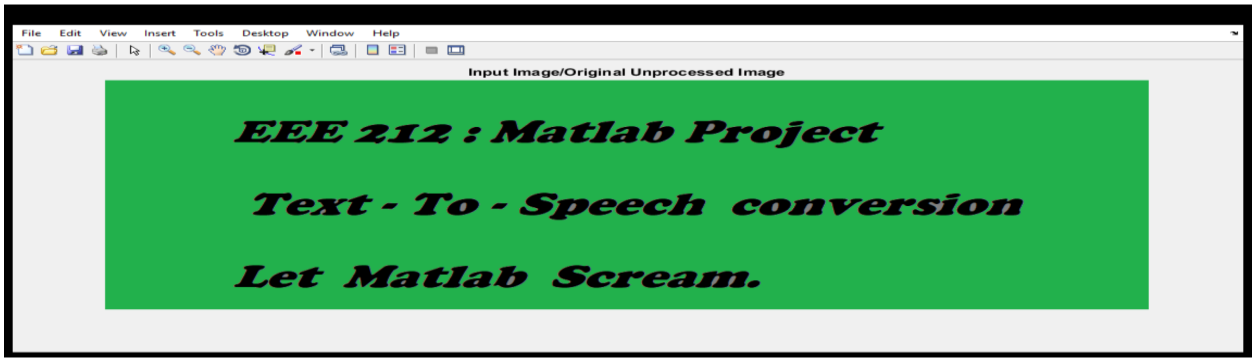
## MSER(Maximally Stable Extremal Regions):

MSER is a method for blob detection in images. The MSER algorithm extracts from an image a number of co-variant regions, called MSERs: an MSER is a stable connected component of some gray-level sets of the image .

- ❖ MSER is based on the idea of taking regions which stay nearly the same through a wide range of thresholds. All the pixels below a given threshold are white and all those above or equal are black.
- ❖ If we are shown a sequence of thresholded images $I_t$ with frame t corresponding to threshold t, we would see first a black image, then white spots corresponding to local intensity minima will appear then grow larger. – These white spots will eventually merge, until the whole image is white.
- ❖ The set of all connected components in the sequence is the set of all extremal regions.
- ❖ Optionally, elliptical frames are attached to the MSERs by fitting ellipses to the regions. Those regions descriptors are kept as features
- ❖ The word extremal refers to the property that all pixels inside the MSER have either higher (bright extremal regions) or lower (dark extremal regions) intensity than all the pixels on its outer boundary

## Implementation, Testing & Results:

- ❖ Image Processing Phases: Input Image:



Input Image/Original Unprocessed Image

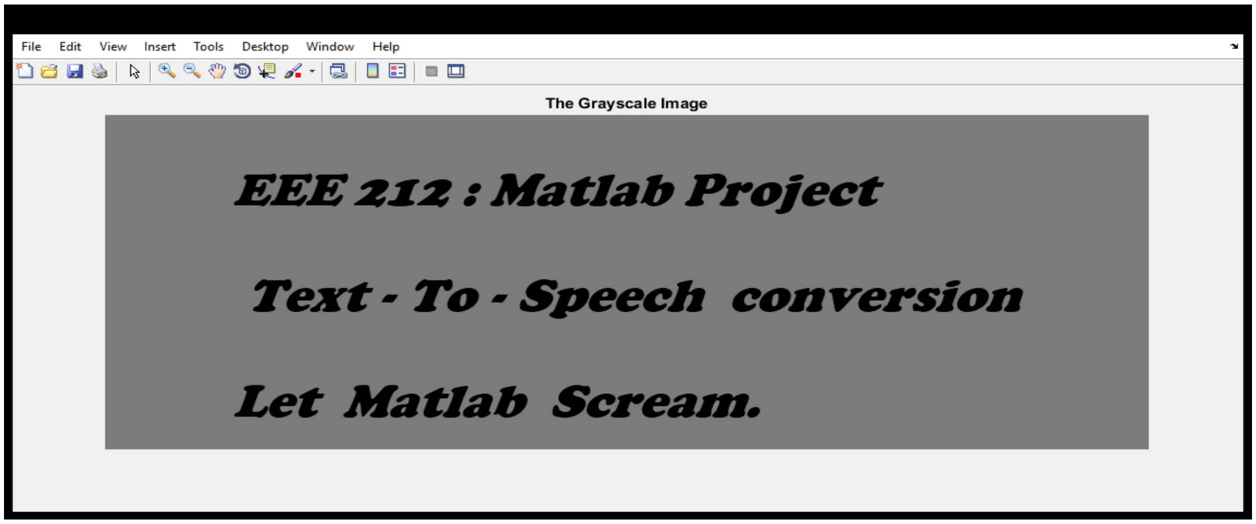
EEE 212 : Matlab Project

Text - To - Speech conversion

Let Matlab Scream.

Fig1: Input Image

- ❖ Converting the RGB image to Grayscale image.



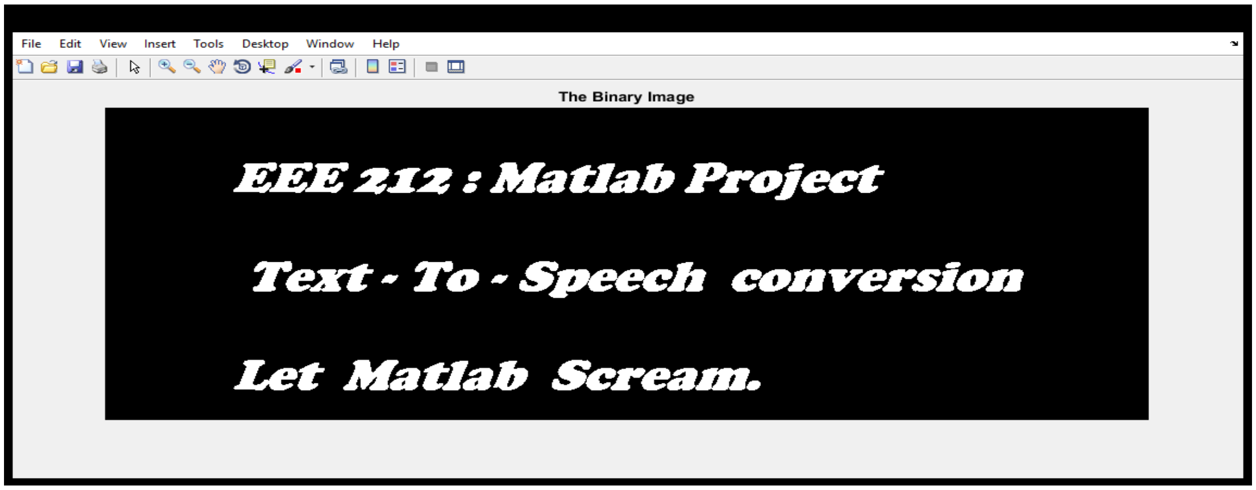
EEE 212 : Matlab Project

Text - To - Speech conversion

Let Matlab Scream.

Fig2: The Grayscale Image

- ❖ Converting the grayscale image into binary one.



EEE 212 : Matlab Project

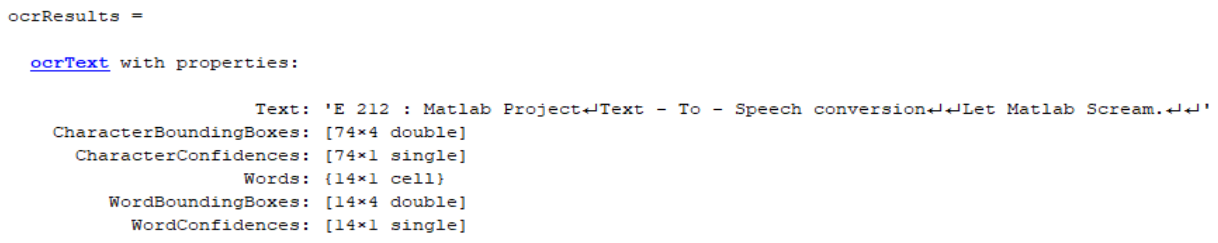
Text - To - Speech conversion

Let Matlab Scream.

Fig 3: Binary Image

## OCR Phase Of Programme:

- ❖ Implementing Matlab's built-in 'ocr' function.



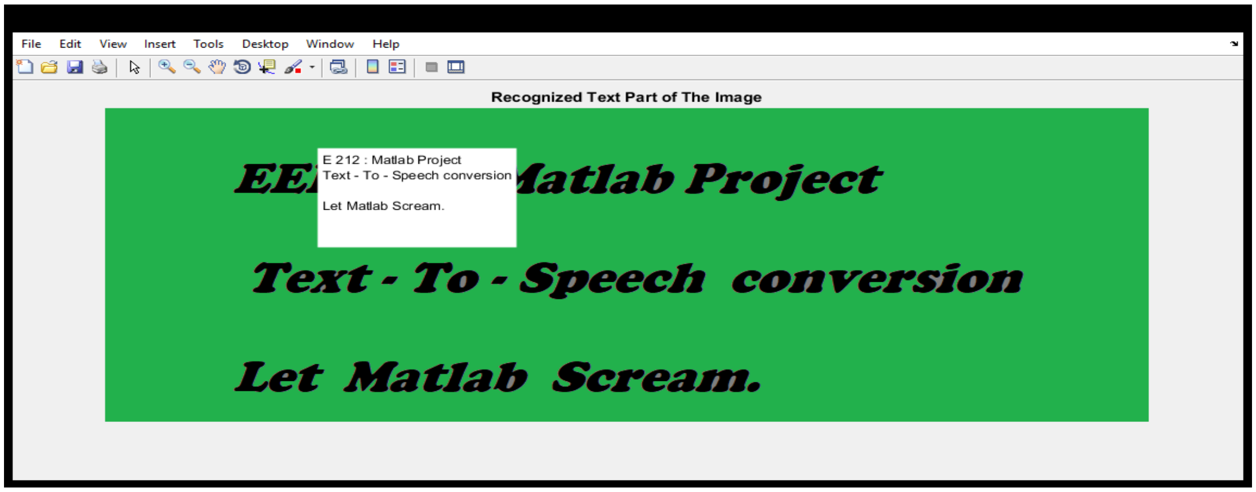
```
ocrResults =

  ocrText with properties:

                      Text: 'E 212 : Matlab Project↵Text - To - Speech conversion↵↵Let Matlab Scream.↵↵'
    CharacterBoundingBoxes: [74×4 double]
     CharacterConfidences: [74×1 single]
                     Words: {14×1 cell}
        WordBoundingBoxes: [14×4 double]
         WordConfidences: [14×1 single]
```

Fig 4:OCR Functions Output

❖ Highlighting the recognized text part of the image.



Recognized Text Part of The Image

EEE 212 : Matlab Project

Text - To - Speech conversion

Let Matlab Scream.

E 212 : Matlab Project
Text - To - Speech conversion

Let Matlab Scream.

Fig 5: Recognized/Detected Text

❖ Presenting the bounding boxes of words with recognition confidences.

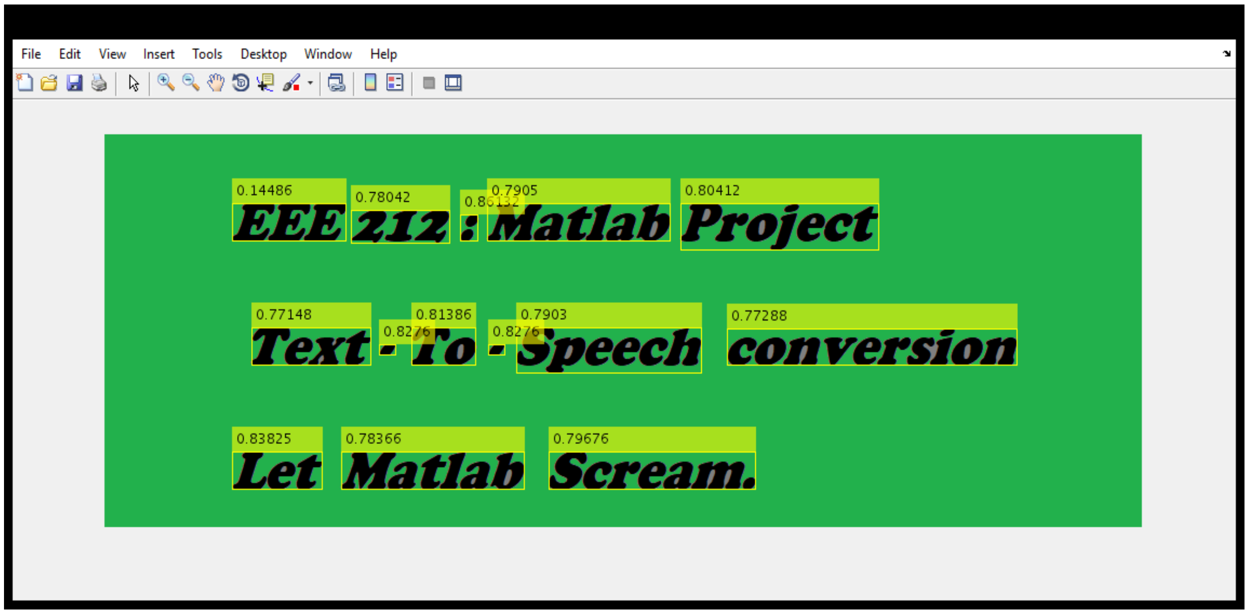
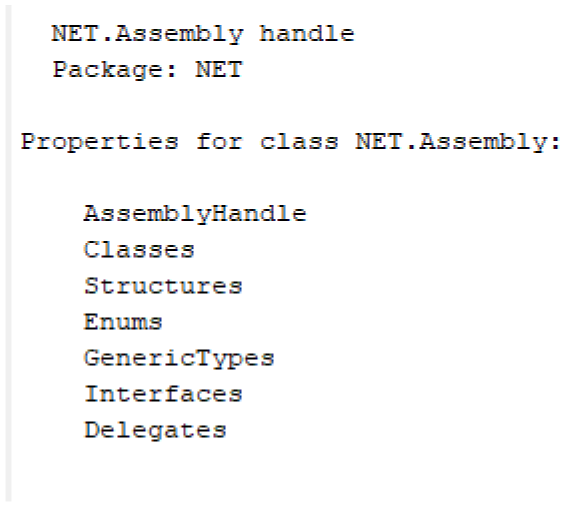


Fig 6: Bounding Boxes of Words

## Speech Processing Phase:

- ❖ Calling the NET.addAssembly to add the library file of the NET class.Here we will implement the ‘System.Speech’ library file to create audio output.



```
  NET.Assembly handle
  Package: NET

Properties for class NET.Assembly:

    AssemblyHandle
    Classes
    Structures
    Enums
    GenericTypes
    Interfaces
    Delegates
```

Fig 7:NET.Assembly

- ❖ We have finally created an object through the ‘audiorecorder’ function to record the speech.
- ❖ Then we have generated a plot of the whole speech using the ‘getaudiodata’ function. Here we have tested the program with ‘Life.png’ file.Here is the plot of the pronunciation of ‘Life’

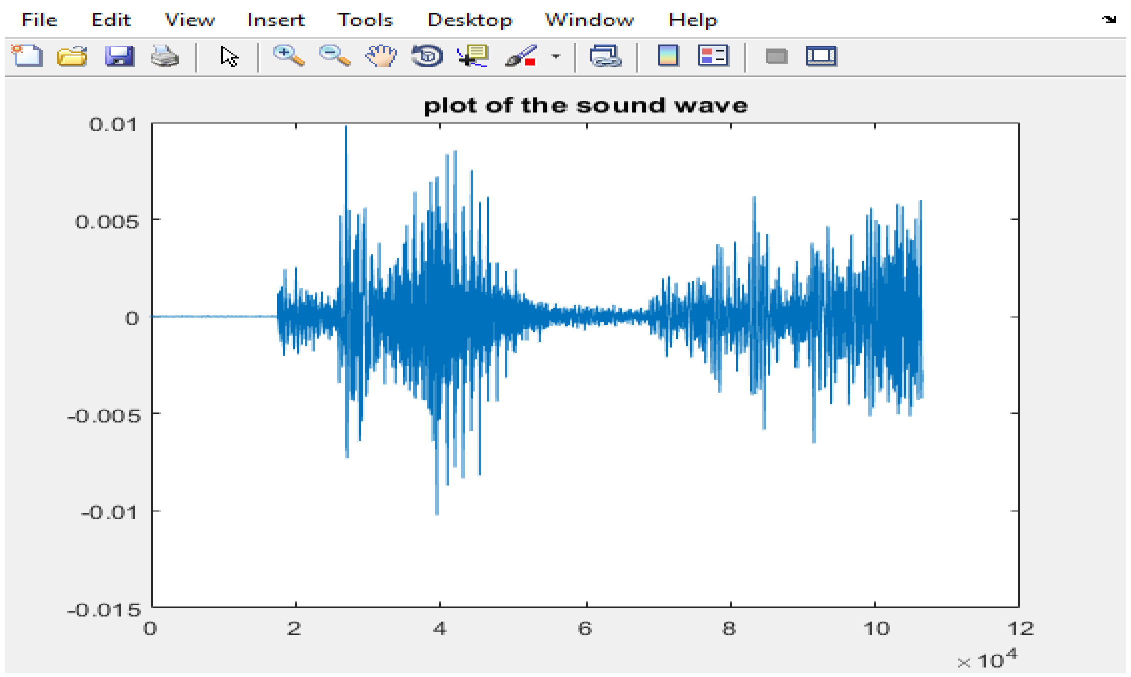


Fig 8: Plot of speech ‘Life’

❖ Finally we have used the ‘Speak’ function to make the matlab speak.

## Testing for Natural/Complex Images:

❖ Taking input image which has objects, texts and other things.

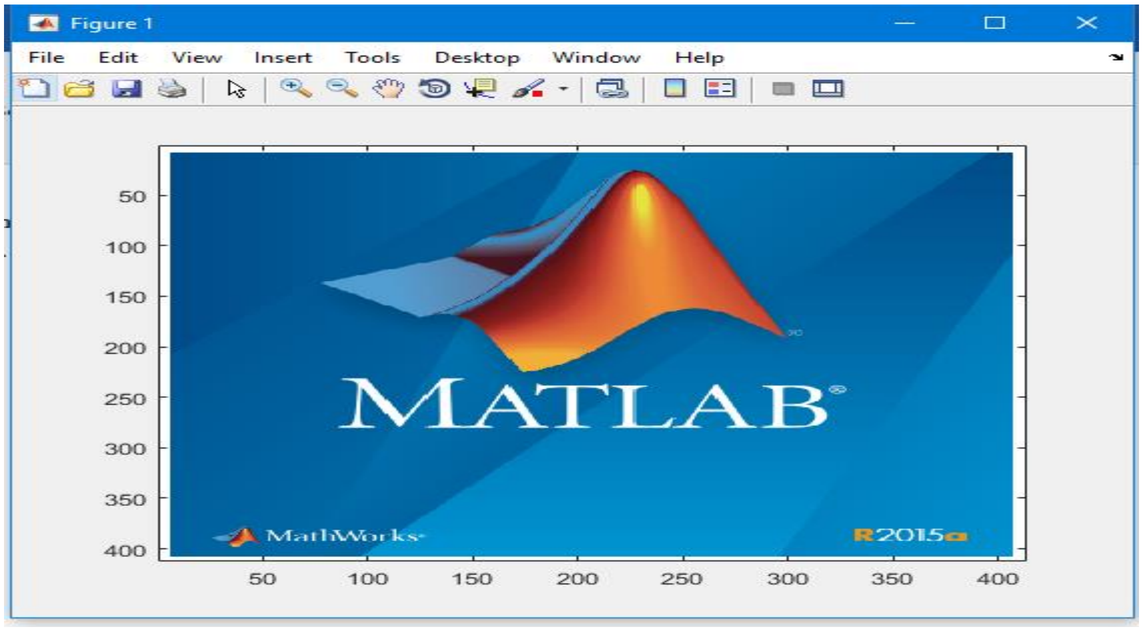


Fig 9: Input Image

❖ Detecting text regions using MSER.

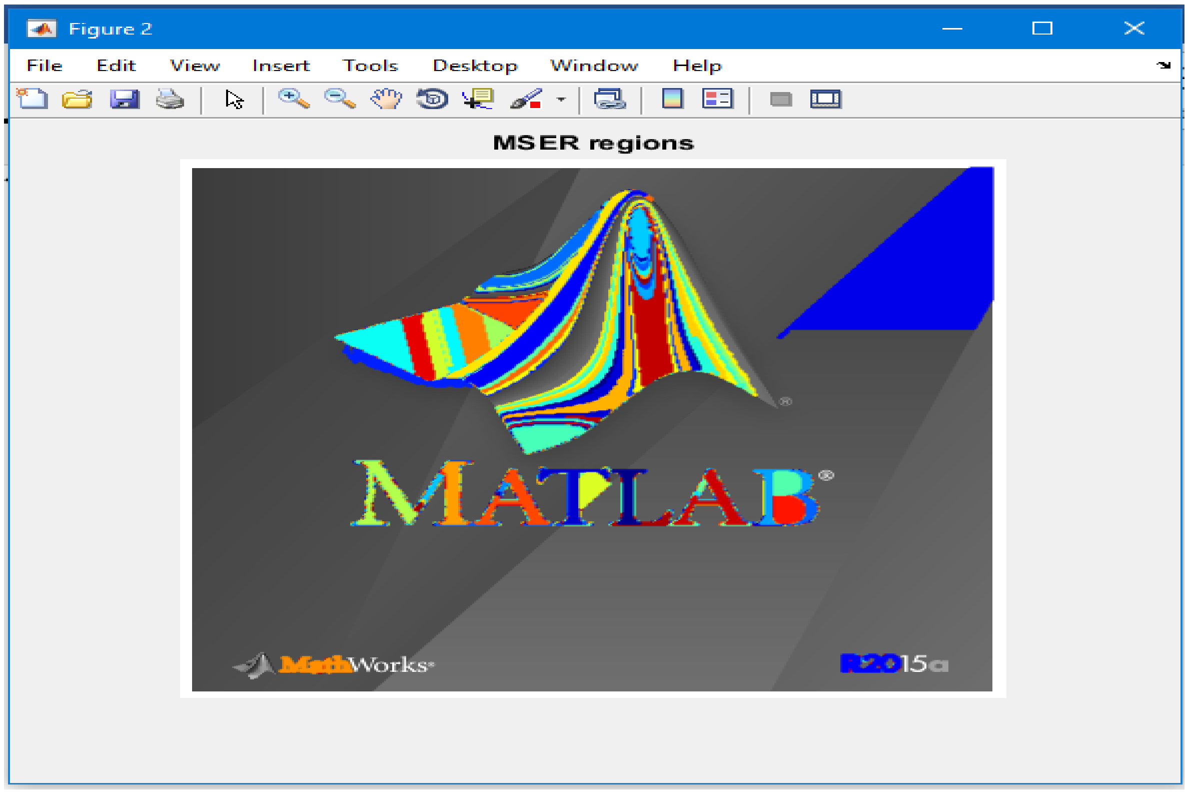


Fig 10: Highlighting MSER regions

❖ Separating the non-text regions based on stroke-width variation.

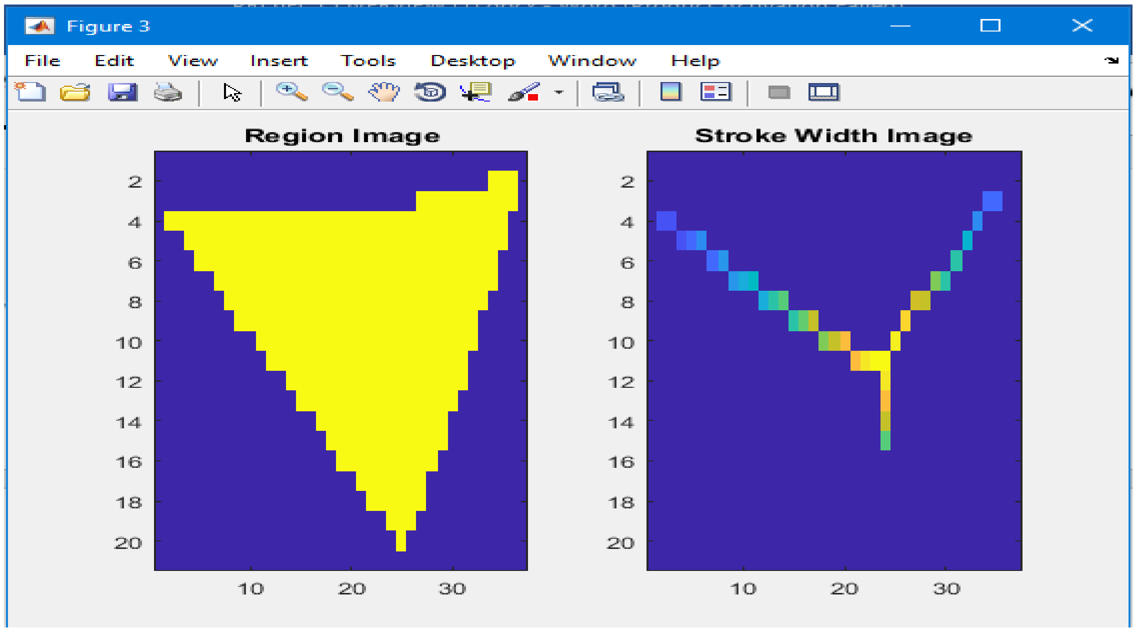


Fig 11: Stroke-width regions

❖ Filtering out the area with objects or which does not contain texts.

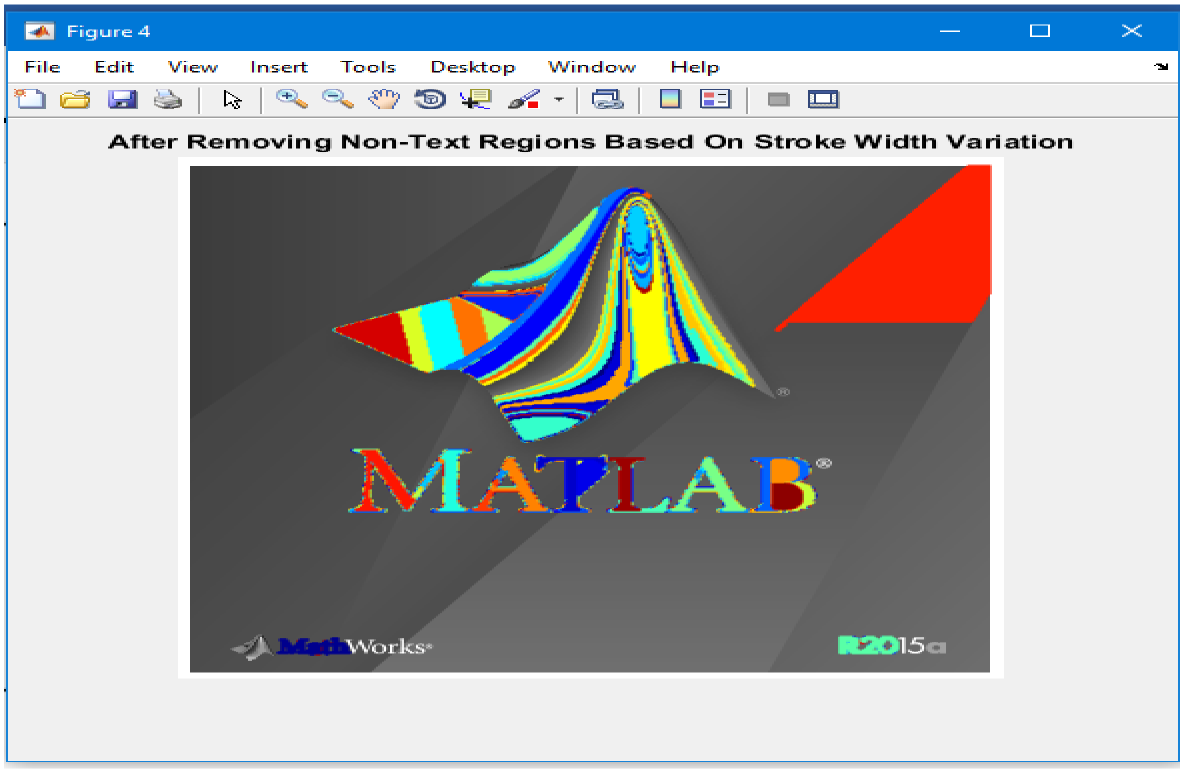


Figure 4
File Edit View Insert Tools Desktop Window Help
After Removing Non-Text Regions Based On Stroke Width Variation
MATLAB
MathWorks
R2015a

Fig 12: Removing Non-text regions.

❖ Merging text regions for final detection result.

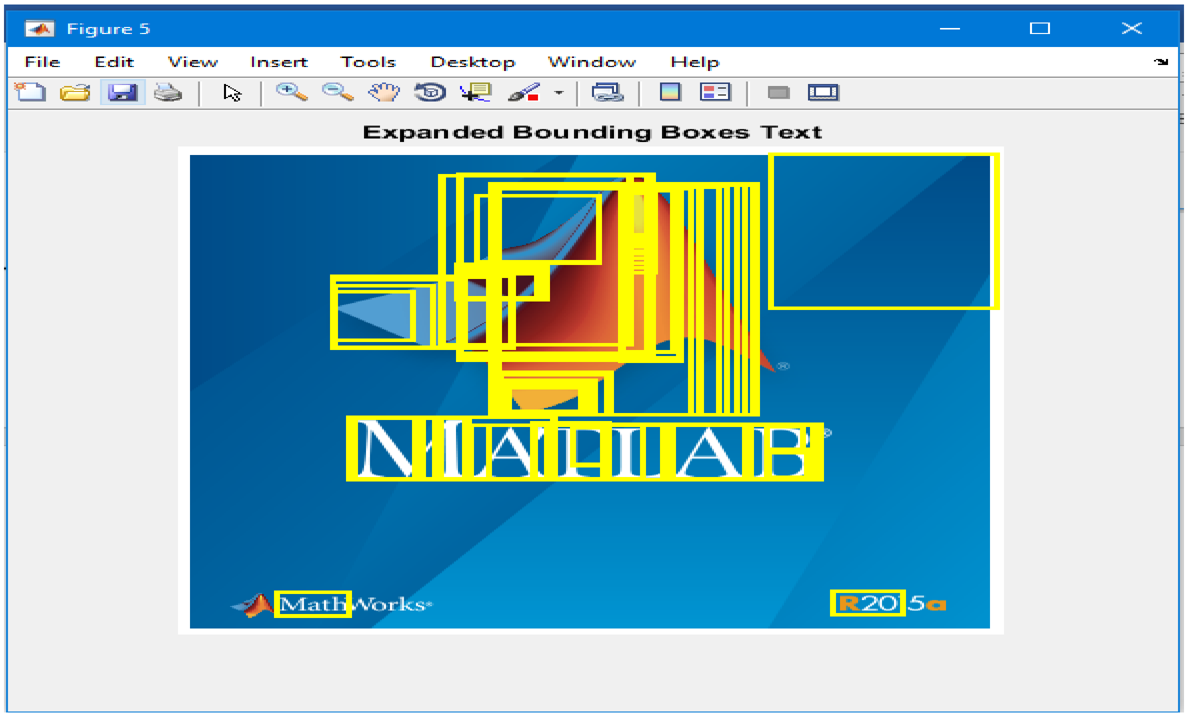


Figure 5
File Edit View Insert Tools Desktop Window Help
Expanded Bounding Boxes Text
MATLAB
MathWorks
R2015a

Fig 13: Bounding Text Box

❖ Final detection of the text in the image.

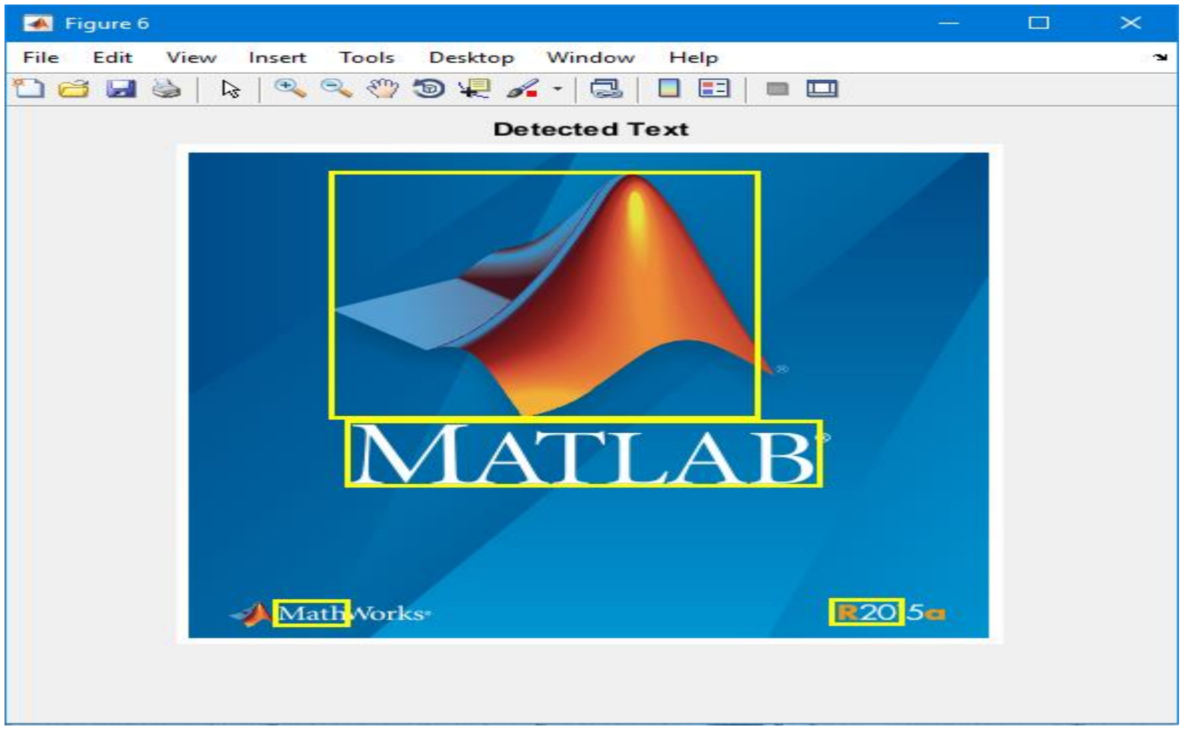


Fig 14: Detected Texts

OCR phase of the programme

- ❖ Implementing Matlab's built-in 'ocr' function.
- ❖ Highlighting the recognized text part of the image.

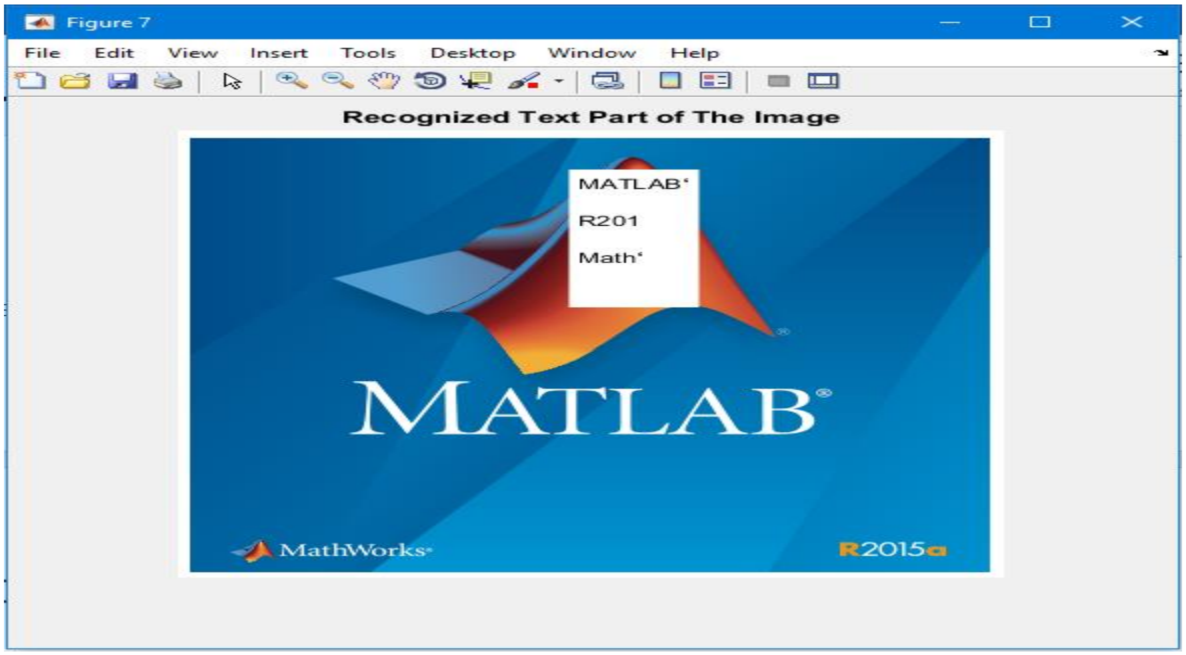
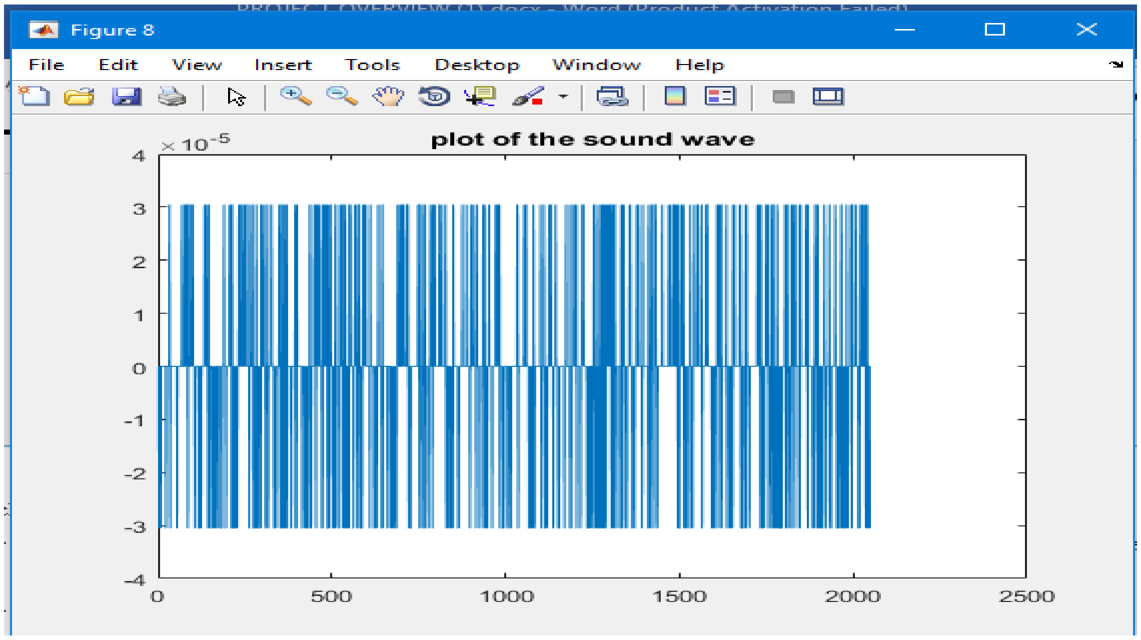


Figure 7
File Edit View Insert Tools Desktop Window Help
Recognized Text Part of The Image
MATLAB'
R201
Math'
MATLAB
MathWorks
R2015a

Fig 15: Recognized Texts in White Box

Speech Processing Phase:

- ❖ We have called the NET.addAssembly to add the library file of the NET class. Here we will implement the 'System.Speech' library file to create audio output.We have implemented 'audiorecorder' & 'getaudiodata' to plot the emerging speech.
- ❖ Finally we have used the 'Speak' function to make the matlab speak.



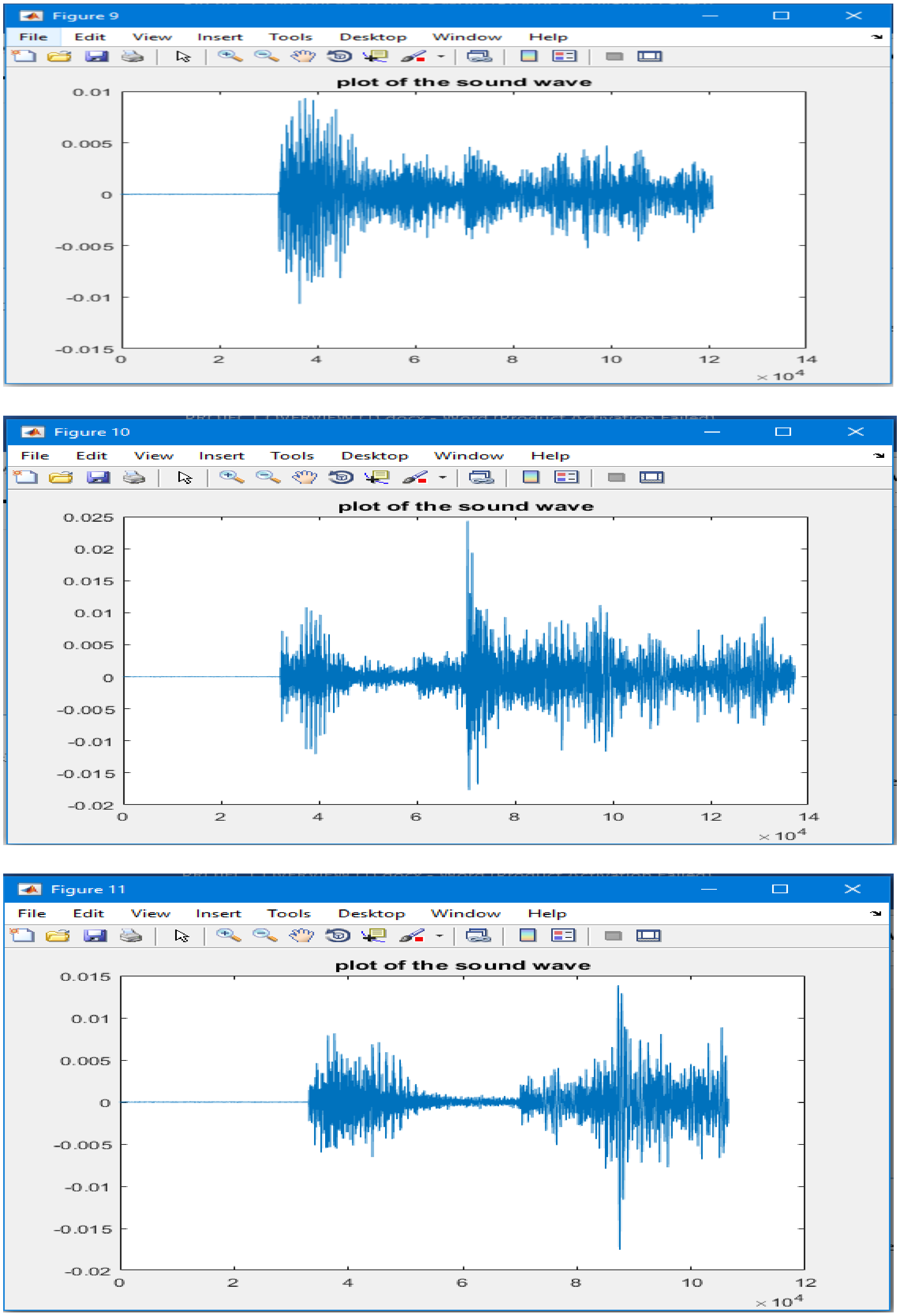


Fig 16: Plot of Emerging Sound Waves.

## Hurdles ,We faced:

- Extracting the text from the images correctly was a paramount task. Infact if this is not done properly, the ‘ocr’ method doesn’t give expected output. We have increased the original image 2-to-4 times to attain the words perfectly.
- If the characters in the image are too close together or their edges are touching, the ‘ocr’ method can’t detect them separately.To overcome this we have used morphology to thin out the characters. Using morphology to thin out the characters separates the characters.
- In binarization of images ,lighting issues are quite important. To check non-uniform lighting issues we have used binarization. If the characters are not visible in the results of binarization ,it indiucates a potential non-uniform lighting issues.We have implemented imtophat function to remove non-uniform illumination.
- The first task faced by any TTS system is the conversion of input text into linguistic representation, usually called text-to-phonetic or grapheme-tophoneme conversion. The difficulty of conversion is highly language depended and includes many problems. In some languages, such as Hindi or Telugu, the conversion is quite simple because written text almost corresponds to its pronunciation. For English and most of the other languages the conversion is much more complicated. A very large set of different rules and their exceptions is needed to produce correct pronunciation and prosody for synthesized speech.

- Special characters and symbols, such as '$', '%', '&', '/', '-', '+', cause also special kind of problems. In some situations the word order must be changed. For example, *$71.50* must be expanded as *seventy-one dollars and fifty cents* and *$100 million* as *one hundred million dollars*, not as *one hundred dollars million*.. Some languages also include special non ASCII characters, such as accent markers or special symbols.

- The second task is to find correct pronunciation for different contexts in the text. Some words, called *homographs*, cause the most difficult problems in TTS systems. Homographs are spelled the same way but they differ in meaning and usually in pronunciation (e.g. fair, lives). The word *lives* is for example pronounced differently in sentences "Three *lives* were lost" and "One *lives* to eat". Some words, e.g. *present*, has different pronunciations depending on the context. (I was *present* there when he received the *present*). The characters 'th' in ‘mother’ and ‘think’ is pronounced differently. Some sounds may also be either voiced or unvoiced in different context. For example, phoneme /s/ in word *dogs* is voiced, but unvoiced in word *cats.*
  Finding correct pronunciation for proper names, especially when they are borrowed from other languages, is usually one of the most difficult tasks for any TTS system. Unfortunately, it is clear that there is no way to build a database of all proper names in the world.

- To make the speech soothable to listen we have made necessary adjustments in the speech processing part.Especially , the volume and rate of speech need to be synchronized properly for a smooth speech. If there is no breath pauses in speech or if they are in wrong places, the speech may sound very unnatural or even the meaning of the sentence may be misunderstood. For example, the input string "John says Peter is a liar" can be spoken as two different ways giving two different meanings as "John says:

Peter is a liar" or "John, says Peter, is a liar". In the first sentence Peter is a liar, and in the second one the liar is John.

## Applications of Synthetic Speech:

Synthetic speech may be used in several applications. some applications, such as reading machines for the blind or electronic-mail readers, require unlimited vocabulary and a TTS system is needed. The application field of synthetic speech is expanding fast whilst the quality of TTS systems is also increasing steadily.
Speech synthesis systems are also becoming more affordable for common customers, which makes these systems more suitable for everyday use. For example, better availability of TTS systems may increase employing possibilities for people with communication difficulties.

## Applications for the Blind:

Probably the most important and useful application field in speech synthesis is the reading and communication aids for the blind. Before synthesized speech, specific audio books were used where the content of the book was read into audio tape. It is clear that making such spoken copy of any large book takes several months and is very expensive. It is also easier to get information from computer with speech instead of using special bliss symbol keyboard, which is an interface for reading the Braille characters. A blind person cannot also see the length of an input text when starting to listen it with a speech synthesizer, so an important feature is to give in advance some information of the text to be read. For example, the synthesizer may check the document and calculate the estimated duration of reading and speak it to the listener. Also the information of boldor underlined text may be given by for example with slight change of intonation or loudness.

## Applications for the Deafened and Vocally Handicapped

People who are born-deaf cannot learn to speak properly and people with hearing difficulties have usually speaking difficulties. Synthesized speech gives the deafened and vocally handicapped an opportunity to communicate with people who do not understand the sign language.

## Educational Applications

Synthesized speech can be used also in many educational situations. A computer with speech synthesizer can teach 24
hours a day and 365 days a year. It can be programmed for special tasks like spelling and pronunciation teaching for
different languages. It can also be used with interactive educational applications.

## Applications for Telecommunications and Multimedia

The newest applications in speech synthesis are in the area of multimedia. Electronic mail has become very usual in last few years. However, it is sometimes impossible to read those E-mail messages when being for example abroad. There may be no proper computer available or some security problems exist. With synthetic speech e-mail messages may be listened to via normal telephone line. Synthesized speech may also be used to speak out short text messages (sms) in mobile phones.

## Other Applications

In principle, speech synthesis may be used in all kind of human-machine interactions. For example, in warning and alarm systems synthesized speech may be used to give more accurate information of the current situation. Using speech instead of warning lights or buzzers gives an opportunity to reach the warning signal for example from a different room.

## Future Enhancements:

The future scope of the project is to add more emphasis simulation by making the speech sound with emotions. We have seen that delay in sound wave causes the speech to look very unnatural. Removing the delay manually every time may be a tedious job. To overcome this problem, we can think of a method which recognizes the delay and automatically remove it. And the best method would be ‘integration’. If we can integrate the synthesized speech, we can not only avoid delay but also get a continuous flow of speech. In this project we have worked only on text documents. Further we can think of reading word files, scanned data, PDF files etc. Moreover, an interesting enhancement would be real time image recognition and converting the image to speech. This will widen the prospects of this project manifold. Since printed document images archived by many applications are more of historical and poor in quality, there is a need to apply advanced image pre-processing techniques for document analysis. Document image processing algorithms for document image collections need more progress. Schemes that learn from document image collections itself for better performance are needed. Recognition from poor quality documents results in a number of recognition errors. Words are not collected correctly fromm the images.This is one of the main problems . Retrieval of documents in this situation requires more functionality. Effective schemes for retrieval in presence of OCR errors there is need to develop Multilingual OCR system so that we can read more than one language documents for document analysis.On top of that , multilingual OCR to transform any sort of printed document to synthetic speech version.

## Appendix:

```
%% TEXT TO SPEECH %%
%==================%
clc
clear all;
close all;           %Clearing the command window and workspace

%%image processing part

i=imread('Matlab.png');    %Here you have to put which photo you want to
read(MAIN INPUT)
figure
imshow(i)
title('Input Image/Original Unprocessed Image');
gray=rgb2gray(i);
figure
imshow(gray);
title('The Grayscale Image');
th=graythresh(i);

bw=~im2bw(i,th);         %Binary Image
figure
imshow(bw);  %See this image and make sure that image has been processed
correctly,if it not happens correctly then you will get garbage output
title('The Binary Image');
```

```
ocrResults=ocr(bw) %Using Optical Character Recognition for recognizing the
text
%Recognize Text Within an image.
recognizedText = ocrResults.Text;
figure;
imshow(i);
title('Recognized Text Part of The Image');
text(200, 100, recognizedText, 'BackgroundColor', [1 1 1]);

%Display Bounding Boxes Of Words & Recognition Confidences
 Iocr         = insertObjectAnnotation(i, 'rectangle', ...
                           ocrResults.WordBoundingBoxes, ...
                           ocrResults.WordConfidences);
figure;title('Bounding Boxes Of Words & Recognbition Confidences');
imshow(Iocr);

for n=1:numel(ocrResults.Words)     %iterate speech part for all text in the
photo

word = ocrResults.Words{n};         %We are taking each word in a variable
and express it one after one

%%Speech processing part

NET.addAssembly('System.Speech')
mysp=System.Speech.Synthesis.SpeechSynthesizer;    %We are using Matlab's in
built voice synthesizer for speech
mysp.Volume=100;                            %Volume of voice(Range : 1-100)
mysp.Rate=2;                                 %Speed of voice (Range : -10 to 10 )
 a = audiorecorder(96000,16,1);   % create object for recording audio
record(a,5);

Speak(mysp,word);                           %Expressing each word

b = getaudiodata(a);                        %store the recorded data in a numeric
array.
b = double(b);
figure
plot(b);
title('plot of the sound wave');
end
```

## Code For Natural/Complex Image :

```
%Text to Speech conversion for complex/natural  image
clc;
clear all;
close all;
colorImage = imread('MatlabImage.png');figure
image(colorImage);
I = rgb2gray(colorImage);
th = graythresh(I);
% Detect MSER regions.
```

```
[mserRegions, mserConnComp] = detectMSERFeatures(I, ...
    'RegionAreaRange',[200 8000],'ThresholdDelta',th);

figure
imshow(I)
hold on
plot(mserRegions, 'showPixelList', true,'showEllipses',false)
title('MSER regions')
hold off
% Use regionprops to measure MSER properties
mserStats = regionprops(mserConnComp, 'BoundingBox', 'Eccentricity',
'Solidity', 'Extent', 'Euler', 'Image');
% Get a binary image of the a region, and pad it to avoid boundary effects
% during the stroke width computation.
regionImage = mserStats(6).Image;
regionImage = padarray(regionImage, [1 1]);
% Compute the stroke width image.
distanceImage = bwdist(~regionImage);
skeletonImage = bwmorph(regionImage, 'thin', inf);

strokeWidthImage = distanceImage;
strokeWidthImage(~skeletonImage) = 0;
% Show the region image alongside the stroke width image.
figure
subplot(1,2,1);
imagesc(regionImage);
title('Region Image');

subplot(1,2,2);
imagesc(strokeWidthImage);
title('Stroke Width Image')

% Process the remaining regions
strokeWidthThreshold = th;
for j = 1:numel(mserStats)

    regionImage = mserStats(j).Image;
    regionImage = padarray(regionImage, [1 1], 0);

    distanceImage = bwdist(~regionImage);
    skeletonImage = bwmorph(regionImage, 'thin', inf);

    strokeWidthValues = distanceImage(skeletonImage);

    strokeWidthMetric = std(strokeWidthValues)/mean(strokeWidthValues);

    strokeWidthFilterIdx(j) = strokeWidthMetric > strokeWidthThreshold;

end

% Remove regions based on the stroke width variation
mserRegions(strokeWidthFilterIdx) = [];
mserStats(strokeWidthFilterIdx) = [];
```

```matlab
% Show remaining regions
figure;
imshow(I);
hold on
plot(mserRegions, 'showPixelList', true,'showEllipses',false);
title('After Removing Non-Text Regions Based On Stroke Width Variation');
hold off
% Get bounding boxes for all the regions
bboxes = vertcat(mserStats.BoundingBox);
% Convert from the [x y width height] bounding box format to the [xmin ymin
% xmax ymax] format for convenience.
xmin = bboxes(:,1);
ymin = bboxes(:,2);
xmax = xmin + bboxes(:,3) - 1;
ymax = ymin + bboxes(:,4) - 1;
% Expand the bounding boxes by a small amount.
expansionAmount = 0.01;
xmin = (1-expansionAmount) * xmin;
ymin = (1-expansionAmount) * ymin;
xmax = (1+expansionAmount) * xmax;
ymax = (1+expansionAmount) * ymax;
% Clip the bounding boxes to be within the image bounds
xmin = max(xmin, 1);
ymin = max(ymin, 1);
xmax = min(xmax, size(I,2));
ymax = min(ymax, size(I,1));
% Show the expanded bounding boxes
expandedBBoxes = [xmin ymin xmax-xmin+1 ymax-ymin+1];
IExpandedBBoxes =
insertShape(colorImage,'Rectangle',expandedBBoxes,'LineWidth',3);

figure
imshow(IExpandedBBoxes);
title('Expanded Bounding Boxes Text');
%Compute the overlap ratio
overlapRatio = bboxOverlapRatio(expandedBBoxes, expandedBBoxes);
% Set the overlap ratio between a bounding box and itself to zero to
% simplify the graph representation.
n = size(overlapRatio,1);
overlapRatio(1:n+1:n^2) = 0;
% Create the graph
g = graph(overlapRatio);
% Find the connected text regions within the graph
componentIndices = conncomp(g);
% Merge the boxes based on the minimum and maximum dimensions.
xmin = accumarray(componentIndices', xmin, [], @min);
ymin = accumarray(componentIndices', ymin, [], @min);
xmax = accumarray(componentIndices', xmax, [], @max);
ymax = accumarray(componentIndices', ymax, [], @max);

% Compose the merged bounding boxes using the [x y width height] format.
textBBoxes = [xmin ymin xmax-xmin+1 ymax-ymin+1];
% Remove bounding boxes that only contain one text region
numRegionsInGroup = histcounts(componentIndices);
textBBoxes(numRegionsInGroup == 1, :) = [];
% Show the final text detection result.
```

```
ITextRegion = insertShape(colorImage, 'Rectangle', textBBoxes,'LineWidth',3);

figure;
imshow(ITextRegion);
title('Detected Text');

%Using optical character recognition for recognizing the text.
ocrtxt = ocr(I, textBBoxes);
%Recognize text within an image
recognizedText = [ocrtxt.Text];
figure;
imshow(colorImage);
title('Recognized Text Part of The Image');
text(200, 100, recognizedText, 'BackgroundColor', [1 1 1]);
val =  numel(ocrtxt);
[ocrtxt.Text]
for n=1:val     %iterate speech part for all text in the photo

word = ocrtxt(n,1).Text;          %We are taking each word in a variable and
express it one after one
%Speech processing part
NET.addAssembly('System.Speech')
mysp = System.Speech.Synthesis.SpeechSynthesizer;    %We are using Matlab's
in built voice synthesizer for speech
mysp.Volume=100;                                %Volume of voice(Range : 1-100)
mysp.Rate=2;                              %Speed of voice (Range : -10 to 10 )
a = audiorecorder(96000,16,1);    % create object for recording audio
record(a,10);
Speak(mysp,word);                             %Expressing each word
b = getaudiodata(a);                          %store the recorded data in a numeric
array.
b = double(b);
figure
plot(b);
title('plot of the sound wave');
end
```

## References:

1. https://www.mathworks.com/help/vision/examples/automatically-detect-and-recognize-text-in-natural-images.html
2.https://www.mathworks.com/help/vision/ref/ocr.html?searchHighlight=ocr&s_tid=doc_srchtite
3.http://www.micc.unifi.it/delbimbo/wpcontent/uploads/2011/03/slide_corso/A34%20MSER.pdf
4. https://www.youtube.com/results?search_query=OCR+matlab
5. **International Journal of Advanced Trends in Computer Science and Engineering**, Vol.2 , *Special Issue of ICETEM 2013 - Held on 29-30 November, 2013 in Sree Visvesvaraya Institute of Technology and Science, Mahabubnagar – 204, AP, India*

6. **International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 5, Issue 1, January 2015)**
7. **International Journal of Electronics, Electrical and Computational SystemIJEECSISSN 2348-117XVolume 6, Issue 11November 2017**

**8.** *International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 7– No. 2, April 2014 – www.ijais.org* **Design and Implementation of Text To Speech Conversion for Visually Impaired People**