Bangladesh University of Engineering & technology

DEPARTMENT OF ELECTRICAL& ELECTRONIC ENGINEERING

EEE 212: NUMERICAL TECHNIQUE LABORATORY

MALAB PROJECT: TEXT TO SPEECH CONVERSION

LET MATLAB SCREAM

PROJECT SUBMITTED TO-

MD. MUKHLASUR RAHMAN TANVIR

LECTURER, BUET

SHOILIE CHAKMA

LECTURER, BUET

GROUP MEMBERS-

TASEEN AFRID 1606141)

ISHRAQ TASHDID (1606142)

RATUL KUNDU (1606147)

PRASUN DATA (1606148)

Table of Contents

List of illustration

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.

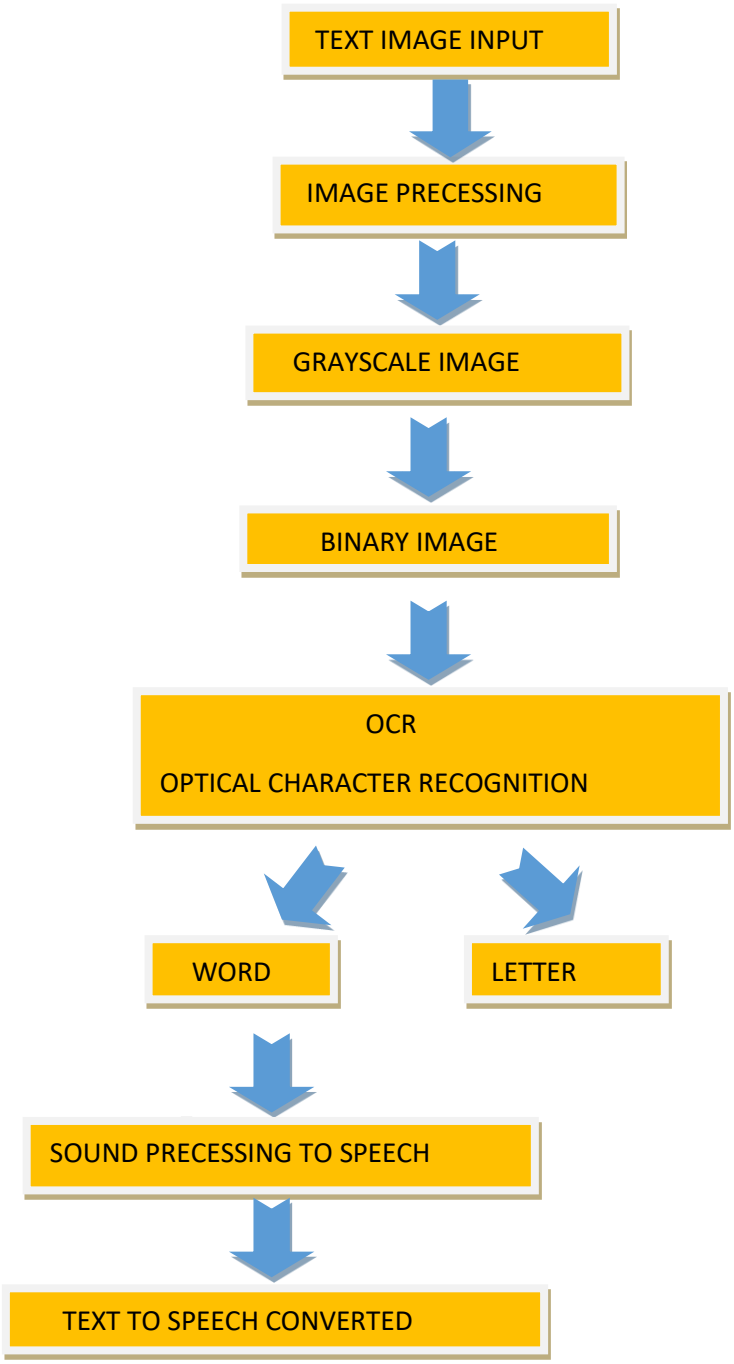
Introduction:

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.

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.

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. Segmentation is followed by feature extraction, which is concerned with the representation of the object. Feature extraction and classification are the heart of OCR.

TEXT TO SPEECH CONVERSION BLOCK DIAGRAM



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.
- 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.

Methodology:

The entire program is basically divided into 3 main parts.

- Image Processing Phase.
- Calling MATLAB's built 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.

Rgb2gray:

This method converts the RGB image to greyscale image.

Syntax:

I = rgb2gray(RGB);

Description:

I = rgb2gray(RGB) converts the true color 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.

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 imbinarization. The graythresh function uses Otsu's method, which chooses the threshold to minimize the intra-class variance of the black and white pixels.

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).

Ocr:

Recognize text using optical character recognition.

Syntax:

txt =OCR (I)

Description:

txt = OCR(I) returns an OCR text 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.

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.. For example:

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

The assembly name is System.Speech.

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

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.

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 audio-recorder 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.

Processing complex/natural input image:

Complex scenes are images that contain undetermined or random scenarios. This is different than structured scenes, which contain known scenarios where the position of text is known beforehand. Segmenting text from a complex image 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.

❖ Deleting non-text regions based on basic geometric properties

There are several geometric properties that are good for discriminating between text and non-text regions including

- Aspect ratio
- 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.

❖ 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.
- The output of Conn comp 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.
- Finally, before showing the final detection results, false text detections are suppressed by removing bounding boxes made up of just one text region.

Implementation, Testing & Results:

❖ Image Processing Phases: Input Image:

Fig1: Input Image

❖ Converting the RGB image to Grayscale image.

Fig2: The Grayscale Image

- Converting the grayscale image into binary one.

Fig 3: Binary Image

OCR Phase of Program:

- 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.

Fig 5: Recognized/Detected Text

- Presenting the bounding boxes of words with recognition confidences.

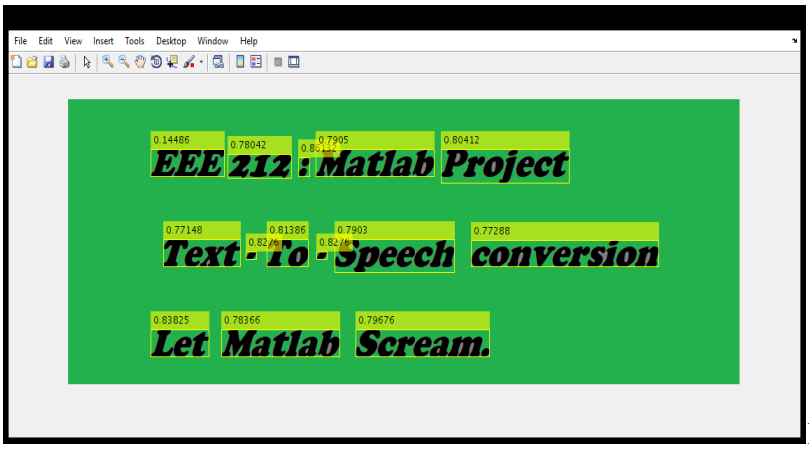


Fig 6: Bounding Boxes of Words

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.

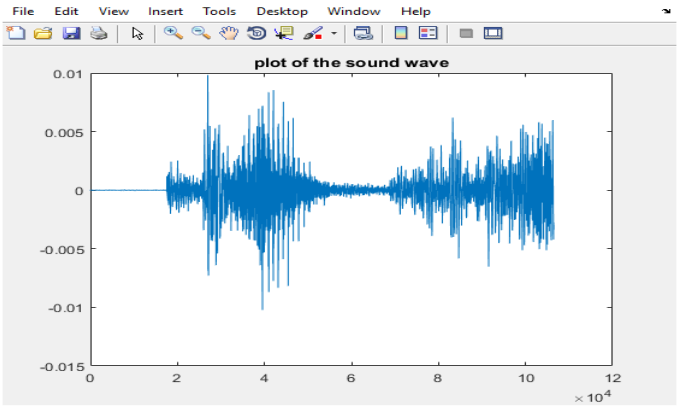


Fig 8: Plot of speech 'Life'

- 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.

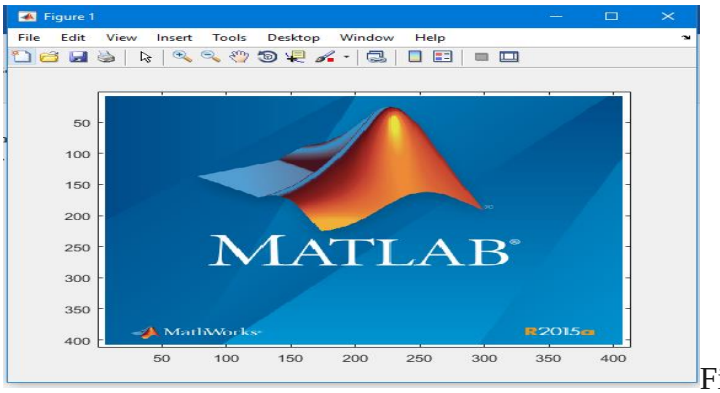


Fig 9: Input Image

- Detecting text regions using MSER.

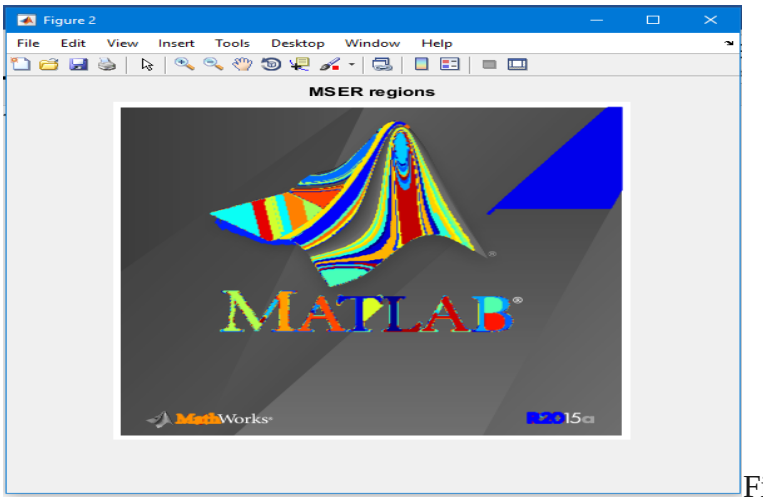


Fig 10: Highlighting MSER regions

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

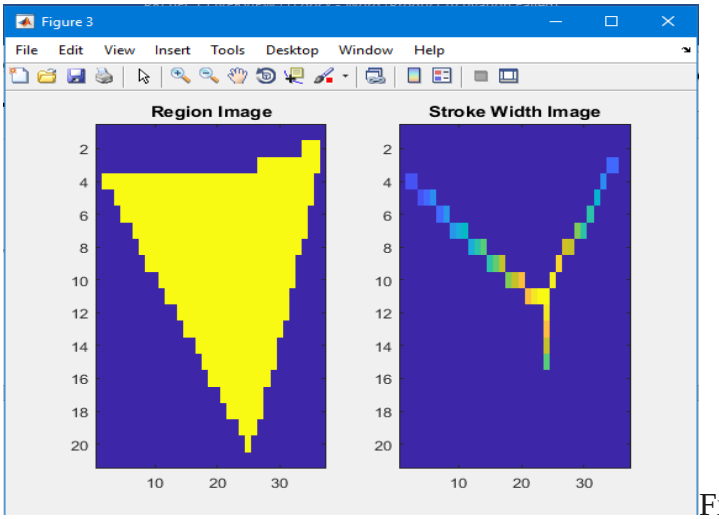


Fig 11: Stroke-width regions

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

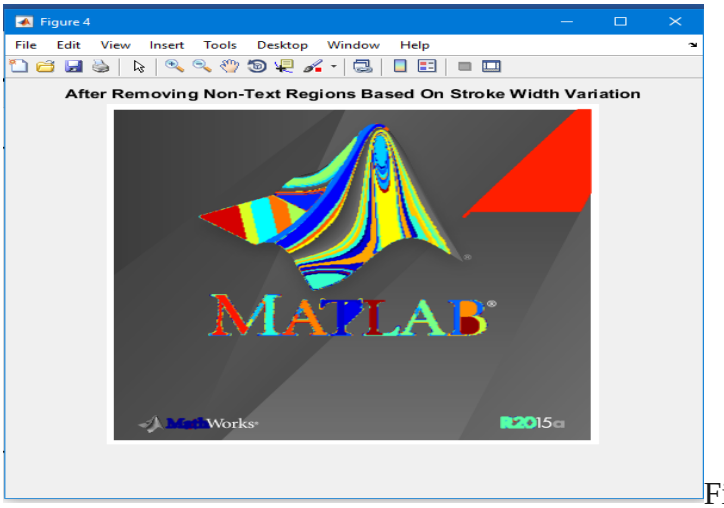


Fig 12: Removing Non-text regions.

❖ Merging text regions for final detection result.

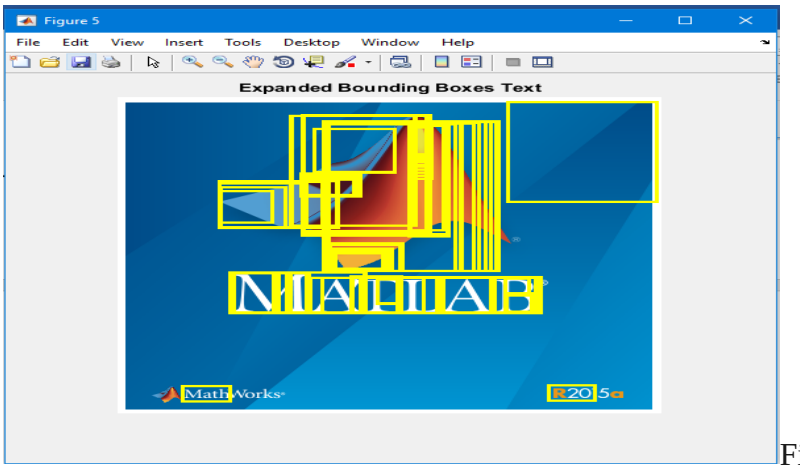


Fig 13: Bounding Text Box

❖ Final detection of the text in the image.

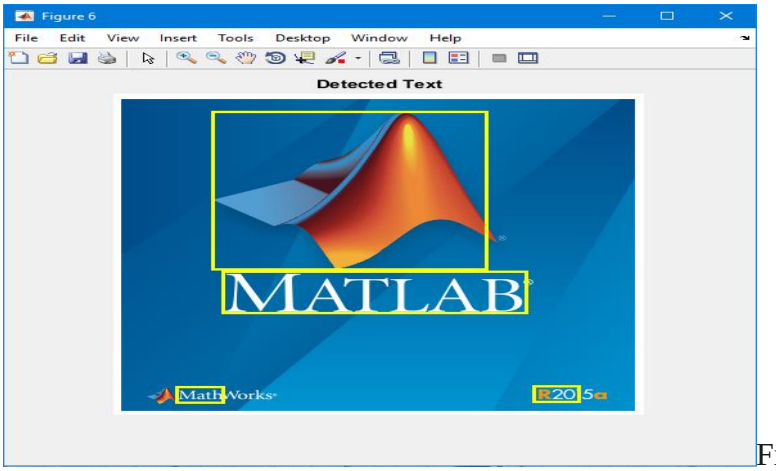


Fig 14: Detected Texts

OCR phase of the program:

❖ Implementing MATLAB's built-in 'OCR' function.

- Highlighting the recognized text part of the image.

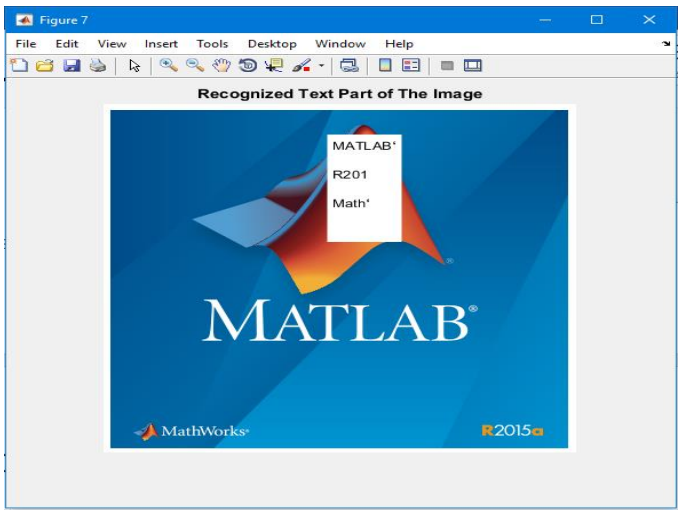
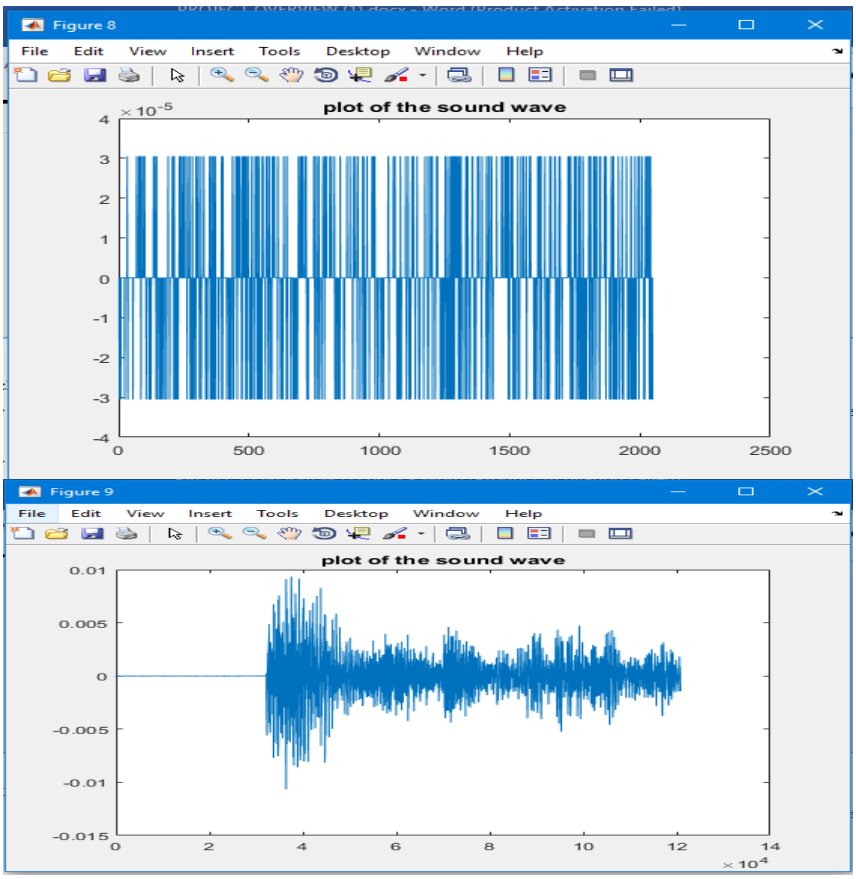


Fig 15: Recognized Texts in White Box

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.



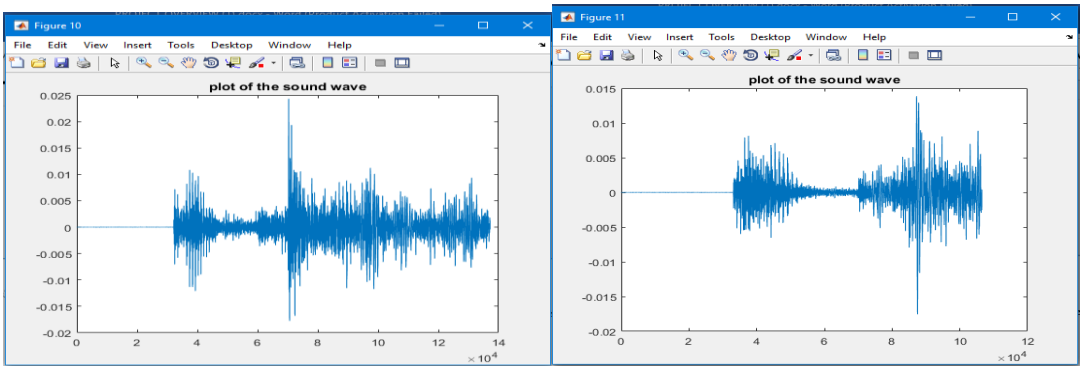


Fig 16: Plot of Emerging Sound Waves.

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.
- To check non-uniform lighting issues we have used binarization. If the characters are not visible in the results of binarization, it indicates a potential non-uniform lighting issues.
- The first task faced by any TTS system is the conversion of input text into linguistic representation, usually called text-to-phonetic conversion.
- To make the speech soothable to listen we have made necessary adjustments in the volume and rate.

Applications for the Blind:

Probably the most important and useful application field in speech synthesis is the reading and communication aids for the blind.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. 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 bold or 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.

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. To overcome this problem, we can think of a method which recognizes the delay and automatically remove it.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. Recognition from poor quality documents results in a number of recognition errors. Words are not collected correctly from the images.

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) = [];

% 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 .**