Mini-Project: Optical Character Recognition

By,

Team Data Pirates

Ashok Aravind S N – 1BG18IS007

Srinivasan R – 1BG18EC110

Ramanand Kulkarni – 1BG18EC123

INTRODUCTION:

Text recognition is one of the evolving techniques which plays a pivotal role in differentiating printed text as well handwritten text in documents. It is sometimes referred to as Optical Character Recognition. There are various applications wherein the concept of OCR can be implemented in order to get successful results.

ALGORITHM:

The techniques or steps used in OCR in order to yield accurate results in Text Detection are:

* Detect text with MSER regions:

• MSER regions are connected areas characterized by almost uniform intensity, surrounded by contrasting background.

• The selected regions are those that maintain unchanged shapes over a large set of thresholds.

• The selected regions are those that maintain unchanged shapes over a large set of thresholds. The MSER regions works well for finding out the text regions its because of the colour and contrast difference which helps in stabilising intensity levels of the text

* Removing non text regions based on their given geometric properties:

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

* There are several geometric properties that are good for discriminating between text and non-text regions including:
* Aspect ratio
* Eccentricity
* Euler number
* Extent
* Solidity
* Remove non text regions based on 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

* Merge text for 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 neighbouring text regions and then form a bounding box around these regions. To find neighbouring regions, expand the bounding boxes computed earlier with regionprops. This makes the bounding boxes of neighbouring text regions overlap such that text regions that are part of the same word or text line form a chain of overlapping bounding boxes.
* Recognize detected text using OCR
* After detecting the text regions, use the OCR function to recognize the text within each bounding box. Note that without first finding the text regions, the output of the OCR function would be considerably more noisy.
* Convert the text to audio using speech synthesizer.

CODE SNIPPETS:

OCR:

global path;

colorImage = imread(path);

I = rgb2gray(colorImage);

% Detect MSER regions.

[mserRegions, mserConnComp] = detectMSERFeatures(I, 'RegionAreaRange',[200 8000],'ThresholdDelta',4);

% Use regionprops to measure MSER properties

mserStats = regionprops(mserConnComp, 'BoundingBox', 'Eccentricity', 'Solidity', 'Extent', 'Euler', 'Image');

% Compute the aspect ratio using bounding box data.

bbox = vertcat(mserStats.BoundingBox);

w = bbox(:,3);

h = bbox(:,4);

aspectRatio = w./h;

% Threshold the data to determine which regions to remove. These thresholds

% may need to be tuned for other images.

filterIdx = aspectRatio' > 3;

filterIdx = filterIdx | [mserStats.Eccentricity] > .995 ;

filterIdx = filterIdx | [mserStats.Solidity] < .3;

filterIdx = filterIdx | [mserStats.Extent] < 0.2 | [mserStats.Extent] > 0.9;

filterIdx = filterIdx | [mserStats.EulerNumber] < -4;

% Remove regions

mserStats(filterIdx) = [];

% 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);

% Compute the stroke width variation metric

strokeWidthValues = distanceImage(skeletonImage);

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

% Threshold the stroke width variation metric

strokeWidthThreshold = 0.4;

strokeWidthFilterIdx = strokeWidthMetric > strokeWidthThreshold;

% Process the remaining regions

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

mserStats(strokeWidthFilterIdx) = [];

% 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 = 0.02;

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));

% Compute the overlap ratio

expandedBBoxes = [xmin ymin xmax-xmin+1 ymax-ymin+1];

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, :) = [];

% Display text

ocrtxt = ocr(I, textBBoxes);

global sentence;

sentence = strcat(ocrtxt.Text);

app.TextArea.Value = sentence;

TEXT TO SPEECH:

global sentence;

sentence = app.TextArea.Value;

try

NET.addAssembly('System.Speech');

Speaker = System.Speech.Synthesis.SpeechSynthesizer;

if ~isa(sentence,'cell')

sentence = {sentence};

end

for n=1:length(sentence)

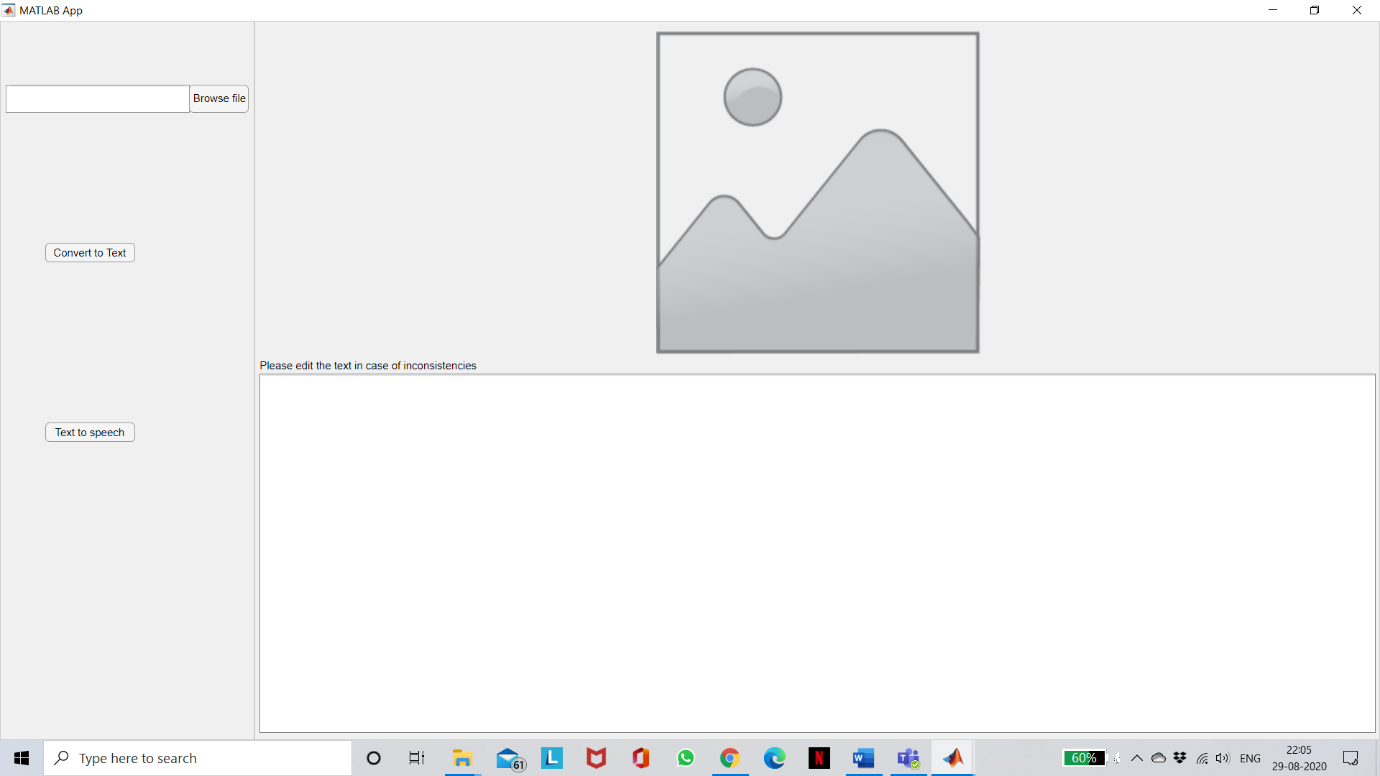
Speaker.Speak (sentence{n});

end

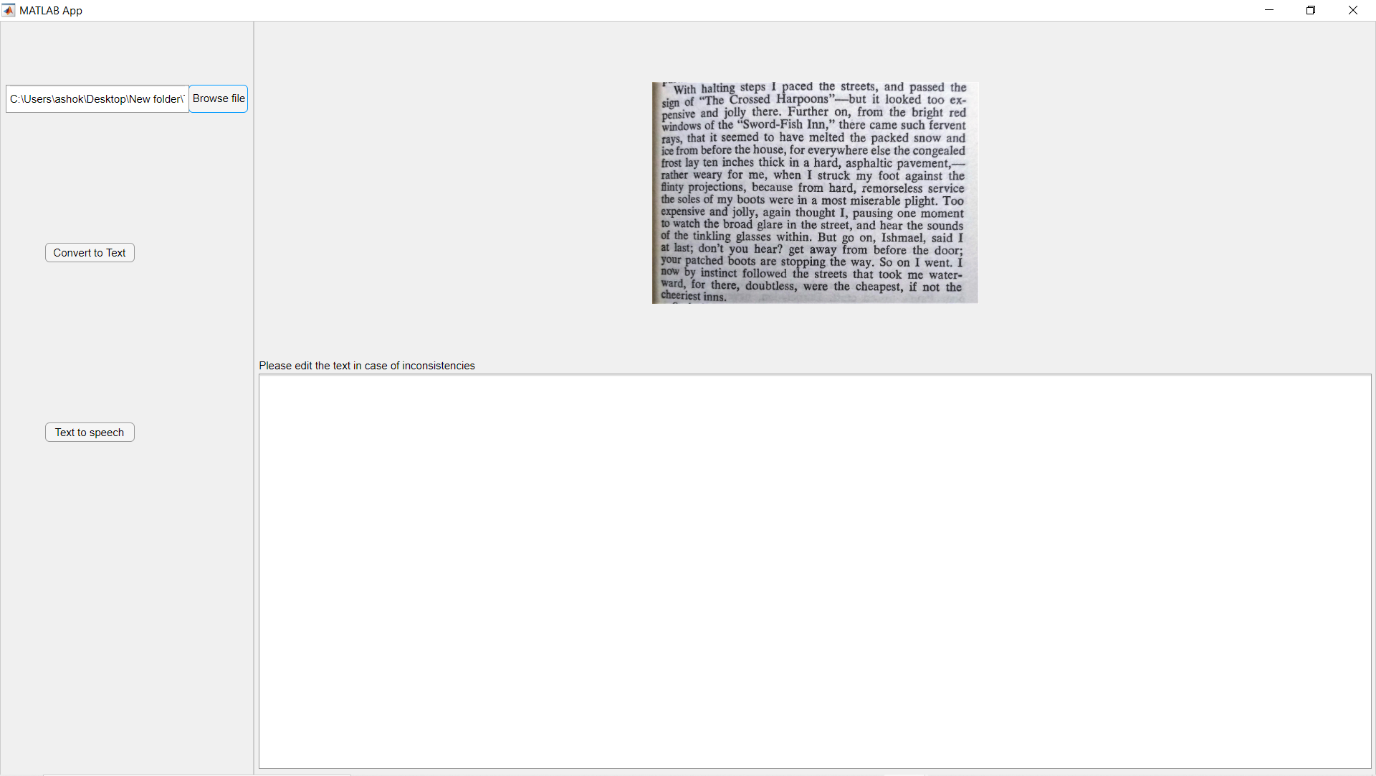
end

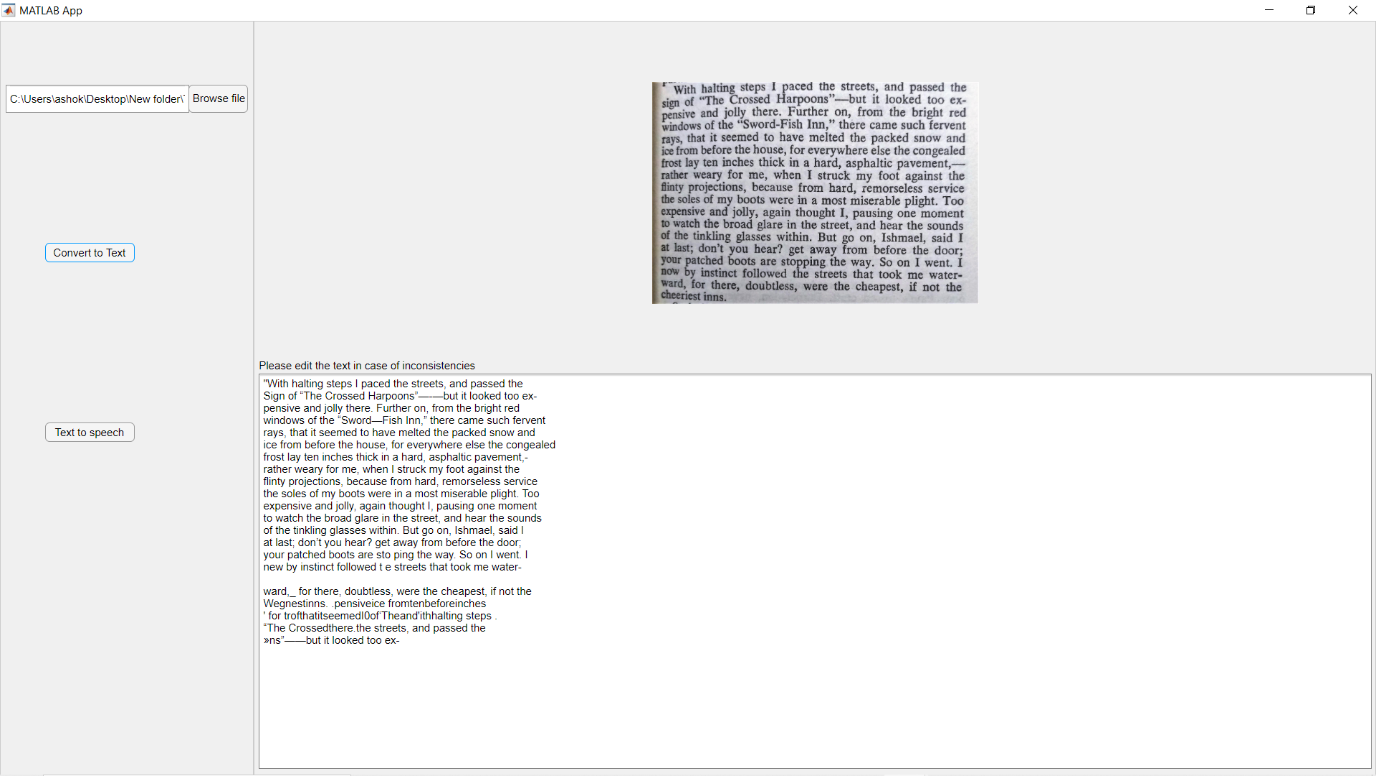
RESULTS:

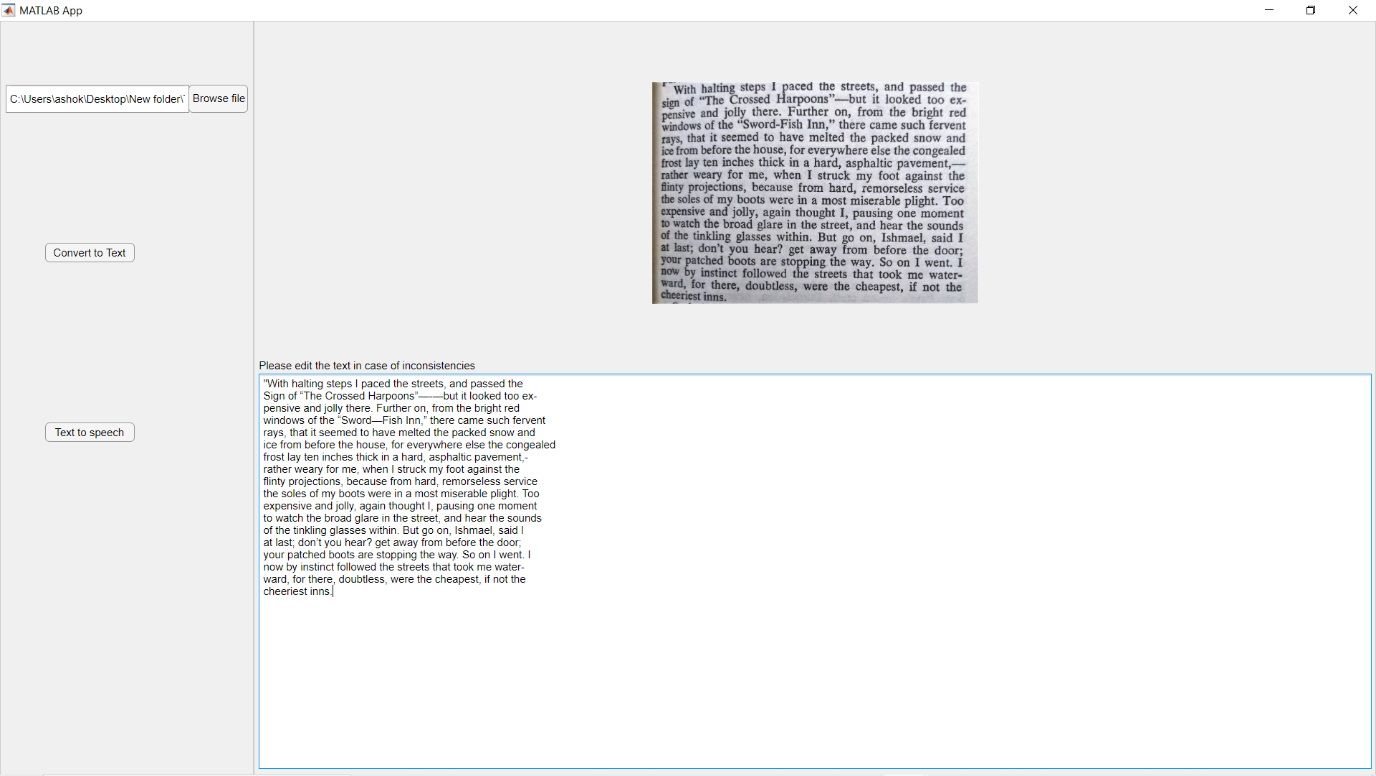
The app is opened



The Image being loaded



The Image is processed by the ocr and the text is displayed

The text is corrected by the user

Speech is synthesized from the text