TEXT RECOGNITION ANDROID APP FOR LOW-VISION

Ashwani Kumar
(17JE003253)

Shubham Mishra
(17JE003230)

Under the supervision of

Dr Haider Banka

IIT ISM Dhanbad

Acknowledgement

Our sincere thanks go to our mentor, Dr Haider Banka, who has guided us throughout this project's stages. His expertise in Machine Learning and Computer Vision and his work experience in these domains have significantly improved the quality of our final report.
We would also like to thank our college for giving us this chance to complete our project and learn many new things.

Abstract

Blindness makes life complicated for those who suffer from it, but technology will assist with some day-to-day activities. The current project focuses on the creation of a photo-to-speech application for the blind in this context. Camera Reading for Blind People is the name of the project, and it aims to create a smartphone application that enables a blind person to “read” text (a sheet of paper, a signal, etc.).To accomplish this, a series of frameworks for Optical Character Recognition (OCR) and Text to Speech Synthesis (TTS) have been integrated, allowing a user to take a picture with a smartphone and hear the text that appears in the image.

Table of Contents

1. Introduction

People who are blind are unable to perform visual tasks. Text reading, for example, necessitates the use of a braille reading machine or a digital speech synthesiser (if the text is available in digital format). The majority of printed works released do not contain braille or audio copies, and digital versions are also in the minority. Blind people, on the other hand, are unable to read basic signs written on walls or signals that surround us. As a result, developing a mobile application that can convert images to expression, whether written on a wall, a sheet of paper, or another support, has a lot of potential and usefulness..Optical character recognition (OCR) technology allows for the recognition of text from image images. This technology has been commonly used to turn scanned or photographed documents into electronic copies that can be edited, searched, played, and easily carried. A text in digital format can be synthesised into a human voice and played via an audio device using speech synthesis technology (TTS). TTS aims for a native speaker of the language to automatically convert sentences into spoken discourse in a natural language that is similar to the said form of the same text. Over the last decade, this technology has advanced significantly, with several devices now capable of producing synthetic speech that is very similar to the natural voice. Speech synthesis research has increased as a result of its growing relevance in a variety of new applications.This paper describes the process of creating a prototype of an iPhone/iPod Touch/iPad mobile application (iOS) that enables a blind user to use the system camera to get a reading of the existing written text in the captured image. The system uses existing OCR and TTS systems, integrating them in such a way that they can produce the desired results when used together.

2. Related Work

This part presents an introduction to OCR and TTS technologies used in this project and studies and work within the theme of this project.

2.1.Optical Character Recognition (OCR) overview

Optical character recognition, abbreviated as OCR, is the method of automatically recognising and converting existing characters in a written-support image into text format, which can then be used in a variety of applications.The OCR has been extensively researched and has shown significant improvements in terms of efficiency and accuracy of the obtained data. Optical character recognition can be described as a series of steps that must be followed.

- Optical image acquisition
- Location and segmentation
- Preprocessing
- Feature extraction
- Classification
- Post Processing

The final text is then converted into the desired document format (rtf, txt, pdf, etc.).

2.2. Text-to-Speech (TTS) overview

TTS (acronym for Text-To-Speech) is a computer device that can read aloud any text, regardless of its origin.The aim of TTS is to artificially generate human speech. To generate an intelligible and natural result, voice synthesis is a complex process requiring complex algorithms.Natural language processing methods are used in TTS synthesis. Since the text to be synthesised is the system's first entry, it must also be the first to be processed. A synthesised voice can be created using a variety of methods.

- Articulatory synthesis
- Formant synthesis
- Concatenation synthesis
- Hidden Markov models synthesis

The approaches used in the research and production of speech synthesis systems are the key synthesis techniques presented above. However, using a combination of the different techniques in the production of future systems speech synthesis is one way to benefit from the inherent advantages of each technique.

Naturalness and intelligibility are two factors that decide the efficiency of a speech synthesis. Naturalness refers to how similar the sound produced by TTS is to that of

a human voice, while intelligibility refers to how well the sound is understood in complex situations.

2.3. State of Art

Within the image-to-speech research sector, some work is being done.
The Phone Reader thesis describes an approach to a method that uses the Tesseract OCR to recognise text in a photo, and then reads the text using TTS in the selected language. The OCR processing and translation are done on an external server, which necessitates a data link between the server and the client for communication.

In the paper Optical Character Recognition, an application is provided that uses OCR Tesseract to recognise existing text in images and then performs speech synthesis of the recognised text. This application entails Since there will be no preprocessing or identification of the image limits, all processing will be performed locally on the mobile device, and these will be identified as future work.

The paper Open Source OCR Framework Using Mobile Device defines a project in which an image is captured and converted to a bitmap format using the device's camera. A single-channel intensity image is generated from a multi-channel RGB image. Then it performed post-processing, which included removing the noise caused by OCR, creating an ASCII code for the recognised text, and removing both alphabetical and numerical characters with no filter. This is mentioned as future work, which will include fine-tuning the findings using OCR cross-referencing with the dictionary, as well as other high-level semantic techniques.

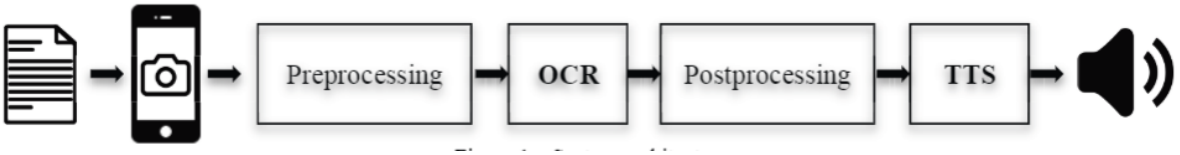
The application of a canning method for obtaining text stored in photographs is explored in OCRdroid: A Framework to Digitize Text Using Mobile Phones. This is a project whose aim is to create an Android application that preprocesses images as they are taken, correcting existing lighting and ensuring that the images are correctly placed for better text recognition, and then sends the obtained image to an external server that processes the image recognition OCR. In this region, there are some public mobile applications that can recognise text using OCR and read it using the system's internal TTS.

Despite the existence of studies and applications that address the theme proposed in this project, it appears that existing solutions continue to have some limitations, which are related to recognition performance, especially in situations where shooting conditions are less than ideal. This field of research also needs to mature in order to address the limitations of real-world circumstances where light conditions, concentration, orientation, or the presence of shadows prevent OCR from effectively recognising images.

As a result, the project Text Recognition for Blind People makes a valuable contribution to the field by providing additional knowledge and laying the groundwork for the creation of similar Android applications.

3. Text Generation Model

3.1 Model Overview



3.2 Preprocessing

The steps in a typical machine learning OCR pipeline are as follows::

1. Remove the noise from the image
2. Remove the complex background from the image
3. Handle the different lightning condition in the image

These are the basic methods for preprocessing images in a computer vision mission, and they can all be accomplished with OpenCV functions.

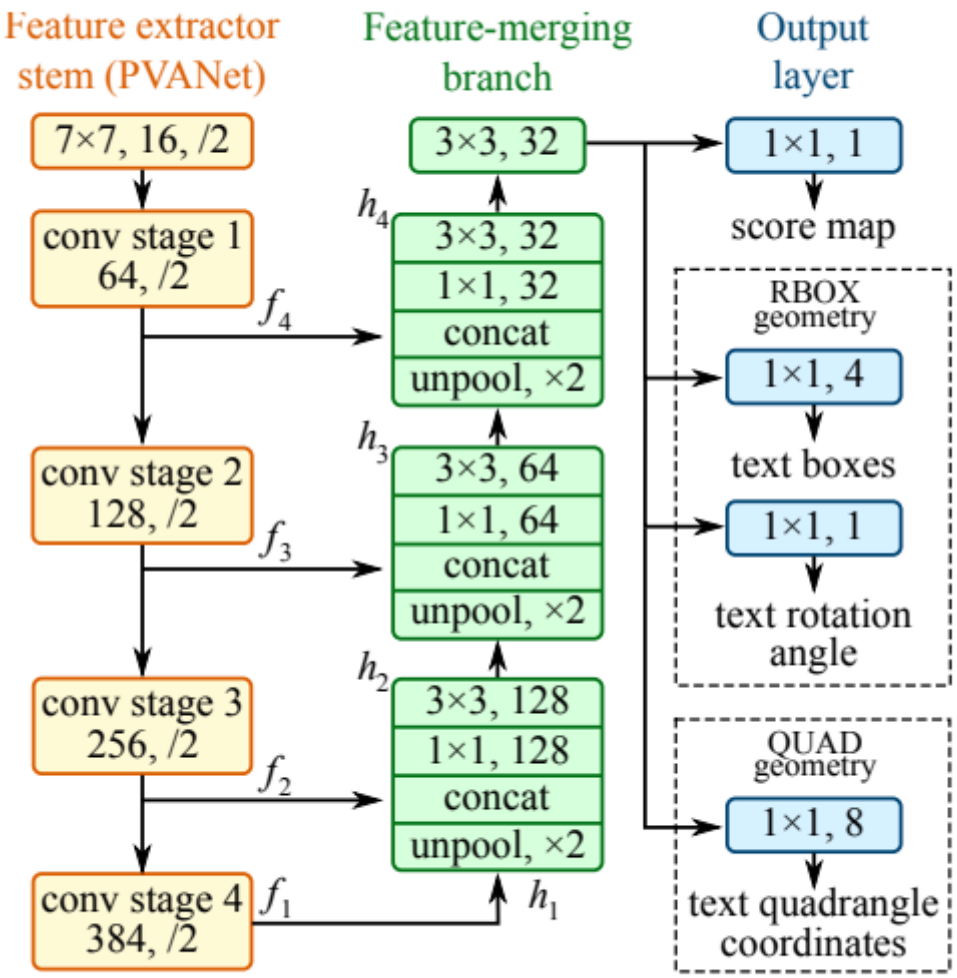
3.3 Text Detection

EAST (Efficient accurate scene text detector)

Based on this paper, we've developed a very robust deep learning method for text detection. Although it is only a text detection tool, it is worth noting. It can locate bounding boxes that are both horizontal and rotated. Any text recognition tool can be used in conjunction with it.

This paper's text detection pipeline has no redundant or intermediate steps and only has two phases.

To directly generate word or text-line level prediction, one uses a completely convolutional network. The final output is generated by processing the produced predictions, which could be rotated rectangles or quadrangles, through the non-maximum-suppression stage.



EAST can recognise text in both images and video. It runs near real-time at 13FPS on 720p images with high text detection accuracy, as stated in the paper. Another advantage of this strategy is that it can be implemented in OpenCV 3.4.2 and OpenCV 4. This EAST model, as well as text recognition, will be demonstrated.

Dataset:

EAST has been trained on ICDAR 2015, COCO-Text and MSRA-TD500. On the ICDAR 2015 dataset, the proposed algorithm achieves an F-score of 0.7820 at 13 fps at 720p resolution.

3.4 Text Recognition

The next move is to identify text after we've detected the bounding boxes with text. The text can be recognised using a variety of methods. In the following section, we'll go over some of the most effective techniques.

Machine Learning OCR with Tesseract

Tesseract was developed between 1985 and 1994 at Hewlett-Packard Laboratories. HP made it open-source in 2005. According to Wikipedia, Tesseract was one of the most accurate open-source OCR engines available in 2006.

The Tesseract's capabilities were mainly limited to organised text data. It will struggle to work well in unstructured text with a lot of noise. Since 2006, Google has been funding tesseract research and development.

Unstructured data is best handled by deep-learning-based approaches. Tesseract 4 introduced deep-learning capabilities with an LSTM network (a form of Recurrent Neural Network) based OCR engine that focuses on line recognition but also supports the legacy Tesseract OCR engine from Tesseract 3, which recognises character patterns. On July 7, 2019, the new stable version 4.1.0 was announced. This edition is also much more precise when it comes to unstructured text.

We'll use some of the images to demonstrate both EAST text detection and Tesseract 4 text recognition. In the following code, we'll see text detection and recognition in motion. The information in this article proved to be very useful in writing the code for this project.

3.5 Code

We'll use some of the images to demonstrate both EAST text detection and Tesseract 4 text recognition. In the following code, we'll see text detection and recognition in motion.

```
#The requisite packages are being loaded.
import numpy as np
```

```
import cv2
from imutils.object_detection import non_max_suppression
import pytesseract
from matplotlib import pyplot as plt


#Creating an argument dictionary to hold the code's default
arguments.
args                                                    =
{"image":"../input/text-detection/example-images/Example-im
ages/ex24.jpg",
"east":"../input/text-detection/east_text_detection.pb",
"min_confidence":0.5, "width":320, "height":320}
```

Creating argument dictionary with some default values

Here, I am working with essential packages. OpenCV package uses the EAST model for text detection. The tesseract package is for recognizing text in the bounding box detected for the text. Created a dictionary for the default arguments needed in the code. Let's see what these arguments mean.

- *image: The location of the input image for text detection & recognition.*
- *east: The location of the file having the pre-trained EAST detector model.*
- *min-confidence: Min probability score for the confidence of the geometry shape predicted at the location.*
- *width: Image width should be multiple of 32 for the EAST model to work well.*
- *height: Image height should be multiple of 32 for the EAST model to work well.*

```
#Give location of the image to be read.
#"Example-images/ex24.jpg" image is being loaded here.

args['image']="../input/text-detection/example-images/Examp
le-images/ex24.jpg"
image = cv2.imread(args['image'])

#Saving a original image and shape
orig = image.copy()
(origH, origW) = image.shape[:2]

# set the new height and width to default 320 by using args
#dictionary.
(newW, newH) = (args["width"], args["height"])

#Calculate the ratio between original and new image for
both height and weight.
#This ratio will be used to translate bounding box location
```

```
    on the original image.
    rW = origW / float(newW)
    rH = origH / float(newH)

    # resize the original image to new dimensions
    image = cv2.resize(image, (newW, newH))
    (H, W) = image.shape[:2]

    # construct a blob from the image to forward pass it to
    EAST model
    blob = cv2.dnn.blobFromImage(image, 1.0, (W, H),
(123.68, 116.78, 103.94), swapRB=True, crop=False)
```

Image processing

```
    # load the pre-trained EAST model for text detection
    net = cv2.dnn.readNet(args["east"])

    # We would like to get two outputs from the EAST model.
    #1. Probability scores for the region whether that contains
    text or not.
    #2. Geometry of the text -- Coordinates of the bounding box
    detecting a text
    # The following two layer need to be pulled from the EAST
    model for achieving this.
    layerNames = [
       "feature_fusion/Conv_7/Sigmoid",


"feature_fusion/concat_3"]
```

Loading Pre-trained EAST model and defining output layers

```
    #Forward pass the blob from the image to get the desired
    output layers
    net.setInput(blob)
(scores, geometry) = net.forward(layerNames)
```

Forward pass the image through EAST model

```
    ## Returns a bounding box and probability score if it is
    more than minimum confidence
    def predictions(prob_score, geo):
       (numR, numC) = prob_score.shape[2:4]
       boxes = []
       confidence_val = []

       # loop over rows
       for y in range(0, numR):
            scoresData = prob_score[0, 0, y]
            x0 = geo[0, 0, y]
            x1 = geo[0, 1, y]
            x2 = geo[0, 2, y]
```

```
                x3 = geo[0, 3, y]
                anglesData = geo[0, 4, y]

                # loop over the number of columns
                for i in range(0, numC):
                        if scoresData[i] < args["min_confidence"]:
                                continue

                        (offX, offY) = (i * 4.0, y * 4.0)

                        # calculating the sine and cosine of the
        rotation angle for the forecast
                        angle = anglesData[i]
                        cos = np.cos(angle)
                        sin = np.sin(angle)

                        # Using the geo volume to determine the
        bounding box's dimensions
                        h = x0[i] + x2[i]
                        w = x1[i] + x3[i]

                        # Calculate the text prediction bounding
        box's beginning and ending points.
                        endX = int(offX + (cos * x1[i]) + (sin *
        x2[i]))
                        endY = int(offY - (sin * x1[i]) + (cos *
        x2[i]))
                        startX = int(endX - w)
                        startY = int(endY - h)

                        boxes.append((startX, startY, endX, endY))
                        confidence_val.append(scoresData[i])

            # return bounding boxes and associated confidence_val
            return (boxes, confidence_val)
```

EAST model prediction bounding box decoding feature

Only horizontal bounding boxes are decoded in this exercise. It's more difficult to decode revolving bounding boxes from scores and geometry.

```
        # Find predictions and  apply non-maxima suppression
        (boxes, confidence_val) = predictions(scores, geometry)
boxes = non_max_suppression(np.array(boxes), probs=confidence_val)
```

After non-max suppression, obtaining final bounding boxes

After applying non-max-suppression, we can now derive the bounding boxes. We'd like to see the image's bounding boxes and learn how to extract text from the observed bounding boxes. The tesseract is used to do this.

```
##Text Detection and Recognition

# create the results list from scratch
results = []

# To find the coordinates of bounding boxes, loop over the
bounding boxes.
for (startX, startY, endX, endY) in boxes:
   # To represent the bounding box on the original picture,
scale the coordinates based on the respective ratios.
   startX = int(startX * rW)
   startY = int(startY * rH)
   endX = int(endX * rW)
   endY = int(endY * rH)

   #Identify the area of concern
   r = orig[startY:endY, startX:endX]

   #To convert an image to a string, change the
configuration setting.
   configuration = ("-l eng --oem 1 --psm 8")
       #This will recognize the text from the image of
bounding box
   text          =          pytesseract.image_to_string(r,
config=configuration)

   # to the list of data, append bounding box coordinates
and related text
results.append(((startX, startY, endX, endY), text))
```

Creating a list of bounding box coordinates and text that can be recognised in the boxes

The bounding box coordinates and related text are stored in a list in the code above. We'll take a look at how it appears in the picture.

In our case, we used a particular tesseract configuration. There are a variety of tesseract configuration options available.

3.6 Results and Examples

Text detection is handled by the OpenCV EAST model, and text recognition is handled by the tesseract. The Tesseract's PSM has been adjusted to match the picture. It's worth noting that Tesseract usually needs a clear picture to function properly.

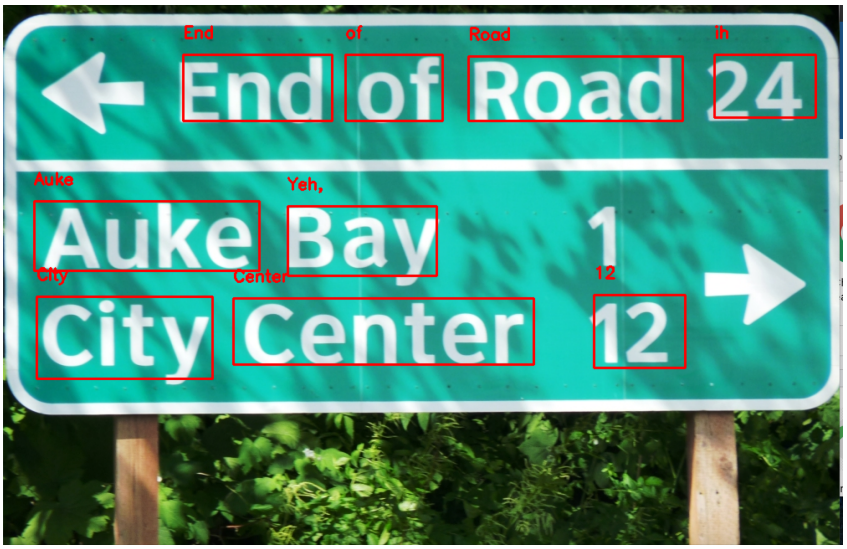
Owing to the difficulty of implementing rotating bounding boxes, we did not include it in our current implementation.

However, in a real-world situation where the text is rotated, the above code would fail. In addition, if the picture is not quite simple, the tesseract will have trouble correctly recognising the text.

Some of the output generated through the above code are:

The code could deliver excellent results for all the above images. The text is clear and the background behind the text is also uniform in these images. The model performed pretty well here. But some of the alphabets are not recognized correctly. You can see that bounding boxes are mostly correct as they should be. Maybe a slight rotation would help. But our current implementation does not provide rotating bounding boxes. It seems due to image

clarity, tesseract could not recognize it perfectly.



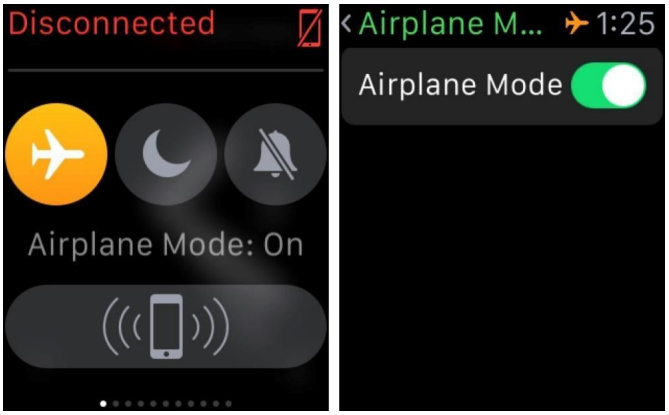
The model performed pretty decently here. But some of the texts in bounding boxes are not recognized correctly. Numeric 1 could not be detected at all. There is a non-uniform background here, maybe generating a uniform background would have helped this case. Also, 24 is not properly bound in the box. In such a case, padding the bounding box could help.

It seems that a stylized font with shadow in the background has affected the result in the above case.

We can not expect the OCR model to be 100 % accurate. Still, we have achieved good results with the EAST model and Tesseract. Adding more filters for processing the image would help in improving the performance of the model.

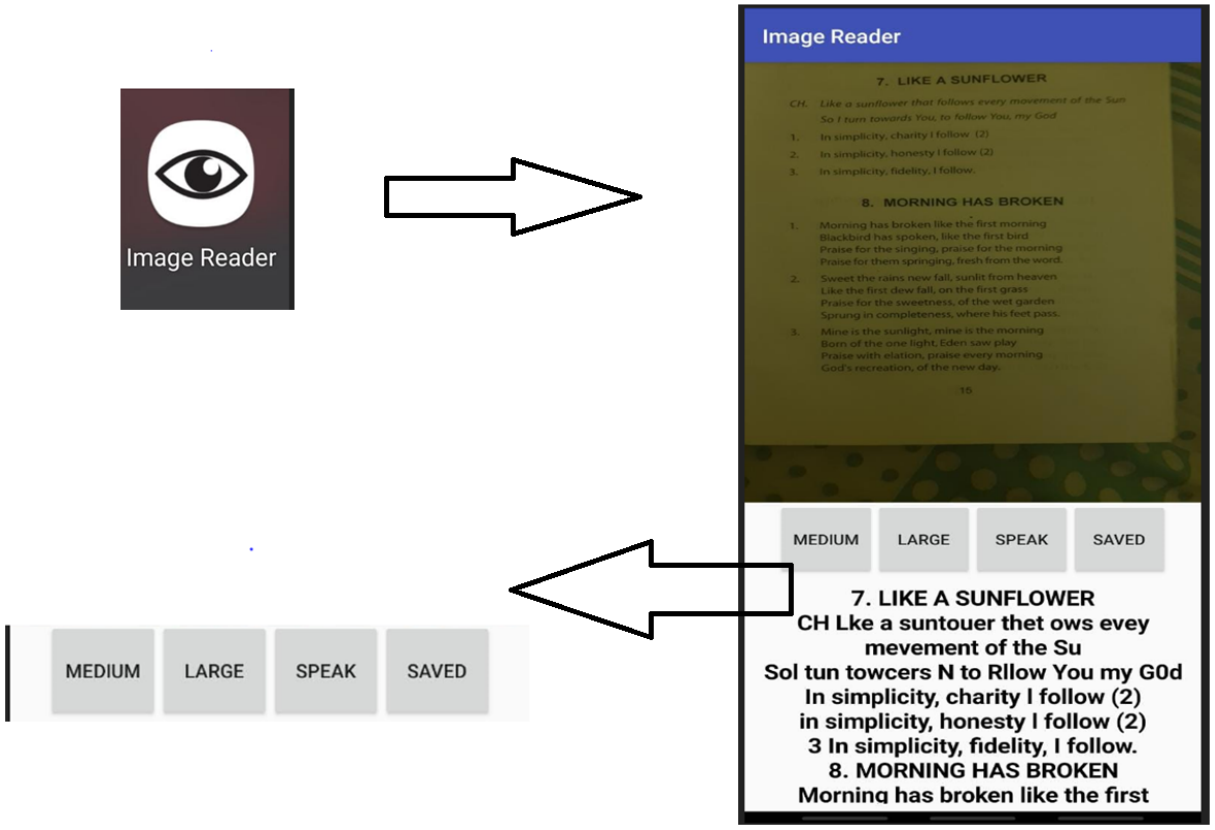
4. Working of Android

The blind man/low vision person can point the camera to surroundings and using our saved trained models we identify all the textual data present in the view. The android mobile then speaks out aloud the caption generated and the person will be able to hear the text . An important feature of our app is that it can work in airplane/ offline mode which is very useful in low internet bandwidth areas and where there is no signal.



Internal Working of Android App

The main activity of the android app has 4 buttons.

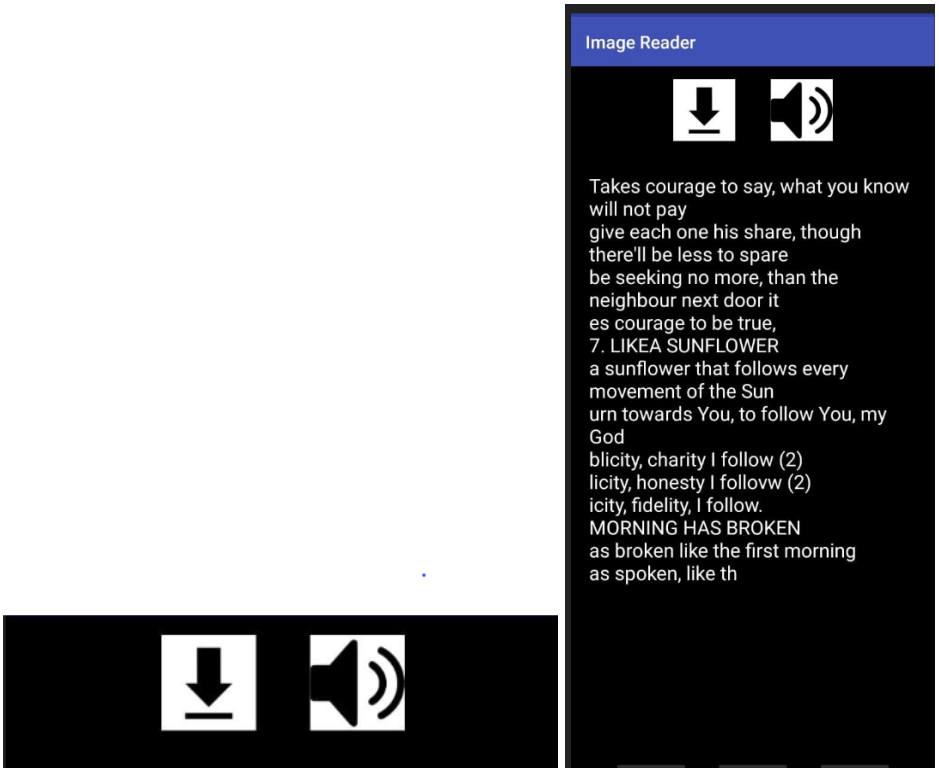


When you point the camera towards the scene you want to extract text from. The text content displays at the bottom of the screen. The 4 button provides additional features.

The first button (Medium) can be used to increase the font size to a slightly greater size.

The second button(large) can be used to see the text in a large size and can specially be helpful to low vision people.

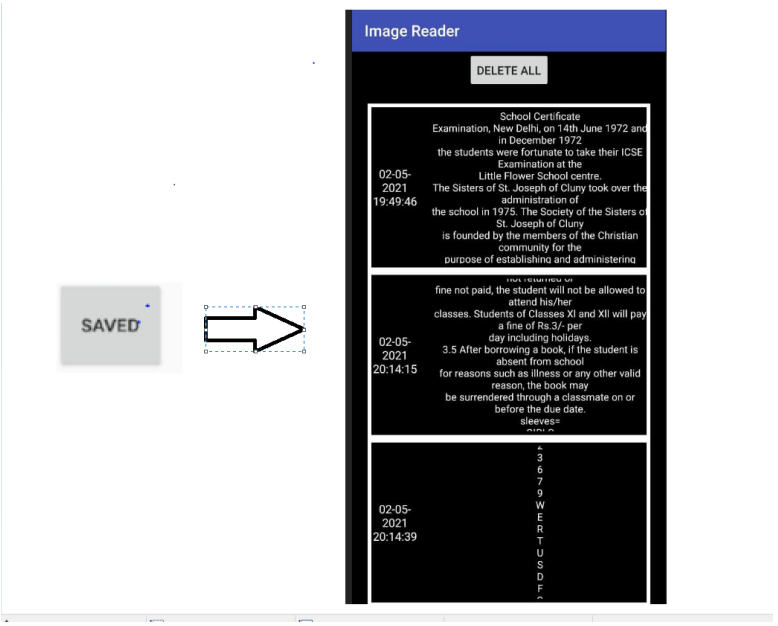
On clicking on the third button (Speak) opens a new Screen with the current identified text from the scenery/ document. This screen has two buttons .



The left button is for saving the identified text for future reference. We save the text in a sql database of Android. On successfully saving text a message appears “Saved”.

The right button is for Speaking out the identified text. On clicking on this button the Android speaks out the text using the inbuilt TTS engine of Android Smart phones.

Clicking on the fourth button (Saved) opens a list of items that was previously saved in a very optimised representation of list in Android known as Recycler View. This screen contains the data and time along with the saved text in a table format.



1

5. Conclusion

Our Android app is capable of viewing an image and extracting textual information from it. Our app is also capable of speaking out this text and also saving the text for future reference. Moreover all this functionality can work even in offline mode. Hence, our app can be of great help to the blind or low vision people to identify small text in any document or any far text in a distant view.

6. Future Prospects

Our app can be put to great use for the visually impaired. The model proposed integrated with an android or ios application works as a real-time scene text viewer.

This can be a really efficient solution for helping the visually impared.

7. References

[1] "Extracting text from images using OCR on Android". June 27, 2015. Archived from the original on March 15, 2016.

[2] Tappert, C. C.; Suen, C. Y.; Wakahara, T. (1990). "The state of the art in online handwriting recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

1