Real Time Text Detection Using Mobile Phones

Himanshu Gupta (B.Tech-13085028), Shantanu Agrawal (B.Tech-13085080), Shikhar Trivedi (B.Tech-13085055)
Department of Electrical Engineering
Indian Institute of Technology (BHU)
Varanasi, India
himanshu.gupta.eee13@itbhu.ac.in, shantanu.agrawal.eee13@itbhu.ac.in, shikhar.trivedi.eee13@itbhu.ac.in

***Abstract*— The intelligent analysis of text images is in wide demand now days. Text search is one of the important aspect of these analysis. In today's world, no one wants to wait, and seek for fast information extraction. Here, this report presents the work on Text Search in real time. The current state of art is to first store down the image and then output the search result after processing in OCR. This takes unnecessary storing of image and time for the same. We have developed an Android-platform based text searching application that is able to recognize the text, search the text string and display the bounding box around the text onto the screen of mobile in real time. In this report, we demonstrate the system flow, the text detection algorithm and detailed experiment result.**

***Keywords*—Text Detection, Text Search, Android, Tesseract OCR, OpenCV.**

I. Introduction

We always feel the need to search a word or phrase in a document, book, novel etc and wish that there should be a Ctrl+F(like In Windows) function to just type the word and search for it in real time, instead of first capturing the image and then perform the text search on the scanned image.

We have build the same model by incorporating image recognition, Text Analysis and Machine Learning into a dynamic prototype (e.g. Android Camera). It has the feature of searching the text you wish to search by typing it in the search bar. A square box will hover around the text we are searching for. It is just like the face recognition which is seen in the mobile camera. We have focused on dense text like in novel, newspaper, magzines etc. We have not focused on natural scene for now.

A. *Application*

- Suppose we are reading a novel. Suddenly we feel to search relevance of a particular word in previous few pages. We simply hover the mobile phone over the text to search for the same word instead of reading whole lines finding that word.
- Assessment of a particular document on the basis of particular keywords.
- Searching of a particular library book amongst others in book stack.
- Searching a particular licence plate of a car in traffic.
- Help in categorising paper works in offices based on certain keywords.

B. *Prior and Related Work*

1. ***Text Extraction*:** Text extraction techniques are widely studied because text embedded in images and videos provides important information. Many characteristics of text regions have been summarized and characterized effectively by several features, e.g. text pixels have near-homogenous color, character strokes form distinct texture, etc. Y. Hasan and J. Karam developed a text extraction algorithm that utilized morphological edge/gradient detection [1]; Algorithm by Epshtein et al tackled the problem from another approach using text stroke transform [3].

2. ***OpenCV*:** OpenCV stands for Open Source Computer Vision. It is a library of programming functions for real time computer vision. The library has more than 2000 optimized algorithms and has been widely used around the world. Befitting from this, android programmers are able to implement many digital image processing algorithms in Android phone platform.

3. ***Optical Character Recognition*:** OCR, Optical Character Recognition, is developed to translate scanned images of handwritten, typewritten or printed text into machine-encoded text. A lot of OCR software have been developed to accomplish this mission. Tesseract, originally developed as proprietary software at Hwelett- Packard between 1985 and 1995, now sponsored by Google, is considered to be one of the most accurate open source OCR engine currently available. It is capable of recognizing text in variety of languages in a binary image format.

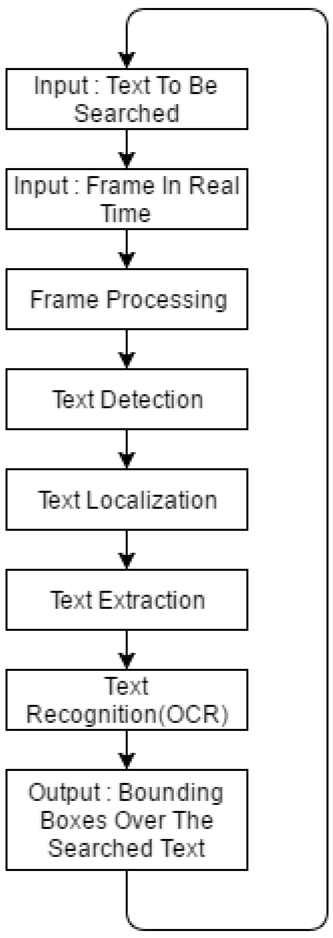

Input : Text To Be Searched
Input : Frame In Real Time
Frame Processing
Text Detection
Text Localization
Text Extraction
Text Recognition(OCR)
Output : Bounding Boxes Over The Searched Text

Fig 1 : Flow Chart Of Android Application For Real Time Text Search.

II. SYSTEM FLOW

In this paper, we propose a text detection / recognition/ translation algorithm that consists of following steps:

1. Morphological edge detection
2. Text feature filtering
3. Text region detection
4. Text recognition
5. String Matching
6. Display of Results

A. Step 1 - *Morphological Edge Detection*

To perform the edge detection algorithm, we first convert the input RGB colour image to a grey-scale intensity image Y using (1), where R, G, and B represent red, green and blue components of the input image.

$$Y = 0{:}299R + 0{:}587G + 0{:}114B \quad (1)$$

Next Canny edge detection algorithm is used to detect edges in the characters. It is a multi-stage algorithm which goes through stages described below.

1. Noise Reduction

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter. We have already seen this in previous chapters.

2. Finding Intensity Gradient of the Image

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel as follows:

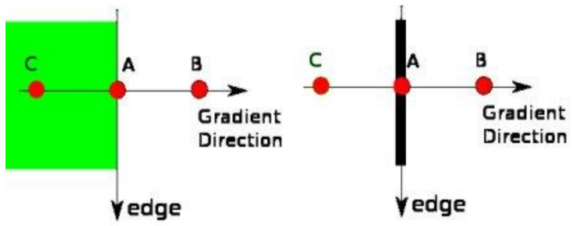
$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

3. Non-maximum Suppression

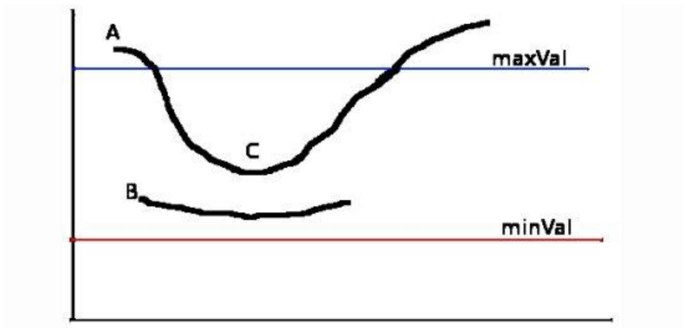
After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient.



Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero). In short, the result you get is a binary image with "thin edges".

4. Hysteresis Thresholding

This stage decides which are all edges really edges and which are not. For this, we need two threshold values, *minVal* and *maxVal*. Any edges with intensity gradient more than *maxVal* are sure to be edges and those below *minVal* are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded.



The edge A is above the *maxVal*, so considered as "sure-edge". Although edge C is below *maxVal*, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above *minVal* and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select *minVal* and *maxVal*accordingly to get the correct result.
This stage also removes small pixels noises on the assumption that edges are long lines.
So what we finally get is strong edges in the image.
The image so obtained is a binary image with edges drawn in white colour.

B. Step 2 – *Text Feature Filtering*[4]

In order to reduce the number of connected components that have to be analysed, a close Operation [5] (dilation followed by erosion) with a 5 by 5 structuring element is performed to the binary edge image obtained from Step 1.

C. Step 3 – *Text Region Detection*[5]

After step 2 the number of connected components are reduced. In order to detect the candidate text region MSER [6] region are found from the image obtained after step 2. Each individual region is an array of (x, y) coordinates of the boundary points of the connected component.
MSERs are connected regions,that are defined by an extremal property of the intensity function in the region and on its outer boundary. MSERs have properties that form their superior performance as stable local detector. The set of MSERs is closed under continuous geometric transformations and is invariant to affine intensity changes. Furthermore MSERs are detected at different scales.
Now for each contour so obtained, a bounding rectangle is found. The rectangle is marked by the coordinates of the top left corner (x, y), height h, and width w. For each rectangle the corresponding part of the grey scale image is send to the text recognition algorithm.

D. Step 4 – *Text Recognition*:

After the text detection, Tesseract OCR [2] recognizes the text in the region which is send to it by our text detection algorithm. It recognizes the text and converts it into a string and sends it to the string matching system.

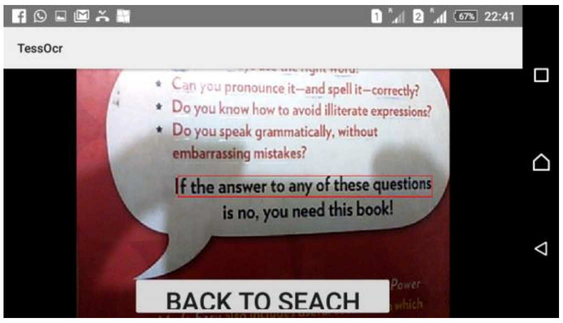
E. Step 5 – *String Matching and result matching*

The text recognized by tesseract is than matched with the text query and if the match is found, a bounding box around the text region is drawn.

III. MULTITHREADING

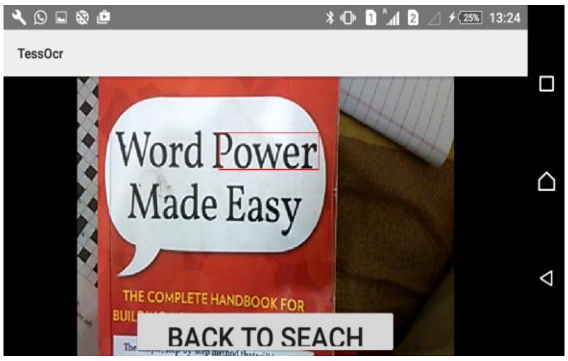
The proposed method was quite slow when we implemented it in python on desktop (around 40 secs). So we implemented multi-threading and as expected the time reduced drastically to 11 secs (still slow but fine considering python).
The text regions were divided in three different groups
(since we used three threads) and these groups were processed in parallel. The main program waits until the three threads finishes executing and finally displays the results.

IV. TEXT SEARCH RESULTS

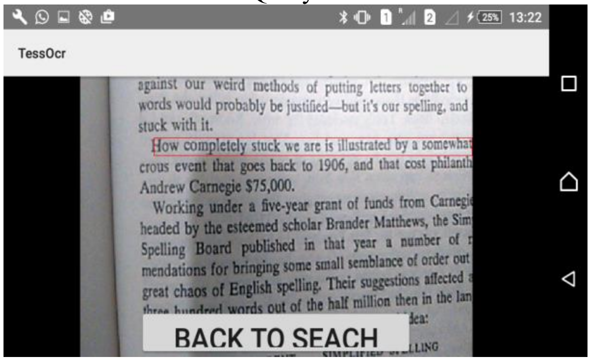


• Can you pronounce it—and spell it—correctly?
• Do you know how to avoid illiterate expressions?
• Do you speak grammatically, without embarrassing mistakes?
If the answer to any of these questions is no, you need this book!

Search Query: "answer"

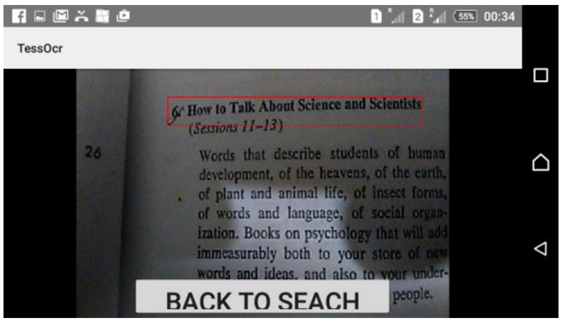


Search Query: "Power"



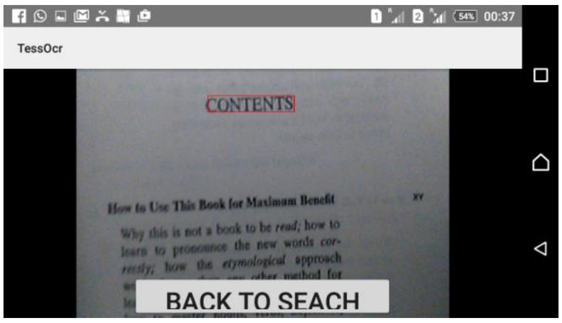
against our weird methods of putting letters together to words would probably be justified—but it's our spelling, and stuck with it.
How completely stuck we are is illustrated by a somewha
crous event that goes back to 1906, and that cost philanth
Andrew Carnegie $75,000.
Working under a five-year grant of funds from Carnegi
headed by the esteemed scholar Brander Matthews, the Sim
Spelling Board published in that year a number of r
mendations for bringing some small semblance of order out
great chaos of English spelling. Their suggestions affected
three hundred words out of the half million then in the lan

Search Query: "stuck"



How to Talk About Science and Scientists
(Sessions 11–13)
26
Words that describe students of human development, of the heavens, of the earth, of plant and animal life, of insect forms, of words and language, of social organization. Books on psychology that will add immeasurably both to your store of new words and ideas, and also to your under-
people.

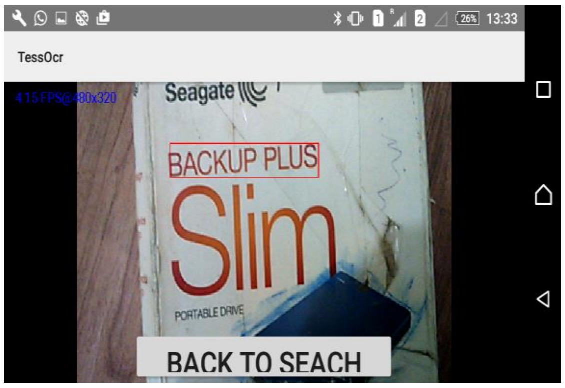
Search Query: "Science"



CONTENTS

How to Use This Book for Maximum Benefit xv
Why this is not a book to be read; how to learn to pronounce the new words correctly; how the etymological approach
other method for

Search Query: "CONTENTS"



Search Query: "BACKUP PLUS"

V. EXPERIMENTS AND ACCURACY

We Conducted tests on dense text by searching the words using our application. We hovered the phone over the text of different sizes and calculated the search accuracy. We have kept an average distance of 12.5 cm between camera and text.

Type Of Text	**%Error**	**%Accuracy**
Very Small Text	4.93	95.07
Small text	4.27	97.38
Big Text	4.64	95.36
Medium Text	2.62	95.73

VI. CONCLUSION

We have achieved an android based application on real time text detection and recognition. From the performance evaluation of our system, we concluded that our application is quite robust for large text. Following work needs to be done in order to drive our application into commercial product:

- Adapt fast text extraction algorithm to smaller and denser text.
- Detect and recognize text in Natural scenes.
- Further enhance the speed and improve accuracy.

VI. ACKNOWLEDGMENT

We would like to than Dr. S. K. Singh for his guidance and support which helped us to do our work on time and build a usable application.

VII. REFERENCE

[1] Yassin M.Y.Hasan and Lina J.Karam, Morphological Text Extraction from Images. IEEE Transaction on Image Processing Vol.9 No.11, Nov 2000.

[2] B. Epshtein, Detecting Text in Natural Scenes with Stroke Width Transform. Image Rochester NY, pp. 1-8.

[3] http://code.google.com/p/tesseract-ocr/

[4] http://homepages.inf.ed.ac.uk/rbf/HIPR2/guidecon.htm

[5] http://docs.opencv.org/2.4/doc/tutorials/tutorials.html

[6] MSER-based Real Time Text Detection And Tracking. Llu´ıs G´omez and Dimosthenis Karatzas.