Asian Journal of Convergence in Technology Volume IV Issue III
ISSN NO: 2350-1146 I.F-5.11

Search and Highlight of Required Substrings in Printed Documents using OCR

Implementation of software application to digitally highlight desired text in hard copies

A. Athul Krishna A, Student, College of Engineering TVM, B. Bharath Kartha, Student, College of Engineering TVM and C. Vishnu S Nair, Studnet, College of Engineering TVM

Abstract—The implementation of a software application to search for and highlight desired text in a printed document is explained in this paper. An image feed of the hard copy in which search is to be done is given as input to software along with the desired substring whose location is to be identified within document. The program explained in this paper coverts the image to a text document using the Optical Character Recognition (OCR) engine Tesseract, searches through it, highlights the desired substring in the image and displays it, thereby making the detection of its location in the actual hard copy an easy job.

Keywords—dynamic optical character recognition, substring search in printed documents

I. INTRODUCTION

Human brains have evolved such as to devote maximum image processing power to identifying faces, patterns and other natural phenomenon. Consequently, they are bad at locating specific details in information dense situations. This becomes apparent especially in text available as printed documents and images (as opposed to digital text documents where most usually a search option is present for easy location of desired substring). Thus in such applications, wherein human have to search for words and patterns in a text dense copy or image, the process is often unnecessarily time consuming and frustrating for the user. The technology introduced in this paper provides a search facility to real life documentations. This technology reads characters from a desired image and finds desired letters or words from the aforementioned image input.

By using a camera (or any other such imaging devices) a digital copy of the desired document or dense text is generated and input to the software. The technology elucidated in the forthcoming sections leverages advances in Optical Character Recognition systems (OCR) to generate a word searchable file of the letters and words identified from the input image. This file can then be searched to locate desired keywords and substrings by the user. To facilitate ease in locating the position of the identified text in the original input image (of the printed document), a digital copy is generated highlighting the located word in the image itself. In the prototype developed, Tesseract[1] is used as the OCR engine, python[2] is used as the programming language, and various libraries within python like PIL[3], Tkinter[4], os [5] etc are used.

The rest of this paper is divided into the following sections : Section II deals with how the software was implemented, Section III details the algorithm used, Section IV describes one specific application where this technology can be utilized, Section V some miscellaneous applications, Section VI gives some test results of the software, Section VII gives some possible future work and improvements and Section VIII is the conclusion.

II. IMPLEMENTATION

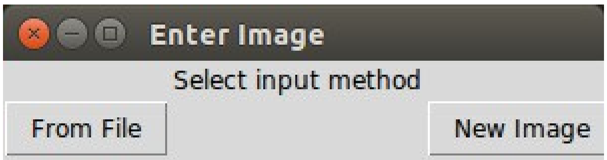
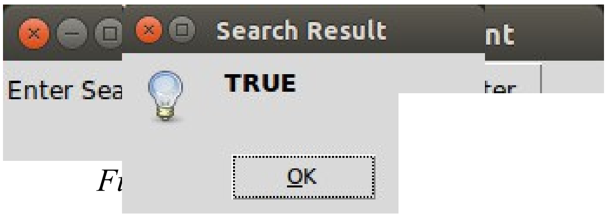


Fig 1: GUI to select image for OCR engine



Search Result

TRUE

OK

Fig 3: GUI to disp search result

Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License, Version 2.0, and development has been sponsored by Google since 2006. In 2006 Tesseract was considered one of the most accurate open-source OCR engines then available.

For this particular project, the Tesseract engine has been utilized for the purpose of Optical Character Recognition. Python-tesseract[6] is a wrapper for `Google's Tesseract-OCR Engine[7][8].Python-tesseract is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Python Imaging Library(PIL), including jpeg, png, gif, bmp, tiff, and others, whereas tesseract-ocr by default only supports tiff[9] and bmp[10]. Additionally, if used as a script, Python-tesseract will also print the recognized text instead of simply writing it to a file.

However, its efficiency for the said purpose of this paper has not been studied and rather Tesseract engine itself has been utilized using the 'os' library of python. The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux. In this particular project, the operating system used was Linux (Ubuntu 14.04 LTS).

Further, for image manipulation, the Python Imaging Library (PIL) has been used. PIL (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux. Its successor project Pillow adds support for Python 3.x. For the project described in this paper, Python version 2.7.6 has been utilized. The 'sys'[11] library has also been used to parse arguments of input image file from user and a flag from the user, to be used during execution of the program.

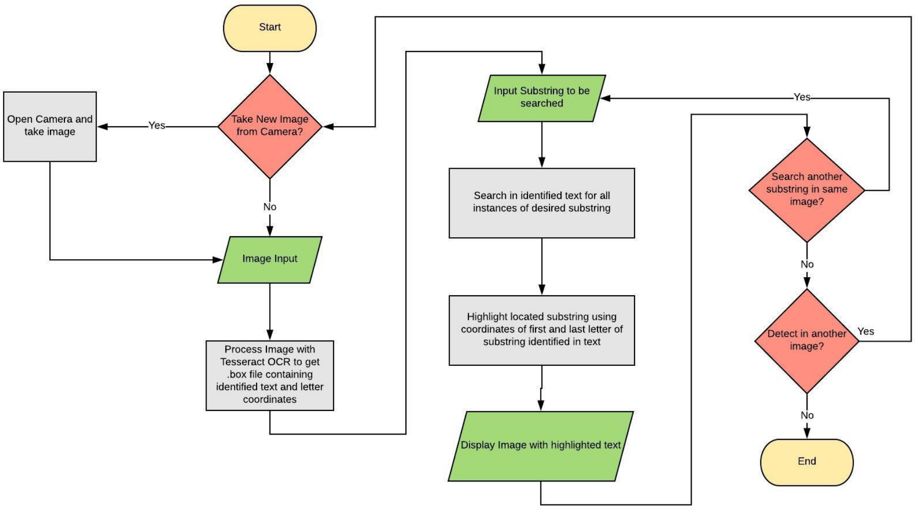

Start
Take New Image from Camera?
Yes
Open Camera and take image
No
Image Input
Process Image with Tesseract OCR to get .box file containing identified text and letter coordinates
Input Substring to be searched
Search in identified text for all instances of desired substring
Highlight located substring using coordinates of first and last letter of substring identified in text
Display Image with highlighted text
Yes
Search another substring in same image?
No
Detect in another image?
Yes
No
End

Fig 4: Flowchart representation of working

III. ALGORITHM

The flowchart given in Figure 4 illustrates the overall algorithm followed for implementation.

Pseudo Code:

1)Prompt user for input image - select existing or take new image from web cam or any such imaging device connected to device used for the purpose.2)Pass the input image through the Tesseract engine via the OS library with appropriate parameters so as to produce the .box file. This file format is output from the Tesseract engine itself and contains the individually identified text in the input document as well as the coordinates of these individual letters within the original image.

3)Input substring to be searched from user. This can be letters, words or phrases, input as String type data.

4)When substring has been received, calculate its length and search the generated .box[12] file for all instances of the substrings of same length as the input, with equal letter positions(not counting the spaces between words, which is not identified by Tesseract).

5) Once substring location has been identified (Output "True") within the image, highlight the desired region using coordinate data of individual letters as can be selected from the .box file within a rectangle (using the PIL library). If no substring could be identified in image, return output "False" and skip to step 7.

6)Display the image with identified substring(s) highlighted.

7)Prompt user to search for another substring in image. If yes, go back to step 3. Else, to step 8.

8)Prompt user to input another image to be searched. If yes, go back to step 1. Else, end the program.

For non-GUI application, the first argument (zeroth argument is name of file itself) passed via the terminal and parsed by the program, is a flag (taking value 0 or otherwise) which the user can input to tell the program whether to regenerate the .box file; and the input image name (.jpg file) can be passed by user as second argument in the terminal. The former, the first flag argument, is useful as generation of the .box file is the most time and memory consuming process in the application and inclusion of a flag to make its generation optional helps increase efficiency, especially when user wants to search for multiple substrings within the given image and/or accidentally or otherwise closed the application and would like to search for another substring in the last input image itself.

IV. APPLICATION IN BOARD BRING-UP

A primary motivation for implementing this project came to the authors while working on a Board Bring Up session at an embedded systems industry. Board bring-up is a phased process whereby an electronics system, inclusive of assembly, hardware, firmware, and software elements, is successively tested, validated and debugged, iteratively, in order to achieve readiness for manufacture. It is an absolute must before any embedded product is brought to the market via mass production and is unavoidably done by all manufacturers to prevent loss and ensure reliability of the product. Board bring up is often a very lengthy and arduous process that takes up a lot of development time and is done manually by trained verification engineers.

The first and most important step of Board bring up is known as visual inspection. This is often also the most tiring and time consuming phase, especially for more complex circuits. Visual inspection phase consists of the verification engineer manually examining the printed board for defects. This includes checking for short circuits, break in connections, improper soldering, and making sure all the components that should be in place are in their correct places in the correct configuration and polarity, and also making sure that components marked as Don't Place (or its equivalent terminology) haven't been placed(soldered) in the circuit. For the same, during this procedure the engineer has to go through text and image dense technical schematics, the verbose Bill of Materials(BoM) etc. These are often provided to him/her as printed documents and are thus not digitally search able. This proves as a disadvantage as well as a waste of productive time for the engineer as he/she has to look up and find specific combination of numbers and words - substrings - within an document and identify the location of multiple such word number combinations iteratively many number of times.

It is in such a scenario that the application mentioned in this paper comes to use. The verification engineer only needs to input an image of the printed document and he/she can search it easily for required component names etc, any number of times easily and efficiently. Thus locating a component mentioned on the Bill of Material on the schematic, and thus on the printed circuit board (PCB) becomes a much easier job for the engineer. This application thereby increases the work efficiency of verification engineers and thus saves time, energy and resources.

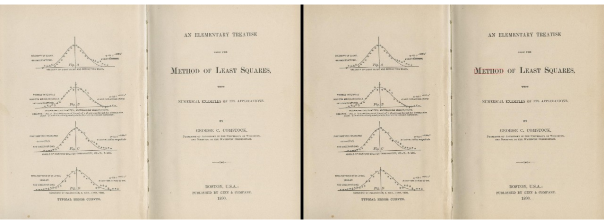
V. MISCILLANEOUS APPLICATIONS

The solution can be used for easy searching text in printed material. It can be a very tiresome activity to search for a single word among thousands of them in a book or other printed documents. With this solution all that an user has to do is to enter the word to be searched for in the application and scan the document. The application will dynamically highlight the respective text in the content. This makes it very comfortable and time saving, improving the work efficiency an user.

Most of the applications available provides features like find, find and replace, etc. But there may be cases where this facility is not available. This solution can be of great usage in such scenarios. As almost all of the soft copy content is in standard fonts and not handwritten, this solution could be easily used as an alternative there as tesseract is equipped to deal with exactly such type of data.

In some cases, in a text content, there will be texts in different fonts and orientations. Text sizes may also differ. In such cases it will be very difficult to search and the find feature may also not work. But since this application uses image processing for character recognition, it can find texts of any printed font, size or orientation, making it a robust solution.

VI. TEST RESULTS



AN ELEMENTARY TREATISE

METHOD OF LEAST SQUARES,

WITH

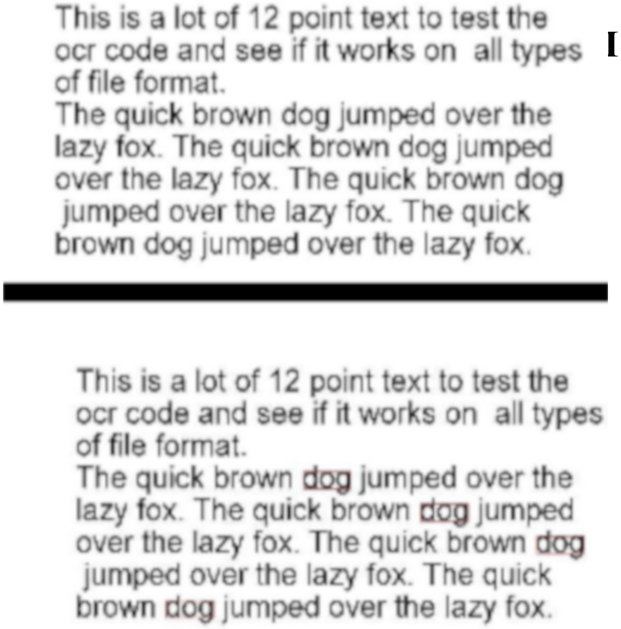
NUMERICAL EXAMPLES OF ITS APPLICATIONS.

BY

GEORGE C. COMSTOCK,

BOSTON, U.S.A.:
PUBLISHED BY GINN & COMPANY.
1890.

Fig 6: Test Result with non White background and multiple images



This is a lot of 12 point text to test the ocr code and see if it works on all types of file format.
The quick brown dog jumped over the lazy fox. The quick brown dog jumped over the lazy fox. The quick brown dog jumped over the lazy fox. The quick brown dog jumped over the lazy fox.

This is a lot of 12 point text to test the ocr code and see if it works on all types of file format.
The quick brown dog jumped over the lazy fox. The quick brown dog jumped over the lazy fox. The quick brown dog jumped over the lazy fox. The quick brown dog jumped over the lazy fox.

Fig 5: Test image identifying multiple words within same text

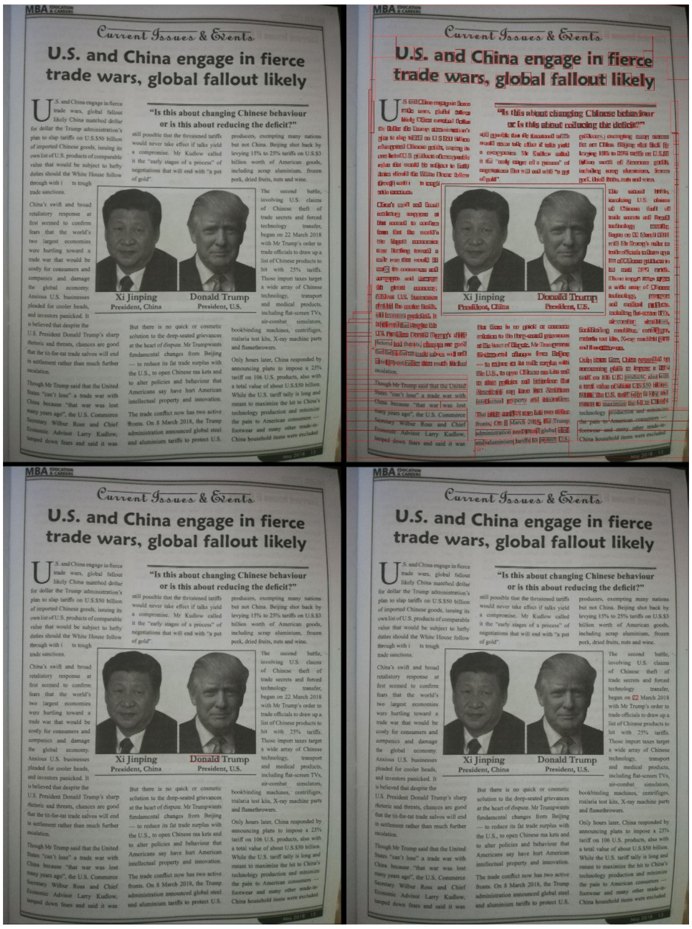

Current Issues & Events
U.S. and China engage in fierce trade wars, global fallout likely
"Is this about changing Chinese behaviour or is this about reducing the deficit?"
Xi Jinping
President, China
Donald Trump
President, U.S.

Fig 7: Clockwise from top left - Image taken from Mobile Camera (12MP) of dense text, All detected letters in image by OCR Engine, The number '22' located and highlighted in image, The word "Trump" highlighted in image

VII. FUTURE WORK

Future works envisioned for this project include better GUI, a more optimized OCR engine and more cross platform usability. Another major area of development would be to develop this project, currently working on linux OS desktops to Mobile Phones. The feasibility of such a move has to be studied in terms of processing power required and average processing time for a standard image across OS and hardware specification. Developing this project as a mobile application would hugely increase its scope of usage and ease of access. Making the application voice interactive can prove to be more comfortable. Then, instead of typing the word to be searched, a user need only speak it out before scanning. Going further, it can also be integrated into cutting edge technologies such as Smart Glasses like Google Glass.

VIII. CONCLUSIONS

The proposed solution can be of huge advantage in many situations. It is an example of converging different technologies to make smarter solutions. It has the capability to revolutionize the work in may fields, making them more efficient and time saving. It is clear that the solution is simple and robust.

REFERENCES

[1] R. Smith, "An Overview of the Tesseract OCR Engine," Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Parana, 2007, pp. 629-633. doi: 10.1109/ICDAR.2007.4376991 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4376991&isnumber=4376969
[2] https://www.python.org/
[3] https://pillow.readthedocs.io/en/5.1.x/
[4] https://docs.python.org/3/library/tk.html
[5] https://docs.python.org/2/library/os.html
[6] https://pypi.org/project/pytesseract/
[7] https://opensource.google.com/projects/tesseract
[8] https://en.wikipedia.org/wiki/Tesseract_(software)
[9] https://en.wikipedia.org/wiki/TIFF
[10] https://en.wikipedia.org/wiki/BMP_file_format
[11] https://docs.python.org/2/library/sys.html
[12] https://www.reviversoft.com/file-extensions/box