Book Organization Checking Algorithm using Image Segmentation and OCR

Mohammad Azim Ul Ekram, Anjani Chaudhary, Ashutosh Yadav, Jagadish Khanal, Semih Aslan
Ingram School of Engineering, Texas State University, San Marcos, Texas, USA

***Abstract*—Organizing and rearranging library books in appropriate order requires attention and care of librarians. The book indexing and organizing algorithm will detect misplaced library books and suggest a proper position to the user. It can be implemented in a smartphone or a server or an autonomous embedded system. It first segments the captured image to find proper tag area in the book. Then crops that tag area and process that with OCR to detect string. From the string, it is possible to determine if a book is on the right shelf or row.**

Keywords—Segmentation; OCR; Image Dilation; Image Erosion; Grayscale Processing; Color Processing; Pattern Recognition; Rectangle Recognition

I. INTRODUCTION

Image segmentation plays a significant role in the field of computer vision as well as image processing. Image segmentation partitions the image to segments associated with objects or a particular region. Image segmentation is highly application specific and is not a general-purpose algorithm. There are many factors in image segmentation, such as nature of image, illumination condition and spatial constraints, all determine the type of segmentation techniques required.

Optical Character Recognition (OCR) is a technique for automatic recognition of characters or a whole block of words and sentences. Using OCR, nowadays old books are digitized and reprinted with better visual aesthetics. Among many OCR engines, the Tesseract OCR engine is the most used and worldwide recognized for both academic and commercial purpose. It began as an HP research prototype between 1984 and 1994 [1], [2]. In 2005, it was released as an open source project.

Books in libraries are arranged by call numbers. These call numbers are generated for each book in which the classification and the reference numbers are printed and pasted on the book side cover. These call numbers are used by readers and librarians to find a book in specific shelf and row. There are three major English book classification system, namely, Dewey Decimal Classification (DDC), Library of Congress Classification (LCC) and Universal Decimal Classification (UDC) [3]. Among these, LCC and DDC are mostly used.

Many books in libraries can be misplaced and placed on a wrong shelf after they are used. Finding any misplaced book on a shelf is very difficult because all book call numbers need be checked manually to find the misplaced book. Also, the checking procedure can also be erroneous. The process of checking the printed call numbers on the side cover of books can be automated using image segmentation and OCR. After the tag portion is segmented, the OCR will parse the call numbers, which is a sequential numbering system. They can be processed based on LCC, DDC or UDC format. From the format and sequences, out of order call numbers can be determined. It is also possible to communicate with the library server for a list of books and their call numbers which can be compared with the OCR string and find lost books.

This proposed work proposes an automated algorithm to solve this problem of checking books. The paper is organized into four sections. Section II covers the proposed algorithm for image segmentation and OCR, and Section III covers the experimental results, and Section IV includes the conclusion and future work.

II. ALGORITHM

The algorithm has three main steps as shown in Figure 1. First part is to segment the tag portion of the book from the whole image. Secondly, parsing the call number from the segmented image using OCR. Thirdly, compare the call numbers sequentially with help from the server and determine the correctness of book position. This algorithm can be implemented in a Smartphone or in an autonomous robot which can take images of books and determine the correctness of book order.

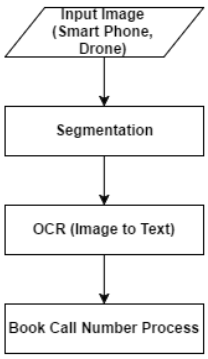


Fig. 1. Basic Steps of the Proposed Algorithm

A. Segmentation

As mentioned above, Image segmentation is an application-oriented problem. To get desired results, most of the time image segmentation requires pre- and post-processing steps. Sometimes, segmentation results are highly subjective. Pre-processing is a significant step where the image filters are

applied to make the image smooth and noise free, which simplifies the subsequent segmentation steps. The user can directly intervene in the segmentation process by giving input to the algorithm. In this article, segmentation is processed in four major steps namely, edge-preserved smoothing, morphological operation, binarization and region detection.

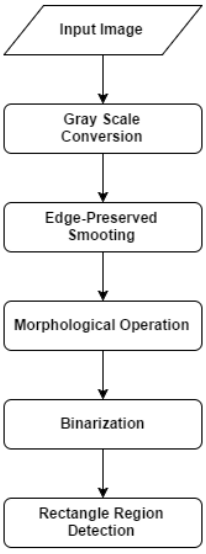


Fig. 2. Steps of Segmentation Algorithm

The steps are shown in Figure 2. It is required to convert the color image to grayscale for segmentation. From this point out, most of the processing will be assumed in grayscale format unless otherwise mentioned.

1) Edge-Preserved Smoothing

A gradient filter is applied to the image to get fine edges [4]. Mathematically, gradient magnitude Δf is defined as:

$$\Delta \mathbf{f} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (1)$$

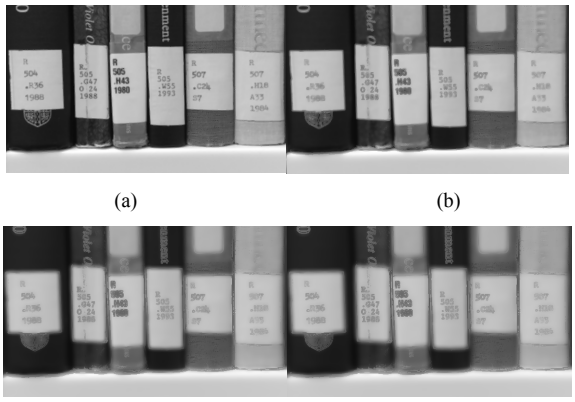
$$\Delta f = mag(\Delta \mathbf{f}) = \sqrt{\left[g_x^2 + g_y^2\right]} \quad (2)$$

Here $g_x$is the gradient magnitude along x direction and $g_y$is the gradient magnitude along y direction. To compute the gradient magnitude, 'Sobel' filter and its transposed version for x and y direction is used [5]. The output of gradient filter is shown in Figure 3.

This gradient magnitude is then binarized and used as a map for an edge enhancer while other parts of the image are smoothed for unnecessary edge removal. This is called guided image filtering [6].

(a) (b)

Fig. 3. (a) Input Image, (b) Gradient Magnitude Output



(a) (b)

(c) (d)

Fig. 4. Image Gradient for Different Neighbor size of (a) 3x3, (b) 5x5, (c) 9×9 and (d) 12×12

The kernel size used for this edge-preserving smoothing function is 3x3. Furthermore, 5×5, 10×10 can also be used too. However, as shown in Figure 4, higher kernel introduces unwanted edges around the edge figures which could cause extra segmentation. For this reason, neighborhood size of 3×3 is selected.

2) Morphological Operations

Two morphological operations, erosion, and dilation followed by reconstruction are necessary to remove the textual background. Dilation and erosion are two fundamental technique for image segmentation. Creating binary images of our desired region is the primary goal. To do that, first, the foreground marker for our tag area needs to be determined. The foreground marker can be generated by two morphological techniques called "Opening by reconstruction" and "Closing by reconstruction." "Opening by reconstruction" tries to restore the original shape of the objects which were reduced by erosion [7].

The structure element used by the closing and opening is important. It is imperative that the vertical and horizontal edges are intact while reconstructing. Hence, a 'Square' shaped structure element is used. A "Closing" which follows an "Opening" usually removes some dark spots and haziness in the image, which will make it possible to find a better the foreground marker [7]. The results of an opening followed by a closing are shown in Figure 5.

(a) (b)

Fig. 5. Effect of Opening Followed by a Closing. Output After (a) Opening by Reconstruction, (b) Output Followed by Closing by Reconstruction

3) Binarization

Binary images are a bivalued function of spatial coordinates. Morphologically, binary images are a set of 1's and 0's where 1's denote the foreground where target object relies on. As binary images contain only 0's and 1's, an application from set theory is feasible and practical. For instance, if A and B are binary images and C=A∪B, then C is also a binary image which can be defined as;

$$C = \begin{cases} 1 & if\ either\ A(x,y)\ or\ B(x,y)\ 1\ or\ both\ are\ 1 \\ 0 & otherwise \end{cases} \quad (3)$$

It can also be written in a set form as:

$$C = \{(x,y) | (x,y) \in A \text{ or } (x,y) \in B \text{ or } (x,y) \in (A \text{ and } B)\} \quad (4)$$

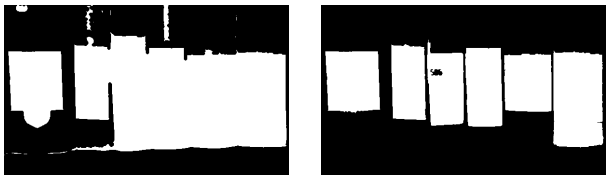
Binarization is straight forward. In this article, binarization is done by simply thresholding the morphed output from previous stage [7]. Another intermediary binary image is created from the gradient of the image. After converting the image to binary, dilation and hole filling are performed on the binary image. This gradient intermediary image acts as a mask for the binary image of the morphed stage to remove unnecessary pixels. Also, small region is removed using binary area calculation [7]. The output is shown in Figure 6.

4) Rectangle Region Detection

Detecting which regions are a rectangle is directly done. The segmented regions are detected using boundary box and calculated area by pixels. If the calculated area by using boundary matches closely with the pixel calculated area, then the desired segmentation is achieved.

B. Optical Character Recognition

Some pre-processing is required for the OCR to parse properly. Most OCR techniques require three steps, first, acquisition of image or digitally scanned documents. Second, recognition of characters, which typically involves converting documents to a stream of pixels which representing letters of recognized words and the final element is to store as a group of words or letters.



(a) (b)

(c)

Fig. 6. (a) Intermediary Binary Image from Image Gradient, (b) Final Image Binarization, (c) Final Image Segments Calculated

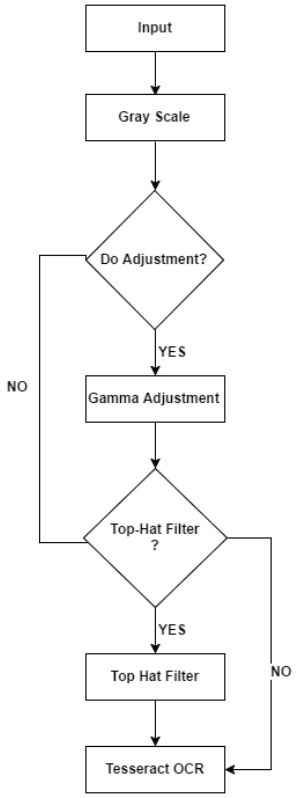


Fig. 7. OCR Preprocessing Algorithm

The OCR's ability to recognize characters is proportionally dependent on the quality of images. The OCR mostly assumes the input image totally consist of text with little to none pictorial cluttering. The layout of texts in image documents is also important. If the texts are skewed and/or rotated, the OCR normally doesn't work well without specific parameters [8]. The OCRs accuracy varies from 16% to 100% based on various factors such as colors, rotation, skewing, formatting, page border, and handwriting [9].

In this project, the segmented images from binarization need to be pre-processed before feeding them into OCR. The flowchart in Figure 7 shows the important parts of the pre-processing. Histogram equalization and a "closing by reconstruction" are the first pre-processing that needs to be done. Although this is an optional pre-processing, it is mostly used to achieve better results. Also after adjustment processing, Top-

Hat filtering is applied to a disk-shaped structural element of size 12 to remove uneven background illumination. This creates a darker backed image with more processable tex and is shown below in Figure 8 (a,b).

III. Experimental Results

The experiment was implemented on real samples using 200 books. These books were captured using a smartphone, where conditions like brightness, tilt, out-of-focus, typed-tag, printed tag were considered.

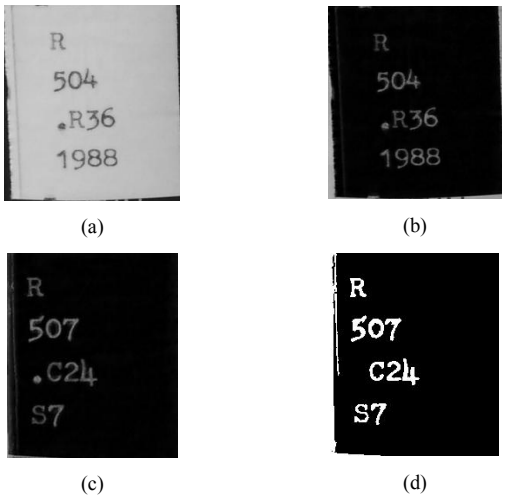


Fig. 8. Pre-Processing Output Before OCR a) Image Adjustment, (b) Top-Hat Filtering (a) Top-Hat without Adjustment, (b) Binarization

Each sample image has at least three tags and at most six tags to maintain good zoom ratio. While capturing the samples, the distance of the smartphone to the shelf is kept at a fixed distance, usually between 0.5 to 1.5ft.

For convenience, the algorithm is implemented in MATLAB. The sample images are resized to approximately 6MP and then processed for segmentation. Built-in Tesseract OCR engine in MATLAB is used.

The number of sample images tested with this algorithm is 100. They are taken from a university library in sufficient lighting condition, from an approximately fixed distance. Each image contains between 3 and six tags. A total number of tags in those samples are 230. Of those 230 tags, 175 of them were successfully parsed by Tesseract OCR. There are some "defective" tags which are taken into the sample set. Two main defective conditions are white tags on a white background and typed tags. Some of the books are very old, and the tags were typed rather they are printed. Typed tags tend to fade and blur over time, which makes them hard to detect. Also, some were tilted almost horizontally. The OCR algorithm cannot detect call numbers if they have different angles. With all those considerations, the success rate is 76%. The Table I below shows detailed report on the accuracy.

TABLE I. Accuracy Report

Total Tag Count	Successfully Parsed Tags (including defective)	Defective Tags among total tag	% Accuracy	% Defect
230	175	56	76.08%	24.34%

Total tag count is the total number is tag samples which were in the image. The defective ones are the ones which as at least one deformity in them, e.g. tilted, white background or blurred tags. The successfully parsed tags are tags which were segmented by the algorithm and parsed by the OCR including the defective ones. The following two formulas calculate accuracy and defect percentage.

$$\%Accuracy = \frac{Successfully\ Parsed\ Tags}{Total\ Tags} \times 100 \quad (5)$$

$$\%Defect = \frac{Defective\ Tags}{Total\ Tags} \times 100 \quad (6)$$

IV. Conclusion

Although the accuracy is not close to 100%, this algorithm can have a better accuracy if the Hough Transformation is added to the algorithm. Using Hough Transformation, lines and contours can be used for better segmentation. The book tilting and the roundness of the book can be transformed into a better image for OCR. Currently library specific machine-learned training set is being developed, in which case, the output will be much accurate. If properly developed and implemented in real time systems, this algorithm can help libraries, save hours of searching for lost book and rearranging them. A similar idea can be used to develop smartphone apps that can help blind people to identify items on shelves at grocery stores.

References

[1] R. Smith. "An overview of the Tesseract OCR engine. Presented at Document Analysis and Recognition", ICDAR 2007.
[2] R. Mithe, S. Indalkar and N. Divekar. "Optical character recognition", International Journal of Recent Technology and Engineering 2(1), pp. 72-75. 2013.
[3] A. G. Taylor, "Wynar's Introduction to Cataloging and Classification", Library and Information Science Text Series, 9th Ed., 2004.
[4] D. Wang, "A multiscale gradient algorithm for image segmentation using watershelds", Pattern Recognition 30(12), pp. 2043-2052, 1997.
[5] M. Nagao and T. Matsuyama, "Edge preserving smoothing. Computer Graphics and Image Processing", pp. 394-407. 1979.
[6] K. He, J. Sun, and X. Tang, "Guided image filtering", Presented at European Conference on Computer Vision,2010.
[7] R. C. Gonzalez and R. E. Woods, "Image processing", Digital Image Processing 2007.
[8] L. R. Blando, J. Kanai and T. A. Nartker, "Prediction of OCR accuracy using simple image features", Proceedings of the Third International Conference On, 1995.
[9] C. Patel, A. Patel, and D. Patel, "Optical character recognition by open source OCR tool Tesseract: A case study", International Journal of Computer Applications 55(10), 2012.