Mobile Phone Book Finder

Yizheng Liao
Electrical Engineering Department
Stanford University
yzliao@stanford.edu

Jonathan Lu
Electrical Engineering Department
Stanford University
jonylu97@stanford.edu

Meng Wu
Electrical Engineering Department
Stanford University
mengwu@stanford.edu

Abstract— **We have implemented an Android bookfinder application utilizing both Java and Matlab. Our application uses image processing algorithm that will search for the correct book on a shelf given the book name that we are searching for. After implementing this application, we have found that the application has an 86% success rate in choosing the correct book off a shelf. The application is run locally on the phone and thus has improvements in terms of speed over applications that require computation power from a remote server. Since our algorithm uses both an RGB MAP detector and template matching, it works not only for books with text, but also for books that do not contain text along the spine, thereby expanding the program's applicability.**

Keywords: Book Spine Recognition, Android, edge detection, RGB MAP detector

I. INTRODUCTION

Finding the reference books in library is very important for teachers, students, and researchers. Traditionally, people first find the call number of the reference book, which is provided by the library catalog system from the database. Then they are led by the call number to the specific bookshelf where the book should be located. After that, they either compare the call number or check the book name of every book on the bookshelf. In many libraries, each bookshelf may have many books. In addition, the location of each book may vary on bookshelf, or have been misplaced. On top of that, many books on the shelves may not necessarily have any text along the spine, requiring the user to pull the book off the shelf to look at the cover for the title. The culmination of these issues makes the traditional way of finding a specific book among many books to be an inefficient and time-consuming task. In this paper, we propose an Android-based mobile application that provides a tool for finding books in the library by taking a video of the shelf and in real time will look up the image of the book spine from a pre-generated database. The application will automatically point out the location of the desired book on the mobile phone screen.

There have been some literatures on the topic of book spine recognition and identification [1-2]. In [1], the authors proposed a book inventory management system, which recognizes book spines and tracks book location. In [2], the authors proposed a hybrid book spine recognition method, which identifies image features and text in parallel. In both systems, the mobile phone takes the photo and transfers the image to a powerful computer server. All the computations are performed on the server. Thus such applications require the mobile phone to have a wireless connection with the server during the entire image processing and recognition stage. However, in many libraries, the wireless connection is not necessarily reliable and can have low signal-to-noise ratio. In this paper, we propose a real-time book spine recognition mobile application that will run locally on the phone. In our mobile application, the desired book image and data is either preloaded to the phone or downloaded from the online database. All the processing after this stage is performed on the mobile phone and no network access is required. The testing results show that the detection probability of our system is fairly high at 86%.

In Section II, we will discuss the system flow of our application. In Section III, we will discuss the test setup for data collection. In Section IV, we will show the test results and discuss the performance. In Section V, we will conclude the paper.

II. SYSTEM FLOW

When our application starts up, the user will be prompted to provide the name of the book (see Fig 1). After pressing "Search", the mobile application process the input video image from the camera and try and match pre-saved book training data from its database that was pre-segmented in Matlab to what the camera is viewing right now. We will describe the algorithm used in the Android application next.

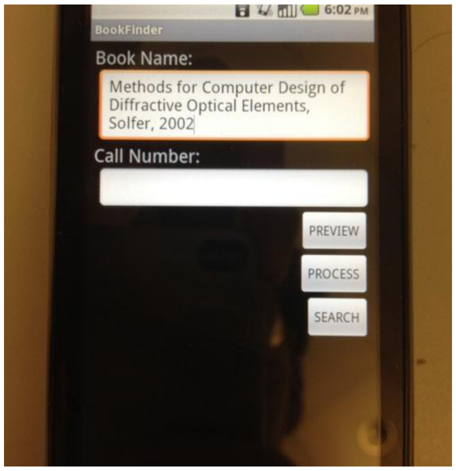

6:02 PM
BookFinder
Book Name:
Methods for Computer Design of Diffractive Optical Elements, Solfer, 2002
Call Number:
PREVIEW
PROCESS
SEARCH

Figure 1. Startup screen prompts user to type in name of book.

In our mobile application, the book spine segmentation and recognition algorithm consists the following steps:

1. Book Spine Vertical Segmentation
2. Book Spine Horizontal Segmentation
3. MAP Detection in RGB Space
4. Template Matching in Grayscale

The book segmentation includes two steps. The first step is vertical segmentation and the second step is horizontal segmentation. For the vertical boundaries detection, we first perform Sobel vertical edge detection. Then we perform erosion and dilation to remove small and non-vertical edges. After that, we extract the vertical lines as book boundaries from the binary image. For horizontal segmentation, we use a similar algorithm. The image is first filtered by a Sobel horizontal edge detector. Then we pick up the top and bottom boundaries within two vertical boundaries of the book.

After segmentation, we use an RGB MAP detector [3-4] and template matching [5-6] to find the desired book spine. In MAP detector, we compute the color distributions for all segmented book spines with similar height-to-width ratios. Then we compare the color distribution with the desired book spine color distribution and the application will outline on the phone the first three books on the shelf with the best match to the book that we are looking for.

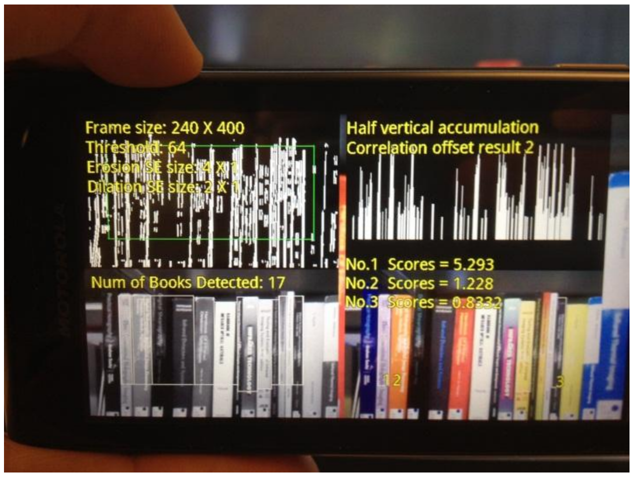


Figure 2. Application segments the books on the shelf and measures their correlation with the book we are searching for.

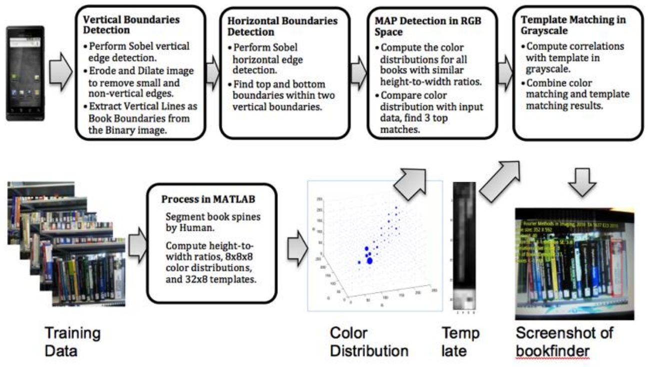

Vertical Boundaries Detection
• Perform Sobel vertical edge detection.
• Erode and Dilate image to remove small and non-vertical edges.
• Extract Vertical Lines as Book Boundaries from the Binary image.
Horizontal Boundaries Detection
• Perform Sobel horizontal edge detection.
• Find top and bottom boundaries within two vertical boundaries.
MAP Detection in RGB Space
• Compute the color distributions for all books with similar height-to-width ratios.
• Compare color distribution with input data, find 3 top matches.
Template Matching in Grayscale
• Compute correlations with template in grayscale.
• Combine color matching and template matching results.
Process in MATLAB
Segment book spines by Human.
Compute height-to-width ratios, 8x8x8 color distributions, and 32x8 templates.
Training Data
Color Distribution
Temp late
Screenshot of bookfinder

Figure 3. System Flow Chart.

In our image processing stage, we use the best one out of five frames to perform segmentation and recognition. In this way, we can achieve a real-time image processing. With this algorithm, we are able to reduce the complexity and latency of our image processing, thus eliminating the need for server power.

In summary, the training data book spine images are pre-collected and stored in the phone. In our application, we used Matlab to collect the desired book spine image. Since this is not done on the phone, we can use the Canny edge filter to segment the reference bookshelf photo, which is taken by the mobile phone. Then we compute the height-to-width ratio, the color distribution of the segmented book spine in RGB space. Also, we save template features of the segmented book spine in grayscale. Finally, we save all the information in a text file and upload the file to the mobile phone or the sample database.

III. TEST SETUP

In order to test out our implementation, we ran our book finder application on a Motorola Android Phone. Using the android application we performed tests on a few test data sets. Our data sets consisted of four different shelves of books. We then performed a search on 68 books from these shelves and examined the reliability and robustness of our bookfinder application.

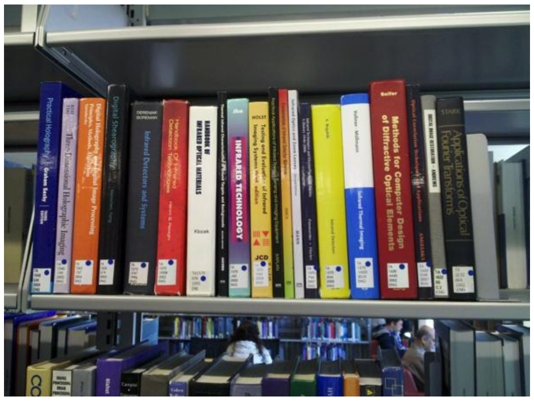
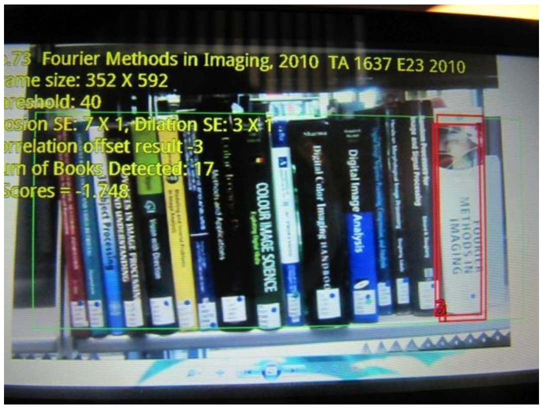


Figure 4. Example of one data set consisting of several books for our bookfinder application to search.



```
.73 Fourier Methods in Imaging, 2010 TA 1637 E23 2010
ame size: 352 X 592
reshold: 40
osion SE: 7 X 1, Dilation SE: 3 X 1
rrelation offset result -3
m of Books Detected: 17
Scores = -1.748
```

Figure 4. Our algorithm working on a data set, and highlights the white book on the right.

Before running the Android application, we used Matlab to help us with segmentation of the training data that will be input into the phone's database. The Android application itself will run locally on the phone without need of a server.

As far as categorizing the performance of our application goes, for each book that we performed a search on, we would grade the performance of the application on a scale from 1 to 3. A score of 1 indicates that the phone application is robust, and that one of the highest correlating overlaid boxes that indicates the search location of the book highlights the correct book and does not suffer from oscillations to other book. The markers will spend the majority of the time centered on the correct book on the shelf. We will also refer this score of 1 as being "strongly detected". A score of 2 indicates that the phone application will highlight the correct book but may spend a majority of time jumping around to other false positives. We may also refer this score as being "weakly detected". A score of 3 indicates that the phone application does not highlight the correct book. We refer to this case as "no detection". Using this scale we can measure the performance of our application.

IV. Results and Discussion

The Matlab edge image segmentation to assist us with edge detection required approximately 20 seconds. The Android application performs its book identification where the video rate is 10 frames per second in real time. Because our code and database are stored locally on the phone, our application runs faster than applications that require communicating with a remote server to perform its algorithms.

After testing out the application on our data sets, we achieved the following results. Out of the 68 books we tested out in our data base, 44 books are categorized as strongly detected, 14 books are categorized as weakly detected. Lastly, 10 of the books are categorized as not detected. This corresponds to having 65% of our books as strongly detected, 21% of our books as strongly detected, and 15% of our books lacking detection. We have summarized our results in table 1.

	Strongly detected (1)	Weakly detected (2)	No detection (3)
Percentage of data set (out of 68)	65%	21%	14%

Table 1. This table summarizes our test results.

While some may view the weakly detected books as a downside to the application, in reality, we feel that dividing our data into two cases: success and failure is sufficient to measure performance of the application. We feel that combining the strongly detected and weakly detected books into one case labeled "success" while renaming the "no detection" group as "failure" is enough. The reason for this is that from a practical standpoint, the goal of this application is merely to *narrow* down quickly where the book the patron is looking for to help him find the book. If the application was able to highlight the correct book on the shelf, even briefly, then the patron will immediately look at "top candidates" as pointed out by the application, and thus, the application will still succeed in pointing out roughly where the book is on the shelf. This is why even the weakly detected book case can be considered as a success since the book finder has done its job in helping the patron.

From this standpoint, we see that our application has a failure rate of 14% which is fairly robust. Unlike other studies which focus on text or feature recognition, our application focuses on using RGB MAP detection which relies on the color on the training data spines. Thus our algorithm has the added advantage of recognizing spines that do not have text or images on them.

V. Conclusion

We have implemented an application on an Android phone that will help to point out the correct book spine on a shelf of books. This Android application is fast, since the algorithms are run locally on the phone, and it is fairly accurate with an 86% success rate in pointing out the correct book on the shelf. Since our algorithm relies on book spine color and shape, the Android application will work for detecting both book spines that lack text and other shape features. Such an application will hopefully make book finding for patrons a quicker and easier task.

VI. Acknowledgements

We wish to thank our mentors David Chen and Derek Pang for their helpful discussions and support throughout this project and class. Finally, we are grateful to Professor Bernd Girod for providing a great class this quarter.

VII. References

[1] D. Chen, S. Tsai, B. Girod, C-H. Hsu, K-H. Kim, “Building book inventories using smartphones”, ACM Conference of Multimedia (MM), October, 2010.

[2] S. Tsai, D. Chen, H. Chen, C-H. Hsu, K-H. Kim, J. Singh, B. Girod, “Combining image and text features: a hybrid approach to mobile book spine recognition”, ACM Conference of Multimedia (MM), November 2011.

[3] Brand J, Mason J A Comparative Assessment of Three Approaches to Pixel-level Human Skin-detection, 2000.

[4] Leahy, EE368 Project, 2003

[5] Aksoy, M et al An industrial visual inspection system that uses inductive learning, 2003

[6] Brunelli, R Template Matching Techniques in Computer Vision: Theory and Practice, 2009

I. Appendix

All group members contributed to the code for segmentation and detection.
All group members contributed to working on the poster
All group members contributed to the final report.