Book and Audiobook Synchronization

Wendy Nie, Tugce Tasci, and Rohit Pidaparthi

Abstract—Our purpose was to create an application that allows a user to jump to the correct section of an audiobook by taking a picture of the page they are reading in a paper book. We used image processing algorithms such as the Hough transform to rotate pictures of the page, Otsu’s method to threshold and binarize the image and brick wall coding (BWC) to detect features on the page. We then used a k-nearest-neighbor algorithm to assign a score to a test page by comparing it to pages in our database. We returned the closest matching page to the user generated picture. We obtained 40% accuracy over a dataset of 20 pages. In future work we could improve our accuracy by using angles between features which are more invariant to the pictures the user takes.

Keywords—Book, Audiobook, Synchronization, BWC, brick wall coding, k-nearest-neighbor.

I. INTRODUCTION

Today, avid readers can access their media in a variety of ways. E-readers are portable, and audio books allow listeners to perform other tasks while following a story; however, some traditionalists prefer the physical experiences associated with paper books. Within the course of a single novel, a reader might have use for all three formats. For example, a user may need to go grocery shopping while following a story; however, some traditionalists prefer the physical experiences associated with paper books. Within the course of a single novel, a reader might have use for all three formats. For example, a user may need to go grocery shopping while reading a paper book. If the user was to switch to the audiobook format, they could continue enjoying the novel while driving a car. On day trips, a quick skim of a digital book in a cell phone might be preferable.

Manually synchronizing these formats can be cumbersome, especially since E-readers have different page schemes depending on font size, audiobook audio files aren’t precisely divided into separate paragraphs, and the same novel by different publishers have different page schemes.

Now that large databases are easily accessible through the cloud, a means of automatically syncing all three formats is possible. Companies like Amazon have implemented methods of synchronizing audiobooks and digital e-reader formats. We propose to bridge the gap between paper books and audiobooks using image processing. The main challenge is matching a chapter of a physical book to a timestamp in the corresponding audiobook.

II. PREVIOUS WORK

We will focus on extracting features from images of a physical books pages and matching them to specific timestamps of an audio book. Although we plan to implement a proof of concept with a few pages in a single book, there is an additional challenge in scaling this to a database of thousands of books. Scanned pages of a physical book will be used to train a classifier while pictures taken by an Android-based smartphone will ultimately be used to search for a page given a database features.

Previous work has been accomplished in this area. In particular, Hull provided brick wall coding (BWC), a feature extraction technique for blurry text documents involving the wordlengths of horizontally and vertically adjacent words [2]. Without noise, classification accuracy was 100%. Other feature coding techniques include the zig-zag and spiral coding. Moraleda demonstrated the effectiveness of the spiral feature vector over the zig-zag and BWC [Moraleda_2010].

Also, Locally Likely Arrangement Hashing (LLAH) has been used as a mapping features to a database of documents [4]. There have been other approaches such as k-nearest-neighbour to determine which features are most similar.

We intend to combine these algorithms in the context of a large-scale database. Our goals are to discover the optimal tradeoff between processing speed, database size, and matching accuracy, and to create a demonstration of this concept on an Android-based phone. The phone would take a picture of any page of a book and automatically play back the relevant time stamp of its corresponding audiobook.

III. METHODOLOGY

A. Image Preparation

A grayscale text image of interest was first binarized by exploring two different methods. One of them was locally adaptive thresholding using by sliding a window over the image and apply Otsu method according to the uniformity of this window. Another one was morphological image binarization which is based on finding the background of the image and then subtracting this background from the image. According to our experiments, locally adaptive thresholding
is faster, but morphological thresholding gives more accurate results. So, we decided to use morphological thresholding for image binarization. The difference between the grayscale image and the morphologically filtered background was then binarized through Otsu’s process to display white text on a black background.

The contour of each letter was found and ordered based on height and width from smallest to largest. Under the assumption that letter height and width are similarly distributed between pages, letter height was estimated to be around the 87th percentile and letter width was estimated to be in the 90th percentile when dimensions are ordered from lowest to highest values. Larger values were selected to ignore the presence of small white patches, such as the dots on i’s.

Additionally, the Hough transform was used to rotate the image so that lines of text are horizontal. The text near bindings are not usable as features and was cropped out.

B. Identifying Text Boxes

Letters were morphed through vertical erosion (SE is $h \times 2$) and horizontal dilation (SE is $2 \times w$) so that those that are part of a common word are connected. The contours of these components were then found and box sizes chosen based on the largest and smallest x and y contour values.

C. Feature Detection

**BWC:** Brick Wall Coding (BWC)[1] is the main feature of interest. The feature is composed of a word box, and the word boxes immediately above and below that overlap with it horizontally. The feature value of interest is the ratio between word box width and height. Although the size of each feature varies, to simplify the problem, the feature is represented by a vector of five values. The first two are the ratios of the two leftmost boxes above the word of interest, the next is the ratio of the word of interest, and the last two are the ratios of the leftmost boxes below the word of interest. To be considered a feature, a word must have at least two eligible boxes below, and two boxes above it.

**ZigZag:** Zigzag coding[3] is the alternative feature extraction method. In this coding method, the feature is composed of the specific wordbox of interest and wordboxes forming a downward left-right zigzagging lane starting from this wordbox. Feature values are the angles between this successive wordboxes forming zigzagging line.

D. Feature Matching

Original, undistorted pages were processed and their features detected. A query image’s features are detected using identical processing.

For each feature of the query image, the $L_2$ norm between said feature and every feature in a page of interest is computed. The smallest norm is saved. A matrix of the smallest norms is created, where each row is for a different query feature change and each column is for a different page number. The column with the smallest norm corresponds with the estimated page of the query image.

For a proof of concept, the feature matrix was trained using the first twenty pages of "Great Expectations".

E. Geometric Verification

Given query features and their corresponding x,y-coordinates, and training features and their x,y-coordinates, geometric verification operates by first identifying similar features between query and training sets. If the similarity (judged by $\|L_2\|_\text{norm}$) surpasses a threshold, the coordinates of the training features are matched to the coordinates of the query features.

Relative angles between different query points and their corresponding relative angle between different training points are computed, and the squared error values between these angles are computed. The mean of these errors is then computed.

This mean squared angle error value is computed between a set of features and each set of page training features. Pages can be ranked according to this error value, and the page that yields the smallest one is the estimated page.

IV. RESULTS

A. Image Size

Image size is an important consideration on Android. If the image is too large, processing time increases. If the image is too small, connected component analysis fails to identify as many valid word boxes because words in adjacent lines tend to move together. After running with image sizes of $320 \times 240$, $640 \times 480$, $1280 \times 960$, and the original test image size, $2560 \times 1920$, it was decided a good size is $1280 \times 960$.

B. Text Box Location & Feature Detection

Binarize and crop worked as expected (Figure 3).

The bounding box function was verified to return only relevant text boxes (Figure 4). By drawing these boxes to the binary image, the location of the boxes was also verified.

Through binary image modification, the BWC feature detector was confirmed to correctly detect BWC features (Figure 5).
C. Page Matching: $L_2$ Norm

Twenty training images were used to test the concept of the $L_2$ norm technique. When training images were used as query images, the page matching algorithm returned the correct answer 100% of the time.

Next, twenty test images were taken using a phone, one of each page. Out of the twenty tests, the correct page ranked in the top five choices six times, which isn’t statistically significant enough to declare this a good classifier.

D. Geometric Verification

Again, twenty training images were used to test the concept of geometric verification. Unfortunately, geometric verification was not confirmed to operate correctly when querying with the training images. More tests must be completed.

E. Optical Character Recognition

Through the library, Tesseract, an optical character recognition (OCR) was used as a test against feature-based coding. It was found that OCR is effective when the photo is clear and correctly-oriented. 6 shows the results of running OCR on a corner of a page. The words in the middle are well recognised. Near the edge, tesseract makes mistakes because the words are cut off.

V. Conclusions

Although the BWC features were detected successfully, matching features between pages or portions of pages proved to be more of a challenge. Several factors may have attributed to this. First of all, the width-height ratio of words doesn’t vary very much so most BWC features are very similar. This makes distinguishing between different features difficult so the $L_2$...
norm might not be ideal. In terms of future work, the effectiveness of different features should be explored, especially ones that, like the BWC, exploit the relative spatial relationships between bounding boxes such as angle or neighbors.

Other errors stem from the variance between testing pictures. Not only can an image be rotated, but the camera may also be skewed (i.e. face the surface of a page by an angle). The former may be compensated through a Hough transform followed by a rotation; more sophisticated techniques would be required to solve the latter. As well, clear and blurry images of books are very different. While a horizontal dilation may have the desired effect on the blurry image, it may not on the clear one, adding variance to the features. Since BWC features are already similar, this strongly exacerbates the problem.

Since time was limited, OCR, which seems to be a promising path given that text databases exist already, was not explored as fully as it could have been. As well, it wasn’t the focus of the project, although its performance acted as a good reference point for the other algorithms.

Lastly, matching algorithms should be explored further. A brute force technique may be enough for a small database of twenty pages; however, for hundreds of thousands of pages, this would be a slow technique. We would like to look at methods that partition our search space into subgroups to limit the number of searches required. For example, a vocabulary tree would give us $\log(n)$ lookup speed instead of the $n$ comparisons we perform with k-nearest neighbor currently.

Acknowledgement

Wendy wrote the bwc code in both Android and Matlab, which involved image binarization, word segmentation, BWC feature extraction, and in Matlab, brute force feature matching. She also contributed to parts of the final report.

Tugce wrote the zig-zag algorithm in Matlab, worked on the poster, and searched through papers to help identify potential implementation techniques. She also wrote code to create an sqlite database for the images.

Rohit worked on setting up the Android application and version control for the project using git. He worked on getting images from the camera and playing the audiobook with Android’s media libraries. He also wrote code to perform k-nearest-neighbor matching on both Android and Matlab. He worked on code for the Android app to perform OCR using the tesseract library. He also contributed to the final report.

References


