International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 3, pp 227-231, April-2024

https://doi.org/10.47001/IRJIET/2024.803032

Image to Text Conversion Using Android Studio

[1]Prof. Rahul D. Dhongade, [2]Srividya Ravishankar, [3]Harshali B. Sabale, [4]Abishek C. Verma

[1,2,3,4]Department of Computer Engineering, AISSMS Polytechnic, Pune, Maharashtra, India

***Abstract* - Text recognition is a crucial aspect of computer vision, playing a vital role in various applications such as document analysis, image search, and robot navigation. Despite significant research efforts, current methods still face challenges and have room for improvement due to factors like different fonts, text orientation variations, and varying image quality. Detecting text accurately is essential for techniques like content-based image retrieval (CBIR). In response to these challenges, this study aims to develop a flexible method for extracting text from challenging images using basic image processing techniques. The proposed system primarily targets processing English text and seeks to create an Android application capable of recognizing English content in images captured with smartphones. Leveraging tools like Mobile Device, Android, and cropping techniques, along with expertise in Android App Development, the goal is to develop a robust solution that can accurately identify and display text in an editable format on the mobile screen. By integrating these elements effectively, the proposed system aims to improve text recognition capabilities on mobile devices and enhance user experience.**

Keywords: Mobile Device, android, cropping, Android App Development.

I. INTRODUCTION

Over the past two decades, the growth of mobile applications has been extraordinary, positioning them as the primary repository of information in human history. Through efficient, fast, consistent, and reliable tools offered by internet and mobile applications, information technology has become deeply integrated into human life, significantly impacting individuals worldwide.

There's now a substantial demand for the digital storage of paper-based information such as books, newspapers, government and hospital forms, and business cards on mobile devices. Despite the advancements, many documents, like government and hospital forms, remain predominantly in print form. Hence, there's a critical need for solutions that swiftly convert paper documents into digital copies, allowing users to store and edit them on their computers and smartphones.

While various tools exist for storing information by scanning text, they often save it as a picture, hindering further processing. For instance, scanned text images do not allow for easy reading word by word or line by line, and reusing the text necessitates manual re-entry of the entire content. Extracting content from everyday scenes and presenting relevant information aids in understanding the broader context. Embedding such applications on smartphones enhances mobility, enabling users to access and utilize the tool wherever they go.. In today's world, where almost everyone has a smartphone and digital content is everywhere, being able to pull text out of images is really useful. This opens up a lot of possibilities, like turning paper documents into digital files or making new kinds of apps for your phone.

1.1 OCR

OCR technology allows the conversion of scanned images of printed characters into text or any other desired information using an Android mobile device.

In its initial stages of development, Optical Character Recognition (OCR) faced numerous challenges, particularly due to limitations in the quantity and complexity of hardware. However, despite these obstacles,over the years, OCR technology has significantly matured, thanks to advancements in technology and the contributions of prominent companies like Hewlett-Packard, Microsoft, International Business Machines (IBM), Google, and others. Ongoing research efforts continue to enhance the capabilities and effectiveness of OCR systems, making them increasingly reliable and versatile.

The process involves three main phases:

1. Scanning: This phase involves capturing the document as optical images using the mobile camera.

2. Recognition: In this phase, the captured images are processed to recognize the characters and convert them into a character stream representing letters of recognized words.

3. Accessing or Storing: The final phase involves accessing or storing the converted text, which is essentially the extracted text from the scanned images.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 3, pp 227-231, April-2024

https://doi.org/10.47001/IRJIET/2024.803032

II. DESIGN AND IMPLEMENTATION

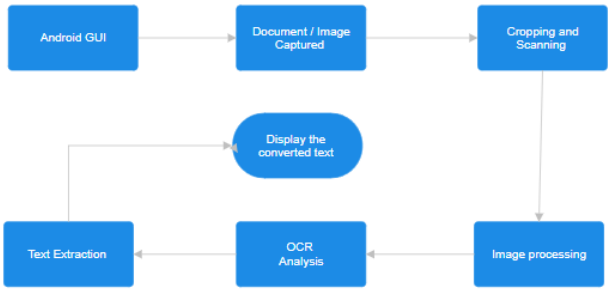

Figure 1: Block Diagram of the App

2.1 Image captured

The image capture module in a text detection app serves as the initial interface for users to input visual data containing text. Leveraging the camera functionality of a mobile device, this module allows users to capture images of documents, signs, or any other text-containing surfaces. Once an image is captured, the app's text detection algorithm processes the image, identifying and extracting text elements from the visual data. This module is crucial for enabling users to digitize physical text and utilize it in digital environments, facilitating tasks such as text translation, data extraction document digitization.

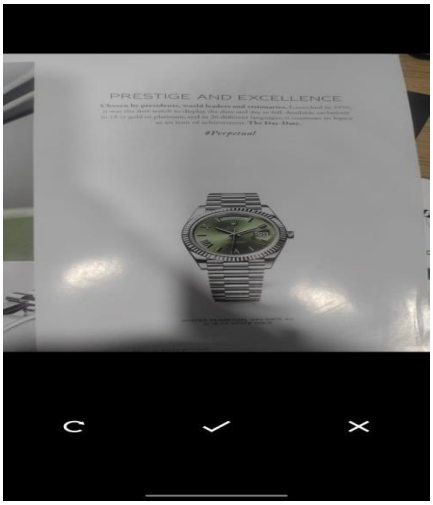

Figure 2: Image Captured

2.2 Image Processing

The image processing module in a text detection app plays a pivotal role in enhancing the quality and clarity of captured images containing text. It employs various algorithms and techniques to preprocess images, such as noise reduction, contrast enhancement, and edge detection, optimizing them for efficient text extraction. This module is crucial for improving the accuracy and reliability of the text detection process, ensuring that the extracted text is clear and legible. Additionally, it may incorporate advanced features like perspective correction and image rectification to correct distortions and align text properly, further enhancing the overall performance of the text detection app. The process involves three main phases:

- **Correcting camera orientation:** Sometimes, photos are captured with the camera held at an angle, resulting in a portrait image appearing sideways. Image rotation allows you to straighten the image and present it in the intended orientation.
- **Image Rotation:** Image rotation refers to the process of digitally turning an image around a fixed point. This can be done in various degrees, such as 90 degrees for a quarter turn, 180 degree.
- **Cropping and Adjustments:** Once you've positioned the crop tool using the gridlines as guides, you can crop the image and achieve the desired composition.
- **Image Ratio:** In an image-to-text detection app, image ratio (aspect ratio) itself doesn't directly impact the functionality of text extraction.

Figure 3: Image processing

2.3 Text Extraction

The text extraction module in a text detection app utilizes optical character recognition (OCR) technology to analyze captured images and extract text content. It processes the image data, identifies individual characters, and converts them into machine-readable text format. This module is essential for converting visual text into editable and searchable digital text, enabling users to perform tasks such as translation, data analysis, and document editing with ease. Additionally, it may

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 3, pp 227-231, April-2024

https://doi.org/10.47001/IRJIET/2024.803032

include features for language detection and text formatting to enhance the accuracy and usability of the extracted text.

Text extraction is the heart of an image-to-text detection application. It's the process of converting the identified text regions within a captured image into usable digital text format. Here's a detailed look at how this happens in an Android Studio app:

- **Firebase Integration:** Set up Firebase within your Android Studio project following the official documentation. This involves adding the necessary dependencies to your (build.gradle) file and configuring your Firebase project.
- **Creating a FirebaseVisionImage Object:** Once you have the captured or selected image, you need to convert it into a format suitable for ML Kit. Use the FirebaseVisionImage. From Bitmap (bitmap) method to create a FirebaseVisionImage object from the image bitmap.
- **Processing the Image:** Pass the FirebaseVisionImage object to the process Image method of the FirebaseVisionTextRecognizer instance. This method initiates the text recognition process on the image.
- **Extracting Recognized Text:** Once the process Image method completes, it returns a Task object. This task holds the result of the text recognition process. You can use callbacks or asynchronous operations to handle the completed task and extract the recognized text. The task object will provide a FirebaseVisionText object containing information about the detected text blocks and lines within the image.
- **Building the Final Extracted Text:** By iterating through text blocks, lines, and characters, you can extract the entire recognized text content from the image. You can then display this text on your app's UI or store it for further processing.

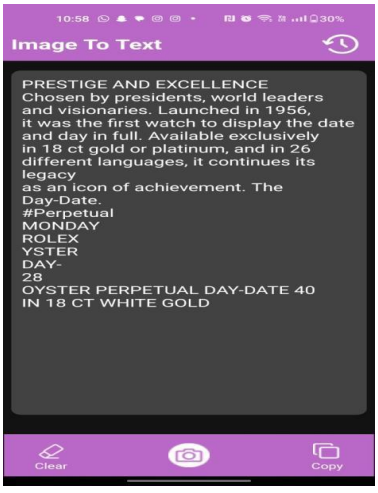

Figure 4: Text Extraction

2.4 Integrating the Copy Button

- **Adding a Button:** In your app's layout file (XML), add a button element representing the copy functionality. You can customize the button's appearance using different attributes like text, background color, and size.
- **Button Click Listener:** Set up a click listener for the copy button. This listener will be triggered whenever the user taps the button.
- **Accessing Extracted Text:** Within the click listener method, access the extracted text content from your text detection process. This text can be stored in a variable or retrieved from a data structure depending on your implementation.
- **Clipboard Manager:** Obtain an instance of the Android Clipboard Manager system service. You can achieve this by using the get System Service (Context.CLIPBOARD_SERVICE) method.
- **Creating a ClipData Object:** Construct a Clip Data object containing the extracted text as the label and the text itself as the content. You can use the Clip Data. PlainText class to create this object.
- **Setting the Clipboard Content:** Use the set Primary Clip method of the Clipboard Manager instance to set the newly created ClipData object as the current content of the clipboard.

Figure 5: Text Copied

2.5 Integrating the Clear Button

- **Adding a Button:** In your app's layout file (XML), define a button element for the clear functionality. You can customize its appearance using attributes like text, background color, and size.
- **Button Click Listener:** Set up a click listener for the clear button. This listener will be triggered whenever the user taps the button.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 3, pp 227-231, April-2024

https://doi.org/10.47001/IRJIET/2024.803032

- **Clearing Relevant Elements:** Within the click listener method, identify the UI elements you want to clear when the button is pressed. This might involve:
- **Image View:** If your app displays the captured image within an ImageView, set its image source to null or a placeholder image to clear the displayed image.
- **Text View:** If the app displays the extracted text in a TextView, set its text content to an empty string to clear the displayed text.

Figure 6: Text Cleared

2.6 Implementing the History Feature

- **Viewing Past Image Processing Results:** Users might want to revisit previously captured images and their corresponding extracted text for a variety of reasons. For instance, they might need to reference specific information from a document they scanned earlier. Perhaps they want to compare the results of different text extraction attempts on the same image, especially if the image quality was low or the text was challenging to recognize. Additionally, the history can be helpful for tasks like copying previously extracted text snippets or sharing them with others.
- **Managing Processed Data:** The history can provide users with options to manage the stored data, such as deleting specific entries or clearing the entire history.
- **Delete Individual Entries:** Users might want to remove specific entries from the history that are no longer relevant or needed. This could be helpful for managing storage space on the device, especially if the app stores thumbnails of processed images. The history view should provide a clear and easy way to delete individual entries, potentially with confirmation prompts to prevent accidental deletion.

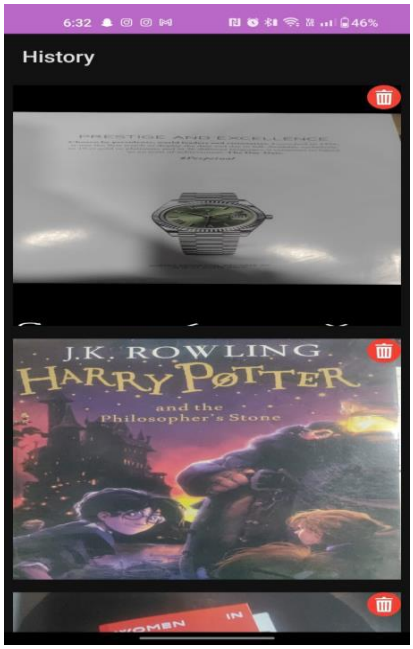

Figure 7: Image History

2.7 Deleting the History

- Deleting Captured Image: Allow users to discard an image they've captured before proceeding with text detection.
- Deleting History Entries: Enable users to remove specific entries or the entire history of processed images and extracted text.
- Confirmation Prompt: Before deleting the captured image, display a confirmation dialog asking the user to confirm their choice. This prevents accidental deletion of an image they might intend to process.

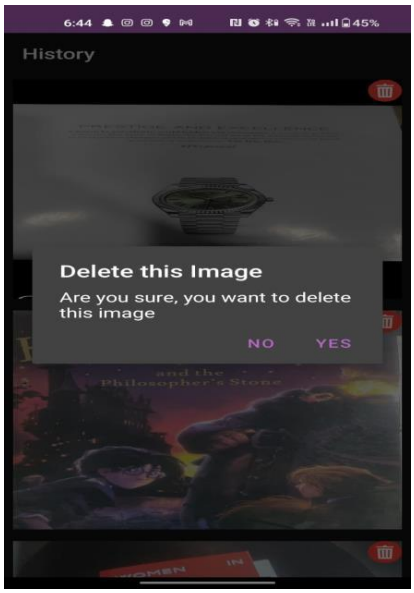

Figure 8: Deletion of Image

ISSN (online): 2581-3048
Volume 8, Issue 3, pp 227-231, April-2024
https://doi.org/10.47001/IRJIET/2024.803032

III. RESULTS AND DISCUSSIONS

We've accomplished the extraction of text from an image by employing cropping techniques, as demonstrated in Figure 1. Additionally, we've implemented multiple cropping, as illustrated in Figure 2, and successfully extracted text from these cropped sections, as shown in Figure 3. This process allows us to target specific areas of interest within the image and extract the relevant text, enhancing the efficiency and accuracy of our text extraction methodology.

IV. CONCLUSION

The article discusses an Android app developed for recognizing text from challenging images using basic image processing techniques. It evaluates the results both quantitatively and qualitatively across various images with diverse fonts and backgrounds. Furthermore, the performance of the app is compared with three existing commercial apps. The conclusion drawn is that the proposed app achieves good accuracy, surpassing previous apps in correctly identifying text from challenging images.

REFERENCES

[1] Jian Yuan, Yi Zhang, KokKiong Tan, Tong Heng Lee, Text Extraction from Images Captured via Mobile and Digital Devices, International Journal of Computational Vision and Robotics, 2009.

[2] SathiapriyaRamiah, Tan Yu Liong, Manoj Jayabalan, Detecting Text-Based Image With Optical Character Recognition for English Translation and Speech using Android, IEEE Student Conference on Research and Development (SCOReD), 2015.

[3] Nilesh Jondhale, Dr Sudha Gupta, Reading text extracted from an image using OCR and android Text to Speech Mobile, International Journal of Latest Engineering and Management Research (IJLEMR), vol. 03, pp. 64-67, 2018.

[4] Xiaoming Huang, Tao Shen, Run Wang, Chenqiang Gao, Text Detection and Recognition in Natural Scene Images," International Conference on Estimation, Detection and Information Fusion (ICEDIF), pp. 44-49, 2015.

[5] Qixiang Ye and David S. Doermann, Text Detection and Recognition in Imagery: A Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.37, 2015.

[6] Shalin A. Chopra, Amit A. Ghadge, Onkar A. Padwal "Optical Character Recognition" International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 1, January 2014.

[7] Dishank Rajesh Palan, Ghoshil Bharat Bhatt, Kinjal Jayesh Mehta, Kunal Jayesh Shavdia, Mansi Kambli "OCR on Android Travelmate", International Journal of Advanced Research in Computer and Communication Engineering ,Vol. 3, Issue 3, March 2014.

[8] Jin, "A Flash Card Android Application Development Applied with OCR technology," Helsinki Metropolia University of Applied Sciences, Thesis 26 May 2014.

[9] Computer Applications Volume 55– No.10, October 2012.

[10] Sonia Bhaskar, Nicholas Lavassar, Scott Green, "Implementing Optical Character Recognition on the Android Operating Systems for Business Cards" EE 368 Digital Image Processing.

Citation of this Article:

Prof. Rahul D. Dhongade, Srividya Ravishankar, Harshali B. Sabale, Abishek C. Verma, "Image to Text Conversion Using Android Studio", Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 8, Issue 3, pp 227-231, March 2024. Article DOI https://doi.org/10.47001/IRJIET/2024.803032

****

