iRiS : A Wearable Device for Reading Printed Text

Abel Saju Augustine*, Aswin Suresh Krishnan†, Dishit Dak‡, Sharon Sanjeev George§, Dr. Ramesh Kini¶
Email:*abelsaugustine@gmail.com, †aswinsureshk@gmail.com, ‡ddak7579@gmail.com, §sharon.george1993@gmail.com,
¶rameshkinim@rediffmail.com

***Abstract*—Visually Impaired (VI) people find it difficult to access information in the form of printed text. Existing solutions in the market these days make tradeoffs between processing speed, size and efficiency. We introduce a finger-mountable device, iRiS, that enables reading of printed text. Using an intelligent OCR-based algorithm coupled with real-time auditory and tactile feedback, iRiS provides the user with an assistive text-reading experience. iRiS is designed in such a way so as to permit easy setup-and-use with minimum calibration and high mobility. Testing and evaluation with VI people is yet to be performed.**

***Keywords*—Assistive technology; Visually Impaired; Real-Time OCR; Finger worn interface;**

I. Introduction

In today's world, information is present in many forms, the primary medium being printed text. In such a scenario, visually impaired people tend to fall at a disadvantage in terms of accessing information. This makes them depend on other alternative methods of reading like braille, text scanners, screen readers and many other pieces of technology that are cumbersome and provide a poor reading experience because of their slow processing speeds and poor accuracy. Preliminary studies conducted with visually impaired people signalled the need for a solution that can overcome the shortcomings mentioned above. With this project, our aim is to use technology to bridge the gap that deter the visually impaired from accessing information. Therefore, in this paper, we present our work from the past eight months in tackling this issue by building a mobile device, iRiS, that will allow visually impaired people to read text off printed materials in real-time. Some of the features of iRiS are as follows:

1) The prototype is mountable on a finger with all the necessary modules assembled into a single apparatus.
2) A point-to-read input mechanism that performs local sequential scanning of text with real-time auditory and tactile feedback.
3) iRiS employs assistive text-reading techniques that help the user keep track of where reading is performed.

Besides text reading, the device presents the user with some additional functions as well, namely, the translation of words from one language to the other and the use of an English Dictionary to find word definitions.

II. Related Works

iRiS is a wearable device that detects printed text by performing word detection in real-time, keeping the finger position as reference. We have also added guidance signals that help users to move up, down and to the next line. In addition to this, it also provides features like word definition and translation using built-in language packs. In **Table 1**, we present a comparison of recent methods for text-reading for the VI.

Although MIT FingerReader [7] and OrCam [6] employ a real-time point-and-read word detection system, most others do not. For instance, Zoom-Twix [8] scans the entire page and then reads the content and thus does not provide instant feedback and the kNFB K-Reader [2] works by taking a snapshot of a text passage that is then processed. Moreover, most of the devices do not have any guidance signals that aids the user to navigate to the next line upon reaching the end of a line. Amongst the text readers compared in the **Table 1**, none of the devices support word definition. k-Reader Mobile [2] is the only device that can translate between languages while JAWS [3], Eye-Pal ROL [4] and Zoom-Twix [8] are available in various languages. Most of the devices mentioned are portable and can read printed text. However, JAWS [3] is a desktop-based application and can only read on-screen text.

III. Hardware

iRiS is a finger-worn device that assists visually challenged people in reading printed text. The device chassis is fabricated using ABS material. The angled projection of the front side of the device allows the camera to be held at a fixed position, capturing both the fingertip and the printed text in a single frame. It, initially, locks the finger position and allows the user to point to the text that is to be read. The device captures the images through the Raspberry Pi camera module fitted in the hardware model, which is then converted to string format and later, gives a speech output. The processing was tested on desktops as well as the Raspberry Pi 2. The Raspberry Pi 2 allows us more portability and is a cheaper alternative. The hardware model also contains high brightness LEDs to illuminate the printed text.

For the use of this device by visually-challenged people, we must compensate for instances such as when the user strays from the printed line or reaches the end of line etc. Such situations require feedback, which is provided through tactile and auditory stimuli [1]. Two vibration motors are present to provide tactile feedback when the user strays above or below the printed line. Auditory signals are emitted when the user reaches the end of line. The user is then guided to the new line and a second tone is emitted at the start of the new line. These four signals, two tactile and three auditory signals, act as guidance signals for the users.

Device	Portability	Speed	OCR	Translation	Text Material
1. kNFB K-Reader Mobile [2]	YES	Processor dependent Not real-time	YES	YES	Reads most Printed documents
2. JAWS(Job Access with Speech) for Windows(Standard) [3]	NO	500 ms	YES	NO	Reads on-screen texts
3. Eye-Pal ROL [4]	YES	Typically three seconds or less	YES	NO	Standard printed pages and Glossy magazines
4. Voice Stick [5]: Text Scanning Device for VI	YES		YES	NO	Can read words and letters
5. OrCam [6] assistive eye glasses	YES	Real-time	YES	NO	Can read all printed text
6. MIT FingerReader [7]	YES	Real-time(<20 ms)	YES	NO	Printed and on-screen texts
7. ABiSee Zoom-Twix [8]	YES	20 pages/min	YES	NO	Printed Texts

TABLE I: Recent methods for Text-Reading for the VI

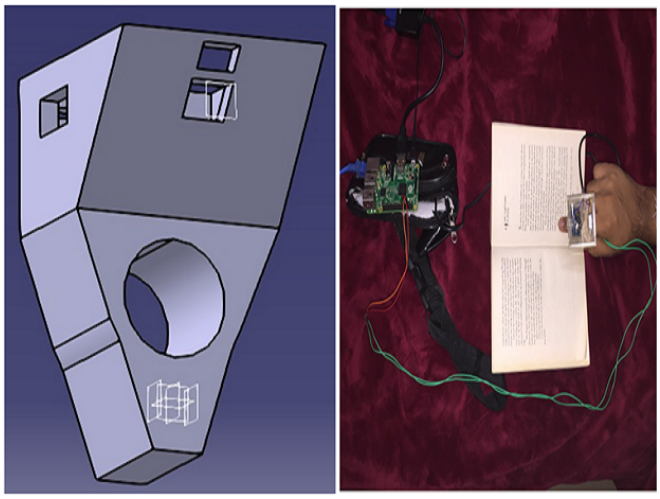

Fig. 1: Design Prototype and Assembled Hardware

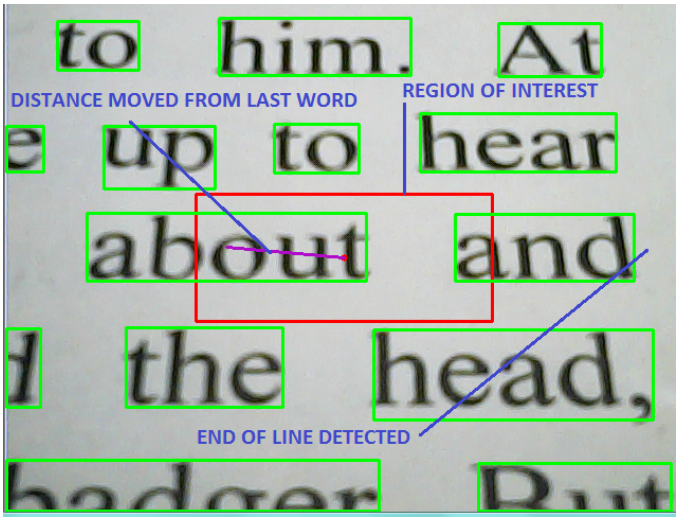

Fig. 2: Software detecting words and End of Line while reading. Last extracted word: **about**

IV. SOFTWARE

We developed a program that consists of text extraction and real-time image processing algorithms. It integrates Tesseract OCR and Python Text-To-Speech (TTS) and is done with the help of Open Source Computer Vision Library. The program supports three modes, namely, reading, translate and define. On start, the user is asked to choose the required mode of operation. The reading mode simply scans for text and provides a real-time auditory feedback. The translate mode provides a real-time translation of the detected text to any of the included language packs. An internet connection is mandatory for this mode to operate. The define mode provides the user with the definition of the scanned word along with all possible usages, exactly as in a dictionary.

A. *Working Mechanism*

The program calibrates by detecting the fingertip wherein the user is asked to hold the finger steady for a few seconds until the tip coordinates are acquired. These coordinates are later used for finding the Region of Interest (ROI) for text extraction. A close-up view of the printed text is required as input for text extraction. We scan the frame continuously for words and detect all the words in a given frame. Thereafter, we look for words that lie in the ROI and extract them. These words are then sent to the OCR Engine. The recognized text undergoes a dictionary check in order to ensure validity. Repetitive detection is avoided. If the recognized word is found valid, we invoke the TTS engine to utter the word.

*1) **Word Detection**:* To the captured frame, we apply image binarization and blurring in order to get clear text. Then, all the edges in the frame are calculated. Hough Transform (HT) is [10] used to find lines through each edge pixel at every possible angle. These lines are carefully chosen in such a way that they overlap the letters of the same word and no line cross-connects two distinct words in the text. The transformation between feature space and parameter space is done [9] during HT. For using HT, we rely on the fact that every point in the image space can lie on a line of the form r=xcos(θ) + ysin(θ) (where r is the length of a normal from the origin to the line and θ is the orientation of r with respect to the x-axis) in the Hough parameter space. For every black pixel, we find lines along the range of angles specified and then increment the accumulator by one. Finally, we retrieve a resultant matrix, which contains values indicating the number of points that lie on a particular line. Lines that contain more pixels will be attributed a higher value than those that contain fewer pixels. A low threshold is set to discard lines with small values. Maximum distance between two points to be on the same line is also chosen exquisitely so that the length of each line is restricted. Once this is established, we find the contours and draw bounding figures over the detected contours for the upcoming processes of extraction and recognition. **(Fig 4: Process 1)**

*2) **Word recognition**:* The extracted word is resized and passed onto the Tesseract OCR for the recognition process (image-to-text conversion). The returned string undergoes a dictionary check. If a match is found, then the word is stacked up for text-to-speech conversion. **(Fig 4: Process 2)**

*3) **Text-to-Speech conversion**:* When new data is found in the stack, it is emptied and the text is passed on to the TTS engine for text-to-speech conversion and the word is uttered aloud.

For transfer of data extracted to engine and between the two engines, we use a Last-In-First-Out (LIFO) approach by which we give higher priority to the most recently obtained data. If the program finds out that the user has moved ahead quickly and the earlier elements in the stack are of no use, then it is cleared and the process continues. **(Fig 4: Process 3)**

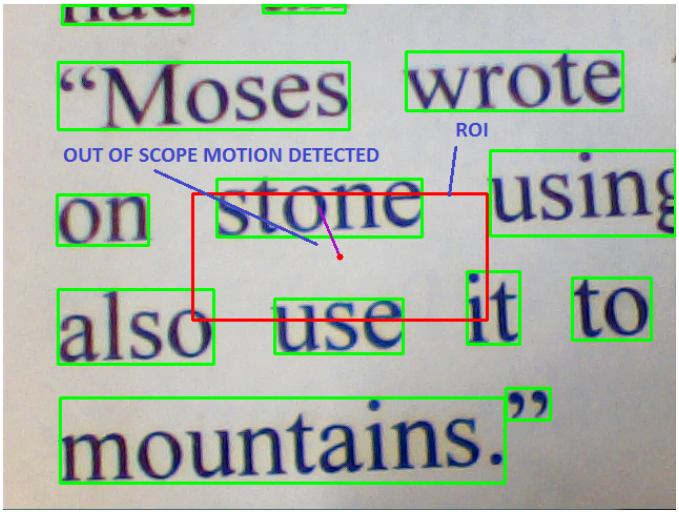

Fig. 3: Accidental motion below the line. Warning to move up is triggered

B. *Finger guidance mechanism*

We keep track of the position of the finger with respect to the current line of scan. If the user deviates to a position above or below, we trigger a tactile feedback, warning the user to return the finger to its correct path. When no more words are detected further ahead in the current line, an end of line warning is issued and the user is guided to the line just below. The user is expected to make a return path along this new line until the start of the line is reached. During this movement, the word extraction is at a halt and will continue after the user has successfully managed to maneuver the finger to the start of the line. To ensure successful transfer of finger to the next line, we keep track of all the words below the reading line and use a nearest neighbour algorithm to find out the last word of the next line.

Periodically, we check for End of Line and Start of Line conditions by searching for words to the left and right of the Region of Interest. If there are no more words to read in the current line, we trigger the End of Line condition. Also, the distance from the point of maximum interest (finger-tip position) to the center of the last detected word is continuously tracked to ensure the path of motion. If the user deviates to a position significantly higher or lower than what is allowed (set by threshold) with respect to the center of previous word, then a guidance mechanism is activated, which helps the user to return to the line.

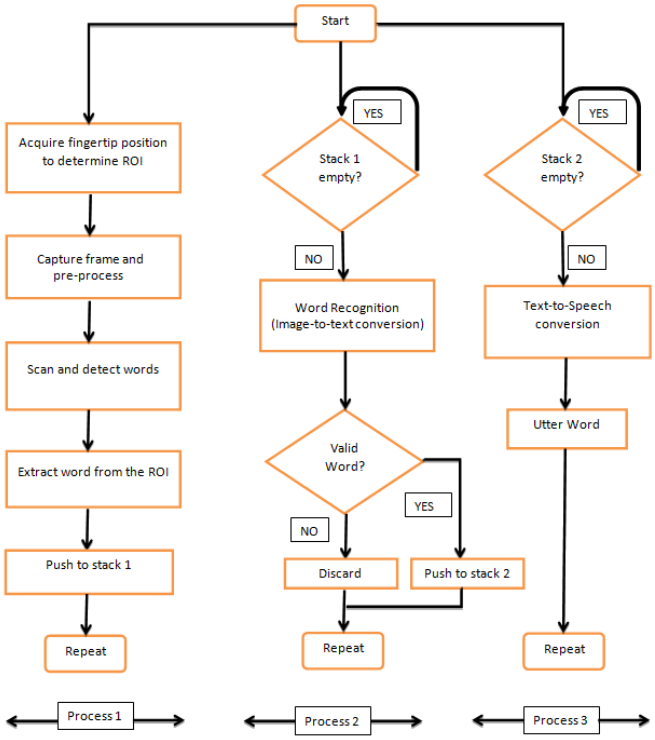

Fig. 4: Text Reading Algorithm Flowchart

C. *Compatability*

Our program runs on Windows and UNIX machines with high efficiency. It also runs on Raspbian OS but with higher processing time. A necessary condition for real-time feedback was to perform real-time processing with minimum run-time. For that, we make the processes of word detection and extraction, image-to-text conversion and text-to-speech conversion execute simultaneously. [11], thus reducing program lag and processing time.

V. Observations and Results

To evaluate the performance of our program, we make empirical estimations of the time required for execution of some of the core processes in the program. The time taken and the corresponding delay that occurs for specific functions such as word detection, word recognition and text-to-speech conversion is recorded. The following was the test sequence used for time/delay estimation:

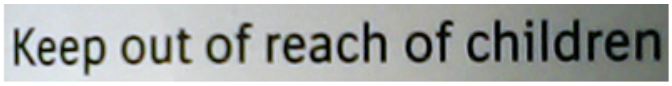

Fig. 5: Test Sequence

A. *Detection Delay*

The first delay observed is the **Detection Delay**, which is the time taken for word detection; the time taken to capture a frame, crop out the image of target word and store.

The following were the readings recorded for each of the words from the test sequence in respective order:

Word 1 - **0.039 seconds**
Word 2 - **0.047 seconds**
Word 3 - **0.058 seconds**
Word 4 - **0.044 seconds**
Word 5 - **0.058 seconds**
Word 6 - **0.044 seconds**

Average Delay = **0.048 seconds**

B. Recognition Delay

The second delay observed is the **Recognition Delay**, which is the time taken for word recognition; time taken from obtaining the image of word from the queue to recognizing the text.

The following were the readings recorded for each of the words from the test sequence in respective order:

Word 1 - **0.202 seconds**
Word 2 - **0.217 seconds**
Word 3 - **0.201 seconds**
Word 4 - **0.211 seconds**
Word 5 - **0.201 seconds**
Word 6 - **0.197 seconds**

Average Delay = **0.205 seconds**

C. Speech Delay

The third delay observed is the **Speech Delay**, which is the time taken for text-to-speech conversion; time taken from the point when the recognized text is obtained from the queue to the point when the word is spoken out.

The following were the readings recorded for each of the words from the test sequence in respective order:

Word 1 - **1.643 seconds**
Word 2 - **1.418 seconds**
Word 3 - **1.424 seconds**
Word 4 - **1.718 seconds**
Word 5 - **1.424 seconds**
Word 6 - **1.781 seconds**

Average Delay = **1.568 seconds**

D. Comparision of Delays

1) Total delay in case of serial processing = **1.821 seconds**
2) Worst case delay in case of parallel computation = **1.568 seconds**
3) Reduction = **0.253 seconds**
4) Percentage reduction in delay due to parallel processing = **13.89** %

VI. Conclusion

We introduced a novel idea and implemented it in the form of a device that enables scanning of printed text and generates real-time audio output of the recognized text as speech. The efficiency and performance of the device under normal working conditions is collated in the previous sections. iRiS is novel in its style of operation due to its periodic feedback mechanism. A user review of iRiS is necessary in order to further evaluate and correct any inaccuracies that might have occurred in the developmental stages of either the hardware or the software or both.

References

[1] FingerReader: A Wearable Device to Explore Printed Text on the Go, http://fluid.media.mit.edu/sites/default/files/FingerReaderCHI15

[2] KNFB kReader mobile, 2010. http://www.knfbreader.com/products-kreader-mobile.php.

[3] JAWS Screen Reader,2008. http://www.freedomscientific.com/Products/Blindness/JAWS.

[4] ABiSee. EyePal ROL, 2013. http://www.abisee.com/products/eye-pal-rol.html.

[5] Sungwoo Park. Voice Stick, 2008. http://www.tuvie.com/voice-stick-portable-text-scanning-device-for-the-visually-impaired/

[6] OrCam, 2013. http://www.orcam.com/.

[7] FingerReader : A Wearable Device to Support Text Reading on the Go. http://fluid.media.mit.edu/sites/default/files/paper317.pdf

[8] ABiSee. Zoom-Twix, 2007. http://www.abisee.com/products/zoom-twix.html

[9] Multifont Arabic Character Recognition Using Hough Transform and Hidden Markov Models by Nadia Ben Amor and Najoua Essoukri Ben Amara.

[10] A Hough Transform based Technique for Text Segmentation by Satadal Saha, Subhadip Basu, Mita Nasipuri and Dipak Kr. Basu.

[11] http://chimera.labs.oreilly.com/books/1230000000393/ch12.html , Python Cookbook , Chapter 12- Concurrency.