

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/349447419>

Design of an Automatic Reader for the Visually Impaired Using Raspberry Pi

Chapter · February 2021

DOI: 10.1007/978-981-15-7533-4_14

CITATION

1

READS

2,124

4 authors, including:



Nabendu Bhui

National Institute of Technology Sikkim

4 PUBLICATIONS 19 CITATIONS

[SEE PROFILE](#)



Pratyay Kuila

National Institute of Technology Sikkim

61 PUBLICATIONS 2,133 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Application of heuristics/meta-heuristics for predictive maintenance of manufacturing systems [View project](#)



Workflow Scheduling [View project](#)

Chapter 14

Design of an Automatic Reader for the Visually Impaired Using Raspberry Pi



Nabendu Bhui, Dusayanta Prasad, Avishek Sinha, and Pratyay Kuila

1 Introduction

Visually impaired people are incapable of doing most of the visual tasks and out of them reading is the most important task. They face many troubles due to inaccessible infrastructure and social challenges. Visually impaired people always read any document through the Braille machine, where text is converted to Braille literature. Braille technology uses combinations of raised dots to spell letters and numbers. Actually Braille is not a language—it's a system of writing. Transfer any paper or book into Braille is time taking and complicated. It is not possible to transfer daily information into Braille literature and for that reason they are not updated with the community [1]. The most valuable thing for a disabled person is gaining independence. A blind person can lead an independent life with some specifically designed adaptive things for them. There are lots of adaptive equipment that can enable a blind person to live their life independently, but they are not easily available in the local

N. Bhui (✉) · P. Kuila

Department of Computer Science and Engineering, National Institute of Technology Sikkim,
South Sikkim 737139, India

e-mail: nabendu1421bhui@gmail.com

P. Kuila

e-mail: pratyay_kuila@yahoo.com

D. Prasad

IBM India, Bengaluru 560045, India

e-mail: dusayantaprasad@gmail.com

A. Sinha

Department of Computer Science and Engineering, I.K.G.P.T.U., Jalandhar, Kapurthala, Punjab
144603, India

e-mail: iavisheksinha@gmail.com

shops or markets. A blind person needs to hunt and put much effort to get each equipment, which can take them one step closer toward independence. The main objective to develop this device is to read a printed paper or book which is written in multi languages (English, Bengali and Hindi). All the processing is to be done on a low power consuming computer: Raspberry Pi. By using this reader they fell independent because they don't need any third person. They can feel most of the things by touch and for that reason we built our project based on switches. By pressing the switches any person can control this reader machine.

2 Related Works

In [1], the authors proposed a prototype which helps any blind person to listen any text images of English and Tamil language. Reading of text is happened by taking images of the text and converting the image to audio output in the above mentioned languages. This was done with the help of Raspberry Pi 3 model B, a web-camera, Tesseract OCR (Optical Character Recognition) engine and Google Speech API (Application Program Interface) which is the text to speech engine. The disadvantage of the system is that it produces unclear output with incorrect regional accent and also problem in speech for the Tamil language. In [2], the authors proposed a smart book reader for visually challenged based on Optical Character Recognition. The Raspberry Pi 3 kit and Raspberry Pi Camera Module are used here. The Google Tesseract is used for OCR and Pico is used for text to speech in this project. On pre-processing stage this method uses binarization, de-noising, deskewing and segmentation techniques for image clarity purpose. Sometimes a mobile application allows blind people to "read" text by using a Photo-to-speech application. A combination of OCR and Text-to-Speech (TTS) framework is integrated. With the help of smart phone take a picture and hear the text that exists in the picture. Some drawbacks are, it's not providing any automatic system for capturing images [3]. Generally, optical character recognition (OCR) recognizes the texts from the data of captured images. Conversion of scanned or photographed document into electronic transcript is happening here. Digital text synthesized into voice by using the technology of speech synthesis (TTS) and played through any audio system. This system is constructed by using raspberry pi, HD camera and Bluetooth headset [4]. In [5], the authors proposed a model which enables any user to hear any text in real-time without taking any pain of reading. The whole process is established with the help of OCR (Optical Character Recognition) and TTS (Text-to-Speech) frameworks. This combination is happening into Raspberry Pi v2. The disadvantage of the system is that captured image was blurred and for that reason sometimes OCR gives wrong result. The proposed system in [6] guaranteed to read the text present in anywhere for assisting blind persons. The disadvantage of the system is that spell problem for OCR output. In [7], the authors proposed a camera-based label reader for blind persons to read any text. A camera is used for capturing the image of text or board, and then the image is pre-processed and separates the label from that processed image with the help of open CV library. After

identifying the text, pronunciation of text is happened through voice. A motion-based method is applied to detect the object or the text which is written on the board or hoarding or in any places. Vasanth K et al. [8] proposed a self-assistive device where any live streaming speech is sent to Google API and after conversion of speech to text, speak the speech via speaker and displaying the result onto the LCD screen. But, a good internet facility is needed for this method purpose. In [9], the authors designed a voice based navigation system for blind people in a hospital environment. With the help of ultrasonic sensors and an RFID reader (which are interfaced with Raspberry Pi3) an obstacle avoidance system is designed to locate the exact place in the hospital.

Most of the models depend on good internet and most of these models have OCR problems, due to these shortcomings in our proposed method, there is nothing related to internet. Also, there are some processing steps for better result in OCR.

3 System Design

3.1 Working Principle

There are three **push buttons** into our proposed model. The first push button is to choose particular language, the second button is to capture images and the third button is to read the text. Flowchart for proposed model is shown in Fig. 1 and the working principle of proposed system is mentioned below.

- At first, we put a paper or book under the Raspberry Pi camera.
- Press the first button and count how many times button is pressed, because for every count there is a language.

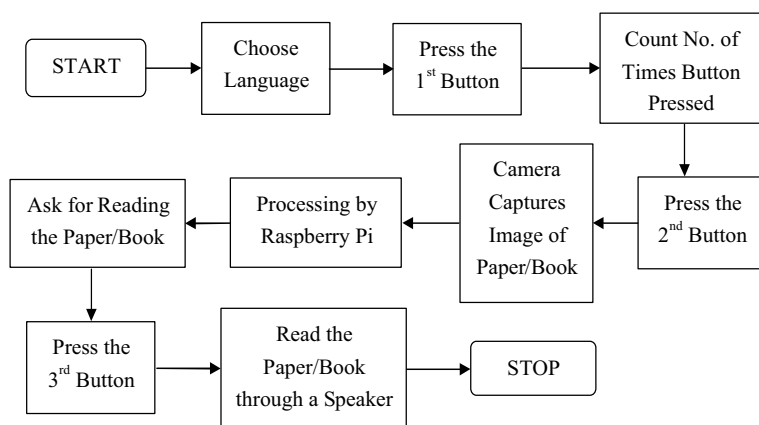


Fig. 1 Flowchart of the approach

- Then by pressing the second button image is captured with the help of a **Raspberry Pi Camera**. Camera module takes an image of the paper or book.
- Converting that image into Grayscale in pre-processing stage, by using **OCR (Tesseract)** convert image into text and then read the written text by using **TTS (eSpeak)**, these three processes are done through Raspberry Pi.
- At last, it asks for reading the paper or a book and for that reason press the third button. After pressing last button, it reads the given text by using a **Speaker**.

3.2 *Raspberry Pi 4 Model B*

Raspberry Pi 4 Model B is the latest product in the popular Raspberry Pi computer versions. Processor speed, memory, connectivity and multimedia performance are better than previously released Raspberry Pi versions. The Raspberry Pi [10] Foundation provides Raspbian, a Debian-based Linux distribution for download. It has Broadcom BCM2711, quad-core Cortex-A72 (ARM v8), 64-bit 1.5 GHz processor; 1 GB, 2 GB or 4 GB LPDDR4 (depending on model) memory; LAN, Bluetooth 5.0, Gigabit Ethernet, $2 \times$ USB 3.0 ports and $2 \times$ USB 2.0 ports for connectivity; 40 general-purpose input/output (GPIO) pins and a Micro SD card slot for loading operating system and data storage.

3.3 *Used Components in Reader*

Raspberry Pi Camera v2 is used to capture images of text file. High quality 8 megapixel Sony IMX219 image sensor with fixed focus lens is present in Camera v2. On the upper side of the raspberry pi board a small camera socket is present, which is built for camera interfacing [10]. The **USB Mini Speaker** is used to play the audio generated by a speech synthesizer. It has sleek surfaced, USB power input with 3.5 mm Jack and Power Jack: USB 2.0 (Plug and Play). A **Breadboard** is used to make up temporary circuits for testing or to try out an idea. No soldering is required so it is easy to change connections and replace components. Parts are not damaged and can be re-used afterward. A **Push button** is a simple type of switch that controls an action in a machine or some type of process. Push buttons (switches) are often part of a bigger system and are connected through a mechanical linkage; by this linkage we connect our push button with the Raspberry Pi system. Push buttons are putting into the breadboard. **Jumper Wires** are typically used to connect GPIO pins with the push button. Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires typically come in three versions: male-to-male, male-to-female and female-to-female. In our case, we have used male-to-female jumper wires to create a connection between the breadboard and the Raspberry Pi.

3.4 Proposed Hardware Model

By using Raspberry Pi and all the components we have proposed following model:

In the following diagrams we have shown the whole project model with a different view: full frontal view, near view & front View in Fig. 2 and side view, corner view and top view in Fig. 3. Actually, here we create a box type model, where we put our paper or book under raspberry pi camera module and after that all the processes are done by using Raspberry Pi. We create this box because of fixed position of the paper or book. For this fixed position we can capture the right image of given texts.

Raspberry Pi Camera is interfaced to the Raspberry Pi via its customized slot—CSI Camera Connector. Speakers are interfaced to the Raspberry Pi via 3.5 mm audio out socket. The Power supply is given to the Raspberry by using power adapters. The front side of the Raspberry Pi camera module is downward and we put our pages or book under the camera module for taking images.

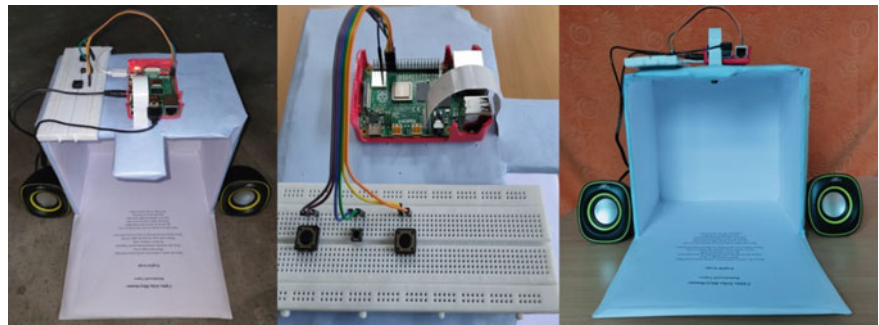


Fig. 2 Proposed Model (Full frontal view, Near view and Front view)



Fig. 3 Proposed Model (Side view, Corner view and Top view)

Table 1 GPIO Interfacing

Pin No.	Pin name	Description
6	Ground	Push Button 2 (Side 1)
9	Ground	Push Button 3 (Side 1)
11	GPIO 17	Push Button 3 (Side 2)
12	GPIO 18	Push Button 2 (Side 2)
13	GPIO 27	Push Button 1 (Side 2)
14	Ground	Push Button 1 (Side 1)

3.5 *GPIO Interfacing*

General-purpose input/output (GPIO) [11] pins act as input or output and controllable by the user at run time. Which pin work for which push button is described in the following table (Table 1). Actually, here we use three push buttons and these buttons are connected to GPIO pins by using jumper wires. 13 and 14 number GPIO pins are used for the first push button, 6 and 12 numbers GPIO pins are used for the second push button. 9 and 11 number GPIO pins are used for the third push button. Where, 6, 9 and 14 number pins are used for ground purpose and 11, 12 and 13 number pins are used for input-output purpose.

3.6 *Proposed Approach*

Blind persons can feel everything and for that reason they can feel by touch any material. For that reason we used three push buttons for controlling the reader device. The first button is used for selecting the language, the second button is used for capturing image and another button is used for reading purposes. Every switch or push buttons are connected with the Raspberry Pi GPIO pins by using jumper wires. And for that reason we have written the pseudo code (Algorithm 1) in python programming. This proposed model support English, Bengali and Hindi languages.

After capturing images at first we process that image for better result. We convert the RGB (specified with: red, green, blue) image into a grayscale image (Algorithm 2). Due to light effect or coloring effect, original image (RGB) may give maximum wrong result. For that reason we convert it into grayscale [12, 13]. Grayscale images are also used for measuring the intensity of light in images, using the following equation: ‘ $Y = 0.299R + 0.587G + 0.114B$ ’; where, Y is the luma of an image. Luma represents the achromatic image (gray image, an intermediate color between black and white) [14, 15]. After that the image is converted to 256 gray levels black and white image. The image has two backgrounds—a dark background on the top and bottom, and a brighter background between them. The binarization preserves the two backgrounds, but deletes the unwanted pattern behind the text. Conversion of a gray-scale image into binary image is happening through thresholding. In this


```

15: end if
16: if (B3 = Pressed) then
17: if (B1press = 0) then
18:     if (B2press = 0) then
19:         Press the Second Button for capturing image.
20:     end if
21:     Press the First Button to choose your language.
22: end if
23: if (B1press = 1) then
24:     if (B2press = 1) then
25:         Read Button Pressed.
26:         Call Subprocess("audio.sh" shell script).
27:     end if
28: end if
29: end if

```

Algorithm 2 : Python Program for Converting Original Image to Grayscale with Binarization.

Input:

I = Image File, which is taken using Raspberry Pi Camera

Output:

I = Image File, which is generated after binarization.

```

1: if (I = Exist) then
2:     Converting original image to Grayscale.
3:     Calculating Weighted Average for RGB pixel:
4:     0.299*pixel[0] + 0.587*pixel[1] + 0.114*pixel[2]
5:     Convert To Grayscale by using Weighted_Average.
6:     Binarizing the Grayscale image.
7:     Convert grayscale image to binary by thresholding.
8:     Save the image.
9: end if

```

Algorithm 3 : Shell Script for OCR(ocr.sh).

Input:

- (1) C = Total Clicks, which are generated from “switch.py”python program
- (2) I = Image File, which is generated after binarization
- (3) For every Language: c1=1, c2=2 and c3=3; where c1 for English, c2 for Bengali and c3 for Hindi

Output:

Generated text file for English/Bengali/Hindi Language (“OutputLanguage.txt”).

```

1: if (C = 0) then
2:     Press the First Button to choose your language.
3:     Exit.
4: end if
5: if (C > 3) then
6:     Maximum press must be Three.
7:     Exit.
8: end if
9: Capturing Image using Raspberry Pi Camera.
10: if (I = Exists) then
11:     Convert Image.
12: end if
13: Running the Tesseract-OCR Engine.
14: if (C = c1) then
15:     Tesseract converts “img.jpg”to “EnglishOutput.txt”.
16: end if
17: if (C = c2) then
18:     Tesseract converts “img.jpg”to “BengaliOutput.txt”.
19: end if
20: if (C = c3) then
21:     Tesseract converts “img.jpg”to “HindiOutput.txt”.
22: end if

```

Algorithm 4 : Shell Script for Playing Audio(audio.sh).

Input:

O = Generated text file (“OutputLanguage.txt”)

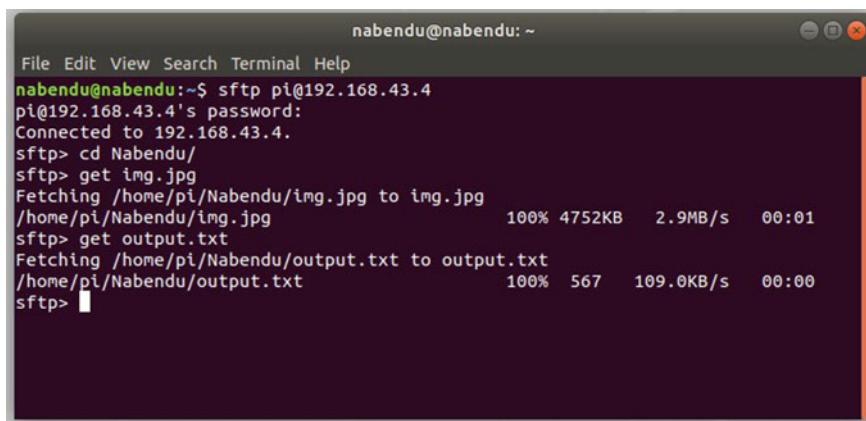
Output:

Audio to speak the output text.

```

1: if (O = Exist) then
2:     Speak “OutputLanguage.txt”text file by using eSpeak.
3: end if

```



```

nabendu@nabendu: ~
File Edit View Search Terminal Help
nabendu@nabendu:~$ sftp pi@192.168.43.4
pi@192.168.43.4's password:
Connected to 192.168.43.4.
sftp> cd Nabendu/
sftp> get img.jpg
Fetching /home/pi/Nabendu/img.jpg to img.jpg
/home/pi/Nabendu/img.jpg          100% 4752KB   2.9MB/s   00:01
sftp> get output.txt
Fetching /home/pi/Nabendu/output.txt to output.txt
/home/pi/Nabendu/output.txt       100%   567    109.0KB/s  00:00
sftp>

```

Fig. 4 Accessing File

Here, the first algorithm shows how to simply convert the image; the second algorithm shows how all push button procedure is working on, the third algorithm shows how text is converted into audio and fourth algorithm shows how original text is generated from the image with the help of an OCR framework for a particular language.

4 Result and Discussion

4.1 *Getting Input Images and Output Text*

To show all outputs for the paper ‘Design of an Automatic Reader for the Visually Impaired using Raspberry Pi’, we connect a Linux system by using common Wi-fi network and with the help of Secure File Transfer Protocol (SFTP). After following all the steps, we get some outputs in the form of audio. Generated outputs are not possible to show normally, for that reason we have taken out all the outputs from the Linux system by using SFTP and extracting procedure is shown in the following Fig. 4.

4.2 *Extracted Images and Generated Output*

It is not possible to show the audio files normally, for that reason we have taken the screenshots of the generated text outputs. Out of these languages we have shown some outputs of capturing images with its output text file. All screenshots, which are taken from Linux System, are shown below.

After applying OCR (Tesseract), captured original image converted into text format. Captured original images and screen shots of generating text output in English, Bengali and Hindi Languages are shown in Figs. 5, 6 and 7 accordingly.

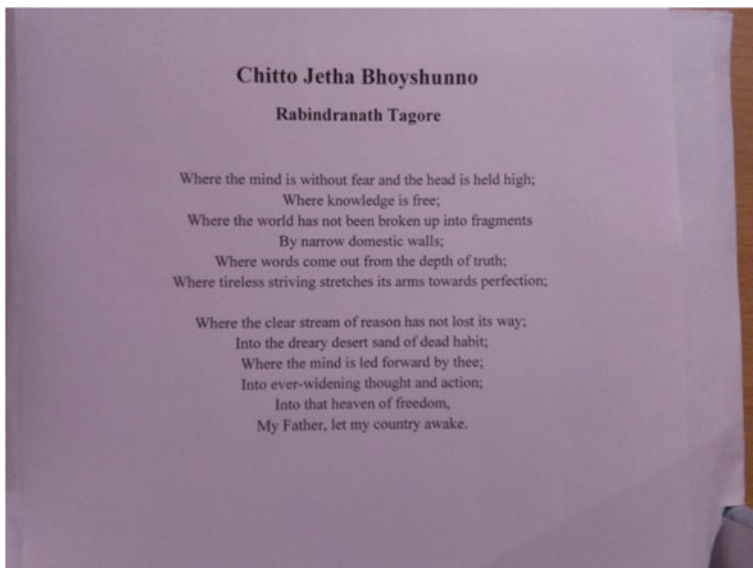
5 Conclusion and Future Work

The paper 'Design of an Automatic Reader for the Visually Impaired using Raspberry Pi' stepwise improved over some similar type projects. This model is built by using different parts: image is captured by using a Raspberry Pi camera, recognize the text by using Tesseract OCR framework and after that read the text through eSpeak TTS. To create a more efficient model with good outcome, processing and optimization are more important. Sometimes OCR gives incorrect text due to processing problem and for that reason final result represents a meaningless text. Due to this reason preprocessing is an important part in the whole system. Actually, if characters are cleared, big then there is no problem, but, due to light issue or small character or presence of images in between text gives little bit unexpected results. In our proposed method due to binarization, it gives better result over English, Bengali and Hindi languages; also, there is no need of internet.

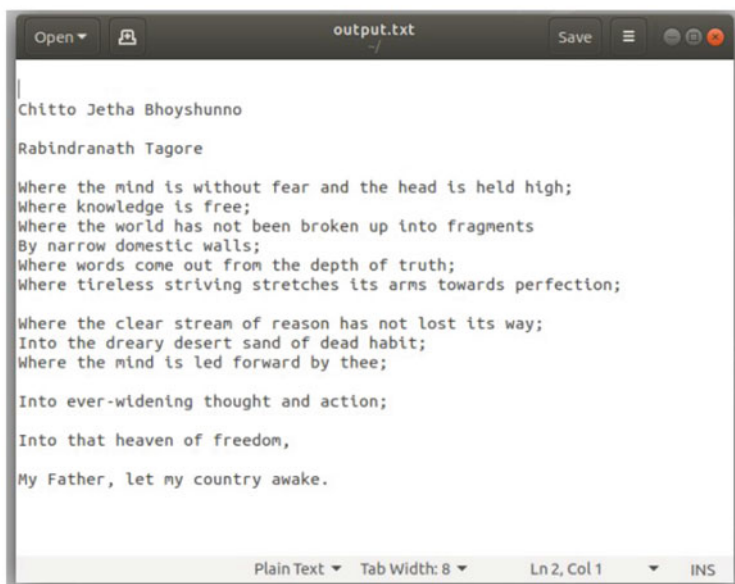
The proposed model is already in a feasible state of use due to the fixed box type model. But there are some future works for this model:

- Recognizing process of large texts is sometimes slow, for that reason introducing of some distributed technique is reserved for future work.
- Now this model can read only English, Bengali and Hindi language, but in future it should allow many other languages.
- The increment of the menu options which allow the user to play and pause the audio containing synthesized text.
- If any image or diagram is present in between text, then recognition of that image or diagram is an important task, which is also reserved for future work.

By extending these future works, many numbers of people will get more benefits from this reader.



(a)



(b)

Fig. 5 Extracted Image and Generated Output for English Language: (a) Captured Image and (b) Output Text

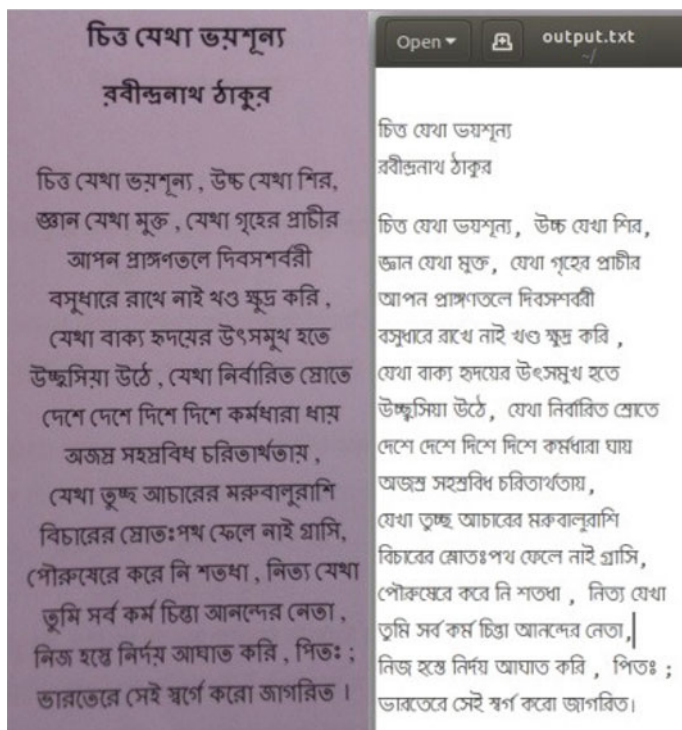


Fig. 6 Extracted image and generated output for Bengali language: captured image and output text

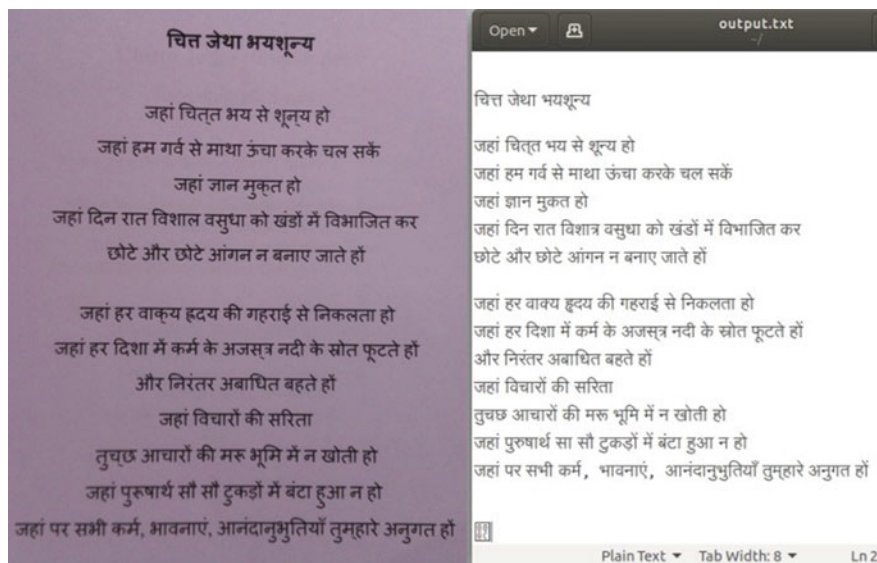


Fig. 7 Extracted image and generated output for Hindi language: captured image and output text

References

1. Akila IS, Akshaya B, Deepthi S, Sivadharshini P (2018) A text reader for the visually impaired using raspberry pi. In: Second international conference on computing methodologies and communication (ICCMC), IEEE, pp 778–782
2. Sonth S, Kallimani JS (2017) OCR based facilitator for the visually challenged. In: International conference on electrical, electronics, communication, computer, and optimization techniques (ICEECOT), IEEE, pp 1–7
3. Neto R, Fonseca N (2014) Camera reading for blind people. *Procedia Technol* 16:1200–1209
4. Thiyagarajan S, Kumar GS, Kumar EP, Sakana G (2018) Implementation of optical character recognition using raspberry pi for visually challenged person. *Int J Eng Technol* 7(3.34):65–67
5. Balaramakrishna JN, Tech M, AP WGD, Geetha MJ. Smart text reader from image Using OCR and openCV with raspberry PI 3
6. Nagaraja L, Nagarjun RS, Nishanth MA, Nithin D, Veena SM (2015) Vision based text recognition using raspberry PI. *Int J Comput Appl* 975:8887
7. Raja A, Reddy KD (2015) Compact camera based assistive text product label reading and voice output for visually challenged people. *IJITECH*
8. Vasanth K, Macharla M, Varatharajan R (2019) A self assistive device for deaf and blind people using IoT. *J Med Syst* 43(4):88
9. Reshma A, Rajathi GM (2018) Voice based navigation system for visually impaired people using RFID tag. In: International conference on intelligent data communication technologies and Internet of Things. Springer, Cham, pp 1557–1564
10. Teach, learn, and make with raspberry pi—raspberrypi. <https://www.raspberrypi.org/>. Last accessed 07 Dec 2019
11. Gpio—raspberrypi documentation. <https://www.raspberrypi.org/documentation/usage/gpio/>. Last accessed 2019/12/07
12. Python programming for raspberry pi, sams teach yourself in 24 hours—Richardblum, Christine bresnahan—Google books. https://books.google.co.in/books/about/Python_Programming_for_Raspberry_Pi_Sams.html?id=RYGsAQAAQBAJ&redir_esc=y. Last accessed 07 Dec 2019
13. Camera—processing for pi, <https://pi.processing.org/tutorial/camera/>. Last accessed 2019/12/07
14. In image processing applications, why do we convert from rgb to grayscale?—quora. <https://www.quora.com/In-image-processing-applications-why-do-we-convert-from-RGB-to-Grayscale>. Last accessed 2019/12/07
15. Grayscale—wikipedia. <https://en.wikipedia.org/wiki/Grayscale>. Last accessed 2019/12/07
16. Glossary—pixel values. <https://homepages.inf.ed.ac.uk/rbf/HIPR2/value.htm>. Last accessed 2019/12/07
17. <https://www.research.ibm.com/haifa/projects/image/glt/binar1.html>. Last accessed 2019/12/07
18. <https://www.programcreek.com/python/example/89427/cv2.threshold>. Last accessed 2019/12/07