

# Cloud Enabled Text Reader for Individuals with Vision Impairment

Abul K. M. Azad<sup>1\*</sup>, Mohammed Misbahuddin<sup>2</sup>

<sup>1</sup>Department of Technology, Northern Illinois University, DeKalb, USA

<sup>2</sup>Department of Electrical Engineering, Northern Illinois University, DeKalb, USA

Email: \*aazad@niu.edu, misbah4064@gmail.com

**How to cite this paper:** Azad, A.K.M. and Misbahuddin, M. (2017) Cloud Enabled Text Reader for Individuals with Vision Impairment. *Advances in Internet of Things*, 7, 97-111.

<https://doi.org/10.4236/ait.2017.74007>

**Received:** July 25, 2017

**Accepted:** September 2, 2017

**Published:** September 5, 2017

Copyright © 2017 by authors and Scientific Research Publishing Inc.

This work is licensed under the Creative Commons Attribution-NonCommercial International License (CC BY-NC 4.0).

<http://creativecommons.org/licenses/by-nc/4.0/>



Open Access

---

## Abstract

The paper describes the development of a text reader for people with vision impairments. The system is designed to extract the content of written documents or commercially printed materials. In terms of hardware, it utilizes a camera, a small embedded processor board, and an Alexa Echo Dot. The software involves an open source text detection library called Tesseract along with Leptonica and OpenCV. The system in its current version can only work with English text. By using the Amazon cloud web services, a skill set was deployed, which would read aloud the detected text utilizing an OpenCV program via the Alexa Echo Dot. For this development, a Raspberry Pi was utilized as the embedded processor system.

## Keywords

Text-to-Speech Converter, Cloud Computing, Image Processing, Embedded Processor, Internet of Things

---

## 1. Introduction

It is challenging for an individual with vision-impairment to extract information from a printed document. Either they have to use a Braille version of the document or someone has to read the document for them. Over the past few decades, machine reading has grown extensively with a number of practical applications [1]-[6]. A unique insight into the process in text-to-speech synthesis is reported by Kind [7]. In this work, a large number of competing techniques have been constructed to make direct comparison between the techniques. In one approach, researchers used LabVIEW to implement an Optical Character Recognition (OCR) based speech synthesis system [8]. One of the limitations of this is

the cost involved with the LabVIEW software itself. In another work, researchers surveyed methods related to character recognition as well as approaches used for text-to-speech conversion for machines [9]. A similar application is a portable bar code reader; designed to help people with vision impairment to identify different products in an extensive product database. This allows one to access product information through speech [10] [11] [12]. A limitation is that it is challenging for a user who is visibly impaired to position the bar code correctly for the reader.

Another approach to assist a person with vision impairment with this is to extract the text from a printed document with a camera and then process it to produce an audible sound. The first part of this can be done by obtaining an image of the printed material and using an OCR technique, while the second part can be implemented through speech synthesis and reproduction of corresponding sound. Ferreira and his co-workers adopted this approach and tested the performance without any control on the camera settings for font type and size as well as for the background [13]. OCR is the process of converting images of printed or handwritten texts (numerals, letters, and symbols) into a computer format text and has become one of the most successful applications of technology in the field of pattern recognition and artificial intelligence [14] [15]. Similarly, speech synthesis is the artificial generation of human speech and is widely used for text-to-speech conversion [16] [17] [18].

All these reported systems have one limitation; they require a standalone dedicated computing system for processing. This makes the system expensive as well as running a larger physical footprint, thereby making it difficult to carry for everyday use. To address this issue this paper describes the design, development, implementation, and evaluation of a system, in which an individual only needs to carry a small hand held system (camera and embedded processor board). The complete system will cost around US\$120. A large volume of the processing is performed over the cloud. The operation of the system involves text extraction from an image and converting the text-to-speech utilizing Amazon cloud. The Amazon cloud is formally known as Amazon Web Services (AWS). On the user side, the system involves a camera, an embedded processor, and an Amazon Echo Dot. Dedicated software has been developed for the embedded processor and AWS. The device is very portable, can be deployed within few minutes, and only requires an Internet access. The developed system then tested for performance in terms of text font, background color, text noise level, and commercially printed materials.

## 2. System Design

The system involved both hardware design and software development (**Figure 1**). The software development part is composed of image acquisition, image pre-processing, and image text extraction; it invokes Alexa services and pass the text to Alexa speech services.

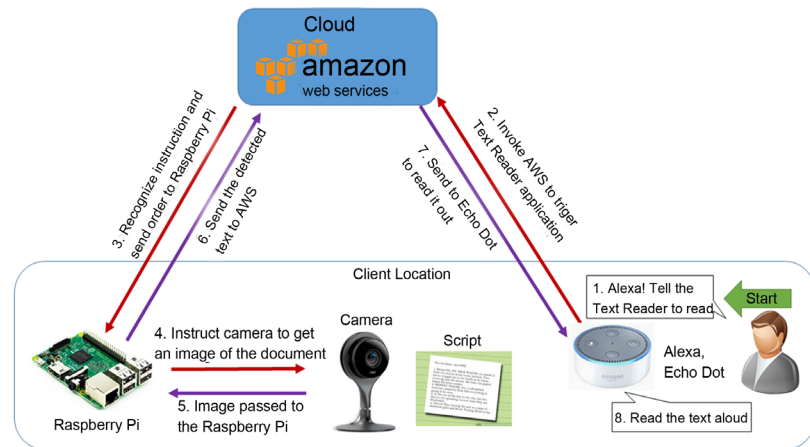


Figure 1. System configuration.

## 2.1. System Configuration

The hardware system is composed of a USB camera, a Raspberry Pi, and an Amazon Echo Dot (2<sup>nd</sup> generation). The Raspberry Pi runs the Raspbian Jessie OS 4.4, with OpenCV 2.3 installed along with Python 2 for compilation. The camera is directly connected with the Raspberry Pi. Both the Raspberry Pi and Amazon Echo Dot are connected to the Internet. A flow diagram of the system is shown in Figure 1 and a picture of the developed system is provided in Figure 2. The user triggers the system with a voice command via the Alexa Echo Dot. This invokes the AWS to instruct the Raspberry Pi to run the text reader application. The Raspberry Pi then instructs the camera to take an image of the script that is placed in front of it. The collected image is received by Raspberry Pi and is processed to extract the text. The software details of the text reader application and AWS are provided in Sections 2.2 and 2.3.

## 2.2. Image Processing

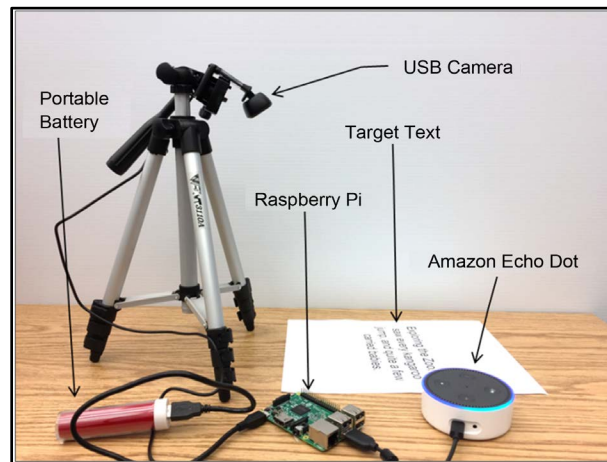
The image processing activity involves the initial image acquisition. This is followed by image pre-processing and subsequent image text extraction. All of these are implemented within a Raspberry Pi.

### 2.2.1. Image Acquisition

The first step is the image acquisition from a printed document that needs to be read. This is done with either a high definition (HD) camera or a normal webcam. The camera is directly connected to the Raspberry Pi, when the Raspberry Pi is connected with the web. An HD camera is preferred to enhance the image quality.

### 2.2.2. Image Pre-Processing

This involves removal of noise with appropriate filtering. It utilizes morphological image filtering using techniques like erosion and dilation, generating the required contours and drawing the bounding boxes around the required text content in the image. Initially the captured image is rescaled to an appropriate size



**Figure 2.** An image of the developed system.

and converted into gray scale image such that it will be more useful for further processing. Then the discrete cosine transformation is applied to the gray image to compress the image, which helps to improve the processing rate. Unwanted high frequency components present in the image are also eliminated by setting the vertical and horizontal ratio. To achieve decompression, the inverse discrete cosine transform is applied. The image then undergoes morphological operations like erosion and dilation [19] [20] [21].

The erosion of an image  $f(x, y)$  by a structural element  $s$  is denoted by  $f \ominus s$ , where  $\ominus$  denotes the erosion between  $f$  and  $s$ . Resultant image  $f \ominus s$  can be presented by Equation (1) [22]. **Figure 3** shows the threshold and eroded images.

$$g = f \ominus s \quad (1)$$

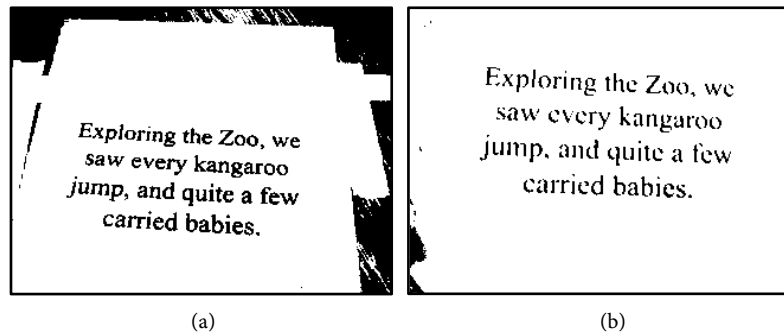
The dilation operation is performed to add pixels to the boundaries of the objects present in the image. The number of pixels added to the objects depends on the size and shape of the structuring element defined to process the image. The dilation of the image  $g(x, y)$  by a structural element  $s$  results in the image  $h(x, y)$ . The relationship is provided in Equation (2) [22].

$$h = g \oplus s \quad (2)$$

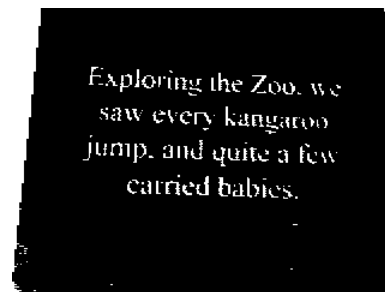
**Figure 4** shows a sample of the dilated image. A line of Python code utilized for the dilation process implementation is provided below:

```
dilation = cv2.dilate(img, kernel, iterations = 1)
```

After the morphological operations, thresholding is applied to the morphologically transformed image. This is followed by the generation of contours of the image using functions in OpenCV. These contours are used to draw the bounding boxes for the objects or elements present in the image. By using these bounding boxes each and every character present in the image is extracted and are then applied to the OCR engine to recognize the entire text present in the image.



**Figure 3.** Preprocessing-Erosion. (a) Threshold image; (b) Eroded image.



**Figure 4.** Preprocessing-Dilation.

```

importcv2                               Importing OpenCV Library
importnumpyasnp                          Importing numpy

img=cv2.imread('j.png',0)                Reading/Opening Image
kernel=np.ones((5,5),np.uint8)           Initializing structural
erosion=cv2.erode(img,kernel,iterations=1) Erosion Process

```

### 2.2.3. Image Text Extraction

To extract text from an image, the OCR is used in conjunction with pattern recognition, artificial intelligence, and computer vision. The OCR is a process of the electronic conversion of images of typed, handwritten, or printed text into machine-encoded text. It is widely used as a form of data entry from printed data records, whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any other documentation. It is a common method of digitizing printed texts so that it can be electronically edited, searched, stored more compactly, displayed online, and used in machine processes such as machine translation, text-to-speech, key data, and text mining.

There have been plenty of studies carried out on different OCR techniques. In one survey researchers looked into 17 different approaches of OCR and highlighted their limitations [23]. However, Tesseract is an open source platform that allowed many developers and enthusiasts to work, train and improve the library. It has grown into a reliable tool for text extraction applications. In addi-

tion, the availability of Tesseract to process multiple languages has made it even more popular in the field of text extraction. In this project, OCR was performed using the Tesseract library (discussed in Section 2.2.4) coupled with OpenCV to render images. The image taken by the camera is passed onto the OpenCV for thresholding and filtering. The image is then passed to the Tesseract library, which recognizes the texts and returns it in a string format. Finally, the string format text is passed to Alexa services for text-to-speech conversion.

#### 2.2.4. Tesseract

The image is taken by the web camera connected to the Raspberry Pi and is quickly processed through morphological thresholding and subsequently processed by Tesseract. Tesseract is composed of only two arguments; one is the image and the other is the output file where the detected text will be stored [24] [25]. Tesseract by default picks the extension of output file as .txt. A Python program opens the .txt file and passes the detected text to the AWS to read aloud through Alexa Echo Dot. Tesseract supports about 110 languages around the world. Each language comes with a trained language data file. However, with the limitation of AWS to read English and German and for the ease of this application, the text reader system was confined to English. The Python code to operate Tesseract is provided below:

```
from pytesseract import *           Importing Tesseract Library for Python
img= Image.open(image_file)       Read Image
text= image_to_string(im)         Convert Image to String
```

### 2.3. Utilization of Amazon Web Services (AWS)

The AWS is a secure cloud services platform hosted by Amazon. This can facilitate Internet of Things (IoT) systems, networking, databases, storage, analytical, and mobile development applications. One such service of the AWS is the Alexa voice recognition and speech system, which is a speech recognition and speech synthesis service. Through this service developers can connect their applications to the Alexa speech system. For a given application, the user needs to create an Alexa skill set on the cloud that can interact with the application. In this project, the AWS receives the text from the Raspberry Pi after text extraction and reads the detected text through the Alexa speech synthesis service. Alexa services invoke the text reader application running from the Raspberry Pi to extract the text from the printed document and then pass it onto Alexa services for speech synthesis. Alexa speech then reads the text loud using the Amazon Echo Dot device [26].

The first step in this process is to download, install, and run ngrok on the Raspberry Pi. ngrok is a command line program that creates a secure terminal and can be accessed by an https address generated by the application (Figure 5). Figure 6 shows the ngrok services running on Raspberry Pi with the https address highlighted in red.

The second step involves creating an Alexa skill set. Figure 7 shows the win-

```

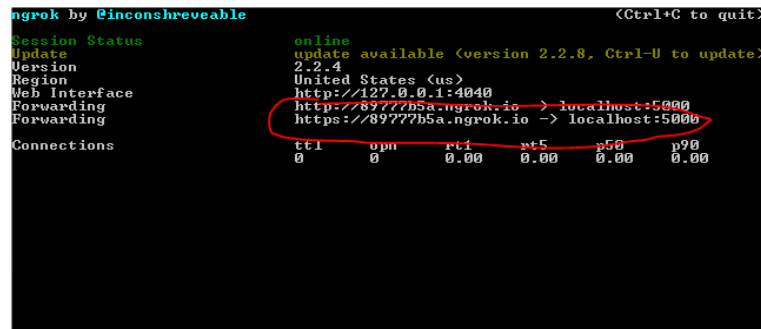
https://ngrok.com/download to get the latest Linux ARM release as a zip and unzip inside the home
directory:
unzip /home/pi/ngrok-stable-linux-arm.zip

Next, run it from the command line with:

sudo ./ngrok http 5000

```

**Figure 5.** Terminal command in Raspberry Pi to run ngrok on port 5000.

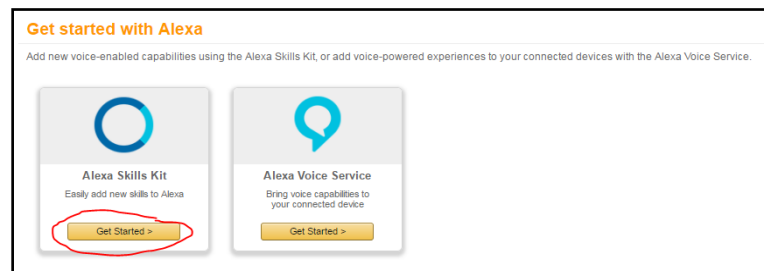


```

ngrok by @inconshreveable <Ctrl+C to quit>
Session Status      online
Update             update available (version 2.2.8, Ctrl-U to update)
Version            2.2.4
Region             United States (us)
Web Interface       http://127.0.0.1:4040
Forwarding          http://89777b5a.ngrok.io -> localhost:5000
                   https://89777b5a.ngrok.io -> localhost:5000
Forwarding
Connections
                   ttl    opn    rcl    wt5    p50    p90
                   0     0     0.00  0.00  0.00  0.00

```

**Figure 6.** Image of the ngrok services on Raspberry Pi.



**Figure 7.** Creating Alexa skill set.

dow on the Amazon developer portal to create a skill. **Figure 8** shows how the skill information is created within the AWS. The invocation name specifies the name of the skill set, which is invoked when Alexa is instantiated to access the text reading skills.

**Figure 9** shows the creation of an interaction model, which is used to communicate with the Python script running on the Raspberry Pi. The intent specifies different functions that need to be assessed within the Python script. The configuration settings that need to be entered on the AWS are provided in **Figure 10**. The service end-point allows the user to enter the ngrok https link used to invoke the Python script on the Raspberry Pi. Once the Python script accesses the text, the reader detects the printed/written text and passes the value to the Alexa services, which then reads it aloud.

### 3. System Evaluation

An evaluation exercise was conducted to verify effectiveness of the developed text reader. The evaluation exercise involved verifying the readability of the text reader for different fonts, background color, level of noise on the text, and time required for the reading process.

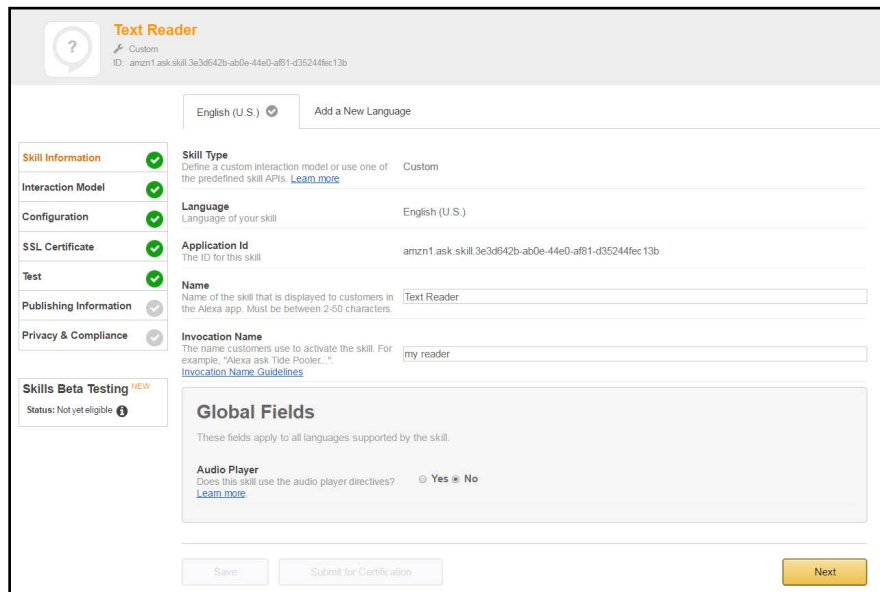


Figure 8. Creating skill set within the AWS.

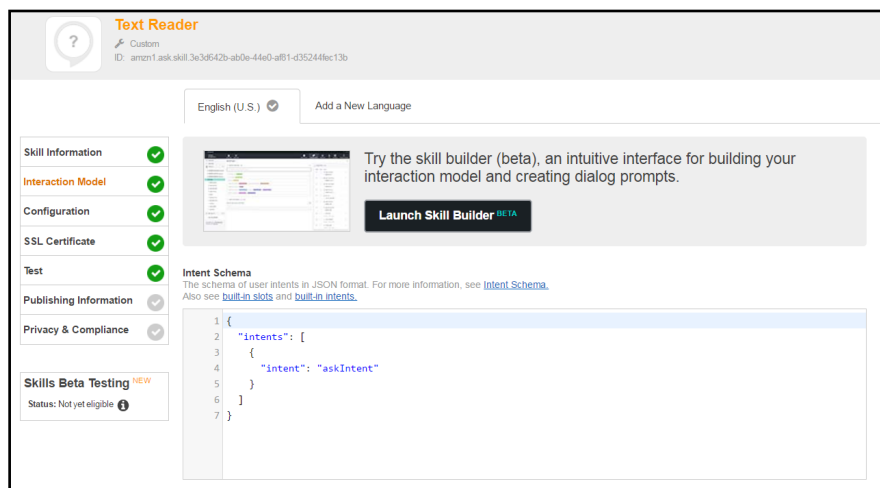


Figure 9. Interaction model.

### 3.1. Variation of Text Font

The first evaluation exercise involved the readability verification using different fonts with varying background colors. For this exercise, a phrase was selected that included all 26 letters of the alphabet in English. The phrase is “*Exploring the Zoo, we saw every kangaroo jump, and quite a few carried babies,*” which has a total of 75 characters including spaces, commas, and full stop. The fonts used were Arial, Times New Roman, Courier New, Bradley Hand, Lucida Calligraphy, and Lucida Handwriting. In addition, two manual handwritten samples were used: Handwriting Cursive and Handwriting Running. The text used for manual handwriting was “*The Quick Brown Fox Jumps Over The Lazy Dog.*” The background colors were white, green, blue, yellow, orange, red, and purple (Figure 11). The reading accuracy of each of these combinations was measured in terms



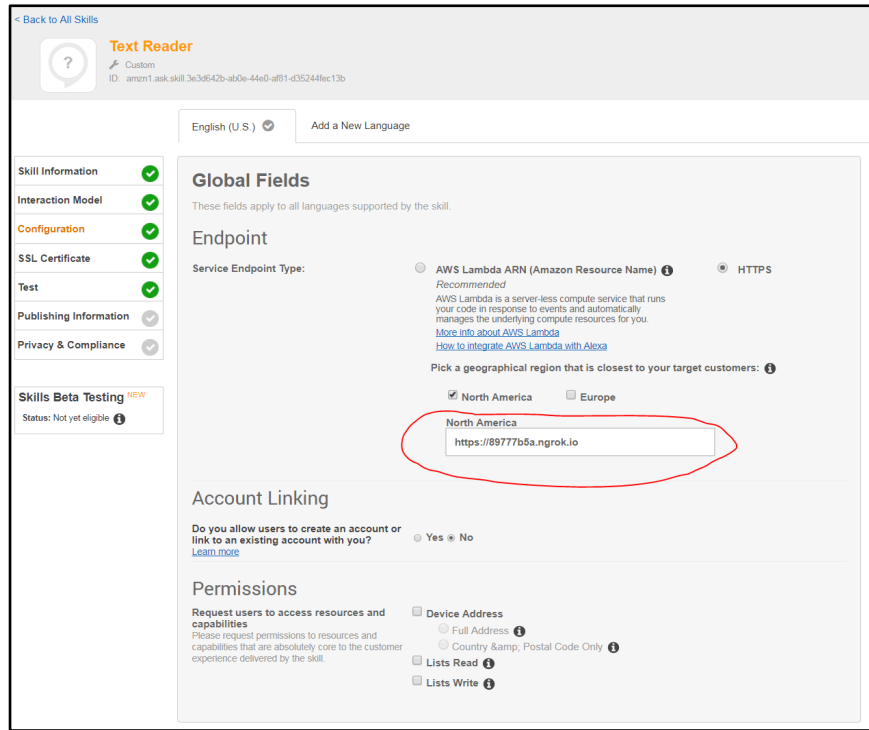


Figure 10. Configuration settings.

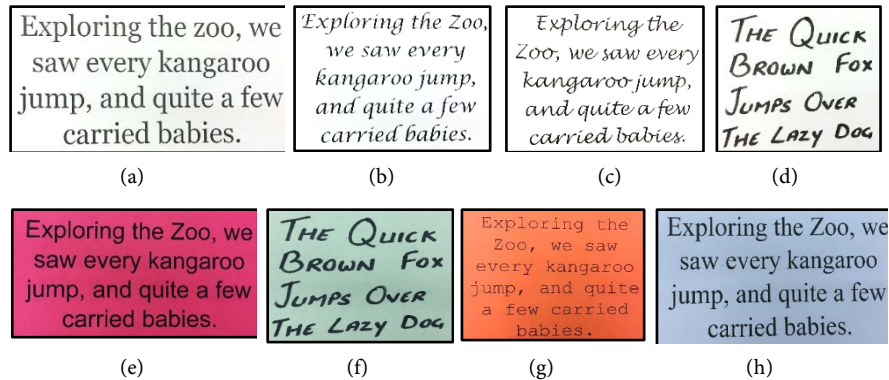


Figure 11. Samples of few fonts and background colors used for the evaluation process. (a) Time New Roman; (b) Lucida Calligraphy; (c) Lucida Handwriting; (d) Manual Handwriting; (e) Red paper; (f) Green paper; (g) Orange paper; (h) Purple paper.

of the number of characters misread and was expressed in % of error. The expression used for % of error calculation is shown in Equation (3).

$$\% \text{ of error} = \frac{\text{Number of alphabets or characters mis-spelled}}{\text{Total number of characters in the sentence}} \times 100 \quad (3)$$

The outcome of this evaluation exercise is provided in Table 1, where a 100% error indicates not readable at all. In terms of computer generated fonts, Arial was the best performer and Lucida Handwriting was the worst. Irrespective of background color, the manual handwriting (cursive) was not readable at all. The level of error for manual handwriting (running letter) was quite high for all

**Table 1.** Error in identifying text with different fonts using various background colors.

Font	% Error (white)	% Error (green)	% Error (blue)	% Error (yellow)	% Error (orange)	% Error (red)	% Error (purple)
Arial	0.00%	9.33%	0.00%	0.00%	0.00%	0.00%	0.00%
Times New Roman	0.00%	8.00%	8.00%	0.00%	0.00%	0.00%	5.33%
Lucida Calligraphy	1.33%	1.33%	2.67%	4.00%	1.33%	4.00%	4.00%
Courier New	2.67%	5.33%	4.00%	1.33%	100%	0.00%	14.67%
Bradley Hand	8.00%	5.33%	5.33%	5.33%	8.00%	6.67%	10.67%
Lucida Handwriting Manual	61.33%	100%	1.33%	60.00%	100%	100%	8.00%
Handwriting (running letters) Manual	17.50%	10.00%	35.00%	12.50%	22.50%	20.00%	10.00%
Handwriting (cursive)	100%	100%	100%	100%	100%	100%	100%

background colors. With computer-generated fonts, most of the cases the purple shows higher % Error than others. So, it can say that the purple background was the worst performer.

### 3.2. Variation of Noise Level

This part of the test was conducted with a phrase in which the texts were eroded/faded with an increased number of non-zero pixels. This reduction of pixel introduces a level of noise for the camera. A level of noise has been introduced for letter 'O' and **Figure 12** shows both the distorted and undistorted images.

To illustrate the development process of the eroded/faded text, let us assume a small section of the image from an undistorted font:

$$I_{UT} = \begin{matrix} 255 & 255 & 0 \\ 255 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \quad (4)$$

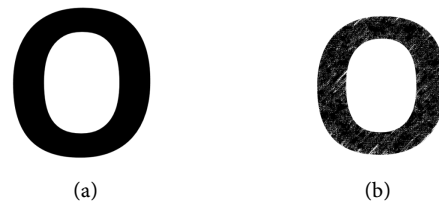
Similarly, assume a small section of the image from a distorted font:

$$I_{DT} = \begin{matrix} 255 & 230 & 200 \\ 0 & 255 & 0 \\ 255 & 0 & 255 \end{matrix} \quad (5)$$

The difference between Equations 4 and 5 generated the difference equation 6:

$$(I_{UT} - I_{DT}) = \begin{matrix} 0 & 20 & 200 \\ 255 & 255 & 0 \\ 255 & 0 & 255 \end{matrix} \quad (6)$$

The number of elements (pixels) in Equation (5) that are non-zero = 6. Which means out of 9 pixels, 6 pixels were distorted, hence noise % =  $((9 - 6)/9) \times 100 = 33.33\%$ . The level of noise is measured using Equation (7).



**Figure 12.** Sample of undistorted and distorted font for the letter “O”. (a) Undistorted font; (b) Distorted font.

$$\% \text{ of noise} = \frac{\text{Number of non-zero pixels}(I_{UT} - I_{DT})}{\text{Number of pixels in } I_{UT}} \times 100 \quad (7)$$

where,  $I_{UT}$  = Image with undistorted text,

$I_{DT}$  = Image with distorted text.

The evaluation exercise was performed with only white paper. **Figure 13** shows three sample texts with different noise levels. The outcome of the exercise is provided in **Table 2**. The table shows that for N1 and N2 the system reads the text without any error, but for N5, it cannot read the text at all. One notable issue is that the % Error is less for N4 than N3. When the % Noise is 3.63 for N4 and 3.26 for N3. Usually one can expect a higher % Error for N3.

### 3.3. Partial Removal of Alphabet

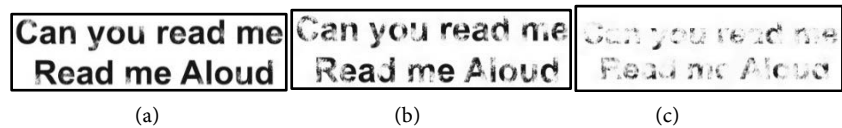
In this approach, part of the text was removed to verify the level of accuracy in readability. An example of the partially removed texts is provided in **Figure 14**. The samples are 75% visibility from the bottom, 75% visibility from the top, and 25% washed out from center. In this case, only the Arial font was used with different background colors. The readability performance of the system with partially removed text with different background colors is provided in **Table 3**. It shows some level of readability only for the case in which 75% is visible from the top. Other two cases cannot be read at all.

### 3.4. Commercially Printed Materials

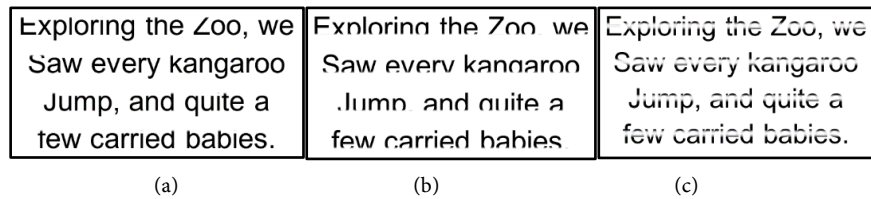
The last evaluation exercise involved commercially printed materials. Under this category, four items were investigated. The items are product catalog, product package, and product brochure—all of them in combination with different background colors. This exercise allows one to understand how to use this system to extract information from everyday household items beyond standard printed letters and documents. Findings of this exercise are provided in **Table 4**. As it appears all of the cases, the system read the text with very little error.

### 3.5. Timing for Text Reading Process

One of the important issues is the total processing time for a given text reading task. Considering the number of process, a part of the time is needed for the



**Figure 13.** Text with different levels of noise. (a) Noise level-N0 (0.8%); (b) Noise level-N1 (1.2%); (c) Noise level-N5 (5.58%).



**Figure 14.** Text with different levels of visibility. (a) 75% visibility from bottom; (b) 75% visibility from top; (c) Center washed out.

**Table 2.** Error in identifying text with different levels of noise.

Noise Level	% Noise	No. Character in Error	% Error
Level-N0	0.80%	0	0.00%
Level-N1	1.20%	0	0.00%
Level-N2	2.91%	2	7.14%
Level-N3	3.26%	8	28.57%
Level-N4	3.63%	5	17.86%
Level-N5	5.58%	Not Readable	100%


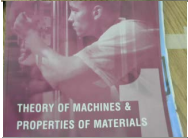
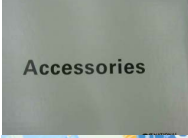

**Table 3.** Error in identifying partially erased text with different background colors.

Text Visibility	% Error (white)	% Error (green)	% Error (blue)	% Error (yellow)	% Error (orange)	% Error (red)	% Error (purple)
75% Visibility from top	48.00%	52.00%	53.33%	52.00%	49.33%	66.67%	46.67%
Center washed out	100%	100%	100%	100%	100%	100%	100%
75% Visibility from bottom	100%	100%	100%	100%	100%	100%	100%

embedded processor and the other part is for AWS. Evaluation was performed for the embedded processor’s processing time, the time taken by the AWS depend on the speed of the Internet from a given location (influenced by various factors) and cannot be evaluated with a certainty. Python’s time function was utilized to conduct this exercise. Test was performed using three different sentences with different background colors. There is almost no effect of background colors in terms of timing. The sentences and their corresponding processing time are provided below:

*Exploring the Zoo, we saw every kangaroo jump, and quite a few carried babies.* 3.22 seconds [printed text]

**Table 4.** Error in identifying text on commercially printed documents.

	WIRELESS GLASSES KIT	0.0%
	THEORY OF MACHINES PROPERTIES OF MATERIALS	2.32%
	Accessories	0.0%
	Element/4 800.463.9275	4.5%

*Can you read me Read me Aloud:* 1.29 seconds [printed text]

*The Quick Brown Fox Jumps Over the Lazy Dog.* 4.13 seconds [handwritten text].

#### 4. Conclusions

The paper describes the design, development, and evaluation of a cloud based text reader. A person who is visually impaired could utilize this for everyday activities. The user needs to carry a small hardware system composed of a camera, an embedded processor board, and an Alexa Echo Dot. The software developed utilizes an open source platform and is easy to access and modify. The evaluation exercise was conducted using texts with different fonts, background color, and level of noise on the text. The system is proven to be mostly accurate while reading commonly used fonts with a variety of background colors. Evaluation was also extended to manually written text as well as commercially printed materials with variety of background colors. In terms of future expansion the evaluation process can be extended to verify some other factors, such as total processing time under different circumstances, different length of sentences, and scientific notations.

In its current form, it looks like that the system is successful with commercially printed materials but had very little success with manually written text. One of the limitations of the system is the dependence on a commercial cloud service. Reliability of this cloud service is an important factor for the system performance. On the other hand, it was possible to make the system cost effective and smaller in size only due to the use of a commercial cloud service. In conclusion, it can be stated that in its current form the system can be used by individuals with visual impairment to extract information from standard printed materials with very little effort.

## Acknowledgements

The authors would like to thank the National Science Foundation (NSF) for its support for the reported work. This paper is based on an NSF TUES project, award number DUE-1140502. Any opinions, findings and conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the NSF.

## References

- [1] Leon, M., Vilaplana, V., Gasull, A. and Marques, F. (2013) Region-Based Caption Text Extraction. In: Adami, N., Cavallaro, A., Leonardi, R. and Migliorati, P., Eds., *Analysis, Retrieval and Delivery of Multimedia Content*, Springer, New York. [https://doi.org/10.1007/978-1-4614-3831-1\\_2](https://doi.org/10.1007/978-1-4614-3831-1_2)
- [2] Chucai, Y. (2015) Text Extraction from Natural Scene: Methodology. Ph.D. Dissertation, The City University of New York, New York.
- [3] Srivastava, A., Kumar, D., Gupta, O.P., Maurya, A. and Sribastava, S.K. (2013) Text Extraction in Video. *International Journal of Computational Engineering Research*, **3**, 48-53.
- [4] Kumar, S., Kumar, S. and Gopinath, S. (2012) Text Extraction From Images. *International Journal of Advanced Research in Computer Engineering & Technology*, **1**, 34-36.
- [5] Prabakaran, M. and Radha, K. (2015) Text Extraction from Natural Scene Images and Conversion to Audio in Smart Phone Applications. *International Journal of Innovative Research in Computer and Communication Engineering*, **3**, 19-23. <https://doi.org/10.15680/ijirce.2015.0301004>
- [6] Gómez, L. and Karatzas, D. (2013) Multi-Script Text Extraction from Natural Scenes. *Proceedings of the 12<sup>th</sup> International Conference on Document Analysis and Recognition*, Washington DC, 25-28 August 2013. <https://doi.org/10.1109/ICDAR.2013.100>
- [7] King, S. (2014) Measuring a Decade of Progress in Text-to-Speech. *Loquens*, **1**, 1-12. <https://doi.org/10.3989/loquens.2014.006>
- [8] Singla, S.K. and Yadav, R.K. (2014) Optical Character Recognition Based Speech Synthesis System Using LabVIEW. *Journal of Applied Research and Technology*, **12**, 919-926. [https://doi.org/10.1016/S1665-6423\(14\)70598-X](https://doi.org/10.1016/S1665-6423(14)70598-X)
- [9] Shetake, P.S., Patil, S.A. and Jadhav, P.M. (2014) Review of Text To Speech Conversion Methods. *International Journal of Industrial Electronics and Electrical Engineering*, **2**, 29-35.
- [10] Al-Khalifa, H.S. (2008) Utilizing QR Code and Mobile Phones for Blinds and Visually Impaired People. In: Miesenberger, K., Jlaus, J., Zagler, W. and Karshmer, A., Eds., *Proceedings of 11<sup>th</sup> International Conference*, ICCHP 2008, Austria, 9-11 July 2008, Springer-Verlag, Berlin, Heidelberg, 1065-1069.
- [11] Mate Galaxy, I.D. (2017). <http://www.envisionamerica.com/products/idmate/>
- [12] Talking Barcode Scanner (2017) Vision Australia. <http://www.visionaustralia.org/learn-more/learning-to-live-independently/using-technology-and-computers/technology-overview/identifying-day-to-day-items-with-technology/talking-barcode-scanner>
- [13] Ferreira, S., Thillou, C. and Gosselin, B. (2003) From Picture to Speech: An Innova-

tive Application for Embedded Environment. *14th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, November 2003.

- [14] Chaudhuri, A., Mandaviya, K., Badelia, P. and Ghosh, S.K. (2017) Optical Character Recognition Systems. In: *Optical Character Recognition System for Different Languages with Soft Computing*, Studies in Fuzziness and Soft Computing, Springer International Publishing, New York. [https://doi.org/10.1007/978-3-319-50252-6\\_2](https://doi.org/10.1007/978-3-319-50252-6_2)
- [15] Eikvil, L. (1993) Optical Character Recognition. Research Report, NorskRegnesentral, Blindern.
- [16] Sercan, O.A., Chrzanowskiy, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Ng, A., Raiman, J., Sengupta, S. and Shoeybi, M. (2017) Deep Voice: Real-Time Neural Text-to-Speech. *International Conference on Machine Learning*, Sydney, 6-11 August 2017.
- [17] Thomas, S. (2007) Natural Sounding Text-to-Speech Synthesis Based on Syllable-Like Units. Masters Dissertation, Indian Institute of Technology Madras.
- [18] Tiomkin, S., Malah, D. and Shechtman, S. (2010) Statistical Text-to-Speech Synthesis Based on Segment-Wise Representation with a Norm Constraint. *IEEE Transactions on Audio, Speech, and Language Processing*, **18**, 1077-1082. <https://doi.org/10.1109/TASL.2010.2040795>
- [19] Vaghela, K. and Patel, N. (2013) Automatic Text Detection using Morphological Operations and inpainting. *International Journal of Innovative Research in Science, Engineering and Technology*, **5**, 1333-1336.
- [20] Bawal, R.K. and Sethi, G.K. (2014) A Binarization Technique for Extraction of Devanagari Text from Camera Based Images. *Signal & Image Processing: An International Journal*, **5**, 29-37.
- [21] Babu, G.R.M., Srimaiyee, P. and Srikrishna, A. (2010) Text Extraction from Heterogeneous Images using Mathematical Morphology. *Journal of Theoretical and Applied Information Technology*, **16**, 39-47.
- [22] Shih, F.Y. (2010). *Image Processing and Pattern Recognition*. John Wiley & Sons Inc. <https://doi.org/10.1002/9780470590416>
- [23] Gera, D. and Jain, N. (2015) Comparison of Text Extraction Techniques—A Review. *International Journal of Innovative Research in Computer and Communication Engineering*, **3**, 621-626. <https://doi.org/10.15680/ijirccce.2015.0302003>
- [24] Smith, R. (2007) An Overview of the Tesseract OCR Engine. *9th International Conference on Document Analysis and Recognition*, Brazil, 23-26 September 2007. <https://doi.org/10.1109/ICDAR.2007.4376991>
- [25] Patel, C., Patel, A. and Patel, D. (2012) Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study. *International Journal of Computer Applications*, **55**, 50-56. <https://doi.org/10.5120/8794-2784>
- [26] Amazon (2017) Amazon Echo Dot. <https://www.amazon.com/Amazon-Echo-Dot-Previous-Generation/b?ie=UTF8&node=14047587011>