IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 41, No. 6, December, 2024, pp. 3183-3192
Journal homepage: http://iieta.org/journals/ts

A Novel Real-Time Text-to-Speech System Using Raspberry Pi for Assisting the Visually Impaired

Ahmed Ben Atitallah[1*], Manel Kammoun[2], Mohamed Amin Ben Atitallah[2,3], Mohammed Albekairi[1], Yahia Said[4,5], Anis Boudabous[6], Khaled Kaaniche[1], Mohamed Atri[7]

[1] Department of Electrical Engineering, College of Engineering, Jouf University, Sakaka 72388, Saudi Arabia
[2] LETI, ENIS, University of Sfax, Sfax 3029, Tunisia
[3] Laboratory of Informatics, Gaspard-Monge, A3SI, ESIEE Paris, CNRS, Gustave Eiffel University, Noisy-le-Grand BP 99 93162 Cedex, France
[4] Remote Sensing Unit, College of Engineering, Northern Border University, Arar 91431, Saudi Arabia
[5] Laboratory of Electronics and Microelectronics (LR99ES30), University of Monastir, Monastir 5019, Tunisia
[6] Department of Computer Engineering and Networks, College of Computer and Information Sciences, Jouf University, Sakaka 72388, Saudi Arabia
[7] College of Computer Sciences, King Khalid University, Abha 11614, Saudi Arabia

Corresponding Author Email: abenatitallah@ju.edu.sa

https://doi.org/10.18280/ts.410634

Received: 5 January 2024
Revised: 15 September 2024
Accepted: 20 November 2024
Available online: 31 December 2024

Keywords:
image preprocessing, visual impairment, Raspberry Pi 4, text-to-speech, optical character recognition, real-time processing

ABSTRACT

Visual impairment is one of the most significant challenges facing humanity, especially in an era where information is frequently conveyed through text rather than voice. To address this, the proposed system is designed to assist individuals with visual impairments. This paper presents the development of a real-time Text-to-Speech (TTS) embedded system based on the Raspberry Pi 4. Our system incorporates a novel approach to enhance the accuracy of text recognition using Optical Character Recognition (OCR) from images. Specifically, a series of preprocessing steps are employed, selected dynamically by a decision-making process based on the content of the image. The image processing is handled using OpenCV2, while the conversion of text to speech is achieved through the pyttsx3 Python library. The entire system is implemented and tested on a Raspberry Pi 4, connected to a USB Full HD camera for high-resolution image acquisition, and controlled via the Traffic HAT-LED module. Experimental results demonstrate that our system achieves a minimum accuracy of 88.33% in text recognition from images.

1. INTRODUCTION

People with visual impairments, also referred to as visually impaired or visually disabled individuals, have varying degrees of vision loss that can range from mild to complete blindness. Visual impairments can be caused by a variety of factors, including congenital conditions, acquired diseases, injuries, or age-related changes. Visual impairments can pose significant challenges to individuals in their daily lives, especially when it comes to accessing printed text. In this context, text recognition in real-time scenarios, such as capturing text from live video streams or on-the-fly recognition from mobile devices, can be challenging. In fact, the algorithm requests to process the video frames quickly while maintaining accuracy. Fortunately, advancements in technology have opened up new avenues for overcoming these challenges. In fact, Zaman et al. [1] proposed a portable virtual text reader based on the Raspberry Pi 3. This reader captures the image from the camera through the graphical user interface. Then the captured image is passed to the Optical Character Recognition (OCR) for text detection and recognition. In the end, the eSpeak software is used to convert the text to audio format. Sarkar et al. [2] presented a smart reader for the visually impaired using the Raspberry Pi B+. In this work, the authors used MATLAB to recognize the text from the captured image and converted it into speech using the Text-to-Speech (TTS) synthesizer. In study [3], a text reader system for blind people using the Raspberry Pi 3B and camera module was presented. The proposed system used the OCR and the convolutional recurrent neural network to detect, localize, and extract the text, and pyttsx3 for the text to speech conversion. The proposed system lacked real-time operation. The Tesseract OCR engine, Google Speech API, and Microsoft Translator concepts were used to design the device by Rithika and Santhoshi [4]. The system failed with high accuracy. Velmurugan et al. [5] used the image processing toolbox to simulate their system in MATLAB and discovered that the algorithm successfully processed the image and clearly interpreted it. Gurav et al. [6] modeled an OCR-based system utilizing computer software and a photoelectric device. Despite being a successful assistive device for visually impaired people, the model turned out to be quite noisy, which is a major drawback. Chavan et al. [7] utilized Tesseract OCR and OpenCV2 to extract text from scanned images, which was

then converted into voice using Google Text to Voice. This enabled individuals with visual impairments to read the text.

This work presents a novel approach for real-time conversion of text contained in images and videos into speech. We specifically designed the method to aid individuals with visual impairments. In fact, our methodology relies on several preprocessing steps, including image resizing, noise reduction, and binarization. In addition, we integrate a decision-making procedure to choose the most efficient preprocessing methods for enhancing the precision of text detection, extraction, and recognition using the Tesseract OCR. For converting the Text-to-Speech (TTS), the pyttsx3 Python library is used. Further, our approach is implemented and evaluated on the Raspberry Pi 4 board, which is connected to the USB Full HD camera for capturing image and video and to the Traffic HAT-LED card for controlling the system. Our innovation lies in the development of a highly accurate and real-time TTS embedded system utilizing a Raspberry Pi 4 and a Full HD camera. What sets our system apart is its integration of a decision-based preprocessing step, which dynamically selects the most appropriate techniques—such as rescaling, noise removal, and binarization—based on the content of the image. This approach significantly improves OCR accuracy compared to existing methods. Moreover, the real-time performance, facilitated by the Raspberry Pi 4 and the Full HD camera, offers a more responsive and user-friendly experience for visually impaired individuals, distinguishing our system from current solutions.

The rest of paper is organized as follows: Section 2 provides an overview of the Tesseract OCR. Section 3 describes our proposed approach for text detection, extraction, and recognition. Section 4 illustrates the development of our TTS embedded system based on the Raspberry Pi 4. The evaluation of our system is presented in section 5. The paper is concluded by Section 6.

2. TESSERACT OCR OVERVIEW

In the recent years, there has been an increased focus on finding solutions for the challenges associated with finding and understanding text embedded in images and videos. Indeed, factors like complex backgrounds, diverse text layouts and fonts, uneven lighting, low resolution, and the presence of multilingual content make the task significantly more challenging compared to working with clean and well-structured documents. Addressing these issues requires the utilization of advanced computer vision and pattern recognition techniques [8-11]. Various methods, including the widely used Tesseract OCR, have been proposed to tackle the problem of text detection and recognition in scene imagery.

The Tesseract OCR [12, 13], originally developed by Google, is an open-source software library that has gained substantial popularity. Its primary purpose is to extract text from images, making it a widely utilized tool for various tasks, including document scanning, automated data entry, and text recognition in numerous applications. To achieve accurate results, the Tesseract OCR employs a multi-step process, as shown in Figure 1, that enables it to recognize and extract text effectively from an input image. However, adaptive thresholding is the first step in Tesseract OCR. It involves converting the input image into binary images. The purpose of this step is to create a clear distinction between the foreground (text) and background in the image. The next step is connected component analysis. This analysis is employed to extract the outlines of individual characters. Once the outlines are obtained, they are converted into blobs. These blobs are then organized into text lines, where the text elements are grouped together based on their spatial arrangement. After that, each text line is chopped into words by dividing the text based on definite spaces and fuzzy spaces. This process helps segment the text into meaningful units for recognition. The recognition of text is performed as a two-pass process, as illustrated in Figure 1. In the first pass, Tesseract OCR attempts to recognize each word in the segmented text. Words that are recognized with a satisfactory level of confidence on the first pass are considered successful recognitions. However, if a word is not recognized accurately in the first pass, it moves on to Pass 2 for further processing. The purpose of the second pass is to improve the recognition accuracy for the words that were initially challenging for Tesseract OCR. Finally, the recognized text is presented in the output.

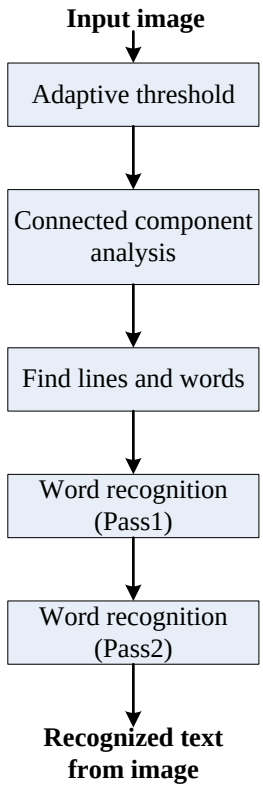

Figure 1. Block diagram of the Tesseract OCR

Tesseract's performance can be significantly impacted by some input picture characteristics (such as blurring, color contrasts, etc.) despite the numerous improvements. In fact, as shown in reference [14], even little salt-and-pepper noise can make Tesseract OCR less effective to the point that text from perfect samples is not recognized at all or results in segmentation mistakes. For that, we propose in next section to enhance the performance of Tesseract OCR.

3. TESSERACT OCR ENHANCEMENT

3.1 Proposed approach

In this section, we propose a new approach to improve the accuracy of the Tesseract OCR engine by implementing a

series of preprocessing steps on the input image, as depicted on Figure 2, before it undergoes text recognition. These preprocessing steps are carefully selected to address common challenges in OCR, such as poor resolution, noise, and contrast issues. Each step contributes to improving the clarity and quality of the input image, which in turn enhances the OCR's ability to accurately recognize text. In the following subsection, we will provide a comprehensive explanation of each preprocessing step, including the reasoning for their selection and their impact on the overall performance of the system.

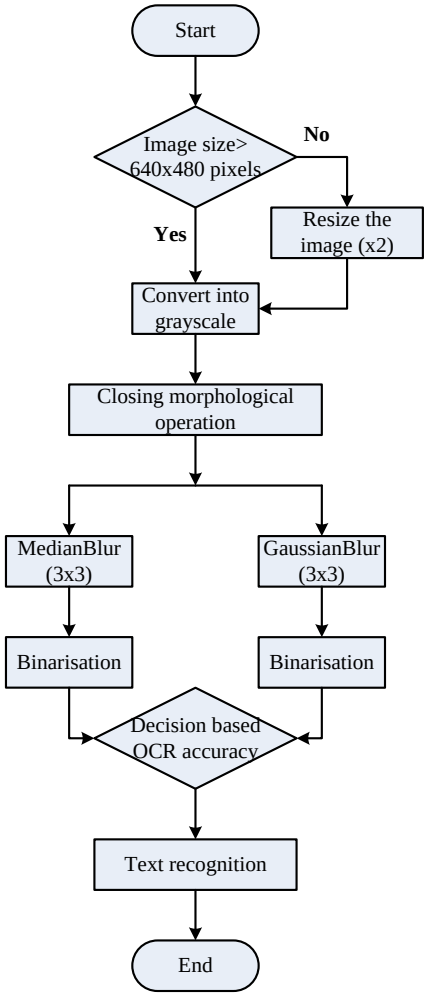

Figure 2. Block diagram of the suggested approach

3.1.1 Rescaling of image

Our approach begins by increasing the size of the images by two when its dimensions is equal to 640×480 pixels or less. In fact, the smaller images frequently contain text that is too small to be accurately detected and recognized by the OCR engine. However, by enlarging the image, we increase the text size, making it easier to read and lowering the risk of losing text detail during OCR processing. Consequently, this stage is especially important for images captured with low-resolution cameras, as text may appear pixelated or blurry at its original size. As a result, rescaling improves OCR accuracy by making the text clearer and more legible.

3.1.2 Conversion to grayscale

After rescaling, the image is converted to grayscale. Indeed, this step simplifies the image by removing color information, which reduces the complexity that the OCR engine needs to process. Nevertheless, grayscale images provide better contrast between the text and the background compared to color images, especially when the background is complex or multicolored. By focusing only on the luminance of the pixels, the OCR engine can more easily distinguish text from its surroundings, which enhances text detection and recognition accuracy.

3.1.3 Closing morphological operation

The next preprocessing step is the application of the closing morphological operation [15]. Indeed, this technique is a combination of dilation followed by erosion, using a 3×3 kernel. However, the aim of this operation is to close small gaps and fill in holes within the text regions of the image. In fact, dilation expands the boundaries of objects in the image, which helps to connect broken parts of the text. Then, the erosion contracts these boundaries to remove any excess expansion caused by dilation. Thereby, this step is particularly effective in cleaning up the text regions by removing small imperfections and making the text more uniform allowing to improve the OCR's ability to segment and recognize individual characters accurately.

3.1.4 Noise reduction via median and gaussian blurring

To further enhance image quality, we apply two noise reduction techniques—median blurring and gaussian blurring—each followed by a binarization step [16, 17]. Nevertheless, the median blurring is a non-linear filtering technique that is effective in removing salt-and-pepper noise while preserving the sharpness of edges. This is achieved by replacing each pixel value with the median value of its neighboring pixels, which helps to remove isolated noise points without significantly blurring the text edges. In contrast, the gaussian blurring is a linear filtering technique that reduces high-frequency noise by averaging the pixel values in the neighborhood of each pixel. This weighted averaging process smooths out the image while preserving the overall structure and details, making it particularly useful for reducing noise in areas with gradual intensity variations.

3.1.5 Binarization using Otsu's method

The final step in our approach is the binarization, which converts the grayscale image into a binary image where the text appears as black pixels on a white background. In fact, we use Otsu's method [18, 19] for this step, which automatically determines the optimal threshold value that minimizes the intra-class variance (the variance within the text and background regions) while maximizing the inter-class variance (the difference between the text and background). This method is particularly effective in ensuring that the text is clearly separated from the background, even in images with varying lighting conditions or complex backgrounds. Therefore, binarization simplifies the image for the OCR engine, allowing it to focus on the text without being distracted by background noise or color variations.

3.1.6 Dynamic decision-making process

Our approach includes a decision-making process to dynamically select the most appropriate preprocessing techniques based on the content of the image. For instance, the system may choose between median and gaussian blurring depending on which technique yields higher OCR accuracy for a given image. This adaptability ensures that our approach is tailored to the specific characteristics of each image, thereby

maximizing the OCR performance.

By combining these preprocessing techniques, our proposed approach optimizes the input image for Tesseract OCR, leading to significant improvements in text recognition accuracy. The experimental results, discussed in the next subsection, demonstrate the effectiveness of our approach in various test scenarios.

3.2 Experimental results

In order to evaluate the performance of our proposed approach for text detection and recognition from complex color images, several images from the International Conference on Document Analysis and Recognition (ICDAR) database [20, 21] are utilized, as depicted in Figure 3. This dataset includes a variety of texts with differing levels of complexity (e.g., different scenes, light, orientation, and pixel sizes). The performance evaluation of our proposed approach is realized in terms of the Tesseract OCR accuracy and the execution time, which are provided on the Intel i7-1165G7@2.80 GHz processor.

In our experiment, the preprocessing techniques (e.g., rescale image, closing morphological operation, median blurring, and gaussian blurring) are incrementally added to our flow to study their impact on the accuracy of recognizing text from complex images through Tesseract OCR. The results of our experiment are recorded in Tables 1 and 2 for the comparison of Tesseract OCR accuracy and execution time, respectively, and Figure 4 for showing the recognized text.

Figure 3. ICDAR dataset test image

Table 1. Comparison of the Tesseract OCR accuracy

	Tesseract OCR Engine	Rescale Image by 2	Closing Morphological Operation	Median Blurring	Gaussian Blurring	Proposed Approach
Image 1	60.83%	60.83%	89.67%	90.5%	95.17%	95.17%
Image 2	23.85%	23.85%	55.5%	88.33%	87.8%	88.33%
Image 3	0%	64%	32%	87.67%	95.67%	95.67%
Image 4	63%	29%	95.33%	95.67%	95.67%	95.67%
Image 5	31.66%	31.66%	87.67%	94.67%	91.333	94.67%
Image 6	0%	96%	89.500	88.5%	91%	91%
Image 7	48%	0%	0%	96%	48%	96%

Table 2. Comparison of the execution time in seconds

	Tesseract OCR	Rescale Image by 2	Closing Morphological Operation	Median Blurring	Gaussian Blurring	Proposed Approach
Image 1	0.384282	0.384394	0.318900	0.206311	0.205457	0.336051
Image 2	0.491994	0.490532	0.436517	0.238606	0.294120	0.408177
Image 3	0.178613	0.319863	0.336656	0.223150	0.235849	0.365757
Image 4	0.230687	0.483995	0.365979	0.206493	0.198623	0.339695
Image 5	0.334723	0.304162	0.249007	0.165266	0.172932	0.268753
Image 6	0.201328	0.433023	0.392255	0.224419	0.232477	0.424364
Image 7	0.200486	0.378631	0.350625	0.266602	0.223872	0.420371

3.2.1 OCR performance

Without any preprocessing, we applied the Tesseract OCR engine directly to the images. The results, presented in Table 1, demonstrate that the OCR accuracy is significantly lower when no preprocessing is applied, with many images showing poor text recognition, particularly in cases of low resolution, noise, or complex backgrounds as depicted in Figure 4(a).

3.2.2 Impact of preprocessing techniques

To determine the impact of each preprocessing step, in our

experiment, we incrementally applied the techniques described in Section 3.1 and measured their effect on OCR accuracy and execution time:

-Rescaling enhanced the OCR accuracy in image 3 from 0% to 64%, as shown in Table 1 and Figure 4(b), and similarly improved performance in other low-resolution images. This demonstrates that increasing the image size allows Tesseract OCR to recognize smaller text more effectively by making the characters clearer. However, this improvement comes with the drawback of doubling the execution time.

-While grayscale conversion alone did not significantly change the accuracy and the execution time, it set the stage for further preprocessing steps by simplifying the image and improving contrast, which indirectly contributed to higher accuracy when combined with subsequent steps.

-The closing morphological operation significantly improved text recognition, especially in images with gaps or broken characters. For example, in Image 5, accuracy increased from 31.66% to 87.67% after applying the operation, as shown in Table 1 and Figure 4(c). This demonstrates the effectiveness of morphological techniques in enhancing text continuity by filling small gaps that hinder character recognition. Additionally, Table 2 indicates that this step reduces the Tesseract OCR execution time by an average of 18%.

-The median blurring step was highly effective in eliminating salt-and-pepper noise, which often occurs in scanned documents or low-light images. In image 7, where noise was a major problem, median blurring raised OCR accuracy from 0% to 96%, as shown in Table 1. This method is vital for preserving edge sharpness while reducing noise, as illustrated in Figure 4(d). Additionally, it reduced the time required for Tesseract OCR to recognize the text by an average of 35%.

-Gaussian blurring proved more effective than median blurring in handling images with gradual intensity variations, such as shadows or uneven lighting. For instance, in image 3, Gaussian blurring boosted OCR accuracy from 32% to 95.67%, as displayed in Table 1. This technique smooths the image, making the text more distinct against inconsistent backgrounds, as shown in Figure 4(e). Additionally, it reduced the execution time required for Tesseract OCR to recognize the text by an average of 35%.

Figure 4. Text recognition accuracy by (a) Tesseract OCR engine only, (b) adding image rescale, (c) adding closing morphological operation, (d) adding median blurring, (e) using gaussian instead of median blurring and (f) proposed approach

3.2.3 Proposed approach performance

After evaluating each preprocessing step, we applied the complete preprocessing steps, which includes rescaling, grayscale conversion, closing morphological operation, median/gaussian blurring, and binarization. The results, as shown in Table 1, indicate that the combined approach significantly outperforms the individual OCR step. Thus, in the worst case, our proposed approach can enhance the OCR accuracy by 34% relative to the Tesseract OCR engine with a small increase in execution time by 21%. In light of the above findings, the image preprocessing techniques proposed in our approach lead to high reliability and accuracy in the recognized text.

3.2.4 Case studies

We conducted in-depth analyses of specific images that posed particular challenges to Tesseract OCR, such as:

-Image 3 (Low-Resolution Text): This image initially had 0% accuracy without preprocessing. After applying the complete preprocessing steps, accuracy improved from 0% to 95.67%, demonstrating the effectiveness of the preprocessing steps in enhancing text visibility and recognition.

-Image 7 (Noisy Background): Median blurring proved to be the most effective for this image, boosting accuracy from 48% to 96%, thereby showcasing the importance of noise reduction techniques in OCR preprocessing.

4. TEXT-TO-SPEECH EMBEDDED SYSTEM

Figure 5 presents the block diagram of the proposed Text-to-Speech (TTS) embedded system.

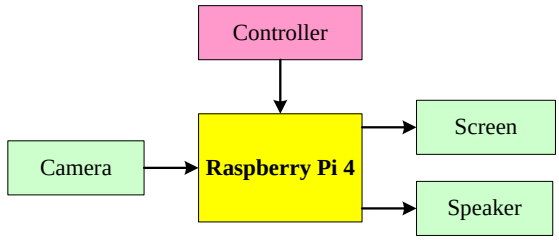

Figure 5. Block diagram of the TTS embedded system

The heart of our proposed system is the Raspberry Pi 4 board. This board receives the video stream from the USB camera, processes the video frame to detect and recognize the text that is displayed on the screen, and then translates it into speech for hearing it through the speaker. The controller allows to synchronize the operation of our system.

4.1 TTS system based raspberry Pi

The Raspberry Pi 4 board [22] (Figure 6) is a versatile and affordable single-board computer that is widely used for prototyping, home automation, media centers, and various other IoT applications. In fact, it offers significant improvements in terms of performance, connectivity, and features compared to its predecessors. The Raspberry Pi 4 board is powered by a Broadcom BCM2711 quad-core Cortex-A72 (ARMv8) 64-bit System-on-Chip (SoC) running at 1.5 GHz. This processor provides a substantial performance boost over the previous models. It is available in three RAM variants: 2 GB, 4 GB, and 8 GB LPDDR4-3200 SDRAM. The increased memory capacity allows for more demanding applications and multitasking.

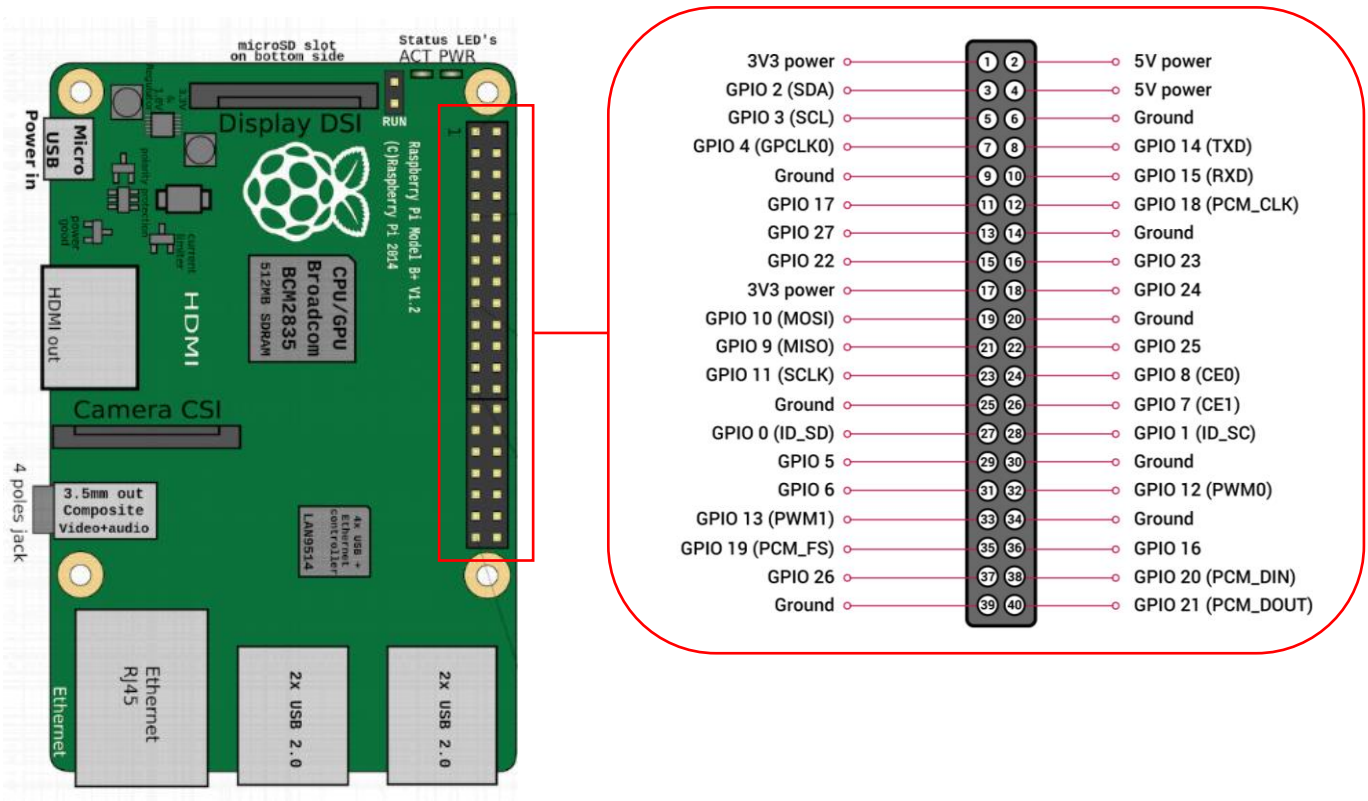

Figure 6. Raspberry Pi 4 board

Figure 7. Logitech C922 Stream Full HD camera

Figure 8. Traffic HAT - LED module for Raspberry Pi

The Raspberry Pi 4 board uses a microSD card slot for primary storage. It also features two USB 3.0 ports and two USB 2.0 ports, which can be used for external storage devices. Further, this board retains the 40-pin GPIO header, as depicted in Figure 6, which allows for interfacing with a wide range of electronic components and expansion boards, making it suitable for prototyping. Moreover, the Raspberry Pi 4 is compatible with various Operating Systems (OS), including the official Raspberry Pi OS, Ubuntu, and many OS distributions.

The Logitech C922 camera (Figure 7) provides the video feed to the TTS embedded system. For maximum video throughput, this camera is attached to the Raspberry Pi 4 through a USB connection. In addition, the Traffic HAT board is used to control the whole system. In fact, as shown in Figure 8, this board contains Red, Orange and Green LEDs and one button. The LEDs provides the status (ready or busy) of TTS embedded system. But, the button allows to go from step to other step. Figure 9 depicts TTS controller block diagram.

However, at the beginning, our TTS system initiates the video stream and detects the blur in the video frame, as illustrated by Figure 9.

Indeed, the blur decreases the quality of the frame, which has a negative impact on the detection and recognition of the text. For this, the fast Fourier transform blur detector [23] is applied to the video frame to detect whether the image is blurred or not. So, if the frame is blurred, the TTS system gets another frame from the video stream; if not, it will go to the next step. In this step, the button on the Traffic HAT board should be pushed to capture the frame, which is processed by the steps indicated in Figure 2 to detect and recognize the text. Then, this text is converted to speech using the pyttsx3 Python library. In the end, the button on the Traffic HAT board should be pushed again to restart the TTS system and capture another frame.

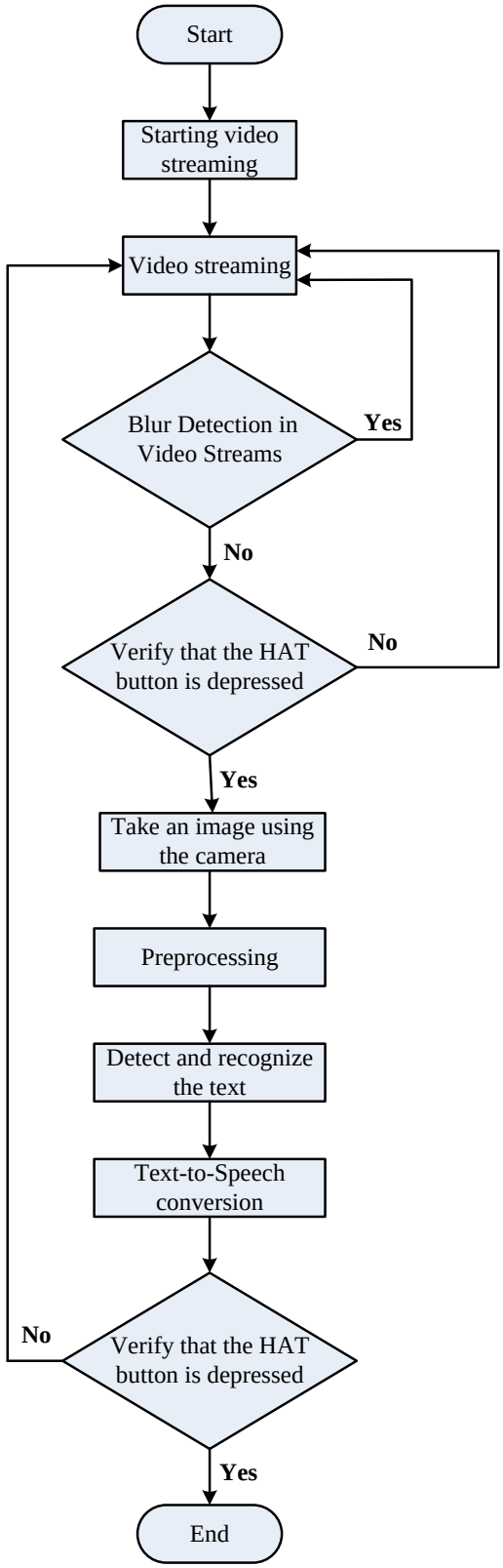

Figure 9. TTS controller block diagram

4.2 System evaluation

Figure 10 presents our prototype of the TTS embedded system. Our prototype contains the touch screen, Logitech C922 Stream Full HD camera, the Traffic HAT board and a speaker which are connected to the Raspberry Pi 4 board through the Input/Output connectors. The Raspberry Pi 4 board is the heart of our prototype and used to execute the Python software code in order to control the whole system.

Figure 10. Prototype of the TTS embedded system

The performance evaluation of our prototype is realized by using several test images from the ICDAR database as depicted by Figure 11. However, Figures11(a and b) present the detection of the blurry and not blurry images, respectively. But, Figure 11(c) illustrates the detection and the recognition of the text from the not burry image. In fact, as shown in Figure 11(c) all word in the test images is detected correctly without any error. In other side, no words are detected from blurry image. These results prove the reliability and the efficiency of our prototype to detect and recognize the text from video frame and transform it into the speech.

Our text-to-speech system built on the Raspberry Pi 4 can greatly improve quality of life for the visually impaired by increasing access to information. This affordable, energy-efficient system bridges the divide between print and digital material. By enabling independence and inclusion, it promotes equity. The Raspberry Pi 4 provides a familiar development platform to create accessible assistive technologies for this community. Overall, this TTS system has the potential to empower people with visual disabilities.

Figure 11. Performance evaluation of the TTS system for (a) blurry image, (b) not blurry image and (c) the detected and recognized text from video frame

4.3 Practical aspects of deploying our TTS embedded system in real-world scenarios

To successfully deploy our TTS embedded system in real-world environments, several practical considerations must be addressed:

4.3.1 Usability for end-users

As the system is designed for visually impaired users, ensuring ease of interaction is crucial. Intuitive, non-visual interfaces such as voice commands will greatly enhance user accessibility. This will allow users to seamlessly interact with the system without requiring visual input, making it suitable for daily use.

4.3.2 Energy efficiency and portability

For mobile and outdoor use, energy efficiency becomes a vital factor. While the Raspberry Pi 4 is an energy-efficient platform, the system's battery life must be optimized for continuous operation. Utilizing low-power components like cameras can extend the system’s usability in portable settings.

4.3.3 Durability and maintenance

In real-world applications, especially outdoor or mobile deployments, hardware durability is crucial. Protective enclosures for the Raspberry Pi and camera should be used to safeguard against environmental factors like dust, moisture, and accidental drops. This will ensure the system's longevity and reduce maintenance needs.

4.3.4 Connectivity and data management

Depending on the deployment scenario, the system may need to support data storage, transmission, or updates. Incorporating wireless capabilities (e.g., Wi-Fi, Bluetooth) would allow for remote updates, integration with other assistive technologies, and cloud-based data management.

5. CONCLUSIONS

In this paper, a real-time TTS embedded system is designed and evaluated to assist people with visual impairments. Our embedded system is based on the Raspberry Pi 4 and a Full HD camera. It integrates a new approach that is based on preprocessing and decision steps in order to increase the accuracy for detection, extraction, and recognition of the text by the OCR from an image or video. In addition, it uses the pyttsx3 Python library to convert the text to speech. The validation of our prototype has shown that our system allows users to listen to the recognized text in a natural voice. This auditory feedback enhances the reading experience for the visually impaired. In the end, we can conclude that our real-time text recognition system based on the Raspberry Pi 4 is a remarkable technological advancement that holds great promise for the visually impaired community. Its portability and real-time capabilities make it a valuable tool for enhancing accessibility and independence and empowering individuals with visual impairments to thrive in a sighted world.

ACKNOWLEDGMENTS

This work was funded by the Deanship of Graduate Studies and Scientific Research at Jouf University under (Grant No.: DGSSR-2024-02-02141).

REFERENCES

[1] Zaman, H.U., Mahmood, S., Hossain, S., Shovon, I.I. (2018). Python based portable virtual text reader. In 2018 Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA), Subang Jaya, Malaysia, pp. 1-6. https://doi.org/10.1109/ICACCAF.2018.8776778

[2] Sarkar, S., Pansare, G., Patel, B., Gupta, A., Chauhan, A., Yadav, R. (2021). Smart reader for visually impaired using raspberry Pi. IOP Conference Series: Materials Science and Engineering, 1132: 012032. https://doi.org/10.1088/1757-899X/1132/1/012032

[3] Shah, T., Parshionikar, S. (2019). Efficient portable camera based text to speech converter for blind person. In 2019 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, pp. 353-358. https://doi.org/10.1109/ISS1.2019.8907995

[4] Rithika, H., Santhoshi, B.N. (2016). Image text to speech conversion in the desired language by translating with Raspberry Pi. In 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, India, pp. 1-4. https://doi.org/10.1109/ICCIC.2016.7919526

[5] Velmurugan, D., Srilakshmi, Umamaheswari, S., Parthsarathy, S., Arun, K.R. (2016). Hardware implementation of smart reader for visually impaired people using Raspberry PI. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 5(3): 2055-2063. https://doi.org/10.15662/IJAREEIE.2015.0503132

[6] Gurav, M.D., Salimath, S.S., Hatti, S.B., Byakod, V.I., Kanade, S. (2017). B-LIGHT: A reading aid for the blind people using OCR and OpenCV. International Journal of Scientific Research Engineering & Technology, 6(5): 546-548.

[7] Chavan, S., Wandhekar, R., Tole, S., Chaukate, A., Chaukate, A. (2023). Text reader for visually impaired person using image processing/Open-CV. International Journal of Advanced Research in Science, Communication and Technology, 3(1): 193-199. https://doi.org/10.48175/IJARSCT-11233

[8] Long, S., He, X., Yao, C. (2021). Scene text detection and recognition: The deep learning era. International Journal of Computer Vision, 129(1): 161-184. https://doi.org/10.1007/s11263-020-01369-0

[9] Mithila, T., Arunprakash, R., Ramachandran, A. (2022). CNN and fuzzy rules based text detection and recognition from natural scenes. Computer Systems Science & Engineering, 42(3): 1165-1179. https://doi.org/10.32604/csse.2022.023308

[10] Xing, L., Liu, W., Liu, X., Li, X., Wang, H. (2022). Use of deep learning in nano image processing through the CNN model. Advances in Nano Research, 12(2): 185-195., https://doi.org/10.12989/anr.2022.12.2.185

[11] Zhong, Y., Liang, X. (2022). Using CNN-VGG 16 to detect the tennis motion tracking by information entropy and unascertained measurement theory. Advances in Nano Research, 12(2): 223-239. https://doi.org/10.12989/anr.2022.12.2.223

[12] Smith, R. (2007). An overview of the Tesseract OCR engine. In Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, pp. 629-633. https://doi.org/10.1109/ICDAR.2007.4376991

[13] Patel, C., Patel, A., Patel, D. (2012). Optical character recognition by open source OCR tool tesseract: A case study. International journal of computer applications, 55(10): 50-56. https://doi.org/10.5120/8794-2784

[14] Sporici, D., Chiroiu, M., Ciocîrlan, D. (2019). An evaluation of OCR systems against adversarial machine learning. In Innovative Security Solutions for Information Technology and Communications: 11th International Conference, SecITC 2018, Bucharest, Romania, pp. 126-141. https://doi.org/10.1007/978-3-030-12942-2_11

[15] Wu, J.C., Hsieh, J.W., Chen, Y.S. (2008). Morphology-based text line extraction. Machine Vision and Applications, 19: 195-207. https://doi.org/10.1007/s00138-007-0092-0

[16] Baskar, A. (2023), Digital Image Processing. Chapman & Hall/CRC.

[17] Burger, W., Burge, M.J. (2022). Digital Image

Processing: An Algorithmic Introduction. Springer Nature.
[18] Otsu, N. (1979), A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1): 62-66. https://doi.org/10.1109/tsmc.1979.4310076
[19] Atitallah, M.A.B., Kachouri, R., Atitallah, A.B., Mnif, H. (2022). An efficient HW/SW design for text extraction from complex color image. CMC-Computers, Materials & Continua, 71(3): 5963-5977. https://doi.org/10.32604/cmc.2022.024345
[20] 2013 ICDAR Dataset. https://paperswithcode.com/dataset/icdar-2013.
[21] Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., et al. (2013). ICDAR 2013 robust reading competition. In 12th International Conference on Document Analysis and Recognition, Washington, DC, USA, pp. 1484-1493. https://doi.org/10.1109/ICDAR.2013.221
[22] Fathy, A., Atitallah, A.B., Yousri, D., Rezk, H., Al-Dhaifallah, M. (2022). A new implementation of the MPPT based raspberry Pi embedded board for partially shaded photovoltaic system. Energy Reports, 8: 5603-5619. https://doi.org/10.1016/j.egyr.2022.04.035
[23] Liu, R., Li, Z., Jia, J. (2008). Image partial blur detection and classification. In 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, pp. 1-8. https://doi.org/10.1109/CVPR.2008.4587465