Vol 8 No 1 (2026) *e*-ISSN: 2682-8383

Journal of Engineering Technology and Applied Physics

YCL Based Smart Glasses for Visually Impaired

Wai Kit Wong[1, *], Yao Wei Loh[1], Wei Xiang Lee[1], Thiruppathy Kesavan V[2], Thu Soe Min[1] and Eng Kiong Wong[1]
[1]*Faculty of Engineering and Technology, Multimedia University, 75450 Jln Ayer Keroh Lama, Malaysia.*
[2]*Faculty of Information Technology, Dhanalakshmi Srinivasan Engineering College, Perambalur - 621212, Tamil Nadu, India.*
[*]*Corresponding author*: wkwong@mmu.edu.my, *ORCiD*: 0000-0003-1477-8449

https://doi.org/10.33093/jetap.2026.8.1.18
Manuscript Received: 16 October 2025, Revised: 19 November 2025, Accepted: 16 December 2025, Published: 15 March 2026

***Abstract*—A key component of assistive technology that helps people with visual impairments access text in their daily lives is a text reader. An image-based text recognition tool for the blind is presented in this article. It takes pictures of the user's surroundings using a dual-camera module. For improving character detection and recognition in these developed smart glasses for the blind and the visually impaired, a hybrid algorithm called the YCL Character Recognition Algorithm that combines You Only Look Once (YOLO), Convolutional Recurrent Neural Networks (CRNN), and Long Short-Term Memory (LSTM) to balance out the drawbacks of each algorithm by learning from its advantages. The suggested YOLO-v8 model is used for real-time text object detection, CRNN is used to extract character features, and LSTM is used to enhance sequential character prediction. An audio output signal is given to the user by the image processing software after the visual data has been processed. These smart glasses have the benefit of being able to view characters from both close and far distances. A specially acquired dataset was used to evaluate the suggested YCL technique, which shows notable speed and accuracy gains over the traditional homogenous algorithms. The suggested method is successful in identifying words and characters and in delivering audio output for the blind and visually impaired, based on users' surveys and experimental data.**

Keywords*—Assistive technology, Visually impaired, Character recognition, Machine Learning, Wearable device, Smart glasses.

I. INTRODUCTION

According to the Lancet Global Health Commission study from 2020, there were 510 million people with uncorrected close vision impairment and 596 million people with distance vision impairment [1]. Visually impaired have a difficult time exploring their surroundings, especially when it comes to accessing textual information in their daily lives. For instance, examining and evaluating printed text, labels, and signs in sophisticated settings. Due to the high cost and complexity of hardware setup, the majority of visually impaired did not use assistive technologies. This issue has gained widespread attention since visually impaired find it more difficult to do everyday tasks in a complex society.

Andrés A. Díaz and colleagues [2] presented a wearable assistive technology for those with vision impairments. To identify items in its immediate surroundings, the gadget makes use of a camera module. Four motors are used by the feedback system to vibrate to guide the user's movement. The fact that the camera and microcontroller are worn on the chest is one of the device's drawbacks. The camera cannot take an eye-level picture with this wearing technique. The four vibration-producing motors are worn around the waist. When vibrating regularly, the user may become more alarmed.

S. Nazim and colleagues [3] created smart glasses for the blind and visually impaired that can recognize and navigate. Ultrasonic sensors were incorporated into the smart glasses to enable object detection. The gadget uses facial and object recognition to let the user get information about their surroundings. The device uses a wired earpiece to provide the user with audible feedback once it has identified the object or face. For those who are blind or visually handicapped, these smart glasses work well. Nevertheless, the microcontroller is connected to several electrical components in the smart glasses. An excessive number of wired connectors complicates the device's component connections for the user.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License.
© Universiti Telekom Sdn Bhd.
Published by MMU PRESS. URL: https://journals.mmupress.com/index.php/jetap/index

To improve the user's autonomy and spatial awareness, D. Brilli and colleagues [4] created the wearable assistive technology known as Alris. Alris uses OCR to read text, detect objects, and describe scenes to provide the user with real-time explanations of their environment. The gadget has lagged when receiving user commands and feedback, which is one of Alris' drawbacks. Alris's reliance on cloud processing causes it to function badly in places with spotty internet.

V. Moran and colleagues [5] created smart glasses with Text-To-Speech (TTS), optical character recognition (OCR), and object detection built in. The user has the option to switch between object detection and text reading by pressing a switch button. Headphones or speakers are used to deliver audio output to the user. This device's weight is a drawback that makes prolonged use uncomfortable.

Nature Machine Intelligence [6] created a wearable AI system. The tool analyzes the environment using a camera using AI algorithms. By converting visual information into sensory impulses, the device enables the user to receive warnings and directional indications through elastic artificial skin on the wrists and bone conduction headphones. This tool's high level of intricacy is a drawback. Additionally, using the feedback system for long periods of time causes discomfort.

There are numerous assistive devices on the market too. The Dutch IT company Envision has developed an assistive gadget known as Envision Glasses [2]. Users with low eyesight and others who are visually challenged benefit from this technology. This technology's function enables it to scan textual information from the environment and transform it into an audio output signal that the user can hear. With a high-resolution camera, the technologies work well for capturing the complicated environment. However, when the information is far away, the technologies are unable to give the user very accurate results. According to the specifications [7], Envision Glasses can only detect text up to 0.6 meters.

Orcam, Inc. created a product called OrCam MyEye [8], which is another smart eyewear gadget for the visually impaired. This gadget helps those with vision impairments by offering item identification, facial recognition, and real-time text reading. By fastening to glasses, it provides hands-free operation, enabling users to get audio input without requiring manual intervention. Nevertheless, despite its many features, it still has drawbacks, such as a high price that prevents many users from affording it and a battery that needs to be charged frequently after two to three hours of use. Furthermore, dim illumination or intricate text layouts with skewed or slanted angles impair its accuracy because they are difficult for it to identify.

Therefore, a hands-free, lightweight, and intuitive character recognition solution is still needed despite the developments in assistive technologies for the visually impaired. The goal of this research is to create a wearable assistive technology that will enable the blind and visually impaired to identify words in natural settings. The device records textual data using imaging sensors and transforms it into audible feedback.

Smart assistive technologies that can read and interpret text from natural scenes and give users real-time feedback have been made possible by recent developments in computer vision and machine learning. Processing images from a wearable camera using object detection and word recognition models is one of the more promising methods in this area. While OCR is effective at identifying text or characters in documents or real-world scenes, it cannot identify complex backgrounds, distorted text, or non-standard fonts [9]. This is also true for most character recognition methods, such as morphology operation [10], template matching [11], connected component analysis [12], and others. According to a comprehensive study in [13], machine learning is essential for smart glasses because it excels at processing low-quality, noisy, and distorted images, understanding word structures and grammar for improved accuracy, and learning from new datasets to improve over time.

Machine learning uses a variety of techniques, including reinforcement learning, supervised learning, and unsupervised learning. For this software-based vision system, supervised learning was selected due to its superiority in character identification tasks. In contrast to unsupervised learning, which necessitates large datasets and high computational resources, or reinforcement learning, which is time- and resource-intensive, the supervised learning approach is effective across various scripts, fonts, and styles using techniques like decision trees and neural networks and performs well with labeled datasets [14 - 17].

In this paper, an image-based word recognition approach for the visually impaired is outlined. Word recognition at 3 cm to 3 m distances is made possible by the proposed tool. To improve user freedom, it makes use of a wearable, high-resolution imaging system. The auditory feedback provided by the TTS system is precise and understandable. The tool provides accurate and intelligible feedback about the user's environment through the TTS system. Additionally, a power-saving mode is developed to reduce energy consumption and greatly increase the tool's operational time, making it more practical for daily usage. Additionally, a YCL character recognition technique is put forth, which combines a the YOLOv8 object detection model with a Convolutional Neural Network - Long Short-Term Memory (CNN-LSTM) model for character recognition. YOLOv8 is used to detect individual character regions from input images, while the CNN-LSTM model recognizes sequences of characters, allowing the system to efficiently handle multiple-character inputs. The suggested technique uses a two-stage machine learning pipeline, where detection and recognition are handled by two different models for increased robustness, in contrast to the conventional smart glasses algorithm, which only uses OCR to detect and recognize the character. It was also trained using custom-captured datasets, such as label photos of character instances from various settings and handwritten characters. With an accuracy of over 85%, the suggested smart glasses for the blind and visually impaired can give the user fast feedback and high accuracy by utilizing YOLOv8 model for character detection and CNN-LTSM model for character recognition. The structure of the paper is as follows: The system framework for the Smart Glasses is presented in Section II. The system's algorithm models are introduced in Section III. The experimental results are presented in Section IV. The conclusion and upcoming developments are finally presented in Section V.

II. SYSTEM FRAMEWORK

The proposed Smart Glasses system framework is shown in Fig. 1. The image capture process is run in the head portion (H), which is the wearable Glasses. The circuit between the Control Panel and Power Supply H is opened and closed by Power Activator H, which regulates the ON and OFF functions. The imaging tool is controlled by the control panel, which turns the environment into visual data. On the waist belt portion, the Power Activator W and Power Activator H serve the same purpose placed in the waist area. The Control Panel provides the visual data to the Image Processor. With an audio speaker, the image processor transforms the image into audio data. The following subsections will discuss the components of the wearable glasses and waist belt in detail.

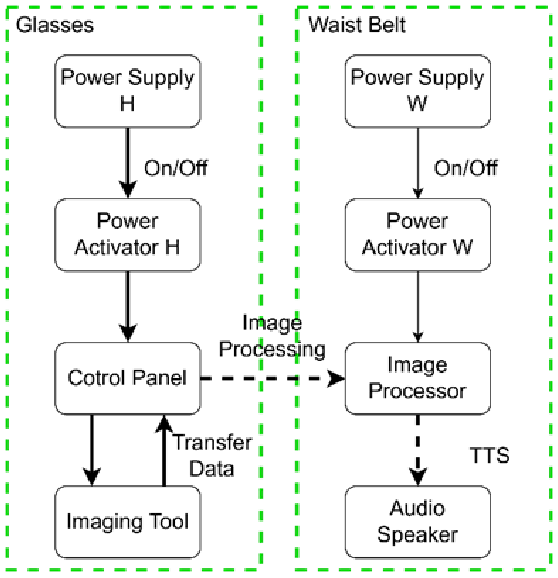


Fig. 1. Smart glasses system framework.

A. Wearable Glasses

The setup for the components used to gather visual information from the environment is displayed by the wearing glasses placed in the head portion (H). They are imaging tool, control panel, power activator and the power supply. The purpose of the imaging tool is to record visual data captured from the user's environment. The control panel serves as a user interface that enables the user to take pictures.

i. Imaging Tool

The imaging tool is used to capture the word from various distances. A high-resolution camera is required to address this issue and enable a more detailed acquired image. The Raspberry Pi HQ Camera was chosen for this project because of its 12-megapixel camera sensor. Connecting an external lens is made possible by the camera sensor. The width and height of a captured sample image is shown in Fig. 2.

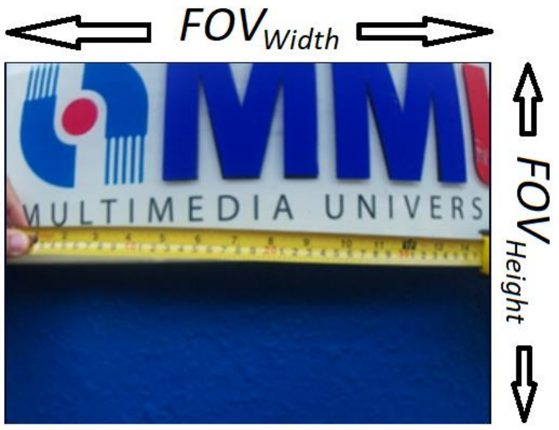


Fig. 2. Width and height of a captured sample image.

The imaging tool to determine θ_{width} (Horizontal Angle of View) in radians is denoted by the Eq. (1) [18]:

$$\theta_{width} = 2 \times \arctan\left(\frac{d_{width}}{2f}\right) \qquad (1)$$

The camera sensor's width, measured in millimeters, is denoted by d_{width}. f stands for the lens's focal length in millimeters. The FOV_{width} represents the width of visible area:

$$FOV_{width} = 2 \times D \times \tan\left(\frac{\theta_{width}}{2}\right) \qquad (2)$$

, where D is the captured object's distance from the camera, measured in meters. θ_{height} (Vertical Angle of View) in radians is denoted by the Eq. (3) [12]:

$$\theta_{height} = 2 \times \arctan\left(\frac{d_{height}}{2f}\right) \qquad (3)$$

The camera sensor's width, measured in millimeters, is denoted by d_{height}. f stands for the lens's focal length in millimeters. The FOV_{height} represents the height of visible area:

$$FOV_{height} = 2 \times D \times \tan\left(\frac{\theta_{height}}{2}\right) \qquad (4)$$

Three cases scenarios been studied to model the suitable lenses, as shown in Fig. 3. A 6 mm external lens was selected for the developed prototype based on the calculations mentioned above. Although the 6mm lens has a smaller capture range, it may capture the image with more detail when zooming in from a distance.

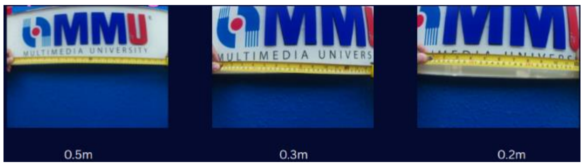


Fig. 3. Three case scenarios to decide lenses (0.5m, 0.3 m and 0.2m).

ii. Control Panel

To control the imaging tool and wirelessly send visual data to the image processor, the Raspberry Pi Zero 2 [19], as shown in Fig. 4, serves as the control panel. Bluetooth and Wi-Fi modules are integrated inside the microcontroller to enable wireless data transfer. Additionally, it works effectively with the Raspberry Pi HQ Camera via the camera port. To guarantee a steady connection, a 15-pin to 22-pin camera cable is utilized. The user interface is a 6-by-6-by-1 4-pin push button that the user can press to take a picture.

Fig. 4. Raspberry Pi Zero 2.

iii. Power Activator H and Power Supply H

An 5000mAh LiPo battery serves as the prototype's power source. The Raspberry Pi Zero 2 and Raspberry Pi HQ Camera's power requirements were taken into consideration when choosing the battery. With a high-capacity battery, the system can run for longer periods of time between charges. The following formula can be used to determine battery capacity:

$$E = \sum_{n=1}^{N} a_n t_n + L \qquad (5)$$

E denotes for the battery's energy capacity, where a_n is the power rate of n_{th} block. t_n is the total time taken of n_{th} block. N are the total blocks in the model. L is the power loss due to the n_{th} block.

Since there are three blocks in the suggested framework in Fig. 2, N = 3, and the power loss L is minimal. So, the Eq. (5) becomes:

$$E = PT = a_1t_1 + a_2t_2 + a_3t_3 \qquad (6)$$

, where a_1 is the power rate of power activator. a_2 is the power rate of Raspberry Pi Zero 2 and a_3 is the power rate of Raspberry Pi HQ Camera. t_1 is the time taken for power activator, t_2 is the time taken for Raspberry Pi Zero 2 and t_3 is the time taken for Raspberry Pi HQ Camera. In full operation power mode, $t_1 = t_2 = t_3$ as these blocks has used the same time taken. However, under power saving mode, $t_1 \neq t_2 \neq t_3$ due to some relevant blocks are turned off during operation.

Both full power and power-saving modes are supported by the developed prototype. The prototype performs at its best when it is in maximum power mode. Continuous high-performance operation, however, uses more energy and shortens the prototype's operational time. Furthermore, over time, frequent charging reduces battery capacity. Given the additional weight of larger batteries, utilizing a power-saving mode is a more efficient solution to this problem than merely switching to a higher-capacity battery.

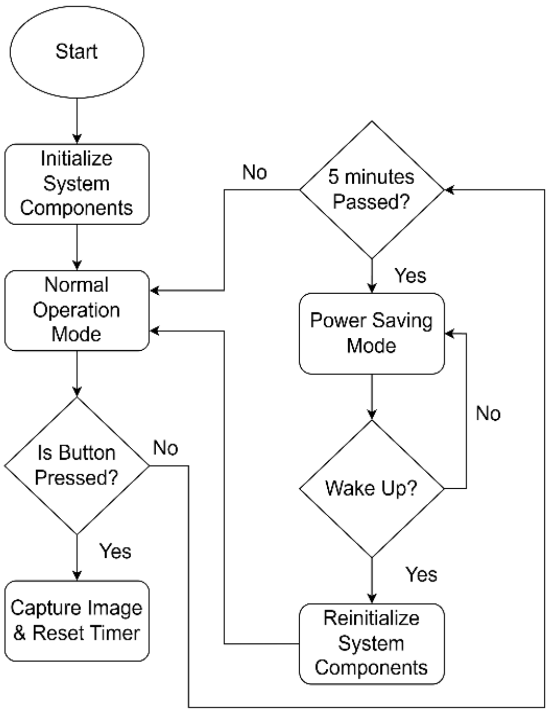


Fig. 5. Power saving mode flowchart.

The flowchart for turning on the prototype's power-saving mode is shown in Fig. 5. The system initializes the system component when the tool begins to function. The system enters Normal Operation Mode to actively monitor user activities after it has been initialized. The user's pressing of the button is constantly monitored by the system. The system instructs the camera module to capture an image and resets the inactivity timer when the button is pressed. The mechanism begins calculating the five minutes of idleness if the button has not been pressed. The system is still in Normal Operation Mode notwithstanding the detection of user involvement. Nevertheless, the device automatically switches to Power Saving Mode after detecting that five minutes had elapsed. When the system is in the Power Saving Mode, it is in a low-power state and doesn't do anything until it senses a wake-up signal, at which point the user clicks the button. The wake-up indicates that every component needs to be reinitialized to return to a fully functional condition.

B. Waist Belt

The waist section contains the image processing unit that processes the captured image. After processing the image using software, the visual information is converted into text. Then, the text information is converted into audio information and transmitted through the audio speaker.

i. Image Processor

The image processor is the Raspberry Pi 4, as seen in Fig. 6. It has 8GB of RAM, which offers adequate performance to manage high-resolution photos and effectively finish challenging image processing jobs. Additionally, it has Bluetooth and Wi-Fi integrated in to facilitate data transfer to the imaging devices (ESP32 Cam) and audio speaker.

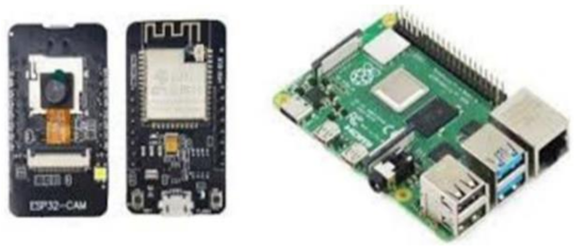


Fig. 6. ESP32 Cam and Raspberry Pi 4.

ii. Audio Speaker

The audio data from the image processor is received using wireless earbuds. They are favoured over wired earbuds because too much wiring makes it more difficult for the user to manipulating the prototype. Most wireless earbuds available on the market have a "Transparency Mode" that lets users perceive background noise.

iii. Power Activator W and Power Supply W

The Power Activator W and Power Supply W have the similar function as the components in the wearable glasses on head section. Since the image processor uses more energy to conduct processing duties and transmit audio, a 10,000mAh LiPo battery is used as the power source for Power Supply W.

iv. Text-to-Speech (TTS**)**

The Raspberry Pi 4 has built-in TTS module. Figure 7 shows a detailed flowchart of the Text-to-Speech (TTS) process. Upon receiving the identified text from the image processor, the TTS process begins. A simple check is performed to ensure the text isn't too short or incomplete. Simple formatting is used to enhance clarity if the text is valid, such as adjusting punctuation and spacing. Following formatting, the system chooses the voice and language options according to the specifications. To produce the voice output, the prepared text is subsequently run via a TTS engine. Before transferring the audio, the system makes sure the earbuds' wireless connection is enabled.

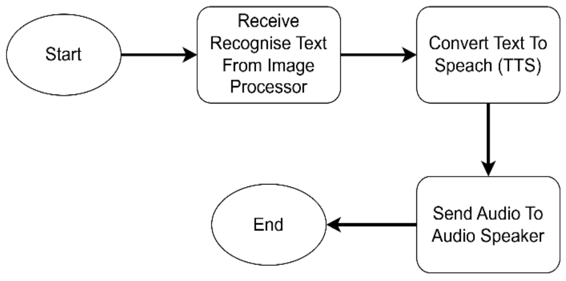

Start
Receive Recognise Text From Image Processor
Convert Text To Speach (TTS)
Send Audio To Audio Speaker
End

Fig. 7. TTS flowchart.

III. YCL ALGORITHM MODEL

The proposed YCL Algorithm Model for the Smart Glasses system framework is shown in Fig. 8. It is the result of combining the YOLOv8 and CRNN algorithms. YOLOv8 is preferred over the newer YOLOv12 due to YOLOv8's established ecosystem, extensive community support, proven stability, and simpler architecture, which can be advantageous in production environments or for us as developers with specific hardware constraints. With its bounding box and real scene character identification capabilities, the YOLOv8 is used for character detection [20]. In the meanwhile, CRNN combines CNN to automatically extract significant features from raw images with LSTM to long-term learn and retain patterns in sequential data, primarily handwriting characters [21, 22].

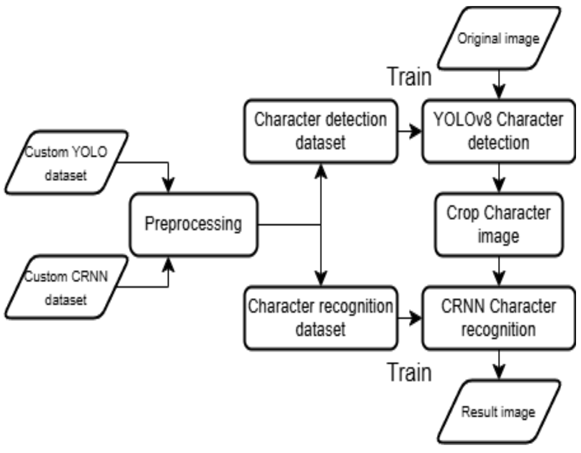

Original image
Train
Character detection dataset
YOLOv8 Character detection
Custom YOLO dataset
Preprocessing
Crop Character image
Custom CRNN dataset
Character recognition dataset
CRNN Character recognition
Train
Result image

Fig. 8. Flowchart of YCL Character Recognition Model.

A. Preprocessing

The preprocessing stage is carried out for the custom dataset prior to training and image prediction. Character detection is done with the custom YOLOv8 dataset, and character recognition is done with the custom CRNN dataset. Preprocessing entails more than just scaling the image; it also entails creating customized character labels for CRNN recognition and an annotation point for the bounding box for detection.

B. YOLOv8 Character Detection

With the usage of a custom dataset in which each character is labeled with a bounding box and the ground truth character for each image, YOLOv8 is used to recognize characters from the original image.

The proposed YOLOv8 model for identifying characters in complicated images is depicted in Fig. 9. Moreover, Fig. 10 shows the location of data.yaml that indicates the bounding box and ground truth character addresses for the YOLOv8 to detect. The Intersection over Union (*IoU*), also known as the Jaccard index, is used:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{7}$$

, where Area of Overlap is the overlapping region between the predicted and ground truth areas, and Area of Union is the total area covered by both the predicted and ground truth areas combined, with the overlapping region counted only once. IoU values range from 0 to 1. An IoU of 0 indicates no overlap between the predicted and ground truth areas. An IoU of 1 indicates a perfect match or complete overlap.

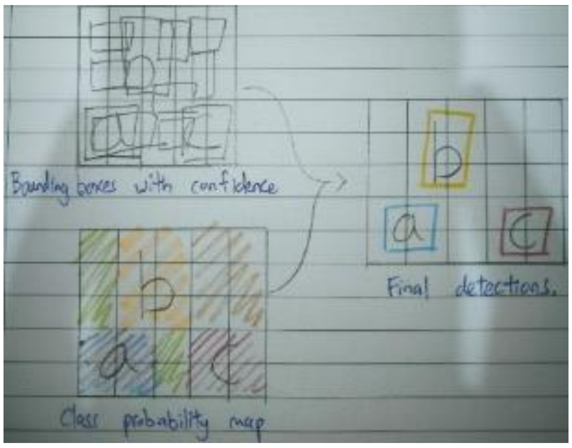
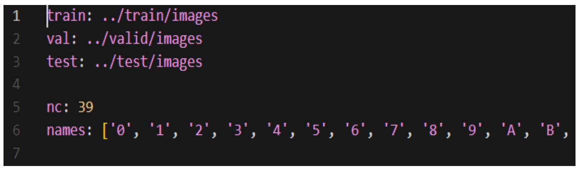


Fig. 9. YOLOv8 process sketch.



```
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 39
names: ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B',
```

Fig. 10. Location of data.yaml.

A. Crop Character Image

The bounding box coordinates (*x_min*, *y_min*, *x_max*, *y_max*) are used to crop off individual characters from the source image once YOLOv8 has detected it:

$$I_{crop} = I(x_{min}, y_{min}, x_{max}, y_{max}) \tag{8}$$

, *I* stand for the original image, and this formula makes sure that only character-rich, pertinent areas are sent to CRNN for recognition, increasing overall accuracy.

B. CRNN Character Recognition

Once the original image has been cropped, it will be fed into a CRNN, which has an LSTM (Long Short Term Memory) layer to collect sequential information and a convolutional layer to extract character features, for character recognition.

Convolutional layer: The convolutional layer converts the original image *I* to grayscale so that spatial features can be easily extracted. Following the convolutional layer, *f* is the output of the flattened feature vector:

$$f = \frac{input\ size - kernel\ size + 2 * padding}{stride} + 1 \tag{9}$$

Reshape: Assuming that the *f* is (batch size, height, width, channels) after CNN, it must be reshaped into (batch size, time steps, channels) because the LSTM only accepts 1D vectors, which must be converted to 1D vectors by multiplying width by height.

LSTM layer: The LSTM layer only accepts 1D vectors since it functions similarly to a character reading sequence and contains three gates: input, output, and forget. It also aids in resolving the vanishing gradient issue:

$$h_t = LSTM(x_t, h_t, c_{r-1}) \tag{10}$$

, where x_t is the input of the time and c_t is the cell state of the time and last the h_t is the hidden state at time. Additionally, this LSTM has bidirectional capabilities to record sequential dependencies in both forward and backward directions, encompassing past and future context.

Dense layer: The dense layer can process without changing the vector because the LSTM is already at a 1D vector. To understand patterns and relationships of the character, the dense layer consists of several neurons working together to analyse characteristics that were extracted by the convolutional, max pooling, and LSTM layers.

C. Result Image

Following the completion of the CRNN's recognition of the cropped image, the result image will be displayed along with a text file that has been saved in a folder for future processing or TTS for the smart glass. As seen in Fig. 8, this completes the overall YCL character recognition pipeline.

IV. EXPERIMENTAL RESULTS

The experimental outcomes of the proposed smart glasses hardware prototype and the performance assessment of the recommended YCL algorithm are presented in this section.

A. Hardware Prototype Evaluation

The project prototype, depicted in Fig. 11 and Fig. 12 has been constructed. The experiment was conducted at Center for Advanced Robotics Laboratory, Multimedia University in Melaka. The appropriate external camera lens for the imaging tool was determined by applying Eqs. (1) - (4). The dimensions of the images taken with the 3mm and 6mm lenses are shown in the findings. The Raspberry Pi HQ Camera's sensor measures 6.287 mm in width and 4.712 mm in height. The FOV angles for 3 mm and 6mm lenses are 92.88° (W) × 76.18° (H) and 55.37° (W) × 42.96° (H), respectively, according to Eqs. (1) and (3).

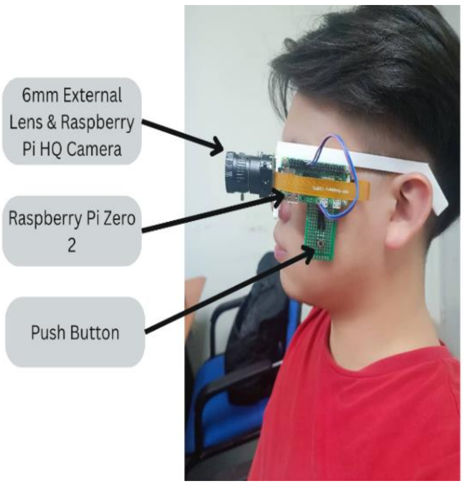


Fig. 11. Hardware prototype of wearable smart glasses.

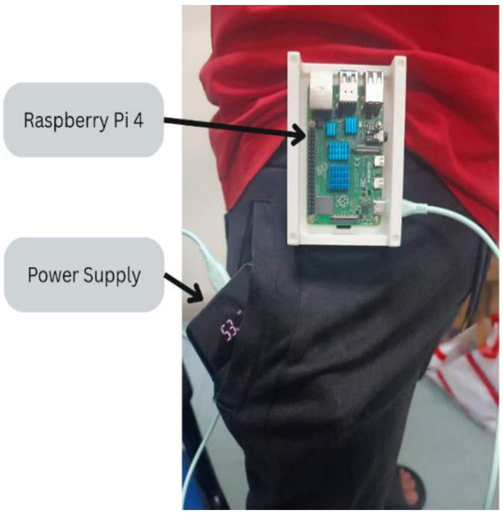


Fig. 12. Hardware prototype of the waist belt.

Table I. Calculated dimensions using Eqs. (2) and (4) for 3 mm and 6 mm Lenses.

Distance (m)	3mm Lens FOV (m^2)	6mm Lens FOV (m^2)
0.03	0.063 × 0.047	0.0316 × 0.0236
0.1	0.21 × 0.16	0.105 × 0.079
0.2	0.42 × 0.31	0.211 × 0.157
0.3	0.63 × 0.47	0.316 × 0.236
0.5	1.05 × 0.78	0.527 × 0.394
1	2.10 × 1.57	1.053 × 0.787
1.5	3.15 × 2.35	1.580 × 1.181
2	4.20 × 3.13	2.106 × 1.574
2.5	5.25 × 3.92	2.633 × 1.968
3	6.30 × 4.70	3.160 × 2.361

The 3 mm external lens offers a broader viewing angle than the 6 mm external lens, according to the results listed in Table I. It provides greater viewing dimensions at different distances. However, it has a lesser level of magnification and cannot create very detailed images. When it comes to duties like reading text from a book or recognizing signs from a distance, the 6mm external lens, on the other hand, offers superior clarity for taking pictures at both short and long distances. Additionally, it is a better option for capturing little characters with greater detail and clarity.

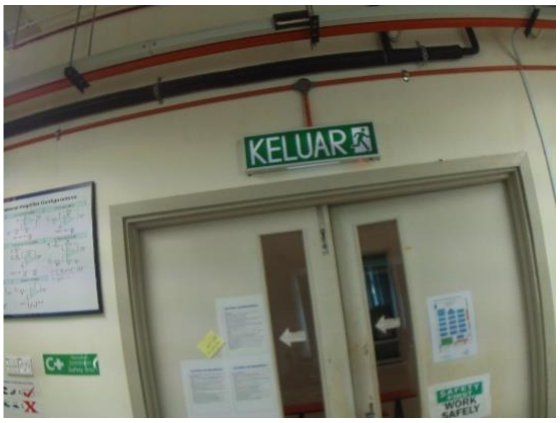


Fig. 13. Prototype captured image from near distance.

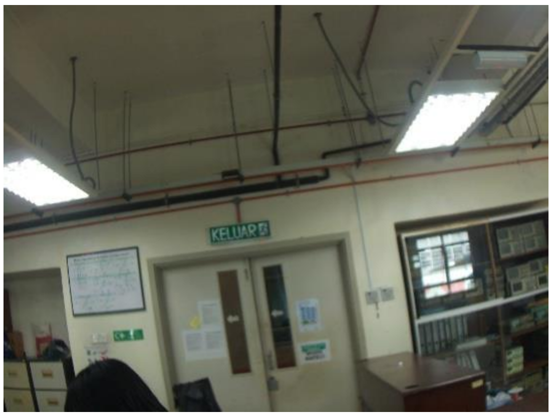


Fig. 14. Prototype capture image from far distance.

The pictures taken with a 6mm external lens at various distances are shown in Fig. 13 and Fig. 14. The image seems crisp and clear when placed close to the "KELUAR" sign. However, the image becomes less clear and shows more noise as the distance rises. For the observed green card on the door left hand side, at closer range, the picture for the character "C" is apparent, but at greater distances, it becomes less distinct. This test shows how well the 6mm external lens captures words with varying font sizes.

When designing wearable and portable electronics, power consumption has gained in importance. Power-saving mode is required to decrease the power drain from the power supply to solve this problem. In the absence of a power-saving mode, power consumption stays high, resulting in shorter running times and faster energy depletion. Assuming $a_1 = a_2 = a_3 = 1$, the energy consumption over time can be approximated using Eq. (6).

$$PT = a_1t_1 + a_2t_2 + a_3t_3$$

$$T_{FullPower} = 3t_1$$

$$t_{delay} = T_{PowerSaving} - 3t_1$$

$$\begin{aligned} T_{PowerSaving} &= 2t_{delay} + t_1 \\ &= 2T_{PowerSaving} - 6t_1 + t_1 \end{aligned}$$

$$2T_{PowerSaving} = 5t_1$$

$T_{PowerSaving} \quad = 2.5t_1$

The energy consumption calculation in full power mode is displayed in the calculations above. The overall energy consumption is shown as $3t_1$ to reflect the total energy utilized during the operation process because each block runs for the same length of time. However, the value changes to $2.5t_1$ in power-saving mode. The device was used for an hour during the testing phase. Table II tabulated the time usage for Full Power and Power Saving Modes. The graph in Fig. 15 below displays the findings.

Table II. Time usage for full power and power saving modes.

Time (m)	Power Mode (mAh)	Saving Mode (mAh)
1	3	3
5	15	14.5
10	30	28
15	45	40.5
20	60	53
30	90	78
40	120	103
50	150	128
60	180	153

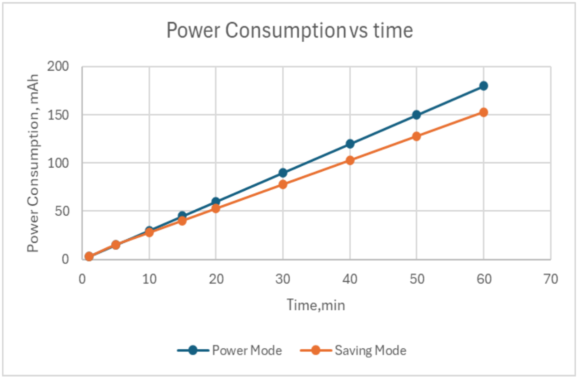


Fig. 15. Graph plotted of Table II.

The plotted graph and experimental findings demonstrate that the power-saving mode successfully lowers the prototype's power usage. The power-saving mode used 15% less energy than the full power mode after an hour of operation. As a result, battery recharges occur less frequently, and operating time is increased.

Table III. Cost of developing the prototype (per unit).

No.	Item	Price (RM)	Quantity	Total (RM)
1	Raspberry Pi 4 Model B (8GB)	369	1	369
2	Raspberry Pi 1.6MP Global Shutter Camera Module	249	2	498
3	Raspberry Pi 6mm Wide Angle Lens (CS Mount)	125	2	250
4	Raspberry Pi Zero 2	90	2	180
5	6x6x1 Push Button 4-Pin	0.50	1	0.50
			Total (RM)	1297.50

Table III shows how much it costs to develop the prototype. The image processor on the list was an RM369.00 Raspberry Pi 4 Model B (8GB). Two sets of Raspberry Pi 1.6MP Global Shutter Camera Modules, each costing RM249.00, have been used. Two Raspberry Pi 6mm Wide Angle Lenses (CS Mount), each costing RM1125.00, have been used. Two Raspberry Pi Zero 2s, each costing RM90.00, are used as control panels. A 6x6x1 Push Button 4-Pin has been used with a cost of RM0.50. The total quantity of materials needed to accomplish the project's goal and functionality is RM1297.50.

The glasses' real dimensions are about 0.19 meters in height and 0.28 meters in width. The glasses weigh 850 grams in total, including all installed parts and structural support. The weight of the waist-mounted Raspberry Pi 4 device is about 100 grams. The entire prototype system weighs about 950 grams. A 20,000 mAh power bank for the Raspberry Pi 4 and a 5,000 mAh supply for each Raspberry Pi Zero 2 make up the battery system. When combined, these offer all modules steady power delivery. The system can function continuously for about 1.5 to 2 hours under normal circumstances before needing to be recharged.

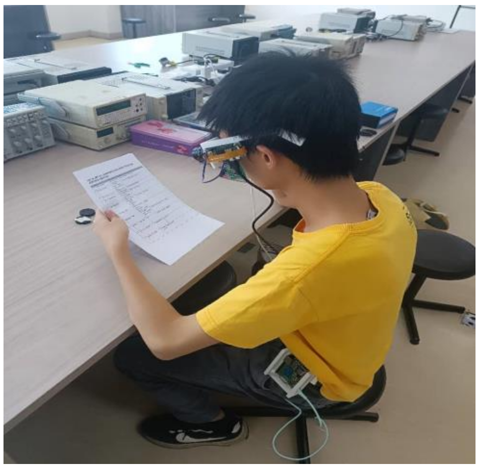


Fig. 16. Participant capturing an image from an article using the developed prototype.

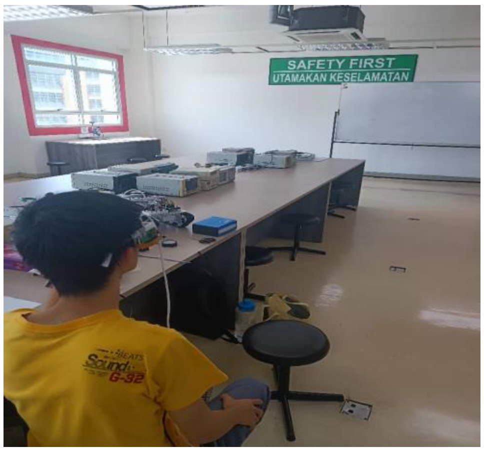


Fig. 17. Participant capturing the word "SAFETY FIRST" from a signboard using the developed prototype.

Multimedia University was chosen as the site of tested the prototype of Image-Based Word Recognition Tool for the Visually Impaired. 32 participants in all were told to utilize the program to record text from their surroundings. Figure 16 and Fig. 17 illustrate the activities, which included reading a signboard from a distance and extracting text from an article. Participants were able to correctly identify 88

e-ISSN: 2682-8383

out of 100 words, or 92.5% of the total. Many participants found it challenging to read small prints at a distance during the study. The absence of autofocus was the cause of this problem, which led to hazy photos. Additionally, the gadget was heavy and difficult for individuals to use for prolonged periods of time. Another issue was battery life because of the short operational period, which necessitated regular recharging. To increase the usage time, participants recommended using a high-capacity battery.

Participants completed a survey using a Likert scale, and the results are presented in Table IV below:

Table IV. Survey results collected using Likert scale.

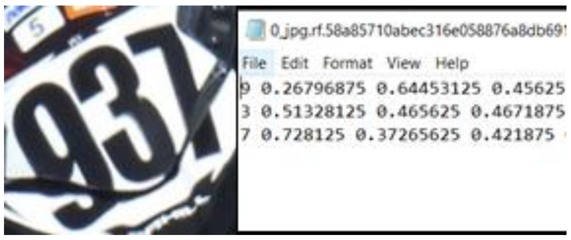
	Number of Users Answered				
	Strongly Disagree	Disagree	Neutral	Agree	Strong Agree
Q1. The tool was easy to use	0	0	7	8	15
Q2 The tool helped me recognize text efficiently	0	0	4	12	16
Q3 I would use this tool for daily activities	0	0	6	15	11

75% of participants found the tool was simple to use, effective at helping them recognize text, and helpful for everyday tasks. Nonetheless, a few participants voiced worries regarding the tool's weight and battery life. Participant recommendations for improvement included the following:

1. **Autofocus Implementation:** The camera modules in the current version lack a focusing function, which lowers image clarity. In order to improve the tool's ability to recognize text in various directions, participants recommended adding this feature.
2. **Weight Reduction:** Because there are so many components, the present version is somewhat heavy. In order to minimize total weight, participants suggested use a more adaptable camera module that could take pictures at various distances using just one lens and module.
3. **Battery Life Extension:** The current version's significant power consumption from several components necessitates frequent recharging. For more effective power utilization, participants recommended expanding the battery's capacity and enhancing the wire connection to the battery.
4. **Button Functionality:** The current version controls both short-range and long-range image capturing with a single button. A tactile feedback system to show whether the instrument is in long-distance or short-distance capture mode was suggested by the participants.

B. YCL Algorithm Performance Evaluation

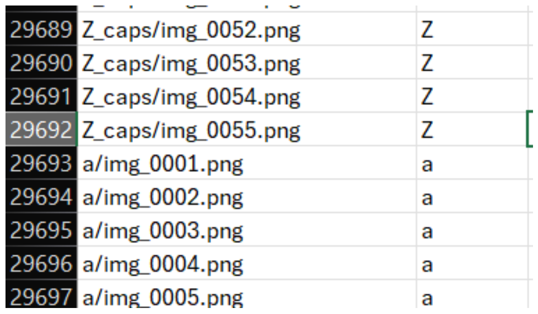
This section demonstrates the YCL results, including the setup of the experiment, the training and prediction results. The experiment setup comes first, and it begins with preparing the custom dataset. The datasets for YOLOv8 and CRNN are distinct; YOLOv8 include not just the train image but also data.yaml, which is the most crucial file for YOLOv8. Fig. 10 already indicates where the data.yaml file and ground truth label are located. The bounding box image and coordinates for YOLOv8 training are displayed in Fig. 18.


0_jpg.rf.58a85710abec316e058876a8db69
File Edit Format View Help

```
9 0.26796875 0.64453125 0.45625
3 0.51328125 0.465625 0.4671875
7 0.728125 0.37265625 0.421875
```

Fig. 18. Example of bounding box image and coordinates labels for it.

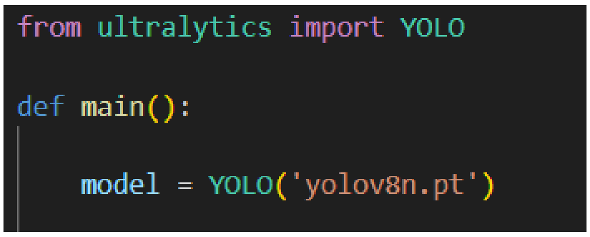
The image path and the name for each image are the two primary components of a separate CRNN dataset. The image path and ground truth label for the set image are shown in Fig. 19. CRNN needs a ground truth label for every image during training because it makes use of LSTM for prediction.



29689	Z_caps/img_0052.png	Z
29690	Z_caps/img_0053.png	Z
29691	Z_caps/img_0054.png	Z
29692	Z_caps/img_0055.png	Z
29693	a/img_0001.png	a
29694	a/img_0002.png	a
29695	a/img_0003.png	a
29696	a/img_0004.png	a
29697	a/img_0005.png	a

Fig. 19. Image path and corresponding ground truth label for each image.

Secondly, the YOLOv8 setup for training is simple in comparison to later CRNN models because it uses a pretrain model, which eliminates the need to create a new YOLOv8 model from scratch. This research uses Nano YOLOv8, which requires less processing power to operate, as seen on Fig. 20.



```
from ultralytics import YOLO

def main():

    model = YOLO('yolov8n.pt')
```

Fig. 20. Import pretrain YOLOv8 model.

Following that is convolutional layer and max pooling layer which the input image is 32 x 32, with 3 x 3 kernel size, 1 stride, and 0 padding. The outcome of the three convolutional and max pooling layers is 2 x 2 x 256, which is then used for the LSTM layer. Therefore, it must be reshaped into a 1D vector, making it 4 x 256 with four 1D vectors to enter the LSTM model [21, 22].

The LSTM itself is composed of three gates, signifying that it has equations for input, output, and forget gates. It also requires an activation function because the Sigmoid and Tanh functions are frequently used to calculate long and short-term memory. Tanh function scales the cell state and outputs between -1 and 1 to preserve learning stability, while Sigmoid function regulates gate activations between 0 and 1, determining what to keep or discard [23]. The LTSM model's equations are highlighted below:

1.) Forget Gate:

$$f = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{11}$$

2.) Input Gate:

$$i = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{12}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t$$

3.) Output Gate:

$$o = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \cdot tanh(c_t) \tag{13}$$

4.) Sigmoid Activation Function:

$$f(x) = \frac{e^x}{e^x+1} \tag{14}$$

5.) Tanh Activation Function:

$$f(x) = \frac{e^x-e^{-x}}{e^x+e^{-x}} \tag{15}$$

Consider that the cell state is 2, the input size is 4, hidden, and that the values for a single time step up are $x_t = (0.1, 0.2, 0.3, 0.4)$, $h_{t-1} = (0.5, -0.5)$ and $c_{t-1} = (0.7, -0.3)$. For initialization, the weight will be set to 1 and the biases to 0, and the $[h_{t-1}, x_t]$ will become (0.5, -0.5, 0.1, 0.2, 0.3, 0.4). The LSTM may learn temporal patterns by using the weight and bias values to decide how much of the input and historical data should be retained or forgotten.

Hence the forget gate become:

$$f_t = \sigma(1 * 0.5 + 1 * -0.5 + 1 * 0.1 + 1 * 0.2 + 1 * 0.3 + 1 * 0.4 + 0)$$

$$f = \sigma(1.0)$$

$$f = 0.731 = i_t = h_t$$

Same for input gate:

$$\tilde{c}_t = tanh(1.0) = 0.761$$

$$c_t = 0.731 * 0.7 + 0.731 * 0.761$$

$$c_t = 1.068$$

and Output gate:

$$h_t = 0.731 * \tanh(1.068)$$

$$h_t = 0.731 * 0.789$$

$$h_t = 0.577$$

Equations (11), (12), (13), (14) and (15) enable LSTM model to learn dependencies from character input, including handwriting, by updating both cell memory and hidden state based on previously provided information.

Following the completion of the experiment setup, the model starts the training process and displays the YCL model's results. YOLOv8's training following detection results with the custom dataset are displayed in Fig. 21. Out of the 21400 total custom datasets, 19800 were utilized for training and 1600 were used to validate the YOLOv8. The findings are shown in Fig. 21.

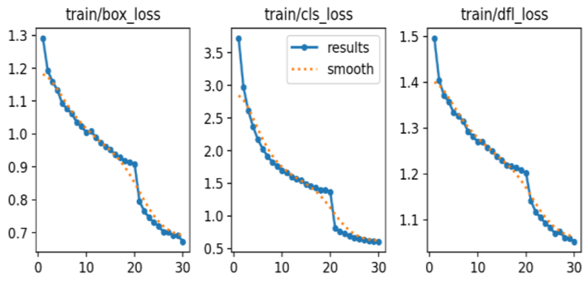


Fig. 21. Training result from YOLOv8.

The overall training losses for the proposed YOLOv8 model are shown in Fig. 21, which runs from left to right and measures the classification loss, the error of bounding box alignment with the ground truth, and the Distribution Focal Loss (DFL) respectively, that aids in improving the accuracy of bounding box coordinates. The graph's plotted data demonstrate that each of these losses drastically decreases around epoch 20, suggesting that the YOLOv8 model gains a significant amount of knowledge during that time. This speeds up training and increases the model's learning rate.

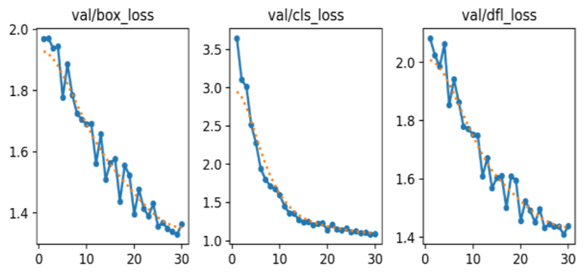
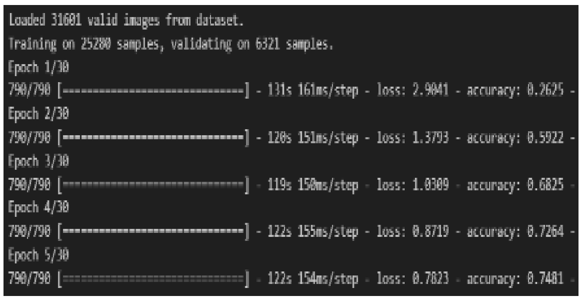


Fig. 22. Validation losses for YOLOv8.

The validation losses of the YOLOv8, which measures the model's performance on unseen data, are shown on Fig. 22. The validation losses functioned similarly to the training losses, measures the classification loss, the error of bounding box alignment with the ground truth, and the Distribution Focal Loss (DFL) respectively, but only for validation. The result in Fig. 22 indicates that while the model struggles with unknown data, it generally declines, indicating that it is relying on unseen data. Validation loss fluctuates a bit on unseen data, but overall, the model continues improving with decreasing loss.

Once the YOLOv8 training phase has ended, the CRNN training phase begins. The performance of CRNN model undergoing training is depicted in Fig. 23. Even though CRNN uses 30,000 images for training, its overall training pace is significantly faster compared to other parallel methods.



```
Loaded 31601 valid images from dataset.
Training on 25280 samples, validating on 6321 samples.
Epoch 1/30
790/790 [==============================] - 131s 161ms/step - loss: 2.9041 - accuracy: 0.2625 -
Epoch 2/30
790/790 [==============================] - 120s 151ms/step - loss: 1.3793 - accuracy: 0.5922 -
Epoch 3/30
790/790 [==============================] - 119s 150ms/step - loss: 1.0309 - accuracy: 0.6825 -
Epoch 4/30
790/790 [==============================] - 122s 155ms/step - loss: 0.8719 - accuracy: 0.7264 -
Epoch 5/30
790/790 [==============================] - 122s 154ms/step - loss: 0.7823 - accuracy: 0.7481 -
```

Fig. 23. Training result from proposed CRNN model.

The training accuracy and losses for CRNN for 30 epochs are displayed on Fig. 24 and Fig. 25. The entire custom dataset for CRNN is 31600, of which 25280 are used for training and 6320 for validating the CRNN. As seen on Fig. 24, the validation for accuracy and losses had a dramatic decline and surge during epoch 11 before returning to normal. The abrupt decrease and rise could be caused by the model's need to validate over 6000+, which it has never seen before. As a result, it struggles a little at first but eventually bounces back and achieves 88% accuracy with losses of less than 5%.

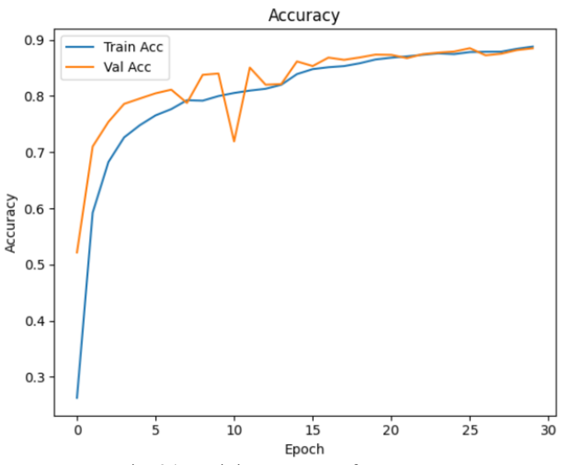


Fig. 24. Training accuracy from CRNN.

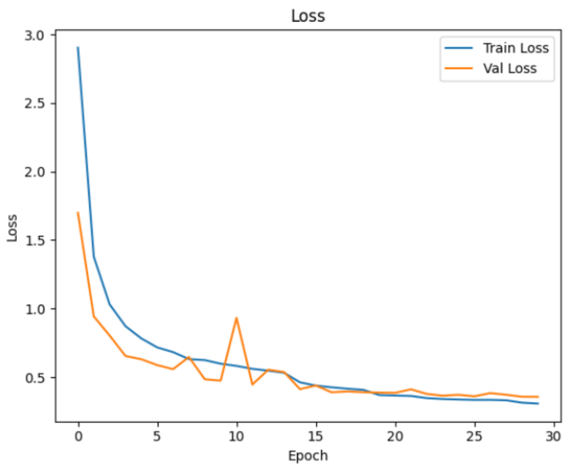


Fig. 25. Training losses from CRNN.

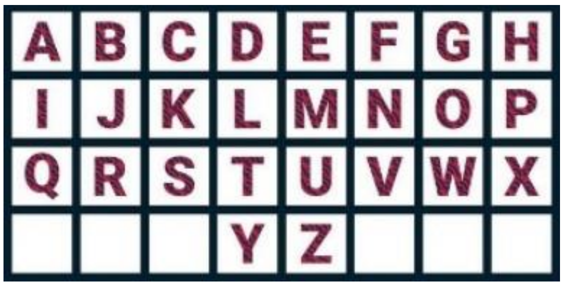


Fig. 26. Sample of each alphabet.

The YCL model prediction result comes last in the experiment results. Figure 8 outlines the exact steps involved in the YCL function. First, the original image is converted to YOLOv8, after which the character is cropped out and sent into a CRNN for character recognition and output. Showing the YOLOv8 cut out character from the original image is hence the first step. Table V displays all the YOLOv8 alphabet results. Figure 26 displays a sample of each alphabet, and Fig. 27 displays the original image after the YOLOv8 cropped.

Table V. Each alphabet results by YOLOv8.

Image class	Crop image (by YOLOv8)	Output result
A		A
B		B
C	undetected	none
D	undetected	none
E		E
F		F
G	undetected	none
H		H
I		I
J		J
K		K
L		L
M		M
N		N
O		O
P		P
Q		Q
R		R
S		S
T		T
U		U
V		V

W		W
X		X
Y		Y
Z		Z

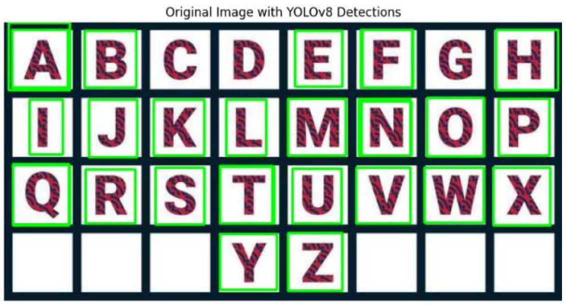


Fig. 27. Alphabet after YOLOv8 detection.

Table V demonstrates that following YOLOv8's detection of the original image, each character was cropped out by YOLOv8 separately and recognized by CRNN. According to the results, YCL has an overall 88.46% detection and recognition rate for the alphabet, missing only three words. Additionally, Table VI displays some actual scene images for YCL to identify and detect. Table VI illustrates how well YCL detects and recognizes the actual scene, apart from a small inaccuracy that prevents it from detecting characters.

As seen in Table VII, in addition to the alphabet, the YCL algorithm model can identify numbers 0 through 9. Figure 28 and Fig. 29 show the original image prior to and following the YCL algorithm model's cropping.

Table VI. Real scene alphabet results.

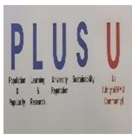
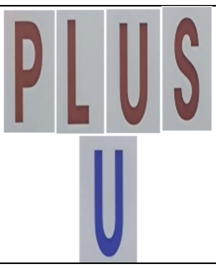
Image class	Crop image	Output result
		MM
		PLUS U

Fig. 28. Sample of each number.

Table VII. Each alphabet results by YOLOv8.

Image class	Crop image(by YOLOv8)	Output result
0		O
1		1
2		2
3		3
4		A
5		5
6		6
7		7
8		B
9		9

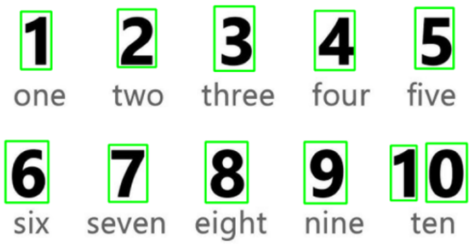


Fig. 29. Number after YOLOv8 detection.

Like Table V, Table VII displays all the number results that YCL detects and recognizes following a crop by YOLOv8. According to the statistics, YCL numbering recognition and detection rate is 70% overall, with a few numbers missed solely because they are almost identical to the alphabet, such as 4 into A and 8 into B. Additionally, Table V evaluates YCL on a genuine scene, but only for numbering. With a small error in the complex numbering combination, YCL produces the same result as Table VI as shown in Table VIII.

Table VIII. Real scenes number detection results.

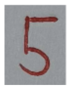
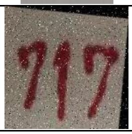
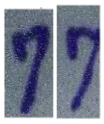
Image class	Crop image	Output result
		5
		77

Figure 30 shows the training speed for all algorithms across 30 epochs, except for LSTM, which only runs for 5 epochs. CNN takes 15 minutes to finish 30 epochs, which means 0.5 minutes per epoch. LSTM only runs for 5 epochs and already takes 75 minutes, which is 15 minutes per epoch. YOLO also runs for 30 epochs and takes 60 minutes, which is 2 minutes per epoch. YCL takes the longest with 180 minutes for 30 epochs, which is 6 minutes per epoch.

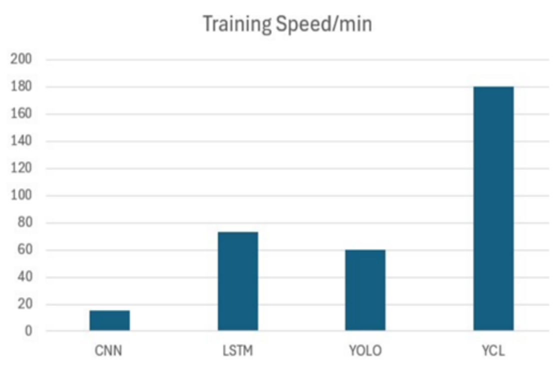


Fig. 30. Training speed for all compared algorithms.

The detection speed for each of the algorithms compared is displayed in Fig. 31. According to the performance comparison, LSTM is the fastest, detecting 19 pictures per second at 52 ms and 19 frames per second. CNN is operating at 16 frames per second and 61 milliseconds. At 510 ms and 2 frames per second, YOLO is the slowest. YCL is 231 ms and 4 frames per second.

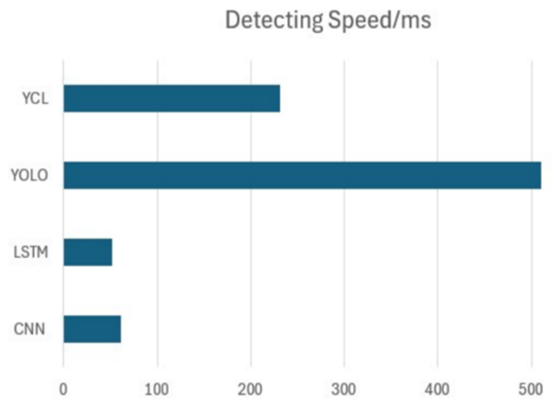


Fig. 31. Detection speed for all compared algorithms.

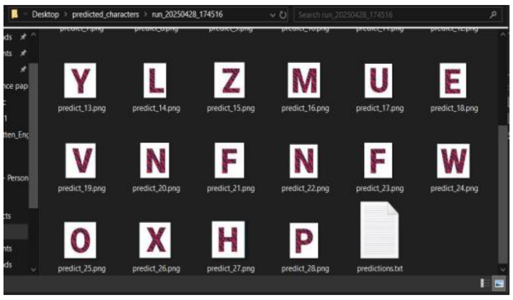


Fig. 32. Location for store the prediction image and txt file.

Following YCL algorithm on detection and recognition of the original image, the result will be displayed in the Visual Studio code. Additionally, the cropped image and recognition result will be saved into a txt file and an entire folder for the smart glasses to convert to Text-To-Speech (TTS), allowing the user to hear the character as shown in Fig. 32.

V. CONCLUSION

The invention of YCL-Based Smart Glasses provides an intuitive means for those with visual impairments to access textual information. Users can obtain information from 3 cm to 3 m with great accuracy owing to the tool's ability to magnify acquired photos utilizing a 6 mm lens and a high-resolution camera module. The user receives comprehensible and clear audio feedback from the Text-to-Speech (TTS) technology. With its ability to confirm text length and complexity, it provides a useful solution for daily use. This allows users to be minimally distracted while continuously receiving fresh textual data from their surroundings. The device's operating time is increased with the addition of a power-saving mode. According to the survey results of 32 participants, the Image-Based Word Recognition Tool has plenty of potential. 75% of them said they found it easy to use, and 72% said they would like to use it for everyday tasks. A lighter design, longer battery life, improved button feedback, and the use of autofocus were among the other ideas put forth by participants. To put it succinctly, the YCL character recognition algorithm has been developed and proven effectively to help people with visual impairments read characters from everyday scenes. The system performed well on a range of test photos by merging a CRNN-LSTM-based model for character identification with YOLOv8 for character detection. According to the experimental findings, the YCL system's overall recognition accuracy for alphabets was 88.46%, while its accuracy for numbers was approximately 70%. Even with complicated backdrops and varying handwriting styles, the system was able to detect and recognize most characters effectively. The only slight inaccuracies were for visually identical letters, like '4' being recognized as 'A' and '8' as 'B'. Additionally, the suggested two-stage pipeline greatly enhances detection and recognition when compared to YCL and conventional OCR techniques, particularly for difficult scenarios. Cropped images and word recognition are among the outcomes of the experiment that are automatically saved for later use, including converting them into word-to-speech (TTS) for smart glasses applications. For future extend, compact components will be utilized to lower the system's total weight while increasing battery capacity and operating duration. To improve clarity at different distances, autofocus features would also be included. Usability could be enhanced by improving button operation with more lucid tactile feedback. Optimizing algorithms for embedded devices may also lower power consumption and processing load. More real-world scene images could be added to the training dataset in the future, the model could be improved for faster processing on embedded devices, and the system could be able to recognize whole phrases or sentences for more comprehensive support.

ACKNOWLEDGEMENT

Authors would like to acknowledge the Center for Advanced Robotics and the Faculty of Engineering and Technology, Multimedia University, Malaysia for the support given throughout the study.

FUNDING STATEMENT

The authors received funding from Faculty of Engineering and Technology, Multimedia University for the research materials of this article.

AUTHOR CONTRIBUTIONS

Wai Kit Wong: Conceptualization, Experimental Design, Methodology, Investigation, Supervision, Implementation, Validation, Writing – Original Draft Preparation, Review & Editing, Manuscript Refinement,
Yao Wei Loh: Hardware Development and Implementation on Data Curation, Formal Analysis, Validation, Visualization, Testing, Data Analysis,
Wei Xiang Lee: Software Development and Implementation on Data Curation, Formal Analysis, Validation, Visualization, Testing, Data Analysis,
Thiruppathy Kesavan V*:* Conceptualization, Experimental Design, Methodology, Investigation, Validation, Writing – Original Draft Preparation, Review & Editing, Manuscript Refinement,
Thu Soe Min: Project Administration, Conceptualization, Supervision, Resources, Technical Guidance, Writing – Review & Editing,
and
Eng Kiong Wong: Project Administration, Conceptualization, Supervision, Resources, Technical Guidance, Writing – Review & Editing.

CONFLICT OF INTERESTS

No conflict of interests was disclosed.

ETHICS STATEMENTS

Our publication ethics follow The Committee of Publication Ethics (COPE) guideline. https://publicationethics.org/

REFERENCES

[1] M. J. Burton, J. Ramke, A. P. Marques, R. R. A. Bourne, N. Congdon and I. Jones, “The Lancet Global Health Commission on Global Eye Health: vision beyond 2020,” *The Lancet Glob. Health*, vol. 9, no. 4, pp. e489-e551, 2021.

[2] A. A. Diaz Toro, S. E. Campaña Bastidas and E. F. Caicedo Bravo, "Methodology to Build a Wearable System for Assisting Blind People in Purposeful Navigation," in *3rd Int. Congr. Informat. and Commun. Technol.*, San Jose, CA, USA, pp. 205-212, 2020.

[3] S. Nazim, S. Firdous, V. K. Shukla and S. R. Pillai, "Smart Glasses: A Visual Assistant for the Blind," in *2022 Int. Mobil. and Embed. Technol. Conf.*, Noida, India, pp. 621 -626, 2022.

[4] D. D. Brilli, E. Georgaras, S. Tsilivaki, N. Melanitis and K.Nikita, "AIris: AI-powered Wearable Assistive Device for the Visually Impaired," in *IEEE 10th RAS/EMBS Int. Conf. Biomed. Robot. and Biomechatron.*, Heidelberg, Germany, 2024, pp. 1236-1241, 2024.

[5] V. Moram, S. Zahruddin and S. Kumar, "Multifunctional Assistive Smart Glasses for Visually Impaired," *SN Comput. Sci.,* vol. 6, pp. 173, 2025.

[6] "Wearable AI System Helps Blind People Navigate," *Nature Publishing Group*, 2025. [Available Online on 14 April 2025] https://techxplore.com/news/2025-04-wearable-ai-people.html.

[7] Envision, “Envision Glasses: Smart Glasses for the Blind and Visually Impaired,” [Available Online] https://www.letsenvision.com/glasses/home.

[8] Noorcam, “NoorCam MyEye,” 2025. [Available Online: 15 June 2025] https://www.noorcam.com/en-ae/noorcam-myeye?srsltid=AfmBOorhamlLIvgsO1iu2CbC11YD8LWDbc2riEhTZ5IwTM8OX4r-kYpi.

[9] P. Ramya, R. Ramya and U. Ponvizhi, “Optical Character Recognition using Python,” in *Int. J. Trend in Sci. Res. and Develop.*, vol. 5, no. 3, pp. 1052-1054, 2021.

[10] S. Reshmi, R. D. Salagara nd S. S. Veni, “Text Detection in Image Based on The Morphology Method,” *Int. J. Eng. Develop. and Res.*, vol. 7, no. 3, pp: 468-471, 2019.

[11] P. Swaroop and N. Sharma. “An Overview of Various Template Matching Methodologies in Image Processing”, *Int. J. Comput. Appl.,* vol. 153, pp. 8-14, 2016.

[12] F. Ashraf and V. A. Nurjahan, “Connected Component Clustering Based Text Detection with Structure Based Partition and Grouping,” *J. Comput. Eng.*, vol. 16, no. 5, pp. 50–56, 2014.

[13] T. Kumar, C. Khosla and K. Vashistha, “Character Recognition Techniques using Machine Learning: A Comprehensive Study,” in *Int. Conf. Appl. Intellig. and Sustain. Comput.*, pp. 1-6, 2023.

[14] G. D. Jennifer and E. B. Carla “Feature Selection for Unsupervised Learning,” *J. Mach.. Learn. Res.,* vol. 5, pp. 845–889, 2004.

[15] J. Gui, T. Chen, J. Zhang, Q. Cao, Z. Sun, H. Luo and D. Tao, “A Survey on Self-Supervised Learning: Algorithms, Applications, and Future Trends,” *IEEE Trans. Patt. Analy. and Mach. Intellig.,* vol. 14, no. 8, pp. 1-23, 2015.

[16] X. Peng, Z. Huang, K. Chen, J. Guo and W. Qiu, "RLST: A Reinforcement Learning Approach to Scene Text Detection Refinement," in *2020 25th Int. Conf. Patt. Recogn.*, Milan, Italy, pp. 1521-1528, 2021.

[17] R. Kajale, S. Das and P. Medhekar, "Supervised Machine Learning in Intelligent Character Recognition of Handwritten and Printed Nameplate," in *2017 Int. Conf. Adv. Comput., Commun. and Contr.*, Mumbai, India, pp. 1-5, 2017.

[18] A. Rowlands, "Physics of Digital Photography," *IOP Publishing,* 2017.

[19] Raspberry Pi, [Available Online on 16 June 2025] https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/.

[20] J. Redmon, S. Divvala, R. Girshick and A. Farhadi “You Only Look Once: Unified, Real-time Object Detection,” in *2016 IEEE Conf. Comput. Vis. and Patt. Recogn.*, Las Vegas, NV, USA, pp. 779-788, 2016.

[21] O. Keiron and N. Ryan. “An Introduction to Convolutional Neural Networks”. *ArXiv*, abs/1511.08458, 2015.

[22] M. Domor, S. Theo and O. George. “Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications,” *Information*, vo. 15, no. 9, pp. 517, 2024.

[23] R. C. Staudemeyer and E. R. Morris, “Understanding LSTM -- A Tutorial into Long Short-Term Memory Recurrent Neural Networks”, *ArXiv*, abs/1909.09586, 2019.