VisionSync: A Blind Assisting Device

Dr. Shahnewaz Siddique
Assistant Professor
Department of Electrical and Computer Engineering
North South University

Ibrahim Mohammed Sofi Uddin
Department of Electrical and Computer Engineering
North South University

Suraiya Islam Meem
Department of Electrical and Computer Engineering
North South University

Rafia Tamanna
Department of Electrical and Computer Engineering
North South University

Mayisha Muntaha Alam
Department of Electrical and Computer Engineering
North South University

***Abstract*—This paper introduces a project aimed at improving navigation for individuals with visual impairments by integrating a Raspberry Pi and camera systems. The project focuses on two key functionalities: providing detailed audio descriptions of the environment through image processing and deep learning, and facilitating real-time object detection with distance estimation using a stereo camera setup. The system enhances the user's spatial awareness by converting image-based information into audio descriptions while also ensuring safety through timely alerts when objects approach within predefined ranges. This report delves into the technical details, highlighting the potential of this technology to foster inclusivity and empowerment for individuals with visual impairments.**

Keywords*— *Blind Assisting Device, Assisting Device, Computer Vision

I. Introduction

A. *Background and Motivation*

In a world where visual perception often dictates our understanding of the environment, navigating daily life can pose unique challenges for individuals with visual impairments. According to statistics given in [1] and Figure 1, an estimated 40 million individuals live with blindness, and around 200 to 300 million suffer from moderate to severe visual impairment. This demographic faces significant hurdles in performing everyday tasks that require spatial awareness and object recognition.

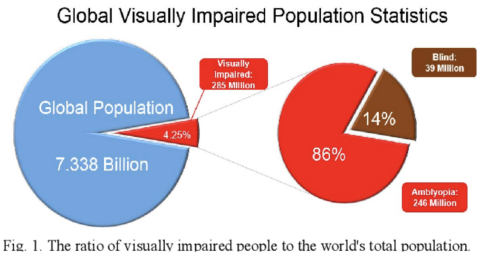

Fig. 1. The ratio of visually impaired people to the world's total population.

Fig. 1: Ratio of visually impaired people to the world's total population

Current assistive technologies for the visually impaired, such as white canes and guide dogs, offer some level of support but have limitations in terms of range and the type of information they can provide. Technological advancements have introduced various electronic aids, yet many of these solutions are either expensive or lack the comprehensive functionality needed to fully assist users in dynamic environments.

To address these challenges, this report unveils a project leveraging the capabilities of the Raspberry Pi along with components such as cameras and other sensors to empower the visually impaired in understanding and navigating their surroundings. Central to this initiative are two pivotal functions: environment description and object detection with distance estimation. By integrating affordable and accessible technology, this project aims to bridge the gap left by existing solutions.

The impact of this project holds the promise of significantly improving the daily lives of visually impaired individuals by enabling them to visualize their surroundings without the reliance on sight. This innovation has the potential to redefine independence, safety, and inclusivity for the visually impaired community, making a meaningful contribution to enhancing their quality of life.

B. *Purpose and Goal of the Project*

The purpose of this project is to develop a wearable device that empowers visually impaired individuals by providing them with auditory or haptic feedback about their environment. The project utilizes the Raspberry Pi, camera modules, and deep learning techniques to offer two main functionalities: environment description and object detection with distance estimation.

The first functionality, environment description, leverages a camera connected to the Raspberry Pi to capture images. These images are processed by a neural network, which generates detailed textual descriptions of the surroundings. These descriptions are then converted into an audio format, allowing users to hear vivid and informative descriptions of their environment without needing to ask others for help.

The second functionality employs a stereo camera setup consisting of two cameras capturing images from slightly different angles. This setup creates a 3D map of the scene by comparing the images, thus estimating distances to various objects. This feature is crucial for providing timely alerts to the user as objects approach a predefined range, with feedback delivered in either auditory or haptic form. This not only assists in navigation but also enhances the user's situational awareness, thereby contributing to their overall safety and independence.

The primary goals of this project are focused on building a functional and effective wearable device that achieves the following:

Build a Wearable Device to Enhance Independence: Develop a wearable system that provides real-time environmental descriptions and distance estimations, significantly reducing the need for external assistance and empowering users to navigate their surroundings independently. Implement Safety Features: Integrate timely alert mechanisms for nearby objects, helping users navigate safely and avoid potential collisions or accidents. This includes auditory and haptic feedback to ensure user safety.

Focus on Affordability and Accessibility: Utilize cost-effective components such as the Raspberry Pi and widely available camera modules to build an affordable solution. Ensure that the device can be easily adopted and maintained by the visually impaired community.

In short, this project aims to leverage advanced technology to construct a practical and impactful wearable tool. By building this device, we seek to improve the quality of life for visually impaired individuals, enabling them to independently understand and navigate their surroundings and fostering a greater sense of autonomy and confidence in their daily lives.

C. *Organization of the Report*

This report has been divided into eight chapters on the basis of various steps and results of the project. Section I discusses the introduction and goal of our project. Section II contains 'Literature Review', which reviews existing literature works that are relevant to our project. Section III explains the research 'Methodology' of the research work for the project. Section IV displays the 'Results' of our project, along with an analysis and a discussion section. Section V displays the planning and budgeting of the project. Finally, Section VI contains the 'Conclusion' of the report, along with limitations and future works.

II. Research Literature Review

In recent years, there have been major developments in the field of computer vision and IoT, enabling us to utilize their capabilities to solve real world problems. This project focuses on these two topics in order to come up with a solution to help people with visual impairment. By leveraging the power of computer vision and the connectivity afforded by IoT, we aim to create a portable wearable device that acts as an aid for the blind, offering enhanced environmental awareness. We take a look at different studies and papers relating to the use of IoT devices to assist the blind, as well as computer vision, more specifically, image caption generators and stereo vision.

The paper [2] puts more focus into hardware and implementation of IoT, and it introduces a solution, encompassing smart glasses, an intelligent walking stick, and a mobile app. This system detects obstacles and records fall-related information when users wear the glasses and use the walking stick. This paper outlines future plans, including the integration of deep learning for image recognition (e.g., traffic signs) and the development of intelligent walking-guidance features.

[3] presents a stereo vision system for calculating object distance in images. Through camera calibration and rectification, the system constructs a 3D scene, determining object distance by analyzing the bounding box's center. Results indicate accurate placements between 10 to 3 meters, with an average error of 2.08. The conclusion emphasizes successful implementation, proposing enhancements like minimizing frame matching flaws and incorporating advanced rectification algorithms. The system's potential extends to driving safety, robotics, and autonomous vehicles. Similar papers to [3] that utilizes stereo camera setup are [6] and [7]. [6] develops a system utilizing a stereo camera for detecting face to screen distance through obtaining a disparity map by the method based on stage geometry where pixels between two images are matched, and [7] uses supervised learning, specifically Linear Regression and Artificial Neural Network Regression. The results we get in [7] when CNN combined with densely connected neuron networks exhibits the lowest error rates, achieving remarkable distance estimation precision with minimal errors of 0.000531, 0.014490, and 0.000048 meters for mean square error, mean absolute error, and mean logarithmic error, respectively.

Paper [4] addresses the evolving needs of the automotive industry by proposing a novel solution for distance estimation using cameras instead of LIDARs. Leveraging the YOLO deep neural network and stereoscopy principles, the system utilizes two slightly moved cameras to capture images for stereoscopy-based measurement, estimating distances to detected objects, as seen in Figure 2. The conclusion emphasizes the presented solution's viability for distance estimation within 20 meters, with potential applications in adaptive cruise control and automatic parking. The paper suggests that, in certain scenarios, cameras could replace LIDARs, though challenges arise beyond 20 meters due to imprecise YOLO boundary boxes.

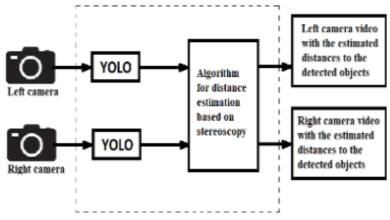

Fig. 2: Two cameras running YOLO model in real time

Paper [5] explores the feasibility of using the Raspberry Pi 2 as a cost-effective embedded computer vision solution for UAVs. By incorporating a multiplexer and two camera modules, the system executes a stereo matching pipeline, functioning as a depth meter for UAV applications. Experimental results reveal the configuration's capability for reasonably accurate depth estimation, particularly at a system speed of 1 m/s under favorable lighting conditions.

Papers [8] and [9] are both focused on image caption generation, where [9] is an improved version of the solution proposed in the paper [8].

In [8] "Show and Tell" presents a major contribution to the field of image captioning by introducing a neural network architecture that seamlessly combines a pre-trained convolutional neural network (CNN) for image feature extraction with a long short-term memory network (LSTM) for sequential caption generation. The model is trained in a supervised manner using pairs of images and corresponding captions. In [9] "Show, Attend and Tell," the paper represents a significant advancement in the field of image captioning. In response to the limitations of the "Show and Tell" model, this work introduces the attention mechanism to the image captioning process, which dynamically focuses on different parts of an image while generating captions. This attention mechanism enables the model to capture fine-grained details and improve contextual relevance.

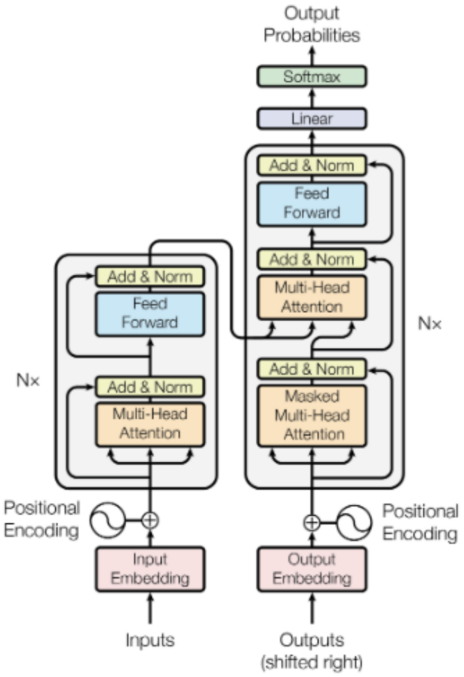

Fig. 3: The Transformer-model architecture

A further improvement can be implemented for the existing solutions in [8] and [9]. The use of transformer architecture[10][Figure 3] in place of recurrent neural networks for the NLP tasks. It relies on a self-attention mechanism that is built-in to process input sequences. Transformers are non-sequential, can process data in parallel, and they can capture long range dependencies more effectively than RNNs.

III. Methodology

A. *System Design*

This project can be divided into three main sections: hardware setup, image caption generation, and distance estimation using a stereo camera setup. Figures 4 and 5 illustrate the overall system architecture and the flow of operations within the device. Figure 4 provides a high-level overview of the workflow, from detecting objects and estimating the distance, to capturing images and providing feedback to the user. Figure 5 details the block diagram, showing how the hardware components connect to the Raspberry Pi to achieve the desired functionalities. This project is a combination of hardware, natural language processing, and computer vision.

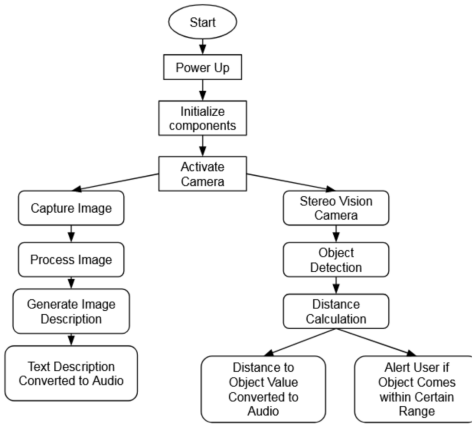

Fig. 4: Flow Diagram

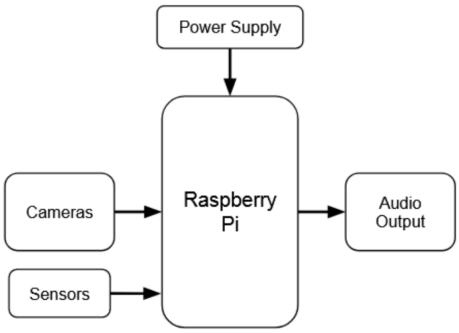

Fig. 5: Block Diagram

B. *Functionality and Components*

*1) **Hardware**:*

Our hardware will consist mainly of a raspberry pi with a dual camera setup, speaker, and buttons. Raspberry Pi, is a small scale single board computer. Given the demanding tasks that involve deep learning models and real-time computations from video streams, we will use the latest model, Raspberry Pi

5 [Figure 6]. The Raspberry Pi 5 does not include a headphone jack, so we can either use a Bluetooth speaker or connect a speaker module via the GPIO pins on a breadboard.

A significant feature of the Raspberry Pi 5 is its dual Camera Serial Interface (CSI) ports, which is crucial for our stereo camera setup that requires two camera modules. This dual CSI capability allows for the simultaneous connection and synchronization of two Raspberry Pi cameras, enabling effective stereo vision for depth estimation.

The Raspberry Pi camera we use is the V1 model. To connect this camera to the Raspberry Pi 5, we will need a camera ribbon adapter. The Raspberry Pi 5 uses the smaller 22-pin CSI (Camera Serial Interface) connectors, whereas older Raspberry Pi camera modules, such as the Camera Module V1 and V2, typically use the larger 15-pin connectors. This adapter converts the 15-pin connector on the camera module to the 22-pin connector on the Raspberry Pi 5, ensuring compatibility and proper connection between the devices.[Figure 7]

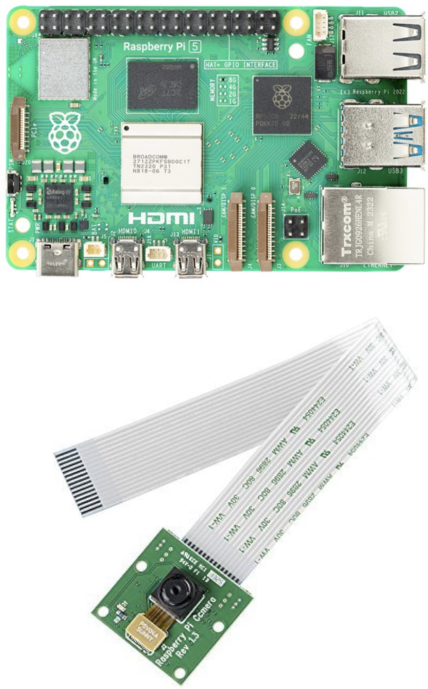

Fig. 6: Raspberry Pi 5 and Camera Module

Fig. 7: 15-pin to 22-pin camera adapter

To capture images and switch between different functions, we will implement a button module using the GPIO pins on the Raspberry Pi. These buttons will be connected to a breadboard, allowing us to interface with the Raspberry Pi for various control operations.

Due to the limited processing power of the Raspberry Pi, we will initially develop and test our code on a more powerful separate device. This approach allows for faster development and testing cycles without being constrained by the Raspberry Pi's performance limitations. Once we have successfully achieved our objectives on the separate device, we will transfer the code to the Raspberry Pi. At this stage, we will make the necessary adjustments to optimize the code and use compatible libraries or software alternatives for the Raspberry Pi, ensuring it runs efficiently within the device's capabilities.

*2) **Image Caption Generator**:*

The Pi camera module uses the image capture process on our Raspberry Pi 5 [Figure 6]. The Raspberry Pi Camera for the Raspberry Pi 5 is a high-resolution camera module designed for capturing still images and video. It connects to the Raspberry Pi via a dedicated CSI (Camera Serial Interface) port. The camera supports various modes and resolutions, offering flexibility for different applications such as photography, video recording, and computer vision projects. Enhanced features may include improved low-light performance, higher frame rates, and advanced image processing capabilities compared to earlier models.

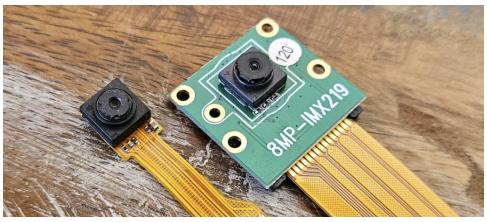

Fig. 8: Raspberry Pi Camera Module

Additionally, we grasp the Gradio Client API to interact seamlessly with a Gradio interface. Gradio itself is a Python library used for building interactive user interfaces for machine learning models, data science workflows, and other computational applications. By connecting to a Gradio app using its URL, we can send inputs and receive outputs from the app. This is useful for automating tasks and integrating Gradio interfaces into larger systems. The client supports various input and output types, authentication, and batch predictions, making it versatile for different applications. The Client class from the gradio_client library plays a vital role in handling communication with the interface and managing tasks.

After the Gradio Client generates captions, we use the pyttsx3 library for text-to-speech conversion. It allows you to convert text to spoken words using the speech synthesis capabilities available on your operating system. Unlike some other text-to-speech libraries, pyttsx3 works offline and does not require an internet connection. It supports multiple voices,

speech rate adjustment, and volume control, making it a versatile tool for various text-to-speech applications.

This text-to-speech conversion allows the captions to be converted into audio format, which can then be played through a Bluetooth speaker [Figure 8]. This integration improves accessibility and usability, providing auditory feedback to users.

Fig. 9: Bluetooth Speaker

*3) **Stereo Vision**:*

Estimating distance with a single camera presents challenges due to the lack of depth information in a 2D image projection of the 3D world. This limitation makes precise distance estimation difficult from a single image alone.

To address this, a stereo camera setup includes a second camera, enhancing computational demands and costs compared to single-camera solutions like monocular depth estimation. However, it significantly improves accuracy. Stereo cameras capture images from slightly different angles, enabling the system to compare these views and create a 3D map of the scene. This method provides reliable distance measurements crucial for the safety and independence of visually impaired individuals, aiding in navigating environments by avoiding obstacles and hazards.

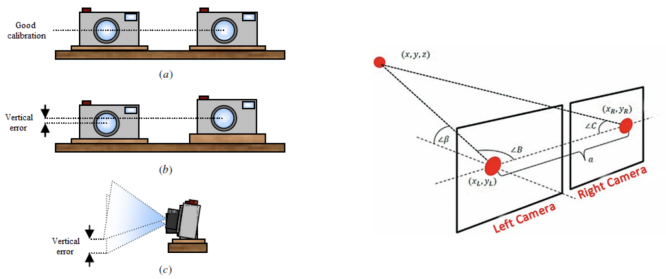

Fig. 10: Illustration of a stereo camera setup and the geometric principles behind distance estimation

Figure 10 shows us the proper way to set our camera up for creating a stereoscopic vision system. The second Figure on the right is all about the camera's geometry. By understanding these principles and identifying the necessary variables, we can derive values that help us estimate distance accurately.

The stereo vision system is implemented in Python, making extensive use of OpenCV. OpenCV manages critical tasks in stereo vision, including camera calibration, image rectification, disparity map computation, and object detection. Supporting libraries such as NumPy and Matplotlib handle data manipulation and visualization. Additionally, the Pyttsx3 library is used for text-to-speech conversion, delivering auditory feedback to users based on processed results.

C. Hardware and Software Implementation

The project integrated both the hardware components and software solutions for setting up cameras with careful calibration using a chessboard pattern to ensure accurate depth perception through stereo vision. Object detection algorithms like YOLOv8n and SSD MobileNet V3 are evaluated for real-time performance, with SSD MobileNet V3 chosen for its balance of speed and accuracy. The system calculates distances to detected objects using stereo images and generates captions for images using the Pi camera module and Gradio Client API, with captions converted to audio for accessibility.

The wearable assistive device includes a Raspberry Pi 5 positioned centrally on a plastic surface, with a breadboard placed above it. Two camera modules are mounted at chest level for accurate stereo vision. It is designed to be worn around the neck for user convenience.

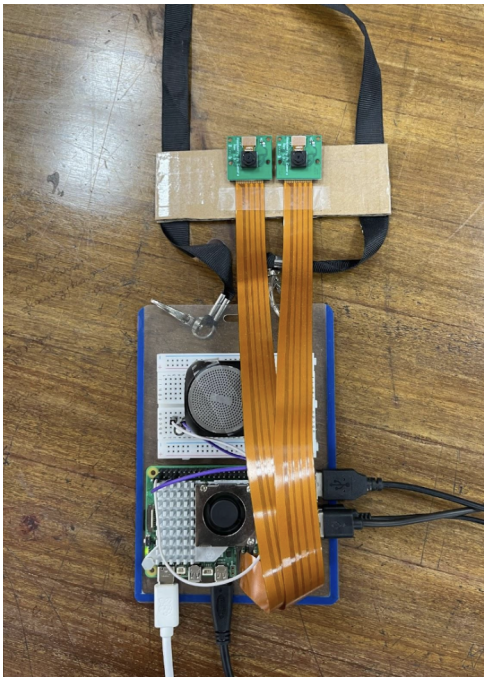

Fig. 11: Assembled Wearable Device

*1) **Raspberry Pi**:*

Raspberry Pi 5 is much more powerful than its predecessor, the Raspberry Pi 4. It has a faster 64-bit quad-core processor running at 2.4GHz, making it 2-3 times faster in terms of CPU performance. It also has a better graphics processor and can support two 4K displays at 60 frames per second through HDMI. Additionally, it has an improved image processor for better camera support.

The RP1 "southbridge" chip handles most of the input/output (I/O) functions, greatly improving the performance and capabilities of connected devices. USB transfer speeds are more than doubled, allowing faster data transfer to external drives and other peripherals. The camera and display interfaces are upgraded to support higher data rates and allow up to two cameras or displays. The SD card performance is also doubled, and a new PCI Express 2.0 interface is included for high-speed peripherals.

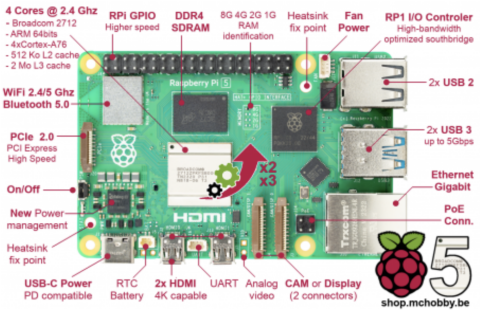

Fig. 12: Raspberry Pi 5

Setting up the hardware for the Raspberry Pi 5 involved a methodical approach to ensure everything was installed and configured correctly. Initially, we downloaded the latest version of the Raspberry Pi OS from the official Raspberry Pi website. We had installed:

- Raspberry Pi OS with desktop
- Release date: March 15th, 2024
- System: 64-bit
- Kernel version: 6.6
- Debian version: 12 (bookworm)
- Size: 1,105 MB

This step was crucial as it ensured we had the most up-to-date operating system with all necessary features and improvements. Verifying the OS image's integrity through checksum verification was also part of this initial phase, ensuring the downloaded file was not corrupted and could be safely written to the MicroSD card.

Next, using a reliable image writing tool, Raspberry Pi Imager, we wrote the OS image onto a high-quality MicroSD card. This process involved selecting the downloaded OS image file and the specific MicroSD card as the target, followed by initiating the write operation. Ensuring the MicroSD card had sufficient capacity and speed was essential for optimal performance of the Raspberry Pi 5.

Once the OS image was successfully written, we inserted the MicroSD card into its designated slot on the Raspberry Pi 5. To provide visual output, we connected an external monitor to the Raspberry Pi 5 using HDMI to USB connectors and HDMI to HDMI cables. This setup ensured a stable video signal transfer, crucial for monitoring the boot process and interacting with the Raspberry Pi OS.

Fig. 13: Raspberry Pi 5 connected to an external monitor

For user input, we connected a standard mouse and keyboard to the Raspberry Pi 5 via USB ports. These peripherals allowed us to interact with the operating system, navigate menus, and execute commands effectively during the setup process and subsequent use.

To power the Raspberry Pi 5, we utilized a USB-C power adapter, ensuring it met the recommended specifications for voltage and current. Activating the power supply initiated the boot sequence of the Raspberry Pi 5, and we monitored the startup process on the connected monitor to confirm proper operation of the hardware and software.

Upon booting into the default Raspberry Pi OS environment, we proceeded to install additional software necessary for our project. Notably, we installed Visual Studio Code (VS Code), a versatile code editor widely used for its rich features and compatibility with various programming languages. Installing VS Code on the Raspberry Pi OS involved downloading the appropriate version from the official website and following straightforward installation instructions provided for the ARM architecture.

With VS Code installed, we were able to open our project files, write code, debug applications, and run scripts directly on the Raspberry Pi 5. This setup provided a stable and efficient environment for developing and testing our applications, leveraging the capabilities of the Raspberry Pi 5 for computing tasks and projects.

*2) **Image Caption Generator**:*

We have used the Pi camera module on a Raspberry Pi 5 to capture images and process them for image captioning. After being captured, the images are stored as a temporary file in the project directory. Once the image is captured, the path is passed to a file named captionOnImages.py to complete the captioning.

The Gradio Client API is crucial for interacting with a Gradio interface, allowing the input image to be captioned. For seamless communication with the interface, the gradio_client library enables the Client class to send the image and receive a caption in return. The Gradio Client handles the images as a list of tuples, each with an image and a corresponding caption.

The pyttsx3 library is used for text-to-speech conversion after the Gradio Client generates the caption for the image. It converts the generated caption into audio, allowing the caption to be played via a Bluetooth speaker for the user. This feature significantly enhances the system's accessibility and usability, especially for users who may benefit from auditory feedback.

The combination of components and libraries results in a comprehensive system capable of capturing, captioning, and audibly showing images.

3) ***Stereo Camera Implementation****:*

Getting started with a stereo camera for depth estimation involves several lengthy steps to set everything up. Below, we will dive into a detailed explanation of each step:

a) ***Camera Setup****:*

In the stereo camera setup, both cameras are positioned at a fixed distance from each other, ensuring they maintain the same level and face in directions that are parallel to each other. This setup is crucial for accurate depth perception because it allows the system to capture images from slightly different viewpoints, mimicking human binocular vision.

The fixed distance between the cameras, known as the baseline distance, plays a critical role in calculating depth information through stereo vision. This distance determines the parallax effect observed between the images captured by each camera, which is essential for computing the disparity and subsequently estimating distances to objects in the scene.

To ensure precise depth estimation, it's important to securely mount both cameras at the designated baseline distance and verify that they are aligned horizontally and facing forward with minimal angular deviation. This configuration optimizes the stereo matching process and enhances the system's ability to generate accurate 3D reconstructions of the environment. Figure 9 is a good example of how we should set up our camera.

b) ***Calibration****:*

Calibration is a critical step in the setup process, performed after capturing images from both stereo cameras. We use a chessboard pattern for calibration, which provides a structured grid of known dimensions to detect key points in the images. OpenCV's built-in function helps detect corners in the chessboard pattern. Before calibration, the dimensions of the chessboard, including the number of inner corners and the size of the squares in millimeters, must be specified. There are different types of patterns that can be used in the calibration process [Figure 14].

By capturing images of the chessboard from various angles, the system determines the intrinsic parameters (such as focal length, principal point, and distortion coefficients) and extrinsic parameters (rotation and translation relative to a reference frame) for each camera. These parameters are essential for correcting image distortions and mapping image points to real-world coordinates.

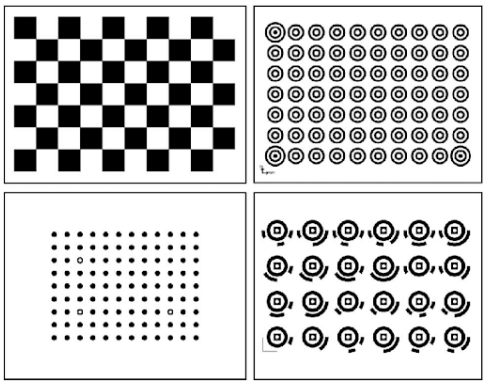

Fig. 14: Different patterns for calibration

After individual camera calibration, stereo calibration is performed. This process uses the calibrated images from both cameras to compute the essential and fundamental matrices. These matrices describe the geometric relationship between the cameras and facilitate the rectification and undistortion of stereo images.

Executing the calibration code generates an XML file named "stereoMap.xml," which contains the calibration parameters for both cameras. This file is crucial for subsequent stages, providing the necessary data for image rectification, disparity calculation, and depth estimation.

We must ensure that the cameras do not move or change orientation from this point onwards, as any slight deviation will require another calibration to be performed again. To overcome this issue, it is advised to mount the camera on a secure platform, particularly one that is able to accommodate both cameras comfortably, such as a flat wooden stick, or a cardboard cutout etc.

c) ***Undistort and Rectify the Images****:*

Before estimating depth in a stereo vision system, it is crucial to undistort and rectify the images captured by the stereo cameras. Undistortion corrects lens distortions, ensuring the images accurately represent the scene[Figure 15]. Rectification then aligns the images so that corresponding points appear on the same scanline, simplifying disparity calculations[Figure16]. The XML file generated during calibration contains the essential matrices for these processes. By loading this file, the camera parameters and distortion coefficients are applied, producing undistorted and rectified images suitable for accurate depth map generation.

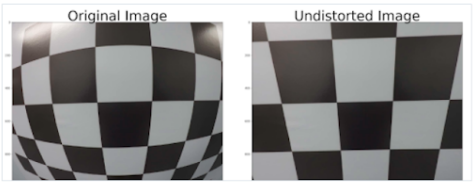

Fig. 15: Distorted and undistorted images

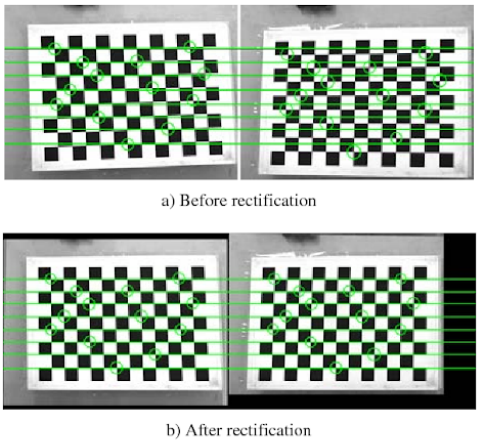

Fig. 16: Before and after rectification

Figure 15 shows us how the camera distorts the actual view from real life onto an image, we must use the calibration file to fix this and undistort the image so that it appears more realistic with proper proportions. In Figure 16, the rectification process aligns corresponding points from both captured images along the same axis, resulting in slightly modified images with black lines or bars at the edges. These transformations align images from different viewpoints, leaving areas outside the mapped regions filled with black to preserve image integrity.

*d) **Generate a depth map through block matching and epipolar geometry**:*

In our stereo vision system, we employ a process called block matching to find similar points or blocks of pixels between our two images. This is a fundamental step in depth estimation, as it allows us to establish correspondences between the rectified images captured by the stereo cameras.

Epipolar geometry underpins this process, describing the geometric relationship between corresponding points in the two images. By leveraging epipolar lines, which are predetermined lines that facilitate the search for corresponding points in the other image, we can make the block matching process more efficient and computationally feasible.

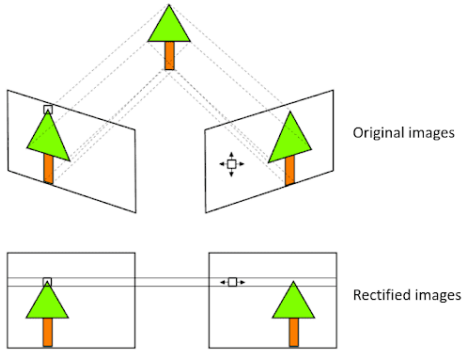

Fig. 17: Block matching between left and right images, with epipolar line

Figure 17 demonstrates an example of how block matching is performed, more specifically the StereoBM algorithm. Figure 18 shows us the epipolar line that crosses the two images marking down similar points between them, and using this line it is possible to calculate disparity and obtain a depth.

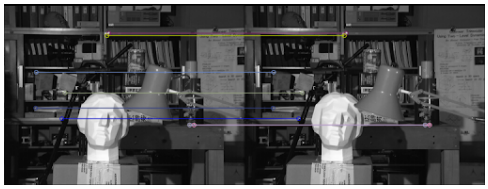

Fig. 18: Epipolar line showing similar points from both images

In our implementation, we use OpenCV's 'StereoSGBM_create' function to generate the depth map, which is crucial for accurate depth estimation. There are two main algorithms available in OpenCV for this purpose: Stereo Block Matching (StereoBM) and Stereo Semi-Global Block Matching (StereoSGBM). While StereoBM is simpler and faster, it often sacrifices accuracy. On the other hand, StereoSGBM, which we use, provides more accurate and reliable results at the cost of increased computational complexity. StereoSGBM optimizes the matching process by considering multiple paths along the image to find the best matching block, while StereoBM looks for the best matching block along a single axis and path.

The depth map, created by comparing corresponding points in the rectified images, enables us to determine the distance of objects in the scene. The disparity values of these corresponding points are inversely proportional to the depth, allowing us to discern whether an object is near or far. For distant objects, there's low disparity between corresponding points; for close objects, the disparity is higher.

*e) **Apply an object detection algorithm**:*

We tested two different object detection algorithms, YOLOv8n and SSD MobileNet V3, to evaluate their performance and suitability for our stereo vision system. These algorithms are pre-trained.

YOLOv8n (You Only Look Once, version 8, nano): YOLO is a well-known object detection algorithm that prioritizes speed and efficiency without significantly compromising accuracy. The YOLOv8n variant is a lightweight, high-speed model designed to run efficiently even on devices with limited computational resources. YOLO operates by dividing the image into a grid and predicting bounding boxes and class probabilities for each grid cell simultaneously. This single-stage approach allows YOLOv8n to achieve real-time performance, making it an excellent choice for applications requiring fast detection and minimal latency.

SSD MobileNet V3 (Single Shot MultiBox Detector with MobileNet V3 backbone): SSD MobileNet V3 is another popular object detection framework that balances speed and accuracy. The SSD algorithm employs a single-shot approach, detecting objects in images with just one forward pass through the network. It achieves this by generating a fixed set of bounding boxes and scores for object categories, followed by a non-maximum suppression step to refine the final detections.

The MobileNet V3 backbone further enhances this process by providing a highly efficient and lightweight convolutional neural network designed specifically for mobile and embedded vision applications. This combination ensures that SSD MobileNet V3 can deliver fast and accurate detections with minimal computational overhead.

Both YOLOv8n and SSD MobileNet V3 are highly customizable, allowing us to fine-tune the algorithms to detect specific objects and ignore others.

We ultimately chose SSD MobileNet V3, sacrificing a bit of accuracy to achieve better frames per second during the detection process. From our testing results, we noticed slight lag and delay when using the YOLO model, but almost no lag and delay with the MobileNet model. It is important that the object detection algorithm runs in real time and provides instant information. The real-time performance of SSD MobileNet V3 ensures that users receive timely and accurate feedback, which is crucial for applications that rely on immediate object detection, such as navigation aids for visually impaired individuals. This balance between speed and accuracy makes SSD MobileNet V3 the optimal choice for our stereo vision system.

*f) **Estimate Distance**:*

After detecting objects using our object detection algorithms, we have to find a way to get the distance from the object to the camera. This process involves matching the detected objects in the images from both cameras and using the disparity between corresponding points to calculate the distance.

Both cameras run the object detection algorithm simultaneously. Then, we write a function to match similar detected objects between both images. Once the objects are matched, we obtain the pixel coordinates of the bounding box's center. Using these coordinates, we can calculate the disparity of similar points between both images with the formula: Disparity d = Xleft - Xright.

Here, Xleft is the pixel count from the center axis of the image to the center of the left image's bounding box, and Xright is the corresponding measurement for the right image.

Finally to get the distance Z, we use the formula "Z = (f . B)/d "[Figure 19], where f is the focal length of the cameras, B is the baseline(distance between both cameras), and d is the disparity of the detected object. This formula highlights that for distant objects, the disparity d is small, resulting in a larger value of Z, indicating a greater distance. Conversely, for closer objects, the disparity is larger, leading to a smaller Z, indicating a shorter distance.

$$Depth\ (mm) = \frac{focal\ length(pixels)\, x\, Baseline(mm)}{Disparity\ (pixels)}$$

$$where\ focal\ length(pixels) = \frac{1}{2}\frac{Xres(pixels)}{tan(\frac{HFOV}{2})}$$

Fig. 19: Disparity formula

In the next section we will look at the results of implementing all these steps, from the setup stage till object detection and distance estimation.

IV. Result, Analysis and Discussion

The results of the Stereo Vision object detection, depth estimation, and image caption generation are discussed in detail in this chapter along with visual depictions of the outputs.

A. *Stereo Vision Results*

Setup is displayed on Figure 20. We used two identical webcam connected to a powerful computer during building and testing of the program in-order to achieve object detection and depth estimation. After completion and testing of the program we move the code onto the raspberry pi which has two raspberry pi V1 camera module connected to it, and we modify some of our initial parameters used with the webcam to make the code compatible with the raspberry pi camera modules.

Fig. 20: Stereo Camera Setup with webcam

Then comes the calibration part. We used a chessboard pattern of dimension 13X9 [Figure 21]. The openCV library generates the lines that can be observed in Figure 22. These lines, known as epipolar lines, are crucial for stereo vision. They help establish correspondences between points in the images captured by the stereo cameras.

Fig. 21: Capturing image of chessboard pattern

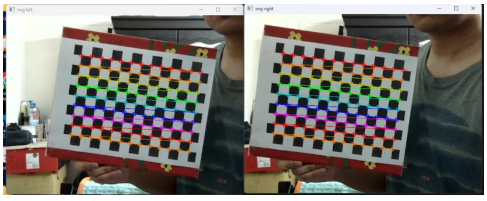

Fig. 22: Calibration process of the chessboard pattern

After calibration, we get an xml file containing the calibration parameters of both cameras. Applying this file on a video capture of the webcam will give us a rectified and undistorted

image. Figure 23 shows us the resulting image formed after calibration.

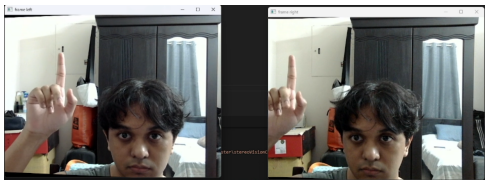

Fig. 23: Rectified and Undistorted Image

This rectified image [Figure 23] has some black lines at the top and bottom of the image. This is due to the alignment process during rectification, where the images are transformed to be on the same plane. The transformation can cause some areas to be uncovered, resulting in black borders at the edges.

The epipolar lines we see in Figure 24 result from applying the block matching algorithm to the images. These lines help us identify corresponding points between the left and right images.

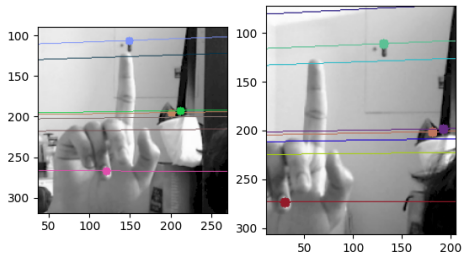

Fig. 24: Epipolar lines showing corresponding points between left and right images

After applying the block matching algorithm, we can generate a depth map using the disparity values obtained from all corresponding similar points in Figure 25. The formation of the depth map is optional, but it is a good indicator that the stereo vision system is working correctly and provides a visual representation of the depth information.

The depth map displayed in Figure 25 illustrates that objects closer to the camera appear brighter in color, whereas objects farther away appear dimmer and can sometimes be hard to spot by just looking at the depth map image.

The final step is to add an object detection algorithm, and use the bounding box to obtain and present information. The algorithm we used during building of the program in Figure 26 is YOLOV8n. The bounding box generated with YOLOV8 seems more stable, accurate and has high confidence, on the other hand the MobileNet bounding box, seems jittery and has lower confidence score compared to YOLO, but the frame rate and performance of the MobileNet is smoother and can give results faster.

We obtain the bounding box center's coordinates by calculating the midpoint of the bounding box. This is done by averaging the minimum and maximum x values to find the x center coordinate, and similarly averaging the minimum and maximum y values to find the y center coordinate. In our

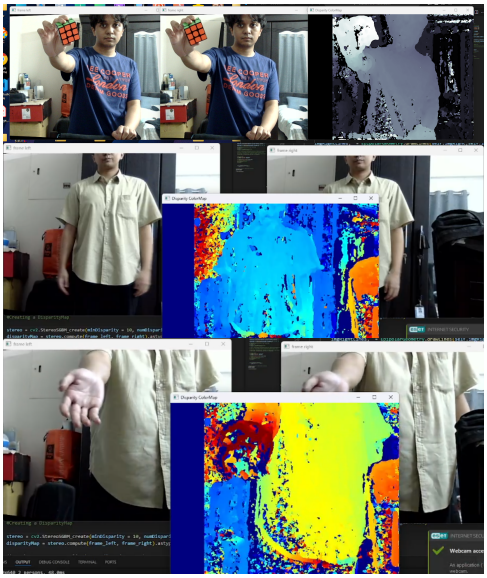

Fig. 25: Depth/Disparity map produced after block matching is successful

disparity calculation, we only use the x coordinates of each bounding box center. Then, we calculate the disparity between x coordinates obtained from the left image bounding box and the right image bounding box using the formula: Disparity d = Xleft - Xright

The disparity is measured in pixels. Then to get depth of the object we use this formula : Z = (f . B)/d

Here, f represents the focal length obtained from the camera manufacturer's specifications, B denotes the baseline distance between both cameras measured with a ruler or measuring tape, and d signifies the disparity in pixels. Figure 26 shows the estimated distance displayed on the top of the bounding box.

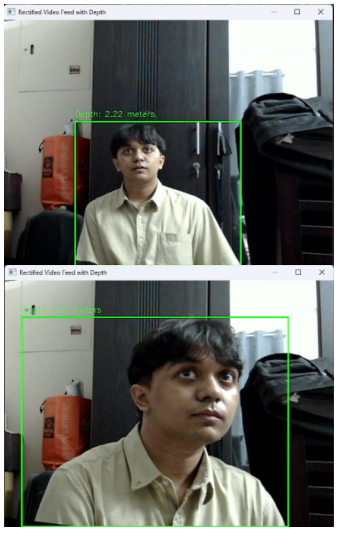

Fig. 26: Distance estimation and object detection

The depth value we get seems to deviate by +/-0.3 meters.

To get a better result we need to optimize our calibration process and use a different chess board dimension.

In addition to object detection, we are incorporating a centerline feature. This feature detects the center of each bounding box and determines whether it lies to the left or right of the centerline. A text-to-speech feature will issue a warning indicating the direction from which an object approaches the camera, triggered only when the object is within 1.5 meters of the camera. Figure 27 shows us the implementation of a centerline in the final image.

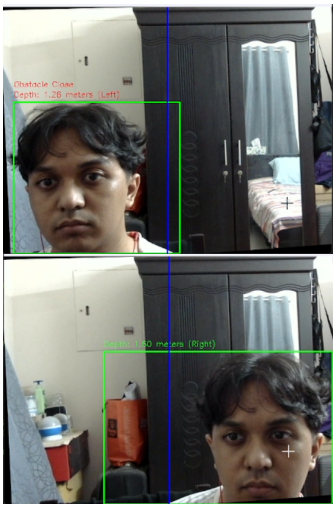

Fig. 27: Distance estimation and object detection with a center line indicating the side where the object lies on

From Figure 27, we can see that the left and right detection feature works, and when an object is less than 1.5 meters away from the camera the bounding box color changes to red indicating a warning.

Switching the object detection algorithm to MobileNet for testing was straightforward, and the steps to accomplish object detection with depth estimation are identical. Results may slightly vary, due to accuracy and confidence score by both models.

We can now deploy the code onto the raspberry pi. We need to update the libraries to use compatible libraries that will work with a raspberry pi camera, such as the use of PiCamera in place of openCV for capturing images. Parameters such as focal length will need to be updated together with a new baseline depending on how we choose to set the raspberry pi cameras up.

B. Image Captioning Results

As seen in the figure below, the Gradio client API image caption generator ran perfectly and generated accurate captions for the input images. There are also accurate captions generated from the images captured by the Raspberry Pi camera module. Based on the accuracy of the captions, it can be seen that the image captioning model was executed flawlessly.

After capturing an image with the Pi camera, it is temporarily stored in the image folder. The path of this image is then passed to the ‘captionOnImages.py‘ file to initiate the captioning process. To facilitate interaction with the Gradio interface within ‘captionOnImages.py‘, the ‘Client‘ class from the ‘gradio_client‘ library is imported.

For image captioning, we utilize the Gradio Client API, which handles our input images. Gradio Client sends these images as tuples of (image, caption) or (image, None) if no captions are provided. It manages communication with the interface, retrieves results, and extracts relevant information. Once the caption is generated, we employ the ‘pyttsx3‘ Python library for text-to-speech conversion. This converts the caption into audio, facilitating playback through a Bluetooth speaker for user accessibility.

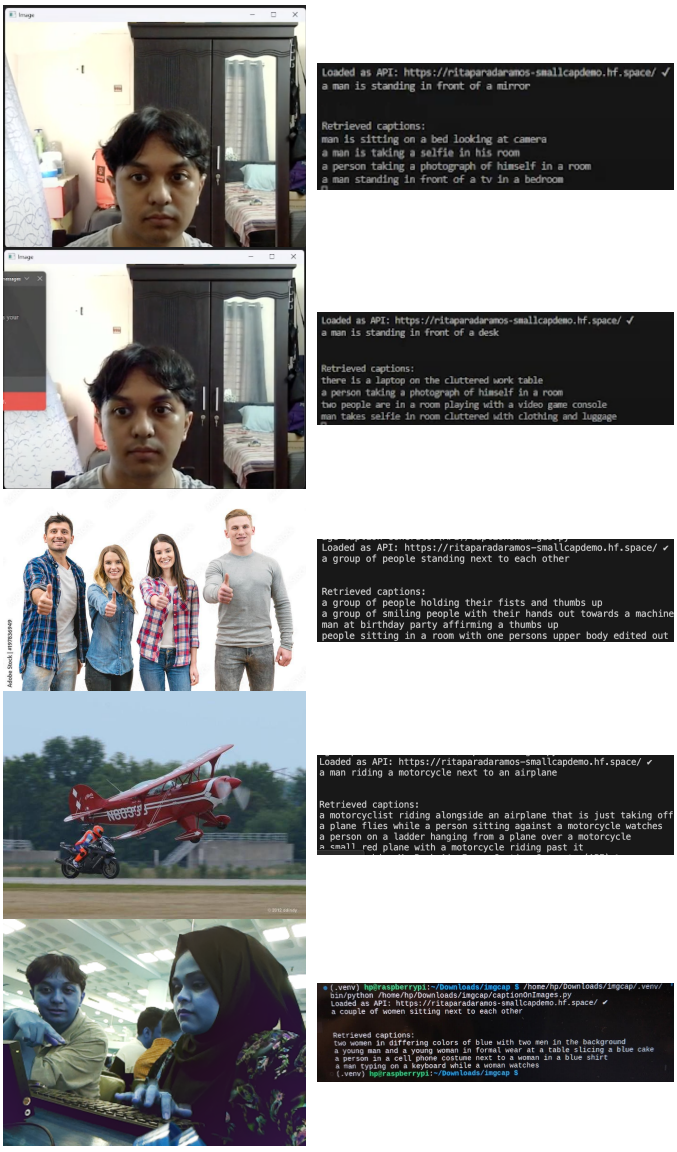

Fig. 28: Execution of Image Captioning Model. Images on the left and its generated caption on the right

C. Final Setup Results

Below is the figure showing the whole implemented device.

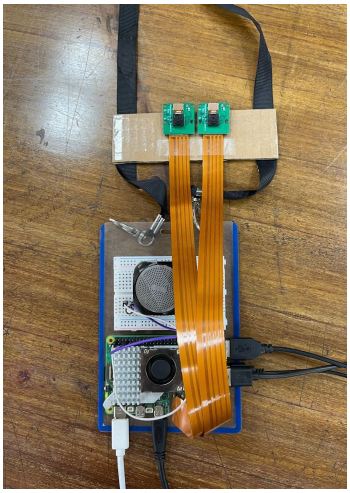

Fig. 29: Implemented Device

After the wearable device is worn, the Pi camera is then initiated. Utilizing stereovision, which involves two camera modules placed at chest level for accurate depth perception, the device processes the images using MobileNet to calculate distances and enhance visual understanding. From the figure below, we can see that the distance changed as the person came closer.

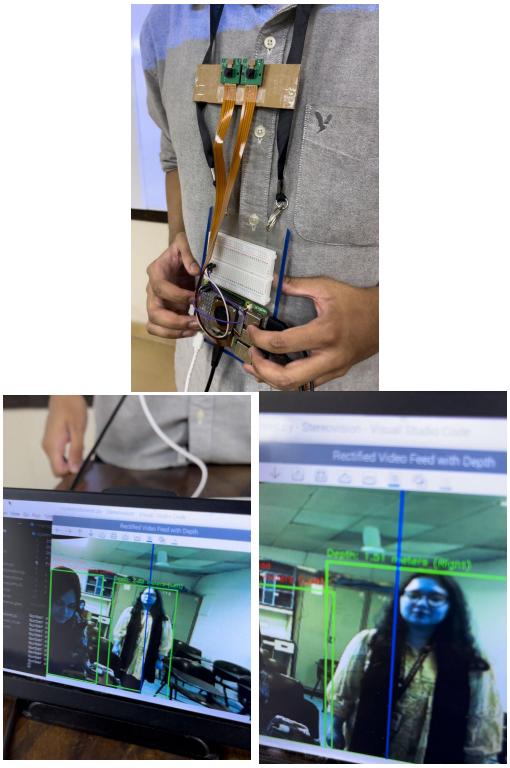

Fig. 30: Stereovision on implemented device performing object detection and depth estimation

The Pi camera captured the image, which was then processed by the image captioning model. The path of the image is sent where the Gradio Client API generates a descriptive caption. This caption is then converted to speech using the pyttsx3 text-to-speech library, enabling the user to hear the description through a Bluetooth speaker.

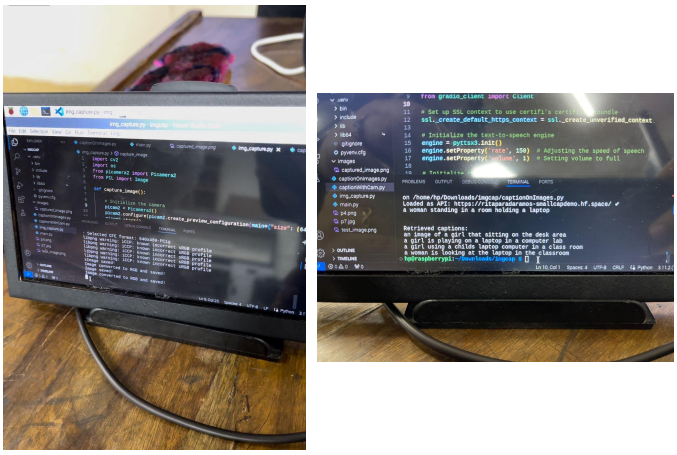

Fig. 31: Image Captioning on Implemented Device

V. Project Planning and Budget

The planning and timeline of the project is displayed below through a Gantt chart. The project was carried out over the duration of two consecutive academic semesters.

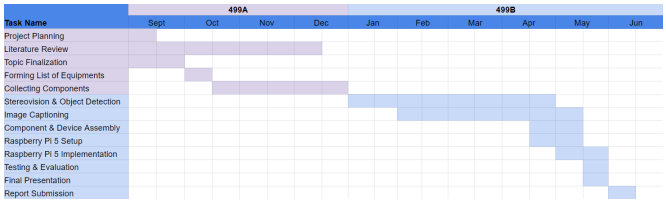

Fig. 32: Gantt chart of our project

The device was built using the bare minimum amount of components. We promoted eco-friendliness by upcycling used material to create the base of the device.

Component	Unit Price (in BDT)	Quantity	Total Cost (in BDT)
Raspberry Pi 5 8GB Complete Set	22,355/-	1	22,355/-
Raspberry Pi Camera Module	880/-	2	1,760/-
Mini Speaker	225/-		225/-
Adapter	290/-		580/-
Push Button	5/-		5/-
Breadboard	80/-		80/-
Subtotal			25,005/-

Fig. 33: The budget table for our project

VI. Conclusions

A. *Summary*

The project is a wearable device that assists visually impaired individuals by providing real-time environmental descriptions and detecting the distance of objects surrounding the user. Using the Raspberry Pi 5 and camera modules, the device captures images, processes them, and provides auditory feedback. This significantly improves user independence and

situational awareness. The environmental description functionality enables users to receive detailed audio descriptions of their surroundings through a Bluetooth speaker that processes the captured images, reducing dependency on external assistance and empowering users to understand their environment autonomously.

Additionally, a stereo camera setup, which provides timely alerts about nearby objects, allows users to detect objects and estimate distances. This system utilizes auditory feedback to warn users of possible obstacles, improving navigation safeness and reducing the risk of collisions or accidents. The project aims to enhance the quality of life for visually impaired individuals, allowing them to navigate their surroundings more confidently and independently.

B. Limitations

While working on the project, we discovered several limitations. The Raspberry Pi 5 does not support Google generative AI to certain extent, deterring the use of cloud-based packages for image caption generation. While it boasts increased RAM options beneficial for larger datasets, its modest memory and storage capacities compared to desktops or cloud solutions can limit the size and complexity of AI deployments.

Additionally, some Python packages are incompatible with the Raspberry Pi 5, impacting overall implementation. We encountered challenges with "torchvision" and "transformers" on Raspberry Pi 5, primarily due to its ARM-based Broadcom BCM2711B0 processor. These Python packages are optimized for x86_64 architectures typical in desktops and servers, requiring extra steps for installation on ARM. Methods include building from source, using "conda" environments, or relying on ARM-compatible binaries. Optimization for ARM is essential for achieving optimal performance on the Raspberry Pi 5.

The device also has practical limitations, such as requiring high-speed WiFi for optimal functionality and being relatively heavy, which inhibits portability and ease of deployment. Furthermore, the system sometimes fails to generate accurate captions, exhibiting a need for further improvement in the captioning process.

C. Future Improvement

Our future work includes improving compactness and reducing execution time to enhance the device's efficiency and comfort as a wearable. This includes optimizing the process of receiving audio output from an image caption to decrease runtime and minimize memory usage, improving user experience with faster response times and enhanced usability.

Additionally, we are executing a computationally lighter model to achieve faster and more accurate object detection. This approach not only speeds up processing but also improves the accuracy of identifying objects in images. Integrating the latest Pi camera for better image quality will further upgrade detection accuracy, assuring more reliable results for users.

Moreover, we prioritize designing a lightweight device that is easy to carry and store, highlighting user convenience for everyday use.

Lastly, enhancing the overall appearance of the device is crucial to its appeal and user experience. We aim to develop a visually appealing and modern design that complements the device's functionality, ensuring it meets both attractiveness and practical needs in diverse scenarios.

VII. REFERENCES

[1] Ackland, P., Resnikoff, S., & Bourne, R. (2017). World blindness and visual impairment: despite many successes, the problem is growing. Community Eye Health, 30(100), 71-73. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5820628/,

[2] Chen, L.-B., Su, J.-P., Chen, M.-C., Chang, W.-J., Yang, C.-H., & Sie, C.-Y. (2019). An Implementation of an Intelligent Assistance System for Visually Impaired/Blind People. 2019 IEEE International Conference on Consumer Electronics.

[3] Singh, N. J., & Nongmeikapam, K. (2019). Stereo System based Distance Calculation of an Object in Image. 2019 Fifth International Conference on Image Information Processing (ICIIP).

[4] Strbac, B., Gostovic, M., Lukac, Z., & Samardzija, D. (2020). YOLO Multi-Camera Object Detection and Distance Estimation. 2020 Zooming Innovation in Consumer Technologies Conference (ZINC).

[5] Cooper, J., Azhar, M., Gee, T., Van Der Mark, W., Delmas, P., & Gimel'farb, G. (2017). A Raspberry Pi 2-based stereo camera depth meter. 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)

[6] DANDIL, E., & CEVIK, K. K. (2019). Computer Vision Based Distance Measurement System using Stereo Camera View. 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT).

[7] Akepitaktam, P., & Hnoohom, N. (2019). Object Distance Estimation with Machine Learning Algorithms for Stereo Vision. 2019 14th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP).

[8] Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and Tell: A Neural Image Caption Generator. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[9] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R.,Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. International Conference on Machine Learning (ICML).

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L.,& Polosukhin, I. (2017). Attention is All You Need. In Advances in Neural Information Processing Systems (NeurIPS).