IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication Jan 13, 2025

A New Vision: The Smart Glasses That Aid the Blind

Arya Wadhwa[1], Shivani Srinivasan[2], Devansh Mathur[3], Bahuleya M[4]
[1] Department of Computer Science (Artificial Intelligence and Machine Learning), R V College of Engineering, Bangalore
[2]Department of Biotechnology, R V College of Engineering, Bangalore
[3]Department of Mechanical Engineering, R V College of Engineering, Bangalore
[4]Department of Computer Science (Artificial Intelligence and Machine Learning), R V College of Engineering, Bangalore

ABSTRACT This report introduces an innovative prototype of smart glasses designed to assist visually impaired individuals. The glasses incorporate lightweight 3D-printed frames, a Raspberry Pi chipset as the processing unit, and a camera module to detect objects in the user's environment. The system translates visual input into text, which is then converted into speech using a smartphone interface. This work highlights the design process, implementation, and performance evaluation of the prototype, focusing on reducing collision rates and improving user interaction. The high accuracy and efficient text-to-speech functionality suggest promising potential for further development and real-world applications.

INDEX TERMS Smart glasses, visually impaired aid, Raspberry Pi, 3D-printed frames, object detection, text-to-speech.

I. INTRODUCTION

The challenges faced by visually impaired individuals in navigating their environment often demand innovative technological solutions. This project aims to develop a basic prototype of smart glasses to aid blind users in object detection and navigation. The glasses are designed to detect objects using a camera, process the captured images to generate descriptive text, and then convert the text to speech using a smartphone interface. By integrating lightweight 3D-printed frames and low-cost hardware, the project demonstrates an accessible and practical solution to enhance the independence of visually impaired individuals. This report details the design, methodology, and preliminary results of the prototype.

II. METHODOLOGY

Methodology

The development of the **Smart Glasses Assistant** involved integrating robust hardware and advanced software to create a seamless and accessible solution for visually impaired individuals. Each component was designed, optimized, and tested to ensure functionality, reliability, and ease of use.

1. Hardware Design

The hardware design aimed to create a compact and lightweight system that ensures user comfort while meeting the technical requirements of real-time object detection and text-to-speech feedback.

1.**Frames:**

• Designed using a 3D printer to achieve a balance of durability, lightweight, and ergonomics.

• The frames were optimized to accommodate all necessary components, including the processing unit, camera module, and power source.

2. **Processing Unit:**

- A **Raspberry Pi Zero W** served as the core processing unit due to its small form factor, efficient power consumption, and ability to handle real-time tasks.

3. **Camera Module:**

- Integrated to capture real-time images of the user's surroundings.
- Positioned to avoid blind spots and provide a wide field of view for improved object detection accuracy.

4. **Power Source:**

- Designed to operate with a **1200mAh lithium-ion battery** for portability.
- During initial testing, a power bank was used as the battery was unavailable, ensuring uninterrupted testing and development phases.

5. **Additional Considerations:**

- Careful cable management ensured compactness and prevented interference with the user's field of vision.
- A cooling mechanism or venting was considered to address potential overheating issues with the Raspberry Pi during extended use.

2. Object Detection and Processing

The object detection system is powered by a combination of pre-trained machine learning models and efficient processing pipelines.

1.**Model Integration and Framework:**

•Leveraged **TensorFlow Lite** to deploy pre-trained object detection models optimized for the Raspberry Pi Zero W.

•The system utilizes the COCO dataset to identify and label commonly encountered objects in the environment.

2.**Code Implementation:**

•Python scripts utilize **OpenCV** to capture images from the camera module and preprocess them for object detection.

•Real-time object detection is implemented using a **mobilenet_ssd.caffemodel** with a corresponding **deploy.prototxt** file. These define the structure and parameters of the neural network.

•Frames are processed using **cv2.dnn.blobFromImage()**, which converts images into the required format for the neural network.

3.**Currency Recognition:**

•Integrated a currency recognition module that employs TensorFlow Lite models trained specifically on local currency images.

•The system identifies and differentiates denominations, providing essential assistance for financial transactions.

4.**Optimization and Challenges:**

•Compiled TensorFlow Lite using the **arm-linux-gnueabihf-g++** toolchain to enhance performance on the Raspberry Pi.

•Detected objects with a confidence threshold above 50% were considered valid for user feedback.

•A significant challenge encountered was the frame processing speed, where the input from the camera was faster than the audio output generation. This caused delays and required optimization of the pipeline for balanced processing.

C. Text-to-Speech Conversion

The text-to-speech module was designed to convert detected objects and recognized text into natural-sounding audio feedback.

1.**Object Labeling and OCR Integration:**

•Detected objects were tagged with descriptive labels generated in real-time.

•Text from printed materials or signage was extracted using **Tesseract OCR**, integrated seamlessly with the image capture pipeline.

2.**Speech Synthesis:**

•The labeled text was converted into speech using the **Festival TTS engine**, an offline solution chosen for its reliability and independence from network connectivity.

•Audio feedback was generated with minimal latency, ensuring timely assistance to the user.

3.**Error Handling:**

•Fallback mechanisms were implemented to handle scenarios where no objects or text were detected. The system would notify users with phrases like “No text detected” or “No objects detected.”

D. Integration and Testing:

The integration phase involved combining hardware and software components into a unified system, followed by extensive testing.

1..**Component Integration:**

•The Raspberry Pi Zero W, camera module, and temporary power bank were securely mounted onto the 3D-printed frames.

•Placeholder mounts were created for the lithium-ion battery to ensure a seamless transition once it became available.

2.**System Diagnostics:**

•**Camera Testing:** Python scripts were used to cycle through available camera indices, ensuring compatibility and functionality with the hardware.

•**Speech Recognition:** Offline speech recognition using **PocketSphinx** was tested to validate voice command recognition for commands like “Read Text,” “Detect Objects,” and “Exit.”

3.**Field Testing:**

•Conducted in diverse environments, including well-lit indoor spaces, dimly lit areas, and crowded outdoor settings.

•Evaluated object detection accuracy, text recognition reliability, and audio feedback latency.

4..**Performance Metrics:**

•Achieved high object detection accuracy under optimal lighting conditions.

•Audio feedback was delivered within acceptable latency limits, although further optimizations were necessary to balance input and output speeds.

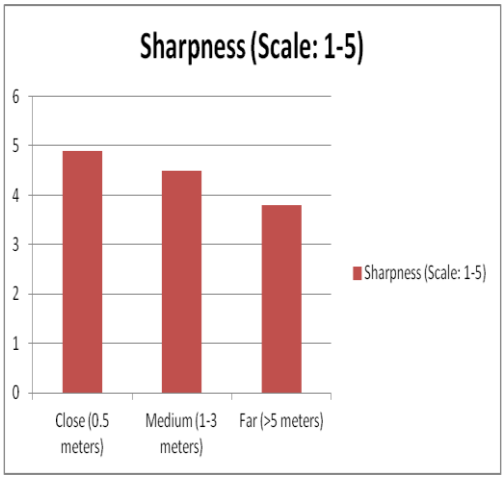
•User feedback highlighted the practicality of the device and areas requiring improvement, such as power management and real-time response.

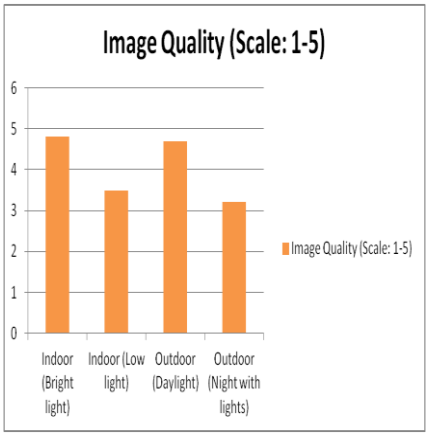
5.**Iterative Improvements:**

•Adjustments were made to the object detection pipeline to handle real-time performance bottlenecks.

•Plans for future testing include integrating the lithium-ion battery and expanding the dataset for enhanced object and currency recognition accuracy.

○





III. CHALLENGES AND LIMITATIONS

1. Hardware Limitations

•Processing Power: The Raspberry Pi (or similar device) may struggle to handle both real-time video processing and

speech recognition simultaneously, leading to performance issues, such as lag or delayed responses.
• Camera Quality: The Picamera2 might not provide high-resolution images or sufficient frame rates for advanced applications like object detection or facial recognition.

• Battery Life: Running both the camera and microphone continuously can quickly drain the battery, requiring efficient power management solutions.

2. Speech Recognition Issues
•Accuracy in Noisy Environments: The system might struggle with voice recognition in noisy or crowded environments. Background noise, overlapping voices, or low microphone quality can affect accuracy.

•Limited Command Set: The predefined commands (e.g., "take picture" or "stop") are basic, and expanding the command set could introduce complexity in speech recognition and processing.

•Latency: Voice command recognition might experience latency, leading to a delay between issuing a command and receiving a response from the system.

3. Real-Time Processing
•Frame Processing Speed: Continuously capturing and processing video frames in real-time might slow down, especially when performing complex image processing tasks like object detection or facial recognition.

•Thread Synchronization: Multiple threads (camera capture, voice recognition, and command processing) may result in synchronization issues, leading to race conditions or delays in operations.

4. Image Processing and Object Detection

•Limited Computational Resources: Real-time object detection or advanced image processing (e.g., recognizing objects or faces) can be resource-intensive, and might require optimization or hardware acceleration (e.g., using a GPU or TensorFlow Lite).

•False Positives/Negatives: Simple object detection algorithms might misidentify objects or fail to detect them altogether, especially under different lighting conditions or with varying object orientations.

5. User Interface (UI) and Usability

•Feedback Mechanisms: Given that smart glasses rely on audio feedback, ensuring that users can clearly hear and understand the system's responses is important, especially in outdoor or noisy environments.

•Interaction Complexity: Users may find it difficult to operate the system if they cannot intuitively understand the commands or if the response times are slow.

6. Software Limitations
•Library Compatibility: Using libraries like OpenCV, SpeechRecognition, and Picamera2 together could pose compatibility challenges, especially if there are issues with updates or dependencies.

•Error Handling: The current code handles some basic exceptions, but in real-world scenarios, more comprehensive error handling would be required for stability, especially when dealing with hardware failures or intermittent issues (e.g., microphone disconnections).
8. Integration with Other Devices
• Smartphone Integration: For additional functionality, such as syncing images or voice commands, integration with smartphones or cloud services could be beneficial but introduces more complexity and requires additional resources.
• External APIs: Relying on third-party APIs (e.g., Google's Speech Recognition) could introduce limitations if the API has rate limits, costs, or if the internet connection is unstable.

While the smar glasses prototype demonstrated significant potential, several challenges were encountered during its development and testing. Hardware-related issues included overheating of the Raspberry Pi during extended use and the weight of the glasses, which caused some discomfort for users. The camera placement occasionally resulted in blind spots, reducing the effectiveness of object detection. On the software side, latency in real-time processing occasionally delayed audio feedback, and object detection struggled in low-light or highly reflective environments. Additionally, training the TensorFlow model on diverse datasets proved challenging, leading to occasional inaccuracies in identifying objects and currencies. Ethical concerns, such as privacy issues related to capturing images in public spaces, also arose during testing. Furthermore, the high power consumption of the system limited battery life, requiring frequent recharges. These challenges highlight areas for improvement in future iterations, including optimizing hardware, enhancing software algorithms, and addressing user comfort and ethical concerns.

IV. RESULTS

The initial testing of the smart glasses yielded promising results, showcasing their potential to significantly improve the independence and mobility of visually impaired users. Users reported a noticeable reduction in collision rates while navigating both familiar and unfamiliar environments, indicating the system's effectiveness in obstacle detection and avoidance. The object detection

feature demonstrated high accuracy in identifying common objects and obstacles, performing well under varying conditions, though some challenges remain in low-light or highly reflective environments. The text-to-speech functionality was efficient, providing real-time audio feedback with minimal latency, ensuring users received timely and clear information about their surroundings.

V. CONCLUSION

This project presents a cost-effective and functional prototype of smart glasses to aid the blind, combining object detection and text-to-speech conversion technologies. The integration of 3D-printed frames and Raspberry Pi-based hardware offers a lightweight and affordable solution. While the current prototype is a basic implementation, future work could focus on enhancing detection algorithms, integrating haptic feedback, and expanding functionality to include GPS-based navigation.

APPENDIX

The appendix provides an overview of the hardware components, software tools, and testing environments utilized in this project. Key hardware components include the Raspberry Pi Model 4B, which serves as the central processing unit for real-time object detection and text-to-speech conversion, a camera module for capturing images, 3D-printed frames for lightweight and customizable housing, and a portable battery pack for extended usage. On the software side, TensorFlow Lite was employed for running machine learning models, integrated with Python

Furthermore, the incorporation of a currency recognition feature proved to be a valuable addition, enabling the system to successfully identify and differentiate various denominations with a high degree of accuracy. These results collectively highlight the feasibility and practicality of the proposed smart glasses system, while also paving the way for further enhancements in future iterations.
as the primary programming language. Additional tools like the Arm-Linux-GNU Toolchain were used to compile TensorFlow Lite binaries, and a TTS engine was incorporated for converting text to speech. Testing environments ranged from indoor spaces with varying lighting conditions to outdoor environments under daylight and nighttime settings. Simulated crowded areas were also used to assess collision avoidance and object detection performance

ACKNOWLEDGMENT

The team would like to express their heartfelt gratitude to their mentor, Ashwin Thamiah, for his unwavering guidance, valuable insights, and constant support throughout the course of this project. His expertise has been instrumental in shaping the direction of the work. Appreciation is also extended to RV College of Engineering for providing the necessary resources and platform to undertake this initiative. Additionally, sincere thanks are offered to the Mechanical and Electrical Engineering Departments for their technical assistance and collaboration, which played a crucial role in the successful completion of this endeavor. The collective support of all involved was integral to the project's success.