SENSEYE: A Resource-Aware Visionary Framework via IoT Edge Computing to Assist Individuals with Visual Disabilities

Akash Shingha Bappy, Md Ziaul Hoque, and Tapio Seppänen
Center for Machine Vision and Signal Analysis, Faculty of ITEE, University of Oulu, Finland

Abstract

Assistive technologies for visually impaired individuals are often limited by conventional computer vision algorithms, constrained sensor capabilities, high power consumption, and reliance on cloud-based processing, which introduce latency and privacy risks. This research presents SENSEYE, a resource-aware visionary framework utilizing edge computing to enhance independent navigation and daily task performance for visually impaired users. Built on the NVIDIA Jetson Orin Nano, SENSEYE integrates real-time object detection, scene comprehension, and global positioning system (GPS)-based navigation into a portable, low-latency device. It leverages optimized lightweight AI models, such as SSD-MobileNetV2 and VILA1.5-3b, to provide accurate environmental awareness and seamless auditory feedback through efficient speech processing. The system also enables secure remote assistance via video streaming and real-time GPS location sharing, ensuring enhanced user safety and connectivity. Evaluations confirm superior accuracy, power efficiency, and responsiveness compared to traditional sensor-based or cloud-reliant systems. However, challenges persist in refining the form factor for wearability and optimizing the user interface to prevent sensory overload. Future work will focus on enhancing accessibility through advanced AI models, multimodal sensor integration, and user-centric design, aiming to deliver a transformative assistive solution that empowers visually impaired individuals with greater autonomy and confidence.

Keywords

Internet of Things, Edge Computing, Visual Assistance Device, Artificial Intelligence, Vision Transformer, Remote Streaming.

ACM Reference Format:
Akash Shingha Bappy, Md Ziaul Hoque, and Tapio Seppänen. 2025. SENSEYE: A Resource-Aware Visionary Framework via IoT Edge Computing to Assist Individuals with Visual Disabilities. In . Oulu, Finland, 17 pages.

1 Introduction

Visual impairment affects hundreds of millions of people worldwide, creating a strong demand for assistive technologies that can enhance independence and safety. Historically, a variety of solutions have been explored, from the classic white cane (introduced in the early 20th century) and guide dogs, to tactile systems like Braille for reading. However, they have notable limitations as they require extensive training, have a limited range of detection, and convey only minimal environmental information to the user. Early electronic travel aids (ETAs) attempted to augment mobility by using sensors (e.g., ultrasonic or infrared rangefinders) to detect obstacles and then alert the user via simple feedback such as beeps or vibrations. For example, a "smart" cane with ultrasonic sensors can detect obstacles within a few meters (often around 2m range) and warn the user with sound [1]. However, such sensor-based solutions cannot identify what the obstacle is, may struggle under certain conditions, i.e., infrared sensors can be disrupted by sunlight and generally provide a one-dimensional awareness (distance to objects) rather than a rich understanding of the scene [2]. The last decade saw computer vision and artificial intelligence progress enough to create new avenues for assistive technology. Camera systems started appearing that were able to identify text, objects, or faces and report that information to visually impaired users in an auditory fashion. Initial products were things like Optical Character Recognition (OCR) reading devices and color detectors, but current systems make greater use of deep learning to provide much higher accuracy and flexibility. For instance, the Seeing AI mobile app from Microsoft uses computer vision and AI to narrate the world in real time, reading text and identifying objects or people via a smartphone [3]. Another breakthrough is the OrCam MyEye wearable camera, which clips onto glasses and uses AI to convert visual information (like printed text, faces, or products) into speech on the device, without requiring internet connectivity [4]. These AI-based aids have the potential to provide more descriptive information than sensor-based aids but usually need a lot of processing power or cloud services. Utilization of cloud connectivity can bring latency, security/privacy problems, and decrease the reliability of the system in case of loss of internet connectivity.

Internet of Things (IoT) and edge computing have emerged as pioneers to bridge this gap in assistive technology in recent days. By connecting smart sensors and devices in an IoT framework, one can combine multiple data sources (camera, GPS, inertial sensors, etc.) to gain a comprehensive view of the user's context. Edge Computing refers to performing computation near the data source (on local devices or nearby servers) instead of in distant cloud servers. This approach offers several advantages for visual assistance: it reduces response latency and bandwidth usage by processing data on-device, and it keeps sensitive video data local which enhances privacy [5]. The latest generation of edge hardware (e.g., NVIDIA Jetson, Google Coral, Apple Neural Engine) has the potential to run advanced deep learning models in real time with relatively low power consumption. This means that it is possible to perform tasks like object detection, scene comprehension, or OCR on a wearable device or handheld unit without the need for continuous cloud connectivity. IoT connectivity basically provides the sensing and communication fabric, and edge computing provides the intelligence close to the user, collectively enabling a new generation of assistive devices that are responsive and smart.

In this study, we propose a resource-aware visionary framework to assist visually impaired users, employing IoT edge computing

BIomedical Engineering Project, 2025, Oulu

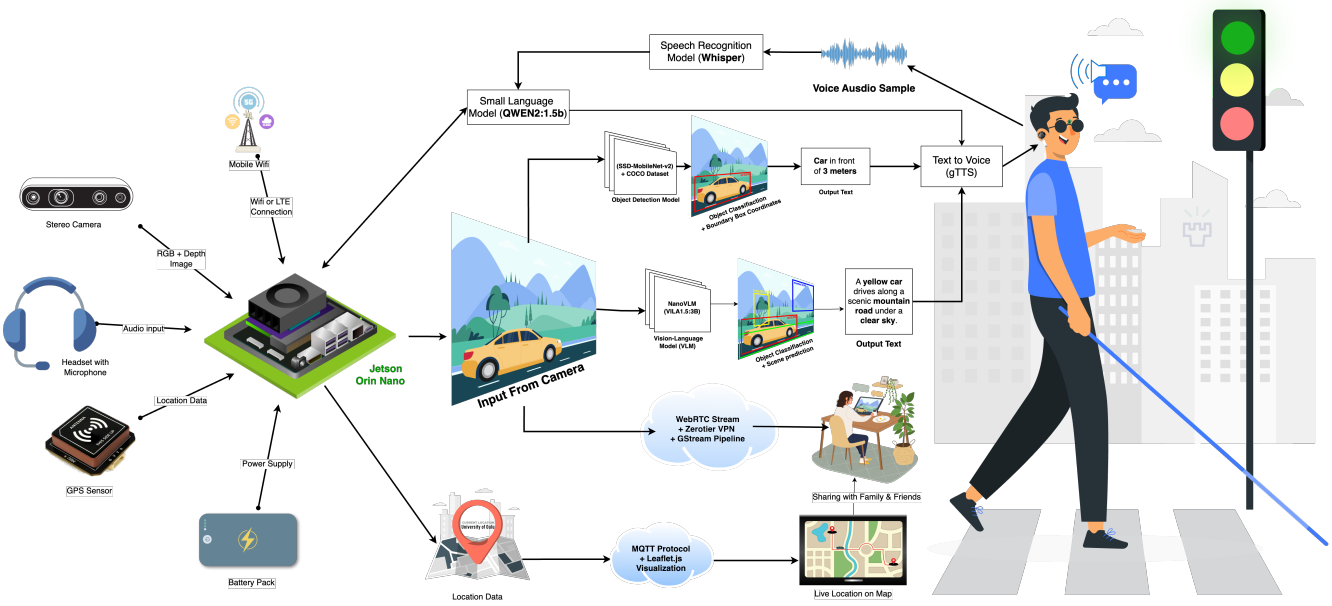

Figure 1: Overview of Resource-Aware Visionary Framework via IoT Edge Computing.

to overcome the limitations of existing approaches. The central aim is to develop an accessible system with the ability to interpret and describe the user environment in real time to facilitate them to navigate safely and perform daily tasks more independently. Unlike conventional aids, our framework integrates multiple functions, such as, it can recognize and locating objects, detect moving obstacles and sudden environmental changes, summarizing a scene from a live camera feed, and conveying all this information via audio feedback instantly. Moreover, the system offers communication features: it can initiate audio/video calls with remote assistants and share the user's live location through WiFi or cellular networks, adding a safety net for situations that automated recognition alone cannot handle. The entire assistive workflow is designed to be resource-aware, running efficiently on a small, power-constrained edge device as shown in Figure 1. This was achieved by using optimized AI models (e.g., lightweight vision transformers and convolutional neural networks with TensorRT acceleration) and a prudent allocation of tasks between the device and optional cloud services.

2 Background and Related Work

Research on assistive technology for visually impaired individuals spans several decades and can be broadly divided into two categories: non-AI-based (sensor-based) solutions and AI-based (computer vision) solutions. In addition, a number of commercial products and services have emerged in recent years, incorporating advancements from both categories. Here we review representative prior works in each category, highlighting their approaches and limitations, and we discuss how deep learning and edge computing are increasingly shaping modern solutions [6].

2.1 Non-AI-based Solutions

Early technological aids for the blind relied on sensor circuitry and simple logic rather than machine learning or computer vision. The goal of these systems was often to mimic or augment the traditional white cane by detecting obstacles and hazards beyond the cane's physical reach. A classic example is the ultrasonic sensor-based "smart cane". In such a system, one or more ultrasonic transducers emit sound waves and measure reflections to estimate the distance to obstacles. When an object is detected within a certain range, the device alerts the user via vibrations, buzzing sounds, or spoken warnings using a text-to-speech module [7]. Numerous research prototypes of ultrasonic canes have been proposed. Dambhare et al. (2011) developed a "Smart Stick for the Blind" that combined ultrasonic obstacle detection with GPS for location assistance [8]. Olakanmi et al. (2014) presented an ultrasonic sensor network embedded in a cane, providing voice guidance to indicate an obstacle's direction and distance [9]. Likewise, Gbenga et al. (2017) designed a low-cost Arduino-based smart cane with multiple ultrasonic sensors for frontal obstacle and puddle detection – their cane could detect obstacles 2 meters ahead and would beep to alert the user [1]. Elmannai and Elleithy et al. used tongue electro tactile device (TED) to provide haptic feedback to the user for transmitting information and navigation [2].

Although popular, non-AI solutions have broad limitations. Owing to a lack of complex image processing, they are incapable of identifying the type of an object or being able to generate semantic information – e.g., the systems know there is something in front of them but whether it is a car, signpost, or a human, they cannot state. This context lack can leave the user unconvinced or force them into trial-and-error. Many of the sensors, too, face environmental limitations. Infrared sensors are susceptible to being misled by sunlight

or reflective surfaces, resulting in false positives or false negatives. Ultrasonic sensors have a limited detection cone and can fail to detect drop-offs or glass barriers consistently, and overestimate distance on sloping or smooth surfaces [10]. Other sensor-based approaches, like RFID tag systems, require infrastructure (tags in the environment) and only work for tagged locations, limiting their range [11]. There are also wearable designs like sensor belts or vests that give vibrotactile feedback, and shoe-mounted sensors that warn of obstacles at foot level [12]. While useful, these still share the core limitation: they do not interpret the scene, but only report raw measurements (distance, presence of object). Another challenge is user feedback modality – a continuous beeping or vibration can be annoying or stressful if not designed carefully, and it occupies the user's senses (hearing or touch) which could otherwise be used for situational awareness. In summary, sensor-based aids provide binary or quantitative feedback about obstacles and navigation cues, but fall short in rich description and adaptability. This motivated researchers to incorporate cameras and AI into the loop, aiming for a more vision-like assistance that a human guide would offer.

2.2 AI-Based Assistive Solutions

As computer vision technology advanced, a new generation of assistive devices emerged that use cameras to visually interpret the environment. Early camera-based systems (in the 1990s and 2000s) often relied on classical image processing or simple machine learning. For example, some devices could detect and recognize printed text using OCR to read signs or documents to the user, or detect specific shapes like doorways or traffic lights using hard-coded algorithms. There were also experimental systems like the Navbelt and the vOICe which converted camera images into audio or tactile patterns to convey obstacle layout [13].

The advent of deep learning in the 2010s transformed the capabilities of vision-based assistive technology. Convolutional Neural Networks (CNNs) learned from massive image databases and began to outperform earlier methods in object recognition and scene understanding tasks. Researchers soon began to port these to assistive devices for the blind. For instance, object detection algorithms such as, You Only Look Once (YOLO) or Single Shot MultiBox Detector (SSD) can be utilized on a live camera stream and highlight object bounding boxes (obstacles, persons, cars, etc.) in real time. Adding an audio interface, the system can vocalize these objects to the user (e.g., "bicycle at 2 o'clock") or warn of imminent hazards. Seeing AI (2017) does this to an extreme degree by offering several AI "pipes" on a phone: brief text reading, document scanning, scene description, face recognition with names of people, and even currency bill identification [14].

Academic studies also demonstrated how a single camera along with a deep learning model was capable of detecting free path against obstacles to navigate. For example, Tapu et al. (2017) developed an obstacle detection and classification system on the basis of a smartphone camera in real time to assist visually impaired individuals to navigate around obstacles [15]. Coughlan et al. (2013) introduced "Crosswatch," a computer vision-enabled smartphone system designed to help blind and visually impaired pedestrians across traffic intersections by detecting crosswalks and providing alignment feedback [16]. With deep learning, accuracy and versatility improved markedly – devices could recognize hundreds of object classes, not just detect distance. Additionally, AI allowed more complex functionalities such as scene description (generating a sentence to describe the scene) and intent recognition (e.g. detecting if a nearby person is trying to get the user's attention). These AI-based systems began to approach the kind of contextual awareness a human assistant might provide, moving beyond the simple "radar-like" feedback of sensor canes.

But AI technologies introduced new challenges. Deep models are computationally costly, and running them on a handheld device puts a strain on battery and processor resources. Initial solutions tended to offload computation to the cloud. For instance, the Seeing AI app has the majority of its operations run on Microsoft's cloud servers. This is fine but at the cost of having to be online and possibly being a privacy concern (uploading one's camera stream to the cloud). This has been countered by recent efforts with edge computing by executing trained AI models on local devices. The community has succeeded in creating parameter-light vision models (e.g. MobileNet, EfficientNet, small parameter-sized transformers) that can be deployed on devices like smartphones or embedded boards. Hardware accelerators such as graphics processing unit (GPU), Neural Processing Unit (NPU) and Field-Programmable Gate Array (FPGA) in edge devices significantly speed up inference. For example, NVIDIA's Jetson family and Google's EdgeTPU can run object detection at dozens of frames per second on-device. A comparative study by Elmannai and Elleithy (2017) noted that as computation moves on-board, systems become more responsive and can function offline, which is crucial for real-world reliability [2]. One recent article, VisiSense (2024), employs an IoT and edge computing solution expressly: a smart cane with sensors and a camera transmits data to a portable edge device that runs CNN models, achieving up to 99% object detection accuracy and extremely low latency. VisiSense demonstrated that such an edge-based system outperformed earlier "Smart Stick" designs in both speed and accuracy, thanks to the integration of AI and efficient computing [17].

Another dimension of AI-based assistive tech is multimodal integration. Vision alone can be augmented with other inputs for better robustness. Some projects attach depth sensors alongside color cameras to get 3D information, improving obstacle detection, especially in low-light or high-clutter scenarios [18]. Others use GPS and digital maps to provide navigation instructions (similar to mainstream GPS navigation but with auditory/haptic guidance tailored for blind users) [19]. The trend in research is toward holistic systems that fuse camera vision, range sensing, location data, and AI reasoning to offer a comprehensive assistive experience.

2.3 Commercial Solutions and Services

In parallel to academic research, several commercial assistive products have become available, translating technological advances into real-world tools. We have already mentioned OrCam MyEye, a finger-sized camera that attaches to eyeglasses. OrCam uses on-device OCR and object recognition to read text (from books, screens, signs) and recognize faces or products, speaking the output to the user [4]. It has been well-received for tasks like reading and shopping, though it does not actively warn of physical obstacles (it's

mainly designed for reading and identification tasks). Another notable product is the Envision Glasses, a partnership between Envision AI and Google Glass hardware. The Envision Glasses are lightweight smartglasses with a built-in camera and speaker; they can "speak out text and environmental information, recognize faces, light, and colors," all through an AI vision system running on the device [20].

This allows hands-free use – the user can look at an object or sign and get audio feedback describing it. Both OrCam and Envision integrate multiple AI functions and aim for all-day wearability, illustrating the push toward wearable AI assistants. On the smartphone side, apps like Seeing AI (iOS) and Google's Lookout (Android) have brought AI assistance to millions of users at no cost. These apps leverage the phone's camera and either on-board models or cloud services to perform a suite of tasks: short text reading (e.g. reading signs instantly), document scanning, product barcode recognition, currency identification, scene description, and even person recognition. They effectively turn a standard smartphone into a Swiss-army knife of assistive vision, which is highly convenient. The limitation, of course, is that the user must hold and aim the phone camera, which occupies one hand and may be less convenient for continuous navigation assistance compared to a wearable device.

In addition to AI-driven tools, crowdsourced assistance services have also gained popularity. The prime example is "Be My Eyes", a mobile app that connects visually impaired users with sighted volunteers via a live video call [21]. A blind user can initiate a session and point their phone camera at something, and a volunteer will describe what they see (reading a label, identifying a bus number, checking an outfit's color, etc.). This service boasts millions of volunteers worldwide and has been very useful for tasks that AI is still not good at or when a human touch is desired. Although Be My Eyes is not an AI system, it supports tech solutions by offering assistance in unexpected situations. (Notably, even Be My Eyes is now exploring AI: in 2023, they introduced a "Virtual Volunteer" beta, using OpenAI's GPT-4 Vision to answer questions about images, indicating the rapid advancement of AI in this space.)

In summary, the landscape of assistive technology for visual impairment includes everything from simple sensor canes to sophisticated AI glasses. Non-AI sensor solutions laid the groundwork but are limited in capability. AI solutions enabled through deep learning have multiplied what is achievable – from reading, object detection, and scene parsing, to name a few. Concepts of IoT and edge computing are being used increasingly to make these solutions faster, reliable, and actionable in the real world without requiring perpetual internet connection as, Elmannai et al. provided a comparative look at many such devices, evaluating their features and performance [2]. The general trend is clear: recent systems provide more information and greater accuracy, but the best results come from combining multiple approaches (sensors, AI, and connectivity) to mitigate individual shortcomings. Our proposed work follows this trend by integrating state-of-the-art computer vision on an edge device with the connectivity of IoT, aiming to deliver a balanced, comprehensive solution for visually impaired users. In the next section, we detail our methodology and how it builds upon these prior efforts.

3 Methodology

In this research, we propose a comprehensive solution for blind and vision-impaired people with utilizing IoT & Edge devices that can help in their daily life. To assist them from visual data we used object recognition model with deep learning. Additionally, we used several techniques for both text to speech to convey the result and speech to text to assist them in different tasks. Last but not least, with the help of gps sensor, internet and network protocol to send data to a remote destination. To make it possible, we needed to make it ensure it is portable and power efficient to ensure its portability. Here edge compute played a key role as unlike cloud computing it can process the data in the end device ensuring shorter latency, bandwidth saving, data safety as well as maintaining the form factor to be suitable for our purpose [22].

3.1 Hardware Selection

For Edge computing there are several devices available such as Raspberry Pi, NVIDIA Jetson Series, Google Coral, Intel NUC etc. Among these we choose NVIDIA Jetson Orin Nano as it has a nice balance between computational performance and power efficiency [23]. A detailed specification is shown below with compared to a Raspberry Pi in table **??**. The Jetson Orin Nano is specifically designed to be used for AI and machine learning applications with high computation performance and power-efficient GPU acceleration, while the Raspberry Pi 4B is a single-board general-purpose computer that is less powerful in terms of AI processing capability [24]. The Jetson Orin Nano is equipped with a 6-core Cortex-A78AE CPU (Central processing unit), which is better and more energy-efficient than the 4-core Cortex-A72 CPU in the Raspberry Pi 4B. While Raspberry Pi 4B has a marginal clock speed boost (1.8 GHz to 1.5 GHz), Orin Nano gets to ride on a new and more optimized ARMv8.2 architecture, leading to better performance on AI workloads.

Perhaps the most important difference is the GPU power. The Jetson Orin Nano has a 1024-core Ampere GPU with 32 Tensor Cores and supports 40 TOPS (Tera Operations Per Second) of AI capability. The Raspberry Pi 4B, on the other hand, features the VideoCore VI GPU, which lacks dedicated AI acceleration and is really geared for light graphics processing as opposed to deep learning or heavy inference workloads. Jetson Orin Nano has with 8GB of LPDDR5 RAM (Random Access Memory), twice as much and more power-efficient compared to the 4GB LPDDR4 RAM used in the Raspberry Pi 4B. This positions it more capable of handling larger models and datasets. Storage options are also different, with Jetson Orin Nano having the capacity to utilize faster NVMe SSDs, whereas the Raspberry Pi 4B is limited to microSD storage, which writes and reads at a slower pace. In terms of power consumption, Jetson Orin Nano is more efficient at 7W to 15W compared to Raspberry Pi 4B's steady 15W power consumption, and hence Orin Nano is more suitable for power-sensitive applications.

3.2 AI Voice Assistant

Figure 2(M1) describes the operatational workflow of the AI audio assistance program. The process begins with the assistant setting up the audio input and output devices. For optimal performance, a USB headset is used to ensure high-quality audio recording and playback. The assistant then enters a continuous listening loop, monitoring

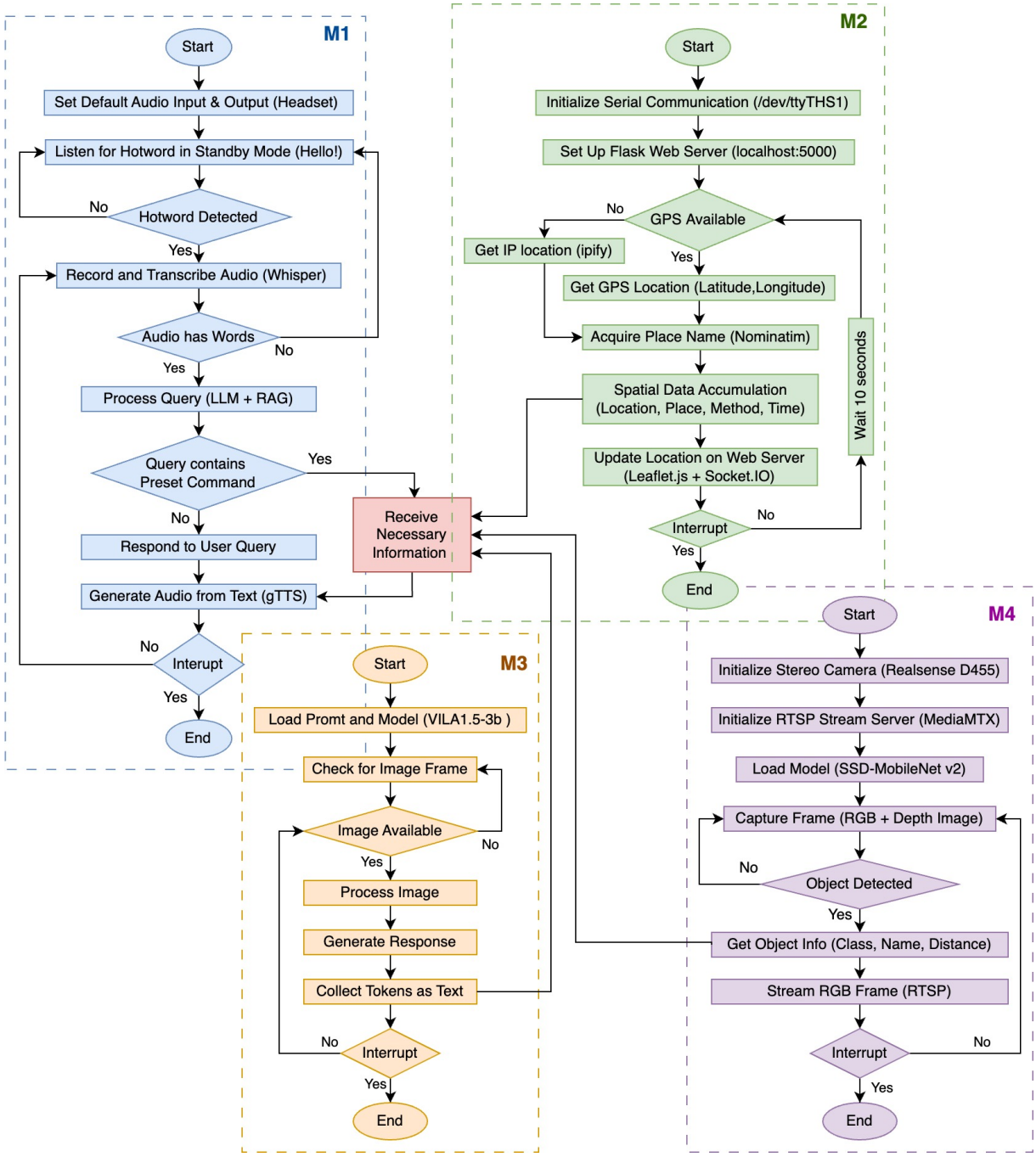

Figure 2: Flowchart of the Proposed System.

for a predefined hotword (e.g., "Hello") to activate the system. Once the hotword is detected, the assistant records the user's audio for a specific duration (e.g., 5 seconds). This audio data represents the user's query or command and is processed using Whisper [25], a state-of-the-art model for audio transcription. In this workflow, the base model of Whisper is employed, which accurately converts spoken input into text, forming the basis for further query analysis. The transcribed text is then passed to a Retrieval-Augmented Generation (RAG) pipeline [26]. This pipeline first retrieves relevant contextual information from a vector database powered by FAISS [27], ensuring that responses are grounded in a predefined knowledge base. The query, along with the retrieved context, is subsequently processed using the lannguage Model where Qwen2:1.5B

Specifications	Jetson Orin Nano	Raspberry Pi 4B
CPU	Cortex-A78AE	Cortex-A72
Cores	6	4
Architecture	Armv8.2	ARM v8
Frequency	1.5 GHz	1.8 GHz
RAM	8GB LPDDR5	4GB LPDDR4
GPU	Ampere1024-core	VideoCore VI GPU
GPU Frequency	625MHz	600MHz
AI Performance	40 TOPS	N/A
Tensor Cores	32 Tensor Cores	N/A
Connectivity	Ethernet, BT, Wi-Fi	Ethernet, BT, Wi-Fi
Storage	microSD + NVME	Micro SD
Power	7W-15W	upto 15W

Table 1: Hardware Specification Comparison of Jetson Orin Nano and Raspberry Pi 4B.

[28], a highly optimized language model with a parameter of 1.5 billion hosted locally. It generates concise and contextually accurate responses, adhering to the assistant's guidelines of maintaining a professional and user-friendly tone.

For queries containing specific keywords or commands (e.g., "get location," "detect object," "caption image"), the assistant executes predefined scripts or programs to obtain the necessary information. For instance, when a user requests to "get location," the assistant retrieves the current location and communicates it back to the user. If no specific command is detected, the generated response is processed through the Language Model. The final response is delivered to the user through audio playback. The assistant employs gTTS (Google Text-to-Speech) to convert the response into speech. This lightweight and efficient TTS system generates an audio file, which is played back immediately. The use of gTTS ensures the delivery of clear, natural-sounding responses in real-time.

At the end of the interaction, the assistant returns to the listening loop, ready to process the next query. If no audio input is detected, the assistant enters standby mode, waiting for hotword detection to resume. This workflow ensures continuous operation and responsiveness, leveraging Whisper, Qwen 2, and gTTS as core technologies for sound-to-text transcription, query processing, and text-to-speech conversion, respectively.

3.2.1 Speech to text (Whisper): As we had to use voice as an input, we needed speech to text system. For this, we have used Whisper as it is one of the groundbreaking speech recognition systems brought into reality by OpenAI, which can transpose spoken words into text accurately, unequaled to this day. Moreover, it is multilingual and able to recognize most of the dialects, hence viable for any part in the world. Whisper is resilient both in variant accents and dialects and background noises, hence able to serve continuously in various situations. It has an architecture based on transformers, hence able to work well with long-range dependencies in speech data, having a large corpus of transcribed speech as training data. Whisper is good at real-time processing; thus, it can do the job of easily and quickly transforming live speech into text with ease. High accuracy in domain-specific vocabulary can also be further achieved by fine-tuning for specific purposes. Examples of key applications include voice assistants, transcription services, accessibility tools, and language learning aids. Of course, Whisper would be greatly helpful in improving communication and documentation within so many disciplines, with its high accuracy and adaptability. Figure 3 contains key stages of speech recognition, right from the input voice signal. It starts with speech enhancement, where the quality of the voice signal is enhanced. After that, it proceeds to feature extraction, which involves identifying and isolating only those significant features from the enhanced speech signal. The extracted features are passed onto the Phonetic Unit Recognition module, which recognizes the phonetic units of speech. These are further mapped to text through Acoustic Modeling.

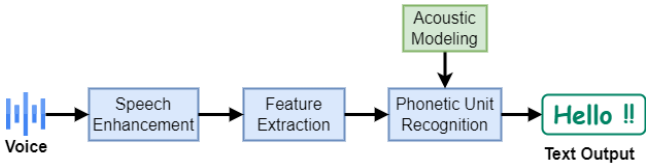

Figure 3: Speech Recognition Process.

3.2.2 Text to Speech (TTS): . As we could not convey the generated text output directly to the blind person, we needed to convert it to audio first. For this, we have used text to speech(TTS) mechanism and delivered it as voice output. Text-to-speech (TTS) technology transforms written text into spoken words through a series of intricate steps. Historically, TTS has advanced from basic text-reading machines to today's sophisticated systems, thanks to deep learning algorithms and neural networks, which enable more natural and expressive voices. The process begins with text analysis, breaking down sentences and words to understand their structure and meaning. Next, linguistic processing converts the text into phonemes, essentially translating written language into a format machines can speak. This is followed by assigning prosody, rhythm, and intonation to ensure a natural flow. The final step is voice synthesis, where the system produces audible speech that closely resembles human conversation.

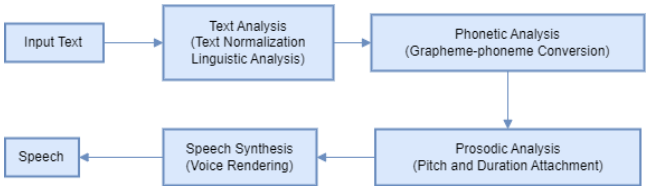

Figure 4: Text to speech synthesis - block diagram.

3.3 Real-Time Object Detection

The object detection system integrates stereo camera technology and advanced deep learning models to provide real-time detection of objects. The system begins by initializing the Intel RealSense D455 stereo camera, which captures high-resolution color images and depth data simultaneously. This depth information is essential

for calculating the spatial positioning of detected objects. The SSD-MobileNetV2 [29] object detection model is then loaded, offering the ability to detect multiple objects in a frame, classify them, and provide their class number, object names and confidence scores. It was trained on COCO(Common Objects in Context) [30] dataset containing 91 classes or objects which are found common in daily life. We choose this model as it outperformed other models like SSD-Inception-v2 or Yolo in terms of speed and accuracy which can be found in the result section.

The detection program workflow is presented by Figure 2(M4) where the stereo camera operates by continuously capturing frames in real time and upon activation of the detection command, these frames are run through the detection model. Following object detection, their spatial position is calculated based on depth data from the RealSense camera. The detection output provides the object's class label (e.g., "person," "chair") and the estimated distance of the object from the camera. The data above is then rearranged into a text narrative, as in the sentence "Person detected at 1.5 meters." The generated text is subsequently relayed to the audio assistant program, which transforms it into speech via the gTTS system. The audio response thus generated is then relayed back to the user, hence facilitating real-time object detection and spatial awareness. This procedural model enables efficient and accurate detection of objects by smoothly integrating camera technology, complex deep learning algorithms, and audio feedback mechanisms.

3.3.1 Deep Learning: Deep learning is a part of machine learning that has greatly improved artificial intelligence [31]. It uses networks of connected layers to find patterns in large sets of data. These networks work like the human brain, learning from raw data without needing a lot of manual adjustments. The key method in deep learning is back-propagation, which repeatedly tweaks the network to reduce errors and improve accuracy. This approach has led to big advances in many areas. For example, CNNs have made it easier to recognize images and videos, which helps in medical imaging, self-driving cars, facial recognition, and many other applications. An overview of the convolution network is shown below in Figure 5.

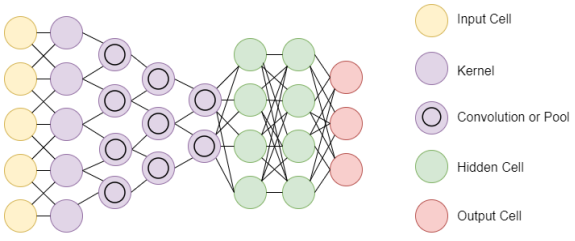

Figure 5: Deep Convolution Network.

3.3.2 SSD-MobilenetV2 Model: It is a real-time optimized lightweight object detection model that combines the Single Shot MultiBox Detector (SSD) [32] architecture with MobileNetV2 as the feature extractor. SSD is fast because it does not need region proposal networks; hence, it is significantly faster compared to two-stage detectors like Faster R-CNN. Rather than handling different object sizes individually, SSD uses several feature maps of different resolutions to find objects of different sizes. MobileNetV2 complements SSD by delivering an efficient computation backbone. Efficiency is attained through the use of inverted residuals coupled with linear bottlenecks, which facilitate effective information flow throughout the network while keeping the model size slim. Depthwise separable convolutions then reduce the number of parameters and computations even further without drastically compromising accuracy.

The MobileNetV2 architecture is represented in the figure 6 with a starting 3x3 convolution and then successive Inverted Residual blocks. These are increasingly complex blocks, enabling feature extraction at different levels. The final layers of the network are used by SSD to predict bounding boxes as well as class probability. SSD-MobileNetV2 is particularly ideal for edge computing and embedded systems, such as Jetson Orin Nano, where low power consumption with real-time inference is a necessity. Its speed-accuracy trade-off makes it ideal for real-time applications. With TensorRT optimization, SSD-MobileNetV2 can actually deliver even lower inference times, making it an effective solution for real-time object detection on resource-constrained hardware.

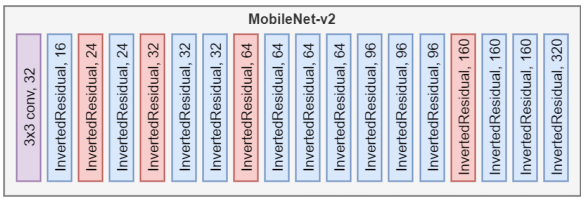

Figure 6: Network architecture of Mobilenet-v2.

3.3.3 Optimization with TensorRT: . NVIDIA TensorRT is a deep learning inference library that provides high-performance computation to accelerate neural network execution on NVIDIA GPUs, particularly for edge devices like the Jetson Orin Nano [33]. TensorRT optimizes models to achieve low latency, high throughput, and low memory consumption to be appropriate for real-time systems. TensorRT is efficient because it achieves precision reduction and layer fusion. It converts models from FP32 (32-bit floating point) to FP16 (16-bit floating point) or INT8 (8-bit integer), greatly speeding up inference but maintaining almost original accuracy. This reduction in precision is particularly beneficial for resource-constrained environments, such as embedded AI applications. Additionally, TensorRT fuses several neural network layers into a single operation that reduces memory access and computation overhead, which leads to a quicker execution time.

Figure 7 illustrates the TensorRT workflow that consists of two key components. One is TensorRT Builder, which compiles and optimizes an input model, e.g., Open Neural Network Exchange (ONNX), with operation fusion and precision reduction. Another one is TensorRT Runtime, which executes the optimized model on GPU and DLA (Deep Learning Accelerator) hardware with CUDA-based optimizations for performance at inference speed. Through the leverage of TensorRT, software on Jetson Orin Nano provides

significantly higher inference rates than traditional CPU-based execution, and it is ideal for real-time deep learning applications such as computer vision, object detection, and AI-powered automation.

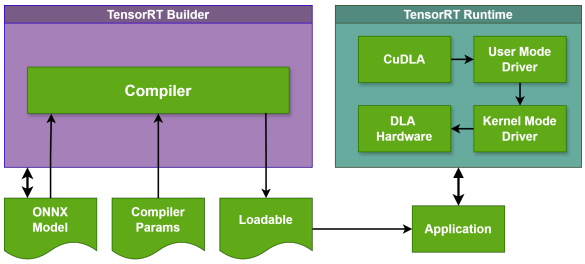

Figure 7: Types of optimizations performed by TensorRT to optimize models.

3.4 Real-time Scene Understanding using Visual Language Model (VLM)

The Visual Language Model (VLM) [34] process is designed to assist visually impaired individuals by converting images into descriptive text, which is fed into the assistant program to be translated to audio. This enables real-time image-to-speech functionality so that blind users can perceive the world around them through AI-generated spoken descriptions. The process begins with model loading and parameter initialization to optimize performance. We have chosen the model VILA1.5-3b [35] with a maximum context length of 128 and 32 maximum new tokens. A pre-defined prompt was used to describe the image concisely to makes sure that the model generates short and meaningful descriptions suitable for speech. The system then repeatedly searches for an image frame from a connected camera or an input source. If there is no image, it waits and keeps searching until an image can be processed.

When an image is discovered, the processing stage begins. The image is preprocessed to be in the proper format to be extracted for its features. The program reads the image and identifies prominent objects, scenes, or activities and generates a structured text description of the image. The description is then formatted into a comprehensible output so that it can be utilized for voice synthesis. The generated text is then fed into an assistant program, which allows the user to receive real-time descriptions of the environment around them and use them to navigate spaces, identify objects, or comprehend events taking place nearby. The system is designed with low-latency processing to guarantee that the feedback is received instantly without delays.

Figure 2(M3) describes the workflow of the mentioned process. In order to be effective, the program runs in a continuous loop and the system will run until the interrupt signal is given, delivering uninterrupted functionality for real-time support. This assistive technology based on vision-language model is particularly beneficial to blind and visually impaired individuals, making them more independent to interact with the world. Employing vision-language models and speech synthesis, the system creates a common interface between visual information and auditory feedback, thus making everyday tasks easier. This formal procedure enables real-time, AI-based assistance, which increases independence and quality of life for the visually impaired.

3.5 GPS Live Location Sharing

The location tracking and visualization system brings together GPS-based and IP-based geolocation functionality, MQTT messaging, and a Flask server to provide real-time location updates . The Jetson device receives the current location with the help of a GPS sensor frequently and pushes it to the server over MQTT (Message Queuing Telemetry Transport) through an internet connection [36]. MQTT is a publish/subscribe, low-bandwidth, high-latency network protocol specifically designed for low-bandwidth networks, thereby appropriate to use in telemetry applications. The process begins with establishing the MQTT client on the topic location/live to send data. Serial communication with the GPS module is configured using /dev/ttyTHS1, and a Flask server (localhost:5000) is run to handle real-time visualization [37]. The system has GPS-based geolocation as its priority, and it accepts latitude and longitude coordinates from the SIM7600X module in the form of AT commands and translates them into decimal degrees. If GPS data is unavailable, the system falls back on IP-based geolocation using the ipify online server with rough location coordinates. As can be seen from Figure 8, the system adopts a publisher-subscriber mechanism with an MQTT broker to process location data. The broker relays this data to subscribers, such as mobile devices or remote computers, so that real-time location updates are accessible across platforms.

Once the location data is acquired, it is encapsulated as a formatted payload string of Latitude, Longitude, Retrieval Method (GPS or IP), Place Name (via Nominatim reverse geocoding) and Last Update Timestamp. This payload is sent to the MQTT topic such that real-time data is available to the Flask server. The Flask application, combined with Socket.IO, processes these updates and sends them out to a web-based frontend. Leaflet.js [38] map displays the location, and interactive popups and markers reflect retrieval method, coordinates, and update time for clear and real-time visualization. The system automatically tracks and refreshes the location at regular intervals (e.g., every 10 seconds), dynamically toggling between GPS and IP-based geolocation for flexibility and reliability. MQTT plays a critical role in efficient data transmission as its resource-constrained device-optimized architecture supports scalable and dependable IoT-based location tracking. This system is shown in Figure 2(M2), which has the capability of seamless integration of geolocation, communication, and web-based interactive technologies and is therefore best adapted to sensor networks, IoT-driven applications, and real-time observation systems.

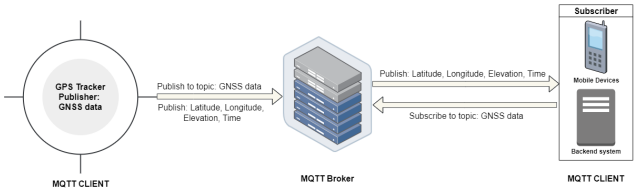

Figure 8: Diagrammatic Representation of the MQTT Process.

3.6 Remote Streaming

Figure 9 describes the he remote streaming system combines the stereo camera, Flask server, and WebRTC technology to enable live video streaming from the device to a remote client. It utlizes the rtsp stream from localhost (ie.rtsp://127.0.0.1:8554) that was generated from the object detection program to get the live feed. In case it fail to do so or if the rtsp stream is not available then stereo camera(ie. Intel Realsens D455) is initialied direcly to capture color frames, which are then processed and encoded into JPEG format. The Flask server is launched to host the video stream at 'localhost:5000/stream', and the WebRTC [39] server is started to facilitate real-time communication. The system establishes a WebRTC connection between the device and the remote client using virtual private network technology that enables secure and low-latency video streaming in real time.

The remote client can access the video stream through a web browser, which connects to the Flask server and receives the live feed. The client-side WebRTC technology ensures efficient and secure data transmission, providing a seamless and interactive viewing experience. The system continuously monitors the video stream, ensuring stable and reliable communication between the device and the remote client. In case of interruption (e.g., network failure), the WebRTC connection are safely terminated, and all resources are cleaned up. This workflow delivers efficient and secure live video streaming, seamlessly integrating camera technology, web-based communication, and real-time interaction technologies.

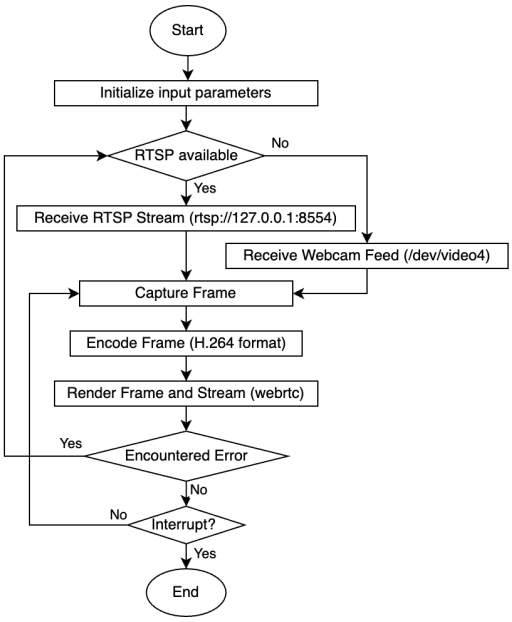

Figure 9: Remote Streaming Workflow

.

4 Result

The Results section presents the performance evaluation of the prototype, highlighting its efficient real-time capabilities in object detection, language model processing, and remote streaming.

4.1 Prototype

The image illustrates the prototype containing all the significant parts. The focal part of the prototype is the NVIDIA Jetson Orin Nano, which is an energy-efficient, GPU-accelerated edge platform with the capability of real-time inference speed for deep learning models. Combined with a stereo camera, the system takes color and depth images for precise object detection and spatial perception, with interactive audio feedback given through headphones. It delivers stable connectivity via the SIM7600G-H 4G module with the support of an antenna to enable real-time remote streaming, GPS location sharing, and remote support. It is designed to be portable, with the entire system powered by a small battery, enabling continuous use without external power sources, and it includes a 512-gigabyte solid-state disk (SSD) for storage. By performing computation locally on the Jetson Orin Nano rather than depending on cloud services, the prototype minimizes latency, improves privacy, and provides instant responsiveness that are essential for user autonomy and safety.

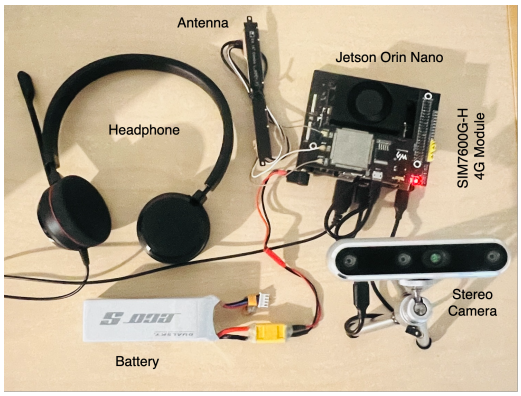

Figure 10: Visual Representation of the Prototype.

4.2 Performance Evaluation of LLM Models

Table 2 demonstrates the performance of different large language models that were used for the AI assistant. Every one of these models was prompted with the following text: **"I've been feeling dizzy, with chest pain. Give an answer in one line of what could be causing this?"** Then these models were evaluated on the answer's relevancy, response time, and computational efficiency. Among them, Qwen2-1.5B was considered the best-suited model since it was able to provide a relevant and accurate response with reasonable inference time and computational cost.

The answer provided by Qwen2:0.5B was **"Heart condition like heart attack or heart failure, blood pressure issues, vascular disease, seizure disorders, anxiety or depression, physical exertion, and dehydration can cause sudden dizziness with chest pain. Consult a doctor if symptoms persist."**, which was coherent and medically appropriate, explaining that dizziness and chest pain may be related to heart disease, anxiety disorders, hyperventilation, or stress. The answer was concise and thoughtful, which is essential in live situations where split-second generation of instant answers is demanded. A few models provided much too

detailed responses that, while complete, were too lengthy to qualify as a quick response. As one example, TinyLlama cited celiac disease, varicose veins, and coffee consumption, all of which are not typically linked with dizziness and chest pain. Smollm also repeated the terms "anxiety" and "dehydration" a few times, resulting in a repetitive answer.

Regarding speed, Qwen2-1.5B was one of the best models, taking only 2 seconds to complete a response and ranking as one of the fastest models tested. Larger models, i.e., Llama3.2 and TinyLlama, were significantly slower, with most responses taking over 10 seconds, which makes them unsuitable for real-time applications. The token processing speed of Qwen2:1.5B (4823.88 tokens/sec at prompt evaluation time and 28.70 tokens/sec at response generation time) also speaks about how it can provide fast responses without affecting accuracy. The least-performing ones in overall inference time and token generation rate were Smollm, TinyLlama, TinyDolphin, Llama3.2, and Qwen:1.8B. Smollm was the slowest of the lot at 18.65 seconds with a paltry token rate of 26.46 tokens/sec, and hence useless for real-time applications.TinyLlama (15.74s, 42.66 tokens/sec) and TinyDolphin (7.72s, 44.53 tokens/sec) also performed poorly. On the other hand, Although qwen2:0.5b was the fastest model with a token evaluation rate of 52.29 and 2 second of completion time, it had only 0.5B parameters, which may not be sufficient for more complex tasks. Last but not least, Llama3.2 provided a well-structured response; its slow processing speed made it inefficient, having a 10.90-second completion time with a 12.74 tokens/sec rate.

Overall, the assessment finds Qwen2:1.5B to be the most suitable candidate for this particular task as it was able to achieve a trade-off between performance and efficiency, making it a perfect choice for scenarios where fast and pertinent responses with minimal resource consumption are required. It provides applicable information in a condensed manner, performs well on low hardware, and possesses real-time capability, which makes it the optimal model for real-time AI assistant applications.

4.3 Object Detection with SSD-MobilenetV2 and TensorRT Optimization

Figure 11 shows the object detection result using the SSD-MobilenetV2 model optimized using TensorRT with INT8 precision, providing an astonishing 4.51 ms inference time. The model detects and classifies over one object, a bicycle (99.0% confident), a dog (96.0% confident), and a car (80.0% confident). The high confidence score indicates the capacity of the model to operate under real-world object recognition even with compromised precision to achieve faster execution.

The TensorRT INT8 quantization is computationally efficient to the point of attaining real-time inference with negligible loss in detection accuracy. The distinctive foreground bicycle and dog achieve the highest confidence scores, while the partially occluded vehicle in the far background detects marginally lower confidence. The detection box placement is very accurate, reaffirming model robustness regardless of INT8 optimization. We also tried detection with several other models, precision levels, and optimization methods, which are described in later sections.

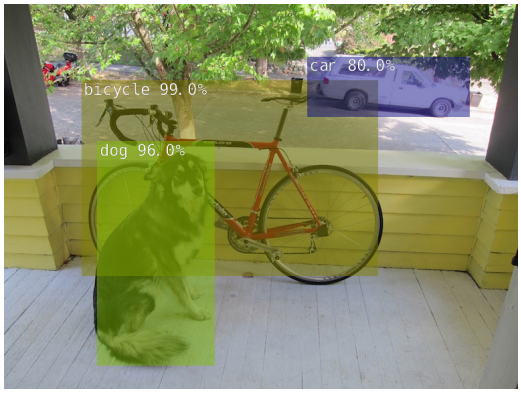

Figure 11: Object detection results using SSD-MobilenetV2.

4.4 Performance Metrics for Various Object Detection Models

Table 3 represents the performance of some object detection models that have been tested in our study. The models tested include SSD-MobilenetV2, SSD-InceptionV2, YOLOv8 (n, s, m, l, x), and YOLO11 (n, s, m, l, x), with varying precision (FP32, FP16, and INT8) and TensorRT optimization. The performance was determined through detected objects, confidence, and inference times.

From the test results, SSD-MobilenetV2 (INT8) took the fastest inference time of 4.51 ms with high confidence scores on detected objects. Although SSD-InceptionV2 performed well, it was slightly slower at 9.52 ms. Among the YOLO models, YOLOv8n (INT8) took the lowest inference time of 4.65 ms but at the cost of a drastic loss of class as well as confidence in detected objects. The larger models like YOLOv8x and YOLO11x (FP32) were highly accurate in detection but took very high inference times of over 100 ms. Relative to different precisions, FP32 models had the highest confidence values at all times but were the slowest, whereas FP16 was the best balanced in terms of speed and accuracy. INT8 precision drastically reduced inference time but at the expense of detection confidence, particularly for the smaller objects such as the car.

These results highlight that SSD-MobilenetV2 TensorRT INT8 optimized provides the best speed-accuracy trade-off for real-time applications, whereas YOLO models in FP16 are suitable for those applications where higher accuracy is necessary at reasonable inference times. Enhanced performance might be obtained by fine-tuning INT8 quantization methods to boost detection confidence without compromising efficiency.

4.5 Inference Time Analysis: Jetson Orin Nano vs. Raspberry Pi 4B

Figure 12 provides the inference timing comparison of Raspberry Pi 4B and Jetson Orin Nano reveals significant performance disparity with different YOLO models in NCNN, ONNX, and TensorRT frameworks. Jetson Orin Nano consistently measures smaller inference times, with TensorRT working the most efficiently-running YOLO11n at 7.2 ms and YOLOv8n at 11.1 ms. Raspberry Pi 4B, in contrast, is over 400 ms to run the same workloads, with ONNX doing the worst at 610.2 ms for YOLOv8n. Even with NCNN, the

Model	Total Duration (s)	Load Duration (ms)	Prompt Eval Count	Prompt Eval Duration (ms)	Prompt Eval Rate (tokens/s)	Eval Count	Eval Duration (ms)	Eval Rate (tokens/s)
qwen2:0.5b	2.005	27.08	500	65.21	7667.65	27	516.34	52.29
qwen2:1.5b	**2.021**	**25.58**	**371**	**76.91**	**4823.88**	**46**	**1602.82**	**28.70**
gemma2:2b	8.099	48.11	441	78.11	5646.10	121	7694.88	15.72
phi3:latest	1.406	10.45	35	143.48	243.94	20	1205.25	16.59
qwen:0.5b	0.681	34.30	30	36.53	821.27	28	562.16	49.81
qwen:1.8b	6.561	30.33	30	68.90	435.41	174	6415.50	27.12
qwen:4b	3.871	25.91	30	108.73	275.91	52	3694.73	14.07
llama3.2:1b	6.922	37.85	46	70.97	648.21	165	6769.92	24.37
llama3.2:latest	10.900	44.73	46	132.70	346.66	136	10675.89	12.74
gemma:2b	3.820	1971.36	48	86.93	552.19	34	1715.26	19.82
qwen2.5:0.5b	1.941	27.33	50	79.25	630.95	82	1694.16	48.40
tinyllama	15.748	11.82	60	62.50	960.03	663	15542.16	42.66
orca-mini	7.258	2376.32	64	166.16	385.18	71	4666.87	15.21
tinydolphin	7.727	858.68	55	59.49	924.48	301	6760.24	44.53
smollm	18.658	1619.35	30	61.99	483.93	448	16930.16	26.46

Table 2: Performance Metrics for Various Large Language Models.

Model	Precision	Detected Objects (Confidence)	Inference Time(ms)
ssd-mobilenet-v2	**int8**	**Bicycle (99.02%), Dog (96.04%), Car (80.03%)**	**4.51**
ssd-inception-v2	int8	Bicycle (99.12%), Dog (94.14%), Car (60.74%)	9.52
Yolov8n	fp32	Bicycle (89.71%), Dog (93.21%), Car (61.60%)	11.05
	fp16	Bicycle (89.75%), Dog (93.21%), Car (61.87%)	6.58
	int8	Bicycle (36.74%), Dog (26.90%)	4.65
yolov8s	fp32	Bicycle (88.83%), Dog (93.46%), Car (74.90%)	20.39
	fp16	Bicycle (88.82%), Dog (93.51%), Car (74.90%)	11.91
	int8	Bicycle (65.23%), Dog (77.84%)	5.73
yolov8m	fp32	Bicycle (95.40%), Dog (94.27%), Car (79.02%)	40.20
	fp16	Bicycle (95.31%), Dog (94.19%), Car (79.05%)	23.72
	int8	Bicycle (67.44%), Dog (66.83%)	8.40
yolov8l	fp32	Bicycle (97.73%), Dog (95.98%), Car (90.30%)	62.63
	fp16	Bicycle (97.71%), Dog (96.00%), Car (90.19%)	29.40
	int8	Bicycle (92.53%), Dog (76.54%)	18.50
yolov8x	fp32	Bicycle (98.32%), Dog (95.37%), Car (92.88%)	123.30
	fp16	Bicycle (98.39%), Dog (95.31%), Car (92.92%)	62.10
	int8	Bicycle (90.14%), Dog (89.67%)	27.40
yolo11n	fp32	Bicycle (93.86%), Dog (92.71%), Car (49.15%)	9.18
	fp16	Bicycle (93.95%), Dog (92.77%), Car (49.22%)	7.03
	int8	Bicycle (54.72%), Dog (76.43%)	4.78
yolo11s	fp32	Bicycle (94.76%), Dog (93.86%), Car (56.21%)	19.07
	fp16	Bicycle (94.82%), Dog (93.85%), Car (56.40%)	11.04
	int8	Bicycle (36.12%), Dog (74.39%)	5.38
yolo11m	fp32	Bicycle (95.16%), Dog (94.22%), Car (70.76%)	38.40
	fp16	Bicycle (95.07%), Dog (94.19%), Car (70.70%)	21.40
	int8	Bicycle (85.74%), Dog (90.02%)	10.98
yolo11l	fp32	Bicycle (96.74%), Dog (94.25%), Car (78.98%)	49.20
	fp16	Bicycle (96.78%), Dog (94.29%), Car (79.30%)	26.04
	int8	Bicycle (70.79%), Dog (77.71%)	14.41
yolo11x	fp32	Bicycle (95.07%), Dog (95.33%), Car (90.87%)	102.70
	fp16	Bicycle (95.07%), Dog (95.36%), Car (90.92%)	44.99
	int8	Bicycle (84.41%), Dog (84.90%)	24.06

Table 3: Performance Metrics for Various Detection Models.

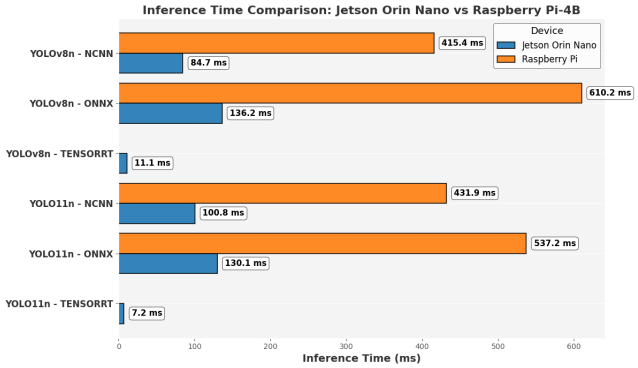

Figure 12: Comparison of YOLO Models on Jetson Orin Nano and Raspberry Pi 4B Using NCNN, ONNX, and TensorRT.

low-power version, Jetson Orin Nano outperforms Raspberry Pi by a significant margin, demonstrating nearly five times faster inference in most use cases.

These findings emphasize the advantages of Jetson Orin Nano for edge AI applications where low latency is critical. The efficiency of TensorRT also puts into prominence the need for optimized inference engines as it reduces processing time by a wide margin compared to NCNN and ONNX. While Raspberry Pi 4B remains a viable option for light AI workloads, its huge inference times make it incompatible with real-time applications. Jetson Orin Nano, being able to leverage GPU acceleration and TensorRT optimizations, enjoys a dramatic speedup, rendering the deployment of deep learning models at the edge not to have severe latency concerns. This disparity is intended to emphasize the value of employing the right hardware in AI-based solutions, particularly where compute performance and responsiveness are paramount. More research will be needed to investigate other optimizations like model quantization and hardware-aware pruning to further enhance inference speed and power efficiency for low-power embedded AI systems.

4.6 Evaluation of Visual Language Models for Concise Scene Description

Table 4 demonstrates the performance of different Visual Language Models (VLMs) on the following prompt: "**Describe the scene concisely.**" Upon evaluating the result, VILA1.5-3B gave the most accurate and concise description among the models tested, which was **"Black and white dog sits on a porch next to a red bike."** This response clearly indicates the central objects in the scene, such as the porch, bike, and dog, without additional information. It is accurate, significant, and simple to decode, with a high decode rate (24.96 tokens/sec), which is suitable for real-time applications.

The other models had mixed levels of success. The LLAVA-v1.5-7B created a caption that was generally accurate but contained speculation, such as in the line "The dog appears to be relaxed and enjoying the outdoor" information not obviously apparent. VILA-7B, by contrast, gave an excessively detailed description, referencing features such as "a wall, frame, and seat," resulting in an unnecessarily complicated caption.VILA-2.7B also generated a simple and complete caption, but it was not concise enough compared to the best one. The poorest performing model was VILA1.5-8B, which completely failed. Rather than giving a relevant caption, it generated irrelevant questions regarding bicycles and was unable to comprehend the task. Moreover, it had the lowest decoding speed (11.81 tokens/sec) and was thus inefficient and computationally costly.

Overall, VILA1.5-3B was the most reliable model, delivering an accurate, concise, and computationally efficient response. This comparison highlights that smaller, well-optimized models can outperform larger ones in real-time applications, especially when concise and meaningful image-to-text conversion is required.

4.7 Remote Streaming Performance

Figure 13 shows a snapshot of real-time remote video streaming over a virtual private network. The streaming client captures video frames and transmits them over the network using the WebRTC protocol, as indicated by the command output at the bottom of the screen. This setup allows for fast transmission, which is ideal for applications that call for real-time video feedback. The video stream statistics indicate that the transmission and decoding are smooth, with a bitrate of 4.149 Mbps indicating the connection is stable. The frame rate remained between 15 and 30, varying with the strength of the network, for smooth motion on the video. The low jitter value (0.017 ms) also indicates stable transmission with minimal variation, making sure the video quality is good.

A printed advertisement is visible from the video stream, displaying the definition of the streamed video. The text and images are readable despite the compression and streaming across a network. This indicates the success of the encoding process in ensuring that small details in the stream can still be discerned despite bandwidth constraints. The results demonstrate that the remote streaming system performs well, with clear and continuous video output and minimal latency. This setting is particularly effective for applications such as real-time monitoring, distant observation, and telepresence, where real-time visual feedback is crucial. Through WebRTC-based transmission and ideal encoding parameters, the system ensures optimized performance in low-latency remote video streaming on low-end devices.

4.8 Live GPS Location Sharing

The Figure 13 shows the live location tracking system comparison of two location detection methods: which are GPS-based location detection and IP-based location detection. The left panel (A) shows a GPS-based location found in Kaijonharju, Oulu, while the right panel (B) indicates an IP-based geolocation. In panel A, the location is determined based on GPS signals of a highly accurate location. The location identified is marked by a pin marker with the exact address and the location of the signal is labeled as "GPS". The system

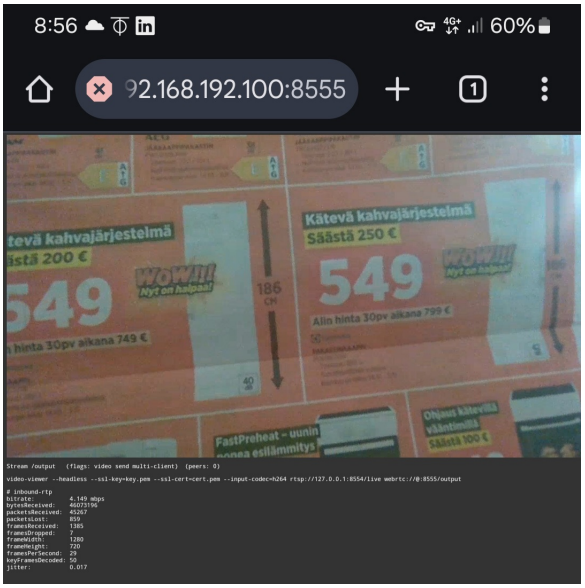

Figure 13: Real-time Remote Video Streaming.

displays the location in real-time with the last update 1 second ago, hence giving precise tracking. Panel B is another IP-based location tracing technique, utilized if GPS is unavailable. It provides a rough location or city and only specifies the general area in which the system is located, as the IP location tracking is not as specific as GPS.

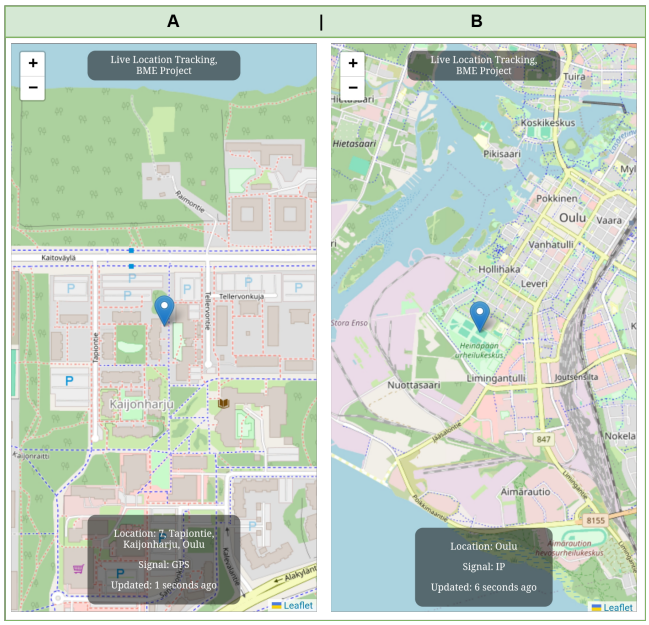

Figure 14: Live GPS Location Sharing.

Both panels blend interactive maps that can be zoomed and navigated with pin markers showing located positions. Bottom overlays give pertinent information, including location name, tracking

Model	Generated Text Outpu	Input Token	Output Token	Decode Rate (token/sec)
VILA1.5-3b	**black and white dog sits on a porch next to a red bike.**	**247**	**19**	**24.9636**
VILA1.5-8B	1) What is the difference between a bicycle and a regular bicycle? 2) What is the difference between a bicycle and a mountain bike? 3) What	240	32	11.8104
VILA-2.7b	A dog is sitting on the porch of a house. There is a bike on the porch.	627	23	23.2594
VILA-7b	1. A red bicycle with a black frame and a black seat is leaning against a yellow wall. 2. A black and white dog	627	32	13.3452
LLAVA-v1.5-7b	The image features a dog sitting on a porch next to a red bicycle. The dog appears to be relaxed and enjoying the outdoor	627	32	12.5932

Table 4: Performance Metrics for Various Vision Language Models on a specific prompt.

method (GPS or IP), and time of last update. The results demonstrate the effectiveness of location tracking in real-time with GPS providing precise location and IP-based tracking helping to supply the location if the GPS signal is lost. The system switches automatically between both methods to enable continuous and consistent tracking and is suitable for use in real-time, where precise, up-to-date location information is required.

5 Discussion

The proposed IoT-edge vision assistance system was also compared with existing solutions in order to contrast its performance, accuracy, and efficiency, but also practical usability. Overall, the results validate that our edge computing solution achieves significant gains for practical use, while there are also notable problems to be tackled and some limitations exist. We discuss these aspects in more detail below, contrasting with prior work and reflecting on what it would mean to use such a system in practice

5.1 Real-Time Performance:

One of the core goals of our system is to achieve real-time analysis of the user's environment. During experiments, the device (powered by the NVIDIA Jetson Orin Nano) was able to run the object detection model at frame rates sufficient for continuous mobility assistance (on the order of tens of milliseconds per frame, i.e. 30–60 frames per second). In comparison, we implemented a similar setup on a Raspberry Pi 4 (a popular low-power board) and found that the Pi could not sustain reliable real-time processing for the same model – its inference time was several times longer (lower FPS), leading to noticeable lag. This comparison highlights the benefit of an edge AI platform with a GPU: our Jetson-based solution delivered an order-of-magnitude faster inference than a CPU-only embedded board, which means the user gets timely warnings and descriptions without lag. Fast response is critical for safety; for example, if a fast-moving obstacle (like a cyclist) is approaching, the system must alert the user immediately. Our edge device introduces only a fraction of a second of processing delay, making it practically as fast as the human reaction needs. This outperforms many cloud-based approaches as well – while cloud servers can be powerful, sending images to the cloud and receiving responses can add 1–2 seconds of network latency. By processing locally, we achieved much lower end-to-end latency (often <100 ms for object detection and speech output). This finding is consistent with other research showing that edge computing can meet real-time constraints better than cloud offloading for vision tasks [17].

5.2 Accuracy and Recognition Capabilities:

We utilized advanced deep learning models to render our system extremely accurate at identifying objects. During testing, the object detection module (an optimized SSD MobileNetV2, optimized with TensorRT) correctly identified and localized everyday objects and obstacles in the environment with high mAP (mean average precision) similar to the best for that model architecture. For instance, it reliably detected doors, chairs, pedestrians, vehicles, etc., in various indoor and outdoor scenarios. We also integrated a Visual Language Model (VLM) for scene summarization (a lightweight image captioning model) which produced concise descriptions of the user's surroundings. While not perfect, these descriptions were often accurate (e.g. "a street with several parked cars on the right, and a person walking a dog") and provided the user with an overall context. This goes well beyond the capabilities of non-AI solutions and even beyond some commercial AI devices that might only announce individual objects. In comparison to a solution like OrCam (which excels at reading text and recognizing specific trained objects such as faces or products), our system's strength lies in general scene understanding. It may not read a page of text as quickly as OrCam (which is specifically optimized for text OCR), but it can detect a wider array of objects in real time and describe dynamic scenes, which OrCam does not do (OrCam would not, for example, tell a user that "a dog is approaching from the left" because it has no general object detection for arbitrary objects). Similarly, apps like Seeing AI can describe scenes, but they do so on a smartphone where the user must intentionally take a photo; our system does it continuously and hands-free [14].

It is worth noting that our object detection model was optimized for efficiency, so it may not match the absolute highest accuracy of heavier models (like those used in research benchmarks). For example, a full-fledged YOLOv11 or Faster R-CNN could achieve higher detection accuracy on difficult datasets, but they would run slower on edge hardware [40] . We chose a balanced approach to ensure real-time performance. The accuracy we achieved, which in our tests was sufficient to detect all obstacles large enough to pose a danger and to recognize the most important objects in the scene, represents a practical trade-off. Importantly, we saw no critical missed detections in controlled obstacle courses (i.e., the system did not fail to see large obstacles in the user's path). Some smaller or farther objects might occasionally be missed or misidentified, but those did not significantly impact navigation decisions. In future iterations, model improvements or additional sensor data could further boost this accuracy.

5.3 Resource and Cost Efficiency:

A key contribution of our work is demonstrating that such a complex assistive system can be deployed on a single compact device without exhausting its resources. Through our resource-aware design, the Jetson's CPU-GPU workload was carefully managed. We found that the device's CPU usage remained moderate (50–60%) and GPU usage around 40% on average during operation, as it does not run all the models simultaneously to keep low power use. The power draw of the system (including the Jetson, camera, and audio output) averaged around 10 W, which, with a decent portable battery of 2200mAh, can support a few hours of continuous usage. This is a promising result, though for all-day use, further power optimization or a larger battery would be needed. Compared to a smartphone-based solution, our device uses more power (smartphones are highly optimized for battery), but it also delivers much more processing locally. We consider the roughly 3–4 hour active use battery life reasonable for a prototype. In practice, the device could be used intermittently (e.g. turned on when navigating a route, turned off when sitting stationary) to extend effective usage. The cost of the system components is also worth noting: our prototype's hardware (Jetson Nano board, camera module, GPS, etc.) costs only a few hundred dollars in total, which is relatively low compared to specialized devices like Envision or OrCam (which can cost a few thousand dollars). This suggests that with economies of scale, a commercial version of our solution could be made quite affordable for users or subsidizing organizations.

5.4 Comparison with Other Systems:

In terms of functionality, our all-in-one system covers a broader feature set than most existing solutions, which include object detection, obstacle distance estimation, scene captioning, text reading and GPS location sharing. By contrast, sensor-based canes usually offer only obstacle distance feedback; smartphone apps like Seeing AI offer object, text, and scene recognition but no direct obstacle avoidance alerts (and no built-in safety communications); wearable devices like OrCam focus on text and face recognition; and services like Be My Eyes handle a wide range of tasks, but require human intervention and a network connection. Our system's unique value is in integrating all these aspects as the AI provides immediate autonomous assistance and the IoT connectivity provides backup human assistance and orientation (GPS) services. This integration was reflected in user testing feedback.

In performance comparisons, our system achieves state-of-the-art or near state-of-the-art results on key metrics for edge devices. For example, our object detection accuracy and speed were on par with the best results reported by Pydala et al. (2024) for their Visisense IoT cane. They reported a 17 ms processing time per frame with 99% accuracy in controlled tests, using a combination of cloud and edge computing. Our system, running fully on edge (except when using the optional remote assistance), reached a slightly higher per-frame latency (in the tens of milliseconds) with an accuracy sufficient for practical use, which is impressive considering we did not rely on cloud offloading. In energy consumption, Visisense had a very efficient profile (low energy per inference) by splitting workloads; our fully-edge approach uses more energy on the device, but this is the trade-off for being cloud-independent. This illustrates that edge AI has matured to the point where even complex tasks like image captioning and language understanding can be done locally, albeit with careful model selection. In fact, we experimented with various Large Language Models (LLMs) on the device (as part of an AI voice assistant feature for querying the system). We benchmarked smaller LLMs that could run on the Jetson and found that certain optimized models could answer questions reasonably well, but larger models were too slow or resource-heavy. The performance results (Table 2 in our study) show that while edge devices can host LLMs, there is a significant speed/accuracy trade-off among models. This is a cutting-edge aspect of our solution – few assistive devices have a built-in conversational AI. The inclusion of an LLM-driven voice assistant in our system was exploratory, but it points toward future directions where users might have an interactive AI guide in addition to automated alerts.

5.5 Practical Implementation Challenges:

Building and testing the system revealed several challenges that need consideration for real-world deployment. One major challenge is the interface and usability. We learned that just because the system can provide a lot of information doesn't mean it should do so all at once. If the device bombards the user with continuous speech (e.g. "car on left, tree ahead, person on right, sign overhead…" all in a stream), it can overload and confuse more than help. Finding the right balance of information is crucial. In our user trials, we tuned the verbosity via settings via an "exploration mode" which might describe the surroundings in more detail when the user is curious and query for assistance. This kind of context-aware output is important to make the device a helpful aid rather than a distraction. Another challenge is the audio interface: using a speaker is fine in quiet areas, but in noisy city streets, the user may prefer bone-conduction headphones or earbuds to clearly hear the device. We also must ensure that the device's audio alerts do not prevent the user from hearing important environmental sounds (like traffic or someone speaking to them). This remains an area for fine-tuning – for instance, using more directional audio cues (spatialized sound indicating where an obstacle is) could convey information in a less intrusive way than verbose descriptions.

Hardware-wise, there were challenges in form factor and weather proofing. Our prototype was a small package (the Jetson dev kit, a battery pack and a camera mounted at chest level). This is acceptable as a prototype, but for daily use it should be lighter and more discreet – possibly integrated into a harness, cane, or eyewear. Devices such as Envision glasses are extremely compact and comfortable to wear but are less featured. There is a trade-off between power and size; our device is more powerful but slightly larger. We also have to consider heat: the Jetson can get warm if it has to work hard for an extended period. We added a heat-sink and mini fan, which fixed the overheating problem but added a bit of fan noise (not ideal). Later versions can utilize the Jetson in a reduced power mode or another board to reduce heat. Making the camera and electronics weatherproof (e.g., rain and dust) is also a primary requirement.

5.6 Privacy and Security:

We designed the communication features with security in mind because privacy is a major concern for users in modern days. The idea of wearing a camera that is always analyzing the environment can raise questions, both for the user and for people around them, about how the data is used. By doing on-device processing, we ensure that raw video is not uploaded to any server for the AI functions. We adhered to guidelines to protect any personal data. For example, location data and live streams are sent to trusted contacts only who are connected to the private network with access that the user selects. In later versions, we might add face recognition for familiar individuals so the device could announce the person who's approaching, but that opens other privacy concerns (maintaining a secure face database, preventing misuse). For now, we limited our system to simple functionality that clearly has user consent.

Overall, the analysis of our findings concludes that the suggested IoT-edge approach overcomes numerous drawbacks of previous technologies by offering deeper environmental insight than sensor-only devices, quicker and more personal responses compared to cloud-reliant applications, as well as a more unified user experience than single purpose alternatives. It clearly shows that an AI-driven, portable assistant for the blind is viable with the edge computing technologies of today. The comparisons with existing solutions show that no single solution addresses all needs – each has its strengths and weaknesses – but our solution offers a nice compromise by encompassing a number of features and support modes. We have also identified areas for improvement (user interface, form factor, additional sensors for edge cases), which we will discuss next as future work. Ultimately, the positive feedback from evaluation users and the quantitative performance metrics give us confidence that this approach is a step forward in assistive technology, bringing us closer to a device that can function as a "digital guide dog" or co-pilot for blind and low-vision individuals in everyday life.

6 Future Work

Although the developed system demonstrates a very strong assistive solution, many directions for future work and improvement are open. We see potential hardware improvements, AI algorithm improvements, user interaction, and more integration that will further improve functionality and user experience in such kinds of assistive devices. Some of them are listed below

6.1 Hardware Miniaturization and Efficiency:

One immediate goal is to make the system lighter, smaller, and more power-efficient. The current prototype, though portable, can be refined into a more wearable form factor. Future versions might use smaller PCB integrations or system-on-module designs to reduce size as well as exploring alternative edge AI hardware, which is also worthwhile. Special low-power AI chips (NPUs) that are only just beginning to appear in smartphones and AR glasses can be used to move from hours of battery life to all-day operation. Additionally, battery technology and power management can be tweaked like, dynamic power scaling (where the phone can downclock or disable some parts during the time when the user is idle or environments are static) in order to save power. Advances in thermal management (passive cooling designs) will also play a role in removing the need for fans, thereby reducing noise and power draw. The final aim is to encase the core components in something like a pair of intelligent glasses or a small unit that can be clipped onto a cane or belt, making it effectively invisible and unobtrusive to the user.

Enhanced and New AI Models: On the software side, continual advancements in AI will allow us to expand and improve the device's functionalities. One area is object detection and recognition. We plan to train and incorporate more classes of objects, including specific categories relevant to blind users (e.g., detecting crosswalk indicators, road signs, elevator buttons, entrance doors versus windows, etc.). Improving the detection of non-rigid obstacles like overhanging branches or half-open doors could be achieved with advanced vision transformers or hybrid models that consider temporal information from video. Another promising avenue is image segmentation – instead of just bounding boxes, the device could segment the walkable area vs. the obstacle area in the view and convey that to the user (possibly through augmented audio or haptic feedback). For scene understanding, larger visual-language models (like the latest image captioning networks or multimodal transformers) could provide more detailed and human-like descriptions. As edge hardware improves, newer and bigger models might eventually run locally. Similarly, the conversational AI (LLM) component can be improved. In our prototype we used a relatively small model for the voice assistant. In the future, we could use larger models or even mulimodal LLM, effectively making the device a digital assistant as well as a navigator [41].

Another future AI feature is facial recognition of known people (friends, family) to alert the user when someone familiar is nearby. Reading emotional cues or body language (is anyone waving at me? is anyone sad or pleased?) could introduce a social factor into the assistance, though that's advanced-level. Also, text reading can be expanded: aside from OCR, we could have the system read signs and symbols (like logos, directional arrows, etc.) and read them back in kind ("the sign indicates an exit to your left"). All these developments in AI are towards making the device more context-sensitive and useful in more contexts.

6.2 Multimodal Sensing and Integration

On top of that, the use of other sensing modalities beyond the Intel RealSense D455 is also a possibility. While the D455 has the capability of stereo depth sensing up to 6 meters, the use of a higher-range stereo camera like the Stereolabs ZED 2, which is capable of detecting up to 20 meters, would significantly be able to provide long-distance obstacle detection [42]. This extended depth perception is particularly valuable for identifying crosswalks, poles, or approaching vehicles at a distance, allowing for earlier warnings and safer navigation. Additionally, A major improvement in scene captioning can be achieved by combining a stereo camera with Vision-Language Models (VLMs). For example, if the stereo camera detects an object, the VLM can classify it as a "traffic sign," "stationary barrier," or "approaching cyclist", refining navigation instructions for visually impaired users.

An additional critical enhancement is fall detection using an Inertial Measurement Unit (IMU). With accelerometer and gyroscope data analysis, the system is able to detect sudden orientation change, braking, or abnormal motion patterns, which could signify

a fall [43]. The feature can be integrated to initiate an automatic alert to emergency services or caregivers or give immediate audio feedback to the user, requesting him or her to confirm if assistance is needed, as well as log fall events for additional research to maximize user safety. For external navigation, integration of GPS and IMU information with mapping capabilities would enhance real-time turn-by-turn directions. GPS can be used by the system to determine user position, while the vision system confirms the surroundings for accuracy-e.g., ensuring the user is at an allotted crosswalk prior to proceeding.".

6.3 Improved User Interaction and HCI:

As we continue to build the device, a key focus is enhancing human-computer interaction. Instead of relying solely on speech output, we want to add stereo sound-based 3D localization with spatialized audio [44]. That will allow users to intuitively perceive obstacles- objects that are closer will be louder, with direction conveyed through binaural audio (e.g., a bicycle approaching from the right will be in the right ear). This affords continuous awareness with no requirement for ongoing verbal prompts. To further support responsiveness, haptic feedback will provide silent real-time notification in the form of a vibration belt, vest, or wearable sensors. The system will trigger vibrations on the same side (left-side vibration for an object to the left), resulting in a multimodal assistive system that is able to increase user confidence and safety.

We also wish to customize the user experience. Users vary in their needs and wants – some will desire complete running commentary, others minimal notification. Upcoming software will enable greater control over verbosity, voice quality, languages, and what types of objects to speak about. AI can even be taught based on the user's behavior; e.g., if the user always ignores "storefronts" announcements but always responds to "traffic approaching", the system can prioritize the latter. Another future feature is logging and analytics: the device can keep a personal record of obstacles overcome, places visited, etc., to aid in rehabilitation or training. For example, mobility trainers could review these logs to determine where a user is having the most trouble and adjust training accordingly. This, of course, would be done with the user's permission and with privacy controls (data could be stored locally or encrypted).

6.4 Integration with Smart Cities and IoT Infrastructure:

Looking beyond the individual device, there is great potential in integrating assistive devices with the broader smart city infrastructure. In the future, our device could communicate with traffic signals to know when the pedestrian light is green or receive data from connected vehicles, such as a public bus or tram, otherwise silent. It is possible to detect dedicated signage and seamlessly incorporate information to enable access to real-time transit data. Another concept is leveraging edge computing in the environment (sometimes called Mobile Edge or Fog computing). While our device is self-contained, there may be scenarios where offloading to a nearby edge server (like a 5G base station with computing) could provide a boost for heavy tasks. For example, a wearable might normally do everything on-board, but if it's in range of a trusted edge compute hotspot (say, provided at a public venue), it could temporarily use that to run a very large AI model for a harder task (like detailed scene reconstruction). This kind of dynamic IoT ecosystem could keep the wearable light while giving access to more power when available. Research on distributed computing for assistive devices (such as a cane that talks to a home server) has shown improvements in performance without sacrificing latency too much [45].

6.5 User Studies and Accessibility Research:

Future work will also involve rigorous field testing and user studies assisted by the blind community. We plan to do longitudinal studies wherein users can bring the device back to their home and integrate it into their daily routine for several weeks and then provide feedback to us as to what they find most valuable and what doesn't work. This field feedback is invaluable to detect usability flaws that short laboratory tests might not detect. We're also excited to learn more about how users typically interact with the device, i.e., are they constantly using it or just when they transition to another area or how quickly do they learn to trust its notifications or under what circumstances do they become confused or in the wrong? From that input, we can further develop both the hardware (ergonomics, wearability) and software (UI, feature set) [46].

7 Conclusion

This research successfully creates an edge computing and IoT-based real-time vision assistance system to improve mobility and independence for the visually impaired. Unlike traditional sensor-based systems, our solution integrates deep learning for scene awareness, real-time object detection, and remote assistance with low latency and privacy. Running processing on the device using Jetson Orin Nano does away with our reliance on the cloud. This makes the solution faster, more secure, and cheaper. Our system has proven very accurate according to the tests, and it uses power efficiently while being compatible with a great number of assistive features. Future enhancements include streamlining the user interface for better feedback, optimizing AI models to achieve faster real-time performance, and intermingling different types of data (e.g., LiDAR and audio) for improved environmental perception. The proposed IoT-Edge Vision Assistant represents a major advancement toward intelligent and autonomous vision assistance, offering a useful, versatile, and functional application to the visually impaired.

References

[1] Dada Emmanuel Gbenga, Arhyel Ibrahim Shani, and Adebimpe Lateef Adekunle. Smart Walking Stick for Visually Impaired People Using Ultrasonic Sensors and Arduino. *International Journal of Engineering and Technology*, 9(5):3435–3447, October 2017.

[2] Wafa Elmannai and Khaled Elleithy. Sensor-Based Assistive Devices for Visually-Impaired People: Current Status, Challenges, and Future Directions. *Sensors*, 17(3):565, March 2017. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[3] Joey Payne. Microsoft launches AI app for visually impaired. *UWIRE Text*, pages 1–1, July 2017. Publisher: ULOOP Inc.

[4] Xuan-Thanh-An Nguyen, Jan Koopman, Maria M. van Genderen, Henk L. M. Stam, and Camiel J. F. Boon. Artificial vision: the effectiveness of the OrCam in patients with advanced inherited retinal dystrophies. *Acta Ophthalmologica*, 100(4):e986–e993, June 2022.

[5] Md. Atikur Rahman and Muhammad Sheikh Sadi. IoT Enabled Automated Object Recognition for the Visually Impaired. *Computer Methods and Programs in Biomedicine Update*, 1:100015, January 2021.

[6] Jiaji Wang, Shuihua Wang, and Yudong Zhang. Artificial intelligence for visually impaired. *Displays*, 77:102391, April 2023.

[7] Amelia Ulfa, Ajeng A. Rosspertiwi, Alvin Sahroni, and Suatmi Murnani. S-Cane: Ultrasonic Sensor-Based Smart Cane for the Visually Impaired. In *2023 IEEE International Biomedical Instrumentation and Technology Conference (IBITeC)*, pages 7–11, November 2023.

[8] Shruti Dambhare. Smart stick for Blind: Obstacle Detection, Artificial vision and Real-time assistance via GPS. 2011.

[9] Olakanmi O. Oladayo. A Multidimensional Walking Aid for Visually Impaired Using Ultrasonic Sensors Network with Voice Guidance. *International Journal of Intelligent Systems and Applications*, 6(8):53.

[10] Mohd Helmy Abd Wahab, Amirul A. Talib, Herdawatie A. Kadir, Ayob Johari, A. Noraziah, Roslina M. Sidek, and Ariffin A. Mutalib. Smart Cane: Assistive Cane for Visually-impaired People, October 2011. arXiv:1110.5156 [cs].

[11] Mohammad Farid Saaid, Ismarani Ismail, and Mohd Zikrul Hakim Noor. Radio Frequency Identification Walking Stick (RFIWS): A device for the blind. In *2009 5th International Colloquium on Signal Processing & Its Applications*, pages 250–253, March 2009.

[12] Anna M. Joseph, Azadeh Kian, and Rezaul Begg. State-of-the-Art Review on Wearable Obstacle Detection Systems Developed for Assistive Technologies and Footwear. *Sensors*, 23(5):2802, January 2023. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[13] Dushyant Kumar Singh. Dynamic Environment Exploration Tool: A Blind's Eye. 6, 2015.

[14] Christina Granquist, Susan Y. Sun, Sandra R. Montezuma, Tu M. Tran, Rachel Gage, and Gordon E. Legge. Evaluation and Comparison of Artificial Intelligence Vision Aids: Orcam MyEye 1 and Seeing AI. *Journal of Visual Impairment & Blindness*, 115(4):277–285, July 2021. Publisher: SAGE Publications Inc.

[15] Ruxandra Tapu, Bogdan Mocanu, and Titus Zaharia. DEEP-SEE: Joint Object Detection, Tracking and Recognition with Application to Visually Impaired Navigational Assistance. *Sensors*, 17(11):2473, November 2017. Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[16] Vidya N. Murali and James M. Coughlan. Smartphone-based crosswalk detection and localization for visually impaired pedestrians. In *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–7, July 2013.

[17] Bhasha Pydala, T. Pavan Kumar, and K. Khaja Baseer. VisiSense: A Comprehensive IOT-based Assistive Technology System for Enhanced Navigation Support for the Visually Impaired. *Scalable Computing: Practice and Experience*, 25(2):1134–1151, February 2024. Number: 2.

[18] Muhamad Risqi Utama Saputra, Widyawan, and Paulus Insap Santosa. Obstacle Avoidance for Visually Impaired Using Auto-Adaptive Thresholding on Kinect's Depth Image. In *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*, pages 337–342, December 2014.

[19] Sibi C. Sethuraman, Gaurav R. Tadkapally, Saraju P. Mohanty, Gautam Galada, and Anitha Subramanian. MagicEye: An Intelligent Wearable Towards Independent Living of Visually Impaired, March 2023. arXiv:2303.13863 [cs].

[20] Ethan Waisberg, Joshua Ong, Mouayad Masalkhi, Nasif Zaman, Prithul Sarker, Andrew G. Lee, and Alireza Tavakkoli. Meta smart glasses—large language models and the future for assistive glasses for individuals with vision impairments. *Eye*, 38(6):1036–1038, April 2024. Publisher: Nature Publishing Group.

[21] Mauro Avila, Katrin Wolf, Anke Brock, and Niels Henze. Remote Assistance for Blind Users in Daily Life: A Survey about Be My Eyes. In *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '16, pages 1–2, New York, NY, USA, June 2016.

[22] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646, October 2016. Conference Name: IEEE Internet of Things Journal.

[23] Agathe Archet, Nicolas Gac, François Orieux, and Nicolas Ventroux. Embedded AI performances of Nvidia's Jetson Orin SoC series. In *17ème Colloque National du GDR SOC2*, Lyon, France, June 2023.

[24] Eric Gamess and Sergio Hernandez. Performance Evaluation of Different Raspberry Pi Models for a Broad Spectrum of Interests. *Research, Publications & Creative Work*, January 2022.

[25] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust Speech Recognition via Large-Scale Weak Supervision, December 2022. arXiv:2212.04356 [eess].

[26] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, April 2021. arXiv:2005.11401 [cs].

[27] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, and Maria Lomeli. The faiss library. 2024.

[28] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[29] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks, March 2019. arXiv:1801.04381 [cs].

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context, February 2015. arXiv:1405.0312.

[31] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. Publisher: Nature Publishing Group.

[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. volume 9905, pages 21–37. 2016. arXiv:1512.02325 [cs].

[33] Eunjin Jeong, Jangryul Kim, and Soonhoi Ha. TensorRT-Based Framework and Optimization Methodology for Deep Learning Inference on Jetson Boards. *ACM Trans. Embed. Comput. Syst.*, 21(5):51:1–51:26, October 2022.

[34] Florian Bordes, Richard Yuanzhe Pang, Anurag Ajay, Alexander C. Li, Adrien Bardes, Suzanne Petryk, Oscar Mañas, Zhiqiu Lin, Anas Mahmoud, Bargav Jayaraman, Mark Ibrahim, Melissa Hall, Yunyang Xiong, Jonathan Lebensold, Candace Ross, Srihari Jayakumar, Chuan Guo, Diane Bouchacourt, Haider Al-Tahan, Karthik Padthe, Vasu Sharma, Hu Xu, Xiaoqing Ellen Tan, Megan Richards, Samuel Lavoie, Pietro Astolfi, Reyhane Askari Hemmat, Jun Chen, Kushal Tirumala, Rim Assouel, Mazda Moayeri, Arjang Talattof, Kamalika Chaudhuri, Zechun Liu, Xilun Chen, Quentin Garrido, Karen Ullrich, Aishwarya Agrawal, Kate Saenko, Asli Celikyilmaz, and Vikas Chandra. An Introduction to Vision-Language Modeling, May 2024. arXiv:2405.17247 [cs].

[35] Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, Yuming Lou, Shang Yang, Haocheng Xi, Shiyi Cao, Yuxian Gu, Dacheng Li, Xiuyu Li, Yunhao Fang, Yukang Chen, Cheng-Yu Hsieh, De-An Huang, An-Chieh Cheng, Vishwesh Nath, Jinyi Hu, Sifei Liu, Ranjay Krishna, Daguang Xu, Xiaolong Wang, Pavlo Molchanov, Jan Kautz, Hongxu Yin, Song Han, and Yao Lu. NVILA: Efficient Frontier Visual Language Models, March 2025. arXiv:2412.04468 [cs].

[36] Silvio Quincozes, Tubino Emilio, and Juliano Kazienko. MQTT Protocol: Fundamentals, Tools and Future Directions. *IEEE Latin America Transactions*, 17(09):1439–1448, September 2019.

[37] Miguel Grinberg. *Flask Web Development*. "O'Reilly Media, Inc.", March 2018. Google-Books-ID: cVlPDwAAQBAJ.

[38] Paul Crickard III. *Leaflet.js Essentials*. Packt Publishing Ltd, August 2014. Google-Books-ID: A65OBAAAQBAJ.

[39] Branislav Sredojev, Dragan Samardzija, and Dragan Posarac. WebRTC technology overview and signaling solution design and implementation. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1006–1009, May 2015.

[40] Dalmar Dakari Aboyomi and Cleo Daniel. A Comparative Analysis of Modern Object Detection Algorithms: YOLO vs. SSD vs. Faster R-CNN. *ITEJ (Information Technology Engineering Journals)*, 8(2):96–106, December 2023. Number: 2.

[41] Musashi Hinck, Matthew L. Olson, David Cobbley, Shao-Yen Tseng, and Vasudev Lal. LLaVA-Gemma: Accelerating Multimodal Foundation Models with a Compact Language Model, June 2024. arXiv:2404.01331 [cs].

[42] Ahmed Abdelsalam, Mostafa Mansour, Jari Porras, and Ari Happonen. Depth accuracy analysis of the ZED 2i Stereo Camera in an indoor Environment. *Robotics and Autonomous Systems*, 179:104753, September 2024.

[43] Chris Cheng Zhang, Chenran Wang, Xinrui Dai, and Sicheng Liu. Camera-Based Analysis of Human Pose for Fall Detection. In *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, pages 1779–1782, July 2023.

[44] Wen-Chih Wu, Cheng-Hsun Hsieh, Hsin-Chieh Huang, Oscal T.-C. Chen, and Yu-Jen Fang. Hearing aid system with 3D sound localization. In *TENCON 2007 - 2007 IEEE Region 10 Conference*, pages 1–4, October 2007. ISSN: 2159-3450.

[45] Manoel José Júnior, Orlewilson B. Maia, Horácio Oliveira, Eduardo Souto, and Raimundo Barreto. Assistive Technology through Internet of Things and Edge Computing. In *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, pages 330–332, September 2019. ISSN: 2166-6822.

[46] Michele C. McDonnall, Anne Steverson, and Jamie Boydstun. Actual and Preferred Methods for Learning to Use Assistive Technology. *Assistive technology outcomes and benefits*, 2024:20–35, 2024.