Audo-Sight: Assistive Conversational Perception through Collaborative Edge-Cloud Intelligence

Jacob Bradshaw
jacob.bradshaw@unt.edu
University of North Texas College of Engineering
Denton Texas USA

Mohsen Riahi Alam
mohsen.riahialam@unt.edu
University of North Texas College of Engineering
Denton Texas USA

Bhanuja Ainary
bhanujaainary@my.unt.edu
University of North Texas College of Engineering
Denton Texas USA

Minseo Kim
minseo.kim@unt.edu
University of North Texas College of Engineering
Denton Texas USA

Mohsen Amini Salehi
mohsen.amini@unt.edu
University of North Texas College of Engineering
Denton Texas USA

Abstract

Despite advances in assistive technologies, Blind and Low-Vision (BLV) individuals continue to face challenges in understanding their surroundings. Delivering concise, useful, and timely scene descriptions for ambient perception remains a long-standing problem in accessibility. Existing solutions often fail to identify user expectations for real-time and accessible responses. Moreover, for a given task, they either rely on cloud offloading, which imposes a significant delay, or edge-based AI, which often sacrifices accuracy. To address this, we present Audo-Sight, an AI-driven assistive system that spans across Edge-Cloud continuum and enables BLV individuals to perceive their surroundings through voice-based conversation. Audo-Sight provides low-latency, accurate, and human-friendly responses through a novel mechanism that seamlessly fuses Edge and Cloud responses. The system also addresses challenges in catering to BLV users through response editing informed by BLV needs. Audo-Sight orchestrates a set of AI models based on user query contextual analysis to infer intent and adjust for a variety of situations. In urgent cases where users require fast responses, Audo-Sight leverages parallel Edge and Cloud pipelines and seamlessly combines responses through its Response Fusion Engine. Systematic evaluation shows that Audo-Sight delivers speech output around 80% faster for urgent tasks and generates complete responses approximately 50% faster across all tasks compared to a commercial cloud-based solution—highlighting the need for customized AI-based solutions. Human evaluation of Audo-Sight shows that it is the preferred choice over GPT-5 for 62% of BLV participants with another 23% stating both perform comparably. Speed and interruption evaluations demonstrate that in most situations, the system can seamlessly respond at a rapid pace to keep up with BLV expectations.

SIGACCESS, Porto, Portugal

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

CCS Concepts

• **Human-centered computing** → **Accessibility systems and tools**; • **Computing methodologies** → *Natural language generation*; • **Networks** → *Cloud computing*.

Keywords

Multimodal Large Language Models (MLLMs), AI Models, Edge-Cloud continuum, Context-Aware AI.

ACM Reference Format:
Jacob Bradshaw, Mohsen Riahi Alam, Bhanuja Ainary, Minseo Kim, and Mohsen Amini Salehi. 2018. Audo-Sight: Assistive Conversational Perception through Collaborative Edge-Cloud Intelligence. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (SIGACCESS).* ACM, New York, NY, USA, 16 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

1.1 Overview

Assistive technologies aim to help Blind and Low-vision (BLV) individuals gain ambient perception; thus, they can perform their daily activities more efficiently and, in turn, can improve their independence and social inclusion. Despite remarkable advances in AI (particularly LLMs) and computing systems in recent years, their responses are still not timely and accurate enough for BLV individuals in several situation; hence, effective ambient perception continues to be a hurdle for people with BLV [6, 21].

Effective ambient perception relies on back-and-forth interaction between users and assistive technologies to promptly analyze surroundings during daily tasks. For example, consider a BLV user ordering food in a restaurant without time pressure: the user must locate the counter, explore the menu, place an order, and complete payment. In contrast, a BLV traveler running late at an airport faces a far more urgent scenario, requiring rapid navigation to the correct terminal, identification of the departure gate, and timely completion of boarding procedures. A powerful conversational system to completely understand the user situation and awareness of surroundings through different AI models and sensors are required in the assistive technologies. Recent advances in collaborative intelligence across the edge-cloud continuum offer promising opportunities to realize active ambient perception for people with

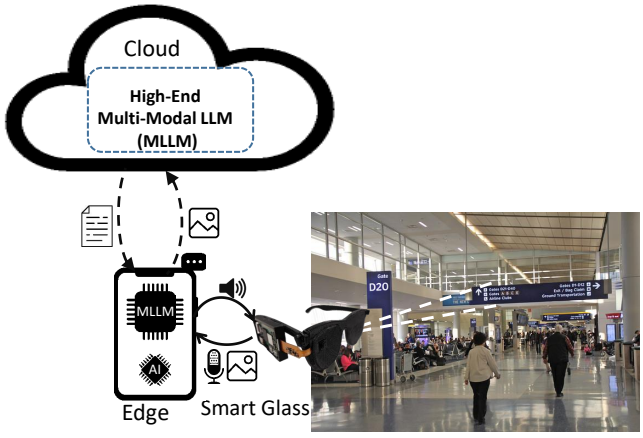

Figure 1: Overview of the Audo-Sight framework that can provide conversational ambient perception for BLV individuals.

BLV. These include the multi-agent AI systems for tasks such as face recognition and object detection[18, 27], the integration of multimodal large language models (MLLMs) for vision-language reasoning [10], and the progress in wearable assistive technologies across edge-cloud [2, 37]. In the software domain, all of these systems utilize multiple AI agents, and their performance depends on the performance of each individual agent as well as how they are managed within the edge–cloud continuum infrastructure.

Accordingly, the goal of this study is to investigate the challenges inherent in integrating AI models and MLLMs on the edge–cloud continuum to enable a timely, accurate, context-aware, human-centric framework for BLV users. We develop Audo-Sight, an AI-driven assistive framework for BLVs based on the smartglasses shown in Figure 1. The smartglasses are paired with an AI-enabled mobile edge system and a cloud backend. User queries about the surrounding environment are conveyed to Audo-Sight to be processed across edge-cloud and deliver context-aware and accurate responses while minimizing end-to-end latency. Within this framework, multiple AI models are deployed to address user intent, enabling the system to meet users' needs across a range of everyday scenarios, particularly in time-sensitive or urgent situations. This framework also fuses and refines outputs from multiple MLLMs to make responses more aligned with BLV users' requirements.

Efficient integration of multi-agent AI systems and MLLMs to assist BLV users presents several challenges, including selecting the appropriate AI models and tiers for deployment, determining how to invoke them based on user intent and the situation, and, when multiple AI models are required for a task, ensuring efficient coordination of the models and fusion of their outputs.

1.2 Problem Definition

Selecting appropriate AI models, LLMs, and their tiers along the edge–cloud continuum for deployment presents a key challenge, as these choices must be closely aligned with the specific needs of BLV users. For instance, both conversational intelligence and vision-based capabilities are essential components of such systems, and they can be deployed on edge devices, in the cloud, or across both, with each approach introducing distinct trade-offs in terms of latency, accuracy, availability, energy consumption, cloud performance jitter and cost [38].

Another key challenge is selecting the most appropriate AI models based on user intent and context. An assistive system must automatically identify and employ the most relevant AI models to generate an effective response. This process depends on accurate interpretation of the user query and an understanding of user intent. To enable this functionality, a lightweight and low-latency AI model must be deployed at the edge to infer the user's intent and contextual information associated with each query and to route the request to one or more appropriate AI models or LLMs for processing.

Based on semi-structured interviews with BLV users of assistive technologies, we found that an effective conversational ambient perception system must account for *urgent* situations. For example, consider the time-sensitive nature of responding to a traveler's query about the "direction to the departure gate" at an airport, compared to a query about "the ingredients and calorie content of a menu item" at a restaurant. In the former case, the user prefers a concise and rapid response, whereas in the latter, the user expects a comprehensive and highly accurate response. Existing assistive tools often generate lengthy voice-based responses containing unnecessary details, which are difficult to track due to the sequential nature of voice [8], and can cause noticeable delays in user decisions that sometimes leading to serious consequences[34], such as accidents. To address this challenge, similar to the solution for the previous challenge, an edge-deployed AI model, can be integrated into assistive systems to distinguish, based on context, between queries that demand urgent, minimal, and fast responses and those that require more comprehensive answers.

The third challenge arises in certain scenarios, when multiple AI models across edge and cloud may generate responses to the same user query with varying levels of accuracy and latency. For instance, a smaller and faster MLLM deployed at the edge and a larger MLLM hosted on the cloud may be jointly invoked for tasks such as scene description. Their responses may be either consistent or contradictory. This necessitates the development of a mechanism to effectively fusing the outputs of these models, ensuring both low-latency responses and high-accuracy results for the user. Accordingly, this raises the question of how to seamlessly fuse the high-quality responses generated by cloud-based AI model with low-latency responses produced by edge-based AI model.

Interviews with BLV users further indicated that two design factors related to audio feedback can improve performance and user experience in the use of assistive devices. The first factor is how quickly the device initiates audio output. Minimizing latency is essential, as users expect audio feedback to begin as soon as possible to reduce perceived waiting time. A potential solution is for assistive technologies to initiate audio output as early as possible using a lightweight model that produces brief, less detailed information, and then continue with a more capable model that generates more accurate and comprehensive speech. However, implementing such a strategy is non-trivial. For instance, when outputs from the two models are consistent, the system must avoid redundant or repetitive content, whereas in cases of conflicting responses, it must

ensure that the final output resolves discrepancies and preserves correctness.

The second is how the device can support user control of different speech rates in its audio feedback. Voice responses have a sequential nature, and BLV users typically have a much higher auditory perception than others. Therefore, it is very important for them to be able to increase the speech production rate, sometimes up to twice the normal speed or more. This effectively expands the bandwidth for information intake, allowing them to stay more quickly informed about their surrounding environment. Consequently, this feature should be considered in the design of assistive technologies, both during AI models selection and response generation and refinement, ensuring that speech synthesis is not compromised.

1.3 Contributions

In this study, we investigate these challenges and enable BLV users to effectively perceive their environment through Audo-Sight and its conversational interactions. To mimic the perception abilities of sighted people, Audo-Sight classifies incoming queries as either *urgent* or non-urgent (i.e., *normal*) and provides fast-track and normal-track pathways to support latency-driven and accuracy-driven interactions, respectively.

At the heart of Audo-Sight, there is a "Cognition Module" that includes a set of expert AI models and MLLMs deployed within both edge and cloud, working in tandem to achieve a high accuracy and low latency perception reliably (robust against transient edge cloud disconnections). Model selection was performed based on BLV user requirements in different environments and the available edge capacity. Cognition Module equipped with methods that dynamically employs the underlying AI models to balance the responsiveness of edge with the high-accuracy offered by high-end processing on the cloud, thereby, minimizing the latency without compromising accuracy for the urgent tasks, and maximizing accuracy for the non-urgent ones.

For urgent tasks, we develop a mechanism that seamlessly fuses the rapid preliminary response from edge with a more accurate but slower response from the cloud to provide timely and reliable response to the user. This is very much like when a person responds quickly about an initial thought, while in the background of his/her mind prepares a more accurate response. Lastly, the generated responses must be adapted based on the perception capabilities of the users from the environment. As such, Audo-Sight is equipped with a method that can assure blind-friendliness of the generated responses.

We have developed a prototype of the entire Audo-Sight framework (including the smartglasses and the software platform across tiers), and curated a dataset of day-to-day images (taken by BLV users) and their pertinent urgent and non-urgent queries. These artifacts will be revealed upon acceptance of the paper. To conduct a comprehensive evaluation of Audo-Sight, we conducted both system-level performance assessments and a human-study using BLV participants.

In sum, the key contributions of this study are as follows:

- We developed Audo-Sight as an assistive framework that uses edge–cloud AI solutions to provide timely, accurate, and context-aware perception through conversational interaction with BLV individuals. Audo-Sight distinguishes between different queries and routes them accordingly to balance accuracy and latency requirements.
- We propose "Cognition Module" as the reasoning engine of Audo-Sight that can not only distinguish between urgent and normal queries but also select and utilize the most appropriate AI models across edge-cloud to accurately and swiftly meet user needs.
- We develop a seamless edge–cloud response fusion mechanism that initiates audio output as early as possible by dynamically merging fast edge responses with more accurate cloud-generated responses. This mechanism enables both real-time and accurate assistance for users.
- We examined Audo-Sight with BLV human subjects and analyzed the accuracy, latency, and blind-Friendliness (accessibility) of the generated responses.

While Audo-Sight is developed in the context of BLV users and assistive technology, many of its challenges and solutions are common across many other edge-cloud-AI systems and they can be adapted to be used for them. Specifically, this study addresses the broader challenges related to: The role of edge relative to cloud; Achieving accuracy and low latency in AI Fusion across edge and cloud; And transient disconnected operation of the edge. The rest of this paper is organized as follows: Section II reviews the relevant literature, while Section III describes the proposed solution and methodology of Audo-Sight. Section IV presents the experimental evaluation and BLV human study. Finally, Section V concludes the paper with a summary of the key findings.

2 Related Work

Nowadays, many assistive systems for BLV individuals leverage multiple AI models that work together across various platforms to address users' needs [9, 35]. As mentioned in the introduction, many challenges in this area arise from how to achieve more efficient integration of AI models and better coordination among them. Therefore, we first review prior work on the integration of AI models that address problems closely related to our own, with a focus on approaches that tackle similar challenges. Subsequently, to ensure that our proposed method aligns with the needs of BLV users, we examine domain-specific studies in assistive technologies, highlighting how these works have approached and addressed challenges in BLV-assistive system design. Previous studies have emphasized the importance of selecting appropriate AI models for accessibility-focused applications. In particular, several works suggest that leveraging powerful cloud-based LLMs [5, 9, 10, 26] can significantly enhance system performance. For instance, Cheema et al. [10] proposed an audio description system that directly prompts GPT-4V[1] to generate detailed audio descriptions tailored for BLV users. NaviSense [35] employs two powerful cloud-based models, an LLM for voice interaction, GPT-4o-mini [29] and MoonDream 2B [3] Vision Language Models (VLM) for open-vocabulary object detection. WorldScribe[9] generates rich, hierarchical descriptions (general overviews to fine-grained details) using cloud-based GPT-4V [1] to support BLV scene understanding. Compared to the cloud-based BLV assistance systems, we also use a cloud-based LLM, but

our focus is on reducing response latency and minimizing waiting times in urgent situations via expert models and edge MLLM.

Alongside using powerful cloud-based AI models, assistive technologies also utilize some smaller AI models deployed on edge devices. Almost all of them incorporate speech-to-text (STT) and text-to-speech (TTS) models to support conversational interaction with BLV users. WorldScribe [9] leverages YOLO-World [13] on the edge to perform real-time object detection, producing word-level labels to support immediate scene understanding. Several assistive technologies[19][4] also deploy small on-device AI models for tasks such as document recognition, basic OCR, currency recognition, and simple light or color detection. These capabilities support offline operation and ensure low-latency responses to meet the real-time needs of BLV users. For urgent generic queries, unlike existing edge-based AI systems (e.g., [16, 18, 27]) that target offline operation or low-latency inference, our approach leverages edge MLLM as part of a coordinated latency-reduction strategy that combines edge and cloud responses to reduce response time while maintaining cloud-level response quality.

BLV users have diverse needs that require ensembling AI models to improve service quality and performance. However, using model ensembles introduces a routing challenge before inference: selecting the most suitable models for each query without running all models, to balance accuracy, latency, and efficiency. Existing approaches can be divided into training-free routers and training-required routers[11]. LLM routing [17, 23, 28] uses a router to assign each query to the most suitable model, directing simple queries to smaller LLMs and complex ones to larger models to balance quality and cost. RouterDC [12] introduces a learned router that selects the most suitable LLM for each query. It uses two specialized learning techniques called contrastive losses to match queries with models and to capture query similarities. Results show that it outperforms existing routing approaches, with strong generalization to unseen data.

After the inference phase, if multiple AI models have been invoked, how their outputs are utilized becomes important. In some applications, a more accurate answer can be obtained by leveraging multiple responses from different models. This is achieved by sending a user query to multiple AI models and subsequently merging their outputs to produce a final response. In [24], a fusion module is proposed to combine multiple responses by leveraging their strengths and mitigating their weaknesses, formulated as a sequence-to-sequence task. Following this approach, the authors employ Flan-T5-XL [14], a 3B-parameter model, as the fusion component due to its strong performance and relatively compact size. In this work, we evaluate this fusion-based approach against our proposed fusion method under the same settings.

A staged response strategy is proposed in [9] to begin generating audio as quickly as possible by using a multi-model description pipeline that sends scene frames to three AI models with different latency and granularity levels. It streams outputs sequentially, first returning faster object-level labels, followed by short descriptions with spatial relationships, and finally detailed visual descriptions from the strongest model. The system does not fuse, merge or select outputs, but presents them in order as they become available to the user. Compared with the strategy proposed in [9], our approach places greater emphasis on simultaneously minimizing response latency and maintaining high response quality, particularly in urgent scenarios. The two approaches differ in two key aspects: *First*, our method selects models based on the urgency and context of the user's query, whereas WorldScribe [9] dynamically switches between models according to frame stability and user hand movements. While effective for its target application, this strategy cannot efficiently handle urgent queries. *Second*, WorldScribe presents the outputs of multiple models directly to the user, which can result in redundant or inconsistent information. In contrast, our approach incorporates a fusion engine that identifies and removes duplicate content while resolving conflicts between model outputs during the generation of the final response. Moreover, when responses from smaller or weaker LLMs are incomplete or inaccurate, outputs from stronger LLMs can compensate for these limitations, producing more reliable and comprehensive responses.

The authors in [33] proposed Multitier AI fusion that enables combining results from concurrent execution across multiple tiers to achieve both low latency and high accuracy. Efficiency is achieved by running lightweight models on device, eliminating network overhead, while higher tiers improve accuracy using larger models and additional knowledge not available at the edge, including aggregated data from other devices and inferred or archived information that cannot be deployed to edge devices or restricted data. In latency-sensitive settings, immediate use of device outputs may be preferred over waiting for higher-tier results. Compared with Multitier AI Fusion [33], which employs a higher-tier model to aggregate information from lower-tier models, our approach relies on the cloud-based LLM as the primary source of accurate responses due to its superior reasoning capabilities, while leveraging edge-generated responses to reduce response latency.

Overall, the contribution of our work is to offer an efficient integration and coordination method across multiple AI models to support urgency and low-latency response time simultaneously.

3 Audo-Sight: A Multi-Modal Smart Assistive Technology across Edge-Cloud

3.1 Audo-Sight Architecture

The bird-eye view of the Audo-Sight architecture is shown in Figure 2. The input device in Audo-Sight is a custom-designed pair of smart-glasses that we developed to capture frames from the surrounding environment along with the user's voice query. Upon receiving the voice-based query, it is converted to text (denoted as Q) and paired with the pertinent frame (denoted as f) in the Query-to-Image mapper of the Input Management module. The urgency of the user's query can be identified either from the voice prosody or the text version of it. The (Q, f) pair is then forwarded to the Cognition Module of Audo-Sight.

The Cognition Module, one of the main contributions of this study, acts like the brain of Audo-Sight. At its kernel, it contains the Reasoning Engine, which consists of a set of lightweight *expert* (i.e., specialized) AI models (e.g., Optical Character Recognition (OCR), facial recognition, and object detection) to handle specific queries, and heavyweight MLLMs to handle more general queries. The MLLMs are specifically spread across edge and cloud for two reasons: (a) dealing with the resource and energy limitations of the edge; and (b) assuring availability (reliability) of the interaction

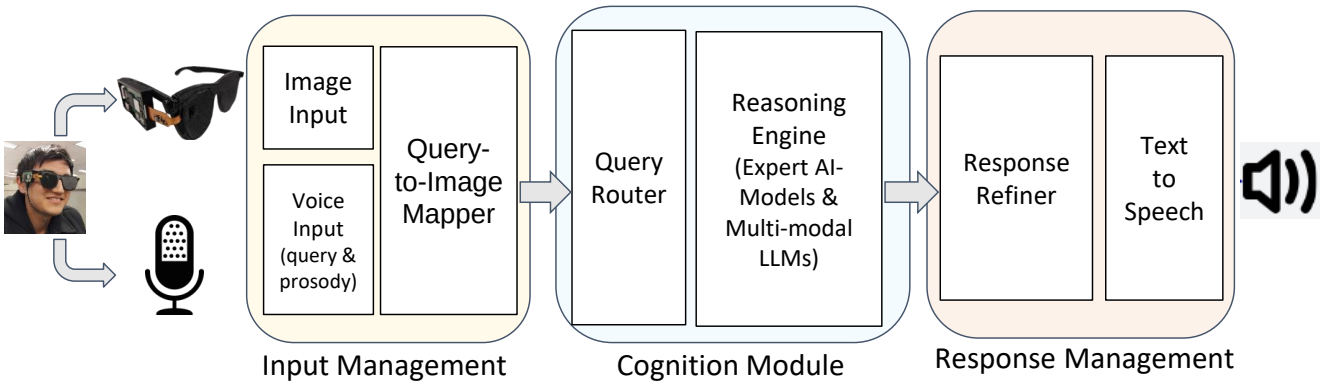

Figure 2: A bird-eye view of the Audo-Sight architecture and its primary components

even in the face of network failures and cloud inaccessibility. Upon receiving (Q, f) pair, the query routing mechanism first detects the urgency of the query and accordingly routes it to the appropriate subsystems with the best-matching AI model in the Reasoning Engine to handle the query based on the given visual context f.

Once a response is generated by the Reasoning Engine, it is further processed and refined by the Response Management module before speech synthesis. In analogy, this module is very much like the cognitive control of our brain that evaluates our thoughts before speaking them. The response refining process can include a wide range of actions. It can customize the raw response generated by the Reasoning Engine to become human-friendly and even blind-friendly via removing or altering the response to ensure effective communication with the user. Importantly, one aspect of Response Refiner, called Response Fusion Engine (described in 3.4.2), is in charge of seamlessly fusing responses generated by both edge and cloud tiers for the same query. This ensures the communication of a fast response (generated by edge) and an accurate response (generated by cloud).

3.2 Input Management Module

The smartglasses used for Audo-Sight are 3D printed frames containing a Raspberry Pi Zero 2 W with a camera. The smartglasses device is connected to the Audo-Sight edge computing platform via WiFi or USB. When the user begins a query, triggered by an input button event, an image frame (f) is captured from the glasses video stream and transferred to the edge system via TCP. The frame, f, from the start of the interaction is analyzed along with the user's voice recording. The textual query, Q, is extracted from the voice input using the Whisper STT engine [31]. These two pieces of input data are paired together in the Query-to-Image Mapper component and sent to the Cognition Engine. Our evaluation focuses on the Audo-Sight edge-cloud processing using pre-captured images and datasets rather than the smartglasses.

3.3 Cognition Module

The Cognition Module consists of three primary components, shown in Figure 3. Upon receiving query Q, first, the edge-based *Urgency Detector* determines whether a user requires a rapid response and accommodates this by activating the "urgent track" subsystem in the Reasoning Engine and Response Management that offer faster response times. Second, the edge-based *AI Router* infers the intent underlying the user's query, enabling the selection of task-appropriate AI models through routing. Finally, the *Reasoning Engine* comprises two sub-parts, the Expert AI Models (on the edge) and the Generic AI Models (on both edge and cloud) that together provide access to a range of generic and specialized AI models to efficiently generate accurate responses. In the following sections, we provide a detailed explanation of the methods developed for each one of these components.

3.3.1 Urgency Detector. To craft an accurate urgency detection system from the human language (i.e., the user queries), we fine-tuned all-MiniLM-L6-v2 [32] (aka MiniLM) sentence-transformers model using the SetFit framework [36]. We employed MiniLM because of its lightweight and edge-friendly design. We examined it against another popular model, called RoBERTa [25], that is used for similar purposes. Our evaluations showed that the F1 scores of MiniLM were slightly more than RoBERTa (0.94 vs 0.93). However, MiniLM far outpaced RoBERTa in terms of speed, with MiniLM processing queries more than twice as fast on the GPU and 4 times as fast on the CPU, which proves its edge-friendliness.

To further increase the speed of MiniLM, we employed SetFit to tune the model based on an urgency training dataset that we crafted to train the MiniLM model for urgent queries. With the tuning, we could substantially reduce the urgency detection overhead from 30 milliseconds to only 9 milliseconds. The training dataset was created using requests from VizWiz dataset [22] for the non-urgent (normal) class, and altered versions with urgent phrasing to create an urgent classification category. Although only the urgent class is used for classification, including a normal category helps differentiate normal requests from urgent ones due to the contrastive learning technique employed by SetFit. The model is trained offline, and the resulting urgency detection model is loaded once at runtime on the edge to evaluate every query. A threshold value is set for the urgency category, such that a query is considered urgent if its estimated probability for the urgent class exceeds this threshold. Based on the initial evaluations of the system, we chose to set the urgency threshold to 0.3 in the experiments.

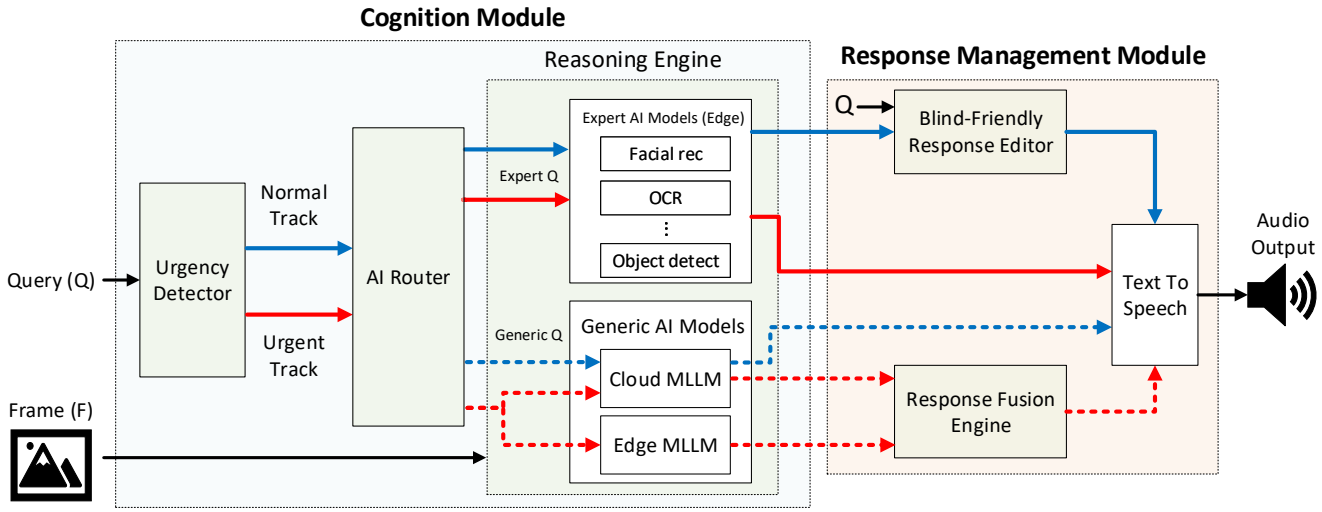

Figure 3: Internal mechanics of the Cognition and Response Management Modules of the Audo-Sight platform

3.3.2 AI Router. The AI Router conducts a secondary analysis of incoming queries following the urgency detector, in order to determine and route the (Q, f) pair to the appropriate AI model within the Reasoning Engine. For decision-making, the AI Router employs a fine-tuned all-MiniLM-L6-v2 model; the same model is utilized in the Urgency Detector. We argue that although using transformer-based methods generally implies more overhead in the system design, however, it avoids developing and deploying multiple solutions (e.g., one for urgency detector and one for AI routing). Hence, it can simplify the system design and maintenance.

The MiniLM model in the AI Router classifies queries into three categories: Face Recognition, OCR, and Object Detection that we have so far implemented for the Reasoning Engine. The probability estimate for the highest-scoring category is compared against a predefined threshold value (we set it to 0.86 based on our initial evaluations), and the corresponding route to Expert AI models is selected if this threshold is exceeded. The queries that do not meet the threshold are classified as generic request and are routed to the Generic AI models in the Reasoning Engine.

In summary, as shown in Figure 3, for query Q, Urgency Detector and AI Router work in tandem to route it through one of the following pipelines: Normal-track Expert, Urgent-track Expert, Normal-track Generic, and Urgent-track Generic.

3.3.3 Reasoning Engine. The Reasoning Engine comprises two subparts: the *Expert AI Models* and the *Generic AI Models*. The Expert AI Models are orders of magnitude faster than MLLMs and are responsible for generating response content from image analysis. In the Urgent-track, the generated response is directly delivered to the user; whereas, in the Normal-track, the response is sent to the Response Editor (in the Response Management module) for further processing and making it more blind-friendly.

Within the Generic AI Models part, both the cloud-based MLLM GPT-5 [30] and the edge-based MLLM Gemma3:4B [20] are employed to process the assigned (Q, f) pairs and handle general queries (i.e., those which cannot be handled by Expert AI Models). As there is no urgency associated with Normal-track Generic queries, they are only routed to the high-accuracy MLLM on the cloud. In addition to improving response accuracy, this preserves the edge battery lifetime and in case of a transient network failure, the user can resubmit the query (much like a best-effort request). The queries in the Urgent-track, however, are routed to both the edge and cloud MLLMs to be processed more quickly and reliably (assuring a response in case of network unavailability). Because in this case two responses are generated for the same urgent query, there must be a subsequent method, called Response Fusion Engine in the Response Management Module, to seamlessly combine the high quality cloud response with the fast paced edge response.

3.4 Response Management Module

The Response Management serves as an interface between the Cognition Module and the user, refining the output responses from different AI models in each track category to suit BLV users. It delivers responses to users and includes Blind-Friendly Response Editor, Response Fusion Engine and a TTS component to synthesize voice output. BLV users typically prefer faster speech output rates relative to sighted users, so this module allows users to adjust the rate of speech production. This speed increment must not cause interruptions when the system is speaking and therefore requires careful consideration based on the AI models' token generation rates and capabilities.

As illustrated in Figure 3, the urgent responses generated by the Expert AI Models and normal responses generated by the Generic AI Models are transmitted directly to the TTS module. In contrast, normal responses produced by the Expert AI Models, together with the corresponding user queries, are first forwarded to the Blind-Friendly Response Editor for additional processing prior to TTS conversion. Furthermore, Urgent responses from the Generic AI

Models are routed to the Response Fusion Engine, where supplementary processing is performed concurrently as the responses are being passed to the TTS module. The subsequent sections provide a detailed description of each of these components.

3.4.1 Blind-Friendly Response Editor. At times, the raw mixture of experts system outputs are not optimally user-friendly. This can be attributed to several causes: raw data wording problems, inaccuracies, or irrelevance. When considering the user query, the raw data wording may seem awkward or unresponsive to the user's request. For instance, in response to the query: "Is there anyone I know here?" the raw facial recognition (an expert AI model) data is "John Doe," but a better response would be "Yes, John Doe is here." Similarly, inaccuracies such as a missing character from OCR can be accounted for by this system. Some information in the raw outputs can be irrelevant to the user's query, such as people listed in the object detection's response to "What are these objects?", these responses undergo editing through Blind-Friendly Response Editor by being given to an edge-side LLM along with the user's original query. The LLM is prompted to answer the query using the results from the image analysis, and the output is streamed to the user. This component adds a notable latency overhead, so it is bypassed during urgent requests. In the future, this subsystem can be extended to enhance accessibility in the systems' responses even further. During the system tests, we asked the BLV individuals how the system's responses could be improved, and some of their suggestions could be implemented the Blind-Friendly Response Editor subsystem, though updated to have generic responses routed through it as well. The specific feedback is described in the Experimental Study/Evaluation section 4.4.

3.4.2 Response Fusion Engine. The Response Fusion Engine works with the Cloud MLLM, Edge MLLM, and the TTS technology to provide a seamless transition from the low-latency Edge MLLM response to the more accurate, but slower cloud MLLM output. The lower accuracy Edge MLLM can make mistakes, and because the same request is evaluated by two independent models, this can introduce redundant or conflicting information. This component seamlessly fuses the two responses in real-time while self-correcting mistakes and avoiding detail repetition. This system is triggered during urgent, generic requests, and it leverages the simultaneous streaming of both the edge and cloud MLLMs.

As described in Algorithm 1, The Response Fusion Engine monitors the Cloud and Edge MLLMs and keeps track of which one begins generating first. If the cloud generates first, then the edge is not needed because it is assumed that the cloud is more accurate, so in this case, as shown in Line 8 of Algorithm 1, the local is stopped and the cloud output is used instead. If the Edge begins streaming first, as shown in Line 14 and Figure 4, the edge response is streamed to the TTS engine's queue (Q_{TTS}). Meanwhile, the cloud response is collected over time. When the full cloud response has been received, as denoted by t_2 in Figure 4, we prepare to run the Fusion Engine LLM. First, the edge MLLM is stopped to save resources, as shown in Line 17 of Algorithm 1, and the necessary inputs for the Fusion Engine LLM are collected. The required inputs include the full cloud MLLM response and the local MLLM response up to the sentence or word being spoken at the time when the Fusion Engine LLM is estimated to stream its first phrase, denoted as t_3 in Figure 4. We

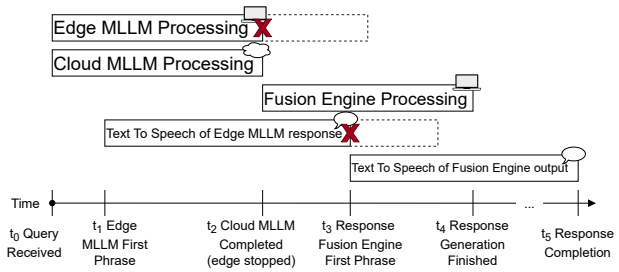

Figure 4: Schematic view of Response Fusion Engine. The edge MLLM processing is interrupted (red X symbol in the figure) once the higher quality cloud MLLM response is ready.

define this estimated time value as the 95th Percentile of the first phrase arrival time (0.5s) and denote it by T_{est}. This value, together with TTS speaking rate ($TTSR = 15$), is used to calculate the index of the last character, p, spoken by the TTS engine at the time when the Fusion Engine LLM begins streaming its response. The index is also limited (Line 19 to 27) to prevent mispronunciation of partial words.

The response data following this character is truncated from Q_{TTS} in Line 28, and the Fusion Engine LLM is given the response generated by the edge MLLM up to character p along with the full cloud MLLM response. In the prompt, the Fusion Engine LLM is told to continue from the partial Edge LLM response and to treat the full Cloud MLLM response as the ground truth, correcting itself where necessary. It is also instructed to leave out redundant information.

A key element influencing the quality of an output is the seamlessness of audio generation, characterized by smooth transitions and uninterrupted audio feedback. This, in turn, depends on two additional factors: the rate of generating input text for the voice synthesizer from fusion engine and the speed of the generated audio, which is regulated by the utilized TTS model and user-selected speech rate. In this subsection, the factors influencing the text generation speed of the Response Fusion Engine are examined. These factors include external delays caused by interruptions in input streams from edge and cloud-based MLLMs, as well as internal delays resulting from latency within the internal LLM during processing and output generation process.

We calculate these factors from our experiment's chunked data using the following equations which correspond to Figure 4. We iterate through phrases consisting of hard-punctuation-separated chunks, estimate their speaking time, and compare this to the time taken to generate each phrase from the LLM. Let n_e, n_c, and n_m denote the number of phrases produced in the edge MLLM, cloud MLLM, and internal LLM inside Fusion Engine, respectively. Each phrase is associated with a corresponding character length: L_e, L_c, and L_m, representing the number of characters per phrase. Each phrase generation time is described by τ_e, τ_c, and τ_m, corresponding to each phrase produced in the edge MLLM, cloud MLLM, and internal LLM, respectively.

The variable t_0 represents the start time of the system or interaction. The variable t_1 denotes the time at which the first edge MLLM

Algorithm 1 Hybrid edge-cloud MLLM output streaming with real-time fusion for seamless text-to-speech

Inputs: $TTSR$ ▷ TTS playback rate (characters/s),
T_{est} ▷ Estimated 95th Percentile fusion first phrase time.
Initialize two parallel MLLM workers:
  $R_{\text{Edge}} \leftarrow$ **ConcurrentGenerate**(Q)
  $R_{\text{Cloud}} \leftarrow$ **ConcurrentGenerate**(Q)
$t_0 \leftarrow$ CurrentTime()
if $\tau_{c1} \leq \tau_{e1}$ **then**
  Stop(*Edge MLLM*)
  Append all *Cloud MLLM* phrases to Q_{TTS}
  $t_2 \leftarrow$ CurrentTime()
else ▷ first phrase arrives from *Edge MLLM*
  $t_1 \leftarrow$ CurrentTime()
  while Cloud MLLM Last Token hasn't arrived **do**
    Append all *Edge MLLM* phrases to Q_{TTS}
  end while
  $t_2 \leftarrow$ CurrentTime()
  Stop(*Edge MLLM*)
  $p \leftarrow \lceil TTSR \times (t_2 - t_1 + T_{est}) \rceil$ ▷ predicted number of characters spoken
  if $p >$ length(R_{Edge}) **then**
    $p \leftarrow$ length(R_{Edge}) ▷ Limit fusion Edge response input to Edge response length
  else if $a \leftarrow$ earliest punctuation in R_{Edge} from p to end **then**
    $p \leftarrow a$
  else if p is in last word and end char is not punctuation **then**
    $p \leftarrow$ Last character of previous word. ▷ Avoid partial words for accurate TTS pronunciation.
  else
    $p \leftarrow$ Last character of current word.
  end if
  Truncate Q_{TTS} after the first p characters of R_{Edge}
  $R_{Fusion} \leftarrow$ StartFusionEngineLLM$\big(R_{\text{Edge}}[0:p],\ R_{\text{Cloud}}\big)$
  while Fusion Engine LLM is generating tokens **do** ▷ This starts at t_3
    Append all *FE* phrases to Q_{TTS}
  end while
end if

phrase becomes available and is defined as:

$$t_1 = \tau_{e1} \tag{1}$$

The variable t_2 denotes the cloud turnaround time, i.e., the time at which the cloud response is completed and becomes available:

$$t_2 = \sum_{i=1}^{n_c} \tau_{ci} \tag{2}$$

The variable t_3 represents the time at which the LLM inside the Fusion Engine generates the first merged output phrase:

$$t_3 = t_2 + \tau_{m1} \tag{3}$$

The variable t_4 represents the time at which the last fusion phrase is received, indicating that the fusion process and the generation of the combined response have been completed:

$$t_4 = t_2 + \sum_{i=1}^{n_m} \tau_{mi} \tag{4}$$

As shown in Equations (1)-(4), these time variables are influenced by the token generation rates of both edge and cloud MLLMs and the Fusion Engine LLM, and they vary depending on the input query.

The parameter p represents the number of characters that must be read from the output of edge MLLM before the first merging phrase. This is estimated using T_{est}, the estimated 95th Percentile for the fusion LLM's first phrase time, and t_1 and t_2. The parameter R represents the TTS speaking rate, expressed in characters per second. We use a buffer b that increases based on how far ahead of the text to speech we are. If it drops below 0, we consider this an interruption.

The TTS module incorporates a buffer that receives input from the various components of the Response Management Module, as illustrated in Figure 3. The parameter C denotes the number of characters currently stored in the buffer. The state of this buffer serves as an indicator of interruptions during speech output, where a negative value of C signifies that the TTS process has been interrupted. Such interruptions occur when the TTS speaking rate exceeds the rate at which tokens are streamed from components of the Response Management Module, resulting in temporary buffer underflow. In this section, to evaluate the seamlessness of the Fusion Engine generated response, we analyze the buffer state and measure the cumulative duration of TTS interruptions caused by the TTS outpacing Fusion Engine token streaming. This metric provides a quantitative assessment of the continuity and smoothness of the spoken output.

We derived mathematical formulas to quantify interruption metrics associated with the primary sources of interruption in the fusion system. For these calculations, we assume a constant TTS rate. Under this assumption, interruptions in speech synthesis (i.e., buffer underflow) can occur for two reasons: (1) the token generation rate is lower than the speaking rate, and (2) no tokens are generated while the TTS system is synthesizing speech. As mentioned earlier, the time interval associated with receiving each phrase from a model is used as the temporal basis of the system evaluation. The buffer state is evaluated at each such time step. Accordingly, upon the reception of the $k-th$ phrase from the model, the buffer state is updated according to Equation 5.

$$C_k = \begin{cases} |C_{k-1} + L_k - R \times \tau_k| & \text{if } C_{k-1} + L_k - R \times \tau_k i \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The first source of interruption can be formalized based on the number of buffer-underflow occurrences during the token generation of each model that fills the buffer as shown in Equation 6,

$$U_n = \sum_{k=1}^{n} \mathbf{1}(C_{k-1} + L_k - R \times \tau_k i \leq 0) \tag{6}$$

where $1(\cdot)$ is the indicator function, equal to 1 when the buffer occupancy is non-positive and 0 otherwise. The buffer-underflow causes a gap in speech synthesis, such that no character is available until the end of the next phrase time step. The total gap is computed as the sum of all intervals that follow a buffer-underflow state as shown in Equation 7,

$$T_{\text{gap}} = \sum_{k=1}^{n-1} \tau_{k+1} \, \mathbf{1}(C_{k-1} + L_k - R \times \tau_k i \leq 0) \tag{7}$$

Thus, whenever the buffer occupancy becomes non-positive, the subsequent time interval contributes to the gap. As previously mentioned, the second source of interruption arises when no tokens are generated during speech synthesis because the invoked model has not yet begun producing output tokens. Two key factors affect this metric: (i) the waiting time to receive the first token from the invoked model and (ii) the number of characters stored in the buffer (C) at the time of invocation. For example, C_{FELLM} denotes the number of characters stored in the buffer at the time the internal LLM within the Fusion Engine is invoked.

The first factor varies across the LLMs in our system. For the edge MLLM, it depends on the query and the edge computational resources. For the cloud MLLM, it depends on the query, network conditions, and other server-related parameters. For the Fusion Engine LLM, it depends on the edge computational resources, the input length, and the complexity of the fusion process. The second factor, C, is equal to zero when the system invokes MLLMs (Algorithm 1, lines 4 and 5) but it can be nonzero ($C_{FELLM} = TTSR \times T_{est}$) when the system invokes FE LLM. This factor plays a critical role in the interruption gap, as underestimating the FE LLM's first-phrase generation time in Algorithm 1 (line 18) can result in the truncation of excess buffered characters in line 28. Consequently, the interruption gap may increase. This gap can be quantified using Equation (8).

$$T_{\text{FE gap}} = \tau_{m1} - T_{est} \tag{8}$$

According to Algorithm 1, if the condition in the *if* statement (Line 7) is satisfied, the interruption is equal to the cloud MLLM first-token waiting time (τ_{c1}) plus the total cloud MLLM gap, as defined in Equation 7. Otherwise (Line 11), the interruption depends on C_{FELLM} and comprises the edge MLLM and Fusion Engine LLM first-token waiting times (τ_{e1} and τ_{m1}), together with the total edge MLLM gap and the total local LLM gap, as defined in Equation 7.

The custom-defined metric M, as expressed in Equation 9, is designed to better reflect the impact of interruptions on the speech synthesis quality. It penalizes longer interruptions by scaling the interruption time exponentially. A larger M-value means a *worse score* because there are more frequent or longer interruptions. Each interruption's time is increased by one before being raised to the power of 2 to ensure short interruptions are not effectively ignored.

$$M = \sum_{i=1}^{n} \left(\begin{cases} (|C_{k-1} + L_k - R \times \tau_k i| + 1)^2 & \text{if } C_{k-1} + L_k - R \times \tau_k i < 0 \\ 0 & \text{otherwise} \end{cases} \right) \tag{9}$$

4 Evaluation

Experimental design. We tested our system's performance through both automated experiments with datasets, and through real-time interactions with actual blind and visually impaired experiment participants.

4.1 Automated Experiments Setup

4.1.1 Dataset selection. To assess and enhance the system's question-answering capabilities and contextual comprehension, we utilized the VizWiz dataset [22], which consists of photographs captured by BLV individuals and provides a realistic benchmark for evaluating the system's ability to address the real needs of BLV users through visual question answering tasks. For our system evaluation, we created a custom subset by selecting 200 images from the VizWiz dataset, with each image paired with two questions. To thoroughly evaluate the system's performance across different domains, we categorized the image–question pairs into two groups: 100 pairs that are relevant to the Reasoning Engine's expert AI Models, and other 100 pairs that are more general and unrelated to these experts and are instead processed by the Reasoning Engine's generic AI Models. We further divided each group to explore different processing scenarios for both normal and urgent tasks. For half of the pairs, we generated questions in an urgent format, enabling the Urgency Detection module to identify and prioritize these tasks for faster processing via the fast-track pipeline. This approach allows for a comprehensive evaluation of the system's adaptability to varying task urgency, as well as its ability to process pairs based on the queries' context.

4.1.2 Hardware Setup. The Audo-Sight system, as depicted in Figure 3 and tested in the evaluation section, was implemented on a Windows 11 machine equipped with an Intel Core i9-10850K (10 cores / 20 threads), 32 GB of system RAM, and an NVIDIA RTX 3080 Ti with 12 GB of dedicated VRAM. This GPU served as the primary accelerator for the local MLLM inference with Gemma3-4b, object detection with yoloe-v8l-seg-pf [13], and dense embedding extraction with the fine-tuned all-MiniLM-L6-v2. API calls were made on a 5G Wi-Fi connection in the region of North Texas.

4.2 System Performance Evaluation

In this evaluation, we assessed the latency, accuracy, and supported speech rate of the proposed system. Through interviews with BLV users, we identified two primary factors affecting user satisfaction: system responsiveness(i.e., latency) and response quality. The goal of designing the Audo-Sight is to improve system responsiveness while maintaining high response quality. The Audo-Sight 's Reasoning Engine employs multiple AI Models and MLLMs operating across the edge–cloud continuum. For urgent tasks, BLV users expect concise and immediate responses, whereas for non-urgent tasks, BLV users may prefer more detailed yet succinct outputs. Response latency and quality for each task depend on how effectively the Cognition Module employs AI Models and MLLMs. Achieving this objective requires efficient task routing within the reasoning engine, which is enabled by accurate Urgency Detection and AI Router modules. An optimal solution would fully understand user

queries and intents and route them to the most appropriate available AI engine.

Latency Evaluation. We measured two latency metrics: First Token Latency (TTFT) and End-to-End Latency (Turnaround Time). TTFT is defined as the time elapsed until the first word of the response is received, while Turnaround Time refers to the total time required to receive the complete response. BLV users typically employ different TTS applications with varying speech rates based on personal preference. Therefore, we excluded speech synthesis time and playback delays from our latency measurements to ensure a fair comparison.

We present a comparison of TTFT and End-to-End Latency for both urgent and normal tasks across three systems: the proposed system Audo-Sight, a local MLLM, and a cloud-based MLLM (GPT-5 [30]). As shown in the Figure 5(a), the proposed system significantly reduces TTFT for urgent tasks, achieving approximately 80% improvements compared to the cloud-based MLLM. Similar performance gains are observed for normal tasks, where the proposed system improves TTFT by about 50% relative to the cloud-based MLLM. For both urgent and normal tasks, the local MLLM demonstrates lower TTFT, achieving reductions of 71% and 89% relative to Audo-Sight, respectively, and reductions of more than 94% relative to cloud-based MLLMs. The relatively small errors in Audo-Sight are expected, as it less frequently relies on cloud inference, which makes it less sensitive to network latency, cloud server load, and variability in response times. As a result, Audo-Sight not only achieves lower latency overall, but also exhibits more predictable (low variance) response times.

The end-to-end latency of the proposed system is highly dependent on the end-to-end latency of the cloud-based MLLM. Considering the distribution of our experimental dataset, approximately half of the tasks are processed through the MLLM pipeline, while the remaining tasks are routed to expert AI Models in the Audo-Sight system. For normal tasks not routed to expert AI models, Audo-Sight relies on the cloud-based MLLM for response generation, resulting in no significant latency improvement compared to the cloud-based baseline. Similarly, for urgent tasks not routed to expert AI Models, the system also depends on the cloud-based MLLM to complete responses in the fusion engine. As shown in Figure 5(b), the average end-to-end latency of Audo-Sight is roughly 50% of that of the cloud-based MLLM, reflecting the partial routing of tasks to expert AI Models at the edge. For both urgent and normal tasks, the edge-only system exhibits significantly lower end-to-end latency, achieving reductions of 80% compared to Audo-Sight. However, for normal queries, Audo-Sight does not require any edge MLLM requests. In the case of urgent tasks, Audo-Sight requires 40% fewer edge calls compared to the edge-only system, leading to lower energy consumption on the edge. Additionally, as demonstrated in the next section, the accuracy results show that Audo-Sight generates more accurate responses compared to the edge-only MLLMs.

Based on our observations, the difference between TTFT and end-to-end latency for tasks relying on cloud-based MLLMs is approximately 200 *ms* on average and is negligible. This is because measuring the exact model-internal TTFT (i.e., the moment the first token is generated inside cloud-based MLLM servers) is impossible. Instead, we measure the earliest externally observable

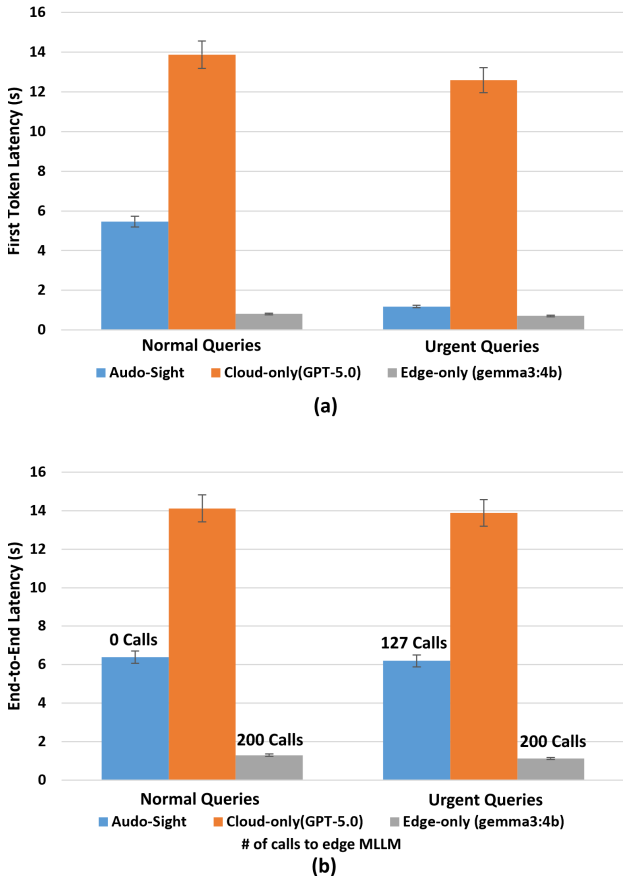

Figure 5: (a) First Token Latency (TTFT) comparison for urgent and normal tasks across three systems. (b) Comparison of end-to-end latency for urgent and normal tasks across three systems. Bars represent mean values obtained from experimental runs; error bars indicate 95% confidence intervals.

TTFT, which is heavily affected by network transmission, batching, framing, serialization, and other sources of latency.

Accuracy Evaluation. In question-answering systems, LLMs can be employed to assess the quality of responses generated by multiple models. In this study, we use LLMs as a scoring method to evaluate and compare the accuracy of three system-generated outputs. This approach requires a ground truth answer. Prior research [5, 9, 10, 26] has leveraged GPT as a robust MLLM in similar systems or studies. Consequently, we use our cloud-based MLLM, GPT-5, as the ground truth answer and evaluate how Audo-Sight and our edge MLLMs, Gemma3:4b respond to a specific user query in comparison. We selected GPT-5 as model to perform the evaluation and the process involves scoring each model's response across four criteria: correctness (accuracy), completeness (extent to which key points from the ground truth are addressed), faithfulness (absence of incorrect or unsupported information), and clarity (organization and ease of understanding). Each criterion is rated on a scale from 1 to 5, and an overall score is derived, typically calculated as the average of the individual criteria ratings. Figure 6

presents the scores for both Audo-Sight and edge-based MLLMs. As shown, Audo-Sight outperforms the Edge MLLMs in terms of correctness, completeness, faithfulness, and overall scores. However, Edge MLLMs achieve a higher score in the clarity criterion. This is because Audo-Sight provides responses without additional refinements in cases of urgency, which can affect the clarity of the output.

Speaking Rate Evaluation The Max-TTS-Speed evaluation metrics is informed by BLV screen-reader usage and preferences from our own interviews and existing research. We noticed that many participants in our human study (discussed in section 4.4) use increased TTS speeds for screen-readers. Another study shows that many visually impaired individuals prefer high speaking rates, with some listening to over 500 words per minute, a speed that is 3X an average english speaking rate [7]. Therefore, the Max-TTS-Speed evaluation metric, denoted as S_{max} reports the highest possible speaking rate of the system without interruption. To determine S_{max}, we use use equation 11, which requires the set of phrases spoken by the system. Let n represent the total number of phrases in a given response from the system, and let τ be the set of generation times per phrase. The first phrase generation time is considered to be zero since TTS output does not begin until after this phrase is generated. Let L be the set containing the number of characters in each phrase chunk generated by the system in a given response. For each integer i from 1 to n, we sum up the number of characters in the first i phrases of the response. We then divide each of these cumulative character counts by the corresponding value of τ. We call the resulting set M, which is the set of accumulated characters over their total time that has passed since text to speech started. By finding the minimum value in M, we obtain the highest sustainable character/second speaking rate that will avoid an interruption or gap in the text to speech output.

$$M = \{\frac{\sum_{k=1}^{i} L_k}{\tau_i} \mid i \in I\}, \quad I = \{1, 2, \ldots, n\} \tag{10}$$

$$S_{\max} = \min(M) \tag{11}$$

Applying this formula to the system output across the various routes shown in fig. 3, the non-urgent generic route can always achieve up to 8.8X normal speaking rate or higher, and similarly, the non-urgent expert route can always achieve 10X normal speaking rate or higher. These rates are much higher than a typical preference for BLV people [7]. Since the full response is made available all at once in the route involving the expert models, the maximum speaking rate is always unlimited in this case. The case of urgent-generic involving the fusion engine will be discussed in-depth in section 4.3.

4.3 Merging System Quality Evaluation

We evaluate the quality of the merging system using three criteria: Semantic Quality Evaluation, which assesses the accurate handling of redundant, contradictory, and supplemental information, Response Max Speed, and Seamlessness Evaluation.

4.3.1 Speed and Seamlessness Evaluation. We ran a post hoc evaluation using our custom dataset with 100 urgent questions paired with images from the VizWiz dataset [22] and recorded the generation phrase timing to find S_{max} for each response as defined in equations 10 and 11. The results are shown in fig. 7 and demonstrate a notable bottleneck compared to other routes taken by the system, which are discussed earlier in this section. When using the cloud or edge model in isolation, the difference in the response time of each model does not influence the maximum speed. However, when using the models in tandem, the difference in their response times must be masked by the text-to-speech output of the first model to ensure a seamless output. This principle is demonstrated by the second evaluation using the metric proposed in Equation 9. The same queries and response data are used to plot the graph in Fig. 8. This metric penalizes interruptions introduced by the merging system without including the "gap" between a complete edge MLLM response and the merged response. The results span many speaking rates, and demonstrate that interruptions are rare and short until around 150 characters per second, which is ten times an average speech rate. At a normal speaking rate, 84% of responses have no interruption, and the rest have minimal interruption, but at a higher rates, interruptions become more common. One solution to this would be to create a "target speaking rate," where the text to speech speeds up or slows down dynamically based on expected cloud turnaround time and buffer size. Additionally, a smaller cloud model would increase the speaking, though most likely at the cost of some accuracy. For requests that are non-urgent and involve an expert model, the response is edited, and tests show that the text to speech rate supported is much faster than that of routes involving multi-modal LLM, which are the bottleneck of the system.

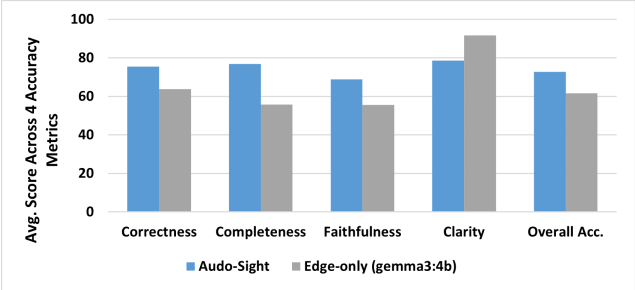

Figure 6: Accuracy comparison of Audo-Sight and edge MLLMs responses across four metrics and their overall accuracy performance, with scores based on GPT-5 responses as the ground truth.

4.3.2 Semantic Quality Evaluation. As discussed in Section 3.4.2, when an identical request is evaluated by two independent AI models, their respective responses may differ, potentially introducing redundant or conflicting information. A fusion system is considered effective when it delivers satisfactory outcomes across inputs that may contain redundant or conflicting information. To examine the performance, we first perform a semantic comparison of the two inputs for all cases processed by fusion engine module. Based on comparisons performed by three independent researchers, each input pair is classified into one of four categories. The first category, Equivalent (Eq), comprises pairs that contain identical information. The second category, Complementary Collaborative (CC), includes

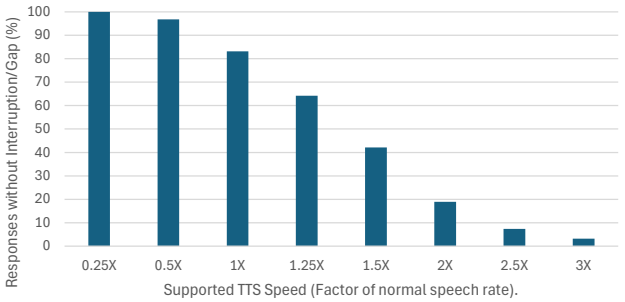

Figure 7: Percentage of Fusion responses without interruption at various Text-To-Speech Speeds (counting 15 characters/second as normal).

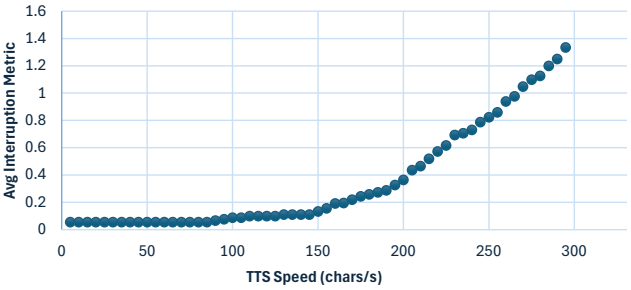

Figure 8: Average Interruption time metric with Varied Voice Speed - Our tests show the system has minimal interruptions even when outputting at 10x a normal speaking speed.

pairs that contribute additional information on the same topic (e.g., car color and car model). The third one, Distinct collaborative (DC), consists of pairs that provide correct information on different topics (e.g., in a scene description task, one input describes a car while the other describes a person within the same scene). The final category, Contradictory (Contra), includes pairs that contain conflicting information. The distribution of categories with the winning vote is as follows: Equivalent (48%), Complementary Collaborative (23%), Distinct Collaborative (2%), and Contradictory (27%).

To assess the semantic quality of our fusion engine outputs, we use responses generated by GPT-5 as a baseline. FLAN-T5[15], used in prior work [24] as the GenFuser module and identified as an effective technique for integrating responses from multiple sources. It is included as a comparative method alongside our proposed approach. We compare the outputs of our fusion engine and those produced by FLAN-T5 against the baseline, using GPT-5 as a judge LLM. In this evaluation, both our fusion model and FLAN-T5 used identical prompts. FLAN-T5 allows for dynamic prompting, which allows us to evaluate its merging ability with our seamless response merging use case. For comparison, we also evaluate the quality of the concatenation merging strategy, that is simply forwarding the cloud output without any merging logic, effectively appending it to the local output spoken so far.

To enable more effective evaluation, we define four metrics: Redundancy Handling (RH) is higher when the merged result contains less repetition of previously stated information. Contradictory Handling (CH) rewards correction where previously stated information contradicted the cloud response. Supplemental Handling (SH) is higher when new, complementary information is included in the merged response. Fluency is higher when the response is more grammatically correct and seamless. Claude Opus 4.7 was tasked with scoring all outputs according to these metrics, 30 percent of its judgments were reviewed by a human researcher and found to be largely accurate. Tables 1, 2, and 3 present the scores for each category, along with the averages across all categories. FLAN-T5 achieves the lowest performance, with average scores in the 40-50 range. T5 performs worse than basic concatenation due to issues repeating previously spoken information. In contrast, our fusion engine module substantially outperforms both FLAN-T5 and concatenation overall, although it does not reach the performance level of the GPT-5 baseline and falls short of concatenation in certain situations. When judging how much supplemental information is included, concatenation performs better than Gemma3 model. This is likely because concatenation includes all the new information while the LLM tends to over-filter, which also results in a high redundancy handling score.

Table 1: Merging Quality Across Response Relationships for Simple Concatenation (Concat).

Metric	Eq	CC	DC	Contra	Avg
RH	49.4	84.4	40.0	99.3	70.9
CH	62.6	60.9	90.0	67.4	64.0
SH	74.0	87.8	100.0	88.2	81.6
Fluency	35.3	43.5	40.0	50.4	41.4
All	55.3	69.1	67.5	76.3	64.5

Table 2: Merging Quality Across Response Relationships for Flan-T5.

Metric	Eq	CC	DC	Contra	Avg
RH	41.7	68.6	100.0	71.8	56.5
CH	47.0	37.1	30.0	40.9	42.9
SH	47.0	40.0	50.0	56.4	47.7
Fluency	34.8	43.8	60.0	38.2	38.2
All	42.6	47.4	60.0	51.8	46.3

4.4 User Evaluation (Human Study)

Overview. To test how the system performs for BVI individuals, we conducted an experiment with 8 legally blind participants, (aged 41-74, 6 female, 2 male; 6 with moderate or higher AI chat/voice assistant experience; 4 blind from birth). The experiment was performed virtually with pre-taken images from different environments and simulated situations to mitigate risks of real-world tests. The participants were asked to query the system as if they were in the given environment, and they provided feedback regarding the system's responses.

Table 3: Merging Quality Across Response Relationships for Gemma3:4b (used by our system).

Metric	Eq	CC	DC	Contra	Avg
RH	92.3	94.8	100.0	99.3	95.0
CH	62.6	53.9	60.0	68.2	62.0
SH	70.2	69.6	80.0	74.1	71.3
Fluency	71.9	66.1	50.0	75.6	71.1
All	74.3	71.1	72.5	79.3	74.9

Procedure. Participants interacted with both our system and GPT 5 without knowing which one they were using. The participants entered their queries textually via a text prompt in our web app. The experiment was split into two situations, each of which involved three images in a sequence. This tested the system's ability to help the user complete multi-step goals, which included finding the restaurant counter and ordering food from a menu in the first situation, or meeting with a traveling companion at an airport and catching a flight in the second. Each situation was completed by each participant twice; once with Audo-Sight and once with GPT-5 via OpenRouter. The system order was alternated for each participant to mitigate bias for or against the first system tested. After participants tested the system, the facilitating experimenter collected feedback from each participant regarding the system's design and trade-offs. The participants also offered potential improvements to the blind-friendliness Response Editor.

Results. When directly comparing the two systems, participants clearly demonstrate a preference for our system, with 62% of responses saying they preferred guidance from Audo-Sight overall compared to 15% of responses preferring GPT-5. The users cited response expediency as a large reason in the direct comparisons and in ratings of each individual system, which correlates with our system's higher measured speed. Additionally, participants rated our system 15% higher that GPT-5 in message quality, saying that our system's responses included a satisfactory amount of detail. When rating whether the system was sufficient for understanding the environment, on average, the participants rated the two systems to be equal in helping participants imagine the environment (4.1/5) and roughly equal in terms of clarity (Audo-Sight- 4.1/5; GPT-5 - 4.3/5). In the discussion following the testing, participants praised the real-time merging system, saying that a fast, self-correcting system is better than a slow one in spite of higher initial accuracy, and they also voiced approval of the ability to control this feature by use of urgent language. The participants also offered some feedback regarding the quality of the responses which could be addressed in future work. The Blind-Friendly Response Editor could be extended to accommodate subjective preferences concerning phrases such as "I see your point," or "Look under your chair," or preferences for when to describe colors and visual details. Instructions such as "Follow signs indicating XYZ," which rely on visual cues, can be avoided through a combination of the subtractive LLM-based Blind-Friendly Response Editor and MLLM-prompting. A suitable prompt combined with depth-sensors for distance information could potentially result in better scene descriptions in relation to the individual with specific angles and distances to provide clarity (e.g., it is positioned at 2 o'clock, 5 feet away). Additionally, participants suggested continuous updates would help maintain a user's trajectory during long, straight walks, which would be an improvement over the "Walk straight" one-time instruction since any minor deviation will be amplified over significant distances.

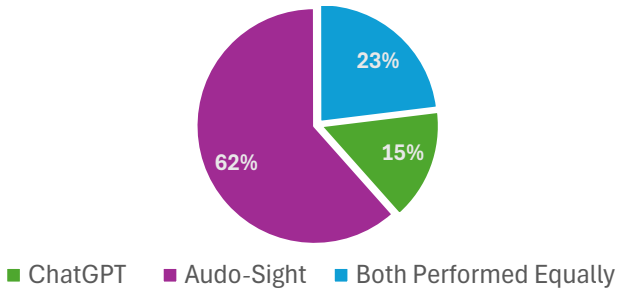

Figure 9: BLV Participant Guidance Preference: 62% of responses were in favor of Audo-Sight for the question "Which system's guidance did you prefer overall?"

5 Discussion

Although recent MLLM systems have demonstrated impressive performance, translating these advances into effective, user-centered assistive technologies is a significant challenge. Despite the promising findings of this study to overcome this challenge, several remaining limitations should be acknowledged.

Importantly, the system performance evaluation was conducted using static images paired with user queries, and the current urgency detection mechanism relies primarily on the user's query. This setting does not fully capture the dynamic and continuous nature of real-world environments. Future work should therefore focus more on system performance evaluation methods that more closely resemble real-world usage patterns. For example, they can incorporate natural, multi-turn conversations in which an AI agent emulates a BLV user and continuously interacts with the system. In addition, short video sequences capturing evolving scenes could be used to evaluate the system performance during a context-aware assistance. Furthermore, incorporating visual information into the urgency detection and routing process could improve robustness and enable more accurate responses in rapidly changing or safety-critical situations. Nevertheless, in this study, we tried to mitigate these limitations by complementing the system performance evaluation with a human study that provided insights into how participants perceived the system during realistic and interactive scenarios. This allowed us to assess not only technical performance but also user experience, perceived usefulness, and practical considerations for real-world deployment.

A key contribution of this study is the joint utilization of cloud-based and edge-based MLLMs and the essence of real-time fusion of their outputs. One may argue that with the rapid development of edge AI, such contribution will have short lifetime. We believe that even though both cloud and edge models continue to advance rapidly, fundamental differences between them are expected to persist due to their inherent characteristics. These differences include model capability and output quality, supported modalities,

inference latency, user privacy, operational cost, energy consumption, and domain-specific knowledge, among other factors. In this work, we exploit the complementary strengths of both paradigms. Specifically, within the Response Management Module, we leverage the low latency of edge-based MLLM to provide rapid initial responses during urgent situations while compensating for their relatively lower accuracy by fusing these responses with those generated by a more capable cloud-based MLLM. In addition, we integrate Expert AI models, such as face recognition, for tasks that cannot be effectively or efficiently handled by general-purpose MLLMs. Consequently, despite the rapid evolution of AI models, the proposed framework, particularly its routing and response fusion mechanisms, will remain applicable in various systems (even beyond accessibility) in the foreseeable future. Rather than relying on the superiority of a single model, our approach unlocks effective *integration* of multiple model with complementary capabilities to achieve a more robust and practical assistive system.

Many existing assistive systems rely heavily on cloud-based services, introducing challenges such as operational costs, privacy concerns, and the requirement for a continuous Internet connection. In contrast, the proposed framework reduces dependence on cloud services by establishing several functionalities to operate offline using edge-based models. Although this design may result in lower response quality in some scenarios, it offers advantages in terms of privacy, availability, reduced communication overhead, and resilience in environments with unreliable network connectivity. While this study focused on efficiency aspects, more exploration is needed to evaluate other aspects, such as privacy and network connectivity adaptation.

6 Conclusion

In this paper, we propose Audo-Sight, which represents a significant advancement in assistive technology for BLV individuals by integrating edge and cloud AI processing to provide timely, accurate, and context-aware conversational ambient perception. More technically, we sought to bridge the gap between the remarkable capabilities of MLLMs and their practical deployment challenges. The system's intelligent query routing, combined with multiple processing pipelines and the Response Fusion Engine, allows it to infer user intent effectively and generate responses quickly for urgent tasks while ensuring highly accurate and comprehensive outputs. Evaluations show that Audo-Sight outperforms cloud-based MLLMs, starting responses to urgent tasks approximately 80% faster and generating complete responses about 50% faster across all tasks, with only a 25% decrease in response quality with respect to high-end cloud MLLM. Furthermore, to improve robustness and availability, Audo-Sight's Reasoning Engine leverages both edge and cloud AI models.

Our future direction is to improve the accuracy of the urgency detector to cover a wider range of implicitly urgent queries, as well as to enhance the AI Router's ability to better understand user intent. We also plan to make the Response Manager more intelligent by improving the blind-friendly Response Editor according to the experiences of interviewed BLV participants in this study. The Response Fusion Engine can also be improved by incorporating two-way communication between the Fusion Engine and the text-to-speech engine for better prediction of the spoken character count. Another goal for our future design is to enhance the Reasoning Engine by incorporating additional expert AI models and reducing reliance on MLLMs through improved edge AI capabilities.

Acknowledgment

We are thankful to the constructive comments of anonymous reviewers of the paper. We are also thankful to Aliyeh Haghollahi, Illango Zaruba, and Sai Donepudi who helped us in the results' verifications. This project is supported by NASA through MUREP ACEIR 2.0 Program, Award# 80NSSC25M0054 and NSF CAREER Award#2419588.

References

[1] 2023. GPT-4V(ision) System Card. https://api.semanticscholar.org/CorpusID:263218031
[2] 2025. *Envision Glasses User Guide Version 2.9.6*. Technical Report. Envision Technologies B.V., The Hague, The Netherlands.
[3] Moondream AI. 2025. *Moondream VLM*. https://moondream.ai/
[4] Risheek Bajaj, Puneet Khanna, Rahul Goel, and Rohan Bhargav. 2023. Empowering the Visually Impaired: A Sustainable ML-Based Currency Recognition System. In *Sustainable Development through Machine Learning, AI and IoT*, Pawan Whig, Nuno Silva, Ahmed A. Elngar, Nagender Aneja, and Pavika Sharma (Eds.). Springer Nature Switzerland, Cham, 39–50.
[5] Be My Eyes. 2023. Be My AI. https://www.bemyeyes.com/bme-ai/. Accessed: 2025-12-05.
[6] Saidarshan Bhagat, Padmaja Joshi, Avinash Agarwal, and Shubhanshu Gupta. 2023. Accessibility evaluation of major assistive mobile applications available for the visually impaired. *ITU Journal on Future and Evolving Technologies* 4, 4 (Dec. 2023), 631–643. doi:10.52953/tnrv4696
[7] Danielle Bragg, Katharina Reinecke, and Richard E. Ladner. 2021. Expanding a Large Inclusive Study of Human Listening Rates. *ACM Trans. Access. Comput.* 14, 3, Article 12 (July 2021), 26 pages. doi:10.1145/3461700
[8] Ruei-Che Chang, Chia-Sheng Hung, Bing-Yu Chen, Dhruv Jain, and Anhong Guo. 2024. SoundShift: Exploring Sound Manipulations for Accessible Mixed-Reality Awareness. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference* (Copenhagen, Denmark) *(DIS '24)*. Association for Computing Machinery, New York, NY, USA, 116–132. doi:10.1145/3643834.3661556
[9] Ruei-Che Chang, Yuxuan Liu, and Anhong Guo. 2024. WorldScribe: Towards Context-Aware Live Visual Descriptions. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 140, 18 pages. doi:10.1145/3654777.3676375
[10] Maryam S Cheema, Sina Elahimanesh, Samuel Martin, Pooyan Fazli, and Hasti Seifi. 2025. DescribePro: Collaborative Audio Description with Human-AI Interaction. In *Proceedings of the 27th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '25)*. Association for Computing Machinery, New York, NY, USA, Article 86, 19 pages. doi:10.1145/3663547.3746320
[11] Lihu Chen and Gaël Varoquaux. 2024. What is the role of small models in the llm era: A survey. *arXiv preprint arXiv:2409.06857* (2024).
[12] Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. 2024. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems* 37 (2024), 66305–66328.
[13] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. 2024. Yolo-world: Real-time open-vocabulary object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 16901–16911.
[14] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tai, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2024. Scaling instruction-finetuned language models. *J. Mach. Learn. Res.* 25, 1, Article 70 (Jan. 2024), 53 pages.
[15] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts,

Denny Zhou, Quoc V. Le, and Jason Wei. 2022. *Scaling Instruction-Finetuned Language Models*. arXiv:2210.11416 [cs.LG] https://arxiv.org/abs/2210.11416
[16] CloudSight, Inc. 2013. TapTapSee. https://taptapseeapp.com. Accessed: 2025-12-04.
[17] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618* (2024).
[18] Envision Technologies. 2017. Envision. https://www.letsenvision.com/app. Accessed: 2025-12-01.
[19] Nasir Uddin Foysal, Thamid Junaeid Thamid, Md Shihab Uddin, Md Fakrul Islam, Muhammad Nazrul Islam, and Faiz Al Faisal. 2023. Advancing AI-based Assistive Systems for Visually Impaired People: Multi-Class Object Detection and Currency Classification. In *2023 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*. 438–442. doi:10.1109/ICICT4SD59951.2023.10303628
[20] Google Gemini Team. 2025. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805 [cs.CL] https://arxiv.org/abs/2312.11805
[21] Ricardo E. Gonzalez Penuela, Ruiying Hu, Sharon Lin, Tanisha Shende, and Shiri Azenkot. 2025. Towards Understanding the Use of MLLM-Enabled Applications for Visual Interpretation by Blind and Low Vision People. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '25)*. Association for Computing Machinery, New York, NY, USA, Article 536, 8 pages. doi:10.1145/3706599.3719714
[22] Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. 2018. VizWiz Grand Challenge: Answering Visual Questions From Blind People. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[23] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031* (2024).
[24] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 14165–14178.
[25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692 [cs.CL] https://arxiv.org/abs/1907.11692
[26] Zain Merchant, Abrar Anwar, Emily Wang, Souti Chattopadhyay, and Jesse Thomason. 2024. Generating Contextually-Relevant Navigation Instructions for Blind and Low Vision People. In *Interactive AI for Human-Centered Robotics (InterAI) Workshop @ Ro-MAN*. https://arxiv.org/abs/2407.08219
[27] Microsoft. 2017. Seeing AI. https://www.seeingai.com/. Accessed: 2025-12-02.
[28] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665* (2024).
[29] OpenAI. 2025. *GPT-4o-mini: OpenAI's Compact Language Model*. https://www.openai.com/gpt-4o-mini
[30] OpenAI. 2025. *GPT-5 (Version gpt-5) [Large language model]*. https://platform.openai.com/docs/models/gpt-5
[31] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust Speech Recognition via Large-Scale Weak Supervision. arXiv:2212.04356 [eess.AS] https://arxiv.org/abs/2212.04356
[32] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.
[33] Mahadev Satyanarayanan. 5555. The Duality of "Carry" and "Find" in Mobile and Pervasive Systems . *IEEE Pervasive Computing* 01 (April 5555), 1–11. doi:10.1109/MPRV.2026.3676231
[34] Patrick Slade, Arjun Tambe, and Mykel J. Kochenderfer. 2021. Multimodal sensing and intuitive steering assistance improve navigation and mobility for people with impaired vision. *Science Robotics* 6, 59 (2021), eabg6594. arXiv:https://www.science.org/doi/pdf/10.1126/scirobotics.abg6594 doi:10.1126/scirobotics.abg6594
[35] Ajay Narayanan Sridhar, Fuli Qiao, Nelson Daniel Troncoso Aldas, Yanpei Shi, Mehrdad Mahdavi, Laurent Itti, and Vijaykrishnan Narayanan. 2025. NaviSense: A Multimodal Assistive Mobile application for Object Retrieval by Persons with Visual Impairment. In *Proceedings of the 27th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '25)*. Association for Computing Machinery, New York, NY, USA, Article 110, 9 pages. doi:10.1145/3663547.3759726
[36] Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient Few-Shot Learning Without Prompts. arXiv:2209.11055 [cs.CL] https://arxiv.org/abs/2209.11055
[37] Ethan Waisberg, Joshua Ong, Mouayad Masalkhi, Nasif Zaman, Prithul Sarker, Andrew G Lee, and Alireza Tavakkoli. 2024. Meta smart glasses—large language models and the future for assistive glasses for individuals with vision impairments. *Eye* 38, 6 (2024), 1036–1038.

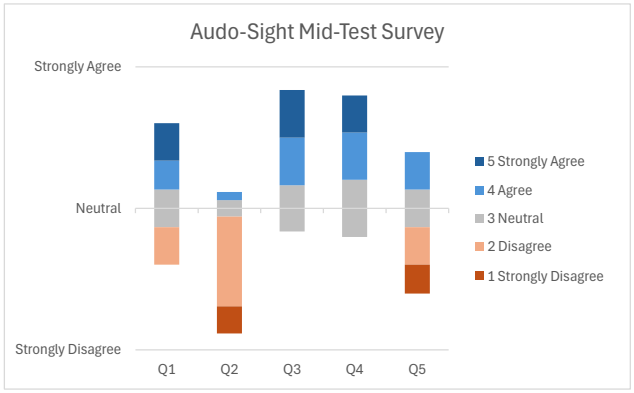

Figure 10: Human evaluation results after testing Audo-Sight.

[38] Biao Yi, Xavier Hu, Yurun Chen, Shengyu Zhang, Hongxia Yang, and Fan Wu. 2025. EcoAgent: An Efficient Device-Cloud Collaborative Multi-Agent Framework for Mobile Automation. https://api.semanticscholar.org/CorpusID:278394644

A Additional Human Testing Information

Tables 4 and 5 list the questions given during the Human Study, and figures 10 and 11 provide diverging bar charts showing the responses from the users. The bar charts' similar patterns may be explained by the structure of the testing procedure. The Mid-Test survey does not ask the users to compare the two systems with each-other, and the alternating order helps cancel out starting or ending bias for either system. Audo-Sight outperforms GPT-5 in the post-test-survey when the users directly compare them after completing the full set of testing.

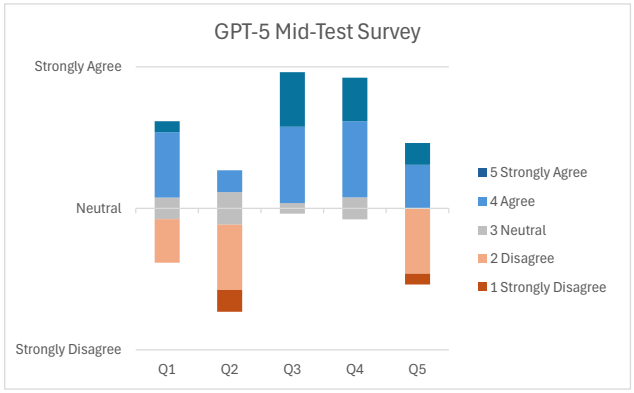

Figure 11: Human evaluation results after testing GPT-5.

Table 4: Mid-Test Survey Questions

ID	Question	Choices
Q1	The system felt responsive or answered quickly.	1-5 (Strongly agree, agree, neutral, disagree, strongly disagree)
Q2	The information provided by the system felt too simple and left out too many details.	1-5 (Strongly agree, agree, neutral, disagree, strongly disagree)
Q3	The system's explanations were clear and easy to understand.	1-5 (Strongly agree, agree, neutral, disagree, strongly disagree)
Q4	The system's guidance helped me imagine and understand the surrounding environment.	1-5 (Strongly agree, agree, neutral, disagree, strongly disagree)
Q5	The system was unresponsive or took too long to answer.	1-5 (Strongly agree, agree, neutral, disagree, strongly disagree)
Q6	Were any of the system's responses confusing or unhelpful in any way? If so, please explain what you found confusing or unhelpful and why.	Free response, optional
Q7	Were any of the system's responses offensive or insensitive? If so, please explain what you found offensive and why.	Free response, optional

Table 5: Post-Test Survey Questions

ID	Question	Choices
Q1P	Which AI guidance felt faster and more responsive?	System 1, system 2, both performed equally
Q2P	Which AI guidance were the most useful for achieving the urgent goal of catching your flight quickly?	System 1, system 2, both performed equally
Q3P	Overall, which AI guidance do you prefer?	System 1, system 2, both performed equally
Q4P	If you have any specific observations comparing the quality of the responses in the last two scenarios, please mention them here.	Free response, optional