SensoryScape: A Modular System for Multimodal Navigation Assistance on Mobile Devices

by

Nikhil Herdt

A thesis submitted in partial fulfillment
of the requirements for the
Master of Science degree in Electrical and Computer Engineering in the
Graduate College of
The University of Iowa

May 2026

Thesis Committee: Terry Braun, Thesis Supervisor
Tyler Bell
Kishlay Jha
Stephen Russell

ABSTRACT

Individuals who are blind or visually impaired face persistent challenges when navigating unfamiliar environments, with downward falls from stairs, curbs, and sudden elevation changes posing the risk of serious injury. Existing assistive navigation technologies tend to focus on a single feedback modality and address either reactive obstacle detection or proactive environmental exploration, but rarely both simultaneously. This thesis presents SensoryScape, a real-time multimodal assistive navigation system that combines spatial audio, wearable haptic feedback, and on-device intelligence on consumer mobile hardware. Built as a native mobile application with a companion wearable component, the system integrates LiDAR-based environment reconstruction, multi-directional raycasting for obstacle and hazard detection, spatialized audio rendering, wearable haptic output, a voice command interface using on-device speech recognition and foundation models, and a JSON-based configuration framework supporting runtime-swappable environment presets and fine-grained user customization. The architecture was developed through a complete migration from a cross-platform game engine to a native implementation, eliminating runtime overhead and reducing frame-to-feedback latency from approximately 150 milliseconds to under 50 milliseconds while maintaining 60 Hz operation. A technical evaluation of the stair and fall hazard detection subsystem across 120 trials spanning varied step heights, stair grades, lighting conditions, and approach directions yielded a 92.5% overall detection rate with no false positives and a mean alert distance of 2.31 feet. Detection performance generally increased with step height, with consistent detection above six inches, while descending approaches produced lower detection rates and shorter warning distances than ascending approaches due to fundamental geometric limitations of depth-based sensing. SensoryScape is designed as a complementary safety layer that augments established mobility aids, leveraging widely available consumer hardware to reduce cost, weight, and social acceptability barriers that have limited adoption of custom assistive devices.

PUBLIC ABSTRACT

For people who are blind or visually impaired, navigating an unfamiliar environment can be dangerous. Stairs, curbs, and sudden elevation changes are difficult to detect in time to avoid a fall, and existing assistive tools like white canes, guide dogs, and electronic devices each have real limitations in coverage, cost, or social acceptance. This thesis presents SensoryScape, a navigation assistance system that runs on an iPhone and Apple Watch. Rather than relying on a single mode of alert, SensoryScape combines directional sound cues, wrist vibrations, and voice interaction to give users a richer, more dynamic sense of their surroundings in real time. The system is designed to complement, not replace, the mobility aids people already rely on. A key goal of SensoryScape was accessibility in the broadest sense, including affordability. By building on consumer hardware most people already own, it avoids the cost and stigma often associated with specialized assistive devices. Users can also customize the system extensively and switch between different environment settings without technical expertise. Testing of the staircase and fall hazards detection capability across 120 trials showed the system correctly identified hazards 92.5% of the time, with no false alarms, giving users an average warning distance of about two and a half feet. Detection was most reliable for taller steps and when approaching from below. Detecting fall hazards from above proved more challenging due to the physics of how the depth sensor of phones work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Individuals who are blind or visually impaired face persistent challenges when navigating unfamiliar environments. Among these challenges, downward falls from stairs, curbs, and elevation changes, pose a risk of serious injury [17,18], as they offer little opportunity for recovery once a step is taken into open space. Traditional mobility aids such as white canes and guide dogs, while effective at detecting ground-level obstacles and providing basic orientation, offer limited information about obstacles above ground level, overhead hazards, pathfinding options in complex spaces, and dynamic environmental changes. Individuals with reduced peripheral vision often maintain relatively intact central vision but may fail to detect hazards or environmental cues appearing in the peripheral field of view. Consequently, they must continually scan their environment using their functional vision, a process that can be cognitively demanding. Furthermore, the adoption of traditional navigation assistance devices may be limited due to concerns about social stigma. In recent years, technological advancements in mobile sensing, wearable computing, computer vision, and spatial audio have opened new possibilities for supplementing traditional aids with real-time sensory feedback through assistive navigation systems.

These technological approaches often rely on sensory substitution, a technique where information normally acquired through one sense (e.g., vision) is conveyed through another modality (e.g., hearing or touch), enabling users to perceive environmental features via alternate sensory channels. However, existing solutions tend to focus on a single feedback modality, either haptic, or audio, limiting their adaptability to different user needs, preferences, and contextual demands. Furthermore, most systems fall into one of two categories: reactive systems that prioritize immediate obstacle detection and collision warnings, or proactive systems that model the environment to provide higher-level navigational guidance. In practice, safe and confident navigation requires both capabilities simultaneously.

Motivation

The motivation for this work emerged from direct experience with two prior assistive technologies developed within our research group: EchoSee and LEOBelt (Low-Vision

Enhancement Optoelectronic Belt). EchoSee is a mobile augmented reality application that uses smartphone LiDAR sensors to reconstruct spatial maps of the surroundings for users and delivers directional audio cues through spatialized sound. The system emphasizes exploratory navigation with minimal training requirements, enabling users to build passive environmental awareness through real-time auditory feedback. LEOBelt, by contrast, is a wearable haptic interface consisting of a belt-mounted LiDAR sensor and vibration motors distributed across the torso of the user. It focuses primarily on reactive obstacle avoidance, providing immediate tactile warnings to prevent collisions with nearby objects and critical hazards such as stairs and sudden elevation changes. Throughout this thesis, all such hazards involving stair edges, curbs, steps, and platform edges are referred to collectively as sudden changes in elevation.

While both systems demonstrated promise in user evaluations, they each addressed only part of the navigation challenge. EchoSee's reliance on auditory feedback and poses limitations in noisy environments and for users with hearing impairments, while requiring continuous headphone use that could isolate users from important ambient sounds. The haptic-only approach of LEOBelt, though effective for immediate threat detection, provided limited support for spatial exploration and longer-range pathfinding. Most critically, neither system alone adequately supported both continuous spatial awareness and rapid hazard detection—two capabilities that real-world navigation demands.

This observation motivated the development of SensoryScape, a unified multimodal assistive navigation system that integrates the strengths of both predecessor systems while addressing their individual limitations. By combining real-time environment reconstruction with both spatial audio and wearable haptics, SensoryScape aims to deliver a more adaptive, configurable and dynamic navigation solution. The system is designed to complement, rather than replace, existing mobility aids such as white canes or guide dogs, allowing users to integrate multimodal feedback with familiar navigational tools.

Research Problem

Individuals with low vision face significant challenges during independent navigation, particularly in environments containing hazards such as sudden elevation changes. Failure to

detect these hazards can result in dangerous downward falls. While mobile devices provide sensing and computational capabilities that could support assistive navigation, it remains unclear whether a multimodal navigation system combining spatial mapping with coordinated audio and haptic feedback can operate in real time on consumer mobile hardware with sufficiently low latency and responsiveness for practical use.

Hypothesis

This thesis tests the hypothesis that a multimodal navigation architecture integrating spatial mapping with coordinated audio and haptic feedback can operate in real time on consumer mobile devices with sufficiently low latency to provide effective navigation assistance for individuals with low vision.

Thesis Objective

This thesis evaluates methods for detecting sudden elevation changes and assesses their performance across a range of real-world conditions as part of a multimodal navigation system designed for mobile devices.

This evaluation encompasses several interrelated technical challenges:

1. **Real-time Performance on Mobile Hardware**: Achieving 60 Hz operation while performing continuous spatial mesh updates, multi-directional raycasting, semantic surface classification, and concurrent audio/haptic rendering on resource-constrained mobile devices.
2. **Cross-Platform Architecture Migration**: Migrating a cross-platform augmented reality (AR) application to a native implementation while preserving functionality, improving performance, and enabling deeper integration with platform-specific frameworks for device communication, audio management, and speech recognition.
3. **Multimodal Feedback Coordination**: Designing and implementing a system architecture that synchronizes spatial audio rendering, wearable haptic output, and voice command processing without introducing latency, audio session conflicts, or communication bottlenecks.

4. **Spatial Feature Detection**: Developing computationally efficient algorithms for detecting critical navigation hazards (stairs, sudden elevation changes, narrow passages) that operate within real-time constraints while handling LiDAR sensor limitations.
5. **Modular Configuration System**: Implementing a flexible framework that enables runtime reconfiguration of raycast patterns, audio profiles, and haptic schemes to support different environment types and use cases without application restart.
6. **Adaptability**: User-level control of system settings and feedback modalities, enabling feedback to be tailored to varying levels of vision and/or hearing loss.

System Design

This thesis makes several specific technical contributions to the field of assistive navigation technology:

Native Platform Migration with Performance Optimization: The EchoSee spatial mapping pipeline was migrated from a cross-platform game engine to a native implementation, providing direct access to the host platform scene reconstruction and semantic segmentation capabilities. This migration eliminated the intermediary runtime overhead, reduced frame-to-feedback latency from approximately 150 ms to under 50 ms, and enabled use of platform-optimized rendering paths for spatial audio.

Multimodal Feedback Architecture: A distributed system architecture was designed and implemented coordinating smartphone-based spatial audio with wearable haptic output. This required solving challenges in real-time data serialization, inter-device message throttling, and audio session management to maintain continuous feedback across both modalities without perceptible lag or dropped cues.

On-Device Natural Language Interface: A voice command system was developed using the Apple Speech framework and Apple Intelligence foundation models that enables hands-free configuration changes. The implementation required careful audio session orchestration to allow simultaneous speech recognition and spatial audio playback, preventing common iOS audio session conflicts that would mute navigation cues during voice interaction.

Stair and Elevation Change Detection Algorithm: A lightweight geometric analysis pipeline was implemented using downward-facing raycasts and depth-gradient heuristics to detect floor boundaries and step-like structures in real-time. Because downward falls represent injury risk for users of assistive navigation systems, the evaluation presented in this thesis focuses specifically on characterizing the subsystem detection efficacy across varying step heights, stair grades, lighting conditions, surface types, and approach directions. The algorithm balances detection sensitivity with computational efficiency, operating within the 16.7ms per-frame budget required for 60 Hz responsiveness.

JSON-Based Configuration Framework: A modular preset system was created enabling environment-specific raycast patterns, audio profiles, and haptic schemes. Presets are hot-swappable at runtime through both UI controls and voice commands, with validation logic preventing invalid configurations that could compromise system stability or user safety.

CHAPTER 2: BACKGROUND AND LITERATURE REVIEW

This chapter provides the technical foundation for understanding the SensoryScape system architecture and implementation. First is an examination of the two distinct predecessor systems, LEOBelt and EchoSee, by focusing on their technical designs, performance characteristics, and architectural limitations. Key technical concepts are described that underpin the SensoryScape implementation, including the ARKit spatial mapping capabilities, LiDAR sensing principles, raycasting techniques, and multimodal rendering approaches. Finally, related work in real-time AR systems and multimodal assistive technologies is reviewed, establishing the technical context and identifying gaps that motivated the integration approach presented in this thesis.

LEOBelt: Technical Architecture and Limitations

LEOBelt was designed as a wearable obstacle avoidance system using haptic feedback to encode spatial proximity information [1]. The system architecture consisted of three primary hardware components: a belt-mounted Intel RealSense D435i depth camera, a Raspberry Pi 4 Model B edge computing device, and eight coin vibration motors (10mm diameter, 3V DC) arranged across a compression vest worn on the torso of the user.

The depth camera provided real-time 3D depth information using Intel RealSense technology, capturing frames at either 30 fps (640×480 resolution) or 60 fps (848×480 resolution) depending on USB connection bandwidth. The Raspberry Pi processed this depth data using a custom C++ pipeline with OpenMP parallelization for efficient computation. The processing pipeline divided the camera's field of view into eight spatial sections corresponding to the eight motor positions around the user's torso. For each section, the system analyzed depth pixels in grid blocks, identifying the closest pixel and calculating its angular position relative to the camera. These angles and depth values were converted to real-world coordinates through trigonometric transformation, enabling distance calculation to detected obstacles.

Motor intensity was calculated by mapping object distance inversely to vibration strength on a 0-255 scale. The system applied a threshold filter transmitting only intensity values between 102 and 255, suppressing feedback from distant objects beyond approximately 3 meters to

reduce false positives. Communication between the Raspberry Pi and haptic motors occurred through a two-stage protocol: RS232 serial at 57600 baud to a master controller, which then distributed commands to individual motor units via ESP-NOW wireless protocol. Each motor received PWM-controlled intensity values directly proportional to obstacle proximity, with closer objects producing stronger vibrations. The system achieved typical sensor-to-motor delays of 35-50ms, critical for reactive obstacle avoidance [1].

User studies demonstrated that LEOBelt improved mobility accuracy and confidence for individuals with severely restricted peripheral vision. When tested with sighted subjects wearing goggles simulating advanced retinitis pigmentosa (RP), the device produced a 44% mean reduction in navigation errors and a 77% mean increase in self-reported confidence [1]. For RP subjects with remaining visual fields between 10° and 20°, LEOBelt doubled mobility accuracy, with confidence scores also doubling in dim lighting conditions [1].

Despite these strengths, LEOBelt exhibited technical limitations. The fixed camera mounting position at waist height created blind spots for overhead obstacles and floor-level hazards below the camera's minimum range of approximately 0.3 meters. Stairs and sudden elevation changes were particularly challenging, as the depth camera's planar field of view could not reliably detect changes in floor elevation without significant geometric analysis that would exceed the system's computational budget. The infrared-based depth sensing approach suffered degradation in bright sunlight due to IR interference, limiting outdoor usability. Highly reflective surfaces such as glass and polished metal, as well as very dark materials, produced unreliable distance measurements or missing data in the depth image.

From a computational perspective, while 30 Hz operation was adequate for basic obstacle detection, more sophisticated processing such as object classification, trajectory prediction, or multi-frame temporal filtering risked frame rate degradation on the Raspberry Pi's processor. The custom vest and belt assembly required manual fabrication and regular maintenance of electrical connections, limiting scalability and preventing rapid prototyping of alternative feedback configurations. Perhaps most fundamentally, the depth-only approach provided no semantic understanding of obstacles, with users receiving identical haptic warnings for walls, furniture, people, or hanging obstacles. These limitations collectively motivated exploration of alternative

architectures leveraging more capable sensing hardware and richer environmental understanding [1].

EchoSee: Technical Architecture and Limitations

EchoSee approached assistive navigation from a mobile augmented reality perspective, leveraging the LiDAR and camera sensors available on iPhone models [2]. The system was built using the Unity game engine running on iOS, with the ARKit framework providing access to the Apple spatial mapping capabilities. The technical architecture operated as a multi-stage pipeline integrating AR session management, raycast-based obstacle detection, spatial audio rendering, and data logging.

The Unity AR Foundation package interfaced with ARKit to establish and maintain an AR session running continuously throughout application lifetime. This session provided access to mesh reconstruction, where ARKit generated triangle meshes representing detected surfaces and updated them incrementally as the device moved through space. Apple's machine learning models performed semantic segmentation to classify surfaces into categories including floor, wall, ceiling, door, window, and table. The session also maintained device pose tracking using six-degree-of-freedom position and orientation tracking at 60 Hz through visual-inertial odometry [2].

From the current position of the device, EchoSee projected directional rays into the reconstructed environment following a fixed geometric pattern. In the published implementation, eight rays were used: five forward-facing rays spread horizontally, two lateral rays at ±90 degrees, and one downward ray [2]. Each raycast queried the AR mesh for intersection points, returning the distance to the nearest surface along that direction. Ray queries were executed using Unity's Physics. Raycast API operating on mesh colliders automatically generated from ARKit's mesh data, achieving raycast frequencies of approximately 30-40 Hz.

Distance measurements from successful raycasts were translated into spatialized audio cues using Unity's audio engine in combination with Apple's Spatial Audio APIs. Each ray direction was assigned a persistent audio source playing a continuous tone. The audio rendering pipeline applied volume modulation using inverse linear scaling where closer obstacles produced

louder sounds, with objects within 0.5 meters playing at maximum volume while objects beyond 5 meters became inaudible [2]. Spatial positioning was achieved by placing audio sources in Unity's 3D coordinate space relative to the camera, automatically applying head-related transfer functions through iOS Spatial Audio to create directional sound perception through stereo headphones.

User studies with blindfolded sighted participants demonstrated EchoSee's effectiveness in navigation assistance. The system enabled participants to achieve substantial reductions in obstacle contacts (39.8% fewer collisions when assisted versus unassisted) and showed evidence of rapid learning effects [2]. Participants demonstrated increased environmental exploration behavior, measured through "seeking" (head- and phone-turning to scan the environment), with the seeking metric showing a positive slope of 2.28 in assisted trials compared to -1.11 in unassisted trials [2]. However, participants navigated more slowly with the system, with completion times increasing by 34% (50 seconds unassisted versus 67 seconds assisted). This suggests users spent additional time interpreting the soundscape rather than quickly traversing the course while tolerating contact with soft, inflatable obstacles posing minimal risk of injury [2].

Despite demonstrating feasibility, EchoSee exhibited technical limitations that prevented optimal performance and motivated the re-architecture presented in this thesis. The Unity engine introduced significant computational overhead, with profiling revealing that approximately 40-60% of CPU time per frame was consumed by Unity's internal systems rather than AR processing or application logic. This overhead manifested as frame rate instabilities in complex environments and increased latency between AR frame capture and raycast execution, typically ranging from 100-150ms [2]. Memory pressure from Unity's managed heap triggered garbage collection pauses causing perceptible audio dropouts.

Unity's cross-platform abstraction layer prevented direct access to iOS-specific APIs essential for advanced functionality. The framework provided no direct access to AVFoundation for fine-grained audio session control, preventing simultaneous spatial audio playback and speech recognition. Integration with WatchConnectivity for Apple Watch communication was impossible without custom native plugins. Access to Apple's Speech framework for voice

commands required similar plugin development. Unity's physics engine, designed for game simulation rather than real-time sensing, exhibited inconsistent raycast performance that degraded with mesh complexity, forcing design compromises that limited ray density [2].

ARKit's mesh reconstruction quality varied significantly with environmental conditions. Fast movement, poor lighting, or highly reflective surfaces degraded mesh accuracy, directly impacting raycast reliability. Unity provided no access to ARKit's raw confidence maps that could have filtered unreliable mesh regions. The downward-facing ray configuration proved insufficient for reliably detecting stairs or sudden elevation changes across varied geometries. iOS audio session management limitations prevented simultaneous use of playback and recording categories, making voice control functionally impossible without extensive native plugin development [2].

These technical constraints, combined with the architectural inflexibility imposed by Unity's runtime environment, motivated a complete re-architecture using native Swift and iOS frameworks to eliminate computational overhead, achieve direct API access, implement efficient algorithms, and enable multimodal integration that neither EchoSee nor LEOBelt could support independently.

Key Technical Concepts

LiDAR Sensing Principles

LiDAR (Light Detection and Ranging) measures distance by emitting pulses of infrared light and precisely timing their return after reflecting from surfaces. The sensor emits light pulses and measures the round-trip time for each pulse, with nanosecond timing precision enabling millimeter-scale distance calculation. Mobile LiDAR sensors integrated into consumer smartphones project grid patterns of several hundred sample points at rates up to 60 Hz, with effective ranges from approximately 0.1 meters to 5 meters under optimal conditions [2,3].

LiDAR sensing exhibits characteristic strengths and weaknesses. The technology excels at capturing geometric detail on matte, diffusely reflecting surfaces like painted walls and wood furniture. Performance degrades significantly with specular or transparent materials including

glass and mirrors, which reflect infrared light away from the sensor, and very dark materials that strongly absorb infrared light. Environmental factors also affect performance—bright sunlight introduces noise, while dust and fog scatter light, attenuating signals. Despite these limitations, LiDAR provides substantially better depth sensing than camera-based approaches in low light conditions [2,3].

Spatialized Audio Rendering

Spatialized audio creates the perception that sounds originate from specific locations in 3D space around the listener. This effect is achieved by applying acoustic transformations that mimic how sound waves interact with the human head and outer ear. The primary mechanism is the head-related transfer function (HRTF), which describes how sounds from different directions are filtered by the anatomy of the listener before reaching the eardrum. Interaural time differences, level differences, and spectral shaping combine to enable perception of sound direction and distance [2].

Modern mobile platforms provide HRTF-based spatial audio rendering frameworks that accept 3D sound source positions and automatically compute appropriate filtering and delays for left and right audio channels. These frameworks support dynamic updates at audio frame rates, enabling smooth position changes as virtual sources move in space. Some consumer headphones include inertial sensors that track head orientation, allowing real-time adjustment to maintain world-locked spatial positions independent of head movement [3].

Distance perception in spatial audio relies primarily on intensity attenuation, where sounds become quieter with increasing distance. While natural environments also provide reverberation cues, simpler intensity-based approaches trade acoustic realism for clarity and rapid interpretation [2].

Haptic Feedback Rendering

Haptic feedback communicates information through the sense of touch using vibration actuators in wearable devices. Modern smartwatches and wearable devices commonly employ linear resonant actuator technology capable of producing discrete taps, continuous vibrations,

and complex haptic textures. Platform-specific frameworks provide APIs for triggering predefined patterns designed for different notification types and interaction contexts, as well as lower-level control for custom haptic event design [3].

Haptic rendering for assistive applications typically distinguishes between continuous events, which sustain vibration with controllable intensity modulation, and transient events, which produce brief, sharp taps. Research in haptic perception has established that vibration intensity must differ by at least 20-30% for reliable discrimination, temporal patterns require minimum 50-100 millisecond inter-stimulus intervals, and simple rhythmic patterns achieve higher recognition accuracy than complex patterns [1,4].

Related Work and Integration Needs

Recent research in assistive navigation shows growing recognition that multimodal systems combining auditory, haptic, and visual feedback offer advantages over single-modality solutions for individuals with visual impairments [1,2,3]. However, most existing systems fall into one of two categories: reactive systems prioritizing immediate obstacle detection and collision warnings (like LEOBelt), or proactive systems modeling environments to provide higher-level navigational guidance (like EchoSee). In practice, safe and confident navigation requires both capabilities simultaneously [1,2].

One technically sophisticated multimodal system combines stereo vision cameras, inertial sensors, and deep learning-based object recognition in a backpack-worn configuration integrating with a smart belt and audio headset [3]. The system offers simultaneous object classification, obstacle detection, and spatial auditory feedback. Despite its technical strengths, the system weighs approximately 4-5 kilograms, requires dedicated GPU-accelerated computing hardware, and consumes power at rates requiring battery replacement every 2-3 hours [3]. Nevertheless, the work demonstrates that layered sensory feedback combining multiple modalities provides superior environmental awareness compared to single-modality approaches, validating the multimodal approach pursued in SensoryScape.

Spatial audio navigation has been explored extensively in virtual reality training contexts. The VR stereo-sonic guidance system created realistic acoustic simulations of indoor and

outdoor environments, allowing blind users to practice navigation skills in safe, controlled settings [6]. Studies demonstrated that directional audio significantly improved navigation accuracy and confidence, especially when paired with spatial anchors. However, research also highlighted that spatial audio alone often lacks immediacy for sudden obstacle warnings, requiring users to actively attend to and interpret subtle acoustic cues that might be missed during distraction or cognitive load [6,7]. This finding underscored the value of adding a fast, attention-capturing channel such as haptics for critical hazard warnings. Research specifically addressing haptic feedback includes the NavBelt system, which used ultrasonic rangefinders to detect obstacles and conveyed distance information through vibration intensity, achieving response times under 100 milliseconds [4]. These works demonstrate the effectiveness of haptic feedback in maintaining user safety and increasing confidence, particularly in indoor environments [1,5].

Several applications have attempted to bring assistive navigation technology to consumer devices. The vOICE system provides image and video sonification, converting images to sound by associating elevation with pitch and brightness with loudness, though it relies on purely 2D image processing without depth information [9]. Microsoft's SeeingAI app provides features including speech-to-text and scene description, though several services require internet connectivity [10]. EchoSee demonstrated the feasibility of LiDAR-based spatial audio navigation on consumer smartphones, validating the user-locked coordinate system approach where sounds move with the perspective of the user [2]. User feedback indicated that the sweeping motion paradigm was easy to learn, with participants demonstrating increased scanning behavior over successive trials without explicit instruction to do so [2]. Although the Unity-based implementation suffered from computational overhead and limited iOS framework integration, with 40-60% of CPU time consumed by Unity's internal systems rather than application logic [2].

Recent systematic reviews of electronic travel aids highlight persistent challenges in achieving widespread adoption. A 2024 review analyzing 73 studies found that while numerous devices show promise in controlled settings, factors including device weight, battery life, cost, training requirements, and social acceptability significantly impact real-world usage [11]. The review emphasized that devices developed without substantial input from people with visual

impairments often fail to meet actual navigation needs, and that successful devices tend to leverage existing consumer hardware rather than requiring custom equipment, reducing both cost and social stigma [11,12].

The technical literature addresses specific algorithmic challenges relevant to the SensoryScape implementation. Stair detection using depth sensors has been investigated through plane-fitting algorithms that identify step-like geometric structures and machine learning classifiers. Plane-fitting approaches achieve detection rates of 80-90% for standard staircases but struggle with irregular stairs and require computational resources that strain real-time budgets on mobile processors [2]. Learning-based approaches achieve higher accuracy (92-96%) but impose even greater computational demands [13]. Research in multimodal feedback design has established that the most effective designs employ complementary information encoding, where each modality conveys different aspects optimized for that channel's perceptual strengths rather than providing redundant information [1,3,14]. This principle guided the feedback design of SensoryScape, using audio for directional environmental awareness while reserving haptics for immediate proximity warnings.

Despite these innovations, several gaps remain. Real-time performance is essential, with latencies above 100-200 milliseconds degrading usability, yet systems requiring cloud processing struggle to meet these constraints [2,3,11]. While multimodal feedback provides superior awareness, most systems still emphasize a single primary modality with limited integration of complementary channels [1,3]. Custom hardware approaches limit scalability despite sometimes offering superior capabilities [1,11]. Few systems adequately address detecting floor-level hazards such as stairs while maintaining real-time performance on mobile hardware [1,2], and most have been tested primarily with sighted blindfolded participants in controlled environments rather than with the target population in real-world settings [2,11]. These insights collectively motivated the architecture of SensoryScape emphasizing native performance optimization, multimodal integration using consumer devices, and efficient real-time algorithms for comprehensive hazard detection including stairs and sudden elevation changes.

CHAPTER 3: METHODS

SYSTEM ARCHITECTURE AND IMPLEMENTATION

This chapter presents the technical design and implementation of SensoryScape, detailing the distributed dual-device architecture, spatial mapping pipeline, multimodal feedback generation, voice command system, and configuration framework. The system is built as a native iOS and watchOS application leveraging ARKit for real-time environmental reconstruction, SceneKit for spatial audio rendering, and WatchConnectivity for haptic feedback delivery. The architecture emphasizes modularity and real-time performance to support future research and customization. Following the architectural overview, we detail the specific implementation approaches used to achieve 60 Hz real-time operation, overcome iOS framework constraints, and optimize system stability.

Overview and Design Principles

SensoryScape operates as a distributed platform composed of an iOS application running on an iPhone and a companion Apple Watch application. Together, these components provide multimodal navigation support by combining real-time environmental perception, spatialized audio feedback, and wearable haptic cues. The system has been designed around an architecture that balances computational efficiency and user responsiveness. Using Bluetooth for communication with feedback components, alternative devices could conceivably be substituted, though the current implementation targets widely available consumer hardware to maximize accessibility.

The Apple hardware and iOS software environment were selected as the implementation platform for several converging reasons. First, the Apple LiDAR-equipped iPhone models provide integrated depth sensing capable of real-time scene reconstruction at consumer price points, a capability not yet widely available on competing mobile platforms. Second, the iOS ecosystem offers tightly integrated accessibility features, including VoiceOver screen reading and assistive touch modes, that align with the needs of the target user population. Third, the Apple Spatial Audio framework and AirPods Pro hardware provide built-in support for head-tracked spatialized audio rendering, eliminating the need for custom audio hardware. The

AirPods Pro also offer sound "pass-through" so that they do not impair environmental sounds. Fourth, the Apple Watch offers commercially available wrist-worn haptic output through its Taptic Engine, providing a practical alternative to custom haptic hardware such as the fabricated vibration vest of LEOBelt. Finally, the prior EchoSee system had already been prototyped within the iOS and Unity environment, making iOS the natural platform on which to build and extend that work. While the architectural principles and algorithms described in this thesis are not inherently platform-specific and could be adapted to other environments such as Android with appropriate depth-sensing hardware, the iOS ecosystem currently provides the most complete integration of the sensing, audio, haptic, and accessibility capabilities required by the system.

The iOS application serves as the primary computational engine, performing real-time spatial mapping, raycast-based obstacle detection, audio rendering, and data logging. Built using native Swift with ARKit and RealityKit frameworks, the application eliminates the runtime overhead present in the Unity-based EchoSee predecessor while enabling direct access to iOS-specific APIs for multimodal integration. The application continuously updates a 3D reconstruction of the user's surroundings as they move through space, projects directional raycasts to evaluate distances to nearby environmental geometry and modulates audio and haptic feedback in real-time based on detected obstacles and floor boundaries.

The Apple Watch application functions as a haptic output interface, receiving proximity data from the iPhone and translating it into vibration patterns on the wrist of the user. The phone itself can also provide haptic vibration patterns. Communication between devices is managed using WatchConnectivity framework, supporting both persistent state synchronization and real-time message passing. Feedback transmission is throttled to maintain system stability while preserving responsiveness for critical hazard warnings. The watch was chosen for its practicality as it is widely available, lightweight, and seamlessly integrates into daily use whereas LEOBelt's custom vest, though offering richer spatial resolution, required specialized fabrication.

The architectural design follows several key principles. First, all processing occurs on device without requiring internet connectivity, ensuring consistent performance across diverse environments and eliminating latency introduced by cloud services. Second, the system maintains a strict real-time constraint, targeting 60 Hz operation for continuous feedback without

perceptible lag. Third, the architecture employs unidirectional data flow where environmental inputs flow through raycasting and voice recognition into centralized state management, with outputs generated based on current state. This model simplifies debugging and ensures deterministic behavior under real-time constraints. Fourth, asynchronous processing is employed throughout to prevent blocking operations, particularly during AR frame analysis, audio playback, and inter-device communication.

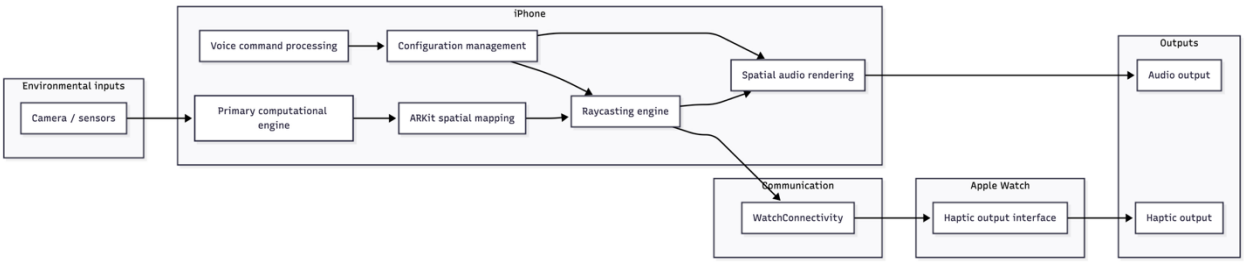

Figure 1: High-Level System Architecture Diagram

iOS Application Architecture

The iOS application is structured as a layered system comprising core data models, service managers, view models, and SwiftUI-based views. This separation of concerns promotes modularity and testability while enabling reactive updates through the Combine framework. The data models define structures for raycast results, audio state, configuration options, floor detection parameters, and voice command parsing. Managers implement key services including AR session control, spatial audio rendering, Watch communication, data logging, and AI interaction. These services are loosely coupled to view models using dependency injection, allowing individual components to be tested and modified independently.

The application leverages the Apple ARKit and RealityKit frameworks as the foundation for spatial perception. ARKit provides high-level abstractions for world tracking, scene reconstruction, and spatial understanding [3]. World tracking fuses data from the device camera, inertial measurement unit (IMU), and LiDAR depth sensor to estimate device position and orientation in 3D space, operating at 60 Hz with typical positional accuracy of ±1-2 centimeters over short durations through visual-inertial odometry [3]. On LiDAR-equipped devices, ARKit generates triangle mesh representations of the physical environment organized as

ARMeshAnchor objects, each representing a localized surface segment typically 1-2 meters across, containing vertex positions, face indices, and optional semantic classification labels. Semantic segmentation applies on-device machine learning models to classify mesh regions into architectural categories including floor, wall, ceiling, door, window, table, and seat, with typical accuracy exceeding 90% for major structural elements in well-lit indoor environments [3]. RealityKit extends ARKit by providing scene understanding APIs that offer convenient access to raycasting functionality through higher-level interfaces that automatically handle coordinate transformations [3].

The AR session manager wraps the ARKit session lifecycle, configuring the framework for world tracking with scene reconstruction and semantic segmentation enabled. The configuration enables both horizontal and vertical plane detection to support enhanced environmental understanding, though the primary obstacle detection relies on direct mesh raycasting. When the session begins, ARKit initializes its coordinate system arbitrarily based on the initial position of the device and orientation. To enable cross-trial comparison during research studies, a fiducial marker registration system was implemented. A physical calibration panel is placed in a consistent location and linked to a virtual object within the ARKit coordinate system. When the marker is observed, the virtual object is automatically positioned at the corresponding real-world location, defining both an origin and neutral rotation. During operation, the application records device position and orientation at each frame, later rotating and translating these data during post-processing to align all sessions to a common reference frame.

The spatial mapping pipeline operates continuously as the user moves through their environment. The iPhone LiDAR sensor projects a grid pattern of approximately 576 sample points updated at 60 Hz, with an effective range from approximately 0.1 meters to 5 meters under optimal conditions. ARKit uses this depth data along with camera imagery to generate triangle meshes representing detected surfaces, organized as ARMeshAnchor objects containing vertex positions, face indices, and semantic classification labels. The mesh updates incrementally, with new geometry generated for previously unseen areas and existing meshes refined as additional observations accumulate. Mesh quality varies with environmental conditions—fast movement, poor lighting, or highly reflective surfaces degrade reconstruction accuracy. To maintain real-time performance, the system processes mesh updates

asynchronously, updating the raycasting internal representation of the raycasting engine without blocking the main rendering thread.

Raycasting and Obstacle Detection

At the center of the environmental awareness capabilities of SensoryScape is the raycasting engine, which projects invisible rays into the reconstructed 3D environment to detect obstacles and measure distances. The location in the environment that the raycast "touches" becomes a virtual sound "source." Closer surfaces can have higher pitch, and/or higher volume. The current implementation uses eight raycasts arranged in a configuration designed to provide comprehensive coverage of navigationally relevant regions (Figure 3). Three sources are positioned across the horizontal plane at the same vertical height: a Forward source at 0° offset, and Left and Right sources at ±30° offset, all playing a G4 tone (392 Hz). An Up source at 20° vertical offset plays a B4 tone (493.88 Hz) to detect overhead obstacles. Four downward-facing rays address floor-level awareness and hazard detection: a Lower ray and a Floor Detector ray, both silent, provide ground-plane tracking and calibration data, while Down1 at 30° downward offset plays an E4 tone (329.63 Hz) and Down2 at 60° downward offset plays a C4 tone (261.63 Hz) to provide tripping hazard detection and floor boundary awareness.

Each frame, the raycasting engine casts these eight rays from the current position of the device into the AR mesh. The optimized raycast implementation of ARKit leverages spatial acceleration structures to achieve query times of 0.1-0.5 milliseconds per ray, well within the 16.7 millisecond per-frame budget required for 60 Hz operation. When a ray intersects mesh geometry, the system records the hit position, distance from origin, surface normal, and semantic classification if available. These intersection results are then used to update the positions of virtual audio sources within the 3D scene.

The Forward source communicates range-to-target information, providing users with direct feedback about obstacles ahead. The Left and Right sources serve as substitutes for peripheral vision, analogous to trailing one's hand along a wall for orientation. The Up source guards against overhanging hazards such as low branches or architectural features. The four downward-facing rays address one limitation of LEOBelt by providing dedicated detection of

floor-level hazards including stairs, curbs, and sudden elevation changes, with the silent Floor Detector and Lower rays supplying ground-plane data for the elevation change detection algorithm described in a subsequent section.

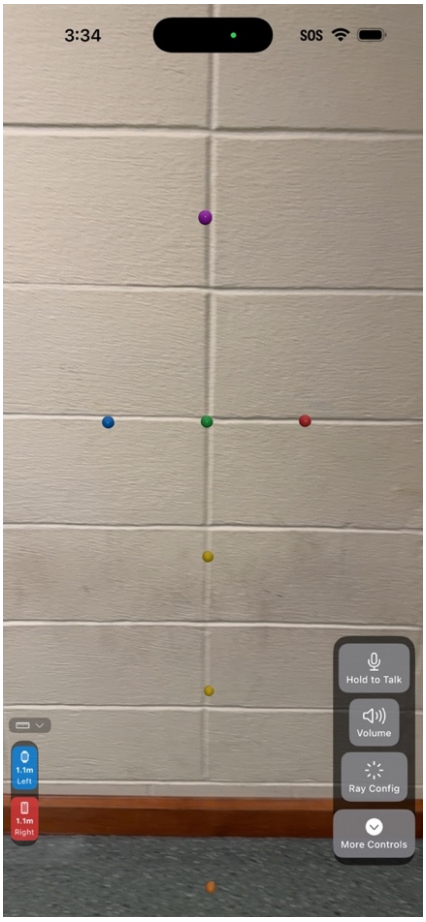

Figure 2: Screenshot of SensoryScape with Ray Visualizations

The colored spheres overlaid on the live camera feed represent the intersection points where each raycast detects a surface in the reconstructed environment. Each sphere is color-coded to a specific ray direction: the purple sphere near the top of the screen corresponds to the Up ray, the blue sphere on the left to the Left ray, the green sphere at center to the Forward ray, the red sphere on the right to the Right ray, the two yellow spheres below center to the Down1 and Down2 rays respectively, and the orange ray representing the Lower ray.

Spatial Audio Rendering

Spatial audio feedback is rendered using the SceneKit audio system integrated with the ARKit spatial tracking. The system creates SCNAudioSource instances corresponding to each raycasting direction, with each source configured to play a continuous pure tone at its assigned frequency. These audio sources are positioned in 3D space within the SceneKit scene graph, which automatically applies spatial audio transformations based on the position of the listener and orientation tracked by ARKit.

The implementation leverages the Apple Spatial Audio framework, which provides HRTF-based rendering using a standardized transfer function optimized for average listener anatomy. The framework supports dynamic updates at audio frame rates of 48,000 samples per second, enabling smooth position changes as virtual sources move. When paired with AirPods Pro or other Apple headphones equipped with inertial sensors, the framework tracks head orientation in real time, allowing world-locked spatial positions to be maintained independent of head movement. This head-tracking capability enhances the realism and stability of spatial cues during navigation, though the system remains functional with standard stereo headphones at reduced spatial fidelity.

The SceneKit audio engine handles the low-level details of spatial audio rendering, applying head-related transfer functions and managing the relationship between virtual sound source positions and the tracked head position of the user. Volume and panning for each audio source are dynamically adjusted based on raycast intersection results. Distance mapping applies volume attenuation where closer obstacles produce louder sounds, and more distant obstacles fade toward silence. The system implements clamping and scaling logic to ensure that very close objects (within 0.5 meters) trigger higher volume, while objects beyond the maximum ray range become inaudible.

Rather than being world-anchored, audio sources are user-locked, moving with the perspective of the device. This design allows users to sweep the phone across their environment and immediately hear differences in the spatial layout, effectively painting a sonic picture of nearby obstacles through motion. The audio source positions are updated each frame based on

raycast intersection results, with the SceneKit audio system automatically handling the spatial transformations required for realistic directional audio perception. While world-anchored sound models where cues remain fixed to objects in space could provide complementary benefits for landmark-based navigation, the current user-locked approach emphasizes relative awareness and rapid feedback for immediate obstacle avoidance. World-anchored modes represent a potential extension for future versions.

The integration with the SceneKit rendering pipeline ensures that audio updates remain synchronized with visual tracking even though SensoryScape does not render visual output. This synchronization maintains audio-spatial consistency, preventing the disorienting effect that would occur if audio sources appeared to drift relative to the physical environment during device movement. The audio system operates on the SceneKit update cycle, which runs in coordination with the ARKit frame delivery, ensuring that spatial audio transformations reflect the most recent tracking data.

Apple Watch Integration and Haptic Feedback

SensoryScape employs a distributed haptic feedback strategy using both the iPhone and a companion Apple Watch application to convey directional proximity cues. The left-facing raycast is paired with the Apple Watch haptic motor, while the right-facing raycast is paired with the built-in haptic engine of the iPhone. This asymmetric dual-device design enables users to infer obstacle directionality through the location of the vibration itself, providing an extension of peripheral awareness.

The Apple Watch application functions as a dedicated haptic output interface for left-side proximity cues. It receives distance measurements from the iPhone corresponding exclusively to the left raycast and translates them into vibration patterns on the wrist of the user. Communication between devices is managed using the WatchConnectivity framework, which supports both real-time interactive messaging and background state synchronization. Interactive messages are used to transmit left-ray proximity updates formatted as lightweight key-value dictionaries containing two fields: "intensity" (a floating-point value from 0.0 to 1.0) and

"pattern" (an enumeration specifying pulse timing). Message payloads are kept minimal, typically under 50 bytes, to reduce serialization overhead and transmission latency.

In parallel, the iPhone generates haptic feedback for right-side proximity cues using Core Haptics. The right-facing raycast drives vibration intensity and repetition rate directly on the phone through CHHapticEngine, enabling immediate, low-latency feedback without requiring inter-device communication. The haptic engine is configured with distance-based intensity mapping where closer obstacles trigger stronger vibrations. This division of responsibility reduces messaging overhead to the watch and ensures that critical right-side obstacle cues remain responsive even if connectivity is temporarily degraded.

Proximity is encoded through vibration intensity and repetition rate for both devices. When the paired raycast detects a close obstacle within approximately 1 meter, the corresponding actuator delivers a strong pulse at high repetition rate. As obstacle distance increases, intensity decreases and repetition rate slows proportionally. Beyond the maximum detection range, vibration ceases entirely. Each actuator operates independently based on its assigned raycast, avoiding composite or collapsed haptic patterns that could obscure directional information. The haptic patterns are generated using the CoreHaptics transient event type, which produces short, percussive taps rather than continuous rumbling, providing clearer temporal resolution for users to distinguish individual pulses.

This dual-channel approach parallels haptic signaling strategies used in multimodal navigation aids that preserve spatial correspondence between stimulus location and physical feedback. By mapping left and right rays to physically distinct actuators, SensoryScape leverages the innate ability of the user to localize tactile cues, reducing cognitive load and accelerating the learning curve compared to single-channel haptic encoding schemes that require learned associations between abstract vibration patterns and spatial directions.

The haptic rendering layer relies on the Apple Core Haptics framework on the iPhone and WatchKit haptic APIs on the Apple Watch. Core Haptics provides lower-level control through CHHapticEngine, allowing specification of continuous and transient haptic events with precise timing and intensity parameters. The Apple Watch Taptic Engine, based on linear resonant

actuator technology, supports predefined haptic patterns as well as custom sequences triggered through WatchKit APIs. These platform-specific capabilities enable the fine-grained intensity modulation and temporal patterning described above, translating abstract proximity values into perceptually distinct tactile cues.

To balance responsiveness with communication stability, watch-based haptic updates are throttled to 0.5-second intervals rather than being transmitted every frame. This throttling is implemented using a simple time-based gate that compares the current timestamp against the timestamp of the last successful transmission, only allowing messages through when the elapsed time exceeds 500 milliseconds. The approach reduces message traffic from a theoretical maximum of 60 messages per second to a constant 2 messages per second, while maintaining adequate temporal resolution for proximity warnings. The watch application maintains a finite state machine with three states (Idle, ActiveNear, ActiveFar) tracking the most recent haptic command and implements smooth transitions between vibration patterns to avoid abrupt changes that could confuse users or feel mechanically harsh.

The modular architecture ensures that alternative haptic devices could be integrated with minimal modification. The iPhone HapticManager application implements a protocol-based interface defining methods for setIntensity and setPattern, allowing specific device implementations to translate abstract proximity data into device-specific commands. While the current implementation targets a dual-device configuration using the Apple Watch and iPhone, future extensions could support richer wearables such as haptic vests or belts offering greater spatial resolution through the same protocol-based abstraction.

Sudden Elevation Change Detection

SensoryScape detects sudden elevation changes and tripping hazards through raycast-based geometry inference rather than mesh analysis. Performing in-depth mesh analysis for even a small section of the reconstructed environment can severely throttle runtime performance, largely eliminating the benefits that would otherwise come from the increased quantitative knowledge of where and what elevation change and tripping hazards exist. To avoid this potential throttling, the system leverages the existing raycasts, using the resulting hit points to

infer spatial structure. Between each pair of rays cast in the vertical plane, a line segment can be constructed from their respective 3D hit positions. Based on the geometric properties of these line segments—their angles, slopes, and spatial relationships—the system infers the presence of sudden elevation changes, stairs, and tripping hazards.

Both detection layers depend on a stable floor height estimate. The system maintains a calibrated floor height and an associated confidence value derived from downward-facing rays—including a dedicated Floor Detector ray—that intersect geometry classified as "Floor" by the ARKit semantic segmentation. Only hits that fall below the current camera height and carry a floor classification are admitted to a rolling sample buffer. When sufficient samples have accumulated, the system computes the median of recent values as the calibrated floor height, selecting the median over the mean to reduce sensitivity to outliers from transient mesh reconstruction errors. Confidence is derived from the variance of the sample buffer, with tighter spread yielding higher confidence. The floor height can also be set manually from the current camera height for environments where semantic classification performs poorly.

For descending elevation change detection, line segments are evaluated each frame. The segment between the Down2 and Floor Detector rays (Figure 4) is evaluated for its angle relative to the position of the user; if it slopes downward away from the user, the system infers the presence of a descending elevation change such as a curb or ledge. The slope-away direction is determined using a dot product between the segment direction and the forward vector of the camera, ensuring that only geometry descending ahead of the user triggers detection rather than surfaces rising toward the user. The segment between the Forward and Lower (Figure 4) rays similarly captures the lip at the edge of a descending elevation change, where the Forward ray intersects the top surface and the Lower ray intersects geometry below the edge. The segment between the Down1 and Down2 rays (Figure 4) provide a further descending elevation change indicator when it slopes downward away from the user. For a segment to qualify as an elevation change candidate, it must span a minimum vertical distance of 4 centimeters and its tilt from vertical must fall within 5° to 50°, a range characteristic of step and curb geometry. A general case also requires that at least two independent forward-and-down ray pairs satisfy the same geometric criteria simultaneously, confirming detection in ambiguous configurations where a single segment alone might be unreliable.

To reduce false positives from transient mesh noise or momentary scanning artifacts, a detection must persist across four consecutive frames before an alert is issued. At least two rays in the elevation change detection set must return valid hits for the line-based analysis to execute. Alerts are rate-limited and subject to a cooldown period to prevent the system from repeatedly announcing the same hazard as the user continues to approach it.

Tripping hazards and ascending stairways are detected using the same line-based geometric approach, applied to a different set of ray pairings. Where descending elevation change detection evaluates geometry descending away from the user, tripping-hazard and stair detection evaluates geometry rising ahead of the user. The system constructs two line segments: a lower segment between the Lower and Forward hit points, and an upper segment between the Forward and Up (Figure 5) hit points. For each segment the angle from vertical is computed, and the relationship between the two angles determines the classification.

If the upper segment is nearly vertical—within 25° of vertical—the configuration is interpreted as a wall rather than a navigable elevation change, and no alert is issued, as walls are handled by the general obstacle detection and audio feedback system described in preceding sections. This wall rejection step is necessary because a flat vertical surface can produce a ray hit pattern that superficially resembles stair geometry. If the upper segment is approximately horizontal (65° or more from vertical) while the lower segment falls within 10° to 50° from vertical, the configuration is classified as a tripping hazard, corresponding to a single step up or raised curb where the lower segment captures the vertical rise and the upper segment captures the horizontal landing surface above it. If both segments fall within the same 10° to 50° range and their angles match within a tolerance of 15°, the geometry is classified as a staircase. The angular similarity indicates a repeating step pattern—both the lower and upper portions of the visible geometry exhibit the same characteristic stair slope—rather than a single isolated step. A confidence value is computed from the deviation of the measured angles from a typical stair angle of 34°, with closer matches producing higher confidence.

When either a tripping hazard or staircase is detected, the system generates a spoken alert indicating the hazard type and its angular offset from the heading of the user using AVSpeechSynthesizer, along with a three-pulse haptic pattern delivered through Core Haptics.

The three-pulse pattern is perceptually distinct from the continuous proximity pulses used for general obstacle feedback, enabling users to distinguish elevation-change warnings from routine distance cues without requiring visual confirmation. Tripping-hazard and stair detection is rate-limited independently of descending elevation change alerts. The user can select from four alert modes—descending elevation changes only, tripping hazards only, both, or neither—through a unified configuration setting, allowing the system to be tuned to the navigation environment and reducing alert fatigue in contexts where certain hazard types are not relevant. All thresholds throughout the subsystem—the angle bands, persistence frame count, minimum segment span, and elevation change depth—were determined through empirical testing to limit false positives arising from normal gait dynamics and scanning motion while maintaining useful detection sensitivity.

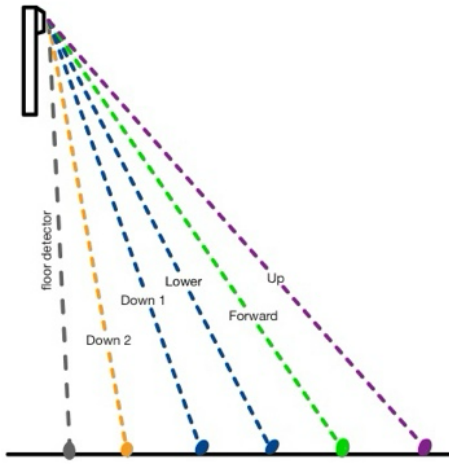

Figure 3: Ray Position Example on Flat Ground with Defined Ray Labels

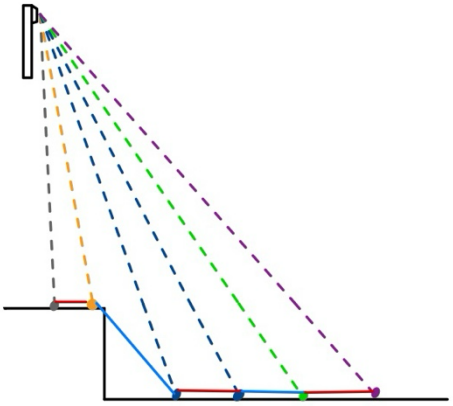

Figure 4: Ray Position for Descending Tripping Hazard (ray labels as defined in Figure 3)

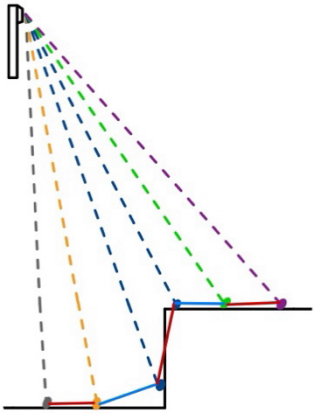

Figure 5: Ray Position for Ascending Tripping Hazard (ray labels as defined in Figure 3)

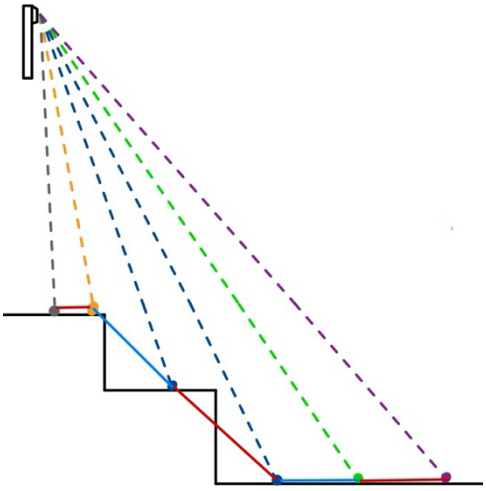

Figure 6: Ray Position for Descending Staircase (ray labels as defined in Figure 3)

Voice Command System and Audio Session Management

To support hands-free interaction, SensoryScape incorporates a voice command subsystem enabling users to modify configuration settings, activate or deactivate raycast directions, and switch environment presets through natural language. The system uses the Apple Speech framework for wake-word detection and speech-to-text conversion, integrated with the Apple FoundationModels framework for intent parsing and command validation.

The voice interaction pipeline operates through five sequential stages. First, the application continuously monitors audio input for a wake phrase ("Hey SensoryScape") or responds to a push-to-talk button press on the device screen. The wake-word detection uses SFSpeechRecognizer in continuous recognition mode with a low-latency configuration optimized for short phrases. Second, when voice input is detected, the Speech framework transcribes spoken words into text with real-time partial results appearing during speech and final confirmed transcription delivered when the user pauses for more than 1.5 seconds. Third, the transcribed text is sent to the Apple FoundationModels framework with a structured prompt describing available commands ("switch to indoor mode", "increase volume", "disable upper ray", etc.) and current system state including active preset, volume level, and ray activation status. Fourth, the model returns a parsed intent formatted as a JSON object specifying the requested action (e.g., "changePreset") and any parameters (e.g., {"preset": "indoor"}). Fifth, the application validates the command against current configuration constraints using the validation logic of ConfigurationManager and executes the change if valid, providing audio confirmation through text-to-speech synthesis saying "indoor mode activated" or similar acknowledgment.

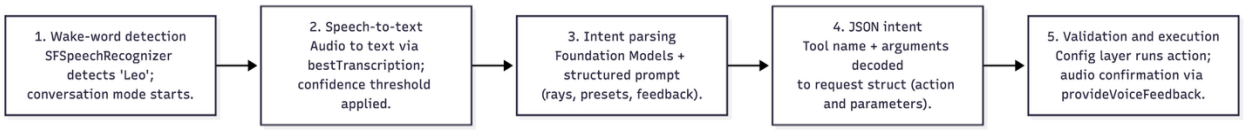

Figure 7: Voice Command Flow Chart

A critical technical challenge in implementing voice commands alongside spatial audio feedback is iOS audio session management. The operating system strictly controls audio sessions to prevent interference between applications and system functions. Applications must declare their audio session category, which determines whether they can play audio, record audio, or both simultaneously. The default AVAudioSession.Category.playback enables audio output but prevents microphone access, while the ".record" category enables input but mutes output, creating a fundamental conflict for simultaneous operation.

To enable simultaneous spatial audio playback and speech recognition, SensoryScape employs careful audio session orchestration. The application configures its audio session with the ".playAndRecord" category and sets the ".defaultToSpeaker "option to route output to headphones while capturing input from the microphone. The ".allowBluetooth" and ".allowBluetoothA2DP" options permit use of AirPods Pro, which provide both audio playback and microphone capabilities. The audio session uses the default mode, which provides standard audio behavior suitable for general playback and recording without specialized optimizations for voice chat or video recording.

Priority routing ensures that short command phrases are captured without muting or significantly attenuating obstacle cues. The Speech framework real-time transcription allows the system to detect command completion quickly through silence detection after 1.5 seconds, minimizing the duration during which audio feedback might be affected. Output mixing prevents system interruptions or dropped spatial cues by coordinating audio playback with voice recognition states.

Commands are validated using confidence thresholds to prevent accidental activation from ambient speech or misrecognized phrases. The Speech framework provides transcription confidence scores from 0.0 to 1.0, and SensoryScape requires a minimum confidence of 0.7 to process commands. This threshold was selected to balance between accepting legitimate commands and rejecting spurious activations from background noise or similar-sounding phrases. Context is preserved across conversation turns through a simple state machine that maintains the topic of the most recent command for 30 seconds, enabling follow-up requests such as "increase the volume" after an initial command like "adjust audio settings" without requiring users to re-specify which parameter they wish to modify. The validation logic also prevents invalid configurations that could compromise system stability or user safety, such as disabling all raycasts simultaneously (which would leave users without any feedback) or setting audio volume to inaudible levels below 10% of maximum.

Configuration Framework

SensoryScape implements a flexible configuration framework that enables users to switch between environment-specific presets while supporting fine-grained customization of sensing and feedback parameters. The framework allows configurations to adapt to both user preferences and varying navigation contexts without requiring modification of the underlying application logic. All configurations are serialized using JSON through Swift's *Codable* framework, ensuring consistent storage, portability, and extensibility across application sessions.

The central configuration object, *RaycastConfiguration*, encapsulates ray definitions, global sensing parameters, and feedback settings, including maximum sensing distance, global audio volume, and haptic feedback assignments. Environment presets are instantiated at runtime through the createDefaultPresets() routine and stored as individual JSON files within the application document directory using the naming convention ray_config_<name>.json. Both predefined and user-generated presets follow the same storage structure, enabling uniform loading, modification, and persistence behavior. The system additionally records the name of the most recently used preset in a separate file, allowing automatic restoration of user preferences when the application is relaunched.

Rather than relying on a formally defined external schema, parameter validity is enforced programmatically within the configuration layer. Minimum ray distances are constrained to 0.1 m to prevent unstable sensing behavior, maximum sensing ranges are defined according to preset requirements, and audio volume values are restricted to the normalized interval [0.0, 1.0]. Users may create, modify, and store custom presets through either the SwiftUI configuration interface or voice-based commands, with each configuration persisted independently to maintain modularity and prevent unintended cross-parameter interactions.

Three predefined presets—Indoor, Outdoor, and Navigation—are provided to support common mobility scenarios characterized by differing spatial constraints and sensing priorities. The Indoor preset prioritizes short-range sensing suitable for confined environments by limiting the maximum raycast distance to 3 m and reducing global audio volume to 0.8 to minimize auditory overload. Five rays are enabled in this configuration, consisting of Forward, Left, Right, Down1, and a silent Floor Detector ray used for ground awareness. In contrast, the Outdoor preset extends environmental awareness for open spaces by increasing the sensing range to 5 m and setting global audio volume to 1.0. An additional Up ray is introduced to improve detection of overhead obstacles while maintaining lateral and downward sensing coverage. The Navigation preset emphasizes guided forward motion and terrain safety by prioritizing forward sensing and elevation change detection while maintaining reduced peripheral feedback to limit cognitive load during directed travel.

Floor detection behavior and update logic are not independently defined within individual presets but instead rely on shared processing implemented within RaycastConfiguration and ARSessionManager. This separation ensures consistent environmental interpretation across configurations while allowing presets to modify sensing emphasis through parameter tuning rather than duplicated logic. Collectively, this configuration framework enables SensoryScape to dynamically adapt sensing behavior and multimodal feedback across diverse environments while maintaining a consistent and extensible system architecture.

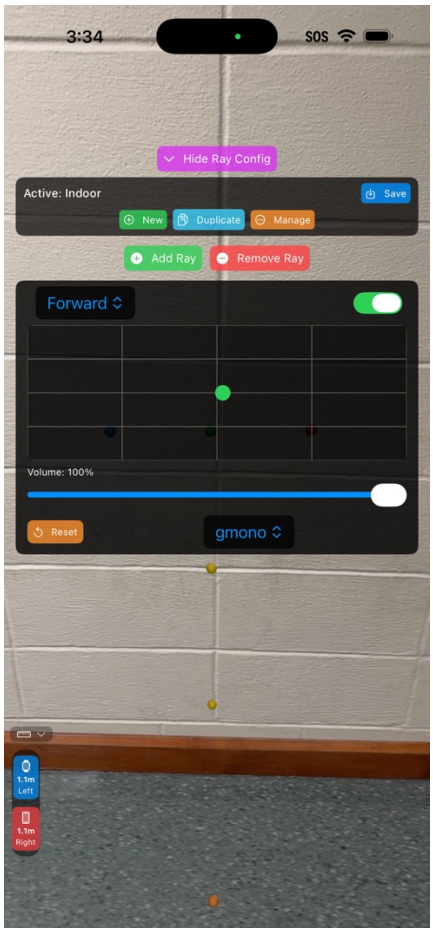

Figure 8: Screenshot of SensoryScape Ray Customization Panel

The configuration panel is displayed as a semi-transparent overlay on top of the live camera feed and active ray visualizations. At the top of the panel, the currently active preset is identified (“Indoor”) with a Save button to persist changes. Below the preset label, three management buttons allow users to create a New preset, Duplicate the current one, or Manage existing presets. The Add Ray and Remove Ray buttons provide the ability to modify the total number of raycasts in the active configuration. The central portion of the panel shows a per-ray editor: a dropdown selector identifies which ray is being configured (“Forward” in this capture), and a toggle switch enables or disables that ray. The black grid below represents the camera’s field of view, with the green dot indicating the selected ray’s current screen position; users can drag this dot to reposition where the ray originates relative to the device screen. A volume slider controls the audio output level for the selected ray, with the current value displayed as a percentage (100%). The Reset button restores the selected ray to its default position and settings.

A dropdown at the bottom of the editor (showing “gmono”) allows the user to select the audio tone type assigned to the ray. Behind the overlay, the ray visualization spheres and the distance readouts in the bottom-left corner remain visible, providing continuous spatial feedback while the user adjusts configuration settings.

Performance Optimization and System Stability

Achieving reliable 60 Hz operation required careful optimization across multiple subsystems. The ARKit configuration enables scene reconstruction with semantic classification to provide rich environmental understanding. Both horizontal and vertical plane detection are enabled to support enhanced spatial awareness and potential future features that could leverage explicit plane anchors, though the current obstacle detection system primarily relies on direct mesh raycasting for real-time responsiveness.

Error handling and graceful degradation were implemented to maintain operation during transient failures. When AR tracking quality degrades due to insufficient lighting or rapid movement, ARKit reports tracking state through its session delegate. SensoryScape monitors this state and responds by gradually adjusting system behavior as tracking quality decreases, maintaining some functionality rather than failing completely. When tracking is fully lost, audio and haptic feedback indicate system unavailability to prevent misleading feedback that could endanger users by suggesting clear paths when environmental reconstruction is unreliable.

Watchdog timers were implemented for inter-device communication to detect and recover from failures. If the iPhone does not receive an acknowledgment from the Watch within 2 seconds of sending a haptic command, it retries the transmission up to 3 times with exponential backoff (2s, 4s, 8s delays). After 3 failed attempts, the system logs the failure and continues operation without watch haptic feedback, relying solely on iPhone-based haptics. A background reconnection task attempts to re-establish the Watch connection every 30 seconds, automatically resuming dual-device operation when connectivity is restored. This recovery logic was essential for handling situations where users move their wrist out of Bluetooth range or when the Watch application is terminated by the operating system due to memory pressure.

Overall, the implementation represents a substantial engineering effort to overcome the architectural limitations of the Unity-based EchoSee system. The native Swift implementation achieves the performance targets required for real-time assistive navigation while maintaining code maintainability and extensibility for future development. The modular service-based architecture enables rapid iteration on individual subsystems without risking stability in other components, and the comprehensive error handling ensures that transient failures degrade gracefully rather than producing catastrophic malfunctions that could endanger users. The native iOS application architecture leverages ARKit and RealityKit for continuous spatial mapping and semantic surface classification, providing the perceptual foundation on which all feedback is built. The eight-ray raycasting engine projects configurable directional queries into the reconstructed mesh each frame, achieving sub-millisecond query times that remain well within the 16.7 ms budget required for 60 Hz operation. Spatial audio rendering through SceneKit places virtual sound sources at raycast intersection points, with HRTF-based processing and optional head-tracking through compatible headphones delivering directional cues that move with the perspective of the user. The distributed haptic feedback system pairs left and right raycasts with the Apple Watch and iPhone haptic actuators, respectively, encoding proximity through vibration intensity and repetition rate while throttling inter-device communication to maintain stability. The elevation change detection subsystem uses line-segment geometry constructed from pairs of downward-facing raycast hit points to infer the presence of stairs, curbs, and fall hazards through angular and slope analysis, with persistence filtering and rate-limited alerts reducing false positives. The voice command system integrates on-device speech recognition with foundation model intent parsing, requiring careful audio session orchestration to support simultaneous speech input and spatial audio output. Finally, the JSON-based configuration framework provides environment-specific presets and fine-grained parameter customization, enabling users to adapt sensing and feedback behavior to diverse navigation contexts without requiring modification of application logic.

CHAPTER 4: RESULTS

Technical Evaluation

Among the many environmental hazards encountered by blind and visually impaired individuals, downward falls represent one of the greatest risks of serious injury [17,18]. Stairways, curbs, and sudden elevation changes require reliable advance warning because, unlike frontal collisions with walls or furniture, a missed downward hazard can result in an immediate fall with little opportunity for recovery. For this reason, the technical evaluation of *SensoryScape* prioritizes the stair and elevation change detection subsystem, examining its performance across conditions representative of real-world navigation. While previous chapters describe the complete system architecture, this evaluation focuses on the component whose failure presents an obvious safety risk to users. Testing was conducted on an iPhone 17 Pro Max running iOS 26.2, representing available mobile hardware equipped with LiDAR sensing capabilities.

The evaluation methodology follows a structured testing protocol designed to measure hazard detection reliability under varied environmental conditions. To ensure consistency across trials, device scanning behavior was standardized using controlled lateral sweeps combined with slow, steady forward motion. This procedure minimized variability introduced by inconsistent user operation and reduced reconstruction errors within the LiDAR-generated spatial mesh.

Testing was performed across multiple environmental and structural variables, including indoor and outdoor staircases, diverse lighting conditions (well-lit indoor environments, dim lighting, direct sunlight, and nighttime conditions), varying step heights, differing stair gradients, and both ascending and descending hazard scenarios. These conditions were selected to approximate the range of environments likely to be encountered during independent mobility.

Each test condition was repeated five times to account for variability in LiDAR mesh reconstruction and real-time scene understanding. A trial was classified as successful when the hazard detection system generated a warning prior to the user reaching the hazard boundary. Trials were performed by users with normal vision. Trials in which the user reached the hazard without receiving a warning were recorded as failures. False positives were defined as warning

activations occurring when no hazard was present within the sensing field. For all successful detections, the distance between the user and the beginning of the hazard at the time of alert activation was recorded to evaluate available reaction time.

Overall Detection Performance

A total of 120 detection trials were conducted across eight hazard locations with step heights ranging from 4.5 to 7.5 inches, step lengths from 10.5 to 12.75 inches, and stair grades from 22° to 33°. Four indoor staircases were tested under controlled artificial lighting, and four outdoor locations (three staircases and one curb) were tested under both daytime and nighttime conditions. Each unique combination of location, direction, and lighting condition was tested with five repeated trials.

The system achieved an overall detection rate of 92.5% (111 of 120 trials). The mean alert distance for successful detections was 2.31 feet (SD = 0.91 ft), with a median of 2.0 feet and a range of 0.5 to 4.5 feet. No false positive activations were recorded. The nine missed detections were distributed across five of the eight test locations.

Table 1: Total Detection Data

Metric	Value
Total Trials	120
Successful Detections	111 (92.5%)
Missed Detections	9 (7.5%)
False Positives	0 (0.0%)
Mean Detection Distance	2.31 ft
Median Detection Distance	2.0 ft
Standard Deviation	0.91 ft
Range	0.5–4.5 ft

Daytime versus Nighttime Detection

The 80 outdoor trials were evenly divided between daytime and nighttime conditions. Daytime trials yielded a 95.0% detection rate (38 of 40) with a mean alert distance of 2.22 feet (SD = 0.71 ft). Nighttime trials produced a 90.0% detection rate (36 of 40) with a mean distance of 1.88 feet (SD = 0.63 ft).

Ascending detection showed a larger day–night difference than descending. Daytime ascending trials achieved 100% detection (20 of 20, mean 2.58 ft), while nighttime ascending dropped to 90.0% (18 of 20, mean 2.19 ft). Descending detection was 90.0% under both conditions (18 of 20 each), with mean distances of 1.83 feet (day) and 1.56 feet (night).

Indoor trials under controlled artificial lighting achieved 92.5% detection (37 of 40) with a mean distance of 2.82 feet (SD = 1.07 ft).

At the shallowest outdoor location (4.5-inch step height, 22° grade), descending detection was lower in daytime sunlight (3 of 5, 60%) than at night under artificial lighting (5 of 5, 100%).

Table 2: Outdoor Detection Performance by Direction

Condition	Detection Rate	Mean Distance	SD
Day – Overall	38/40 (95.0%)	2.22 ft	0.71 ft
Day – Descending	18/20 (90.0%)	1.83 ft	0.64 ft
Day – Ascending	20/20 (100.0%)	2.58 ft	0.59 ft
Night – Overall	36/40 (90.0%)	1.88 ft	0.63 ft
Night – Descending	18/20 (90.0%)	1.56 ft	0.51 ft

Night – Ascending	18/20 (90.0%)	2.19 ft	0.57 ft

Detection Performance Across Step Heights

Detection rate and mean alert distance both generally increased with step height. At 4.5 inches, the system detected hazards in 85.0% of trials (17 of 20, mean 1.68 ft, SD = 0.88 ft). At 5.5 inches, detection rose to 92.5% (37 of 40, mean 2.00 ft, SD = 0.55 ft). Steps of 6.0 inches and 7.5 inches both achieved 100% detection, with mean distances of 2.52 feet (SD = 0.66 ft) and 2.98 feet (SD = 0.79 ft) respectively. The 7.0-inch steps achieved 85.0% (17 of 20, mean 2.85 ft, SD = 1.32 ft), with both failures occurring at a single indoor location.

Table 3: Performance Across Step Heights

Step Height	Trials	Detection Rate	Mean Distance	SD
4.5"	20	85.0% (17/20)	1.68 ft	0.88 ft
5.5"	40	92.5% (37/40)	2.00 ft	0.55 ft
6.0"	30	100.0% (30/30)	2.52 ft	0.66 ft
7.0"	20	85.0% (17/20)	2.85 ft	1.32 ft
7.5"	10	100.0% (10/10)	2.98 ft	0.79 ft

In outdoor trials, the interaction between step height and time of day varied by height. At 4.5 inches, daytime detection was 80% (8 of 10) and nighttime was 90% (9 of 10). At 5.5 inches, daytime achieved 100% (20 of 20) while nighttime dropped to 85% (17 of 20). At 6.0 inches, both conditions achieved 100%.

Table 4: Outdoor Detection by Step Height

Step Height	Day Rate	Day Mean Dist.	Night Rate	Night Mean Dist.
4.5"	8/10 (80%)	1.75 ft	9/10 (90%)	1.61 ft
5.5"	20/20 (100%)	2.23 ft	17/20 (85%)	1.74 ft
6.0"	10/10 (100%)	2.60 ft	10/10 (100%)	2.35 ft

Detection Performance Across Step Lengths

Step lengths ranged from 10.5 to 12.75 inches across the seven staircase locations (excluding the curb). The 10.5-inch tread achieved 100% detection (10 of 10, mean 2.60 ft, SD = 0.91 ft). The 11.0-inch tread produced 80.0% (8 of 10, mean 2.75 ft, SD = 1.13 ft). The 11.5-inch tread achieved 85.0% (17 of 20, mean 1.68 ft, SD = 0.88 ft). The 12.0-inch tread achieved 92.5% (37 of 40, mean 2.51 ft, SD = 1.05 ft), and the 12.75-inch tread achieved 100% (20 of 20, mean 2.48 ft, SD = 0.53 ft). No consistent monotonic trend was observed between step length and detection rate.

Table 5: Detection by Step Length

Step Length	Trials	Detection Rate	Mean Distance	SD
10.5"	10	100.0% (10/10)	2.60 ft	0.91 ft
11.0"	10	80.0% (8/10)	2.75 ft	1.13 ft
11.5"	20	85.0% (17/20)	1.68 ft	0.88 ft
12.0"	40	92.5% (37/40)	2.51 ft	1.05 ft

12.75"	20	100.0% (20/20)	2.48 ft	0.53 ft

Detection Performance Across Stair Grades

Stair grades ranged from 22° to 33° across seven locations (excluding the curb). At 22°, detection was 85.0% (17 of 20, mean 1.68 ft, SD = 0.88 ft). At 26°, detection was 95.0% (38 of 40, mean 2.26 ft, SD = 0.62 ft). At 31°, detection was 85.0% (17 of 20, mean 2.85 ft, SD = 1.32 ft), with all three failures at two specific indoor locations. At 33°, detection was 100% (20 of 20, mean 2.79 ft, SD = 0.85 ft).

Effect of Approach Direction

Ascending approaches achieved a 95.0% detection rate (57 of 60) with a mean distance of 2.74 feet (SD = 0.82 ft). Descending approaches achieved 90.0% (54 of 60) with a mean of 1.86 feet (SD = 0.77 ft). Six of the nine total failures (67%) occurred during descending approaches.

Table 6: Detection Across Lighting Conditions

Condition	Desc. Rate	Desc. Mean	Asc. Rate	Asc. Mean
All Trials	54/60 (90.0%)	1.86 ft	57/60 (95.0%)	2.74 ft
Indoor	18/20 (90.0%)	2.18 ft	19/20 (95.0%)	3.42 ft
Outdoor Day	18/20 (90.0%)	1.83 ft	20/20 (100.0%)	2.58 ft
Outdoor Night	18/20 (90.0%)	1.56 ft	18/20 (90.0%)	2.19 ft

Missed Detections

Of the nine missed detections, three occurred indoors under controlled lighting, two during outdoor daytime conditions, and four during outdoor nighttime conditions. By step height, three failures occurred at 4.5 inches, four at 5.5 inches, and two at 7.0 inches. No failures occurred at 6.0 or 7.5 inches. By direction, six occurred while descending and three while ascending.

The three failures at 4.5 inches all occurred at the same outdoor location (22° grade), with two during daytime descent and one during nighttime ascent. The four failures at 5.5 inches were distributed across two outdoor locations: one nighttime curb descent and two nighttime trials (one descending, one ascending) at a low-light staircase with a 26° grade. The two failures at 7.0 inches occurred at a single indoor location (31° grade), one descending and one ascending.

CHAPTER 5: DISCUSSION

Detection Reliability and Safety Implications

The system achieved an overall stair detection rate of 92.5% with no observed false positives. In typical indoor and campus environments, this level of performance would provide advance warning for most encountered staircases, curbs, and sudden elevation changes. The mean alert distance of 2.31 feet provides sufficient margin for users walking at normal pace to initiate a stopping response.

The absence of false positives is particularly important for long-term usability. Assistive warning systems that generate frequent incorrect alerts often lead to alarm fatigue, causing users to ignore warnings altogether—including legitimate hazard notifications. Maintaining conservative alert behavior therefore directly supports user trust and sustained adoption.

Despite strong overall performance, the 7.5% missed detection rate remains a significant limitation for safety-critical deployment. Unlike collisions with walls or obstacles, missed downward hazards offer little opportunity for recovery and likely carry a higher risk of injury [17,18]. Even infrequent failures become consequential when individual failures may result in falls. The concentration of missed detections during descending approaches further emphasizes this concern, as descending falls typically produce more severe outcomes. These results reinforce the intended role of SensoryScape as a complementary safety layer rather than a replacement for established mobility aids such as the white cane.

Effects of Illumination on LiDAR Performance

Lighting conditions influenced detection distance even when overall detection rates remained similar. Reduced ambient illumination decreased the mean detection distance by approximately 15%, compressing the warning window from 2.22 feet to 1.88 feet. At typical walking speeds, this reduction corresponds to roughly 75 milliseconds less reaction time, which may meaningfully affect users with slower motor responses.

Interestingly, performance did not uniformly favor daytime conditions. At the shallowest outdoor step, detection accuracy decreased under direct sunlight compared to nighttime artificial

lighting. This behavior aligns with the operating characteristics of the iPhone direct time-of-flight LiDAR sensor, which relies on infrared emission and return timing. Sunlight introduces additional infrared energy that degrades signal quality, particularly when elevation changes are small and geometric cues are already weak. Under nighttime conditions, the emitter of the device dominates the sensing environment, producing cleaner depth measurements despite reduced visible light.

In contrast, intermediate step heights showed improved daytime performance. Here, geometric features were sufficiently pronounced to tolerate solar interference, while nighttime conditions likely degraded the ARKit visual-inertial odometry due to reduced visual texture. These opposing trends indicate that detection reliability emerges from an interaction between scene geometry and sensor illumination rather than any single optimal lighting condition. Future systems may benefit from dynamically adapting detection thresholds using ambient light measurements. Increased resolution, distance, and sensitivity in LiDAR systems on future hardware may also improve the performance.

Geometric Constraints on Detection

Detection performance increased with step height, confirming that larger vertical discontinuities generate stronger depth gradients within the reconstructed mesh. The results suggest a practical reliability threshold near six inches, above which detection remained consistent across tested lighting conditions. This finding is encouraging for deployment, as most building codes specify stair riser heights within or above this range. However, smaller non-standard elevation changes—such as shallow outdoor steps or curbs—remain more vulnerable to missed detection.

Step length showed little influence on performance, indicating that the algorithm primarily responds to vertical elevation change rather than tread depth. This outcome is consistent with the raycast-based detection strategy, which evaluates floor elevation ahead of the user rather than analyzing full stair geometry.

A more consequential finding was the directional asymmetry between ascending and descending detection. Ascending approaches produced higher detection rates and longer warning

distances, while descending hazards were detected later and less reliably. This asymmetry arises from a fundamental geometric limitation: ascending stairs present positive depth discontinuities, whereas descending hazards initially appear as empty space until the lower surface becomes observable. Because the current approach depends on detecting reconstructed geometry, warning generation is inherently delayed during descent. Addressing this limitation may require forward-angled raycasts, boundary-based mesh analysis, or temporal reasoning about missing floor geometry rather than relying solely on positive depth signals.

Relationship to Prior Systems

The observed performance directly addresses limitations identified in earlier systems. The LEOBelt waist-mounted depth camera struggled with floor-level elevation changes due to viewing geometry and limited onboard computation, while EchoSee relied on general proximity feedback without explicit hazard classification. SensoryScape extends these approaches through a native Swift implementation that applies targeted depth-gradient analysis directly to LiDAR mesh data, enabling stair-specific detection within the existing real-time spatial mapping pipeline.

Compared with deep learning–based stair detection methods reported in prior literature, the achieved detection rate is competitive given substantially tighter computational constraints. Many vision-based approaches exceed 95% accuracy under controlled conditions but rely on GPU inference and operate as standalone perception modules. SensoryScape instead prioritizes computational efficiency and system integration, allowing stair detection to function alongside spatial audio and haptic feedback within a unified multimodal assistive framework.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

Future Work

SensoryScape establishes a functional foundation for multimodal assistive navigation on consumer mobile hardware, but several directions remain open for future development that could substantially extend the capabilities of the system and practical utility.

A primary area for continued development is the deeper integration of AI systems into the navigation pipeline. The current voice command subsystem uses the Apple FoundationModels framework for intent parsing, but this capability represents only an initial step toward more sophisticated on-device intelligence. Future iterations could leverage foundation models to provide real-time scene description, enabling the system to convey not only the presence and distance of obstacles but also their semantic identity and contextual relevance. For example, rather than signaling a generic obstruction at two meters, the system could communicate that a bench is ahead to the left and a pedestrian is approaching from the right. On-device language models could also support more conversational interaction, allowing users to ask open-ended questions about their surroundings such as “is there a door on this wall?” or “what is the object to my right?” and receive natural language responses synthesized from the spatial mapping data and semantic classification already available within the pipeline. Expanding the AI integration in this direction would move the system beyond reactive obstacle detection toward a richer, more informative model of the user’s environment.

A second significant direction involves incorporating pathfinding capabilities or explicit navigational instructions for the user. The current system provides continuous spatial awareness and hazard detection but does not offer guidance toward a specific destination or suggest optimal routes through complex environments. Future work could integrate indoor pathfinding algorithms that operate on the reconstructed mesh to identify traversable paths, detect doorways and corridors, and guide users through unfamiliar buildings using directional audio or haptic cues. For outdoor navigation, integration with mapping services and GPS data could enable turn-by-turn routing augmented by the real-time obstacle awareness of SensoryScape, bridging the gap between macro-level wayfinding and micro-level hazard avoidance. Even without full pathfinding, the system could benefit from providing more explicit verbal instructions when

hazards or obstacles are detected, such as indicating the direction and approximate distance a user should adjust their path to safely navigate around an obstruction rather than simply alerting them to its presence. This combination of proactive route guidance with the existing reactive hazard detection would address one of the persistent gaps identified in the related work, where most systems support either high-level navigation or immediate obstacle avoidance but rarely both within a single integrated framework.

A third area for future work is the continued expansion of the system's customization capabilities to allow users to further tailor the application to their individual needs. While the current configuration framework supports environment-specific presets, adjustable raycast patterns, and remappable haptic assignments, there is considerable room to deepen this flexibility. Users could be given the ability to define entirely custom audio mappings, selecting not only tone frequency and volume for each ray but also choosing between different sound types such as tonal, percussive, or speech-based cues depending on personal preference and perceptual strengths. Haptic pattern customization could be extended to allow users to design their own vibration sequences for different alert types, enabling individuals with varying tactile sensitivity to optimize feedback salience. The preset system could be expanded to support location-aware automatic switching, where the application detects transitions between indoor and outdoor environments or recognizes previously visited locations and applies the appropriate configuration without requiring manual intervention. Accessibility-focused customization options such as adjustable alert thresholds, configurable cooldown periods between repeated warnings, and user-defined priority rankings for different hazard types would further allow individuals to calibrate the behavior of the system to their specific mobility patterns, confidence levels, and environmental contexts. Ultimately, the goal of these expanded customization options is to ensure that SensoryScape can adapt to the user rather than requiring the user to adapt to the system, recognizing that the diversity of needs and preferences within the blind and visually impaired community demands a correspondingly flexible assistive tool.

Conclusion

This thesis presented SensoryScape, a real-time multimodal assistive navigation system that integrates spatial audio, wearable haptic feedback, and on-device intelligence on consumer

mobile hardware. The system was developed to address limitations identified in two predecessor systems, EchoSee and LEOBelt, which individually demonstrated the value of auditory and haptic feedback for assistive navigation but each addressed only part of the challenge. By unifying real-time environment reconstruction, directional audio rendering, Apple Watch haptic output, voice-controlled interaction, and a flexible configuration framework within a single native iOS application, SensoryScape provides a more comprehensive approach to multimodal navigation assistance.

The migration from Unity to native Swift eliminated substantial runtime overhead and reduced frame-to-feedback latency, enabling consistent 60 Hz operation while performing continuous spatial mesh updates, multi-directional raycasting, semantic surface classification, and concurrent audio and haptic rendering. The distributed architecture coordinating iPhone-based spatial audio with Apple Watch haptic output required solving challenges in real-time data serialization, inter-device communication throttling, and iOS audio session management, ultimately delivering synchronized multimodal feedback without perceptible lag or dropped cues. The voice command system demonstrated that hands-free configuration changes are achievable alongside continuous spatial audio playback, and the JSON-based configuration framework provides users with meaningful control over raycast patterns, audio profiles, and haptic assignments to accommodate diverse navigation contexts and personal preferences.

The technical evaluation focused on the stair and elevation change detection subsystem, which addresses a recognized risk of injury for users of assistive navigation systems. Across 120 trials spanning varied step heights, stair grades, lighting conditions, and approach directions, the system achieved a 92.5% detection rate with no false positives and a mean alert distance of 2.31 feet. These results demonstrate that reliable hazard detection can be achieved within the computational constraints of real-time, on-device processing on consumer hardware. The evaluation also identified meaningful areas for improvement, particularly in descending hazard detection and performance at shallow step heights, which inform the directions outlined for future work.

SensoryScape is designed as a complementary safety layer that augments rather than replaces established mobility aids such as the white cane or guide dog. The reliance on widely

available consumer hardware for the system, specifically LiDAR-equipped iPhones and Apple Watches, avoids the cost and weight barriers, while also lowering the social acceptability concerns, that have limited adoption of custom assistive devices. By combining continuous spatial awareness through directional audio with immediate hazard warnings through haptic feedback, SensoryScape addresses the dual requirements of proactive environmental exploration and reactive threat detection that real-world navigation demands. The modular architecture and extensible configuration framework provide a platform for continued research and development, supporting the future integration of AI-driven scene understanding, pathfinding capabilities, and expanded customization options that can further improve the utility of the system for the diverse needs of blind and visually impaired individuals.

REFERENCES

[1] F. E. Brown *et al.*, “A novel, wearable, electronic visual aid to assist those with reduced peripheral vision,” *PLoS ONE*, vol. 14, no. 10, Art. no. e0223755, Oct. 2019.

[2] B. S. Schwartz, S. King, and T. Bell, “EchoSee: An assistive mobile application for real-time 3D environment reconstruction and sonification supporting enhanced navigation for people with vision impairments,” *Bioengineering*, vol. 11, no. 8, Art. no. 831, Aug. 2024.

[3] S. Maidenbaum *et al.*, “A wearable platform for closed-loop GNSS navigation for the blind,” *Nature Communications*, vol. 16, Art. no. 758, Jan. 2025.

[4] S. Shoval, J. Borenstein, and Y. Koren, “The NavBelt—A computerized travel aid for the blind based on mobile robotics technology,” *IEEE Trans. Biomed. Eng.*, vol. 45, no. 11, pp. 1376–1386, Nov. 1998.

[5] S. Maidenbaum, S. Abboud, and A. Amedi, “Sensory substitution: Closing the gap between basic research and widespread practical visual rehabilitation,” *Neurosci. Biobehav. Rev.*, vol. 41, pp. 3–15, 2014.

[6] D. Massiceti, S. L. Hicks, and J. J. van Rheede, “Stereosonic vision: Exploring visual-to-auditory sensory substitution mappings in an immersive virtual reality navigation paradigm,” *PLoS ONE*, vol. 13, no. 7, Art. no. e0199389, Jul. 2018.

[7] L. Thaler *et al.*, “Human click-based echolocation of distance: Superfine acuity and dynamic clicking behaviour,” *J. Assoc. Res. Otolaryngol.*, vol. 20, no. 5, pp. 499–510, Oct. 2019.

[8] A. Kolarik, S. Cirstea, S. Pardhan, and B. C. Moore, “A summary of research investigating echolocation abilities of blind and sighted humans,” *Hearing Research*, vol. 310, pp. 60–68, 2014.

[9] P. B. Meijer, “An experimental system for auditory image representations,” *IEEE Trans. Biomed. Eng.*, vol. 39, no. 2, pp. 112–121, Feb. 1992.

[10] Microsoft Corporation, “Seeing AI,” 2023.

[11] S. Pundlik, M. Tomasi, and G. Luo, “Evaluation of a portable collision warning device for patients with peripheral vision loss in an obstacle course,” *Invest. Ophthalmol. Vis. Sci.*, vol. 56, no. 4, pp. 2571–2579, Apr. 2015.

[12] M. J. Smith *et al.*, “Smart assistive technology for people with disabilities: A systematic review,” *JMIR Rehabil. Assist. Technol.*, vol. 11, Art. no. e55776, 2024.

[13] P. Strumillo *et al.*, “Different approaches to aiding blind persons in mobility and navigation in the ‘Naviton’ and ‘Sound of Vision’ projects,” in *Mobility of Visually Impaired People*, Springer, 2018, pp. 435–468.

[14] A. Kristjánsson *et al.*, “Designing sensory-substitution devices: Principles, pitfalls and potential,” *Restor. Neurol. Neurosci.*, vol. 34, no. 5, pp. 769–787, 2016.

[15] W. Elmannai and K. Elleithy, “Sensor-based assistive devices for visually-impaired people: Current status, challenges, and future directions,” *Sensors*, vol. 17, no. 3, Art. no. 565, Mar. 2017.

[16] Apple Inc., “ARKit,” 2026.

[17] R. Legood, P. Scuffham, and C. Cryer, “Are we blind to injuries in the visually impaired? A review of the literature,” Inj. Prev., vol. 8, no. 2, pp. 155–160, Jun. 2002.

[18] R. Q. Ivers, R. G. Cumming, P. Mitchell, and K. Attebo, “Visual impairment and falls in older adults: The Blue Mountains Eye Study,” J. Am. Geriatr. Soc., vol. 46, no. 1, pp. 58–64, Jan. 1998.