Name: Hayes (Hayden) Duong

Student ID: 222610226

RESEARCH / PROPOSAL

I. Introduction

A/ Background

Navigating independently in unfamiliar or complex environments remains a significant challenge for people with visual impairments, cognitive processing differences, or mobility limitations.

- While white canes and guide dogs provide essential support, these tools have limitations in detecting certain types of obstacles or providing contextual awareness.
- In recent years, assistive technologies have emerged to enhance user awareness through sensors, haptic feedback, GPS, and AI-based recognition tools.

At the same time, many commercial solutions are either cost-prohibitive, inaccessible, or ethically restricted due to their use of always-on camera systems.

- In university and public settings, privacy regulations often limit the rapid deployment of camera-based devices.
- Therefore, building modular, non-invasive assistive technologies using low-cost components has become a practical and scalable approach — especially in resource-limited regions like Southeast Asia.

B/ Project Objective

This project aims to develop a two-layer AI-assisted navigation system for people with vision or cognitive impairments, with a particular focus on practical deployment in university environments.

- Phase 1 focuses on designing a sensor-only prototype using the Arduino Uno.
    - The system leverages ultrasonic distance sensors and an IMU module to detect obstacles and orientation.

    - Feedback is provided through vibration motors and audio beeps to support real-time navigation without visual cues.
    - This approach avoids the use of cameras and infrared sensors to reduce cost and eliminate ethical concerns during early-stage field testing.

- Phase 2 expands the system into an AI-capable smart assistant by integrating a Raspberry Pi 5 with a camera and object/text recognition models.
    - This version aims to interpret environmental objects (e.g., chairs, doors, bookshelves) and signage (e.g., room numbers, exits), with speech-based output via text-to-speech.
    - Data fusion between the Arduino and Raspberry Pi systems allows the device to combine spatial awareness and semantic recognition.

C/ Scope and Impact

By implementing both a basic sensor-based navigation system and an AI-enhanced prototype, this project will:

- Demonstrate a low-cost, modular assistive system accessible to a wide range of users.
- Provide a testable, replicable framework suitable for further academic or community development.
- Bridge the gap between simple obstacle avoidance and intelligent environment understanding.

Ultimately, this capstone project contributes to the growing field of inclusive design and aims to empower individuals to navigate safely and independently in everyday spaces.

II. System Overview (Mechanism)

The proposed assistive navigation system consists of two integrated subsystems, each addressing different levels of environmental awareness:

A/ Sensor-Based Subsystem (Arduino Uno):

- Responsible for detecting obstacles and orientation using ultrasonic distance sensors and an IMU.

- This subsystem provides tactile (vibration) and audio (buzzer) feedback to alert users of nearby hazards in real time.
- It operates independently without a camera or internet connection, making it suitable for early testing and deployment in privacy-sensitive areas.

B/ AI Vision Subsystem (Raspberry Pi 5):

- Responsible for understanding the environment at a higher level. Using a camera, the system performs object detection (e.g., doors, chairs, tables) and text recognition (e.g., room numbers, signage).
- Based on the visual input, the Raspberry Pi delivers speech-based feedback through a built-in speaker.
- It also receives sensor data from the Arduino via serial communication, allowing the system to combine spatial and semantic information in real time.

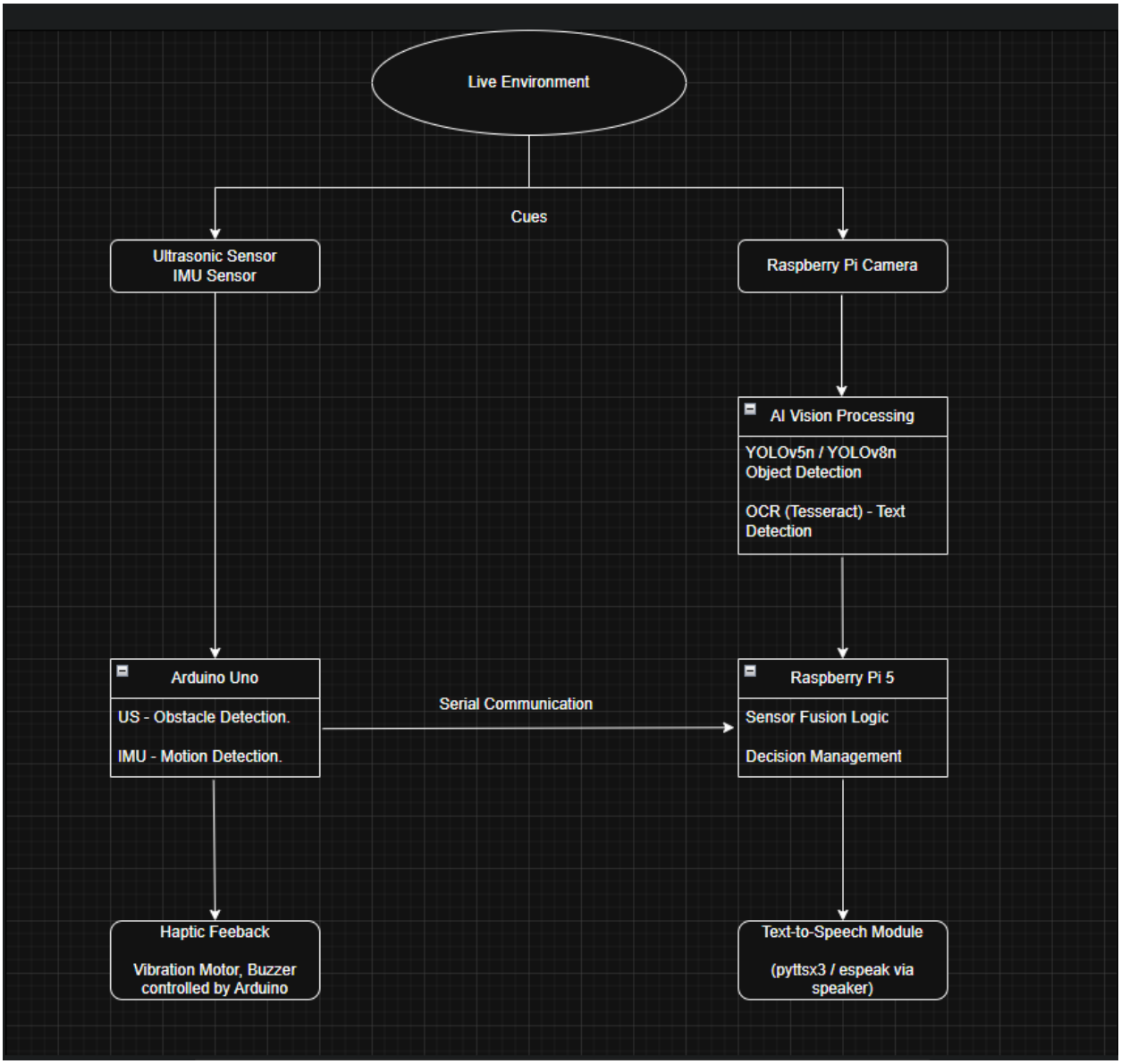

Diagram I – Suggestion Diagram for Prototype.

To ensure seamless integration between the physical sensing module and the AI-based vision module, the Arduino Uno communicates directly with the Raspberry Pi via a USB-based serial connection.

- This communication link allows the Raspberry Pi to receive real-time data from various sensors connected to the Arduino, including ultrasonic distance readings and inertial motion data from the MPU6050 IMU.

The primary purpose of this serial communication is to enable the Raspberry Pi to make more accurate and context-aware decisions.

- For example, if the ultrasonic sensor detects a nearby obstacle (e.g., within 30 cm), the Raspberry Pi can prioritize delivering a critical spoken warning via the TTS module, even if the vision module has not yet detected an object.
- Conversely, if both the camera and sensors provide confirming data (e.g., visual recognition of a “chair” and close-range obstacle detection), the system can combine these to enhance the richness and reliability of the user feedback.

Furthermore, the serial interface supports lightweight, structured data transfer — typically formatted in plain text or JSON — which makes it simple to parse and extend.

- An example data packet sent from Arduino to Raspberry Pi may look like the following structure

```json
{
  "distance_cm": 25,
  "imu_angle": -5,
  "obstacle": true
}
```

→ This bidirectional architecture also enables system redundancy.

- In the event of partial system failure (e.g., the AI model fails to detect an object or the camera is obscured), the system can still rely on sensor data to deliver immediate warnings through haptic (vibration motor) or auditory (buzzer) feedback triggered directly from the Arduino.
- This ensures a basic level of functionality always remains intact, which is crucial for real-world deployment and user safety.

In summary, the serial communication bridge not only connects the two subsystems but also forms the core of their cooperation — combining environmental sensing with intelligent interpretation.

III. System Architecture & Integration

A. Sensor-based System (Arduino-side)

The initial phase of the AI-assisted navigation device focuses on building a robust, low-cost, and non-invasive sensor-only prototype using the Arduino Uno.

This system leverages a combination of ultrasonic sensors and an Inertial Measurement Unit (IMU) to provide real-time spatial awareness, complemented by haptic and auditory feedback mechanisms.

This design intentionally avoids cameras or infrared sensors to mitigate ethical concerns and reduces initial deployment costs during early-stage field testing.

1/ Hardware & Components

Sensor / Module	Component Name
3x Ultrasonic	HC-SR04
Inertial Measurement Unit (IMU)	MPU6050
Vibration Motor	Eccentric Rotating Mass (ERM)
Buzzer	Passive Buzzer Module

Table 1 – Suggested hardware and components for phase 1.

Ultrasonic Sensors HC-SR04

Operating Voltage	5V DC
Operating Current	15mA
Operating Frequency	40KHz
Range	2cm – 400cm
Accuracy	3mm
Measuring Angle	< 15°

Table 2 – HC-SR04 specification [1].

- Three HC-SR04 ultrasonic sensors are strategically deployed to provide comprehensive obstacles and distance detection.
    - One sensor will be positioned at the front to detect obstacles directly ahead.
    - Two additional sensors will be placed on the left and right sides of the user, offering peripheral awareness and aiding in detecting obstacles from various angles.

IMU MPU6050:

Power Supply	2.375 – 3.46 V
Gyro FSR	±250/500/1000/2000
Gyro Sensitivity Error	±2%
Gyro Rate Noise	0.005dps/√Hz
Accel FSR	±2/4/8/16
Accel Sensitivity Error	±3%
Accel Noise	400μg/√Hz

Table 3 – MPU6050 specification [2].

- A single MPU6050 Inertial Measurement Unit is integrated to track the user's motion, tilt, and step detection through measuring the rotational velocity / rate of change of the angular position over time along the X, Y, and Z axis through a 3-Axis accelerometer and 3-Axis gyroscope

ERM Vibration Motor [3]:

- An Eccentric Rotating Mass (ERM) vibration motor is incorporated to provide tactile feedback to the user.
- This type of motor, often found in a compact module with an integrated driver circuit, is selected for its cost-effectiveness, ease of control with an Arduino, and widespread availability.
- This haptic feedback mechanism is lightweight and controllable via Pulse Width Modulation (PWM), allowing for variable intensity or patterns to convey different types of alerts or directional cues.

Passive Buzzer Module [4]

- A passive buzzer module serves as an audio feedback mechanism.

- It can generate beeps to indicate distances to obstacles or provide other stepwise information, acting as a simple yet effective alert system.

2/ Software Stack / Libraries

The Arduino-side software will primarily be developed using the Arduino IDE, leveraging a combination of standard Arduino functions and specific libraries for sensor interfacing.

- ***Arduino IDE & Core Libraries:***
  - The development environment for coding the microcontroller.
  - Standard Arduino libraries (e.g., digitalWrite(), analogWrite(), pinMode(), delay()) will be used for basic input/output operations, controlling the buzzer, and modulating the vibration motor intensity.

- ***NewPing Library:***
  - For the HC-SR04 ultrasonic sensors
  - This library provides efficient and accurate distance measurement functions, simplifying the process of obtaining reliable readings from multiple ultrasonic sensors by handling timing and pulse management effectively.

- ***Adafruit MPU6050 Library and Wire Library:***
  - To interface with the MPU6050 IMU.
  - This library, in conjunction with Arduino's built-in Wire library (for I2C communication), facilitates easy reading of accelerometer and gyroscope data, allowing for calculations of orientation and motion.

3/ Functionality

- ***Obstacle Detection (Front/Above/Sides):***
  - The three ultrasonic sensors continuously measure distances to objects in front, to the left, and to the right of the user.
  - The Arduino processes these readings to identify potential collisions or proximity warnings.

- ***Orientation and Motion Tracking:***
  - The IMU provides data on the device's tilt, motion, and potential step detection, which can be used to understand the user's current orientation and movement patterns.

- ***Feedback via Buzzer or Vibration:***
    - Based on the processed sensor data, the Arduino triggers the vibration motor and/or buzzer.
    - For instance, increasing vibration intensity or faster beep rates could indicate closer obstacles, providing intuitive, non-visual cues for safe navigation.
    - Activated through digital output (e.g., digitalWrite(pin, HIGH).

Situation	**Vibration Pattern**
Obstacle within 30cm	Long buzz (300ms)
IMU detects tilt/fall	Short repeated buzz (100ms x3)
No obstacle	No vibration

Table 4 – Vibration pattern suggestions.

4/ Limitations

While the sensor-based system provides a fundamental level of spatial awareness and obstacle avoidance, it inherently carries several limitations due to its design and the nature of the chosen sensors:

- **Lack of Semantic Understanding**:
    - The system can detect the presence and distance of an object but cannot identify what the object is (e.g., a chair, a door, a sign).
    - This limits the user's ability to understand their environment beyond simple physical barriers.

- **Line-of-Sight Dependency for Ultrasonic Sensors:**
    - Ultrasonic sensors require a clear line of sight to detect obstacles.
    - They may struggle with:
        - ***Soft, sound-absorbing surfaces***: Materials like thick curtains or certain fabrics can absorb sound waves, leading to inaccurate or missed detections.
        - ***Angled or irregularly shaped surfaces***: Sound waves might be reflected away from the sensor, causing it to miss an obstacle.

        - ***"Blind spots":*** The sensors cannot detect objects outside their conical beam width or around corners.

- **Limited Information Density from Feedback:**
    - The auditory (beeps) and haptic (vibration) feedback mechanisms, while effective for immediate warnings, cannot convey complex environmental information or nuanced details that an AI vision system with speech output would provide.

- **No Environmental Mapping or Persistence:**
    - The system reacts to real-time, immediate surroundings but does not build or maintain a persistent map of the environment.
    - Each detection is independent, without memory of previously encountered obstacles or landmarks.

- **Environmental Noise (for Ultrasonic Sensors):**
    - While generally robust, external factors such as strong wind or rapid temperature changes could theoretically introduce minor inaccuracies in ultrasonic readings, though typically not significant for short-range indoor use.

These limitations highlight the necessity for Phase 2 of the project, which aims to augment this basic sensor system with AI vision capabilities for a richer understanding of the environment.

B. AI Vision System (Raspberry Pi-side)

The second phase of the navigation device involves augmenting the sensor-based system with advanced AI vision capabilities.

This intelligent assistant will leverage the computational power of a Raspberry Pi to interpret the environment visually, identify objects, read text, and provide spoken feedback to the user, thereby bridging the gap between simple obstacle avoidance and nuanced environmental understanding.

a/ Hardware & Components

Hardware	Raspberry Pi 5

Camera Module	Camera Module 3 (from Raspberry Pi)
Audio Module	TBA

Table 5 – Suggested hardware and components for phase 2.

Raspberry Pi 5:

- Chosen as the central processing unit for its significantly enhanced processing power, improved graphics capabilities, and dedicated AI accelerators (compared to previous generations), making it suitable for on-device machine learning inference tasks such as object detection and text recognition.
- Its versatile GPIO pins and various connectivity options (USB, Ethernet, Wi-Fi, Bluetooth) facilitate integration with a camera module and audio output.

Raspberry Pi Camera Module 3 (or similar):

Resolution	11.9 megapixels
Sensor size	7.4mm sensor diagonal
Pixel size	1.4µm × 1.4µm
Horizontal / Vertical	4608 × 2592 pixels
Diagonal Field of View	75°
Common video modes	1080p50, 720p100, 480p120

Table 6 – Camera Module 3 specification [5]

- A high-resolution camera module designed specifically for the Raspberry Pi will serve as the primary visual input.
- This module provides raw image and video streams, which are essential for the AI models to process and analyze the surrounding environment for object and text recognition.

External Speaker or Audio Output Device (TBA):

- To provide spoken feedback via text-to-speech (TTS), an external speaker or a pair of headphones will be connected to the Raspberry Pi's audio output (e.g., 3.5mm jack or via USB audio adapter).
- This ensures clear and effective communication of environmental information to the user.

b/ Software Stack / Libraries

The AI Vision System on the Raspberry Pi will rely on a robust software stack to handle image processing, AI inference, and text-to-speech functionalities.

The primary programming language for this phase will be Python, given its extensive ecosystem for machine learning and embedded development.

Operating System:

- Raspberry Pi OS (64-bit): A Debian-based operating system optimized for the Raspberry Pi.
- This provides a stable and familiar Linux environment for installing necessary libraries, frameworks, and running Python scripts.

Core Libraries & Tools:

Function	Library / Framework	Purpose
Image Capture	picamera2 or cv2.VideoCapture	Captures real-time frames from the Raspberry Pi camera
Object Detection	YOLOv5 or YOLOv8 via Ultralytics, torch	Runs pretrained or fine-tuned object detection model
Text Detection (OCR)	Tesseract OCR with pytesseract	Extracts readable text from images of signs/labels
Image Preprocessing	OpenCV (cv2)	Converts, filters, and crops images for object/text detection
Array & Matrix Operations	numpy	Used for handling image data, ROI cropping, and numerical operations
Text-to-Speech (TTS)	pyttsx3 or espeak	Converts text to audio instructions for the user
Serial Communication	pyserial	Enables Arduino ↔ Pi communication via USB or UART
Model Training (optional)	torch, labelImg, Roboflow, etc.	If training custom YOLO model is needed later

Table 7 – Libraries suggestion for phase 2.

c/ Functionalities

This section breaks down the AI functionalities into modular components to demonstrate the role, logic, libraries, expected input/output, and real-world examples for each.

1. Object Detection

Tools/Libraries:

- YOLOv5n / YOLOv8n (Ultralytics).
- Torch (PyTorch) – model engine.
- OpenCV – frame handling.

Processing Steps:

- Capture a frame from the camera.
- Resize and normalize the frame to match model input requirements (e.g., 640×640).
- Run inference using YOLO.
- Parse output: bounding boxes, class labels, confidence scores.

Output Example (Json format):

```
{ "label": "chair", "confidence": 0.87 },
{ "label": "bookshelf", "confidence": 0.75 }
```

Feedback Logic:

- If object confidence > threshold (e.g., 0.7), announce:

  → “There is a chair ahead.”

- If multiple objects:

  → “Bookshelf on the right. Door straight ahead.”

- Object label mapping is customizable via class index.

2. Text Detection (OCR)

Tools/Libraries:

- Tesseract OCR (pytesseract)
- OpenCV (preprocessing)

Processing Steps:

1. Crop the area (bounding box labeled "sign").
2. Convert to grayscale, apply threshold or blur.
3. Use pytesseract.image_to_string() to extract text.
4. Sanitize text (remove non-alphanumeric characters, correct case).

Preprocessing Techniques:

- Gaussian Blur.
- Adaptive Thresholding.
- Dilation/erosion for better edge definition.

Output Example:

"Room 2.103"

Feedback Logic:

- If room number found → speak: “You are near Room 2.103”
- If text is unreadable → fallback: “Unable to read sign”

3. Text-to-Speech (TTS)

Tools/Libraries:

- pyttsx3 (offline, TTS engine)

- espeak-ng (backend engine for Linux)

Processing Steps:

1. Receive message string (e.g., “Bookshelf detected”)
2. Use engine.say(message) from pyttsx3
3. Call engine.runAndWait() to play audio

Output Examples:

- “Obstacle detected. Please stop.”
- “You are near a bookshelf.”
- “Room 3.106 detected.”

C. Possible setup / integration blueprint

a/ Communication Method

b/ Fusion Logic

c/ System behavior example

4. Sensor Fusion and Context Management

Purpose:

Merge Arduino sensor data (proximity, motion) with AI results to create layered context and prioritize feedback.

Tools/Libraries:

- pyserial for Arduino communication.
- json or struct for parsing incoming data.

Decision Flow:

1. Receive distance data from Arduino: e.g., distance = 22 cm
2. Check: is the obstacle too close? (e.g., <30 cm)
3. Receive AI detection: e.g., “chair” with confidence 0.87
4. If both are present:
    - Prioritize immediate danger → “Obstacle detected”
    - Else → “You are near a chair.”

D/ Suggested Setup & Integration Blueprint

Component	**Connection**	**Role**
Arduino Uno	USB to Raspberry Pi	Sends distance + IMU data
Ultrasonic Sensor	GPIO on Arduino	Detects nearby obstacles
IMU (MPU6050)	I2C on Arduino	Detects tilt, motion, orientation
Raspberry Pi 5	CSI Camera + USB + Speaker	Handles vision + TTS + decisions
Speaker / Headphones	Audio out (3.5mm/USB)	Delivers spoken feedback
Haptic Feedback	GPIO on Arduino	Obstacle alert, motion warning

Table 8 – Setup suggestions between Arduino Uno and Raspberry Pi.

- Possible software integration flow:
    1. Arduino detects obstacles via HC-SR04 or motion via IMU
    2. Sends data over serial (e.g., JSON or plain text) to Raspberry Pi
    3. Pi receives frame from camera, runs:
        a. Object detection
        b. OCR (if needed)
    4. AI module determines priority (e.g., “Obstacle detected!” vs “Bookshelf nearby”)
    5. Uses TTS to speak the final message & send haptic feedback depending on how close the user to nearby obstacles or destination is.

IV. Technologies considerations

A. Object detection

Criteria	Options Considered	Final Choice	Rationale
Framework	YOLOv5, YOLOv8, SSD-MobileNet	YOLOv5n / YOLOv8n	Lightweight, easy to implement on Raspberry Pi, excellent detection accuracy
Platform	TensorFlow, PyTorch	PyTorch	Required by YOLOv5/YOLOv8; more stable on Pi
Training	Custom dataset, COCO pretrained	COCO pretrained	Avoid time-consuming dataset preparation in early phase

B. Text Detection (OCR)

Criteria	Options Considered	Final Choice	Rationale
OCR Engine	Tesseract, EasyOCR	Tesseract OCR	Lightweight, offline, easy to integrate with OpenCV
Input preprocessing	Raw image, Grayscale + Threshold	Preprocessed with OpenCV	Improves accuracy significantly

C. Text-to-Speech (TTS)

Criteria	Options Considered	Final Choice	Rationale
TTS Engine	Google TTS, pyttsx3, espeak	pyttsx3 + espeak	Offline, responsive, customizable without internet

Criteria	Options Considered	Final Choice	Rationale
Voice control	Fixed or dynamic	Adjustable via pyttsx3	Voice speed, pitch, and gender can be controlled programmatically

D. AI Hardware Platform

Criteria	Options Considered	Final Choice	Rationale
Board	Jetson Nano, Raspberry Pi 5	Raspberry Pi 5 (8GB)	Budget-friendly, supports camera + GPIO, large community
Memory	4GB, 8GB	8GB	Required for running YOLO + OCR + TTS concurrently

E. Sensor + Microcontroller

Criteria	Options Considered	Final Choice	Rationale
Board	Arduino Uno, ESP32	Arduino Uno	Easy to use, well-supported, compatible with ultrasonic & IMU
Obstacle Detection	Infrared, Ultrasonic	Ultrasonic (HC-SR04)	Cost-effective, reliable in indoor environment
Orientation/Motion	IMU (MPU6050)	MPU6050	Provides basic tilt/motion sensing
Feedback	Buzzer, Vibration Motor	Both	Enables multi-modal feedback

V. Future Considerations

While the current prototype focuses on integrating fundamental sensing and AI-based vision features for indoor navigation assistance, there remain several key areas that offer potential for future development.

These enhancements may significantly improve usability, adaptability, and the real-world deployment potential of the system.

1. AI Model Optimization and Customization

The current implementation utilizes pre-trained YOLOv5/YOLOv8 models trained on general-purpose datasets (e.g., COCO).

- While effective, these models may lack specificity for certain indoor environments such as university campuses or public institutions.
- A logical progression would be to collect a domain-specific dataset—consisting of local room signage, hallway layouts, and furniture—and fine-tune the model to improve object detection accuracy and contextual relevance.
- The training process could be conducted using lightweight configurations suitable for edge devices or delegated to cloud infrastructure if needed.

2. Depth Awareness and Enhanced Imaging

To further enhance navigation support, particularly in unfamiliar or cluttered environments, the system could be extended to incorporate depth sensing.

- This may be achieved through stereo vision, LiDAR, or depth cameras such as Intel RealSense.
- With depth data, the system can provide more nuanced guidance, such as measuring relative distances between multiple obstacles and offering path suggestions.

Additionally, upgrading to a wide-angle or dual-lens camera may improve peripheral object detection, allowing for a more comprehensive understanding of the user's surroundings.

3. Indoor Positioning using Bluetooth Beacons

A limitation of the current system is the absence of precise indoor localization. While inertial sensors offer orientation cues, they lack positional accuracy.

- To address this, Bluetooth Low Energy (BLE) beacons could be deployed in key indoor locations.
- The Raspberry Pi would scan for these beacons and determine proximity based on signal strength (RSSI).
- This would allow the system to identify rooms or zones with higher certainty, enabling context-aware guidance such as: “You are near the library entrance.”

This feature was excluded from the initial prototype due to cost and infrastructure constraints but remains a valuable future enhancement.

4. Voice Interaction Capabilities

At present, user interaction is unidirectional, with feedback delivered via audio and haptic signals.

- Future iterations may include voice-based input through speech recognition libraries such as Vosk or Whisper Lite.
- This would allow users to issue verbal commands (e.g., “Read this sign” or “Describe the room”) and receive real-time responses.

Incorporating such interactivity would make the device more intuitive, particularly for users with diverse accessibility needs.

5. Expansion of Multimodal Feedback Systems

The existing feedback mechanisms include audio (text-to-speech) and haptic (vibration motor).

- Future versions could expand this by integrating additional feedback types, such as gesture-based input or capacitive touch sensors for silent operation.
- Feedback intensity or vibration patterns could also be personalized based on user preference or feedback collected during testing.

6. Cloud Services and Mobile Integration

With proper privacy and ethical handling, the system may be integrated with cloud platforms for scalable processing, multilingual support, and analytics.

- Cloud-based OCR or AI models (e.g., Google Vision, Microsoft Azure Cognitive Services) may offer improved accuracy. A mobile application companion could be

developed to provide configuration options, push notifications, or historical logs of routes taken by users.

7. Ethical and Regulatory Considerations

It is important to note that any future enhancement involving camera data, voice recordings, or cloud connectivity must comply with ethical research protocols and data privacy regulations.

- For broader deployment, especially involving visually impaired participants, formal ethics approval should be sought. Face anonymization, data minimization, and local storage should be enforced to protect user privacy.

References

[1] https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

[2] https://product.tdk.com/en/search/sensor/mortion-inertial/imu/info?part_no=MPU-6050

[3] https://deepbluembedded.com/arduino-vibration-motor-code-circuit/

[4] https://circuitmagic.com/arduino/passive-buzzer-module-with-arduino-step-by-step-guide/

[5] https://www.raspberrypi.com/products/camera-module-3/

[*] https://www.mouser.com/datasheet/2/737/mpu6050_6_dof_accelerometer_and_gyro-2489019.pdf

[*] https://thinkrobotics.com/blogs/learn/arduino-ultrasonic-sensor-tutorial-complete-hc-sr04-guide-for-distance-measurement-projects