Mid-Progress Report on Autonomous Navigation System for the Visually Impaired

TEAM HVBR
Members: Vijay Aravynthan (2023111007), Bhaskar Itikela (2023111032), Harshil Singh (2023111003), Monosij Roy (2023111016)

Introduction
Our team—Vijay Aravynthan, Bhaskar Itikela, Harshil Singh, and Monosij Roy—is developing an autonomous navigation system aimed at aiding visually impaired individuals. Guided by Professor Nagamani and TA Ansh, we utilize the Qualcomm Innovators Development Kit (QIDK) as the primary hardware platform. Our objective is to create a real-time navigation solution that combines computer vision capabilities with intuitive guidance mechanisms.

Problem Statement
The goal is to create an assistive system that can guide a visually impaired person safely through different environments using visual markers and cues. The system detects obstacles and indicates directions using color detection and object recognition.

Hardware

Qualcomm Innovators Development Kit (QIDK)
The QIDK is the core of our project, providing processing power and a built-in camera for real-time data capture. Its key features include:

- **Processing Power:** The onboard Neural Processing Unit (NPU) allows complex deep learning models to run at high speeds, crucial for real-time applications.
- **Camera Integration:** The integrated camera ensures immediate data capture, minimizing latency between image capture and processing.
- **Connectivity:** Multiple interfaces for peripherals allow the integration of additional sensors or devices if required.

Software

YOLOv8 (You Only Look Once) Segmentation
We use YOLOv8, a high-performance deep learning model, for object detection and segmentation due to its accuracy and speed. Key aspects include:

- **Real-Time Performance:** Optimized to run on the QIDK's NPU, ensuring minimal delay between data capture and user feedback.
- **Color Detection:** Custom algorithms integrated with YOLOv8 allow accurate detection of specific colors (e.g., red and blue), essential for navigation.
- **Obstacle Avoidance:** Bounding box outputs enable the system to determine obstacle positions relative to the user.

Deployment

Wheelchair Implementation

- **Dual-Camera Setup:** Includes a main forward-facing camera for obstacle detection and a bottom-facing auxiliary camera for detecting floor-based direction markers.
    - **Main Camera:** Identifies obstacles and visual cues (red and blue markers) on walls.
    - **Bottom Camera:** Detects floor markers, enhancing direction guidance at junctions.

Handheld Device Implementation

- **Single-Camera Configuration:** A portable solution using a single main camera.
- **Functionality:** Recognizes obstacles and pathways with minimal hardware, suitable for handheld devices.

Logic

The system leverages wall and floor markers to provide spatial cues. By analyzing the positions and colors of these markers, the system delivers navigation guidance.

- **Obstacle Detection:**
    - **YOLOv8 Model:** Detects and classifies objects in real-time.
    - **Region Analysis:** Divides the camera feed into left, center, and right regions to identify open paths.
- **Color Cues Interpretation:**
    - **Blue Markers on Both Sides:** Indicate an open path on the left and right sides.
    - **Red Marker on One Side:** Suggests a blocked route on that side.
- **Direction Detection:**
    - **Bottom Camera Cues:** Uses color cues to detect junctions and indicate the nearest exit.
    - Combines cues from both cameras to better interpret spatial positioning and navigate corners.

Software Implementation

Code Overview

The software, written in Python, uses OpenCV for image processing and the Ultralytics YOLOv8 model for object detection. Key components include:

- **Dual-Camera Processing:**
    - **Main Camera (process_main_camera_frame):** Performs obstacle detection, divides the frame into three regions for path assessment, and identifies red and blue wall markers.
    - **Bottom Camera (process_bottom_camera_frame):** Detects direction markers on the floor, analyzing color combinations to determine exit directions.

Feedback and Interaction

- **Audio Feedback:**
    - **Pre-Recorded Messages:** Plays audio cues corresponding to visual guidance.
    - **Onboard Speakers:** Outputs audio guidance via device speakers.
- **Visual Display:**
    - **Overlaid Messages:** Displays navigation instructions on-screen.
    - **Processed Frames:** Annotated camera feeds assist nearby individuals.

Limitations

- **Hardware Constraints:**
    - **Processing Power:** Despite the QIDK's NPU, complex models may experience latency.
- **Environmental Challenges:**
    - **Lighting Conditions:** Variable lighting affects color detection, though compensation methods are in place.
    - **Structural Variability:** Detection issues arise with doors and non-standard walls lacking predefined markers.
- **Marker Dependency:**
    - **Reliance on Markers:** The system requires red and blue markers for navigation guidance.

Steps to Complete Solution

- **Enhance Depth Perception:** Implement stereo vision or depth sensors to improve obstacle detection.
- **Improve Color Detection:** Refine algorithms for better handling of lighting variations and color differences.
- **Advance Door and Wall Detection:** Develop specialized models or use additional sensors to detect doors and walls without markers.
- **Optimize Performance:** Fine-tune the YOLOv8 model for better QIDK performance, potentially using model pruning or quantization.

Next Steps

- **Android Development:** Explore porting the application to Android to improve portability and accessibility.
- **Efficient Door Detection:** Investigate edge detection or other computer vision techniques for more reliable door detection.
- **User Testing and Feedback:** Conduct user trials with visually impaired individuals to gather feedback and identify improvement areas.

Conclusion

This mid-progress report outlines our efforts in developing a navigation aid for visually impaired individuals using the QIDK and YOLOv8. While initial results show promise in

providing real-time guidance, further refinements in depth perception, color detection, and environmental adaptability are necessary. With continued development, this solution holds potential for safe, reliable navigation assistance, enhancing the independence and quality of life for visually impaired users.

DETAILED EXAMPLE WITH DIFFERENT SCENARIOS

Scenario: Navigating Through a Junction

Imagine a user in a wheelchair is navigating through a corridor and reaches a junction where the path splits into multiple directions. There are **visual cues** (red and blue markers) on the **walls** and on the **ground** to indicate available paths and the exit direction.

Step-by-Step Example

1. Initial Path Detection

- **Environment Setup:**
  - The user is in a corridor.
  - The corridor is split into three parts: **left**, **center**, and **right**.
  - There are no visual markers on the wall initially, but there are obstacles.
- **Main Camera Analysis:**
  - The main camera divides the frame into **three vertical regions**: left, center, and right.
  - The model scans the frame:
    - **Left Region**: Empty.
    - **Center Region**: Contains a small object (e.g., a chair).
    - **Right Region**: Clear.
- **Output Decision:**
  - The code identifies that the **center region** is partially blocked.
  - Since the **right region** is clear, the model sets the message to **“Right path is clear.”**
  - No turn instructions are provided yet since there are no visual cues detected on the walls or ground.

2. Approaching the Junction with Wall Cues

- **Environment Update:**
  - As the user moves forward, the main camera detects **two markers on the wall**.
  - There are two markers in the center of the frame: one **red** and one **blue**.
  - These markers are positioned side by side and indicate that the user is approaching a potential turn.
- **Main Camera Analysis:**
  - The camera detects two visual cues (one red and one blue) positioned adjacently.
  - The code sets the **visual clue message** to **“You can turn right ahead.”**

- **Output Decision:**
    - The model sets the message to "**Approaching junction**."
    - No audio or turn instruction is given yet because the **bottom camera** hasn't confirmed ground markers, indicating that it's not yet time to turn.

3. Bottom Camera Detects Ground Markers for Exit Direction

- **Environment Update:**
    - As the user moves closer to the junction, the bottom camera detects **two ground markers**:
        - **Blue on the left** and **red on the right**.
    - These markers indicate an **exit to the east**.
- **Bottom Camera Analysis:**
    - The bottom camera scans the ground frame and identifies the color markers.
    - It confirms the presence of blue and red markers positioned side by side, indicating **"Exit to East."**
- **Combined Output Decision:**
    - Since **both the main and bottom cameras** have detected matching markers (visual clues on the wall and ground), the program combines these messages:
        - "**You can turn right ahead and exit to East**."
    - This indicates that the user should turn right at the junction to exit eastward.

4. User Requests an Audio Cue

- **User Input:**
    - The user types **'1'** in the terminal to request an audio cue.
    - The program captures the current messages from both cameras: **"You can turn right ahead and exit to East."**
- **Audio Playback:**
    - The code maps the message to the corresponding audio file in the `audio_files` folder.
    - It plays the pre-recorded audio file for **"You can turn right ahead and exit to East."**
- **User Experience:**
    - The user hears the audio instruction and turns right to follow the exit path.

5. Reaching the Exit

- **Environment Update:**
    - After turning right, the user sees two **blue markers** on the ground (detected by the bottom camera).
    - These indicate that the user has reached the exit.
- **Bottom Camera Analysis:**
    - The code sets the message to **"Exit to North"** based on the blue markers.
- **Output Decision:**
    - The code displays **"Exit to North"** on the main camera feed.
    - If the user requests an audio cue again, it plays the corresponding message, indicating that the user has reached the exit.

Explanation of How the Code Handles This Scenario

1. Region-Based Path Detection (Main Camera)

- The main camera divides the frame into **three regions** (left, center, right) to detect obstacles and provide path clearance feedback.
    - If the **center region** is blocked, the code advises to stop or choose another path.
    - If either the **left** or **right** region is clear, the code suggests moving in that direction.
    - The division into three regions helps the model understand which side of the corridor is clear and which is blocked, giving more precise navigation instructions.

2. Wall Cues Detection (Main Camera)

- The main camera detects **visual cues** like red and blue markers on the walls.
    - These cues suggest **approaching turns** and guide the user on which direction is available (e.g., “Turn right”).
    - However, turn instructions are not immediately given to prevent premature decisions.

3. Junction Detection Using Ground Markers (Bottom Camera)

- The bottom camera detects **ground markers** that indicate junctions and exit directions.
    - It works independently of the main camera and continuously checks for ground markers.
    - When two markers are detected side by side, it indicates a **specific direction** for the exit (e.g., “Exit to East”).
    - The ground markers confirm the presence of a junction and the available exit path.

4. Combining Information from Both Cameras

- The code combines the information from **both cameras** to ensure that turn and exit instructions are only given when:
    - **Visual clues from the main camera** match the **ground markers detected by the bottom camera**.
    - This reduces false positives and prevents premature instructions, making navigation decisions more reliable.

5. Audio Cues on User Request

- The user can request an audio cue at any time by pressing **‘1’**.
    - The code captures the current messages from both cameras and plays the corresponding pre-recorded audio files.
    - This provides additional guidance, especially useful for users relying on auditory feedback.

Why This Code Logic is Effective

1. **Ensures Accurate Path Guidance:**
    - By dividing the main camera feed into three regions, it accurately determines which side is clear.
    - This helps the user understand which direction to follow when moving forward.
2. **Provides Turn Instructions Only at Junctions:**
    - Turn instructions are only provided when **both cameras** detect matching markers, ensuring timely guidance.
3. **Handles Junction Detection with Independent Cameras:**
    - The **bottom camera** handles ground markers independently, providing additional confirmation of the junction and exit direction.
4. **Offers On-Demand Audio Feedback:**
    - Users can request audio feedback at any point, making the navigation system more user-friendly and accessible.
5. **Robust to Lighting and Color Variations:**
    - The color detection algorithm accounts for different shades of red and blue, making it adaptable to varying lighting conditions.

Conclusion

This example scenario demonstrates how the code integrates **region-based path detection**, **visual cues from walls and ground**, and **audio feedback** to provide reliable, real-time guidance to the user. By considering both cameras' inputs, it ensures that instructions are given only at the right time and in the correct context, reducing the risk of errors and enhancing overall navigation accuracy.