Adaptative Cane for Visually Impaired People

Nathan Lambrechts
Simon Dourte
Florian Stormacq
Edwyn Eben
Louca Mathieu
nathan.lambrechts@student.unamur.be
simon.dourte@student.unamur.be
florian.stormacq@student.unamur.be
edwyn.eben@student.unamur.be
louca.mathieu@student.unamur.be
University of Namur, Namur Digital Institute (NaDI) and Research Center on Information Systems Engineering (PReCISE)
Namur, Belgium

Figure 1: Overview of the Adaptive Cane System with Intel RealSense camera, microphone, Raspberry Pi processing unit, and haptic feedback motors

AIM 2025, Namur, Belgium

ACM ISBN 000-0-0000-0000-0/00/00
https://doi.org/00.0000/0000000.0000000

Abstract

People with visual impairments represent a significant proportion of the global population. It is estimated that at least 2.2 billion people have some form of vision impairment. It is therefore essential to develop solutions that meet their needs and improve their independence. With this in mind, this project presents an adaptive connected cane for blind or visually impaired people designed to

improve their perception of their surroundings and increase their safety in the face of potential obstacles and dangers. The cane is based on two main sensors: an Intel RealSense D435 depth camera and a USB microphone. The data from these sensors is processed in real time by a Raspberry Pi 3. The microphone assesses the noise level of the environment, while the camera detects the position and distance of obstacles and the recommended area to avoid them. This information is used to determine one of the device's three alert modes. User feedback is provided by three vibrating motors, each associated with a direction. The intensity and duration of the vibrations indicate the proximity of the danger and guide the user as they move. This prototype is therefore an assistance solution aimed at improving the mobility and safety of blind or visually impaired people in their daily lives.

CCS Concepts

• **Human-centered computing** → **Mixed / augmented reality**; Graphical user interfaces; User interface programming; *Interaction devices*; **Accessibility**.

Keywords

Assistive technology, Visual impairment, Haptic feedback, Depth sensing, Audio processing, Raspberry Pi, Arduino

ACM Reference Format:
Nathan Lambrechts, Simon Dourte, Florian Stormacq, Edwyn Eben, and Louca Mathieu. 2024. Adaptative Cane for Visually Impaired People. In *Proceedings of the UNamur Symposium on Advanced Interaction Methods (AIM 2025 Proceedings), December 1, 2025, Namur, Belgium.* ACM, New York, NY, USA, 5 pages. https://doi.org/00.0000/0000000.0000000

1 Introduction

The goal of the project is to develop an adaptive smart cane designed to improve spatial awareness and safety for visually impaired people. The system relies on two main sensors: a depth camera and a microphone to detect obstacles or danger in real lifetime. The data is collected by a Raspberry Pi which analyzes visual and audio information from the environment.

The system focuses entirely on haptic feedback. A set of 3 vibration motors is placed on the cane to provide tactile information to the user. Each motor has a direction which means one direction is missing. Indeed, we have left out the back direction because the camera will be settled as glasses so they can't see behind the user. The vibration intensity and duration reflect the proximity of the danger allowing the user to interpret the environment with the haptic output.

This cane aims to support blind people during a random urban walk or any action that has potential danger to the user.

2 Related work

Recent work has explored the use of haptic feedback to support navigation for visually impaired people. The system presented in [1] shows that vibration cues can effectively convey spatial information and help users perceive obstacles without relying on vision. However, this approach focuses mainly on haptic modality and does not integrate additional modalities. This limitation highlights the need for devices that combine multiple sources of information to provide a more reliable feedback during navigation.

3 Methodology

The project is structured into several key components to ensure optimal functionality, performance, and maintainability:

3.1 Project Architecture

3.1.1 *Sensors.*

Audio Sensor (Microphone): To capture ambient sound, we use a USB microphone ("USB PnP Sound Device"). The goal is to continuously record the surrounding audio signal in order to determine the noise level around the user. This allows us to know whether the environment is noisy (*e.g.*, a crowd) or quiet.

Video Sensor (Intel RealSense Camera): To capture the image and depth of field, we use a RealSense Depth D435i camera connected to the Raspberry Pi with a USB-C cable. The aim is to continuously record the video stream perceived by the camera, knowing where the danger is and how far away it is, in order to determine which alert mode the cane will be in and where to dodge it.

3.1.2 *Data Capture and Processing.*

To capture and process data from the sensors in real-time, a Raspberry Pi 3 is used.

Note: This device is not the initial Raspberry Pi 1 that were furnished at the beginning of the project, as it was not powerful enough to handle the processing requirements.

The Raspberry Pi runs a Python script, which is responsible for capturing and processing the data from the sensors. This process outputs a simple result: specified intensity levels for each vibration motor, as well as the noise level detected.

The output would be added in a queue, which would be read by an Arduino board to actuate the motors accordingly.

3.1.3 *Arduino Board.*

In this project, Arduino will be used as an interface between the central processing (Raspberry) and the physical actuators (vibration motors). It receives already-processed instructions that specify how the motors should behave.

First, the Raspberry sends data divided in 3 with where the danger comes from, the intensity and the duration of the vibration. The Arduino receives the information, interprets it and converts it into appropriate signals for the vibration motors.

In the overall architecture, the Arduino is responsible for executing the physical output (vibration motors) in real time. It doesn't treat data, it applies the received instructions for the vibration motors.

3.2 Project Implementation

A Python script is implemented to run on the Raspberry Pi 3, which captures and processes the data from the sensors in real-time.

Note: As the sensors "produce" data continuously, the script implements a multi-threaded Producer-Consumer architecture to handle the data efficiently.

A `main.py` file is the entry point of the codebase, which initializes and starts the different components (as the sensors) of the system.

3.2.1 *Sensors Implementation.*

Audio Sensor (Microphone): A Python script in the micro folder is used to capture the data recorded by the microphone. Audio capture relies on the sounddevice library, which creates an audio stream that captures samples in real time. The signal is retrieved as chunks, then sent into a queue to be processed.

To achieve this, the script first searches for the desired audio device. Then, it defines various parameters (chunk size, chunk duration, etc). After that, audio capture is performed through an audio stream (InputStream) associated with a callback function. Each time new data is received, the samples are added to an internal buffer. As soon as this buffer contains enough samples to form a complete chunk, the chunk is extracted and converted into a NumPy array before being sent into a queue for processing.

Video Sensor (Intel RealSense Camera): Video capture is based on the pyrealsense2 library, which is Intel RealSense's official library, as well as other imports such as numpy, cv2 and deque for image processing and data history management.

After initialising the camera and its settings, the script launches a pipeline in depth mode, retrieves the depth to convert the raw values into metres and prepares a history for each direction.

It then divides the image into three areas and calculates the median for each area, then saves them in the corresponding history.

The script then defines the danger level, but only takes into account the area in front (below one metre: alert, between 1 and 2 metres: caution, beyond: peaceful area).

In addition, the script detects where obstacles are located and indicates the recommended area to avoid danger.

Finally, the script displays the information.

All these operations are performed continuously in an infinite loop, which allows the stream to be processed in real time and the histories to be constantly updated in order to avoid sudden variations and obtain a more stable distance for each area.

Both sensor data are captured in separate threads, which act as separate "Producers" in the architecture. However, in the current ideal version of the code, the two producers would share the same queue and will add data into a shared tuple [(microphone data, camera data),...]. This ideal version is not yet implemented, but this would ensure that the data from both sensors are synchronized.

3.2.2 Raspberry Pi Script. The Raspberry Pi script ensures the following functionalities:

(1) The audio processing: Consumes the audio data from the microphone producer thread and computes the necessary audio features (*e.g.*, RMS, noise level, etc.).
(2) The video processing: Consumes the video data from the camera producer thread, computes the necessary features (not yet implemented in the current version).
(3) Add the computed features into a queue, which will be read by the Arduino board to actuate the motors accordingly.

The current version of the code only implements the audio processing part, while the video processing, the synchronization and the data communication with the Arduino board are yet to be implemented.

Limitations of the Raspberry Pi: Several problems were encountered when using the Raspberries.

(1) **Performance Issue:** The initial Raspberry Pi 1 was not powerful enough to handle the processing requirements of the audio and video data in real-time. Time was lost trying to optimize the code to run on this device. Eventually, it was decided to switch to a more powerful Raspberry Pi 3.
(2) **Configuration Issue:** Setting up the Raspberry Pi with the necessary libraries, configurations and dependencies for the sensors (particularly the Intel RealSense camera) proved to be more challenging than anticipated. Significant time was spent to configure the device properly. However, for reproducibility purposes, all the steps followed to set up the Raspberry Pi 3 are documented in a separate installation guide.
(3) A particular note can be made about the installation of the Intel RealSense Python library, which required to clone the library from GitHub and build it from source, as the precompiled binaries were not compatible with the Raspberry Pi architecture. This process involved lots of resources and time.
(4) It is assumed that the synchronization of the data from both sensors will be challenging.

3.2.3 Arduino Board Implementation. The Arduino implementation is structured in two main parts. The first part is the initialization phase where the necessary functions are prepared so they can be reused across different folders of the project. The second part consists of the main program logic where the code is executed.

The main loop continuously receives data from the Raspberry Pi and applies the instructions to the output pins. These instructions determine which vibration motor should be activated, at what intensity and for how long.

For now, the vibration motors have temporarily been replaced with LEDs for testing purposes due to a voltage issue between the Arduino and the vibration motors. This allows the output logic to be verified and fixes the potential problems.

On the Raspberry part, the team has switched to a producer-consumer architecture to handle receiving data from the camera and the microphone. The next step for the Arduino will be to stay in agreement with the Raspberry so it is essential to have a producer-consumer in the Arduino part too.

4 Contribution

TO BE DONE

5 Evaluation

The evaluation section will be completed once the full system is implemented and tested with users.

6 Discussion

6.1 What's Next

On the Raspberry part, we have to finish the receipt of danger from the two modalities and finish the producer-consumer architecture. Same for Arduino where the PC architecture has to be implemented.

To follow a happy end of the project, we will need to finalize all unwritten sections of this report. We have to add the evaluation

and conclusion part. These parts require the project to be over and so it will be done right after the project's end.

7 Conclusion

The conclusion will be written upon completion of the project implementation and evaluation.

Open Science

Our GitHub repository with code is accessible at https://github.com/fstormacq/Master1-IIA-project.

Acknowledgments

This work was supported by the University of Namur, the Namur Digital Institute (NaDI), and the Research Center on Information Systems Engineering (PReCISE). The authors would like to thank all individuals who contributed to this project. A particular thanks to the providers of the hardware components that made this research possible.

References

[1] Research Team. 2023. Haptic Feedback for Navigation Support of Visually Impaired People. *ACM Digital Library* 1, 1 (2023), 1–10. https://dl.acm.org/doi/abs/10.1145/3711931 Article on haptic feedback systems for navigation assistance.