ENHANCING USER SAFETY WITH REAL-TIME OBJECT DETECTION AND VIBRATION FEEDBACK: A MOBILE APPLICATION INTEGRATED WITH YOLOV5, FLUTTER, AND NRF52 HARDWARE

Zhefu Lei

Lexington Christian Academy, 48 Bartlett Ave, Lexington, MA 02420

ABSTRACT

Providing a means of navigation for hard of seeing individuals has always been a challenge. The most infamous is the white walking cane that people usually associate with the hard of seeing. However, while the cane has undoubtedly helped many navigate their surroundings, the limitations posed are overtly clear when it comes to navigating with a white cane. The solution posed in this paper, the EchoSense, is a vest composed of an onboard microprocessor, distance sensors, haptic motors, and a corresponding mobile application. The vest will alert the user of immediate obstructions to their path using the distance sensors and haptic motors, while the mobile app uses advanced real-time object detection to alert the user of specific objects in their path via vibrations from the mobile app as well as vibrations from the vest in the direction of the recognized object [1]. The onboard microprocessor is a bluefruit nRF52, and uses Adafruit distance sensors and haptic motor controllers. The mobile application was builtusing the Flutter framework, and is available on both iOS and Google Play stores [2]. During the development process, many challenges were encountered and overcame. The biggest challenge posed was the development of the mobile application, as compiling many different libraries in a very bleeding edge framework is bound to have some conflicting dependencies. To test to ensure the efficacy and efficiency of the EchoSense project, multiple rounds of experimenting were performed to find the response time of the real-time object detection and hardware [3]. All experimental results were satisfactory, and can be viewed in this paper. The EchoSense is a great choice for the hard of seeing to rely on for their daily navigation, as it is cost effective, efficient, and provides a great deal of information about the user's surroundings without overwhelming their senses.

KEYWORDS

Real-time object detection, Mobile application, Yolov5, Bluetooth connectivity, Vibration feedback

1. INTRODUCTION

There currently exist few products that can efficiently aid the visually impaired to navigate outdoor environments [4]. Additionally, the options that do exist and are employed are quite limited in their function. For example, the classic white cane is merely a walking stick and does not provide any other feedback other than what it can physically touch. On top of this the more advanced existing accessibility options mostly aid with indoor tasks such as reading price tags or

David C. Wyld et al. (Eds): SIGI, CSTY, AI, NMOCT, BIOS, AIMLNET, MaVaS, BINLP – 2025
pp. 299-311, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151924

selecting groceries, but few provide the navigation needed for traversing outdoor obstacles, especially in major city streets. In a study conducted in Norway, research found that, in a random sample of individuals with visual impairments, 68% of hard of seeing individuals have experienced at least one serious life threatening event that can be attributed to their visual impairment. Compared to the sample of random individuals without visual impairments, who scored a 60%, the likelihood of a hard of seeing individual encountering a serious lifethreatening result is higher. Employing some medium of navigating their surroundings more efficiently could pose a great benefit for those who are hard of seeing, as it can help reduce this statistic and help them stay safe.

The three methodologies overviewed in this paper include the Tango smartphone developed by the Synoptik Foundation, the 'Acoustic Prototype' proposed by Cambridge University, and lastly the WeWalksmartcane by WeWalk. Tango is a very similar solution to the EchoSense mobile app proposed in this paper. However, the Tango does not provide full 360 degree coverage around the user, and is only intended to help the user make sense of their surroundings- not to alert them of immediate objects in their path. The Acoustic Prototype developed from the Cambridge University utilizes many different mediums of interpreting the user's surroundings, as it uses microphones, infrared sensors, and has a built in camera into the glasses for image processing. While it is an all encompassing system, the prototype may overwhelm the user as well as impede their ability to interpret their own surroundings due to the forced use of headphones. Lastly, the WeWalk is a smart cane that aims to provide more intelligent feedback for the user, such as vibration for immediate objects in their path, and has an onboard GPS system for directions. However, the WeWalk does not provide full 360 degree coverage and the pricing may be out of some user's price range. Additionally, the advanced intelligent voice assistant feature on WeWalk is a subscription service. The EchoSense project aims to provide full 360 coverage for the user to alert them of immediate obstructions to their path as well as detailed object analysis using on board real time object detection.

The methodology proposed in this paper, the EchoSense, is a vest that the user can wear that alerts them of immediate objects and obstructions in their path using real time object detection, distance sensors, and haptic motors for feedback.

Within this paper, multiple experiments are performed to ensure the quality and consistency of the EchoSense project. The first experiment includes testing to find the average response time across different mobile phones to ensure the real time object detection was consistent in providing vibrations to inform the user of objects. In this experiment, we found that the app averaged around 9 frames per 10 seconds when processing with the real-time detection model. This is due to the YOLOv5 model running inferences in real time, which can be a resource intensive process depending on the hardware. The second experiment was performed to find the consistency in response times from the EchoSense v1 hardware. This experiment included sending test signals to the nrf52 hardware, and timing how long it took for the haptic motors to vibrate. The average vibration response time was around 3.478 seconds. This did include, however, the duration of sending requests to all of the haptic motors, not just one at a time.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Improving YOLOv5 Object Detection in Varied Environments

When implementing the Yolov5 computer vision model for real-time object detection, one potential challenge is ensuring accurate and consistent performance across various lighting conditions and environments [5]. Poor lighting or cluttered backgrounds can cause the model to misidentify objects or miss them altogether. To address this, I could use data augmentation techniques to train the model on a wider range of environmental conditions, ensuring it learns to recognize objects in diverse settings [6]. Additionally, optimizing the model for mobile devices by reducing its size and complexity without sacrificing accuracy could improve performance and reduce processing time.

2.2.Ensuring Seamless Bluetooth Integration in Flutter Mobile Apps

With the development of the Flutter mobile application, a major challenge lies in ensuring smooth communication and integration between the mobile app and the hardware (via Bluetooth). Mobile devices have varying Bluetooth capabilities, which could result in inconsistent connections or delays. To resolve this, I could implement Bluetooth connection retries and timeout mechanisms to handle disconnections gracefully. Additionally, ensuring that the app can run efficiently on both Android and iOS platforms, while maintaining a consistent user interface and performance across devices, would be essential to offer a seamless user experience for all users [7].

2.3.Optimizing nRF52 Bluetooth Stability for Real-Time Data Transfer

The nRF52 Bluetooth-connected hardware presents its own set of challenges, particularly with ensuring a stable and low-latency connection between the hardware and the mobile device. Bluetooth interference, weak signals, or connectivity dropouts can disrupt real-time data transfer, which is critical for immediate object detection feedback. To address this, I could use error correction and data packet acknowledgment strategies to minimize the impact of interference. Additionally, optimizing the firmware on the nRF52 to handle reconnections automatically and efficiently when signal loss occurs would help maintain a reliable connection, improving overall system stability and user experience.

3. SOLUTION

The three major components that make up the EchoSense project include the trained Yolov5 computer vision model, the Flutter mobile application, and the nRF52 hardware vest configuration. The flow of the EchoSense project starts when the user opens the mobile app and presses the “Configure Device!” button on the homescreen. The user is then brought to the configuration page, where they are able to find and connect to their EchoSense_v1 hardware. Upon connecting, the user can then head back to the “Take a Walk” page, where they can start walking around. When the user’s device camera identifies an object that is in their immediate path, the position of the object relative to the camera is calculated. Depending on the size and position, an appropriate signal is sent to the EchoSense hardware to vibrate the corresponding motor(s). Additionally, the EchoSense mobile app can work independently regardless of the presence of hardware, however, it is not capable of alerting the position of the object. While the EchoSense_v1 hardware is present, however, the user has the ability to tell what direction the object is in and get an idea of the size and danger level of the object. The EchoSense_v1 was programming using Arduino code, specifically referencing the appropriate Adafruit libraries.

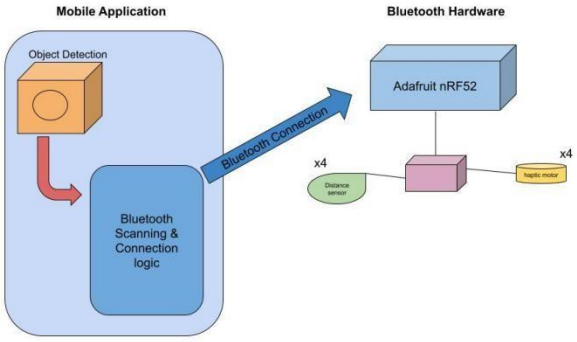

Figure 1. Overview of the solution

For the object-detection within the mobile app, we employed the use of the Yolov5 computer vision model, which is a model capable of detecting a set amount of objects. After detecting an obstacle that the program deems an impending danger that the user needs to avoid, the mobile app will notify the user through vibration signals

Figure 2. Screenshot of object detection

```dart
Future<void> checkBoundsAndVibrate(Size screen) async {
  if (yoloResults.isEmpty) return;

  double factorX = screen.width / (cameraImage?.height ?? 1);
  double factorY = screen.height / (cameraImage?.width ?? 1);

  // Calculate the actual bounds using the percentages.
  double xLowerBound = screen.width * .3;
  double xHigherBound = screen.width * .7;

  // Iterate through the detected objects.
  for (int i = 0; i < yoloResults.length; i++) {
    Map<String, dynamic> result = yoloResults[i];

    double objectX = result["box"][0] * factorX;
    double objectWidth = (result["box"][2] - result["box"][0]) * factorX;

    // Object boundaries in screen coordinates.
    double objectLeft = objectX;
    double objectRight = objectX + objectWidth;

    // Check if either edge is within the defined bounds.
    if ((objectLeft >= xLowerBound && objectLeft <= xHigherBound) ||
        (objectRight >= xLowerBound && objectRight <= xHigherBound)) {

      // Trigger vibration if available.
      if (await Vibration.hasVibrator() ?? false) {
        await Vibration.vibrate(duration: 200);
      }

      // Show the caution ring.
      setState(() {
        showCautionRing = true;
      });

      // Hide the caution ring after a brief period.
      Future.delayed(const Duration(milliseconds: 300), () {
        setState(() {
          showCautionRing = false;
        });
      });

      // Once vibrated for this frame, you can exit the loop.
      break;
    }
  }
}
```

Figure 3. Screenshot of code 1

In the screenshot above, the checkBoundsAndVibrate function is depicted. The checkBoundsAndVibrate function is responsible for taking the identified objects that the Yolov5 model has found and determining if they are in the immediate path of the user. If the object is in the immediate path of the user, a Vibration.vibrate() function is executed, vibrating the phone and alerting the user of the immediate danger. The first step of the function is to check and see if there are any yoloResults. If there are none present, then the function returns and no vibration is necessary. If there are yoloResults, the bounds of the screen and the 'path bounds' are found. Upon finding the bounds, the yoloResultsare iterated through, and are checked to see if they are in the 'path bounds.' If the object is found to be in the path bounds, then the phone will vibrate and display a caution ring around the screen.

For the mobile app itself, we used the Flutter framework to create the UI [8]. The mobile app's purpose is to allow the user to connect to the hardware via bluetooth, and to also serve as a transfer station between the video input and the vibration output. The app is where the computer vision model is embedded and running.

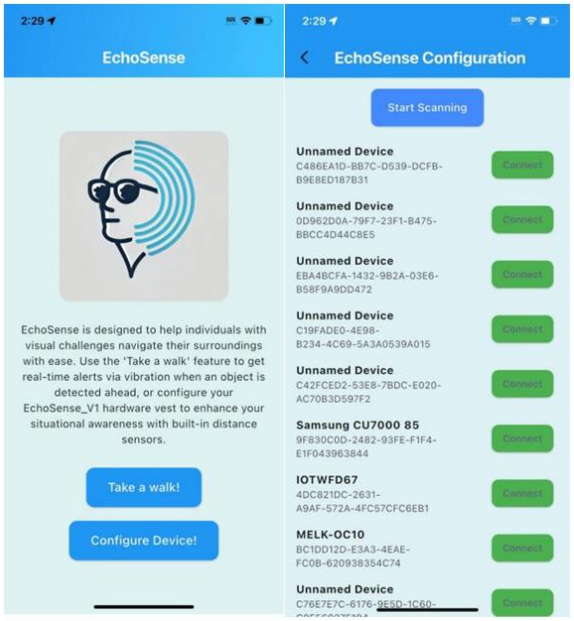

Figure 4. Screenshot of EchoSense

```
void _startScan() async {
  if (!_isScanning) {
    setState(() {
      _isScanning = true;
      _scanResults.clear();
    });

    try {
      print("STARTING SCANNING!");
      FlutterBluePlus.scanResults.listen((results) {
        setState(() {
          _scanResults = results;
        });
      });

      await FlutterBluePlus.startScan(timeout: const Duration(seconds: 5));
      setState(() {
        _isScanning = false;
      });
    } catch (e) {
      setState(() {
        _isScanning = false;
      });
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Error during scanning: $e')),
      );
    }
  }
}
```

```
void _connectToDevice(BluetoothDevice device) async {
  try {
    await device.connect();
    setState(() {
      _connectedDevice = device;
    });

    // Discover services and characteristics
    List<BluetoothService> services = await device.discoverServices();
    for (BluetoothService service in services) {
      print('Service UUID: ${service.uuid}');
      for (BluetoothCharacteristic characteristic in service.characteristics) {
        print('Characteristic UUID: ${characteristic.uuid}');
      }
    }

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Connected to ${device.name}')),
    );
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Failed to connect to ${device.name}')),
    );
  }
}
```

Figure 5. Screenshot of code 2

As seen in the above screenshots, the major Bluetooth connection logic is facilitated in two main functions: the _startScan() and _connectToDevice() methods. The _startScan method is responsible for the scanning of Bluetooth devices in the area, and keeping track of a list of them in the _scanResults List. Within the _startScan function, the use of the FlutterBluePlus library to start a scan using the startScan method can be seen. This FlutterBluePlus library is the Flutter library we used to interact with the onboard Bluetooth devices on the host device [9]. The _connectToDevice method is responsible for receiving the BluetoothDevice object to connect to as a parameter and establishing the connection to said device. After receiving the BluetoothDevice object, the function awaits the device.connect() request then awaits the discoverServices() request to find all available services on the device. The function then checks the services and any underlying characteristics and prints them. This is not for any particular reason other than to confirm their presence. Additionally, for both of these methods, the use of

try-catch statements was employed to safely handle any hardware or connection issues-which there frequently are with Bluetooth configurations.

The last major component within the EchoSense project is the nRF52 hardware vest configuration and the corresponding software for it, dubbed the EchoSense_v1. The onboard software is powered by aAdafruit nRF52 microprocessor, and the software is written in the familiar Arduino C language, with some appropriate Adafruit open source libraries [10].

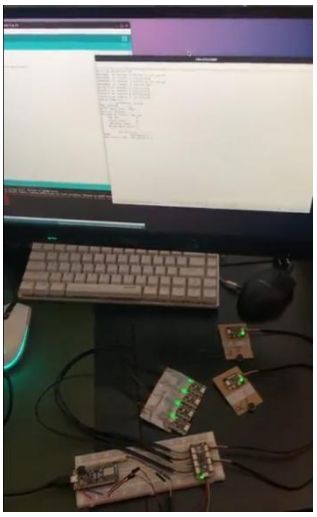

Figure 6. Screenshot of the hardware

EchoSense

```
#include <bluefruit.h>
#include <Wire.h>
#include <Adafruit_DRV2605.h>
#include <Adafruit_VL53L1X.h>

// Define the I2C address of the PCA9548
#define PCAADDR 0x70

// Create instances for the haptic motor controllers
Adafruit_DRV2605 drv1;
Adafruit_DRV2605 drv2;
Adafruit_DRV2605 drv3;
Adafruit_DRV2605 drv4;

// Create instances for the VL53L1X distance sensors
Adafruit_VL53L1X vl53_4;
Adafruit_VL53L1X vl53_5;
Adafruit_VL53L1X vl53_6;
Adafruit_VL53L1X vl53_7;

// Define BLE service and characteristics UUIDs
//#define MOTOR_SERVICE_UUID       "12345678-1234-1234-1234-123456789abc"
//#define MOTOR_CONTROL_CHAR_UUID  "abcdef12-3456-7890-abcd-ef1234567890"

uint8_t MOTOR_SERVICE_UUID[16] = {
  0x12, 0x90, 0x78, 0x56, 0x34, 0x12, 0x90, 0x78,
  0x56, 0x34, 0x12, 0x90, 0x78, 0x56, 0x34, 0x12
};

uint8_t MOTOR_CONTROL_CHAR_UUID[16] = {
  0x90, 0x78, 0x56, 0x34, 0x12, 0xEF, 0xCD, 0xAB,
  0x90, 0x78, 0x56, 0x34, 0x12, 0xEF, 0xCD, 0xAB
};

BLEService motorService(MOTOR_SERVICE_UUID);
BLECharacteristic motorControlChar(MOTOR_CONTROL_CHAR_UUID);
```

```
// Callback function to handle write events
void onMotorControlWrite(uint16_t conn_handle, BLECharacteristic* chr, uint8_t* data, uint16_t len) {
  Serial.print("Received write request: ");

  if (len == 1) {
    uint8_t motorNumber = data[0];
    Serial.print("Motor Number: ");
    Serial.println(motorNumber);

    if (motorNumber >= 1 && motorNumber <= 4) {
      Serial.print("Controlling motor number: ");
      Serial.println(motorNumber);
      controlMotor(motorNumber);
      blinkLED();
      Serial.print("Motor ");
      Serial.print(motorNumber);
      Serial.println(" successfully controlled.");
    } else {
      Serial.println("Invalid motor number received.");
    }
  } else {
    Serial.println("Invalid data length received.");
  }
  blinkLED();
}
```

Figure 7. Screenshot of code 3

In the screenshots above, several libraries are imported: Wire.h allows the Arduino to communicate with devices via the L2C bus, Adafruit_DRV2605 is used to control the haptic motor controllers, and the Adafruit_VL53L1X library is for handling and reading from the distance sensors. Immediately following the imports, instances to correspond with the actual hardware objects are declared including four haptic motor controller and distance sensor objects. Following that, the UUIDs for the service and characteristics that enable connected devices to control the motors are declared. As seen in the code above, they are declared in hexadecimal

arrays. Lastly the service and characteristic objects are initialized and given their respective UUIDs, and the onMotorControlWrite method as seen in the second screenshot is for handling the requests made from connected devices to the service.

4. EXPERIMENT

4.1. Experiment 1

It is important that the Bluetooth connection between the mobile phone and the nRF52 hardware is quick and stable, as an immediate response to potential danger is crucial to ensuring user safety.

To test to ensure a quick response time between the signal sent from the user's mobile phone and the nRF52 hardware response, an experiment will be performed to find the average time delay between sending the signal and the following vibration from the motor. Two experimenters will participate, the first being responsible for sending the signal from the mobile phone using the 'test motors' feature and the other being responsible for keeping track of the time. The first experimenter will press the test motor button and inform the second of their starting, upon which the second experimenter will start a timer. Upon receiving the first experimenter's signal to stop, the second will then stop the timer. The experimenters will record their findings in a spreadsheet and create a chart, finding the average time.

Figure 8. Vibration response time over 5 trials (seconds)

After performing the experiment and creating the chart that can be seen above, the average response time from the experiment was around 3.478 seconds. This data is not inherently surprising, as the experiment was quite simple, and the functionality we were testing has proven to be stable as Bluetooth is quite popular. Additionally, it is important to note that for this experiment we were waiting for the response of the second motor (the right side) as we did not have the first motor plugged in. As seen in the code below, the testing cycle works on a 3 second delay between motors. It is very possible that the few second delay that we experienced during our trials was due to this timed offset that we have in the code. After performing the experiment, we are quite satisfied with the consistency in response time and stability of the connection.

```
void _testMotors() async {
  if (_connectedDevice == null) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('No device connected')),
    );
    return;
  }

  setState(() {
    _isTesting = true;
  });

  try {
    for (int i = 1; i <= 4; i++) {
      await _sendMotorCommand(i, true);
      await Future.delayed(const Duration(seconds: 3));
      await _sendMotorCommand(i, false);
    }
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Error testing motors: $e')),
    );
  } finally {
    setState(() {
      _isTesting = false;
    });
  }
}
```

Figure 9. Screenshot of code 4

4.2. Experiment 2

Another crucial ability the EchoSense app should be able to maintain is the quickness of the Yolov5 model ability to process each frame and detect objects consistently. It is important that this work well as to ensure the safety of the user with any device.

To find the consistency and timeliness of the Yolov5-nano model on different devices, two experimenters will perform an experiment to find, in a given amount of time of 10 seconds, how many frames the mobile phone is able to classify with the model and if the model detects an object immediately in view. The first experimenter will start the real-time object detection processing and notify the timekeeping second experimenter to start the timer. Upon the time starting, the first experimenter will state 'yes' if the object is detected in that frame and 'no' if not. The total number of yes or no's will be totaled up at the end of the 10 seconds, and the count of accurate classifications will be recorded.

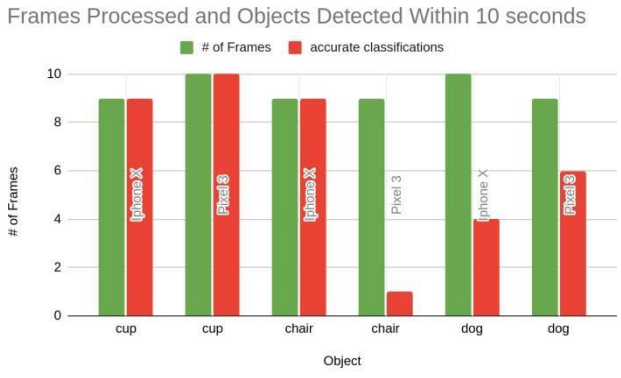

Figure 10. Frames processed and objects detected within 10 seconds

Following the experiment, it is clear to see that on both the testing devices, the Iphone X and Pixel 3, that the real-time object detection system seems to perform similarly. Taking a look at the produced data from our experiment above, it is clear to see that the average number of frames

processed is around 9 to 10 frames per 10 seconds. This averages to be around a frame a second, which we are satisfied with given the processing being done on mobile device hardware. However, as can be seen with the chair object trial, the Pixel 3 does drop off in the amount of accurate classifications. We believe this has to do with the quality of the Pixel 3's camera, as it is noticeably foggy and the device was on quite low battery. It is evident that the camera is a limiting factor here for the EchoSenseapp, however, most modern devices have cameras that are sufficient. Other than that, however, the objects do seem to be detecting quite sufficiently, given a clear line of sight to the object.

5. RELATED WORK

Figure 11. The smartphone

The Tango smartphone is a smartphone developed as a Synoptik Foundation research project [11]. The project is very similar to the solution described in this paper in that it proposes a developed system for hard of seeing individuals to identify objects in their path and navigate their surroundings. However, It is important to note that, while the project does include a unique developed software with an advanced spatial recognition system, it does not provide a 360 degree view of the user's surroundings. The focus of the Tango smartphone is to help the user mentally put together their surroundings, not to alert them of impending obstacles in their immediate vicinity.

Figure 1. Acoustic Prototype.

Figure 12. Acoustic prototype

The 'Acoustic Prototype' proposed by Cambridge University as a Sensory Navigation Device for the Blind [12]. The acoustic prototype system is an all encompassing system for hard of seeing individuals in that it takes in all information utilizing infrared distance sensors, microphones, as well as an onboard camera for image processing. The Acoustic Prototype holds all of these components in a pair of glasses and headphones that the user wears, and is able to monitor all of these in real time and give the user immediate unique sound feedback based on the corresponding sensor. However, the system is proven to work and detect objects in the user's vicinity up to only 5 meters away and does impede the user's ability to hear given the headphones are a must for the system to work. The EchoSense system works purely on vibrations, and as long as the object detection system is able to see the object it is capable of determining what it is.

Figure 13. The WeWalk

The WeWalk is a commercial replacement for the standard white cane for hard of seeing individuals that provides more encompassing and detailed feedback for the user [13]. The WeWalk comes with an onboard microcontroller, haptic feedback motors, a GPS system, and an ultrasonic sensor. The WeWalk utilizes the GPS system to provide directions utilizing the haptic motors to places given the user's input on their mobile device that is Bluetooth paired to the cane. Additionally, the cane is able to use the ultrasonic sensor to detect immediate objects in

front of them and give feedback to the user as well. While the WeWalk is undoubtedly an advanced tool, it does not provide the full 360 degree coverage that the EchoSense project does and cannot grade the severity of a situation given identified objects surrounding the user.

6. CONCLUSIONS

While the EchoSense mobile app and vest have proven to be a solid tool in navigation for the hard of seeing, it is clear that the project is limited to outdoor navigation [14]. When indoors, typically objects become much closer to the camera, which can inadvertently trigger vibrations. As a potential solution to objects being too close to the user inside, measuring the height and width of the specific object to establish a fabricated z-coordinate could be a way to determine how far away an object is. This would allow us to customize when the EchoSense should vibrate even further, keeping the distance in mind. However, currently, when inside the best results are collected when relying solely on the distance sensors to provide full 360 degree coverage. Additionally, EchoSense lacks the capability of reading words and conveying them to the user via text to speech or some other medium. Previously, EchoSense did not intend on supplying text recognition, however, the task is a crucial feature for the hard of seeing and would likely be a very useful addition.

Throughout the development of the EchoSense project, I gained extensive experience in application development using the Flutter framework with Android Studio. I also developed a strong understanding of building embedded systems and the processes involved in general hardware development. Finally, I learned how to create software that effectively applies machine learning and AI in practical scenarios [15]. Overall, the project proved to be an overwhelmingly positive experience and we believe the project to serve as a useful tool for the hard of seeing.

REFERENCES

[1] Islam, Rashedul, Rofiqul Islam, and TohidulMazumder. "Mobile application and its global impact." International Journal of Engineering & Technology 10.6 (2010): 72-78.

[2] Faust, Sebastian. Using Google ´ s Flutter framework for the development of a large-scale reference application. Diss. Hochschulbibliothek der TechnischenHochschule Köln, 2020.

[3] Zhao, Zhong-Qiu, et al. "Object detection with deep learning: A review." IEEE transactions on neural networks and learning systems 30.11 (2019): 3212-3232.

[4] Sorokin, Maks, et al. "Learning to navigate sidewalks in outdoor environments." IEEE Robotics and Automation Letters 7.2 (2022): 3906-3913.

[5] Sun, Ke, et al. "Study on rice grain mildewed region recognition based on microscopic computer vision and YOLO-v5 model." Foods 11.24 (2022): 4031.

[6] Maharana, Kiran, SurajitMondal, and BhushankumarNemade. "A review: Data pre-processing and data augmentation techniques." Global Transitions Proceedings 3.1 (2022): 91-99.

[7] Gyorödi, Robert, et al. "A comparative study between applications developed for Android and iOS." International Journal of Advanced Computer Science and Applications 8.11 (2017).

[8] Donglan, Zou, Mohamad Yusof Bin Darus, and AzlinBintiRamli. "Investigating Developer Experiences with UI Components in Flutter: Challenges and Implications." 2024 International Visualization, Informatics and Technology Conference (IVIT). IEEE, 2024.

[9] Oka, Dennis Kengo, et al. "Survey of vehicle IoTbluetooth devices." 2014 IEEE 7th international conference on service-oriented computing and applications. IEEE, 2014.

[10] Montironi, Maria Alessandra, Binsen Qian, and Harry H. Cheng. "Development and application of the ChArduino toolkit for teaching how to program Arduino boards through the C/C++ interpreter Ch." Computer Applications in Engineering Education 25.6 (2017): 1053-1065.

[11] Due, Brian L., et al. "Technology enhanced vision in blind and visually impaired individuals." Circd Work. Pap. Soc. Interact 3 (2017): 1-31.

[12] Dunai, Larisa, et al. "Sensory navigation device for blind people." The Journal of Navigation 66.3 (2013): 349-362.

[13] Saud, SafaaNajah, et al. "Smart navigation aids for blind and vision impairment people." International Conference on Computational Intelligence in Information System. Cham: Springer International Publishing, 2021.

[14] Brunes, Audun, and TrondHeir. "Serious life events in people with visual impairment versus the general population." International Journal of Environmental Research and Public Health 18.21 (2021): 11536.

[15] Helm, J. Matthew, et al. "Machine learning and artificial intelligence: definitions, applications, and future directions." Current reviews in musculoskeletal medicine 13.1 (2020): 69-76.