Team 27

Machine to Human Vision
Final Report

27 April 2024

Client	Sami Bensellam
Advisor	Alexander Stoytchev

Sami Bensellam	Project Lead
Alexander Black	Hardware Lead
Jacob Burns	Software Development
Yogesh Chander	Software/Hardware Integration
Jacob Lyons	Component/System Design
Sergio Perez-Valentin	Software Lead

Contact	sdmay24-27@iastate.edu
Website	https://sdmay24-27.sd.ece.iastate.edu/

Contents

List of Figures

1 Introduction and Background

1.1 PROBLEM STATEMENT

Visually impaired individuals, ranging from partial blindness to complete blindness, find it difficult to perceive objects and surroundings daily. This interference gets amplified when they are put into scenarios they have not encountered before or get disoriented. To mitigate these issues, we are attempting to create a product that can scan an environment and relay vital information to its users. We are attempting to do so in a manner that is intuitive and is representative to that of someone with normal eyesight.

1.2 INTENDED USERS AND USES

The main beneficiary of the project is targeted toward total blindness users, with a partial utilitarian use for partial blindness and other spatially impaired users. Such users will use the product to create mental images of objects and surroundings that will allow them to navigate within said environment. Expected use cases are:

1. **Navigating Hallways** - Guide the user through hallways, corridors, and tight spaces to prevent collisions with walls or furniture.
2. **Sidewalk Obstacle Avoidance** - Detect objects on sidewalks, such as parked cars, street signs, or pedestrians.
3. **Park Exploration** - Enhance outdoor experiences by identifying rocks, trees, and other features for the user.
4. **Office Navigation** - Assists in locating restrooms, break areas, and other sections of an office.
5. **University Navigation** - Guide the user by helping to identify buildings and entrances.

1.3 WORK CONTEXT

Below find similar ideas and products. The main difference and potential advantage this product has over existing solutions is that it is at a closer stage to release and relies on vibration rather than sound. Thus, being more suited for everyday use in different scenarios.

1. Manuel Zahn, Armaghan Ahmad Khan, Obstacle avoidance for blind people using a 3D camera and a haptic feedback sleeve. arXiv:2201.04453v1 [cs.HC]
2. Haptic Feedback testing adequacy for relaying 3D information: https://arxiv.org/pdf/2303.16805.pdf
3. The vOIce: https://www.nvaccess.org/audioScreen/

2 Revised Design

2.1 REQUIREMENTS AND CONSTRAINTS

The following is an outline of the requirements identified for the product to be usable and solve the problem statement. The figures provided were initial minimums, as most figures were exceeded in the final state of the product.

2.1.1 Functional Requirements

1. True modeling of surroundings.
    (a) Track objects up to 7 meters. **(constraint)**
    (b) Vibration displacement of 15 zones. **(constraint)**
    (c) Distinct frequencies per zone relative to detected distance in mm spans.
2. The frequency of data input and output is sufficient for users to stay orientated.
    (a) The stereo camera refresh rate shall have a minimum of 15 Hz. **(constraint)**
    (b) The response time for the user from the time of measurements shall be below 200 ms. **(constraint)**
3. Device power limitations shall stay reasonable for user usage.
    (a) Power draws shall not exceed 12 watts. **(constraint)**
    (b) Device runtime shall extend past 2 hours. **(constraint)**
4. Data transfer shall be within the bounds of use.
    (a) Spatial metrics to the breadboards shall be transmitted via USB.
    (b) Image data shall remain for intended use cases and discarded.

2.1.2 Usability Requirements

1. Comfortable and intuitive design language.
    (a) Easy to assemble and put on.
    (b) Can be worn for extended periods of time.
    (c) Has a sleek, low-profile design that does not stick out.
2. Climate, weather, and solid resistance.
    (a) Functional in a temperature range of 0 to 100 degrees Fahrenheit. **(constraint)**
    (b) Be IP24 rated - constant handling and splashing from all angles. **(constraint)**
3. Consistent and reliable modeling of surroundings.
    (a) Rescans of an object shall have repeatable measurements with a mean deviation of less than 10 mm. **(constraint)**

(b) Low visibility (fog, low light, etc.) must not impair accuracy.

(c) System logging for fast and easy diagnosis and repair.

4. Reasonable cost of production.

   (a) The first Prototype must not exceed $1000. **(constraint)**

2.1.3 Performance Requirements

1. The frequency of data input and output is sufficient for users to stay orientated.

   (a) Shall output to the user at least 15 times each second. **(constraint)**

2.1.4 Maintainability Requirements

1. Components must be modular for easy replacement.

2.2 ENGINEERING STANDARDS

Below are the identified engineering standards required for the project:

1. **IEEE 2671-2022** - This is the IEEE Standard for General Requirements of Online Detection Based on Machine Vision in Intelligent Manufacturing and is needed to standardize transmission processes, data formats, and quality standards for applications of machine-to-human vision.

2. Connection Standards:

   (a) **GMSL2 FAKRA** - Used for the transmission of video out of the camera.

   (b) **USB 3.1 Type C** - Used on the outside of the device for charging and other data transmission.

   (c) **IEEE-488 Ribbon Cables / IDC Cables** - Used as a bridge between the Raspberry Pi and haptic motors wire hub.

2.3 SECURITY CONCERNS AND COUNTERMEASURES

Within the system, the stereo camera poses the greatest security concern not only for the user, but for other individuals scanned by the stereo camera. Due to the inherent nature of how such devices work, the stereo camera is continuously taking live pictures to triangulate depth. That in itself is unavoidable, but to put users and affected passerby's at ease, all live images are contained to the internal memory of the Raspberry Pi and immediately disposed of when done with. In practice, a live image might exist at most for a fraction of a second.

Furthermore, all computation and storage is done locally, and the Raspberry Pi is blocked from connecting to remote signals such as Bluetooth or WiFi. Changes or updates of the firmware would either require a hard connection to a network, file transfer from a thumb drive, or the replacement of the SD card storing the OS. That is, unless physical access is acquired by an attacker, the device itself can be considered safe from malicious attacks that could cause harm to a user.

2.4 DESIGN EVOLUTION

The research and prototyping conducted during 491 saw positive results in the early versions of the full product, allowing for most of the initial design to be incorporated in the final product. Such aspects of the design included the head strap and mounting of the stereo camera, the chest rig to support the battery pack and Raspberry Pi, and the software itself had little modification due to success from early testing with a Kinect depth camera. However, the following shows the progression of the design choices that were determined not suitable for the final product.

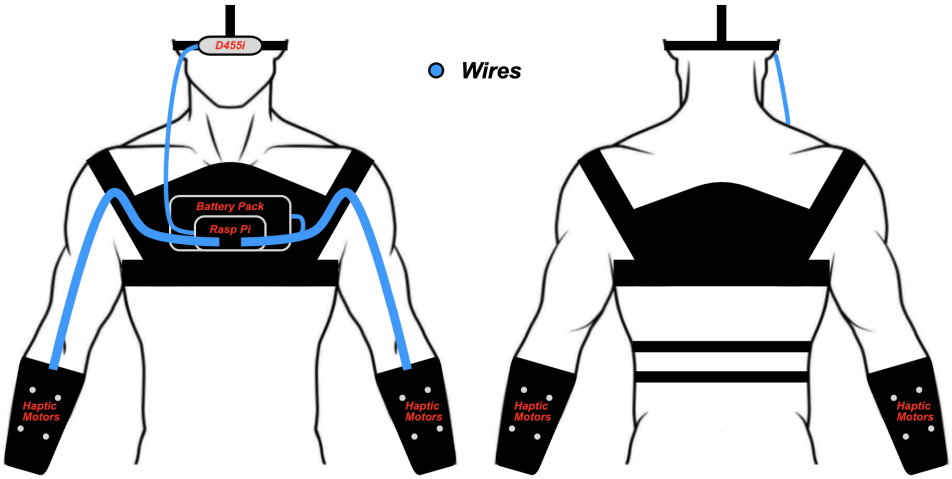

Figure 1: Full Concept Visual - Sleeves (Prototype)

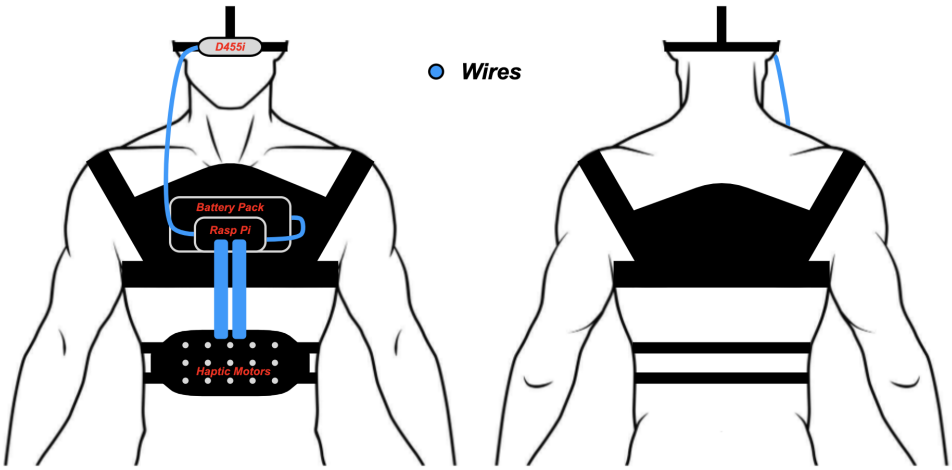

Figure 2: Full Concept Visual - Haptic Rig (Release)

The initial design called for the haptic motors to be worn on a sleeve around the user's forearms (Figure 1). However, complications quickly arose around wire management and mounting. Each haptic motor requires two lead wire, one for power and the other for ground. With 8 haptic motors per sleeve, it became unfeasible to practically route the wires in a way that was

not protruding to the user and easy to slip on. Furthermore, there was an added fear that the common movements a user would do with their arms daily would add to much strain on the wires, potentially ripping connections and decreasing the average life span of the product. Additionally, the sleeves were connected through velcro. Testing showed an unaccompanied user would greatly struggle attaching the two velcro strips. Therefore, the decision was made to mount the haptic motors to a haptic rig (Figure 2). This saw the reduction in count of the haptic motor grid from 16 motors to 15, with 3x5 shape. Routing wires became manageable due to the increase in surface area of the haptic rig. A circuit board with two IDC male connectors was attached, and acts like a hub for the rest of the haptic motor leads to run to. A female to female IDC bridge is used to connect the Raspberry Pi and the haptic motor haptic rig together.

Furthermore, due to the nature of the human stomach and the lining of the haptic rig, it is comfortable for the user to wear. Equipping the rig became easier for the user, as it now acts similar to a belt with two clips on the back for quick attach/release. Additional adjustability was also granted as a belt is more accommodating for different size user compared to a sleeve.

The final major design evolution was caused due to a lack of power delivered by the Adafruit. This was bypassed with the addition of a custom circuit board (Figure 3). Detailed specifications will be provided in section 3 - Implementation details.

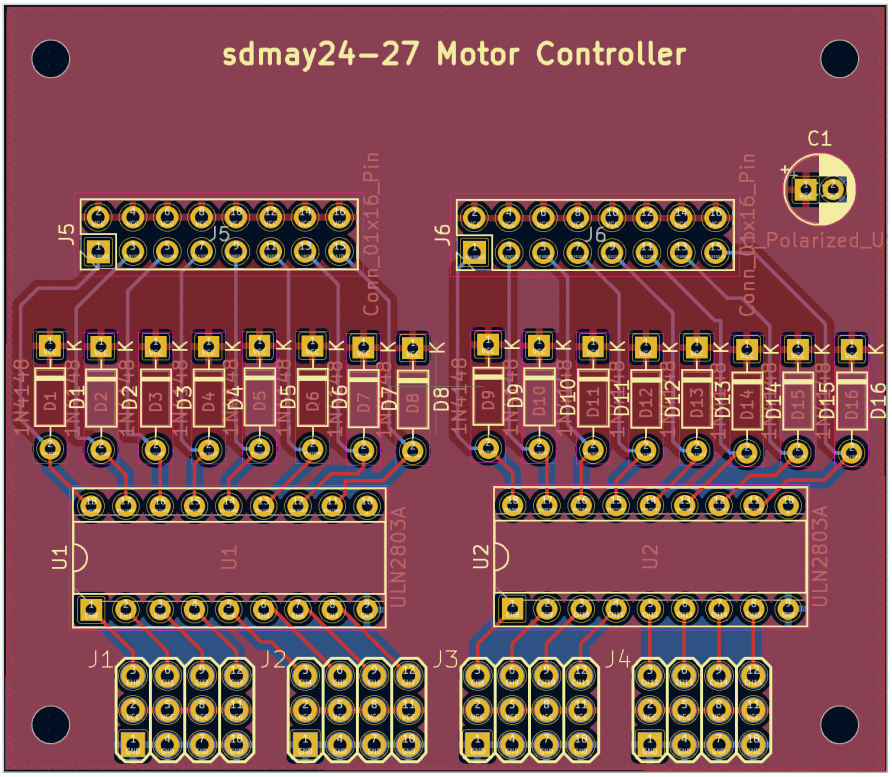

Figure 3: PWM Motor Controller

3 Implementation Details

3.1 DETAILED DESIGN

The design consists of five main components: a stereo camera, a computation device, a pulse-width modulation (PWM) configuration, a haptic rig, and a battery. The stereo camera provides depth information to the computational device, which translates that information into vibrations (Figure 4).

Figure 4: Intel Realsense stereo cameras.

3.1.1 Stereo Camera

For the design specifications, a camera that can accurately and reliably provide depth information in multiple settings was required. The camera needed to have high accuracy and reliability, along with the ability to perform in a majority of contexts, while being lightweight and modular. Various cameras were explored, including the Oak-D-Lite, ZED mini, D415, D435, D435i, and even Lidar modules like the L515.

Many of these cameras failed the reliability tests. The Lidar module failed to satisfy the requirement of working with natural sunlight, and its battery usage was also a concern. The ZED mini and the OAK-D-Lite failed the reliability testing in low light conditions. Finally, the D415 and the D435 variants failed the distance requirements of reliably assessing distance up to 7 meters. Consequently, the D455 was chosen as the most suitable option.

3.1.2 Haptic Rig

The haptic rig required the most design consideration, as it involved experimental territory. This involved understanding the neurosensitivity of different parts of the body, along with

considerations for comfort, extended use, and ease of removing and wearing the haptic rig. Many design iterations were explored, experimenting with the back of the palms, sleeves, back, and stomach. For these configurations, an optimal way to deliver vibration signals without the issue of carryover vibration was needed. The rig required the motors to be attached to a non-rigid material; thus, a multi-layered textile was used to dampen vibration and minimize carryover. A grid of 5x3 coin haptic feedback vibration motors (3 Volt) was attached to this textile.

Additionally, the textile required cable management to minimize the risk of the user being punctured by a wire. The mesh in the layering was used to mask the wires inside the fabric, minimizing risk and securing the wires in place to increase the device's reliability. All the wires were connected to an IDC female connector in the front, allowing for an easy way to connect the haptic rig to the computation device. The device was customized by utilizing elastic bands to optimize the rig's curve, ensuring optimal contact between the haptic feedback motors and the user.

3.1.3 Computation Device and PWM Configuration

The computation device needed to be reliable, capable of handling depth computations, power-efficient, and modular in its ability to connect to all the devices. Various options were considered, such as Arduino and other devices with similar configurations. However, the Raspberry Pi 4 was chosen due to its extensive documentation and widespread usage. It satisfied the power consumption requirements and had extensive documentation on hardware integration with the RealSense library for the D455 camera. Additionally, the Raspberry Pi has programmable GPIO ports, which are crucial for translating vibration.

Computation Bash scripting was utilized to run custom-made Python scripts that:

1. Connected to the RealSense D455 camera
2. Computed depth data
3. Compressed the relevant depth data in a 15x3 grid
4. Connected to the Adafruit HAT
5. Translated the depth data into numerical values denoting vibration intensity
6. Output the depth data into the corresponding PWM address in the Adafruit HAT

PWM Configuration The device required additional components on top of the Raspberry Pi. Firstly, it required the use of the Adafruit HAT for translating the GPIO signals to PWM signals. The PWM signals from the Adafruit HAT provided adequate voltage levels but failed to meet the haptic motor requirements in amperage for reliable sensing. The maximum output from the Adafruit HAT was 13 milliamps, which was below the specification for reliable motor sensing (tested to be 37 milliamps). Therefore, a custom circuit was created (Figure 5) to amplify the PWM signal by a factor of 3. The circuit was custom-made to be clipped on top of the Raspberry Pi and the Adafruit HAT, optimizing space and allowing for safety, mobility, and rigidity of the modified device.

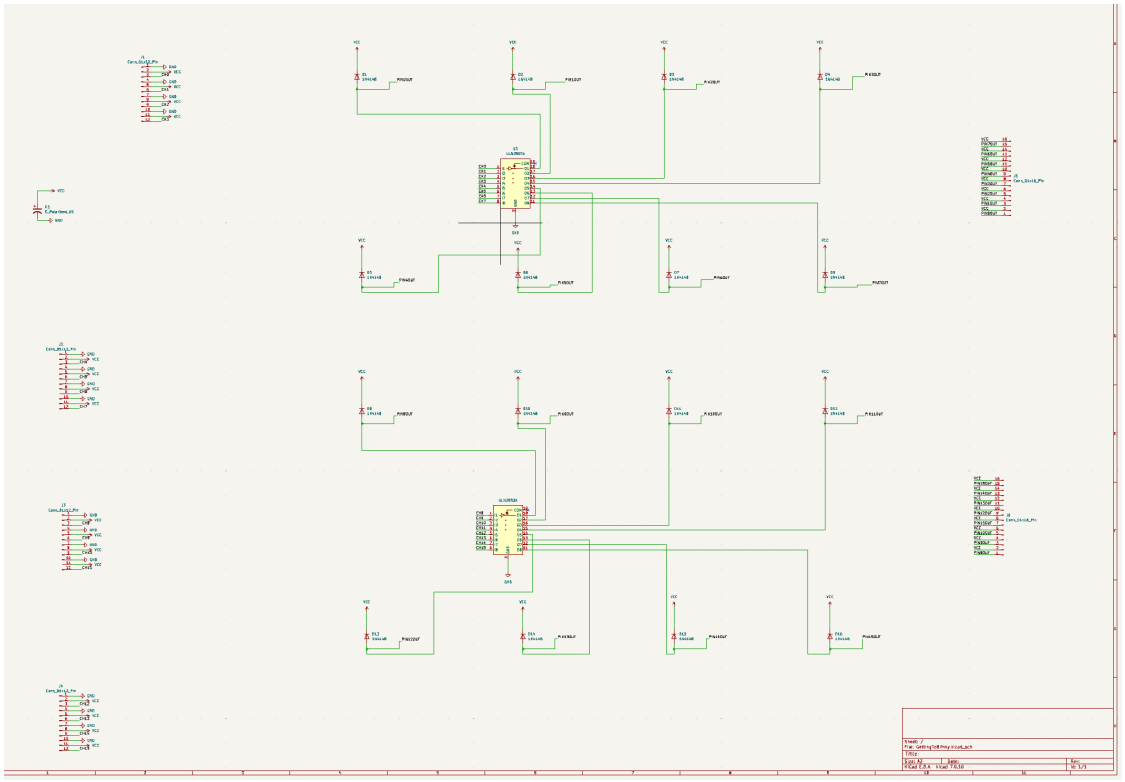

Figure 5: Custom PWM circuit diagram.

3.1.4 Battery

One of the main design considerations for a mobile system was the battery and power consumption of the different parts of the project. This required precise knowledge of the power consumption of each device at maximum throttle to ensure reliable performance under any condition. A diagram was provided with the different devices and their power requirements.

1. 15 Motor Power Requirements - **1.832 Watts Total**
    - Each motor is 37 mA, 3.3V at maximum vibration
    - 0.1221 Watts per motor at maximum vibration:
        - 0.1221 * 15 = 1.832
2. D455 Intel RealSense Camera - **3.675 Watts Total**
    - D4 Board Power Requirements at maximum power consumption:
        - 5.25 V
        - 700 mA
        - 5.25 V * 700 mA = 3.675
3. Raspberry Pi Power Requirements - **6 Watts Total**

In total, the power consumption is 11.5 watts. To facilitate the requirements, a battery that could last for an extended period and reliably power the device for more than 4 hours before needing a recharge was required. Safety was a critical specification, necessitating a battery with an AAA rating for commercial usage. Power banks for charging phones matched all the specifications in terms of safety, reliability, and form factor. The RETMSR 30000mAh power bank was chosen, which had USB-A outputs that could match the specifications.

The output was split into a USB-A to USB-C connector for the Raspberry Pi and a USB-A to DC Barrel Jack for the motors. The fast charging ability of this power bank is optimized up to 10 watts, with an additional 2 watts from the other USB port, ensuring the device's reliability at peak performance.

- 555mA (15 motors) + 700mA (RealSense Camera) + 1200mA (Raspberry Pi) = 2455mA

This allows the device to last approximately 10 hours.

3.2 DESCRIPTION OF FUNCTIONALITY

The device created is a closed-loop system that can be turned on by connecting the USB to the Raspberry Pi, allowing the user to put it on without the need for configuration. Every wearable portion is adjustable to the user, making it a production-ready product that can be used by a wide range of individuals, including those with visual impairments. The device works in multiple lighting conditions with high accuracy, including complete darkness, artificial lighting, and natural sunlight. We are able to manually configure the distance to a range of up to 15 meters with less reliable accuracy, but we were content with 5 meters since it provided a balance between the need for indoor navigation and outdoor navigation.

For haptic motor vibrations, our PWM is linear from 0 to 3.3V and 0 to 37 amps, given that testing showcased that a linear response was best recognized for the range in distance. The camera is strapped to a GoPro mount to mimic natural vision. Additionally, we tested out a different modality for the camera as a handheld device using a 3D-printed modality as an additional step that requires more exploration and testing, but initial testing showed promising results.

Our computation in terms of depth can reliably function at over 15 Hz with a delay below 400ms, bottlenecked by the hardware response time of the motors. Our software uses clustering methods, ensuring an accuracy of over 98% in terms of identifying the nearest object (Figure 6).

3.3 NOTES ON IMPLEMENTATION

One of the primary objectives of the project was to integrate various technologies in a cohesive manner, avoiding the creation of side projects that deviate from the main goal of establishing the primary loop. To achieve this, we focused on modifying and assembling the Adafruit HAT and the chest rig, while custom-creating the PWM signal amplifier, the haptic rig, and the software. By streamlining our efforts and concentrating on the essential components, we were able to efficiently bring together the necessary elements to create a functional and unified

system. This approach allowed us to dedicate our resources and attention to the core aspects of the project, ensuring that each component was designed and implemented to work seamlessly with the others.

3.3.1 Operating System Selection

For the Raspberry Pi software, we decided to use the Ubuntu OS instead of the native Raspbian OS due to the better documentation available for the integration of the RealSense software. This choice allowed us to leverage the existing resources and support for the RealSense camera, ensuring a smoother development process and more reliable performance.

3.3.2 Depth Measurement Challenges

One of the main drawbacks encountered with the stereo cameras was the presence of shadows in the depth map, which interfered with our depth measurements. These shadows created inconsistencies and inaccuracies in the depth data, requiring additional processing and filtering to mitigate their impact on the overall system performance.

3.3.3 Thermal Considerations

An unexpected issue that we faced during the implementation was the heating of various components, specifically during prolonged use of the Raspberry Pi, vibration motors, and camera. After extended periods of operation, we noticed a significant increase in the temperature of each component. While the system remained functional, it is crucial to conduct further testing in hotter environments to assess the potential impact of elevated temperatures on the device's performance and reliability. Addressing these thermal concerns may involve incorporating additional cooling mechanisms or optimizing the power management to ensure stable operation under various environmental conditions.

4 Testing

The Machine to Human Vision project contains many variables outside the control of feasible testing. Each user will be of different proportions, have different sensation receptors, and varying levels of ability to understand the incoming information from the haptic motors. That said, the following testing methods will attempt to mitigate any variability that might be encountered to produce the most linear results amongst users.

4.1 PROCESS

Initial testing saw individual components being tested to ensure they would be compatible for the project.

1. **D435i Camera** - The stereo camera was tested to ensure it can generate accurate depth information with low latency at a framerate of 30 fps. It was tested to ensure it registers important objects, such as a pole, even if they do not take up a significant portion of the stereo camera's vision. The testing was done by using a python script visualizer (Figure 6) displaying crucial data and having a team member validate the depth data retrieved by the camera. Other software functions were validated through unit testing.

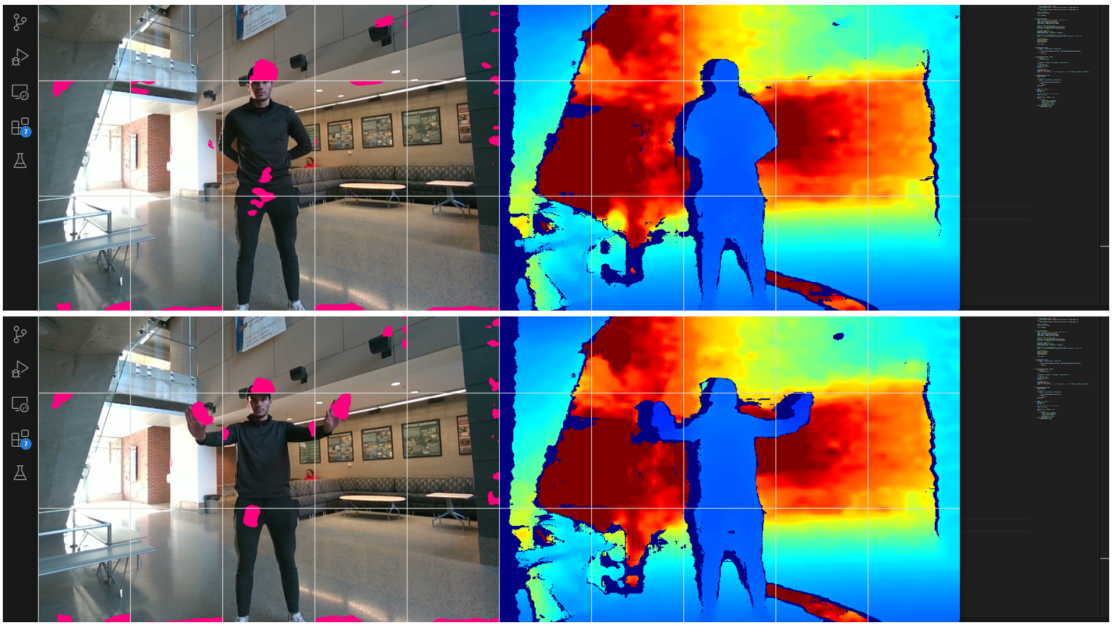

Figure 6: Python script visualizer for testing stereo camera.

2. **Raspberry Pi** - The Raspberry Pi was tested using a multitool to ensure it can power itself, the motors, and the camera at the same time. Tests were also conducted to ensure it can operate at 30 fps, reliably vibrate the motors as desired, and accurately interpret the camera's depth information. The tests were also done via human validation.

3. **Haptic Motor Array** - The haptic motor array was tested by using different arrangements

of the haptic motors, such as a pure 3 x 5 grid or a 3 x 5 grid that widens at the bottom in order to determine which is the easiest to navigate with. Effectiveness testing was done by having a person wear the array and attempt to navigate using it. Testing was also done with different fabrics and fabric thicknesses for the haptic rig to find a design that is easy to navigate with but does not damage the wearers skin. Each motor was put through a durability test to ensure they vibrate at the same strength using an accelerometer over long periods of time.

4. **Battery Pack** - The battery pack was tested using a multimeter to ensure it can generate at least 4.5 amps to power the Raspberry Pi, camera, and haptic motors.

Once the components were integrated into the full system, testing became dependent of the users perception from the vibrations. Since previous testing validated the components were preforming accordingly, late testing of the completed product was mostly calibration on different users to ensure vibrations felt differentiable at different distances.

Our tests were conducted in a controlled environment that simulated everyday occurrences a user might encounter. We would repeat scenarios such as individuals walking in front of the user, introducing random objects, and other edge cases. The user would then relay what they were perceiving and calibration would be adjusted accordingly.

4.2 RESULTS

Ultimately, the final results proved fruitful, and showed the idea itself was viable. Our testing gave us results aligned with the expectations we defined at the beginning of the project. Furthermore, multiple different uses who were able to try the product had positive feedback about their experience.

Figure 7: Shows an individual walking in front of a user, and the user identifying the correct direction of travel.

Figure 8: Shows a user navigating a hallway with individuals blocking the path.

Since the product experience can be subjective, having multiple users successfully complete the below objectives repeatedly proves the product is indeed capable of solving the problem statement.

1. **Navigating Hallways** - Guide the user through hallways, corridors, and tight spaces to prevent collisions with walls, people, or furniture.
2. **Sidewalk Obstacle Avoidance** - Detect objects on sidewalks, such as parked cars, street signs, or pedestrians.
3. **Park Exploration** - Enhance outdoor experiences by identifying rocks, trees, and other features for the user.
4. **Office Navigation** - Assists in locating restrooms, break areas, and other sections of an office.
5. **University Navigation** - Guide the user by helping to identify buildings and entrances.

5 Broader Context

5.1 OVERVIEW TABLE

	Area	Description	
	Public health, safety, and welfare	First and foremost, this project will most greatly affect the welfare of visually impaired people who use it by increasing their ability to navigate their surroundings. This can have many effects on them, such as decreasing safety risks and the likelihood of injury, as well as increasing their opportunities, particularly job opportunities, by allowing them to navigate better. One element of our project uses a camera to see what is in front of our user. This might conflict with privacy laws, and the general public's privacy.	
	Global, cultural, and social	One cultural consideration we must consider is people's comfort with the cameras, particularly in regard to when the product is used in an area where cameras would not be desired, like a public bathroom or locker room.	
	Environmental	How reliable, durable, repairable, and recyclable the project is may have an impact on the amount of waste sent to landfills, toxic substances from the battery in the environment, and on greenhouse gas emissions.	
	Economic	If our project helps visually impaired people to better navigate, then they may be better able to participate in the economic system by having jobs and having money to spend, thereby having a positive effect on the broader economy.	

6 Conclusions

6.1 REVIEW PROGRESS

The main conclusions we can draw from our device are the enormous opportunities that the field of cybernetics has in creating an alternative to vision. This project showcases that an alternative to human vision can be replicated to a large extent, demonstrating that it is not only viable but also has much more potential. We successfully created a well-documented blueprint, paving the way in the pursuit of one day providing visually impaired individuals with a product they can access that allows them to perceive their environment in a seamless manner. The main goals of this project were achieved: providing a viable navigation tool and visual alternative for people with visual impairments.

6.2 DESIGN VALUE

One of the key findings from this project is that using the stomach as a sensory modality is both viable and convenient, even with the relatively inexpensive 3.3V motors. This discovery opens up new possibilities for future products in the cybernetics field and beyond. The haptic rig could be used in other contexts, such as the virtual reality field, where many of the haptic feedback alternatives are much more expensive.

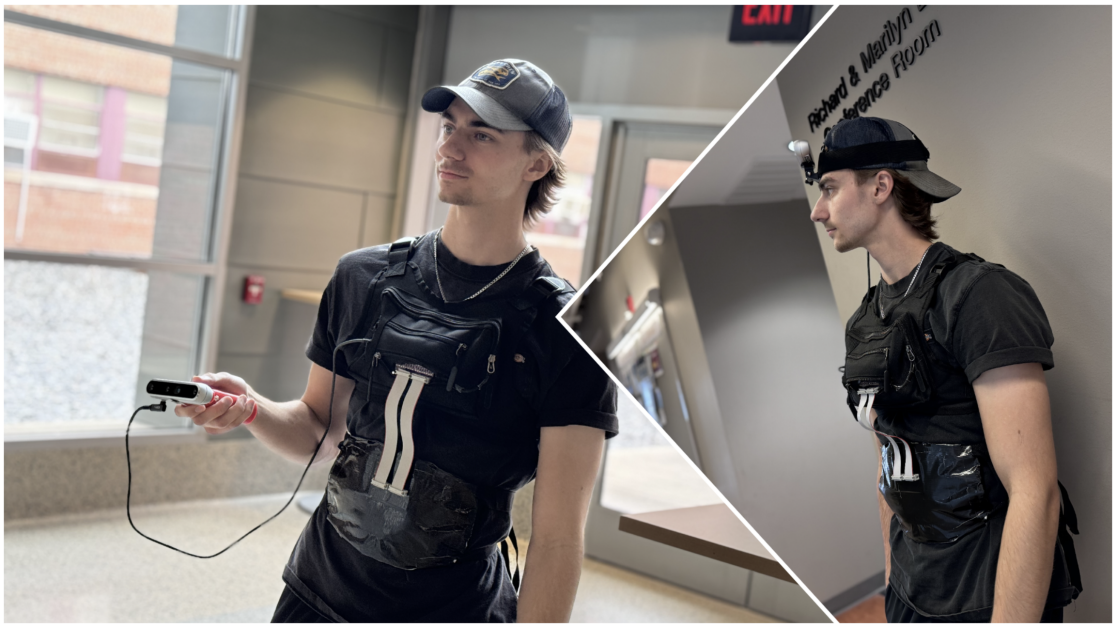

Figure 9: Shows a user with handheld stereo camera.

Another valuable insight is that the stomach is an effective modality, and increasing the number of motors, especially vertically, will enhance sensory awareness for the user. However, it is important to note that the GoPro headmount's location presented some challenges in

localizing objects' height or ground objects, which were not felt by the user. This issue was not observed with the handheld alternative (Figure 9), indicating that further testing and implementation are required to address this concern.

We believe that this end product could be tweaked into a viable product for people, although much more testing is required, especially for day-to-day usage. As an experimental product, this project sets a blueprint for future products in the cybernetics field, demonstrating the potential for creating affordable and effective sensory alternatives for visually impaired individuals.

6.3 FUTURE STEPS

While the current project utilizes artificial intelligence in its depth sensing ability and object recognition, I believe the next steps for the project involve providing more user interaction with the device, allowing it to better inform the user about their environment. My main plan following our progress would be to integrate AI algorithms that can perceive and identify the different objects viewed by the camera and use a speaker to provide the user with not only depth data but also the names of the objects detected. This enhancement would give the user a much better understanding of their surroundings and transform the device from a simple object detection tool into a valuable partner for the user. One of the other avenues we would have liked to pursue is the exploration of alternative stereo cameras, such as the Ray-Ban Stories, which provide a sleek alternative to the more conspicuous and pronounced D455 stereo camera. These options could offer a more discreet and aesthetically pleasing solution for users.

Another promising direction we investigated was using the stereo cameras as a handheld device. While we successfully went beyond our initial design and created a handheld prototype, we had a lot of success in the initial testing phase. This alternative form factor could provide users with more flexibility and control over the device's orientation and positioning. The rapid developments in Large Language Models (LLMs) and AI perception algorithms could enable the creation of a virtual assistant that can serve as a companion to the visually impaired person in their daily life. By integrating these advanced technologies, we could develop a product that provides the closest possible perception of the real world for visually impaired individuals. This is the ultimate goal we set out to achieve. Moreover, conducting extensive user studies and gathering feedback from visually impaired individuals would be crucial in refining the device's design, functionality, and user experience. By closely collaborating with the target user group, we can ensure that the product meets their specific needs and preferences, ultimately leading to a more effective and user-friendly solution.

In conclusion, the future work for this project involves leveraging advancements in AI, exploring alternative hardware options, and incorporating additional features and sensors to create a comprehensive and intuitive assistive device for visually impaired individuals. By continuing to push the boundaries of technology and prioritizing user-centric design, we can work towards our goal of empowering visually impaired people with a tool that greatly enhances their perception and understanding of the world around them.

7 Appendix

7.1 OPERATION MANUAL

No needed actions are required for use by the user once the product is on and vibrating. Looking around with the camera will cause the haptic motors to change in vibration, and with enough practice will allow a user to picture an environment. To achieve the best results, slowly scan your head around. Quick rapids motions will be hard to differentiate initially due to the body perceiving a single large vibration.

The following is a step-by-step guide on how to put on and connect the system. A diagram is also provided as a visual reference (left to right, top to bottom).

1. The product has two main clipping points. The first is the chest rig, and the second is the haptic rig. Slip the chest rig over the users head. Then proceed to clip the two connection points beneath the armpits. Move to the haptic rig, and proceed to clip the connection points behind the back as well. The straps can be adjusted to accommodate different users. Tighten until relatively snug.

2. Connect the usb-c cable into the stereo camera port located next to the head mount standoff. Slip the head mount straps comfortably over the users head and adjust the stereo camera. The stereo camera should be facing forward parallel with the eyes. The user may need to tilt the stereo camera slightly downwards if taller to get view of the ground.

3. Open the battery pack pouch and pull out the battery bank. Plug in both usb-a cables into the power bank. Put the battery bank back into the pouch and zip it shut.

4. Cable manage where needed.

If done correctly in the previous order, the system will boot up and begin operating as intended in around 20 seconds. Looking at different objects will vibrate the motors accordingly. If the system stops responding, reset it by unplugging replugging the battery bank cables.

For developer usage and debugging, unzip the Raspberry Pi pouch and plug in a micro hdmi cable for output. A mouse and keyboard can be connected on the side of the Raspberry Pi. All components have their own unique compartment, and can be swappable if damaged.

Figure 10: Operation manual visual for equipping product.

7.2 ALTERNATIVE/INITIAL VERSION OF DESIGN

During the early stages of the project, we explored various design options and conducted extensive testing to determine the most suitable components and configurations for our haptic feedback system. One of our initial considerations was the use of the Microsoft Kinect sensor, as we had prior experience with this device and its depth sensing capabilities aligned with our requirements. However, we ultimately decided against using the Kinect due to its lack of mobility, which was a critical factor in our design.

To gain a better understanding of the haptic feedback mechanism, we created several iterations of improvised forearm sleeves equipped with vibration motors. These early prototypes allowed us to investigate the intensity and perceptibility of the motor vibrations and calibrate them to a level where users could reliably distinguish between different vibration patterns. Through testing and refinement, we were able to achieve a level of precision where users could differentiate between vibration intensities in increments of 10%, with a margin of error of ±10%.

The development of these initial forearm sleeves provided invaluable insights into the interaction between the motors and the user's sense of touch. By experimenting with various vibration patterns, intensities, and durations, we gained a deeper understanding of how the haptic feedback could be optimized to convey information effectively. This knowledge formed the foundation for the design of our final prototype, as it helped us determine the optimal placement, configuration, and control of the vibration motors. Moreover, the process of iterating on the forearm sleeve design allowed us to identify and address potential challenges early on. For example, we discovered that the placement of the motors and the material of the sleeve played a crucial role in ensuring consistent and comfortable haptic feedback. We experimented with different fabrics and materials to find the most suitable option that would minimize vibration dampening while maintaining user comfort during extended use.

Another important aspect of these initial design iterations was the development of the control system for the vibration motors. We explored various methods for precisely controlling the intensity and duration of the vibrations, such as pulse-width modulation (PWM) and direct current (DC) control. Through rigorous testing and calibration, we determined the most appropriate control method that offered the desired level of precision and responsiveness. The insights gained from these early design iterations were instrumental in shaping the final prototype of our haptic feedback system. By understanding the capabilities and limitations of the vibration motors, as well as the user's perception and tolerance of haptic feedback, we were able to make informed decisions about the design and configuration of the system. This iterative approach allowed us to refine and optimize the design, ensuring that the final prototype was well-suited to meet the needs of our target users.

In summary, the initial design iterations, particularly the development of the improvised forearm sleeves and the exploration of the Kinect sensor, played a crucial role in the evolution of our haptic feedback system. These early experiments provided valuable insights into the technical aspects of the system, as well as the user experience and comfort considerations. By building upon these findings and continuously refining our design, we were able to create a robust and effective haptic feedback solution that has the potential to greatly enhance the spatial awareness and navigation capabilities of visually impaired individuals.

7.3 CODE

main.py - Main loop function. Initializes camera and Raspberry Pi components.

```
from stereo_scene import StereoScene

import sys
sys.path.insert(1, '/home/sdmay24-27/librealsense/release')
import pyrealsense2 as rs

import numpy as np
import cv2

import time
import math
import os

import board
import busio
import adafruit_pca9685

from adafruit_pca9685 import PCA9685


def config():
    # Configure depth and color streams
    pipeline = rs.pipeline()
    config = rs.config()

    # Get device product line for setting a supporting resolution
    pipeline_wrapper = rs.pipeline_wrapper(pipeline)
    pipeline_profile = config.resolve(pipeline_wrapper)
    device = pipeline_profile.get_device()
    device_product_line = str(device.get_info(rs.camera_info.product_line))

    config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)

    if device_product_line == "L500":
        config.enable_stream(rs.stream.color, 960, 540, rs.format.bgr8, 30)
    else:
        config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)

    # Start streaming
    pipeline.start(config)

    # Initialize the PCA9685 using the default address (0x40)
    i2c = busio.I2C(board.SCL, board.SDA)
    hat = adafruit_pca9685.PCA9685(i2c)
    pwm = PCA9685(i2c)
    pwm.frequency = 100  # in Hertz

    return pipeline, pwm


def capture(pipeline):
    # Wait for a coherent pair of frames: depth and color
    frames = pipeline.wait_for_frames()
    depth_frame = frames.get_depth_frame()
    color_frame = frames.get_color_frame()

    # Convert images to numpy arrays
    depth_map = np.asanyarray(depth_frame.get_data())
    color_map = np.asanyarray(color_frame.get_data())

    # Set depth upperbound
    threshold = 9999 # ~10 meters
    depth_map[depth_map > threshold] = threshold

    #depth_map = np.fliplr(depth_map)  # flip on y axis
    scene = StereoScene(depth_map, color_map, VER_CELLS, HOR_CELLS) # create scene

    # process points
    scene.scene_reduce()
    scene.get_cells()
    scene.get_points()

    return scene


def haptic(scene, pwm):
    for index in range(VER_CELLS * HOR_CELLS):
        try:
            pwm.channels[index].duty_cycle = scene.powermap(scene.points[index])
        except:
            continue


def stattrack(visualize, scene):
```

```
    if visualize:
        scene.visualize()

    try:
        STATS[0] = round(1 / (time.time() - START_TIME), 1)
    except:
        STATS[0] = 60
    STATS[1] += STATS[0]
    STATS[2] += 1

    os.system("clear")
    scene.print_meter_map()
    print("------ %3.1f fps ----- %3.1f average fps ------" % (STATS[0], STATS[1] / STATS[2]))


def cleanup(pipeline, pwm):
    pipeline.stop()

    # stop motors
    for index in range(VER_CELLS * HOR_CELLS):
        try:
            pwm.channels[index].duty_cycle = 0
        except:
            continue
    pwm.deinit()


if __name__ == "__main__":
    VER_CELLS = 3
    HOR_CELLS = 5

    # current fps, total fps, total frames
    STATS = [0, 0, 0]

    pipeline, pwm = config()
    try:
        while True:
            START_TIME = time.time()
            scene = capture(pipeline)
            haptic(scene, pwm)
            stattrack(False, scene)
    except KeyboardInterrupt:
        cleanup(pipeline, pwm)
```

stereo-scene.py - Custom stereo camera class that calculates depth data.

```
import cv2
import numpy as np
import math

class StereoScene:

    def __init__(self, depth_map, color_map, ver_cells, hor_cells):
        """
        :param depth_map: (2D array) full depth matrix
        :param color_map: (2D array) full color matrix
        :param ver_cells: (int) vertical cells
        :param hor_cells: (int) horizontal cells
        """
        if np.shape(depth_map) < (1, 1):
            raise ValueError("Depth map must be a 2D array")

        self.ver_cells = ver_cells
        self.hor_cells = hor_cells

        self.depth_map = depth_map
        self.color_map = color_map

        self.dm_cells = None
        self.cm_cells = None
        self.points = None

    """ ================================================================================
    StereoScene Modifiers - modifies the object
    ================================================================================ """

    def scene_reduce(self):
        """
        Reduce depth map shape radially until evenly divisible by cells.
        :return: None
        """
        height, width = self.dm_get_shape()
        if self.ver_cells <= 0 or self.hor_cells <= 0 or (height, width) < (self.ver_cells, self.hor_cells):
            raise ValueError("Cells count must be greater than 0, and less than shape")

```

```
40          lower = lambda l, f: math.ceil((l % f) / 2)  # calculates lower bound (length, factor)
41          upper = lambda l, f: math.floor((l % f) / 2) * -1  # calculates upper bound (length, factor)
42          check = lambda l, b: l if b == 0 else b  # catches upper bound if 0 (length, bound)
43          # recduce depth map
44          depth_map_out = []
45          for row in self.depth_map[lower(height, self.ver_cells): check(height, upper(height, self.ver_cells))
        ]:
46              depth_map_out.append(row[lower(width, self.hor_cells): check(width, upper(width, self.hor_cells))
        ])
47          self.depth_map = np.array(depth_map_out)
48          # reduce color map
49          color_map_out = []
50          for row in self.color_map[lower(height, self.ver_cells): check(height, upper(height, self.ver_cells))
        ]:
51              color_map_out.append(row[lower(width, self.hor_cells): check(width, upper(width, self.hor_cells))
        ])
52          self.color_map = np.array(color_map_out)
53
54      def scene_compress(self, shape):
55          """
56          Compress depth map to shape by averaging point clumps.
57          :param shape: ((int, int)) HEIGHT, WIDTH
58          :return: None
59          """
60          self.depth_map = cv2.resize(self.depth_map, (shape[1], shape[0]), interpolation=cv2.INTER_NEAREST)
61
62      """ ============================================================================
63      StereoScene Returns - returns an object
64      ============================================================================ """
65
66      def dm_get_shape(self):
67          """
68          :return: ((int, int)) depth map dimensions as tuple - HEIGHT, WIDTH
69          """
70          return np.shape(self.depth_map)
71
72      def get_cells(self):
73          """
74          Divides depth map into grid cells. Depth map shape must be divide cells evenly.
75          :return: None
76          """
77          height, width = self.dm_get_shape()
78          if height % self.ver_cells != 0 or width % self.hor_cells != 0:
79              raise ValueError("Shape must divide evenly by cells")
80
81          cell_height = height // self.ver_cells
82          cell_width = width // self.hor_cells
83
84          # create depth map cells
85          arr = np.array(self.depth_map)
86          self.dm_cells = (arr.reshape(height // cell_height, cell_height, -1, cell_width)
87                   .swapaxes(1, 2)
88                   .reshape(-1, cell_height, cell_width))
89
90          # remove depth shadows
91          bound_cells = []
92          for cell in self.dm_cells:
93              threshold = 200 # .2 meters
94              cell[cell < threshold] = np.average(cell)
95              bound_cells.append(cell)
96          self.dm_cells = bound_cells
97
98          # create color map cells
99          arr = np.array(self.color_map)
100         self.cm_cells = (arr.reshape(height // cell_height, cell_height, -1, cell_width, 3)
101                  .swapaxes(1, 2)
102                  .reshape(-1, cell_height, cell_width, 3))
103
104     def get_points(self):
105         """
106         Finds closest prominent depth point in a cell.
107         :return: None
108         """
109         self.points = []
110         for cell in self.dm_cells:
111             cell = np.array(cell).flatten()
112             cell.sort()
113             self.points.append(round(np.mean(cell[:round(len(cell) * 0.05)])))
114
115     def template_match(self, prominence, threshold, step):
116         """
117         Creates a template matrix from one depth value of shape determined by the percentage of depth map
        inputted.
118         Compares multiple template matrix's from max depth value to min depth value until match return value
        is
119         below threshold when compared to depth matrix. If no match is below threshold, returns -1.
120         The closer the match value is to zero, the more similarities there are between a template and depth
        map match.
121         :param prominence: (float 0-1)percent of the depth map to create depth template
122         :param threshold: (int >= 0) match threshold between depth template and depth map
123         :param step: (int > 0) step size between max depth and min depth
```

```
124             :return: a single depth point from depth map based on prominent mask algorithm,
125                      template cords for visualization
126             """
127             height, width = self.dm_get_shape()
128             template_shape = (math.ceil(height * prominence), math.ceil(width * prominence))
129 
130             for depth in range(self.dm_get_max(), self.dm_get_min(), -step):
131                 template = np.full(template_shape, depth)
132                 res = cv2.matchTemplate(self.depth_map.astype(np.float32), template.astype(np.float32), cv2.
    TM_SQDIFF)
133                 min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
134                 if min_val <= threshold:
135                     return (depth, min_loc)
136             return (-1, (0,0))
137 
138         def powermap(self, val):
139             """
140             Translates depth data to motor vibration output.
141             :param val: (int) depth distance
142             :return: (int) value of motor vibration output (0 - 65000)
143             """
144             val = val / 1000 # convert to meters
145             return round(-900 * pow(val-1.3, 4) - 8000 * val + 67500) # power distribution function
146 
147         """ ================================================================================
148         StereoScene Visualizers - visualizes StereoScene
149         ================================================================================ """
150 
151         def visualize(self):
152             """
153             Displays a window with camera perspective and additonal debugging data.
154             :return: None
155             """
156             # Apply colormap on depth image (image must be converted to 8-bit per pixel first)
157             depth_colormap = cv2.applyColorMap(cv2.convertScaleAbs(self.depth_map, alpha=0.03), cv2.COLORMAP_JET)
158 
159             # recolor depth points on color map
160             for index in range(self.ver_cells * self.hor_cells):
161                 for ir, row in enumerate(self.dm_cells[index]):
162                     for ic, col in enumerate(row):
163                         if col <= self.points[index]:
164                             self.cm_cells[index][ir][ic] = [127, 0, 255] # b,g,r
165 
166             # build first horizontal strip
167             recons_colormap = self.cm_cells[0]
168             for index in range(1, self.hor_cells):
169                 recons_colormap = np.hstack((recons_colormap, self.cm_cells[index]))
170             # add continual horizontal strips
171             for start_cell in range(self.hor_cells, len(self.cm_cells), self.hor_cells):
172                 tmp = self.cm_cells[start_cell]
173                 for index in range(1, self.hor_cells):
174                     tmp = np.hstack((tmp, self.cm_cells[start_cell + index]))
175                 recons_colormap = np.vstack((recons_colormap, tmp))
176 
177             # Draw grid
178             dim = (round(recons_colormap.shape[1]), recons_colormap.shape[0])
179             cell_dim = (round(dim[0] / self.hor_cells), round(dim[1] / self.ver_cells))
180             for x in range(self.hor_cells):
181                 for y in range(self.ver_cells):
182                     loc = (x * cell_dim[0], y * cell_dim[1])
183                     cv2.rectangle(recons_colormap, loc, (loc[0] + cell_dim[0], loc[1] + cell_dim[1]),
    (220,220,220), thickness=1)
184                     cv2.rectangle(depth_colormap, loc, (loc[0] + cell_dim[0], loc[1] + cell_dim[1]), (220,220,220)
    , thickness=1)
185 
186             # Show maps
187             cv2.namedWindow("Visual", cv2.WINDOW_AUTOSIZE)
188             cv2.imshow("Visual", np.hstack((recons_colormap, depth_colormap)))
189             cv2.waitKey(1)
190 
191         def print_ascii_map(self):
192             """
193             Creates ascii map on terminal of depth map.
194             :return: None
195             """
196             remapped_points = []
197             for point in self.points:
198                 if point >= 9999:
199                     remapped_points.append(" ")
200                 elif point > 8000:
201                     remapped_points.append("'")
202                 elif point > 6000:
203                     remapped_points.append("*")
204                 elif point > 4000:
205                     remapped_points.append(":")
206                 elif point > 2000:
207                     remapped_points.append("|")
208                 elif point > 0:
209                     remapped_points.append("X")
210                 else:
211                     remapped_points.append("?")
```

```
212         print(np.array(remapped_points).reshape(self.ver_cells, self.hor_cells))
213
214     def print_int_map(self):
215         """
216         Creates int map on terminal of depth map.
217         :return: None
218         """
219         remapped_points = [point for point in self.points]
220         print(np.array(remapped_points).reshape(self.ver_cells, self.hor_cells))
221
222     def print_meter_map(self):
223         """
224         Creates distance map in meters on terminal of depth map.
225         :return: None
226         """
227         remapped_points = ['{:.1f}'.format(round(point/1000, 1)) for point in self.points]
228         print(np.array(remapped_points).reshape(self.ver_cells, self.hor_cells))
```