**\_  
Stesias – Stereo Eyes for the Visually Impaired**

**Abstract**

**Jason C. D’Souza**\*  
University of San Augustin  
jasondsouza2003@gmail.com

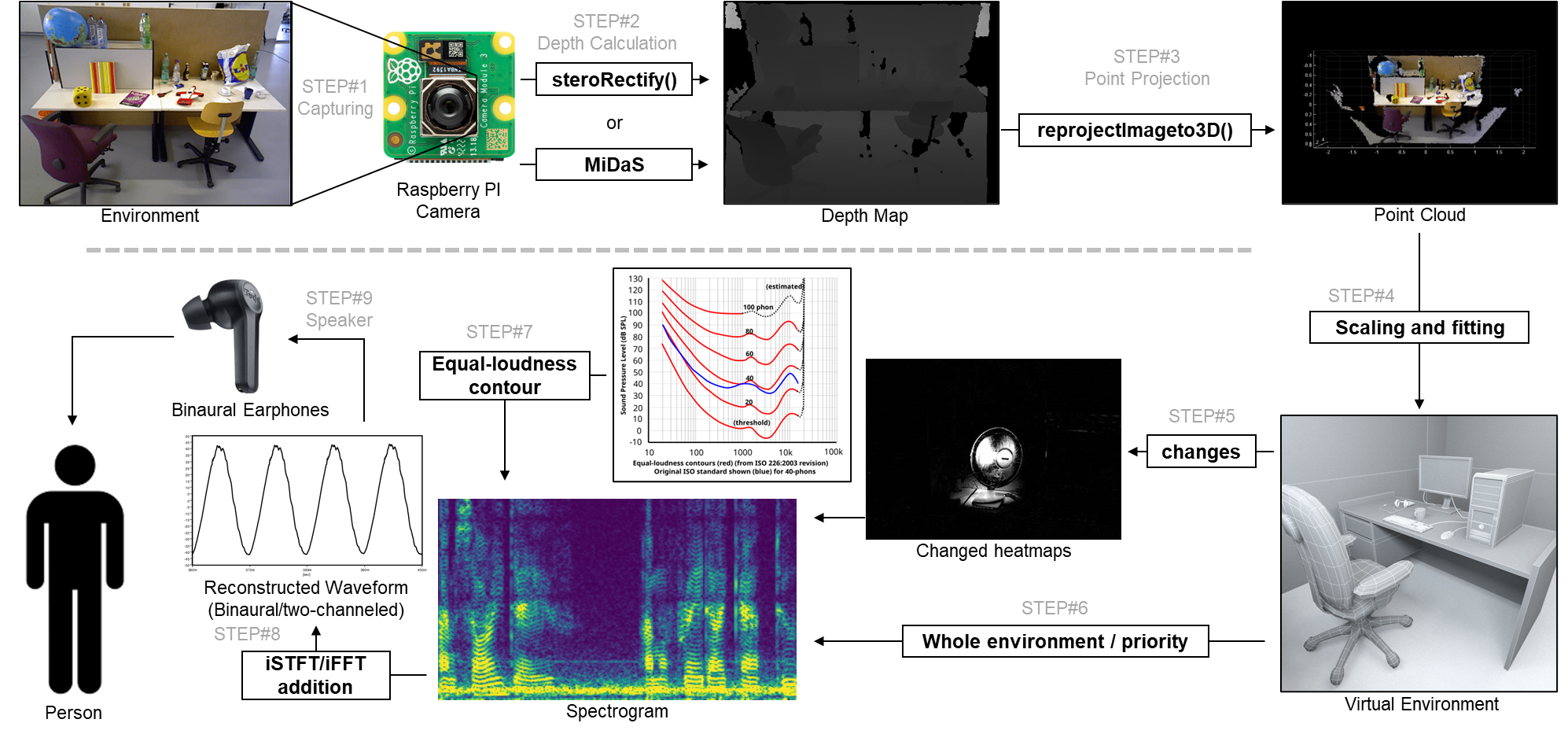
Although pre-existing methods of assisting the visually impaired already exists like walking crane, assistant dogs, eye surgery, or bionetics. They are neither capable of improving most aspects of their life has they are limited to one of their visual limitations like movement or color, nor are they wort their cost like if one is unable to afford surgery. This is where we created a device that converts visual information into audio information, this can be useful if the person is fully blind and need an echolocation-like scan of their environment or if a person with some level of blindness needs to read a text or identify an object far away. It implements several artificial intelligence depending on the condition of the person because not all blindness are the same, but it can include Image to Depth map, Text recognition, Image environment describer, and even Language Model if needed.

1. **Introduction**

The proposed device uses a Raspberry PI has its microcontroller and includes I/O devices to input visual and audial information, outputting audial information, and processing network actions like to deal with software or websites they can’t normally deal with. Below are the components

|  |  |  |
| --- | --- | --- |
| **No.** | **Component** | **Purpose** |
| 1 | Raspberry PI | Main Microcontroller |
| 1 | Microphone | Listens to user request *(optional)* |
| 1 | Binaural Earphones | Outputs visual information |
| 1 | Lithium Battery | Powers the whole device |
| 1 | Eyeglass foundation | Modified eyeglass to hold the components |
| 1 | Printed PCB Board | To be assembled with the glasses |
| 1 | ESP32 Dev Board | Creates a RTSP connection *(if using stereo)* |
| 1 | Stereo Camera | Records two video streams *(if using stereo)* |
| 1 | Raspberry PI Camera | Records a video stream *(if using pi camera)* |
| 1 | Oscilloscope | Detects user head movements |

1. **Method**

This is the whole diagram of the program dataflow, including only the main priority of the program (Visual to Audio Information):  


* 1. **Video Capturing**

There are two options for video capturing depending on our resources, we could use

* A Raspberry pi camera so that it directly connects to the raspberry pi without using a RTSP protocol or networking, and has a higher resolution but requires an AI model named MiDaS to convert the single image into a depth map
* A Stereo camera that either connects to an esp32 microcontroller which then records the video stream has an RTSP protocol which its hosts to the Raspberry PI Access Point
  1. **Depth Calculation**

For the program we used C++ with OpenCV has any alternative like python is too slow or doesn’t have enough online resources like documentations. If the video is through a Raspberry pi camera then the MiDaS model is downloaded and loaded upon program startup which it then processes each unbuffered(so its realtime) individual frame converting it into a depth map. If its through a stereo camera then each pair of unbuffered frame is compared with the “stereoRectify” method to get the depth map, it is a Horizontal stereo setup therefore the equation for Q(depth) based on C1(Camera1) and C2(Camera2):

and thus

* 1. **Point Projection**

The depth map is then converted to a Point Cloud through using OpenCV’s built-in “reprojectimageto3D” method whichs transform a single-channel disparity map(depth map) to a 3-channel image representing a 3d surface where pixels (x,y) is to (X,Y,Z):

* 1. **Scaling and Fitting**

Using the previous Point cloud instance and the change in the Oscilloscope’s measurement, the Point cloud for that instance can be properly mapped into the virtual environment. One problem is that the depth map especially if it’s calculated by an AI algorithm like MiDaS should not be Normalized but instead relative to the captured image. To fix this we can assume that the distance represented in the depth map has a coefficient times the actual distance, to find this coefficient is through using a Root Algorithm, starting from 1 and then increasing or decreasing based on the error margin.

* 1. **Changes in Environment**

Our senses are more sensitive to the logarithmic changes in our environment that linear or static information this is called the “Web-Fechner law”. By applying this logic then the device shouldn’t keep on converting the same environment into audio information, but it should convert the changes in the environment or if the user moves their head a certain way. It also This way the user won’t be overwhelmed with excess information, and since the volume is proportional to the change relative to distance to the user, it allows the user to deal with focusing on near objects even when in a chaotic environment like a party.

* 1. **Environment Priority**

The final result will be a spectrogram, the frequency, volume, and binaural mix can be configured to whatever the user prefers or most prioritize. Like the frequency represents the color of the area, the volume represents the movements of the area, and the binaural mix will based on the direction of the area. Another coefficient represents the whole static environment, since moving objects relative to the observer have a higher priority, the rest of environment have their own coefficient of volume which overtime goes to 0 has the user doesn’t need to know the static environment all the time, but if the user prefers to see the environment again the user can rotate their head in a preferred axis of rotation like the X axis for the coefficient to average to 1 so that the user can hear the whole environment. But this time it has a different configuration has the original configuration would be confusing has normally the environment has a lot of colors and the would-be converted audio with be just a pile of noise, to fix this the frequency now represents the Y distance from the user, volume represents the distance from the user, and the binaural mix represents the same thing, the direction of the area.

* 1. **Equal-Loudness Contour**

Even with the same volume, two different frequencies will sound like they have different volumes. This is because of the “Equal-Loudness Contour”, so when constructing the final spectrogram we would have to take into account this effect so that if there are two static objects infront of the user, one up and one down it wouldn’t sound like the upper one is nearer.

* 1. **Reconstructed Waveform**

Now that we have two spectrograms representing the left and right channels of the earphones, we then convert this spectrogram into a live waveform through the Inverse-Short-Time Fourier Transform(ISTFT) using the “fftw3.h” library, and the “portaudio.h” audio library.

* 1. **Speaker Output**

The earphone or small speaker that is either connected to the raspberry pi through the audio jack or the GPIO pins receive this audio information and then outputs them to the user

1. **Future/Possible implementation**
   1. **Portable Phone**

The raspberry pi is a single board computer, it is therefor also capable of navigating the web and apps. Just with a microphone to receive commands and a speaker to output the results, the user would be then able to navigate the web without touching the device at all.

* 1. **Smart assistant**

With some custom commands through a microphone, different AI models or APIs can be triggered like Object detection so that the user can also know what is approaching them, Text Recognition so the user can also read what is infront of them, Facial Recognition so the user can identify different people, Google Maps so the user can navigate to their destination, and even a Language Model so that the user can make more specific commands or discuss with the AI.

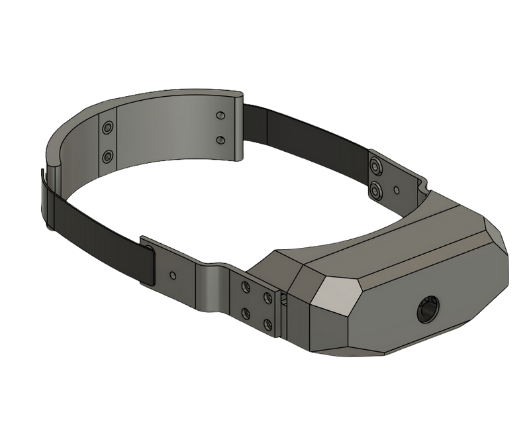
* 1. **Extended eye**

Since the visual recognition is captured by a camera, the camera can be switched depending on the desired measurement like temperature through a Thermal camera, or just the environment using a night goggles. This way the user can also have an extended range of vision if desired

* 1. **Argumented Reality**

This can also act has a smart glass for those that are not visually impared, and by using like an OLED glass piece or a reflected mirror then images can be projected upon the environment allowing the user to have additional information and just visual like with Google maps or object detection or zooming, etc.

1. **Implementation**

The device is a modified glass/goggles with a PCB printed circuit on the back of glass holding the raspberry pi, battery, etc. While the raspberry pi camera or stereo camera is in the top front of the glasses, the speaker/microphone or earphone is in each side of the glasses. Below are previous concept design of the device:  


1. **Conclusion**

{{ Create conclusion }}

1. **References**
   1. Hartley, R., & Zisserman, A. (2019). *Pinhole camera model*. HediVision. https://hedivision.github.io/Pinhole.html
   2. *Camera calibration and 3D reconstruction*. OpenCV. (2024). https://docs.opencv.org/3.4/d9/d0c/group\_\_calib3d.html