Digital Guide Dog

Dongxun Sun
UMass Amherst
Amherst, MA
dsun@umass.edu

He Zhang
UMass Amherst
Amherst, MA
hezhang@umass.educreate

Hui Sun
UMass Amherst
Amherst, MA
huisun@umass.edu

***Abstract*—This paper is about a system design to help visually impaired people avoid obstacles in their way while they are walking. The system has a belt-shaped design with two cameras, one on each side, pointing in the direction of the user's walk. It uses image processing with a Raspberry Pi microcontroller and three vibrators to warn the user about obstacles. This design combines the camera system and the Raspberry Pi to generate a real-time depth map of objects confronting the user and warning the user when needed.**

***Keyword*—image processing, depth map, equipment aid for daily basis.**

I. Introduction

There are about 10 million visually impaired people in the US. It can be difficult for them to navigate obstacles in their way. Some of them use guide dogs[1] for navigation, but it's not easy to train a dog to be a qualified[2]. In this project, we offer an easier way for individuals to avoid obstacles using two vibrators. Our design uses devices such as a single-board computer, cameras, and an Inertial Measurement Unit (IMU) to detect obstacles. Our product design is a wearable device, that is worn on a belt. The design's requirements and specifications are as follows:

- Works during day time.
- Sufficient battery power to finish a trip. Working time: 2 hours
- Works for a person with a normal walking gait.
- Detects objects within a 90 deg horizontal FOV and at a range of 2 – 10 feet
- Light enough for people to easily use and wear on the waist. Weight <= 1.5 lb
- Detect objects with a size of larger than 50 sq inches

The main functions of the system are as follows. Cameras gather live video and send it to the Raspberry Pi. The main system functions are as follows: A camera takes live video that is sent to the main processor (a Raspberry Pi); then, a real-time distance-determining algorithm acts on this video to trigger vibrators to signal the user; and finally, the IMU helps to give cues to the user to aim the camera along their heading. We also implemented a distance-calculating algorithm, improved the stability of the system, and upgraded the design into a wearable device.

II. Related Work

There are already several existing products to aid visually-impaired people, including a guide dog animal, a walking stick, and wearable electronic eyeware. We analyzed them and tried to come up with a better design.

A. Guide dog

There are about 10 million visually impaired people in the United States. If they would like to live their life to the fullest, the team believes that using a guide dog is an effective way to help them. However, it can be hard to train a guide dog. According to PUPPYINTRAINNING.COM[3], it takes about 8 months to train a guide-dog trainer, and another 14 months to train a dog to be ready to guide visually impaired people. A dog needs to bond with its owner. The cost is also significant. According to PUPPYINTRAINNING.COM[3], approximately $40,000-$60,000 is needed to properly train a qualified guide dog. Compared with our project, we only have a budget of $500, and we are able to deliver a working prototype that alerts people to avoid obstacles in their way.

B. WeWalk

WeWalk is a camera-aided walking stick[4]. It can detect obstacles up to 160 cm away and has Bluetooth connectivity to a smartphone. It also has 20 hours of battery time, is controllable through a touchpad, and has a length of 30 cm. These are some nice features that can help visually impaired people to navigate the world. However, we think the most important feature of a visual aid tool for a visually impaired people is to allow hands-free use. Our project uses a belt, which can be tied on the user's waist to free their hands. Tying the camera on the waist can help stabilize the camera. The belt can focus the camera in the user's walking direction so it doesn't need to be constantly recalibrated. From the project design perspective, it is also hard for visually impaired people to use a smartphone. Bluetooth can be used for testing purposes, but for a final product, we don't see how a phone can help the user avoid obstacles in their way. The cost of WeWalk is about $500. Our product costs less than $500.

C. Acesight

Acesight is wearable electronic eyeware, helping people with low vision see the world around them[5]. This product is designed to help visually impaired people see things instead of warning them about obstacles. It has 2 megapixel cameras to capture visual information, 45 degree of field of view, and a floating reading mode. The drawbacks of this product are conspicuous. Due to its 2 megapixel cameras, it costs around $5,000 per unit. Despite its 2 cameras, it can only cover 45 degrees of the users' direction. Our project can cover about 90 degrees and can send alerts when needed. Our design is less conspicuous and more comfortable to wear than Acesight.

III. Design

The design includes a belt worn on a user's waist. There are two cameras for image gathering. An IMU balancer implemented in with system detects if the cameras are in the right position. The data gathered by cameras and the balancer flows to a Raspberry Pi for processing. If the distance calculated by the Raspberry Pi exceeds a trigger value, the vibrator will be activated to warn the user about an obstacle. The block diagram of the system appears in Fig.1.

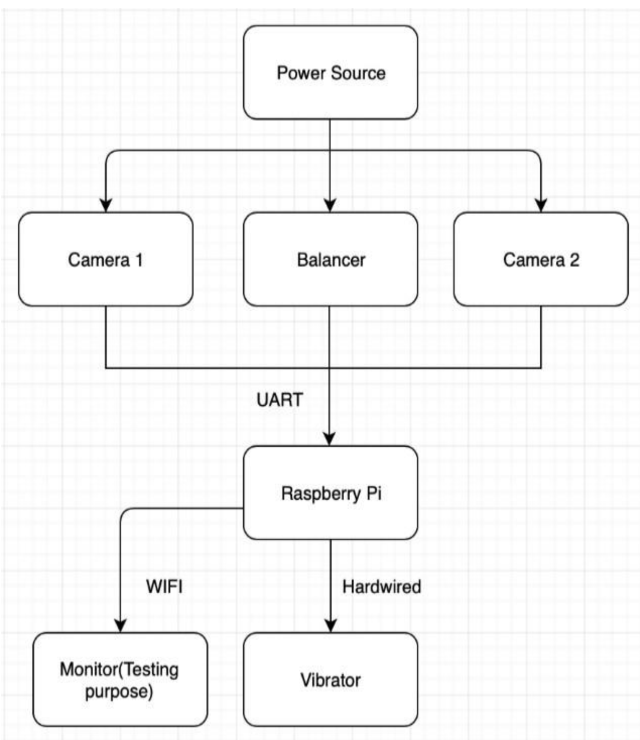


Fig.1. System block diagram

A. Software Design

In our design architecture, there are four logical layers: the hardware layer, hardware drive interface layer, core algorithm layer, and user interface layer. The hardware layer, in yellow in Fig.2, is a collection of physical sensors, such as cameras, the IMU, and the vibrator. The hardware interface layer (orange colored) is responsible for interacting with the physical sensors and obtaining raw data. The blue block is our core algorithm layer. It processes the raw data to determine depth information and decisions. The upper green block is the software interactive layer which is used for user communication.

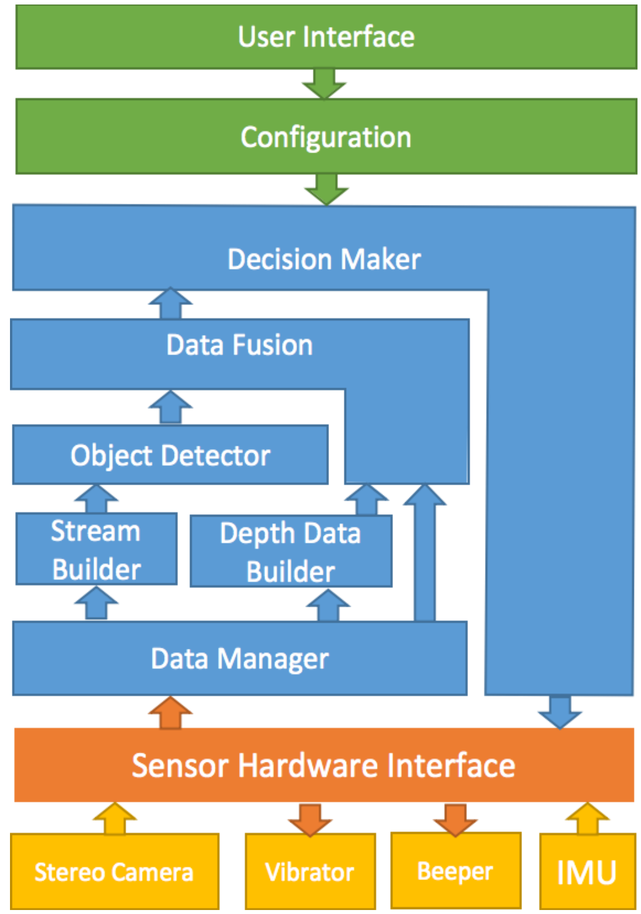
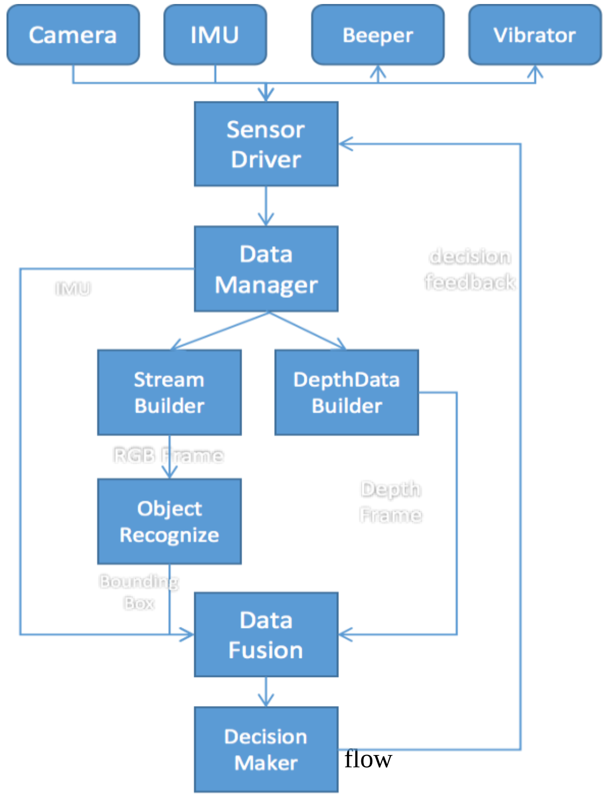


Fig.2. Software architecture

The data flow is shown in Fig.3. Raw data consists of 32 bit RGB pixel data from the cameras, angular velocity, linear velocity, and orientation data from the IMU. The data passes through the sensor driver. The data manager is responsible for allocating the buffer and storing the streamed data. The stream builder and depth data builder construct the RGB and depth frames. After object recognition, object size, type, location and depth data are delivered to the data fusion module. The decision maker module generates feedback to the sensors. The application user interface we designed for our product is shown in the Appendix.



flow

B. Hardware Design

The digital guide dog detects obstacles using cameras. The IMU sensor corrects the tilt state of the device in real-time, and uses a vibration sensor to indicate obstacles ahead. We chose the Raspberry Pi 4B for data processing. Advantages of using this development board are[6]:

- It is based on the Cortex A-72 processor, a 1.5-GHz CPU, and 4GB RAM which provide real-time video acquisition and processing;
- It has significant General Purpose Input/Output (GPIO) pins. The Pi can respond in real-time, modify the working state of the sensor and read data
- It is small and can be used to control sensors. This feature is in line with the lightweight characteristics of wearable devices.

We use the power port, Raspberry PI IIC interface and the PWM function of the GPIO port. The IMU sensor data is read by the IIC interface. The GPIO port with PWM can control three vibration sensors at different frequencies.

We used the integrated Intel D435 industrial camera sensor to collect photos and video information (Fig.4). An IR projector and RGB module are used with one camera on each side. The combination of a wide field of view and a global shutter sensor on the D435 makes it a preferred solution for applications such as robotic navigation and object recognition[7].

Fig.4. Intel D435 sensor physical image, include 2 cameras and 1 IR projector

Its specification parameters are shown in Table I. It can identify objects at distance of nearly 10 feet, with a FOV of 90 degrees. Image and video data transmission are obtained through a USB-C port. The DGD identified obstacles within 10 feet in front of the user.

TABLE I. SPECIFICATION OF THE INTEL D435

Features	**Use Environment**	**Maximum Range**
	Indoor/Outdoor	Approx. 10 meters. (accuracy depends on calibration, scene, and lighting)
	Image Sensor Technology	
	3µm x 3µm pixel size	
Depth	**Depth Field of View (FOV):**	
	Approx. 90° x 60° x 95°	
Components	**Camera Moudle:**	IMU Sensor
	Intel RealSense D430 RGB Camera	3 Vibration Sensors
Physical	**Length x Depth x Height**	**Power**
	90 mm x 25 mm x 25 mm	5V, 2A

Through communication between the Intel D435 and the Raspberry PI, video data processing can be realized. We successfully used the D435 to capture a video stream in real-time and process it as a depth map. We also used the IMU sensor to obtain the location of the camera module in real-time. When the camera pitches, the gyroscope on the IMU sensor checks the rotation and feeds the information back to the Raspberry PI, alerting the user about an incorrect position.

We used a MPU9255[8] shown in Fig.5 as the IMU sensor. It communicates with the IMU through IIC, reads the output of the gyroscope in real-time, and provides feedback. Its IIC communication interface is shown in the schematic diagram in Fig.6. The hardware specification table is shown in Table II.

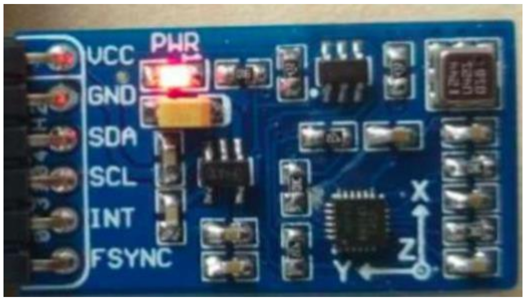


Fig.5. MPU9255 physical object

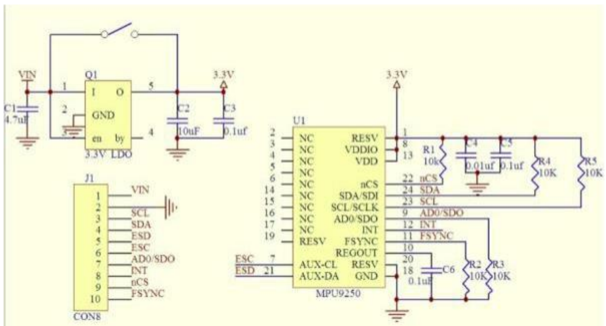


Fig.6. MPU9255 circuit schematic

TABLE II. SPECIFICATION OF MPU9255

Driver IC	MPU9255 (3-axis accelerometer 3-axis gyroscope 3-axisdigital compass)	Built-in 16-Bit AD convertor Gyroscope full-scope range: ±250, ±500, ±1000, ±2000°/sec Accelerometer full-scale range: ±2, ±4, ±8, ±16g Compass full-scale rage: ±4800uT
	BMP280 (Digtal pressure sensor)	Built-in temperature sensor with temperature measurement compensation Pressure measuring range: 300~1100hpa (+9000m ~ -500m relating to sea level) Accuracy: ±0.12hPa(±1m) (700hPa~900hPa,25°C~40°C)
Working voltage	3.3V, 5V	
Supported interface	I2C	
Dimensions	31.2mm*17mm	

The hardware power supply design for each module includes: The system is powered through a 5V/2A (115VAC/DC) power supply. Power is directly supplied to the Raspberry PI. System operation can be controlled with a switch. A 3.3v port built into the Raspberry PI powers the IMU sensor. All vibration sensors are powered using 5V ports.

System hardware specifications are shown in Fig.7 and Table III.

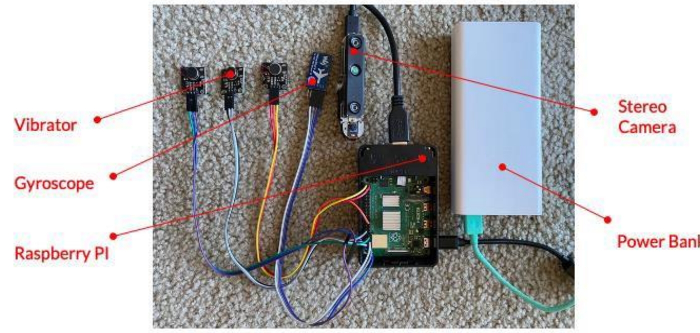


Fig.7. System components

TABLE III. SPECIFICATION OF WHOLE SYSTEM

Features	**Use Environment**	**Detection Range**
	Indoor/Outdoor Enough light	Within 6 feet.
	Image Sensor Technology	
	3µm x 3µm pixel size	
Depth	**Depth Field of View (FOV):**	
	Approx. 90° x 60° x 95°	
Components	**Raspberry Pi 4B**	IMU Sensor
	Intel RealSense D430	3 Vibration Sensors
Physical	**Length x Depth x Height**	**Power**
	To be further determined	5V, 2A

C. Depth Data Quality Evaluation

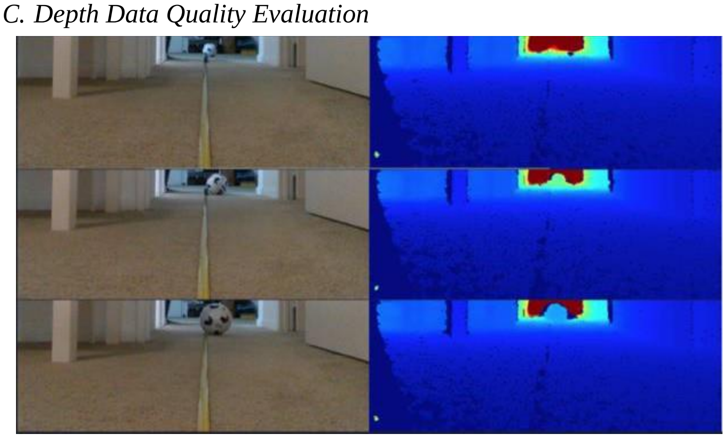


Fig.8. Quality evaluation (different distance)

We evaluated the quality of the depth data we obtained from our system. In our evaluation, we measured the distance from a standard soccer ball to our camera at different distances within 5 meters. These distances formed the ground truth. At the same time, we obtained the actual distances of the depths measured by our system. The difference between the measured and ground truth values are shown in Fig.9. Within a distance of 5 meters, the depth measurement is accurate to within 0.35 meters.

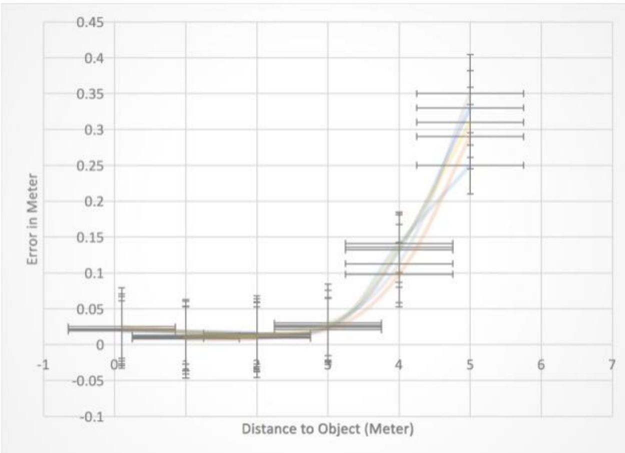


Fig.9. Depth data error

IV. METHOD

A. Depth data processing

In this section, we introduce how to convert the depth data captured from the RealSense depth sensor to the electrical signals to drive the vibrators. The algorithm is shown in Fig.10. There are three stages: the pre-processing stage, the object-detection stage, and the signal clustering and smoothing stage. The clustered signals drive the vibrators.

(1) Pre-Processing Stage: In this stage, the detection range is split into several sections. The camera detection area is split into three consecutive areas with different radii of 0-3m, 3-6m, and beyond 6m. We focus on the nearest area which is the detection area. The observation area is detected with lower priority. In Fig.11, the blue area is the detection area, and the red and yellow areas are observation areas. In the pre-processing stage, we correlated the color space of the detection area to make obstacles more identifiable and the background removable.

(2) Object Detection Stage: After the first pre-processing stage, the depth map includes features with sharpened edges and smooth color surfaces. We extract feature formations using the Canny algorithm[9]. As Fig.12 shows, the object contour can be extracted from the image. After identifying the object, we calculate the image and object bounding box area ratio. This value is used as input to evaluate weight and clustering.

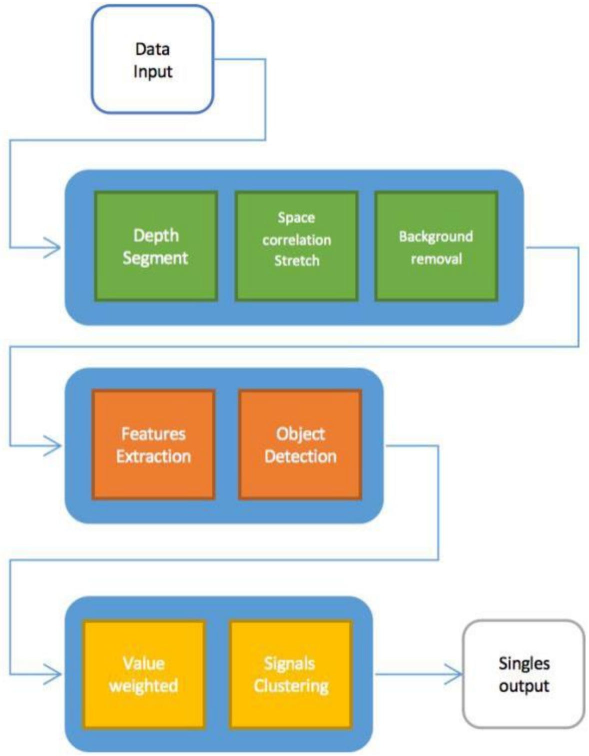


Fig.10. Data processing stage pipeline, the upper portion is the preprocessing stage, the middle portion is the object-detection stage, the lower portion is the signal clustering stage

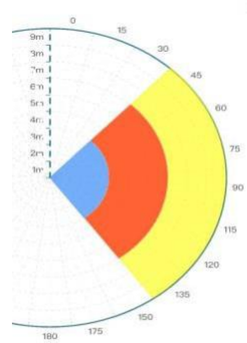


Fig.11. The detecting range is split into areas. The detection area is shown in blue

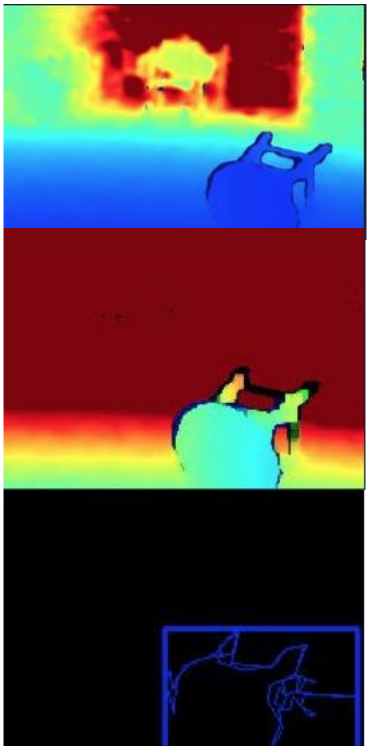


Fig.12. Object detection using correlated stretch color space

(3) Signal Clustering Stage: Using the area ratio and bounding box of the detected object, weights are used to determine whether to turn the vibrators on or off (Fig.13).

Fig.13. Vibrations signals (0 means no vibration, and 1 means vibration)

B.Experiment and Evaluation

We tested static objects 200 times. The detection result are shown in Fig.14. Fig.15 shows that most failures are located at the edges of the depth image. We found that the device had difficulty generating a depth image at the edges. To address this issue, five pixels are cut off from both sides of the vertical edges.

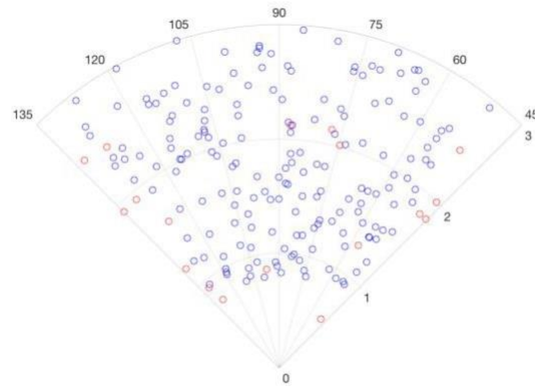


Fig.14. Object detection distribution. The blue circle are successes. Red circles are failures

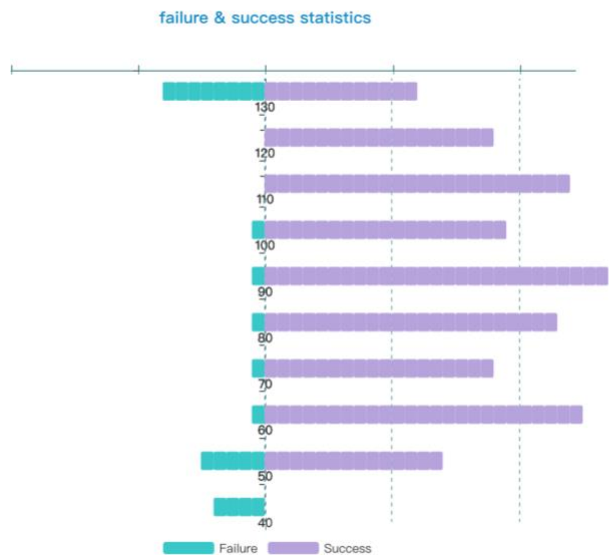


Fig.15. Statistics of the detection experiments. The vertical axis is angular measurements from 45-135. The horizontal axis is the number of successes or failures

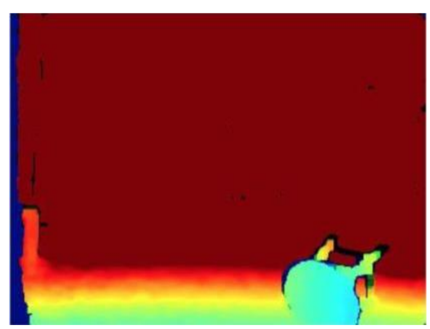


Fig.16. Vertical dark part of the image has incorrect depth values

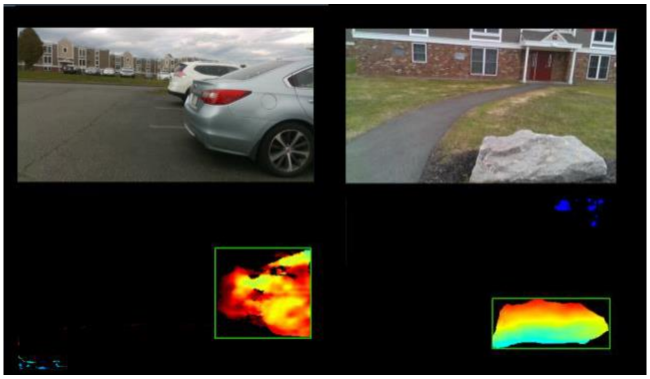


Fig.17. Outdoor scenarios to test the algorithm. The bounding box shows the detected obstacles

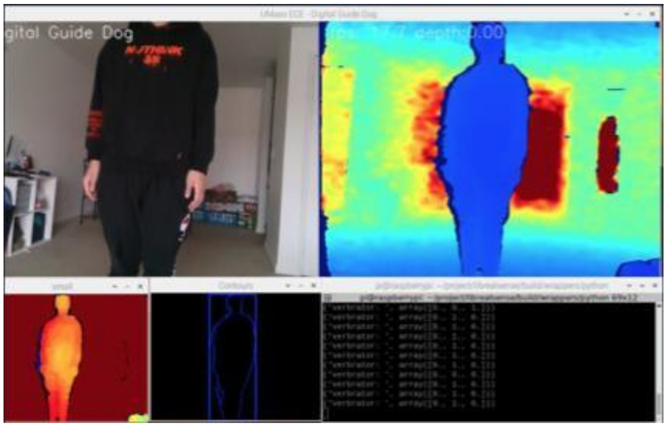


Fig.18 User interface, the upper-left is a RGB image, the upper-right is the depth image, the lower-left is the correlated stretch depth image, the lower-mid is the object detection result, the lower-right is the final signal output

V. ALGORITHM

The basic 3D reconstruction algorithm is described below. First, we calibrate the stereo cameras to find the intrinsic and extrinsic parameters in the location and orientation of two cameras. Calibration is carried out by acquiring and processing more than ten stereo pairs of checkerboard patterns[10]. After calibration, the next step is rectification, which removes lens distortions and turns the stereo pair into a regular standard form. Stereo correspondence finds homologous pixels in the stereo image pair. Triangulation shifts and matches the pixels to generate the depth of each pixel, and get a disparity map[11].

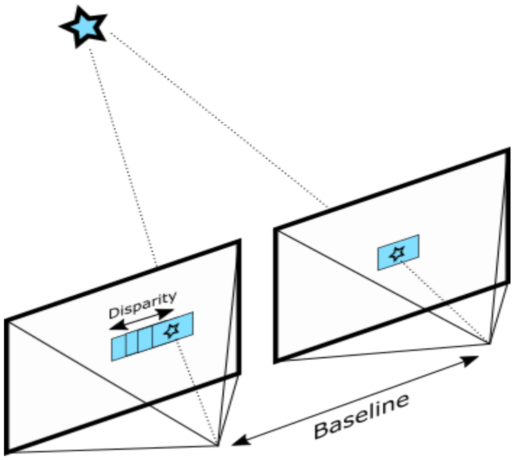


Fig.19. Disparity map (Squares represent cameras and big stars represent object points)

The pseudocode to generate the disparity map which measures the different offset in pixels of two images, pseudo-code is as below[12]:

```
Create a "minSSD" array equal to the size of the image, with large initial values.
Create a "disparity" array equal to the size of the image.
for k = 0 to MAX_SHIFT do
  Step 1: Shift right image to the right by k pixels
  Step 2: Perform Sum of Squared Differences (SSD) between left image and shifted right image
  Step 3: Update the minSSD and disparity array.
      for each pixel coordinate (i,j) do
        if ssd(i,j) < minSSD(i,j) do
          minSSD(i,j) <= ssd(i,j)
          disparity(i,j) <= k
        end
      end
end
```

VI. PROTOTYPE

We made the project website that collected the project idea, detail design block diagram, PowerPoint file for presentation, and demo video. The website screenshot is shown below, visit the project website by the link:

http://www.ecs.umass.edu/ece688/f19-s20/digital-guide-dog/

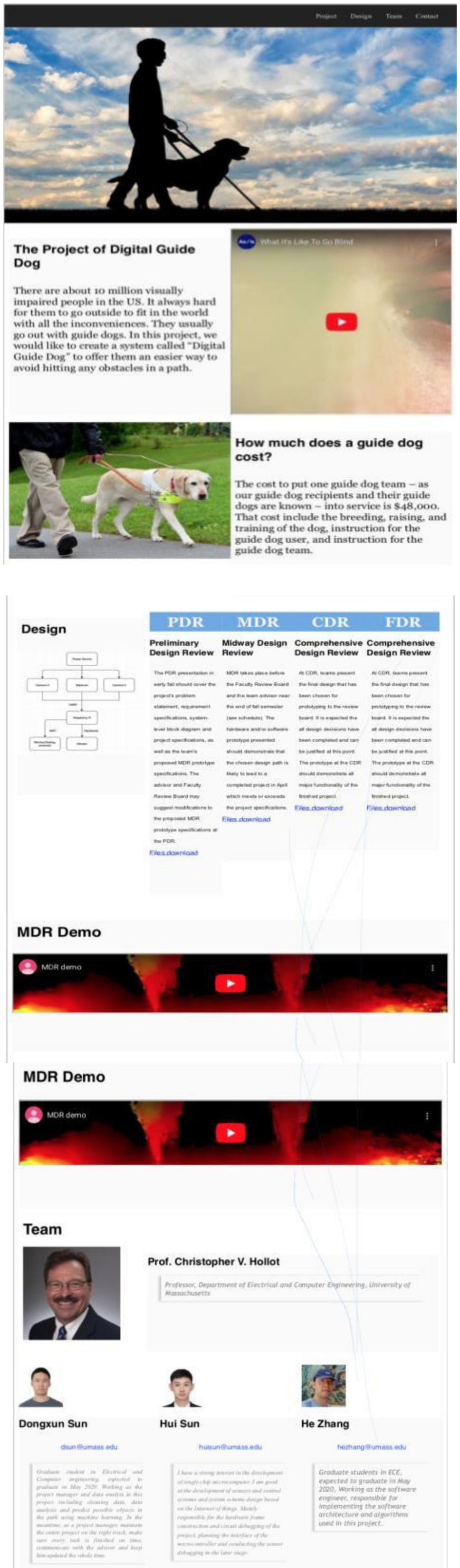


Fig.20. Website page

VII. PLANNED SYSTEM

Our plan was to place a camera in front of the user with a power module and a micro-controller on opposite sides of the body. We expected our design to look like the mockup shown in Fig.21. We completed software debugging of each module and the hardware construction of the product. For the algorithm implementation, we used space correlation and feature detection, to obtain the object depth map, and obtained the image map of the object through background removal. An edge pixel smoothing algorithm was used to identify the camera distance to an object.

The whole system runs and maintain a standby mode. When the system is operating, the Raspberry Pi indicator remains on, and the camera captures the obstacle information in front of it in real-time and feeds it back to the Raspberry PI for processing. Three vibration sensors collect processed feedback signals. The signal outputs of the left, middle, and right vibration sensors are different—the closer the obstacle, the stronger the shock. The IMU sensor corrects the position of the camera in real-time. When the device position deviates in vertical and horizontal position, a feedback signal makes the Raspberry Pi stop collecting camera information, and the three vibration sensors are placed in the lowest state which indicates that the user needs to correct the position of the device. When the device position is in the vertical-horizontal position, the device can feedback the obstacle information in front of it.

According to our MDR plan, machine learning techniques should be used to identify objects in front of a user. Our plan was to extract colored pixels related to people, cars, and dogs, clean up the data from the data set, and train the Raspberry Pi using machine learning models to perform image recognition.

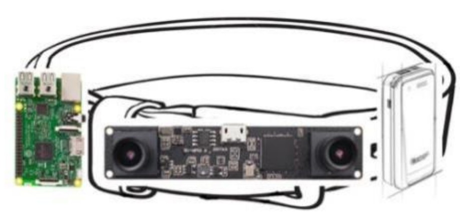


Fig.21. The system we expected to implement

VIII. Results

In the first semester, we reached our goal of generating real-time depth video. We accomplished this goal with the two cameras hardwired together. Data from the two cameras flowed into the Raspberry Pi, which was used to obtain depth video.

For the second semester, we realized the overall function of the system, including obstacle detection, system angle detection, and real-time signal feedback. For further work, detection algorithms could be further improved to allow for an improved recognition rate.

IX. Conclusion

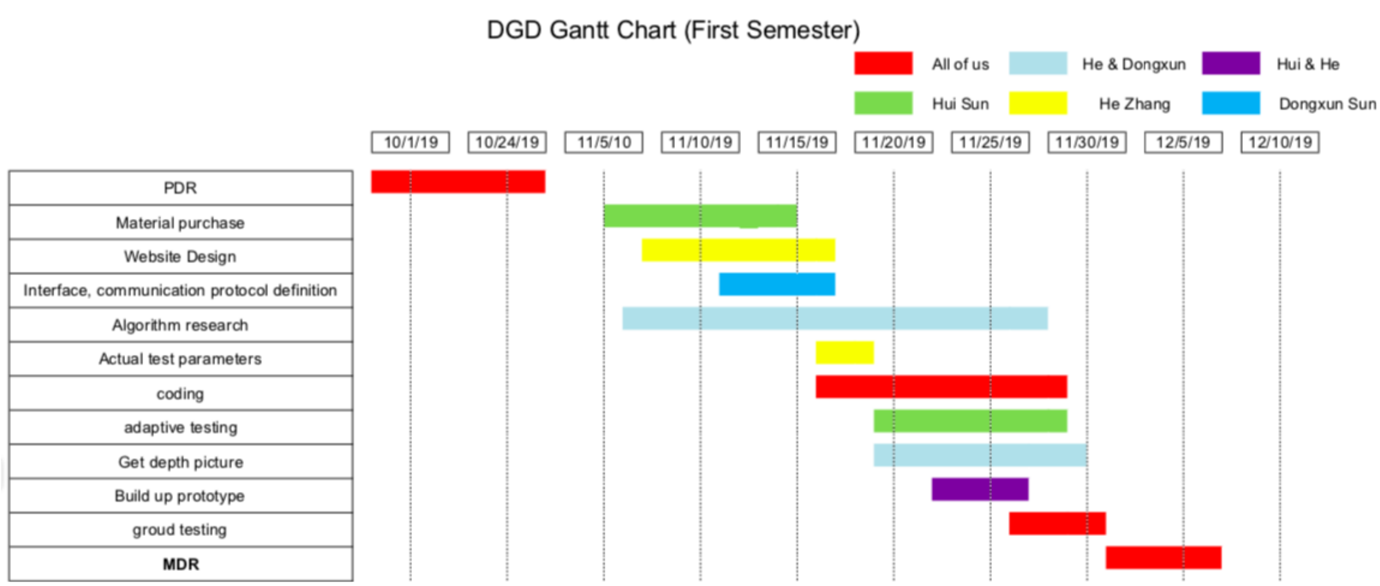
In our project, we created a video stream to assist vision impaired persons. Using an Intel camera and image processing area technology, the system determines the distance between the user and an obstacle and alerts the user of the obstacle. We carried out the software and hardware design of the system and image processing algorithm research to achieve an electronic guide dog prototype system.

X. Project Management

In terms of project management, our team met every Tuesday to give a weekly update to our advisor, Dr. Christopher V. Hollot. The update usually included progress in software and hardware, and the plan for the coming week. Before PDR and MDR, we did review presentations with Dr. Hollot, and he provided constructive advice.

TABLE IV. Division of Labor

Dongxun Sun	Project Manager	Contact with the advisor and keep the project on the right track, gathered data from the system for future work, currently have about 1 GB of clean data.
He Zhang	Software engineer	Design and implement all the core algorithms, Website design
Hui Sun	Hardware engineer	Development board and sensor debugging and implementation, Poster design



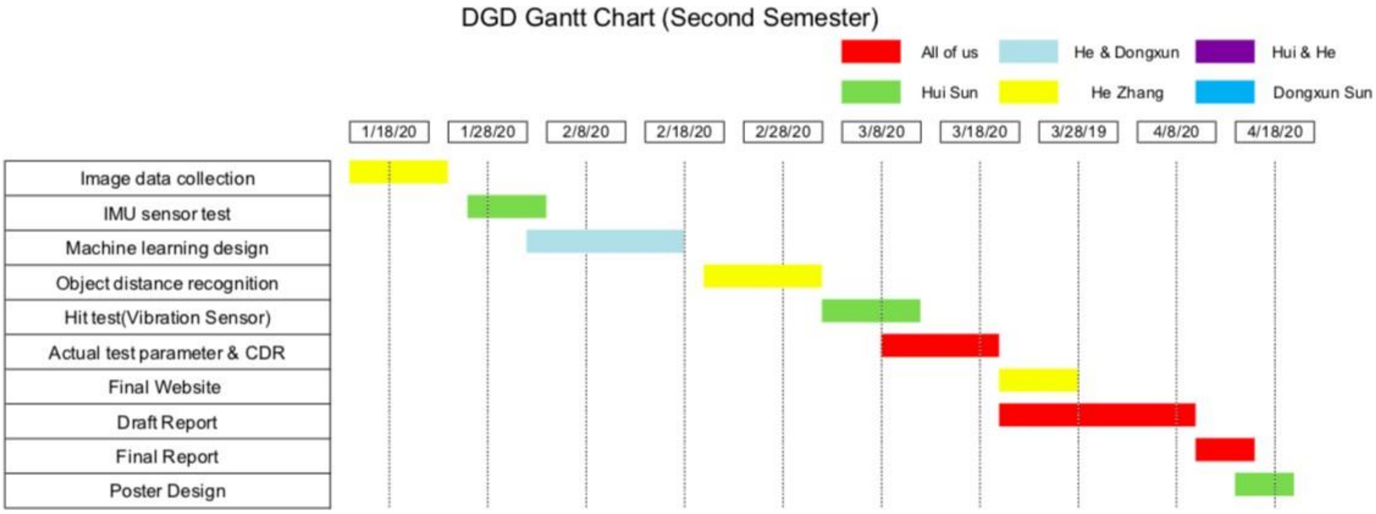


Fig.22. Gantt chart

XI. APPENDIX

Fig.23. DGD application user interface

REFERENCES

[1] Morita, C. (2019). *How Much Does A Guide Dog Cost?*. [online] Puppy In Training. Available at: https://puppyintraining.com/how- much-does-a-guide- dog-cost/ [Accessed 16 Dec. 2019].

[2] Servicedogcentral.org. (2019). *How long does it take to train a guide dog? | Service Dog Central*. [online] Available at: https://servicedogcentral.org/content/node/348 [Accessed 16 Dec. 2019].

[3] "Puppy Training Tips." *Puppy In Training*, puppyintraining.com/.

[4] "About." *WeWALK Smart Cane*, wewalk.io/en/about/.

[5] "Wear, See and Enjoy." Acesight, www.acesight.com/.

[6] Guo, T. (2019). [online] Blog.csdn.net. Available at: https://blog.csdn.net/zhufeng88/article/details/75452271 [Accessed 16 Dec. 2019].

[7] Intel.(2019). [online] Available at: http://www.imtelrealsense.com/depth-camera-d435/ [Accessed 16 Dec. 2019].

[8] "MPU-9255 Product Specification." Https://Datasheet.octopart.com/MPU-9255-InvenSense-Datasheet-32037105.Pdf, InvenSense Inc. , 14 Sept. 2014.

[9] Sahir, Sofiane. "Canny Edge Detection Step by Step in Python-Computer Vision." Medium, Towards Data Science, 27 Jan. 2019, towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123.

[10] Mattoccia, S. (2019). [online] Vision.deis.unibo.it. Available at: http://vision.deis.unibo.it/~smatt/Seminars/StereoVision.pdf [Accessed 16 Dec. 2019].

[11] Intel® RealSenseTM Depth and Tracking Cameras. (2019). *The basics of stereo depth vision – Intel® RealSenseTM Depth and Tracking Cameras*. [online] Available at: https://www.intelrealsense.com/stereo-depth-vision-basics/?cid=em-elq-38282&elq_cid=3569199 [Accessed 16 Dec. 2019].

[12] Li, T. (2019). [online] Blog.csdn.net. Available at: https://blog.csdn.net/real_myth/article/details/51206579 [Accessed 16 Dec. 2019].