Demonstration Paper: Wearable Computing for Image-Based Indoor Navigation of the Visually Impaired

Gladys Garcia
California State University, Northridge
Los Angeles, CA
gladysmae.garcia.999@my.csun.edu

Ani Nahapetian
California State University, Northridge
Los Angeles, CA
ani@csun.edu

ABSTRACT

In this paper, an image-based non-obtrusive indoor navigation system for the visually impaired is presented. The system makes use of image processing algorithms to extract floor regions from images captured from a wearable eye-mounted heads-up display device. A prototype system called VirtualEyes is presented, where floor regions are analyzed to provide the user with voiced guidance for navigation. The floor detection algorithm was tested against over 200 images captured from indoor corridors of various lighting conditions and achieved up to 81.8% accuracy.

Categories and Subject Descriptors

C.3.3 [**Special-Purpose and Application-Based Systems**]: Signal Processing Systems.

General Terms

Algorithms, Design, Experimentation.

Keywords

Mobile computing, wearable computing, assistive technology, image processing, floor detection, computer vision, Google Glass.

1. INTRODUCTION

There are an estimated 285 million people in the world that are visually impaired [1], including 39 million who are blind and 246 million with low vision, i.e. with moderate to severe visual impairment. The visually impaired often require help to navigate unfamiliar environments, including relying on a walking stick or a guide dog, to navigate spaces and avoid obstructions. Researchers have focused on alternatives, with mobile and wearable technology holding the potential to advance research in this area.

In the robotics field, image-based approaches are commonly used for navigational guidance or obstacle detection. Information is extracted from the captured images of the environment with the use of image processing algorithms.

In this work, a floor detection algorithm that was used for the automatic navigation of a mobile robot was adapted to create a mobile indoor navigation system for the visually impaired. A prototype system called VirtualEyes was developed using Google Glass. Google Glass is a wearable device that is capable of running Android applications. It has a camera feature that was utilized to capture images from the surroundings. It is non-obtrusive and is worn in a way that allows the camera to capture an unobstructed view of the user's environment.

The remainder of the paper presents the hardware and software components of the system, along with the technical approach used for navigational guidance using image processing. The algorithms developed for the VirtualEyes system were tested on over 200 different images with the results provided and discussed.

WH '15, October 14-16, 2015, Bethesda, MD, USA
© 2015 ACM. ISBN 978-1-4503-3851-6/15/10…$15.00
DOI: http://dx.doi.org/10.1145/2811780.2811959

2. RELATED WORK

Wearable devices have been widely used for different applications. For example, Najeeb et al. present a wearable system that uses an off-the-shelf EEG device that reads brain signals to select letters, compile words, and create sentences meant for people with paralysis [19]. A wearable system for determining body and arm positioning using ambient light sensors is presented in [22]. A smart watch is used in [20] to recognize arm gestures for hands-free interaction. Altwaijry et al. present a system that uses Google Glass in [21] that can recognize landmarks by capturing an image of the scene and the GPS information if available.

There has also been research in wearable devices for guiding the visually impaired in unfamiliar environments. The underlying technology varies from a modernized version of the walking stick (a.k.a. white cane) to image-based approaches.

Fernandes et al. describe a system that uses RFID tags attached at the end of the white cane [2]. A virtual white cane is presented in [3] by using a laser pointer attached to a smartphone. Both approaches require the use of specialized hardware.

In terms of the image-based approach, different systems made use of a smartphone [4], Microsoft Kinect [5], and custom hardware using two cameras mounted on the user's shoulders [6] as interfaces to gather images of the environment. The use of pre-installed special markers in the environment to identify a safe walking path for the user is also presented in [7].

Such image-based systems commonly adapt the floor detection or obstacle detection implementation in the robotics field. The use of stereo vision is common in this approach as discussed in [6][8]. These systems are able to detect floor regions and obstacles in the environment and calculate the distance of such objects from the user.

The work of Tapu et al. [4] uses monocular vision by utilizing the camera in a smartphone which is a less obtrusive design. Obstacle detection is performed to guide the user when walking in the outdoor environment.

Another common approach in the robotics field is to use image sequences from the video feed as described in [9][10] to track the movement in the scene.

In terms of floor detection, other approaches make use of a single indoor image of the environment to classify floor regions. The implementation presented in [11] makes use of image segmentation to identify floor regions in the image. Authors of [12][13] use horizontal and vertical lines found in the image to detect floor regions.

3. SYSTEM OVERVIEW

In this section, the architectural overview with the hardware and software components of VirtualEyes, a system for the visually impaired navigation guidance, is presented. The system is composed of a paired Google Glass and Android smartphone. These two devices, connected via Bluetooth, work together in gathering image data from the indoor environment using the Google Glass camera, process the data on the smartphone, and provide valuable feedback to the user through the use of the built-in speaker on the Glass.

3.1 Google Glass

The Google Glass, as shown in Figure 1, is a head-mounted, rechargeable battery-operated wearable device, which is capable of running Android applications. This device has features similar to a smartphone ranging from high resolution display, camera, Bluetooth, Wi-Fi, etc. [14]. The camera in the Glass is capable of taking 5 megapixel images. Since the device is worn over the eyes of the user, similar to prescription glasses, the images taken from the camera captures the surroundings in the perspective of the user.

Since this device is still in its early stages, there are a few limitations to its performance. The battery in the device typically lasts one hour of usage, especially with the use of Bluetooth and the camera. Heating of the device could also cause discomfort to the user while the Glass is in operation. Furthermore, there is limited processing power available in the Glass to perform the image processing required in this system. To overcome these limitations, the Bluetooth capability of the Glass was utilized to pair it with an Android smartphone and offload processing that would require substantial power.

3.2 Android Smartphone

Mobile smartphones have been a ubiquitous device that is accessible for most people. The higher processing power and better battery life in these smartphones as compared to Google Glass allows for an ideal mobile and lightweight device for performing powerful operations that might prove difficult to run on the Glass.

The use of Android operating system allows the integration of many open source third party libraries that provides an easy to use framework in performing tasks required by the system such as OpenCV.

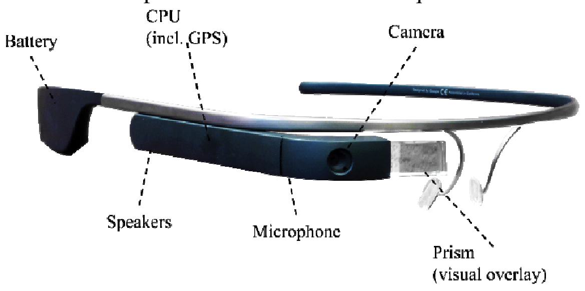

Figure 1. Google Glass is a rechargeable battery-operated wearable device with built-in features such as camera, speakers, Bluetooth, Wi-Fi, etc.

3.3 OpenCV

Open Source Computer Vision (OpenCV) is a widely used library of image processing algorithms. The library supports different operating systems including Android and has interfaces for a variety of programming languages such as C, C++, and Java [15]. The built-in functions in the OpenCV library were used in this system for most of the image processing tasks.

3.4 Mobile Applications

There are two different applications developed for the system which are installed in the respective devices. Figure 2 shows an overview of the functionalities and communication between the applications.

An android application (client app) is installed on the Google Glass that will start up the Glass camera and send captured image frames to the paired Android smartphone. This application also receives text information coming from the Android smartphone and converts this into voice guidance using the Text-To-Speech framework of the Android operating system.

Another android application (server app) is installed on the Android smartphone that is paired with the Google Glass. This application performs various image processing algorithms using OpenCV in order to extract information from the received images. The features from the image are extracted which are then evaluated to analyze the floor region. Once the image analysis has been completed, a feedback is sent to the Google Glass through the Bluetooth connection that has been established.

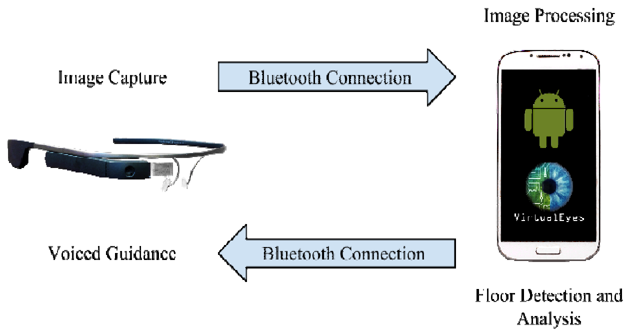

Figure 2. Images captured by the Google Glass are sent over to the Android smartphone via Bluetooth for Image Processing and Floor Detection and Analysis. The results of the analysis are sent back to the Google Glass for voiced guidance.

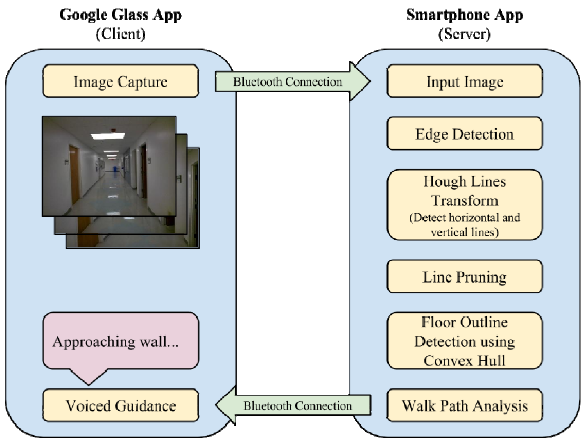

Figure 3. The Google Glass app provides the input and output information from and to the user. The Smartphone app performs Floor Extraction and Analysis by performing a series of image processing algorithms.

4. APPROACH

In this section, the implementation of VirtualEyes is discussed which includes the communication between the paired devices, floor extraction and analysis, and user feedback as seen in Figure 3.

4.1 Device Communication

The paired devices transmit data to each other over a Bluetooth connection. The Glass application continuously sends image frames to the Android smartphone for processing.

A 320x240 RGBA image frame captured by the Glass is about 300 kilobytes. To increase the frame rate of the application, this is compressed to a *jpeg* format using the built-in compression function in the OpenCV library. This reduces the size of the image to less than 100 kilobytes.

4.2 Floor Detection

The images captured by the Glass typically contain the walls, floor, ceiling, and other objects within the frame. The floor region is surrounded by walls in all sides. By detecting the wall-floor boundaries from the image, the floor region can be detected within the image as shown in Figure 4 and later analyzed to provide feedback to the user.

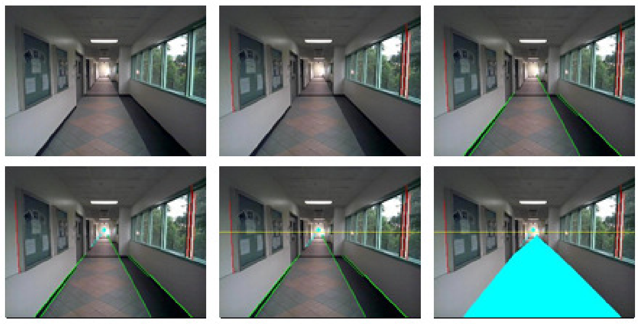

Figure 4. Results of each image processing step. From top-bottom, left-right: (a) input image, (b) vertical lines in red, (c) horizontal lines in green, (d) cyan dots as the intersections of every pair of horizontal lines (vanishing point), (e) yellow line as the average y-axis value of all vanishing points, (f) convex hull of detected horizontal lines.

The floor detection approach discussed in [12] was adapted in the implementation of this system. This approach is capable of detecting floor regions from a single indoor corridor image.

The first step is to apply the Canny Edge detection [16] algorithm in the image to identify the edges in the image. An edge is a region in the image where there is a sudden change in the pixel intensity. This outputs a black and white image where the white pixels are the identified edges in the image.

From the black and white edge image, we try to find vertical and horizontal lines in the image using Hough Line Transform [17]. Vertical lines are defined as lines that are within 10 degrees from the vertical direction. Horizontal lines, on the other hand, can go from 40-70 degrees from the horizontal direction. Due to the noisy conditions in the scene (i.e. posters on the walls, shadows from lighting, etc.), there could be vertical and horizontal lines that are detected which are not part of the wall-floor boundary. In order to minimize the incorrect line extraction, lines that match any of the below conditions are removed:

- Lines that are shorter than 30 pixels
- Vertical lines that exist entirely on the upper half of the image
- Horizontal lines that appear above the vanishing point

All the remaining vertical and horizontal lines are assumed to be part of the wall-floor boundaries. The convex hull for all the endpoints of the lines is computed which gives the rough estimate shape of the floor region in the image. The convex hull implementation in OpenCV [18] was used for the prototype.

4.3 Walk Path Analysis

The output from the previous floor detection step is a polygon indicating the detected floor. In the walk path analysis step, the outline of the floor is used to determine how much floor space is ahead of the user.

When walking along a corridor, the perspective of the user shows the walls on each side of the corridor, floor, and ceiling as seen in Figure 4 (a). The vanishing point of the perspective line in the image is roughly located at the center of the image depending on the height and viewing angle of the user. The floor region in such viewing angle is roughly shaped like a trapezoid where the base is wider than the top. The height of the floor region would indicate the proximity of the user to the end of the corridor. The height decreases as the user approaches the end of the corridor as seen in Figure 5. By using the height of the estimated floor outline, the system can make an analysis on whether the user is safe to proceed walking or should stop to avoid hitting a wall.

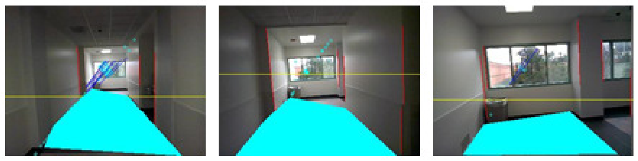

Figure 5. Consecutive image frames showing the decreasing height of the detected floor region as the user approaches the end of the corridor.

The floor detection phase returns a list of points that forms the outline of the detected floor region. The height of this floor region is computed by taking the difference between the lowest and highest point in the outline. By testing the system using 320x240 pixel images from multiple environments, it was found that a good threshold for the floor outline height is 30 pixels. An image where the height of the floor region is less than the threshold value indicates that the user is standing close to a wall. On the other hand, a floor region height that is greater than the threshold indicates that the user has enough walking space from the wall.

4.4 User Feedback

The Google Glass has a built-in speaker that uses bone conduction technology. The speaker is utilized to give guidance to the user while navigating in an indoor environment.

The walk path analysis phase determines whether it is safe for the user to continue walking forward or should the user stop. This information is delivered to the user using the built-in speaker. By the using the Text-To-Speech library in Android, the user can hear alerts from the system. VirtualEyes will tell the user to "Stop" or "Walk" every few seconds.

5. RESULTS

The floor detection and walk path analysis algorithms presented in the previous section are tested using test images taken from various locations in California State University, Northridge (CSUN) campus. The client application was installed in a Google Glass Explorer Edition version 2 with firmware version XE22. This device runs on a Texas Instruments OMAP 4430 SoC 1.2GHz Dual (ARMv7) processor with 2GB of RAM. The server application is installed in a Samsung Galaxy S4 running Android version 4.4. This smartphone has a Qualcomm MDM9215 + APQ8064T 1.9GHz Quad-core with 2GB of RAM.

Different datasets were collected from various corridors in the CSUN campus specifically in Jacaranda Building, Bayramian Hall, and Sierra Hall. The images were captured while walking in a constant pace along the corridor towards a wall. For each of the 7 datasets, the first image was taken with a distance from the user to the wall that ranges from 30 to 60 feet. As the user approaches the wall in a constant pace, this distance becomes smaller as seen in Figure 6. The last few images in the dataset were about 2 to 5 feet from the wall where the floor is no longer visible which is shown in Figure 7.

5.1 Floor Detection Results

The test data were taken from different locations with a variety of color and texture of the floor and walls. Dataset 1 contains images

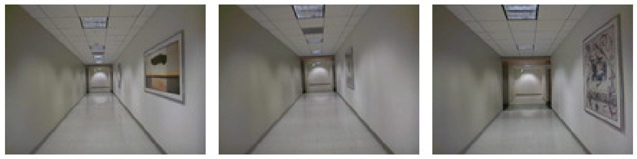

Figure 6. Sample images with enough distance from the user to the wall that shows the floor region.

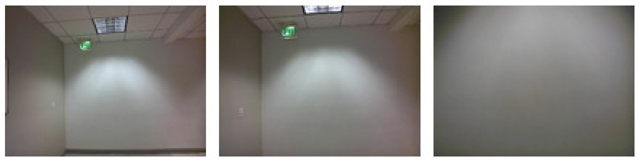

Figure 7. Sample images captured when the user is standing close to the wall.

of corridors with good floor and wall color contrast. The images have varying lighting conditions due to the windows that are present on the right side of the corridor. Dataset 4 contains images where the floor and walls have different colors. These images contain reflective floor surfaces as opposed to dataset 1 and have bulletin boards on the wall. The rest of the datasets are composed of images where the floor and walls have a poor contrast. However, there is a darker colored baseboard that separates the wall and floor in images in dataset 3, 5, 6, and 7. Sample results from different datasets are shown in Figure 8.

The floor detection algorithm relies heavily on edges found in the images. If there is a good contrast between the floor and the wall pixels in the image, the system will more accurately detect the floor region in the image. Datasets 3 and 7 have the highest accuracy out of all the datasets that were tested with about 81.8% and 77.1% respectively. Although the floor and walls have a similar color, there is a darker colored baseboard on the wall that clearly separates floor pixels from the wall pixels. The algorithm, however, failed in situations where the user is turning into another corridor.

Images from datasets 1 and 4 have very distinct floor and wall boundaries but some images were affected by other conditions, shown in Figure 11. Images from dataset 1 contain a window on the right side of the image. Objects outside the window contain edges that were also detected by the edge detection algorithm which negatively affected the floor detection. For dataset 4, bulletin boards that are attached on the wall caused stray edges to be detected which cause the floor detection to incorrectly identify the floor region.

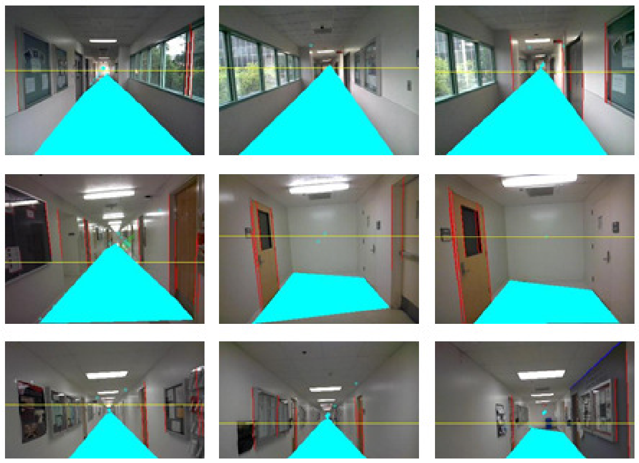

Figure 8. The floor detection algorithm is able to estimate the floor region from captured images of corridors with different color, texture, and lighting. The images above shows the results of the floor detection phase from different datasets.

Table 1. Table shows the number of correctly and incorrectly identified floor regions in the different sets of test images.

Data Set	Total Images	% of Correctly Identified	% of Incorrectly Identified
1	48	75%	25%
2	20	10%	90%
3	22	81.82%	18.18%
4	27	66.67%	33.33%
5	48	77.08%	22.92
6	24	70.83%	29.17%
7	94	65.96%	34.04%

Of all the sets of data for testing, dataset 2 has the lowest accuracy rate with just 10% as seen in Table 1. These are images of corridors where the floor and wall color are very similar as shown in Figure 11. The edge detection step failed to detect the wall-floor boundary which caused the floor detection to fail. In this kind of input images, it might help to have an image pre-processing step that would enhance the edges in the image without making the image noisy.

Furthermore, the floor detection does not perform well on images captured when turning in corridors as shown in Figure 11. The algorithm relies on finding the wall-floor boundaries on both sides of the floor. When turning in corridors, there is only one side where the wall is visible.

5.2 Walk Path Analysis

The floor outline result of the floor detection phase is used as input in the walk path analysis. To analyze the results of the walk path analysis phase, the height of the floor outline was compared against the actual distance of the user from the wall when the image was captured. Since the walk path analysis phase is highly dependent on the accuracy of the results from the floor detection phase, images that did not have successful floor detection results were removed from the dataset for this testing. Furthermore, since dataset 2 had a very low overall accuracy in floor detection, it was not included for this testing.

The results of the comparison are shown in Figure 12. The vertical axis indicates the height of the detected floor region in pixels. The horizontal axis indicates the distance of the user from the wall when the image was captured.

It can be seen from Figure 12 that the overall trend of the graphs indicates a decreasing height of the floor outline. This reflects the decreasing distance as the user walk closer to the wall. At about 10 feet or less, the height in pixels of the floor outline begins to decrease sharply. And at about 5 feet is where the floor outline height drops to 0. This indicates that the floor detection phase no longer detects any floor in the image which is accurate as seen in sample images in Figure 7. This shows that the floor region height can be used as a parameter in estimating the proximity of the user to the wall.

Figure 12 also shows the floor outline height does not linearly decrease along with the decreasing actual distance of the user from the wall. There are random spikes in the graph which indicates that the floor region detected increased in height. This is mainly caused by the user movement while walking. The use of a

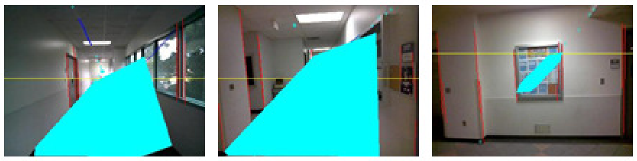

Figure 11. Other elements present in the image that could confuse the edge detection algorithm such as windows and bulletin boards cause the floor detection algorithm to fail.

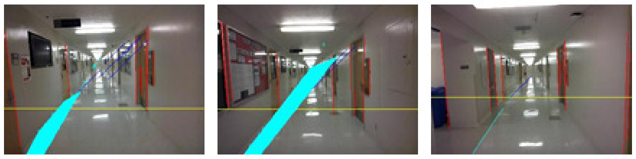

Figure 11. The floor detection fails on images where the floor and wall pixels have low contrast.

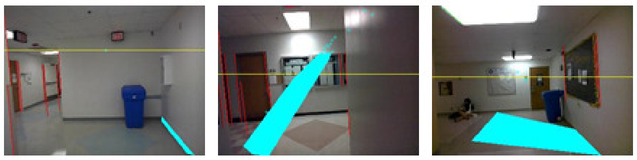

Figure 11. The floor detection algorithm fails on correctly estimating the floor outline when turning in corridors.

head mounted camera is sensitive to changes in orientation. There will be a slight change to the height of the camera as the user makes a step forward. Furthermore, camera orientation will also be affected by movements of the head of the user.

If the floor detection phase returns an inaccurate result, this affects the result of the walk path analysis phase. To overcome this, VirtualEyes was designed to keep a running average of the floor outline height as the user walks forward. The average height of the resulting floor outline of the past 10 images is computed. This value is used in determining the appropriate feedback sent to the user. With this approach, if only one image in a continuous image sequence fails in the floor detection step, this will not greatly affect the results of the walk path analysis phase.

6. CONCLUSION

This paper has shown the effectiveness of using mobile devices

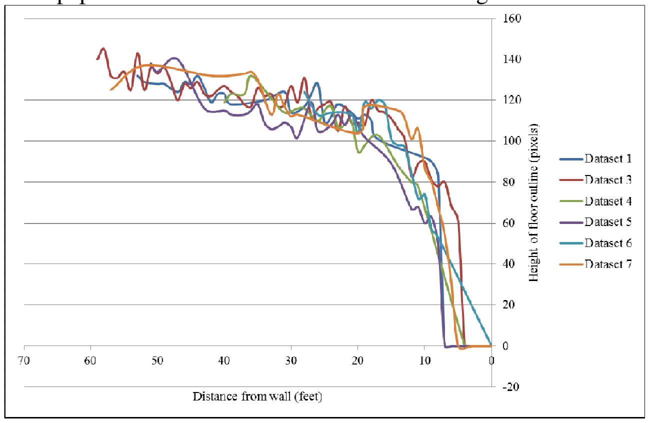

Figure 12. Comparison of the height of the floor outline and the actual distance from the wall for different datasets.

for a navigational guidance system for the visually impaired. The approach can effectively alert the user when the floor outline height reaches a low value which indicates that there is no more walking space ahead of the user.

The system uses floor detection in user indoor guidance, instead of the previously explored obstacle detection. The effectiveness of this approach was demonstrated with the VirtualEyes prototype. The system achieved up to 81.8% accurate detection of the floor on a set of over 200 distinct images. The floor detection algorithm implemented in the system works well in corridors where the wall on both sides are visible and have a distinctive color contrast between the floor and the walls. Detection of floors on images with minimal color contrast could be improved with the use of some image pre-processing algorithms.

7. REFERENCES

[1] WHO Visual impairment and blindness http://www.who.int/mediacentre/factsheets/fs282/en/ Accessed on April 7, 2015.

[2] Faria, J.; Lopes, S.; Fernandes, H.; Martins, P.; Barroso, J., "Electronic white cane for blind people navigation assistance," World Automation Congress (WAC), 2010 , vol., no., pp.1,7, 19-23 Sept. 2010.

[3] Pablo Vera, Daniel Zenteno, and Joaquín Salas. 2014. A smartphone-based virtual white cane. Pattern Anal. Appl. 17, 3 (August 2014), 623-632.

[4] Tapu, R.; Mocanu, B.; Zaharia, T., "A computer vision system that ensure the autonomous navigation of blind people," E-Health and Bioengineering Conference (EHB), 2013 , vol., no., pp.1,4, 21-23 Nov. 2013.

[5] Zenteno Jiménez, Enrique Daniel, and Joaquín Salas Rodríguez. Electronic Travel Aids With Personalized Haptic Feedback for Visually Impaired People. Instituto Politécnico Nacional (IPN), 2014.

[6] Shang Wenqin; Jiang Wei; Chu Jian, "A machine vision based navigation system for the blind," Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on , vol.3, no., pp.81,85, 10-12 June 2011.

[7] Fernandes, H.; Costa, P.; Filipe, V.; Hadjileontiadis, L.; Barroso, J., "Stereo vision in blind navigation assistance," World Automation Congress (WAC), 2010 , vol., no., pp.1,6, 19-23 Sept. 2010.

[8] Okada, K.; Inaba, M.; Inoue, H., "Walking navigation system of humanoid robot using stereo vision based floor recognition and path planning with multi-layered body image," Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on , vol.3, no., pp.2155,2160 vol.3, 27-31 Oct. 2003.

[9] Young-geun Kim; Hakil Kim, "Layered ground floor detection for vision-based mobile robot navigation," Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on , vol.1, no., pp.13,18 Vol.1, 26 April-1 May 2004

[10] Pears, N.; Bojian Liang, "Ground plane segmentation for mobile robot visual navigation," Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on , vol.3, no., pp.1513,1518 vol.3, 2001

[11] Changhwan Chun; Dongjin Park; Wonjun Kim; Changick Kim, "Floor detection based depth estimation from a single indoor scene," Image Processing (ICIP), 2013 20th IEEE International Conference on , vol., no., pp.3358,3362, 15-18 Sept. 2013

[12] Yinxiao Li; Birchfield, S.T., "Image-based segmentation of indoor corridor floors for a mobile robot," Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on , vol., no., pp.837,843, 18-22 Oct. 2010

[13] Barcelo, G.C.; Panahandeh, G.; Jansson, M., "Image-based floor segmentation in visual inertial navigation," Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International , vol., no., pp.1402,1407, 6-9 May 2013.

[14] Google Glass - Tech Specs. https://support.google.com/glass/answer/3064128?hl=en Accessed on February 24, 2015.

[15] OpenCV http://opencv.org/ Accessed on March 8, 2015

[16] Canny Edge Detector - OpenCV 2.4.9.0 Documentation http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html Accessed on March 1, 2015.

[17] Hough Line Transform - OpenCV 2.4.9.0 Documentation http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html Accessed on March 1, 2015.

[18] Convex Hull – OpenCV 2.4.11.0 Documentation http://docs.opencv.org/doc/tutorials/imgproc/shapedescriptors/hull/hull.html Accessed on April 20, 2015.

[19] Dina Najeeb, Antonio Grass, Gladys Garcia, Ryan Debbiny, and Ani Nahapetian. 2014. MindLogger: a brain-computer interface for word building using brainwaves. In Proceedings of the 1st Workshop on Mobile Medical Applications (MMA '14). ACM, New York, NY, USA, 6-11.

[20] Costante, G.; Porzi, L.; Lanz, O.; Valigi, P.; Ricci, E., "Personalizing a smartwatch-based gesture interface with transfer learning," Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European , vol., no., pp.2530,2534, 1-5 Sept. 2014.

[21] Altwaijry, H.; Moghimi, M.; Belongie, S., "Recognizing locations with Google Glass: A case study," Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on , vol., no., pp.167,174, 24-26 March 2014.

[22] Arsen Papisyan, Ani Nahapetian. LightVest: A Wearable Body Position Monitor Using Ambient and Infrared Light. ACM International Conference on Body Area Networks (BodyNets), September 2014.