Visually Impaired Indoor Navigation using YOLO Based Object Recognition, Monocular Depth Estimation and Binaural Sounds

Sukesh Davanthapuram, Xinrui Yu and Jafar Saniie
Embedded Computing and Signal Processing Research Laboratory (http://ecasp.ece.iit.edu/)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, IL, U.S.A.

***Abstract*— This paper presents the development of a real-time spatial audio generating software to assist visually impaired people in indoor navigation using computer vision techniques and binaural sound generations. Our computer vision techniques utilize YOLO (You Only Look Once) based algorithm to detect objects, monocular depth estimation techniques to derive the depth map from a single captured image, and linear interpolation to obtain the azimuth and elevation angles of the detected objects. Based on the obtained results, binaural sounds are generated by HRTF (Head Related Transfer Function), where the intensity of the generated spatial audio is varied according to the distance of the detected object. Our test results show the real-time generated binaural sounds were able to accurately specify the position of the object in 2D space to avoid collisions and to provide surrounding information for navigating visually impaired people.**

Keywords—Visually Impaired, Real-time Object Detection, YOLO, Monocular Depth Estimation, HRTF, Spatial Audio Navigation

I. Introduction

As of 2010, World Health Organization (WHO) estimated that the number of visually impaired people globally is 285 million, of which 39 million are blind [1]. These people face mobility difficulties that impact their quality of life, especially inside buildings that they are unfamiliar with [2]. Inside such buildings, numerous objects of different categories exist and present navigational hazards to the visually impaired. While these obstacles are avoided by sighted people subconsciously, it is very difficult for the visually impaired to avoid them without external help. To solve this problem, object recognition and localization system are needed, preferably portable. Numerous researches exist in this field with a vast range of sensors [2]-[6]. Mostly, the difficulty of such a system lies in real-time object recognition and acquiring depth information. The latter often requires using an RGB-D camera, which can be expensive and not ideal for a portable system [2][5][6].

Thanks to the rapid developments of artificial intelligence and machine learning, more techniques emerged to help visually impaired people navigate in indoor environments. With the help of YOLO object detection, the surrounding objects can be easily identified [7]. To solve the problem of depth estimation using a single RGB camera, a monocular depth estimation algorithm has been developed that generates a depth image with one single input image [8]. The image is processed, and the data of the surrounding objects is converted into spatial audio. This spatial audio signal is given to the visually impaired person. One of the main advantages is that there is no special training required to use this kind of navigation system. The spatial audio will give a very clear picture of the surroundings and helps the visually impaired person to obtain better situational awareness.

II. Indoor Navigation Method

A. System Design Overview

Our system design can be divided into 4 system components as shown in Fig. 1. The first component is responsible for detecting all objects in the captured frame using the webcam, and generate a bounding box for the detected objects. The second component outputs a depth map by the monocular depth estimation module using the captured frame. Once the depth map is generated, the depth of each pixel can be calculated. The third component generates the required azimuth and elevation index information using the bounding box data. All this data is sent to the fourth component, which utilizes the HRTF (Head Related Transfer Function) spatial audio generation module for the spatial audio.

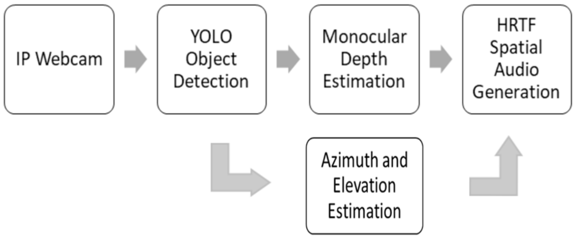

IP Webcam
YOLO Object Detection
Monocular Depth Estimation
HRTF Spatial Audio Generation
Azimuth and Elevation Estimation

Figure 1. Overview of the System Design

B. IP Webcam Application

We utilize an IP Webcam application on a smartphone which could stream and send captured image using the smartphone's wireless communication module. With this application, it enables our software application to access the spatial audio generation software over the Internet connection through the smartphone. As shown in Fig. 2, an IPv4 or an IPv6 address can be used to access the feed of the camera. The captured image from this application is used as an input to the YOLO Object Detection module.

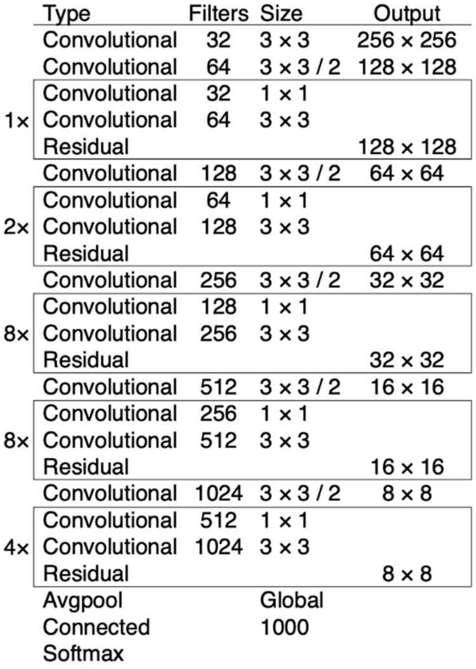
IPv4: http://192.168.43.113:8080
IPv4: https://192.168.43.113:8080
IPv6: http://[2409:4070:2d8c:cc85::28]:8080

Figure 2. User Interface of IP Webcam Application

C. *YOLO Object Detection*

YOLO (You Only Look Once) is an algorithm that is capable of detecting objects using Convolutional Neural Networks (CNN). There are two tasks involved in the object detection mechanism. The first task is to determine the object's location and the second task is to classify those objects. Region-Based Convolutional Neural Network (R-CNN) or its variations could be also used to detect objects, but they are slow and difficult to optimize. A single neural network is applied to the full image as the image is divided into regions and the bounding boxes along with the probabilities are predicted by the network for each region.

The network structure of the YOLO is named Darknet-53 [7] and is shown in Fig. 3, which is called a fully convolutional neural network as it consists of only convolution layers. This model has a total of 53 convolution layers. The neural network down-samples the image by a factor called stride. For example, if the stride of the network is 16, then an input image with size 512×512 will yield an output with size 32×32.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3. Network Architecture of YOLOv3 (Darknet-53) [7]

D. *Monocular Depth Estimation Algorithm*

Although many existing depth estimation algorithms were trained with color input images and their corresponding depth values [8][9][10]. it is not possible to obtain the ground truth depth data of various scenes. Even precise hardware-supported system such as laser scanners can produce inaccurate results due to the reflections and various other factors.

Our system design utilizes a single camera system by adapting monocular depth estimation techniques to obtain the depth data from a single captured image.

KITTI dataset is used in our system [11]. To build the dataset, images are captured with the help of a pair of calibrated stereo cameras. All these images (left and right) were used for the training. Now instead of training this data to calculate the depth, the model has been trained to find the dense correspondence field [12], that when applied to the right image it would be able to create the left image. Similar things can be done to create the right image. Once the algorithm can create a left and right image and the images are rectified [13] it will be able to calculate the depth. Fig. 4 shows the generated depth map for a sample image.

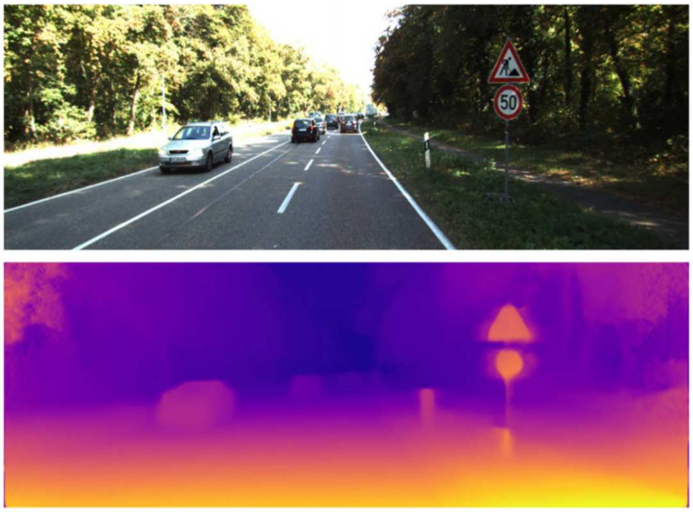


Figure 4. Depth Prediction Result

III. System Implementation

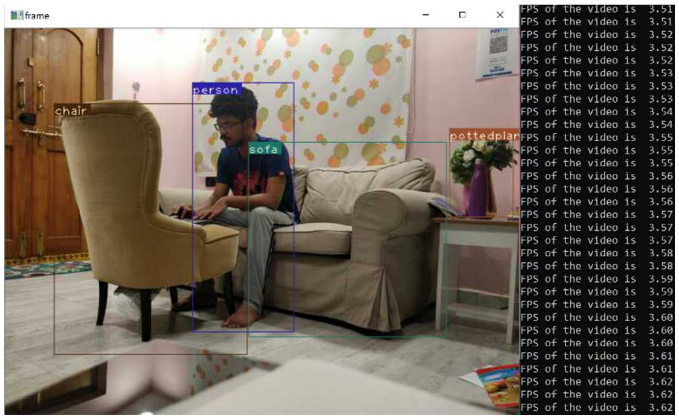
A. *YOLO Object Detection*

We have performed tests on different hardware platforms to seek the feasibility of the hardware for YOLO object detection. We tested on three different platforms; one with no GPU installed, one on an Nvidia Jetson Nano, and one on an Nvidia Jetson 1050. Sample images from the COCO (Common Objects in Context) dataset are given to the YOLO script, and average detection time and frames per second are observed. The results are shown in Table I.

From the results shown in Table 1, we can observe that the first two devices (no GPU and Nvidia Jetson Nano) are unsuitable for real-time YOLO applications. Thus, we utilized an Nvidia GTX 1050 throughout the implementation. When using the video frames from a webcam instead of sample images, the frame rate increased slightly to 15 fps. However, the frame rate dropped to 4-7 fps when using an IP webcam, which is still good for real-time object detection in our software. Fig. 5 shows the frame rate performance results obtained with an IP webcam.

TABLE I. YOLO PERFORMANCE ON DIFFERENT HARDWARE

Hardware	Average Detection Time (seconds)	Frames per Second
No GPU	56	0.018
Nvidia Jetson Nano	0.49	2.05
Nvidia GTX 1050	0.08	12.5



```
FPS of the video is  3.51
FPS of the video is  3.51
FPS of the video is  3.52
FPS of the video is  3.52
FPS of the video is  3.52
FPS of the video is  3.53
FPS of the video is  3.53
FPS of the video is  3.53
FPS of the video is  3.54
FPS of the video is  3.54
FPS of the video is  3.55
FPS of the video is  3.55
FPS of the video is  3.55
FPS of the video is  3.56
FPS of the video is  3.56
FPS of the video is  3.56
FPS of the video is  3.57
FPS of the video is  3.57
FPS of the video is  3.57
FPS of the video is  3.58
FPS of the video is  3.58
FPS of the video is  3.59
FPS of the video is  3.59
FPS of the video is  3.59
FPS of the video is  3.60
FPS of the video is  3.60
FPS of the video is  3.60
FPS of the video is  3.61
FPS of the video is  3.61
FPS of the video is  3.62
FPS of the video is  3.62
FPS of the video is  3.62
```

Figure 5. Frame Rate Obtained using IP Webcam

B. *Monocular Depth Estimation*

Fig. 6 is an example of the output of the monocular depth estimation based on the IP Webcam application's image feed.

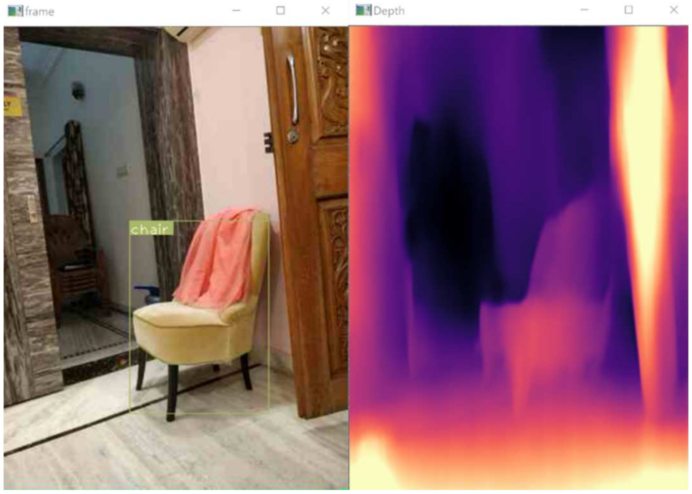


Figure 6. Sample Image and Depth Map Obtained from IP Webcam Feed

The model used is trained on the KITTI dataset. The depth of each pixel is calculated using the formula as in equation 1.1

$$D = fB/d \tag{1}$$

where D is the depth of the image in meters, f is the focal length in pixels, B is the baseline in meters, and d is depth in pixels.

The baseline for the KITTI stereo dataset is 0.54m. The model used for monocular depth estimation has an effective baseline of 0.1 units. Thus, a scaling of 5.4 was applied for depth prediction.

C. *Azimuth and Elevation Measurement*

To measure the azimuth and elevation of a detected object, the angle of view of the camera and the resolution are required. Those parameters vary with the actual camera used and need to be determined from the datasheet of the camera. We have the following equation to determine the relative horizontal position (azimuth) of an object:

$$\theta = \frac{P_h \alpha}{M} - \frac{\alpha}{2} \tag{2}$$

where P_h is the horizontal position of the geometric center of the region, in terms of the number of pixels from the left edge of the image; α is the horizontal field of view of the depth camera in degrees; M is the resolution of the image along the horizontal axis; θ is the azimuth of the obstacle in degrees, 0 indicates dead ahead, negative value indicates on the left, and positive value indicates on the right.

Similarly, the following equation can be used to determine the relative vertical position (elevation) of an obstacle:

$$\varphi = \frac{P_v \beta}{N} - \frac{\beta}{2} \tag{3}$$

where P_v is the horizontal position of the geometric center of the region, in terms of the number of pixels from the bottom of the image; β is the vertical field of view of the depth camera in degrees; N is the resolution of the image along the vertical axis; φ is the elevation of the object in degrees, 0 indicates dead ahead, positive value indicates above the horizon, and negative value indicates below the horizon.

D. *Spatial Audio Generation Using HRTFs*

To give the spatial output to the visually impaired person, a simple mono audio wave file is required. To generate this audio wave file, we have used a simple clapping sound. HRTFs are generated for every 15 degrees azimuth and 5.625 degrees elevations. To generate a spatial audio file, the mono audio wave file is convoluted with a specific HRTF relating to the azimuth and elevation angle. As soon as the audio file rendering is complete the audio will be automatically played in the real time.

IV. RESULTS

A. *Object Detection and Depth Estimation*

To calculate the depth of the detected object, first we calculated the center of the detected bounding box. Then, we used the formula as depicted in equation 1 to accurately calculate the depth of a 50x50 pixels at the center of the detected object's bounding box. Fig. 7 shows the result of the object detection and depth estimation.

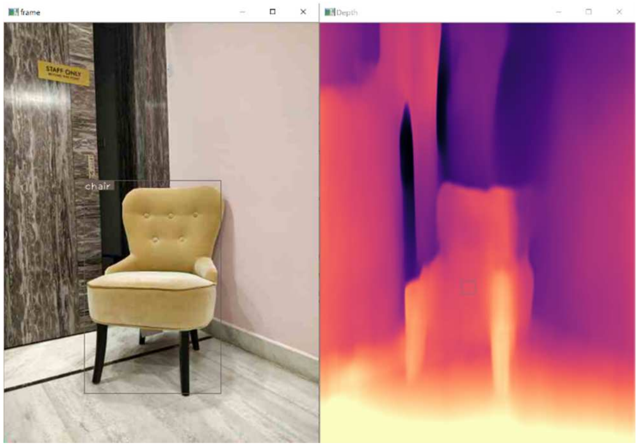


Figure 7. Object Detection, Depth Map Generation and Depth Estimation Result

We used various objects to verify the accuracy of the algorithm. As shown in Table II, the performance of the depth estimation model for various objects. The below results were obtained after fine-tuning and calibration of the predicted depth. Our model was able to predict with an average error of 400mm. Considering the fact that the depth data is obtained from a single image this is an acceptable result.

TABLE II. EXPERIMENTAL RESULTS OF DEPTH ESTIMATION

Object	Actual Distance (mm)	Calculated Distance(mm)
Table	700	540
Chair	1,600	1,820
Refrigerator	1,200	972
Bottle	300	742
Dining Table	600	1,172
Chair	3,200	2,526
Sofa	1,700	1,351

B. *Azimuth and Elevation Calculation*

Fig. 8 is the result of the simple linear interpolation calculation of azimuth and elevation using sample objects including a bottle and a bench. The result shown describes the distance from the camera, as well as the azimuth angle, elevation angle, and the object's direction. HRTFs are measured for small increments of angles. In the HRTF dataset used, the HRTFs are calculated in increments of 15 degrees for azimuths and 5.625 degrees for elevations. So, to be able to use the HRTFs the azimuth index (*aIndex*) and elevation index (*eIndex*) are calculated.

C. *Generation of Audio Navigation Signals Using HRTFs*

After these indexes are calculated, spatial audio is generated in real-time using mono audio signals, provide information of the depth and indexes. The below are some pictures for which the generated audio is saved. In the below pictures the position of a bottle is varied from left to right. In this way, we can see the change in the generated spatial audio. The generated audio is saved in the folder "Generated Audio" along with the numbering as per the above pictures. An example environment and the parameters of the detected objects are shown in Fig. 8. The corresponding audio signal is shown in Fig. 9. It can be seen that the volume for the left ear is larger than that of the right ear, which matches the object placement in Fig. 8.

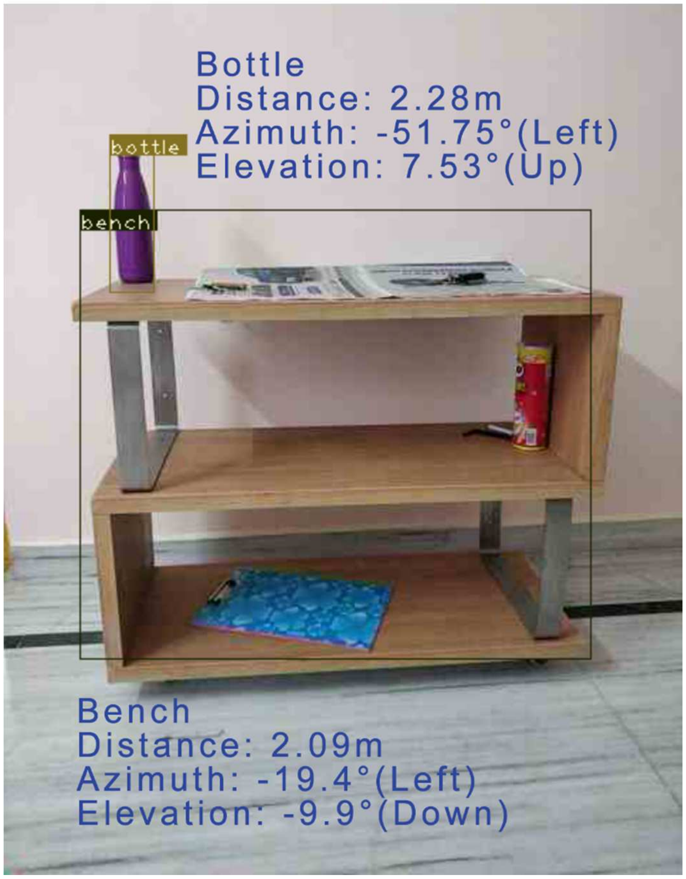


Figure 8. Object Detection in an Example Environment

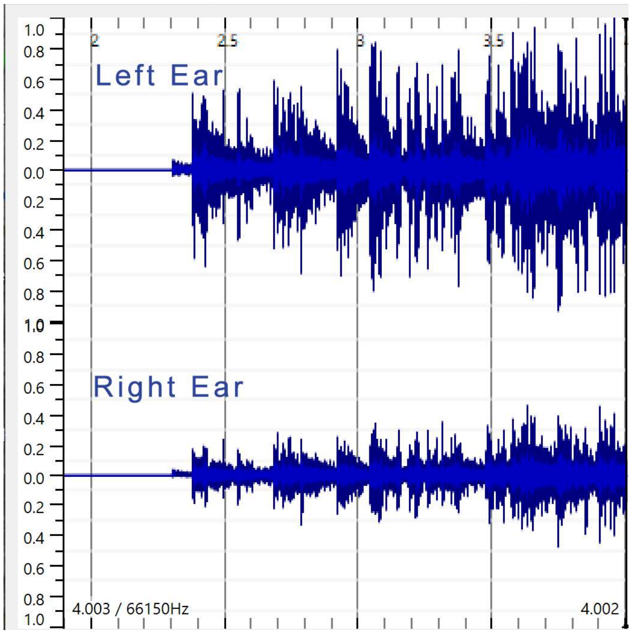
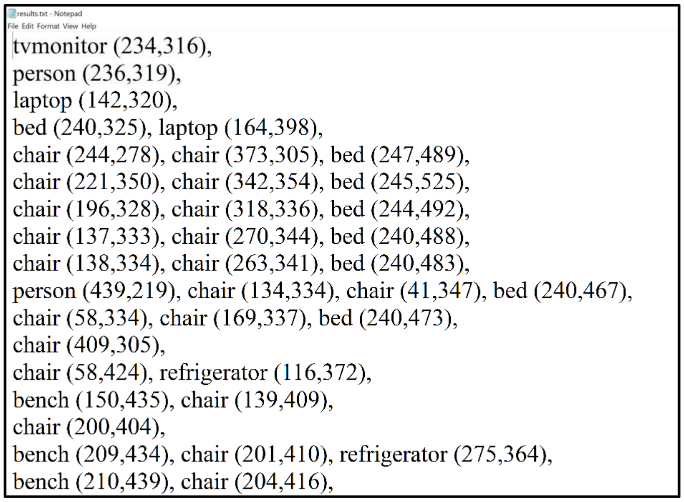


Figure 9. Corresponding Audio Signal

D. Data Logging

Data logging provides necessary information to make decisions and improve the efficiency of a system. The categories of the detected objects and their positions are stored in text files. One example is shown in Fig. 10.



```
tvmonitor (234,316),
person (236,319),
laptop (142,320),
bed (240,325), laptop (164,398),
chair (244,278), chair (373,305), bed (247,489),
chair (221,350), chair (342,354), bed (245,525),
chair (196,328), chair (318,336), bed (244,492),
chair (137,333), chair (270,344), bed (240,488),
chair (138,334), chair (263,341), bed (240,483),
person (439,219), chair (134,334), chair (41,347), bed (240,467),
chair (58,334), chair (169,337), bed (240,473),
chair (409,305),
chair (58,424), refrigerator (116,372),
bench (150,435), chair (139,409),
chair (200,404),
bench (209,434), chair (201,410), refrigerator (275,364),
bench (210,439), chair (204,416),
```

Figure 10. Data Logging of the Detected Objects

V. Conclusion

This paper has explored the feasibility of a system design to navigate visually impaired people in the indoor environment. Utilizing an IP Webcam application, it was possible to obtain the surrounding environment status. Also, with the help of the state-of-the-art YOLO algorithm, monocular depth estimation techniques, and azimuth and elevation estimation, our system design can produce spatial audio generated by HRTF to assist visually impaired people to avoid obstacle collisions and to guide them to the safe path. Our system design has the potential to be expanded to a commercially available and portable indoor navigation system for the visually impaired.

References

[1] Centre, W.M. Visual impairment and blindness. https://www.who.int/blindness/data_maps/VIFACTSHEETGLODAT2010full.pdf?ua=1

[2] M. M. Islam, M. Sheikh Sadi, K. Z. Zamli and M. M. Ahmed, "Developing Walking Assistants for Visually Impaired People: A Review," in *IEEE Sensors Journal*, vol. 19, no. 8, pp. 2814-2828, 15 April, 2019.

[3] W. Chang, L. Chen, C. Hsu, J. Chen, T. Yang and C. Lin, "MedGlasses: A Wearable Smart-Glasses-Based Drug Pill Recognition System Using Deep Learning for Visually Impaired Chronic Patients," in *IEEE Access*, vol. 8, pp. 17013-17024, 2020.

[4] W. -J. Chang, L. -B. Chen, M. -C. Chen, J. -P. Su, C. -Y. Sie and C. -H. Yang, "Design and Implementation of an Intelligent Assistive System for Visually Impaired People for Aerial Obstacle Avoidance and Fall Detection," in *IEEE Sensors Journal*, vol. 20, no. 17, pp. 10199-10210, 1 Sept, 2020.

[5] O. Wasenmüller, M. Meyer and D. Stricker, "CoRBS: Comprehensive RGB-D benchmark for SLAM using Kinect v2," *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Placid, NY, pp. 1-7, 2016.

[6] A. Aladrén, G. López-Nicolás, L. Puig and J. J. Guerrero, "Navigation Assistance for the Visually Impaired Using RGB-D Sensor With Range Expansion," in *IEEE Systems Journal*, vol. 10, no. 3, pp. 922-932, Sept. 2016.

[7] Joseph Redmon, Ali Farhadi, University of Washington, YOLOv3: An Incremental Improvement. https://arxiv.org/pdf/1804.02767.pdf

[8] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In NIPS, 2014

[9] L. Ladicky, J. Shi, and M. Pollefeys. Pulling things out of perspective. In CVPR, 2014

[10] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. PAMI,2015

[11] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, pp. 3354-3361, 2012.

[12] Clement Godard, Oisin Mac Aodha, Gabriel J. Brostow , University College London, Unsupervised Monocular Depth Estimation with Left-Right Consistency http://visual.cs.ucl.ac.uk/pubs/monoDepth/

[13] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003