Navigation Assistance Platform for Blind People

Biagio Cornacchia, Fabrizio Lanzillo, Gianluca Gemini, Matteo Abaterusso
b.cornacchia@studenti.unipi.it, f.lanzillo@studenti.unipi.it, g.gemini@studenti.unipi.it, m.abaterusso@studenti.unipi.it
https://github.com/bfgm-unipi/navigation-assistance-platform-for-blind-people

ABSTRACT

This project aims to assist visually impaired and blind (VIB) people in everyday life by detecting obstacles in an outdoor environment, in order to warn the user and help him/her to move safely. More specifically, it is designed to work in combination with traditional tools such as a walking cane, guide dogs, and human assistant.
The system uses a depth map generated by Google's ARCore Depth Lab API that is used to compute the distance between camera and obstacles. If the user gets too close to them, he/she is alerted via audio and vibration feedback.
In order to not overload the system, only specific points on the depth map have been considered. These points identify the body parts that would be occluded by a potential obstacle. The selected body parts are head, left and right torso, and left and right floor.
Moreover, to evaluate the correct functioning of the system, both static and dynamic tests have been carried out.
Specifically for the static tests, vertical, horizontal and hardly visible obstacles have been examined.
On the other hand, the dynamic tests, which are composed of two tests, focus on moving obstacles. In the first one, two subjects walk towards each other, while in the second one subject walks in front of the subject with the depth camera, in a direction perpendicular to him/her.

1 Introduction

Globally, at least 2.2 billion people have a near or distance vision impairment. Of this number, about 1 billion people suffer from moderate to severe visual impairment or complete blindness due to untreated refractive errors (88.4 million), cataracts (94 million), age-related macular degeneration (8 million), glaucoma (7.7 million), diabetic retinopathy (3.9 million), and near vision disorders caused by untreated presbyopia (826 million)[1].
One of the main difficulties for visually impaired and blind (VIB) people during their daily activities is moving within spaces that contain obstacles. Therefore, the aim of this project is to use an augmented reality device to assist them in addition to their walking cane in an outdoor environment. In particular, the system proposed uses a depth camera to record the distance between the subject and the obstacles. As soon as the subject is getting closer to the obstacle (under a certain distance threshold), the device will warn him/her of imminent danger using a sound and vibration feedback. These functionalities are developed by exploiting the ARCore Google platform.

2 Application Components

As mentioned before, the application exploits ARCore Depth Lab API to construct the depth map using the RGB camera. The depth map associates each pixel of the image with a color representing the estimated distance between the camera and the relative point in the space. Warm colors represent near elements while cold colors represent far elements. From these pixels, it is possible to calculate the distance in millimeters and thus estimate the distance of the objects captured by the camera.
The system has been organized into a pipeline in which a sequence of components is executed each time a new depth map is constructed. The four components of the pipeline will be analyzed below.

2.1 Collision Points Distance Computation

The goal of this component is to calculate and estimate, using a depth map, the distance of specific points on the screen, called collision points.
The collision points mentioned above are intended to simulate the position of the VIB person within the space surrounding him/her. These points are grouped into specific areas, coinciding with particular zones of the body, with the purpose of figuring out which zone is at risk of collision.
The selected areas are head (3 points), left torso (8 points), right torso (8 points), left floor (2 points), right floor (2 points).
Only a limited number of points and not all pixels in the depth map are analyzed, in order to not overload the application in terms of performance.

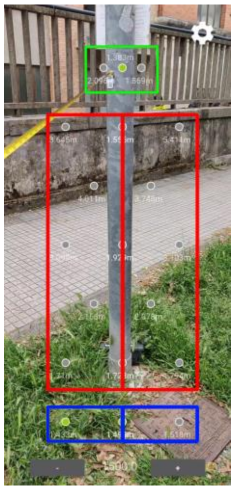

Figure 1: Example of Collison Points Position

A function, *getDistanceInMillimiters*, is used to calculate the distance for a given point, where through its coordinates, its position is retrieved in the depth map and the distance in millimeters is returned.
In order to improve the accuracy of the distance, the distance is not carried out from only the coordinates x, y of a specific point, but an average of the distances of multiple points, which are in a 5-pixel neighborhood of the selected point, is calculated.
Once the distances are obtained for each collision point, each of them is evaluated against a configurable threshold, the distanceThreshold. If at least one point in a zone is below that threshold, the entire zone is declared collision-prone.

2.2 Collision Points Visualization Control

The collision points visualization component provides a feature which allows to display the previously introduced collision points on the smartphone screen. This is useful both for testing the application and to calibrate it. Moreover, it is shown the current distance between each collision point and the subject who is wearing the smartphone. The information about the distance is obtained by exploiting the *distance computation component*. As soon as the user gets too close from an obstacle (if its distance is below a certain threshold), the point on the screen will switch from gray to green.

2.3 Feedback Systems

To alert the user to the presence of obstacles, two feedback systems have been implemented, one based on vibration and the other on audio.
As explained before, obstacles are classified into body parts that will be occluded, to verify the actual presence of obstacles and avoid false and missed detections, a counter is associated for each body part. A counter is incremented (until a maximum value) for each detection of an obstacle in that body part and it is decremented (until minimum value) if the obstacle is no longer detected. The midpoint between the maximum and minimum is the threshold above which the body part is considered truly occluded, below which it is considered no longer occluded.
The two feedback systems take as input a data structure containing body part counters.
The vibration system alternates periods of 500 ms in which the vibration is enabled and periods of 100 ms in which it is disabled.
This system is activated as soon as an obstacle is detected and is only deactivated if there are no more obstacles.
The audio system is based on simple voice prompts, obtained with a Text-to-Speech library, which are used to indicate to the user the presence of obstacles and the part of the user's body that will be occluded, so that the user can avoid them. This feedback system is activated only the first time one or more occlusions are detected, if the occluded body parts change or there are no more occlusions, a new alert is activated.

2.4 Body Parts Visualization Control

The feedback system component periodically produces a filtered list of the body parts that face an obstacle under a certain distance. The application has been provided with a visualization mode that allows these body parts to be shown in real time. To do this, an image containing a human profile has been chosen, subsequently divided into the five areas that the application should be able to notify as free or occupied (head, left and right torso, left and right floor). Two versions were produced for each of these images, a green one indicating that part is free and a red one indicating that part of the body is facing an obstacle. Every time that the feedback system produces a new body part list, the interface will be updated by coloring in red the parts in the list, and coloring the others green.

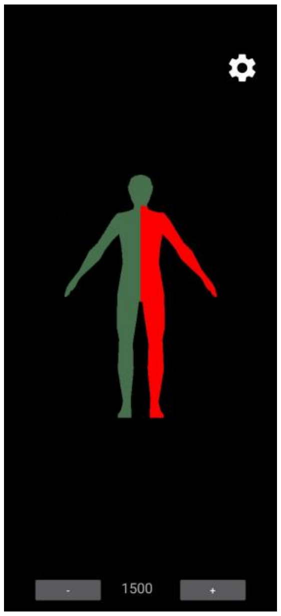

Figure 2: Example of body parts visualization

The application at this point provides multiple visualization modes for debugging purposes (occupied parts, collision points, depth map, etc...), so a quick settings menu is developed to enable and disable the various modes on demand in real time.

3 Experimental results

To evaluate the reliability of the system, systematic tests has been performed considering several common obstacles in an outdoor environment. Each test has been repeated at 3 meters, 2.5 meters, 2 meters, 1.5 meters, 1 meter, and 0.5 meters, respectively. In the evaluation, a distinction is made between what is detected by the depth map and the distances calculated at the chosen points (collision points). In particular, the test with the depth map is considered passed if the obstacle is correctly distinguished from the rest of the environment, while it is considered passed for collision points if the sampled distances from the obstacle are close to the

real distances (error less than 0.1 meters). In addition, the tests have been divided into static and dynamic tests, the first ones keeping the subject stationary in front of an obstacle and the second ones with the obstacle in movement.
To perform these tests a OnePlus Nord has been used, with a Qualcomm SDM765 Snapdragon 765G CPU, Adreno 620 GPU, 8GB RAM, 48 megapixel camera, 4100 mAh battery. This smartphone is not equipped with a dedicated ToF sensor.

3.1 Vertical Obstacles

One type of the possible obstacles to be analyzed in order to test the robustness of the application are vertical-type obstacles, such as a light pole, tree, or person. The light pole has been chosen since it could create problems for the application due to its shape. Indeed, experiments have shown that the detections were not very stable. The light pole is always detected correctly within the depth map, instead the collision points show an error if standing at least 1 meter away from it. Specifically, the distance shown is affected by an error that increases as you move away from it. This causes the application to not signal the subject because the minimum threshold distance is not reached. Regarding the tree, it is always detected correctly within the depth map. Instead, as for the light pole, collision points often show a distance affected by an error if the subject stands at a distance ranging from 2 meters to 3 meters away.

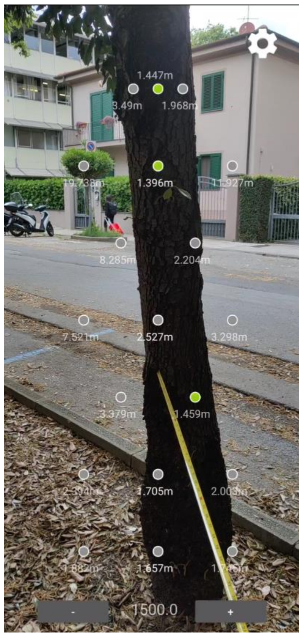

Figure 3: Example of Tree Detection

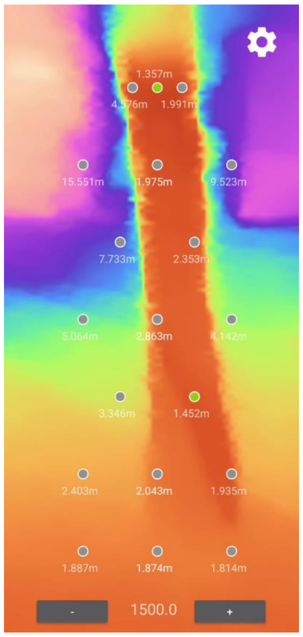

Figure 4: Depth Map of Tree

Finally, the detection test of a person gave good results. In particular, the person is correctly detected both in the depth map and from the collision points, except for the case where the subject is 3 meters away from him/her, reporting a distance affected by error of less than half a meter.

3.2 Horizontal Obstacles

Regarding obstacles extending horizontally, a small wall of about 1m and a sidewalk step of about 15 cm have been chosen.
As long as the wall enters the camera frame, it is correctly detected at all distances considering a negligible error of maximum 10 cm.

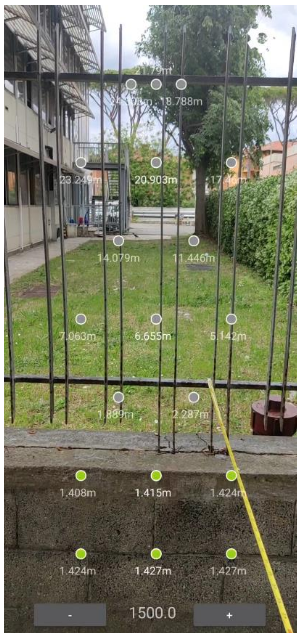

Figure 5: Example of Small Wall Detection

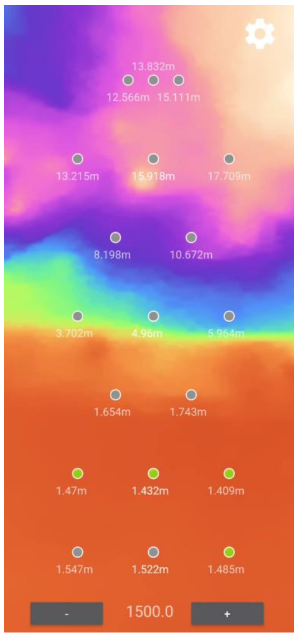

Figure 6: Depth Map of Small Wall

Instead, the sidewalk step is never detected because its height is too low and ARCore detects it as part of the pavement by not showing an obvious color difference in the depth map.

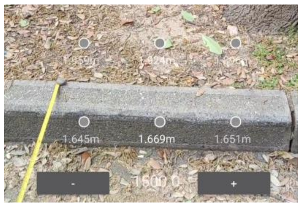

Figure 7: Example of Sidewalk Step

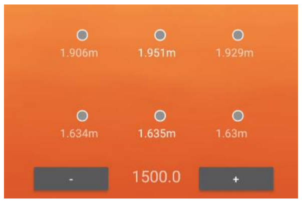

Figure 8: Depth Map of Sidewalk Step

3.3 Hardly Visible Obstacles

The aim of this test is to check whether the system can distinguish hardly visible surfaces and objects at different distances.
In particular, a glass door as a surface and an iron fence as an object have been examined.

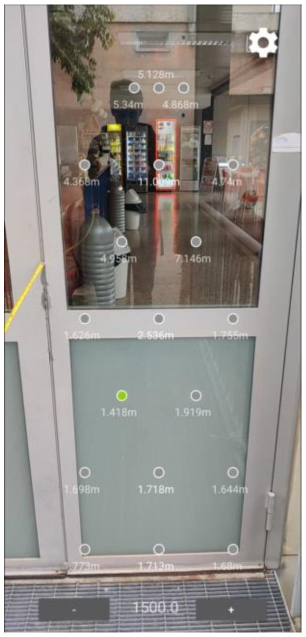

Figure 9: Example of glass door detection

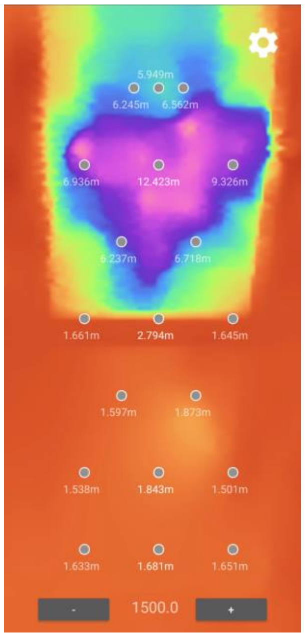

Figure 10: Depth Map of the glass door

As shown in the figures, standing 1.5 meters away, the glass of the door is not detected by the system.
The behavior is almost identical for the iron fence as well, since the iron bars are not detected.
In addition, it has been observed that these behaviors remain constant whether we increase the distance or decrease it.
We conclude from this that the glass being a transparent material is not detected through an RGB camera, while the bars of the iron fence are too thin to reach a detectable thickness in the depth map.

3.4 Dynamic Tests

As last part of the experimentation, dynamic tests are conducted. The purpose of this phase is to verify the application's capabilities to detect moving obstacles.
In particular, two specific tests are conducted. In the first one, two subjects walk toward each other. In the second, on the other hand, a subject walks in front of the subject with the depth camera, in a direction perpendicular to him. The latter has been repeated three times at three different distances (1 meter, 1.5 meters, 2 meters).
In the first test, the depth map starts to show the figure of the subject when he is at a distance of about a meter and an half (as shown in the figure below), but it fails to associate him with a correct distance, which remains the same as the background.

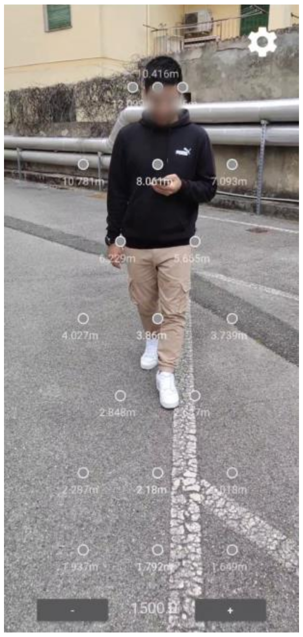

Figure 11: Example of Obstacle Moving Towards the Subject

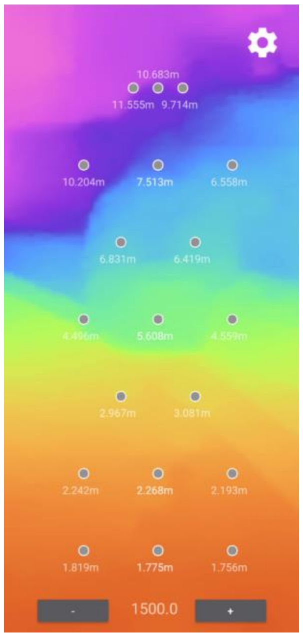

Figure 12: Depth Map of Obstacle Moving Towards the Subject

In the second test, the result is similar. The subject is roughly visible within the depth map after about one second from when he get into the frame. But by not remaining in a fixed position, the detection points don't have enough time to stabilize at the correct distance, causing the application to be unable to correctly detect the obstacle.

4 Limitations and Conclusion

In conclusion, ARCore makes it possible to implement an obstacle detection system, but against a number of limitations.
First, in order to have consistent distance values, the framing must remain stable for a certain amount of time. In fact, this also causes the application to be unable to detect moving obstacles.
In addition, to properly detect obstacles of any size, it is necessary to use a large number of collision points.
Moreover, transparent obstacles and other hardly visible obstacles are not detected.
Finally, because people differ in height and body composition, it is necessary to configure the detection threshold and device placement specifically for each subject.

REFERENCES

[1] World Health Organization. Blindness and Vision Impairment. Available online: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment (accessed on 6 January 2022).
[2] See, A.R.; Sasing, B.G.; Advincula, W.D. A Smartphone-Based Mobility Assistant Using Depth Imaging for Visually Impaired and Blind. Appl. Sci. 2022, 12, 2802. https://doi.org/10.3390/app12062802
[3] Google's ARCore API. Available online: https://developers.google.com/ar?hl=it