Navigation Assistance Platform for Blind People

Team Members:
Matteo Abaterusso
Biagio Cornacchia
Gianluca Gemini
Fabrizio Lanzillo

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

ROADMAP

Introduction

Experimental results

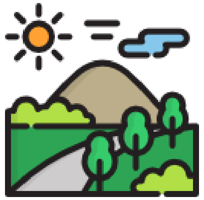
Application Components

Conclusions

Introduction

- **TARGET:** 1 billion people are visually impaired or completely blind (VIB)

- **AIM:** Assist VIB people in their outdoor activities



- **HOW TO DO:** Use a depth map, provided by ARCore Google Platform for android, to detect obstacles and alert the user with audio and vibration feedback

ARCore

Application Components

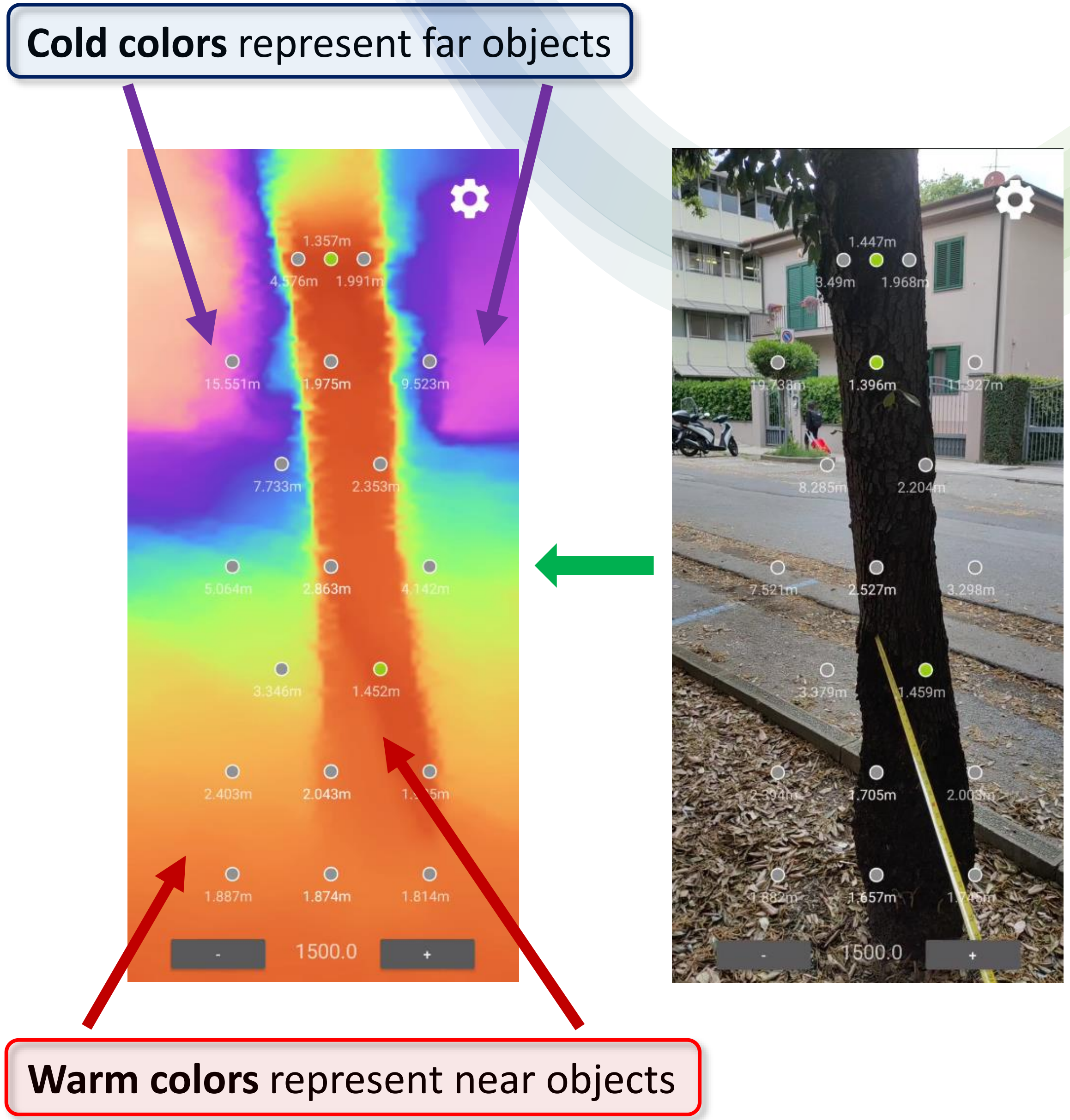
The application exploits **ARCore Depth Lab API** to generate a **depth map** of a captured environment using the **RGB camera**.

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**.

From these points, **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 generated.
These components are:

- *Collision Points Distance Computation*
- *Collision Points Visualization Control*
- *Feedback System*
- *Body Parts Visualization Control*



App. Components: Collision Points Distance Computation

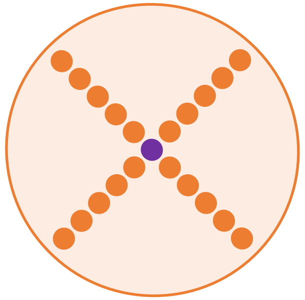
The **goal** of this **component** is to **calculate and estimate**, using the depth map, the **distance of** specific points on the screen, called ***Collision Points***.

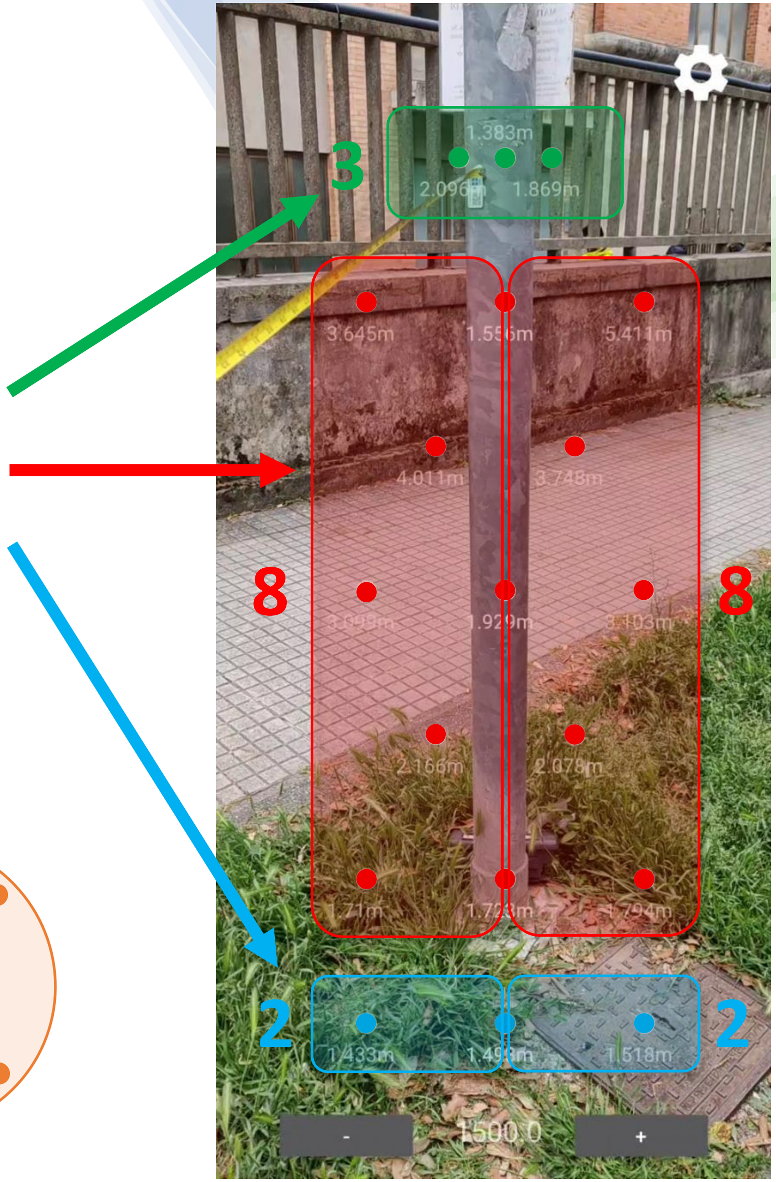
The **collision points simulate** the **position** of the **VIB person within** the **space** surrounding him/her.

These points **are grouped** into **specific areas**, **coinciding** with **particular zones** of the **body**, with the purpose to **figuring out** which **zone** of the body is at **risk of collision**.

The 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 given point (purple)**, but is equal to the **average** of the **distances of multiple points** within an area of **5-pixel radius (orange)**, starting from the given point (purple).





App. Components: Collision Points Visualization Control

The component provides a **feature** which allows to **display** the previously introduced **collision points**.

It also shows the **current distance** between **each collision point** and the **subject** who is wearing the smartphone.

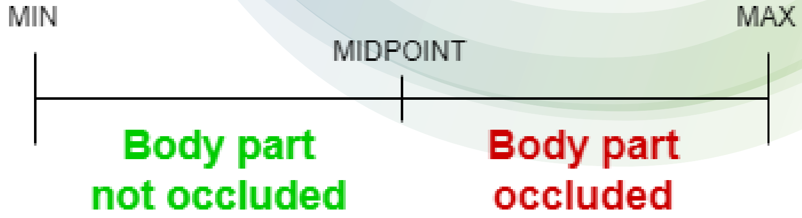
As soon as the subject **gets too close** to an obstacle, the point on the screen will switch from **gray** to **green**.

The information about the distance is obtained by exploiting the **distance computation component**.

App. Components: Feedback Systems

Avoid false and missed detections:

- A counter for each body part
- Increment the counter if an obstacle is detected
- Decrement the counter if the obstacle is not detected



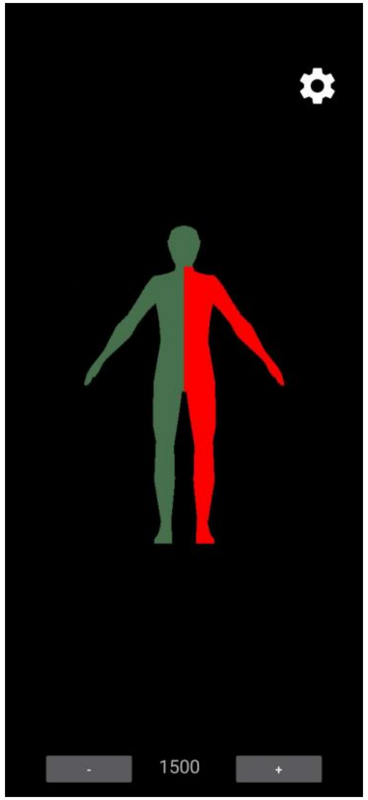
Two feedback systems to alert the user if one or more body parts are occluded

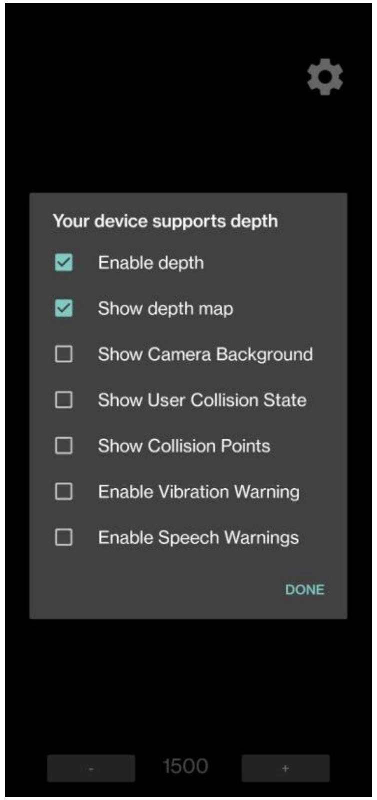
- **Vibration Feedback:** Continuous vibration that ends only if there are no body parts occluded and starts if almost one body part is occluded
- **Audio Feedback:** Useful audio indication for obstacle avoidance, starts once per detection

App. Components: 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.





Experimental Results

To evaluate the reliability of the system, **systematic tests** has been performed considering several **common obstacles** in an **outdoor environment**.

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.

Each test has been repeated at 3 meters, 2.5 meters, 2 meters, 1.5 meters, 1 meter, and 0.5 meters, respectively.

A distinction is made between what is detected by the **depth map** and the distances detected by the **collision points**.

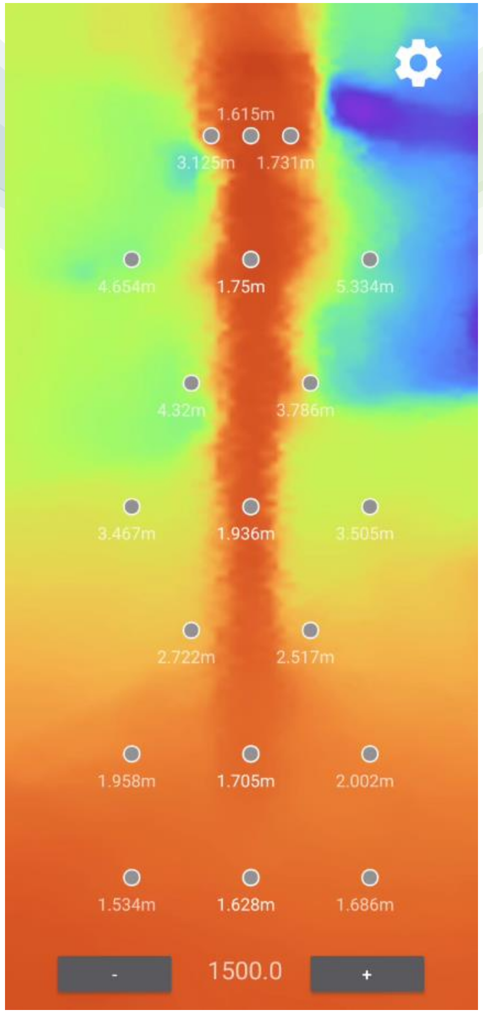
ONEPLUS NORD

- **CPU**: Qualcomm SDM765 Snapdragon 765G
- **GPU**: Adreno 620
- **RAM**: 8GB LPDDR4X
- **Camera**: 48 megapixel
- **Battery**: 4100 mAh

Static Test: Vertical Obstacles

Starting from the **light pole**

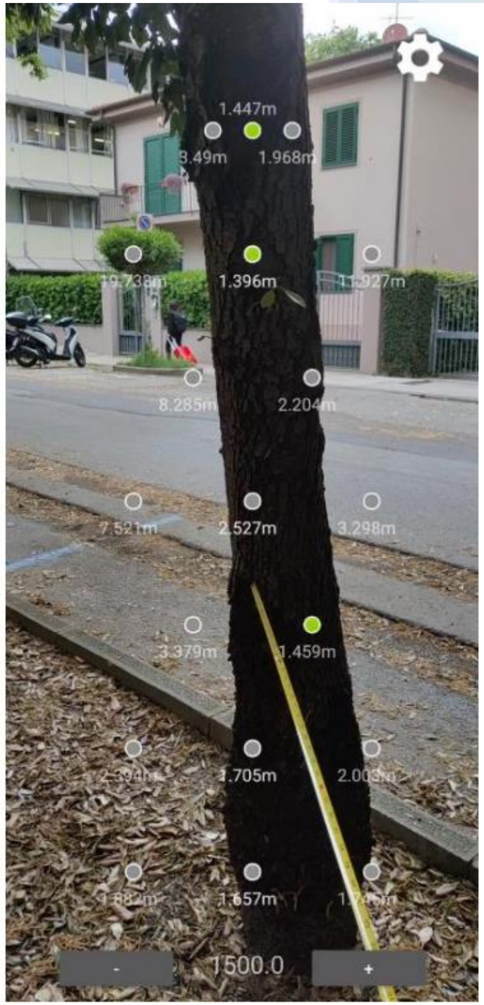
- Experiments have shown that **detections** were **not very stable**
- The **depth map** is able to **detect** it **correctly**
- Unfortunately, the **collision points** show an **error** if the subject stands at least **1 meter** away from it

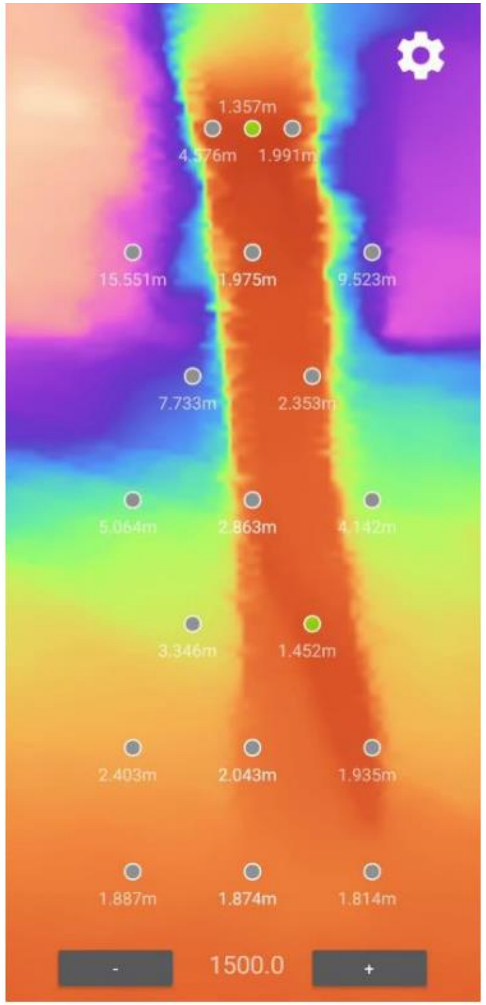


Static Test: Vertical Obstacles

Regarding the **tree**

- It is **correctly detected** within the **depth map**
- The **collision points** often show a distance affected by an **error** if the subject stands **2 meters** and up from it

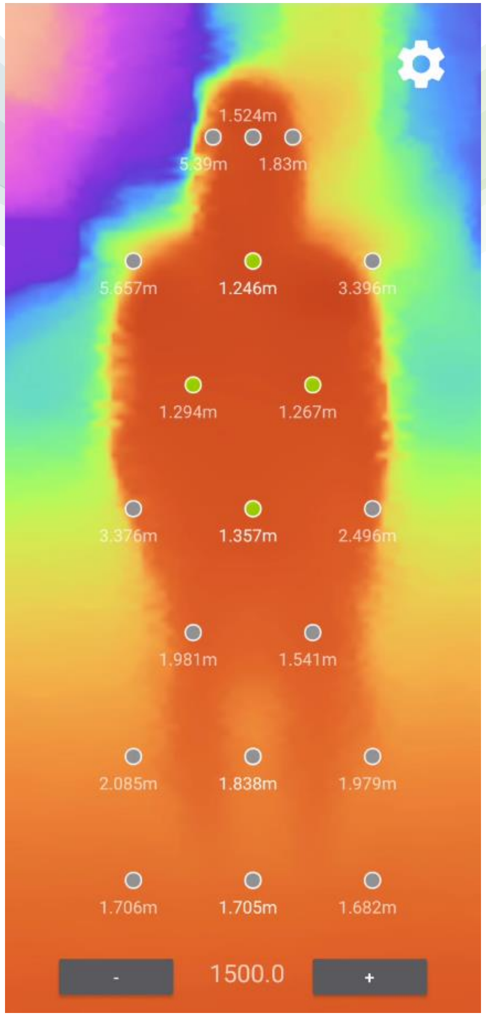




Static Test: Vertical Obstacles

Finally, a **person**

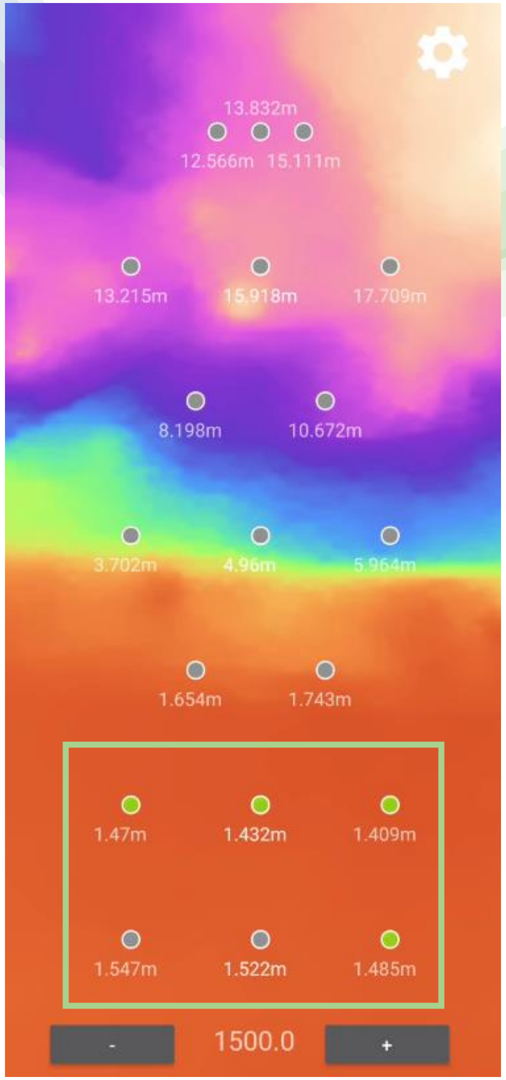
- It is **correctly detected** both in the **depth map** and by the **collision points**
- Except for the case where the subject is **3 meters** away from him/her, reporting a distance affected by error of **less than half a meter**



Static Test : Horizontal Obstacles

Two horizontal obstacles are chosen for tests:

- **Small Wall** 1 meter high : always detected with an error of maximum 10 cm
- **Sidewalk Step** 15cm high: never detected and considered part of the pavement



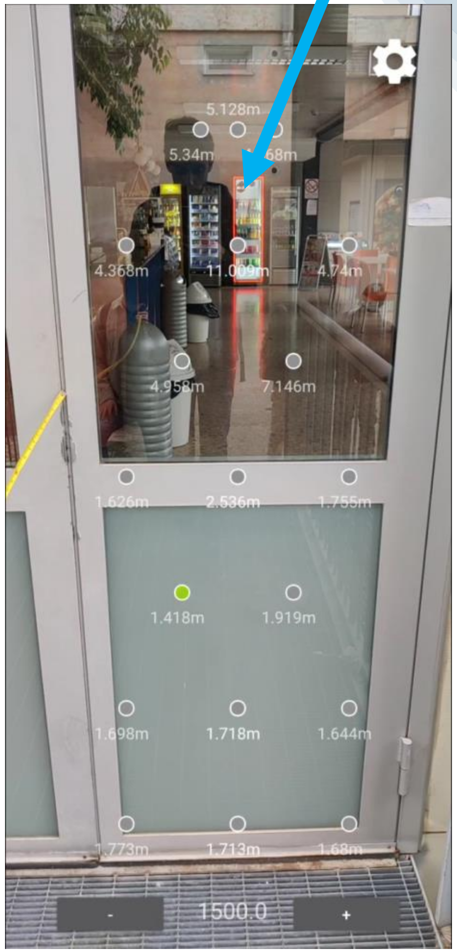
Static Test : Hardly Visible Obstacles

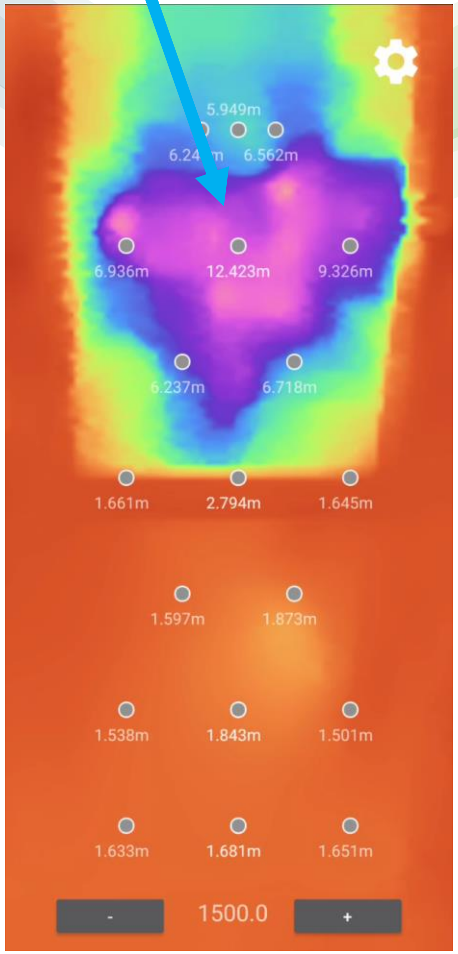
The **aim** of this test is to **check** whether the system can **distinguish** hardly visible **surfaces** and **objects** at different distances. We examined a **glass door** as a surface and an **iron fence** as an object.

As shown in the images, standing 1.5 meters away, the **glass of the door** is neither **detected** in the depth map nor by the **collision points**.

In addition, it has been observed, that **these behaviors remain constant** whether we **increase** or **decrease** the **distance** from the glass door.

Glass Part of the Door



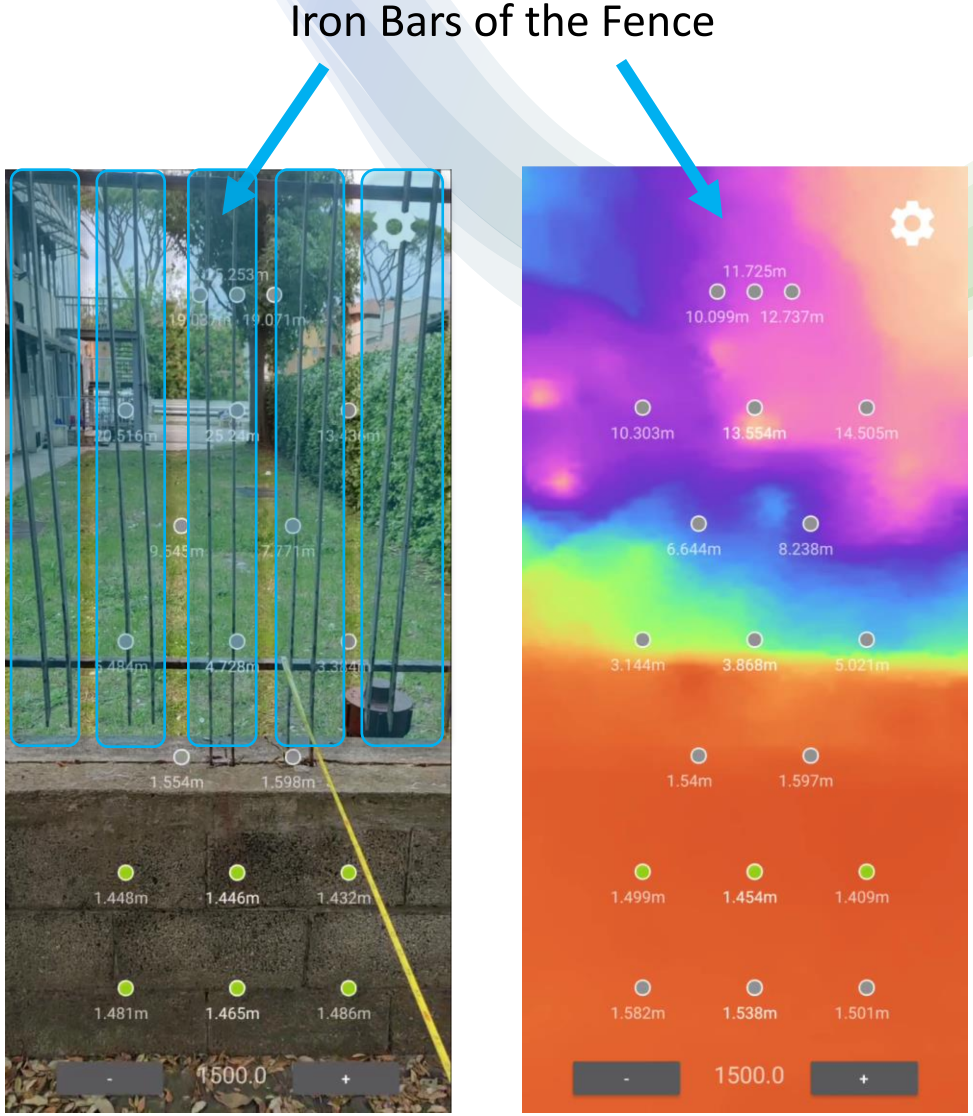


Static Test : Hardly Visible Obstacles

The **behavior** is **almost identical** for the iron fence as well, in fact, the **bars** that we can see in the original image are **neither detected** by the **collision points** nor in the **depth map**.

As for the glass door, the **non-detection doesn’t change** if we **increase** or **decrease** the **distance** from the iron fence.

We **conclude** 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**.

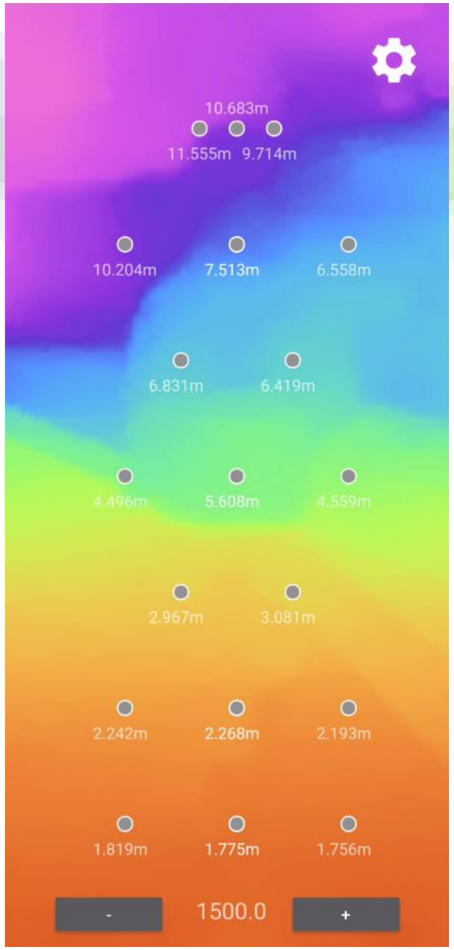


Dynamic Tests: Obstacle Moving Towards

The purpose of dynamic tests is to verify the application's capabilities to **detect moving obstacles**.

In the first test, two subjects **walk toward** each other.

The **depth map** starts to show the figure of the subject when he is at a distance of about a meter and a half, but it fails to associate him with a **correct distance**, which remains the same as the background.

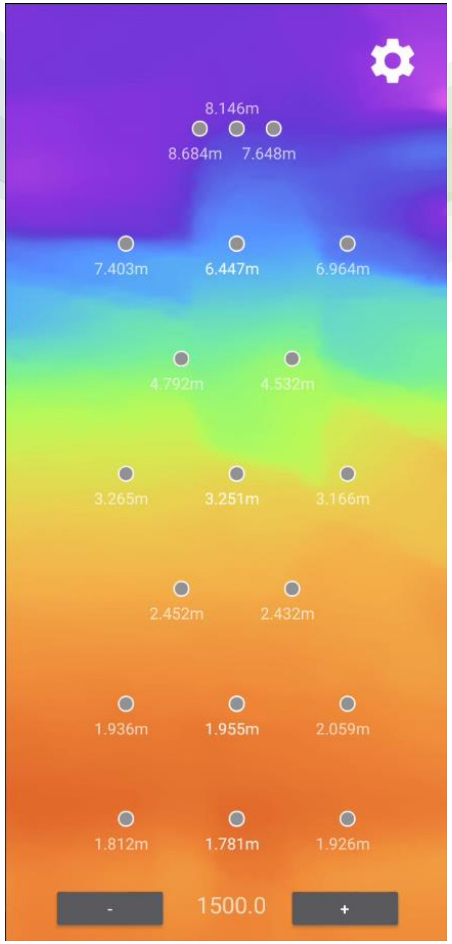


Dynamic Tests: Obstacle Moving Perpendicular

In the second test, another subject **walks** in **front** of the subject with the depth camera, in a **direction perpendicular** to him.

The test has been repetad **three times** at 1 meter, 1.5 meters and 2 meters.

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.



Limitations and Conclusion

ARCore makes it possible to implement an obstacle detection system, but against a few **limitations**:

- In order to have **consistent distance values**, the **camera** must remain **stable** for a certain amount of time
- The application **is not able** to detect **moving obstacles, transparent** obstacles and other **hardly visible** obstacles
- To correctly detect obstacles of **any size**, a larger number of collision points must be used
- Since people differ in height and body composition, it is necessary to **configure** the **detection threshold** and **device placement** specifically for each subject