Mobile Computing Lab SS22
Final Report

Philip Samer

July 12, 2022

1 App Description

The idea was to design an app for visually impaired people, to help them move around without pumping into objects. The app uses the main camera of the smartphone to estimate the distance to objects and tells the user if something is in front of him. The phone is placed in the front pocket of the shirt or hold in one hand (which makes it easier to check different directions). For the communicate with the user headphones are used.

The UI of the app is kept simple to avoid troubles a visually impaired user might have using it. In fact the app should work without any additional input of the user after starting it. For the prototype of the app a startscreen is used to make some adjustment but this could be skipped if the parameter are final. And for testing purpose the camera image or the estimated depth image is shown on the screen (which can be switch between each other with the "Switch" button).

If something is in the way the user gets notified with a short audio signal. The signal is a short beep ton which is played only on the left or right side of the headphone depending where the object is, or on both sides if something is directly in front.

2 Related Research

2.1 Related Apps

There are different apps which are made to assist visually impaired people. One example would be the app from this paper [FKV18]. It uses artificial intelligence and machine learning to recognise objects or text in a photo, to tell the user what is in the photo. But this doesn't help the user to move around an example which goes in this direction would be the app "Be My Eyes".

The idea of "Be My Eyes" is, to allow volunteers to guide blind users through a audio and video connection. This means that this app can be use in basically every situation where the user has a internet connection. But a survey shows that the app is normally not used for navigation [AWBH16]. In my opinion the biggest problem with this approach is the requirement of a volunteer. This is the reason the user of my app should not need any help from others.

2.2 Depth Estimation

To help navigating the user, it is essential to get a depth information of the camera images. There are different possibilities to get such information from a single camera. For my approach I decided to use monocular depth estimation based on deep learning.

There are already many different papers which deal with such an estimation. One of the latest paper is [YZW+20]. The results of the paper show a pretty good estimation and are the reason I wanted to try this approach with my app.

3 Algorithms and models

For my app I used the midas_v2_1_small module [1]. The module is based on a Pytorch version of MiDaS, which performs efficiently and with very high accuracy to compute depth from a single image. The input of the module is a RGB image with a size of 256 pixels by 256 pixels and the output is a depth map of the same size. For the correct usage of the module, some code of the repository from Shubham Panchal was used[2].

A problem with this module was the low frame rate. With my phone the frame rate was about 2 frames per second. So to detect an obstacle a rather simple approach was used. The depth map is split in three parts (left, mid, right) and the pixels which are smaller (=distance) than a threshold are counted. If the number of pixels which are counted in one part are then bigger then a second threshold, a obstacle is detected. This approach is certainly not the best and it would be better to use a second neural network but the already slow frame rate and the lake of Data to train the model was the reason it wasn't done in this way. Depending if an obstacle is detected, a beep sound is then played on one side or on both. This is done by playing three different sound files. One file has the beep sound on both sides and the other two files have the beep sound only on one side.

4 Performance

The performance is not that great. The main problem is the slow frame rate which leads to a lagy behavior. Additionally the depth estimation works fine but if an object is to close to the camera the depth map gets pretty bad. On the other hand it shows that such an approach is promising and works in general. Especially in outdoor environment it showed a better behavior which is the main task.

There are some parameter which influence the performance. One is the movement speed of the user, which is mainly cause by the low frame rate. If the user moves to fast the app hasn't time to react. To increase the performance the user should walk slowly and shouldn't shake the phone to much. Another parameter is the camera of the smartphone. Depending on the angle of view of the used camera, the performance can variate. And one of the biggest influences are the used thresholds. While testing, many different values where tried out, but there is still room to improve. The first threshold is for the distance to an obstacle and the second threshold is for the size of the obstacle. The distance range should be between 4 and 14 and the size of obstacle should be around 10000 (pixels). For me the best result was 7 and 12000.

[1] https://tfhub.dev/intel/midas/v2_1_small/1
[2] https://github.com/shubham0204/Realtime_MiDaS_Depth_Estimation_Android

5 App Overview

As already mentioned the app is intentionally kept pretty simple. After Starting the app the following screen is shown.

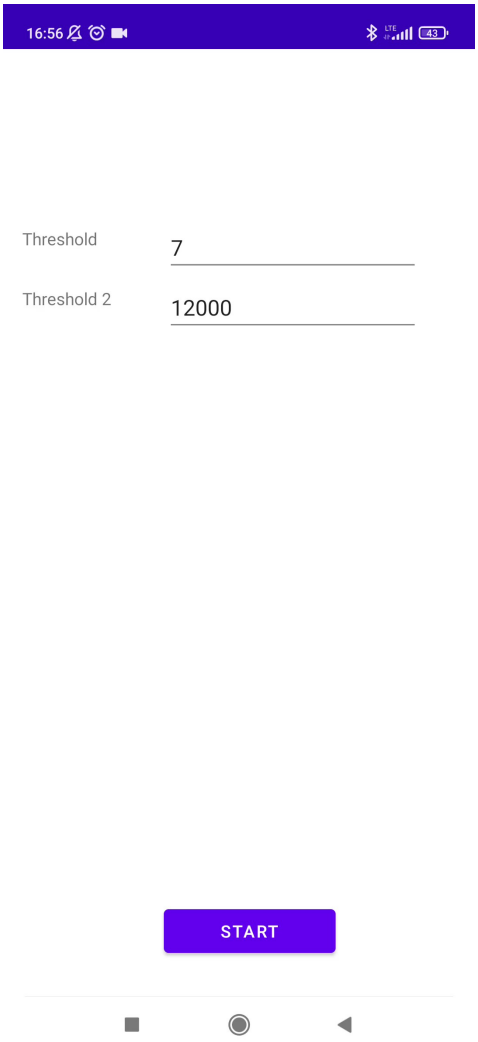

Figure 1: Start window

In the start window the threshold can be changed. And the detection can be started. Again as mentioned at the beginning this is mainly for testing purpose and the app should normally start directly in the detection mode.

After clicking on the "Start" button the following window is shown. In this window we can see the frame rate in the top left corner and the video from the camera. (For the final app version this screen can be empty too)

Figure 2: Detection mode (RGB)

We can switch to the depth image by clicking the "Switch" button. An see the following.

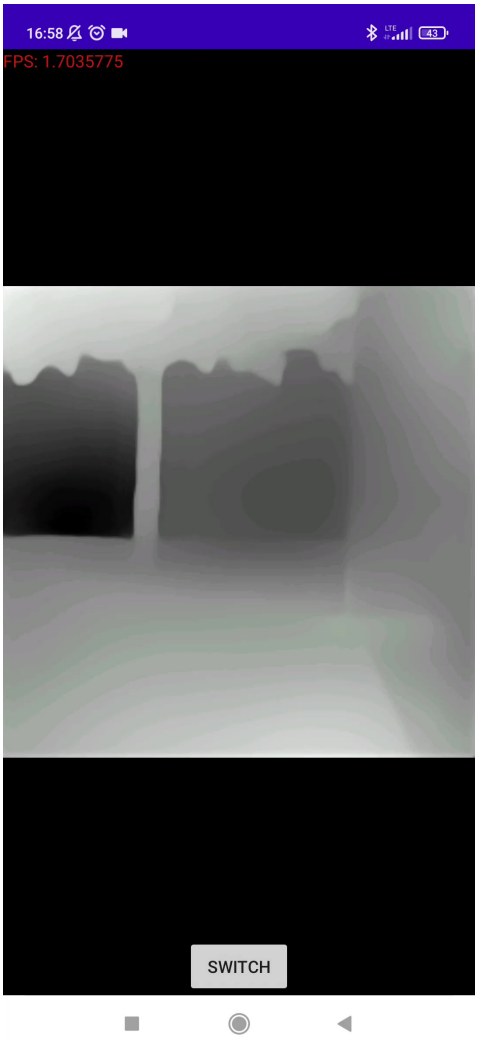

Figure 3: Detection mode (depth)

6 Possible improvements

First of all the frame rate should be improved. This can be presumably done by not using the openCV library (causes many datatype conversions). Secondly for the obstacle detection a neural network should be used. Additionally the audio could be adjusted depending on the distance to the obstacle.

References

[AWBH16] Mauro Avila, Katrin Wolf, Anke Brock, and Niels Henze. Remote assistance for blind users in daily life: A survey about be my eyes. In *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '16, New York, NY, USA, 2016. Association for Computing Machinery.

[FKV18] Shubham Melvin Felix, Sumer Kumar, and A. Veeramuthu. A smart personal ai assistant for visually impaired people. In *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1245–1250, 2018.

[YZW+20] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image, 2020.