EESTECH 2022

Documentation

# In this project we work with BGT60TR13C radar chip.

# We learn how to find the difference between moving and static objects, by the data that the radar chip gives us. When the person is classified as static we make analysis over the vital functions of the person by measure breathing as well as heart rate of the person using the cheap raw data.

The intended audience is every person that wants to analyze it’s vital functionality. Mostly it is projected on elderly people to help them.

Table of contents

[Documentation 1](#_Toc102943144)

[Table of contents 1](#_Toc102943145)

[1 Data overlook 2](#_Toc102943146)

[2 Moving / Static classification 3](#_Toc102943147)

[3 Data overlook 4](#_Toc102943148)

[4 Final App 5](#_Toc102943149)

[Revision history 6](#_Toc102943150)

# Data overlook

Firstly, we made an overlook over the data, that the radar chip delivers us. We measured for 30s three different data sets: static person, static person but further away from the radar and moving person. Our overlook can be found in [data\_overlook.py](https://github.com/ml-lohi/hackathon-private/blob/main/data_overlook.ipynb).

Overall, the major conclusion that we got from there is that the phases of the moving objects are getting a lot higher than the phases of the static objects. Moreover it doesn't make difference which distance is between the objects (the only difference is that less distanced objects are normally more noisy).

# Moving / Static classification

First step is to classify the objects as moving or static.

The detailed overview is provided in the notebook [models.ipynb](https://github.com/ml-lohi/hackathon-private/blob/main/models.ipynb). In the notebook we measured for 100s each of the team members in static as well as in moving state. This delivers us 600s of total data for the analysis.

**Threshold**

The easiest way for the classification, that can our [data\_overlook.ipynb](https://github.com/ml-lohi/hackathon-private/blob/main/data_overlook.ipynb) delivers, is to set the threshold at some specific point.

After doing so we already get very good classification results, approximately 73% accuracy. In real time application it works even without any further analysis good enough.

**Linear Discriminant**

We as well tried to analyze the obtained data with the linear discriminant, provided in the notebook liniar\_discriminant\_analysis.ipynb.

The difference of moving and static object is obvious as discussed (one has higher values that the other). The linear discriminant classifier searches the boundary between the two, moving and static objects. Overall, the result obtained with it is good, but makes slight separations of the data already before the classification. Nevertheless, the results in the notebook are another example, that the data can clearly be separated with a simple threshold.

**Simple dense neural network**

Nevertheless, a simple threshold setting seems to be as a shortcut in the solution. That's why we decided to feed our phases data to the neural network.

Out of the 600s data we created 600 samples, each containing 1000 phases values. By using the network even with one neuron we would get more than 1000 learnable parameters, which is unfeasible for the amount of data that we have. That's why we decided to average the data of every second by a step of 20 and got to the 50 values, that represent every second. This still looked appropriate on the plot and the amount of learnable would be much less.

Nevertheless, we got a very unstable value of accuracy. Of course, the reason for this is because the data is simply not enough.

**1D CNN**

After seeing the problem with amount of learnable parameters we decided to use a 1D CNN. CNN needs a lot less data learnable parameters, because of the sparsity.

The results were a lot better than in the dense neural network, that's why we decided to use both Threshold and CNN classification for the moving/static classification.

# Data overlook

For the vital functionality we analyze breathing as well as heart rate. The detailed analysis is implemented in [fourier.ipynb](https://github.com/ml-lohi/hackathon-private/blob/main/fourier.ipynb)

**Breathing rate**

To find the breathing rate we implemented FFT on the breathing data. We firstly smoothen the data with the gaussian filter, to remove the noise. After applying the low pass filter with 0.1-0.8 we apply the fft. The values 0.2-0.5 correspond to 12-30 breaths per minute, we added and subtracted 0.1 values to the boarders for additional safety. The peak of the fft is set to be the corresponding breathing rate.

**Heart rate**

The same procedure is as well used to find the heart rate, but with a lower sigma for the gaussian filter and high pass filter with boarders 0.7-4.0 (this corresponds 42-240 beats per minute).

**Peak instead FFT**

We as well tried to calculate the heart and breathing rates by simple calculating of the peaks of the smoothened phases data. This turned out as well to be a solid solution.

# Final App

The final state of our project was to develop an app, which can be started, when the radar is connected to the laptop. This laptop plots the phases every seconds, updates the value of the heart rate every 5s and breathing rate every 10-15s.