Individual Report for COMP3516

Wong Chor Sing
u3579076@connect.hku.hk

1 TASK P-1

What I have done (including how to acquire ACF and the shape):

I first divided the frequency and frame length to get the CSI sample per second. I then find the number of samples per ACF window and the step size in the sample.

After setting up all the essential parameters, I initialized the time points for ACF computation and also the lag points for ACF, I also initialized the ACF matrix with zeros with the appropriate dimension.

I copy the function from the tutorial for the ACF calculation, but discovered that it did not work. The reason was the absence of normalization. I fixed it in the code but the original code without normalization was commented. Since the issue was discovered before the TA gave hint on normalization, my implementation is different from the one provided by TA.

The shape of the resultant ACF is (328,68,511). They correspond to the lag points, time points and also number of carriers respectively.

2 TASK P-2

What I have done (including how to acquire MS and the shape):

This task was relatively simple, motion statistics is simple the first sample in ACF. Therefore, after initializing the motion statistics matrix with zeros. I get the motion statistics of each subcarrier by just self.acf[1, : ,:].

I then proceed to calculate the average motion statistics by taking the mean of them.

3 TASK P-3

Method for combing ACF (including reasons):

Before combing the ACF, I had to first decide the value of the topK, i.e. find out the number of subscarrier I need to combine. I plotted all the 511 ACF graphs of each subcarrier and checked the good results available, After some trial and errors. I decided to use the the top 7 subcarriers.

The way to decide which one are better is by sorting the subcarriers by motion statistics. The combined ACF is simply by taking mean of the topK carriers after apply filters on them.

Peak Detection Design (including reasons):

This is the headache of TASK-3, which is also the part I spent most of my time on. The main problem was that I have to fine tune the parameters of the find_peak function such as distance, prominence and width in order to capture the first peak of some boundary cases. Since the first peak of the combined ACF were not very prominent in some cases or the second peak was with comparable prominence as the first peak.

I use a for loop to find the peaks for each time slice and use the find_peaks function to obtain a array of peaks. I assign the first item of the peaks array as the first peak and append it into an array called peak_index. At the same time, I also append calculated breathing rate(br_t). If I failed to detect a peak, I will append -1 into the peak_index array and the breath rate array will append a 0 to denote not breath rate is detected. The reason to append -1 instead of 0 into the peak_index array is related to part 4. Which will be explain later.

Lastly, I make the br_t into a numpy array and also take average of ti to obtain the average breathing rate.

4 TASK P-4

Approaches to detect presence:

I did some online research as to what kind of parameters i needed to use in order to detect presence[1]. But the resources and information available is limited. Motion statistics and breathing rate were used to detect presence in this case. I obtained the data from the get_br function.

I then proceed to determine presence under two conditions. First, the average motion statiscs should be larger than 0.05 for at least 25% of the time. Second, the detected breathing rate should between 12 and 60 for at least 25% of the time in order to detect presence.

In here if 0 is used instead of -1 to append into the peak_index. The highest point will be picked and the breathing rate calculated will be exceptional larger sometimes. This is a problem caused by my find_peak function design.

REFERENCES

[1] Xiaolu Zeng, Beibei Wang, Chenshu Wu, Sai Deepika Regani, and K. J. Ray Liu. 2022. WiCPD: Wireless Child Presence Detection System for Smart Cars. *IEEE Internet of Things Journal* 9, 24 (2022), 24866–24881. https://doi.org/10.1109/JIOT.2022.3194873