UNSUPERVISED CLASSIFICATION OF PHYSICAL ACTIVITY FROM IMPLANTED ACCELEROMETER IN DEEP BRAIN STIMULATION PATIENTS

A Thesis

Presented to

The Faculty of the Department of Electrical and Computer Engineering

California State University, Los Angeles

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Electrical Engineering

By

Farzana Yasmin Boby

May 2023

The thesis of Farzana Yasmin Boby is approved.

Deborah Won, Committee Chair

Curtis Wang, Committee Member

Charles Liu, Departmental Chair and Committee Member

California State University, Los Angeles

May 2023

ABSTRACT

Unsupervised Classification of Physical Activity from Implanted Accelerometer in Deep Brain Stimulation Patients

By

Farzana Yasmin Boby

An externally worn Apple Watch is being considered for feedback control to develop a closed-loop deep brain stimulation (DBS) system to treat Parkinson's disease (PD). Here, we investigate whether accelerometers provide sufficient information to classify different physical activities and compare performance using the implanted accelerometer versus the Apple Watch. We developed a method to classify physical activity into 3 classes (rest, tremor, or voluntary activity) based on accelerometry. During the controlled validation experiments, our clustering algorithm performed with 91% accuracy with a high-end accelerometer and 89% with the Apple Watch. We applied our validated algorithm to 48 different datasets from 5 participants' acceleration streamed from the implantable pulse generator (IPG) with those based on the Apple Watch during daily living activities. The classification based on the two accelerometer signals matched for 83% of the tremor. For the resting activity, there was 30% mismatch; for the voluntary activity, there was 32% mismatch. Overall, the percentage of activity classified as tremor was higher with the Apple Watch classification than the IPG. On the contrary, the percentage of the rest was higher with the IPG than the Apple Watch. The results point toward feasibility of classifying activity continuously based on data streamed from an implanted accelerometer integrated in the DBS device as well as the benefit to using an implanted accelerometer versus an external accelerometer.

ACKNOWLEDGMENTS

At the very beginning I would like to thank my family for their relentless support both emotionally and financially. I know raising two children after my father's death was difficult for my mom, still she always gives her best to us. Without the support from my husband, it would not be possible for me to study abroad as an international student.

Dr. Won, thank you so much for your mentorship and all the amazing opportunities I got in my higher education because of you. You are one of the best mentors and one of the best souls I have ever met. Thank you for all the support and always inspiring me.

Thanks to my thesis committee, Dr. Wang, and Dr. Liu for willing to help me in my thesis. I would also like to thank all the faculties at Cal State LA for their continuous support to the students.

I want to specially thank Erick for working on this project before me and developing the classification model. Next, I will thank Paris and Calysta for helping me perform the validation experiments related to this project.

Finally, I would like to thank CATSUS and the Office of Graduate Studies for supporting my research. Also, I would like to acknowledge financial support for this work by NSF HRD-1547723 to NSF HRD-2112554 and CSU Instructionally Related Activities Fund.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure

CHAPTER 1

Motivation

1.1 Parkinson’s disease, neurological diseases treated by DBS

Parkinson’s disease (PD) is a progressive neurological disorder characterized by tremor, rigidity, and slowness of movement, and is associated with progressive neuronal loss of the substantia nigra and other brain structures [1]. Parkinson’s disease patients have low brain dopamine concentration. Non-motor symptoms may also arise, including issues with mental health, memory problems, skin problems, low blood pressure, pain, fatigue, speech, and communication issues etc. Parkinson’s disease cannot be cured fully, but medications can help control the symptoms. Traditional PD management primarily focused on dopamine replacement therapies, namely medications such as levodopa which increase the dopamine level of the brain. However, after long-term use, on the order of a decade, most of the patients receive inconsistent therapeutic benefit from the medication in treating symptoms such as bradykinesia and rigidity [2][3]. Long-term levodopa therapy in Parkinson's disease may pose various adverse reactions, such as dyskinesia; wearing-off effect (exposure of Parkinsonian symptoms again as the effect of levodopa diminishes, normally after 3-4 hours after taking levodopa); on-off effect (On time means levodopa is working properly to control symptoms, off time means levodopa is not working properly); mental symptoms such as impaired cognitive function, depression, apathy, sleep disorder etc.; and frozen gait [4]. Deep brain stimulation is then offered to medically refractive PD patients.

Deep brain stimulation (DBS) is an invasive electrical stimulation treatment which has provided remarkable benefits for people with a variety of neurologic conditions. Stimulation of the ventral intermediate nucleus of the thalamus can dramatically relieve tremor associated with essential tremor or Parkinson disease (PD) [5]. Similarly, stimulation of the subthalamic nucleus or the internal segment of the globus pallidus can substantially reduce bradykinesia, rigidity, tremor, and gait difficulties in people with PD [5]. It requires a surgical procedure to implant the stimulating device, called the implantable pulse generator (IPG), the leads, and electrode(s) to deliver chronic electrical stimulation of brain region(s). DBS is used either for therapy for disorders affecting the nervous system, especially movement disorders, and has been approved in most countries for the treatment of Parkinson's disease, essential tremor, and dystonia [6]. Other diseases treated by DBS include epilepsy and obsessive-compulsive disorder (OCD).

The implantable pulse generator (IPG) is the most important component of the DBS system, and it requires programming for setting the stimulation parameters. The amount of stimulation is controlled by the IPG which is placed under the skin in the upper chest wall. IPG sends the stimulation to electrodes implanted in the brain via extension leads, which then deliver the stimulation to the targeted areas.

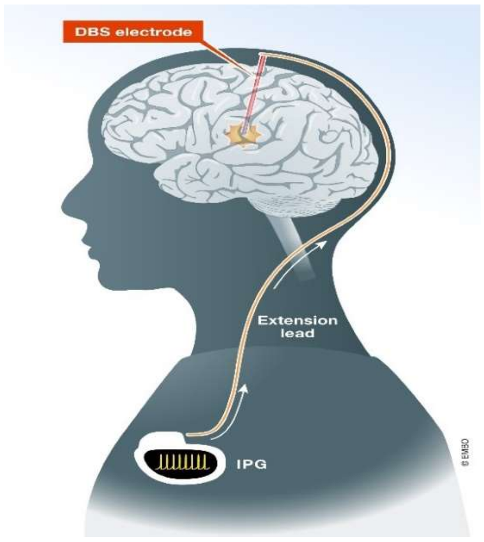


Figure 1. DBS system [6].

1.2 Benefits of DBS therapy

The human brain consists of billions of neurons. These neurons communicate with each other by transferring electric ions. Because of various brain diseases, neurons in different parts of the brain can be less active than other parts. As a result, those brain cells do not work perfectly. Depending on the part of the brain affected, one can have disruptions in the abilities controlled in that area. DBS stimulates those inactive or less active areas of the brain through electrical stimulation, often activating regions which are responsible for inhibiting pathological brain activity. This disruption of pathological neural signals helps manage symptoms of several brain conditions. As a clinical tool, DBS offers several advantages over other surgical approaches for neuromodulation. These advantages include the non-lesional nature of DBS, the capacity to titrate stimulation parameters to maximize benefit and reduce adverse effects and the opportunity to directly interface with the circuit pathology that drives overt symptoms [7].

DBS therapy can help patients with Parkinson’s disease improve their symptoms such as tremors, stiffness, slowness, and dyskinesias. It can also decrease the dose of medication the patient needs to manage their PD. In a pivotal study conducted in 2009 [8], 255 people with advanced Parkinson’s disease were randomized into two treatment arms: 1) DBS or 2) the best alternative care doctors were able to recommend. By 6 months after surgery, patients receiving deep brain stimulation had gained an average of 4.6 hours per day of good symptom control without troubling involuntary movements, called dyskinesia. In contrast, patients receiving standard medical care showed no change, on average, in hourly symptom control.

New features of the implantable pulse generator (IPG) allow fractionation of the electric current into variable proportions between different contacts of the multi-polar lead [9]. Another design consists in leads that allow selective current steering from directionally placed electrode contacts that would deliver the stimulation in a specific direction or even create a directional shaped electric field that would conform to the anatomy of the brain target aimed at, avoiding adjacent structures, and thus avoiding side effects [9].

1.3 Limitations of conventional DBS system

Over 160,000 patients worldwide have undergone DBS for a variety of neurological and non-neurological conditions, with numbers increasing each year [10]. Despite well-developed interface technology, the clinical success of brain stimulation is dependent on variables such as quality of stimulation and exact electrode location [11][12]. Conventional deep brain stimulation instruments use open-loop control. In these systems, stimulation is continuous and parameters such as amplitude, frequency, and duty cycle are fixed. Although these systems are useful and practical in some cases, they have some drawbacks. The brain structures of different people are not quite identical to each other [12]. As a result, using a common program of stimulation in the treatment of different people does not bring the same answer, even in some cases leading to severe complications [13]. In an open-loop DBS system stimulation parameters currently need a lengthy trial and error process and can be set or modified only at the time of clinical visits. Once stimulation parameters are set to a specific setting, how the patient responds with that stimulation is examined. Patients needed to undergo the same process as long as they respond well with the stimulation setting. Therefore, the main limitation of current DBS device is its open-loop control which results in frequent clinic visits for stimulation programming and missing the maximal quality of life improvements that a closed-loop DBS could afford [14].

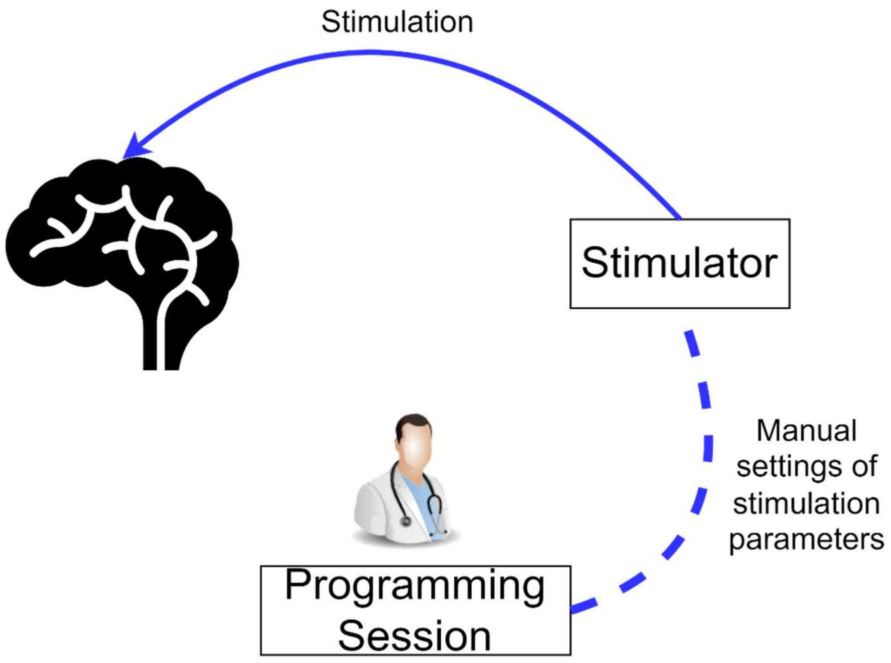


Figure 2. Open loop DBS system.

In the open-loop DBS system, after stimulation amplitude and duration is set once, if the patient needs any adjustment, she or he must wait until the next clinical visit. In reality, patients may need stimulation of more or less amplitude/duration. Adjustments of stimulation parameters are not conducted in real-time based on the ongoing neurophysiological variations in the brain; therefore, adverse effects on the patient may be induced due to brain overstimulation [16].

1.4 Closed loop DBS

Closed-loop DBS systems (also known as adaptive) can be used to solve the mentioned limitations. In these systems, stimulation current can be changed automatically proportional to the recorded brain physiological signals [15]. Closed-loop DBS employs a sensor to record a signal linked to symptoms while open-loop DBS does not use a sensor for recording the brain condition; therefore, stimulation parameters including duration, amplitude, and frequency of the pulse train remain constant in open-loop DBS regardless of fluctuations in the disease state [16]. The recorded signal which contains the information about the disease state is known as a biomarker. In the closed-loop DBS, the stimulation pulses are delivered when the brain is in an abnormal state, or they are automatically and dynamically adjusted based on the variations in the recorded signal over the time [16].

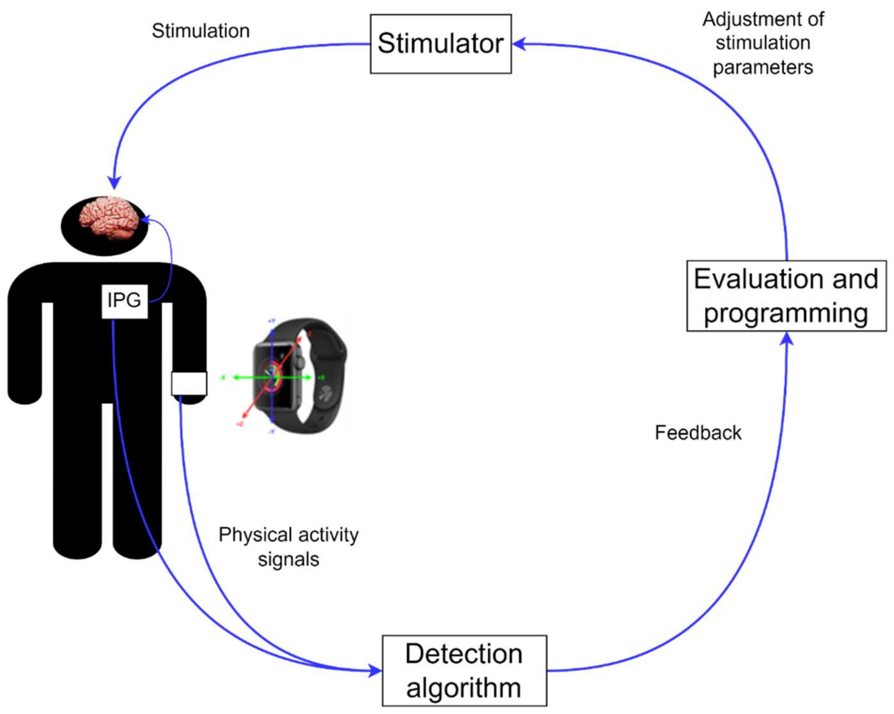

Stimulation
Stimulator
Adjustment of stimulation parameters
IPG
Evaluation and programming
Feedback
Physical activity signals
Detection algorithm

Figure 3. Closed-loop DBS system.

In closed-loop DBS systems, programming of the stimulation parameters is done automatically based on biomarkers. Closed-loop DBS adaptively activates and deactivates stimulation based on brain states. On the contrary, open-loop DBS continues to stimulate regardless of the state of the patient.

To complete the loop of the closed-loop DBS, it is important to find out a potential biomarker for providing the feedback signal. Some examples of electrophysiological biomarkers considered in the feedback loop of adaptive DBS systems are action potentials (APs) [16][17], ECGs [18][19], LFPs [20][21], electroencephalogram (EEGs) [22], electromyogram (EMG) [23] and biochemical [24] signal. Kinematic data [25] such as acceleration signals can also be used as biomarkers. One advantage of using acceleration as a closed-loop feedback signal is the process is non-invasive [25]. Each of these signals needs more research to find out which approach is more feasible, effective, and efficient.

1.5 Thesis objective statement

Medtronic, Inc. (Minneapolis, MN) developed the Summit RC+S, approved for research purposes to aid in the development of closed-loop deep brain stimulation (DBS). The Summit RC+S system consists of two surface or depth leads that are implanted in the brain and a neurostimulator (INS) implanted in the chest. The system is capable of sensing neural activity, acceleration, performing on-board computations, and delivering open-loop or adaptive stimulation based on user-programmed parameters [26]. In this thesis, we analyze acceleration data streamed from Medtronic's RC+S DBS device (intrinsic accelerometry) and externally worn wristwatch (extrinsic accelerometry). We investigate how well accelerometry works as a feedback signal to the adaptive DBS in detecting Parkinson's tremor as well as other daily living activities and compare activity classification results using intrinsic accelerometry with those using the externally worn wristwatch.

Our goal is to detect Parkinson's tremor and other physical activities using acceleration as feedback, so that DBS can be controlled more efficiently and effectively. Parkinson's tremor is considered to happen at 4-7 Hz [27] frequency range. A machine learning algorithm can detect the activity as tremor when the frequency range of the tremor is 4-7 Hz. After tremor is detected, the stimulation parameters could be set automatically to start stimulation at that moment but keep it off during sleep or rest. Similarly, other physical activities such as walking, running, cooking, and so on can also be detected and this information could be used to set the stimulation parameters precisely in real-time.

The objective of this thesis is to demonstrate feasibility of an accelerometer-based physical activity classifier that could be used to close the loop in DBS and determine whether an implanted accelerometer or an externally worn accelerometer can work better in providing a feedback signal to the closed-loop DBS device.

CHAPTER 2

Characteristics and Comparison of Acceleration from Extrinsic Sensor vs Implanted Sensor

In this chapter we are going to discuss the experimental paradigm and data collection process from extrinsic and intrinsic accelerometers. We will be analyzing the correlation between the envelopes of the acceleration signals from both accelerometers by changing the window size of the moving average filter. We will also analyze the spectral power of the acceleration signals by changing the window size used for calculating the Short Time Fourier Transform (STFT).

2.1 Experimental Protocol

A unique aspect of our study was that data was streamed during daily living activities in their everyday life environment, typically from the participants' homes. Data collection was carried out by Dr. Dennis Turner and Dr. Warren Grill's research laboratory at Duke University in accordance with approved protocol from the Duke Institutional Review Board. Six Parkinson's disease patient participants were recruited for this study at Duke University Medical Centre. All 6 participants have severe Parkinson's disease and were eligible for DBS therapy and were implanted with the Medtronic Summit RC+S closed-loop DBS device. This device is developed for research purposes and is not FDA-approved. The RC+S device is capable of both providing stimulation and acquiring acceleration signal, LFP (Local Field Potentials) signal and the stimulation settings. Data was collected from only 5 of the 6 enrolled participants thus far. There was no data available for the sixth subject. We examined 10 datasets from each of the subjects available except subject 4, because there were only 8 overlapping datasets available for subject 4.

The IPG of the RC+S device can record the data while the DBS is on, and data is downloaded from the device by clinicians during the monthly clinical visit; then data is uploaded to Rune Labs' server where it is available to download.

Another way of acquiring the acceleration data is using Apple Watch. All the patients are instructed to wear Apple Watch as often as possible. Apple Watch data is streamed automatically to Rune Labs' server. From Rune Labs' server the data from IPG and Apple Watch is available to download for the researchers, according to the approved IRB protocol.

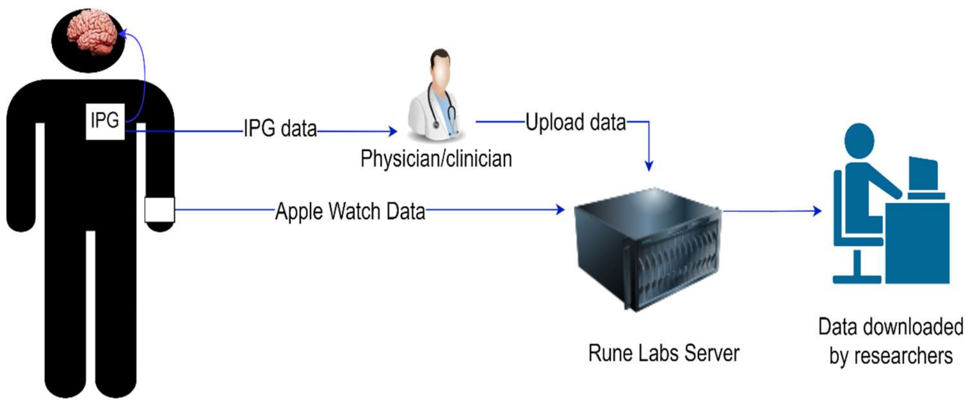
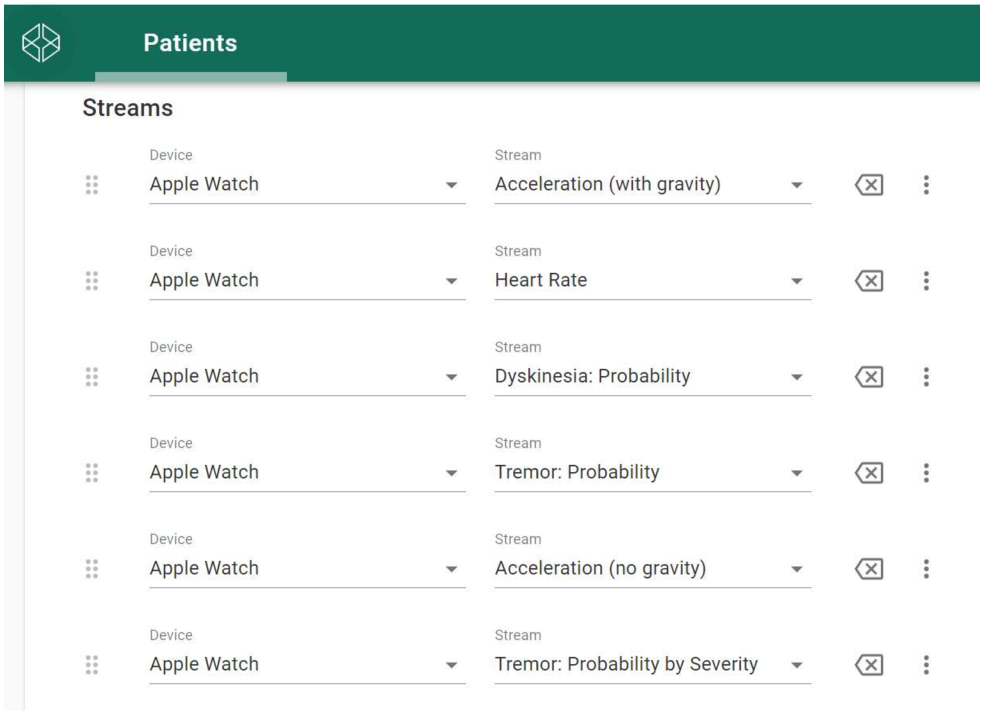


Figure 4. Experimental protocol.

In Rune Labs' web user interface, we can choose the subject, date and time, and the type of data which we want to download. There are several options for Apple Watch data to download, such as: acceleration with and without gravity, heart rate, dyskinesia probability, tremor probability, tremor probability by severity, etc.

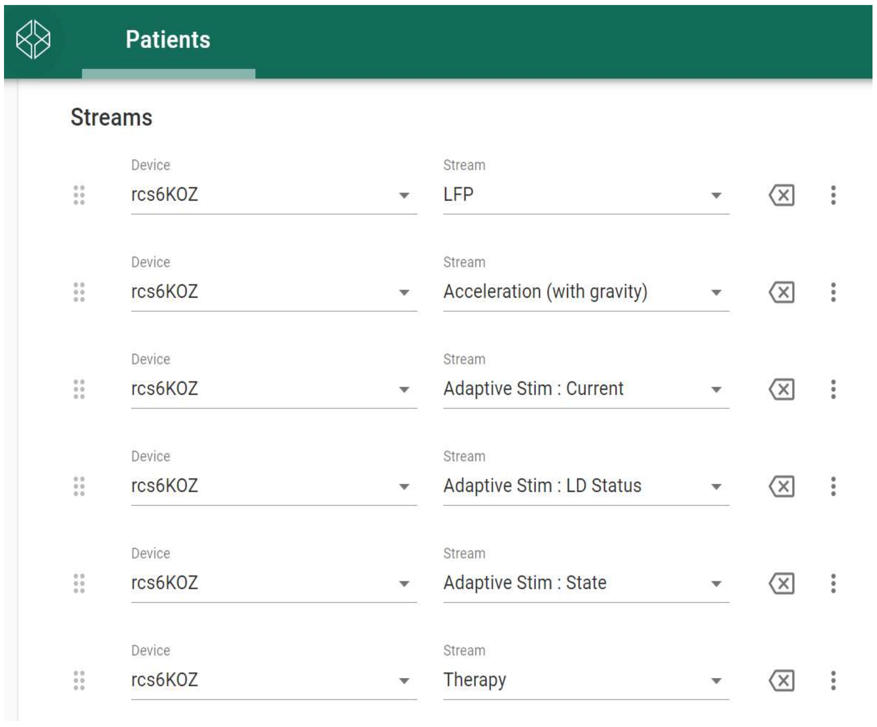

Patients

Streams

Device	Stream
Apple Watch	Acceleration (with gravity)
Apple Watch	Heart Rate
Apple Watch	Dyskinesia: Probability
Apple Watch	Tremor: Probability
Apple Watch	Acceleration (no gravity)
Apple Watch	Tremor: Probability by Severity

Figure 5. Apple Watch available data in Rune Labs.

Similarly, there are several options of data to download from the IPG such as LFPs, acceleration, adaptive stimulation, adaptive stimulation state, therapy etc. In this project we focus only on the acceleration data from both IPG and Apple Watch.


Patients

Streams

Device	Stream
rcs6KOZ	LFP
rcs6KOZ	Acceleration (with gravity)
rcs6KOZ	Adaptive Stim : Current
rcs6KOZ	Adaptive Stim : LD Status
rcs6KOZ	Adaptive Stim : State
rcs6KOZ	Therapy

Figure 6. IPG available data in Rune Labs.

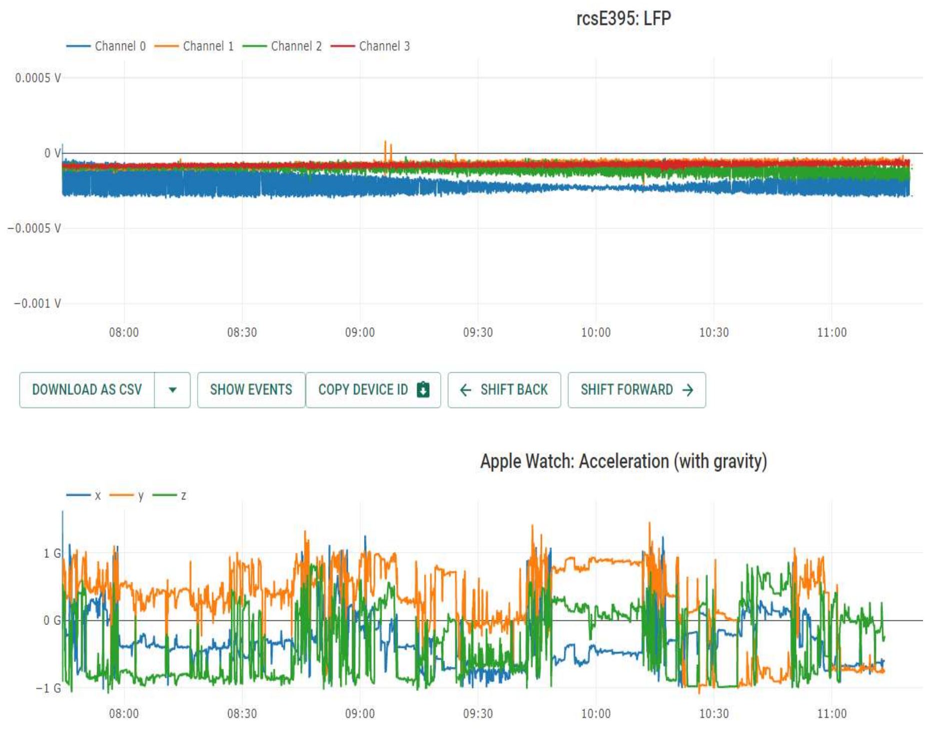


Figure 7. Data visualization in the Rune Labs user interface.

The data can be downloaded using Rune Labs' graphical user interface (GUI) application. For downloading large data which contains hours of data, the file size becomes too large to be downloaded using the GUI. To download larger files, we used Rune Labs' python API (Application Programming Interface), called from a Python notebook. In the Jupyter notebook, the device ID, patient ID, start and end time of the data etc. are specified. The Jupyter notebook file can be found in their GitHub [31]. After downloading the data, we analyzed and processed the data using MATLAB.

2.2 Correlation between Apple Watch and IPG acceleration

For this project we need overlapping data simultaneously streamed from IPG and Apple Watch at the same time. The data from Apple Watch and IPG had different sampling frequencies. Apple Watch had a sampling frequency of 50 Hz and IPG had a sampling frequency of 65 Hz. We performed linear interpolation on the Apple Watch data to synchronize the data to the sampling frequency of IPG data (65 Hz).

After data synchronization, we compared the Apple Watch with the IPG acceleration data by superimposing the two accelerometer signals on each other. Fig. 8 shows overlapping signal content and correlation between the two acceleration signals.

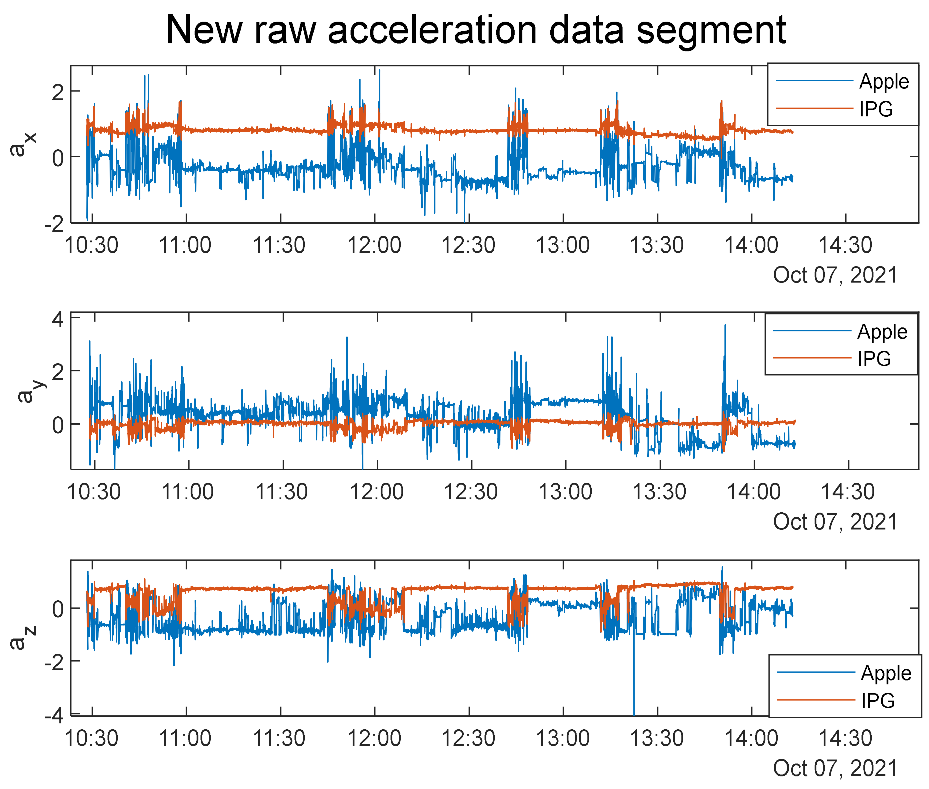


Figure 8. Overlapping acceleration signals for subject 2.

From Fig. 8 we can see that both the acceleration signals have higher intensity activities and lower intensity activities or relatively quiet signal at the same time. At the time when IPG signal has high amplitudes, Apple Watch also has high amplitudes. But, at the time when IPG signal has relatively low amplitude, Apple Watch still shows some high amplitude at the same time. Apple Watch signal is quite noisier than IPG.

To quantify the correlation between the two acceleration signals, we calculated the envelope of each signal and compared the two envelopes. Although the overall activity intensity could be correlated to each other, the high frequency changes of the signals are not expected to be correlated. For getting the envelopes of the signals, we calculated the magnitude of the three axes acceleration signal using the following formula-

$$m(t) = \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2}$$

Where, m(t) is the magnitude of acceleration a, a_x is the acceleration in the x dimension, a_y is the acceleration in the y dimension and a_z is the acceleration in the z dimension. The magnitude was convolved with a rectangular window to obtain the envelope. Different window sizes were used for the rectangular window, namely 10 seconds, 1 minute and 10 minutes to determine the window size for which the envelopes are mostly correlated.

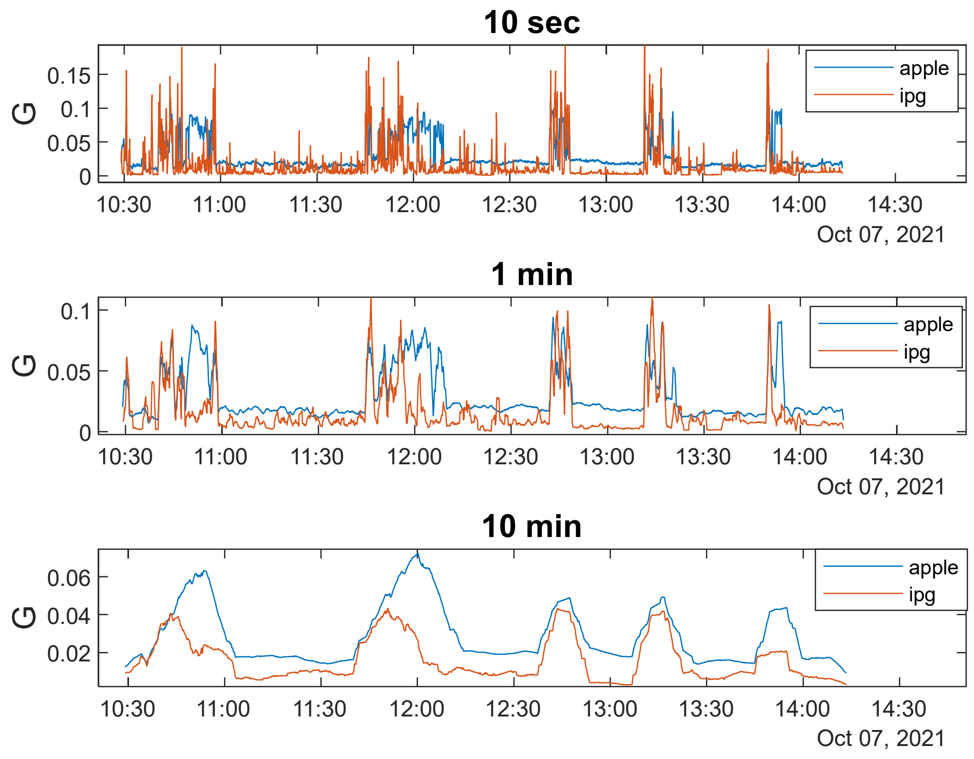


Figure 9. Correlation of acceleration envelopes for different window sizes.

From Fig. 9 visually it looks like a 10-minute window shows the highest correlation of the acceleration envelopes. Then we calculated the Pearson's correlation coefficient using the following formula-

$$r = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^2 \Sigma(y_i - \bar{y})^2}}$$

Where:

r = correlation coefficient

x_i = value of variable x in a sample

$\bar{x}$ = mean of all the values of variable x

y_i = value of variable y in a sample

$\bar{y}$ = mean of all the values of variable y

A Pearson's correlation coefficient value 0 means there is no correlation between the signals, +1 means there is perfect positive linear relationship between the two signals and -1 means there is perfect negative linear relationship. We used MATLAB command $corrcoef$ to calculate the Pearson's correlation coefficient.

Table 1 shows the Pearson's correlation coefficient values for all 10 datasets for subject 2 using 10-second, 1-minute and 10-minute windows.

Table 1. Pearson's correlation coefficient for subject 2.

Dataset	10s	1 min	10 min
1	0.59	0.64	0.89
2	0.42	0.42	0.59
3	0.59	0.67	0.96
4	0.59	0.63	0.74
5	0.5	0.6	0.83
6	0.61	0.72	0.66
7	0.6	0.64	0.79

8	0.41	0.33	0.38
9	0.51	0.71	0.9
10	0.53	0.61	0.72
Average	0.54	0.6	0.75

For most of the datasets, the 10-minute window gave us the highest correlation coefficient values. For subject 2 the average correlation coefficient value was 0.54, 0.6 and 0.75 respectively for 10-second, 1-minute and 10-minute window size. As the window size increased, the correlation between the two acceleration signals also increased. This was true for the other subjects as well. Table 2 shows the average correlation of all five subjects for different window sizes.

Table 2. Average correlation between Apple Watch and IPG acceleration.

	10 sec	1 min	10 min
Subject 1	0.23	0.35	0.57
Subject 2	0.53	0.59	0.75
Subject 3	0.38	0.56	0.59
Subject 4	0.57	0.64	0.77
Subject 5	0.46	0.64	0.81
Average	0.43	0.56	0.70

The average correlation of Apple Watch acceleration with IPG acceleration for among all the datasets was 0.43, 0.56 and 0.7 respectively for 10-second, 1-minute and 10-minute window.

2.3 Spectral analysis of acceleration signals

After analyzing the correlation between the acceleration signals, we analyzed the spectral behavior of the signals to see at which frequencies the activities occurred. The Short Time Fourier Transform (STFT) of the signal was performed using MATLAB command $stft$. After that for visualizing the frequency content coordinated time axis, we used MATLAB command $imagesc$. We used hamming window of 10 second window size with 50% overlap because it gave us the best temporal resolution compared to 20 second, 1 minute and 10 minutes. We got the following plots with different window sizes.

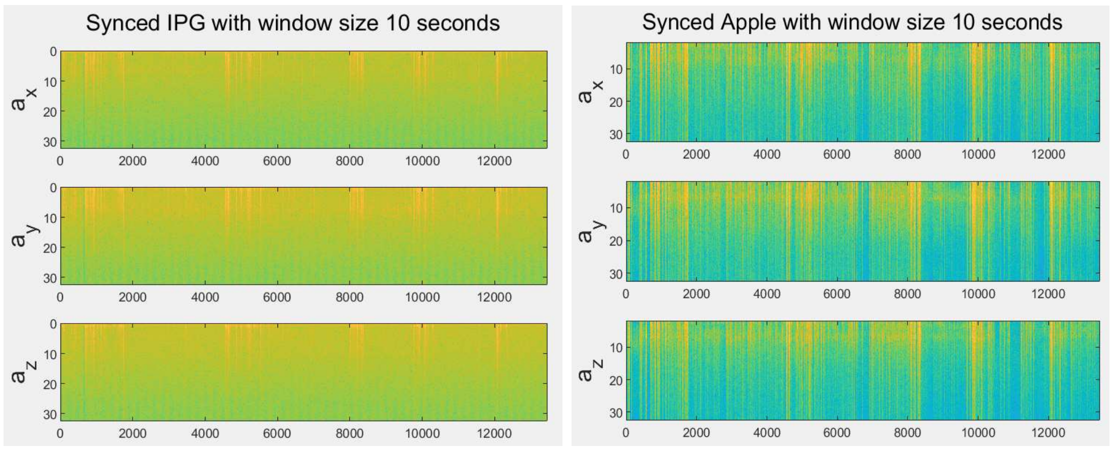


Figure 10. STFT of IPG and Apple Watch acceleration for subject 2 with window size of 10 sec, x-axis is segment windows in seconds and y-axis are the frequencies in which the activities occurred in logarithmic scale.

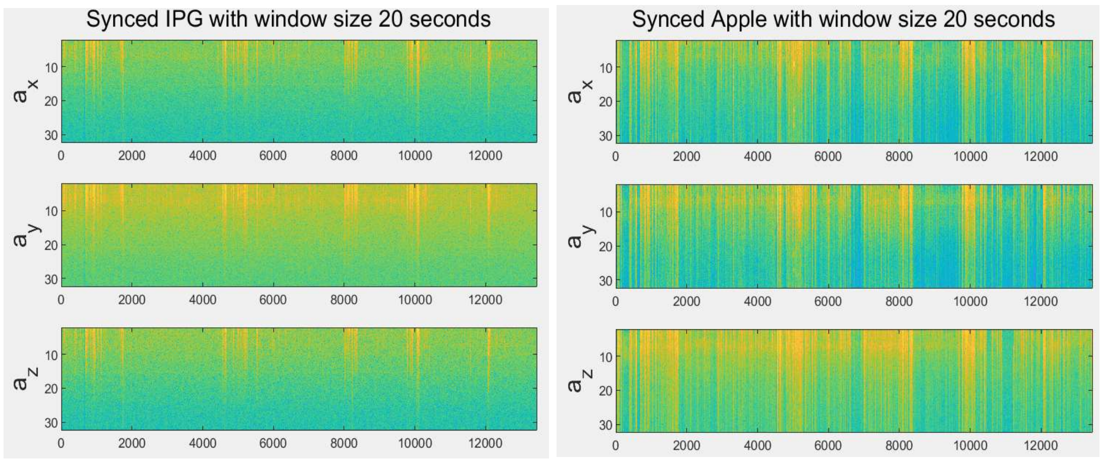


Figure 11. STFT of IPG and Apple Watch acceleration for subject 2 with window size of 20 sec, x-axis is segment windows in seconds and y-axis are the frequencies in which the activities occurred in logarithmic scale.

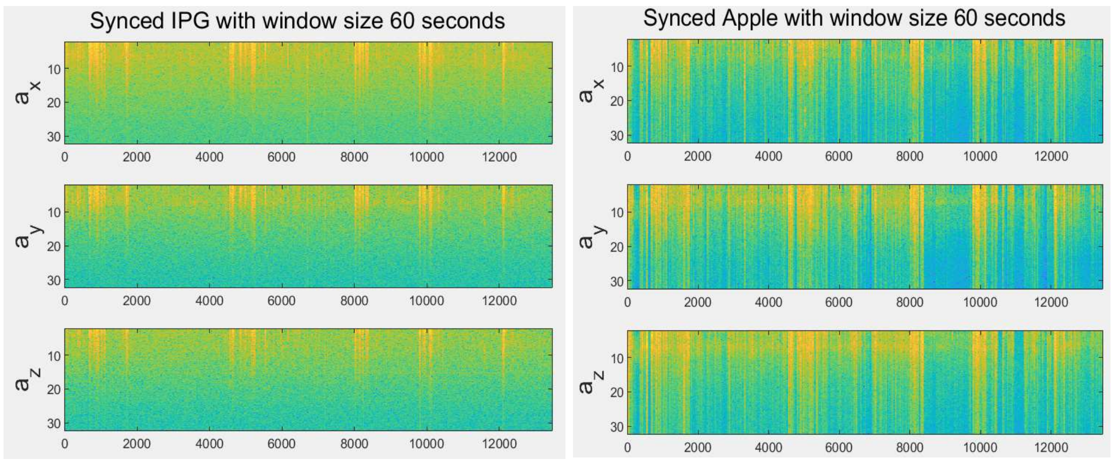


Figure 12. STFT of IPG and Apple Watch acceleration for subject 2 with window size of 60 sec, x-axis is segment windows in seconds and y-axis are the frequencies in which the activities occurred in logarithmic scale.

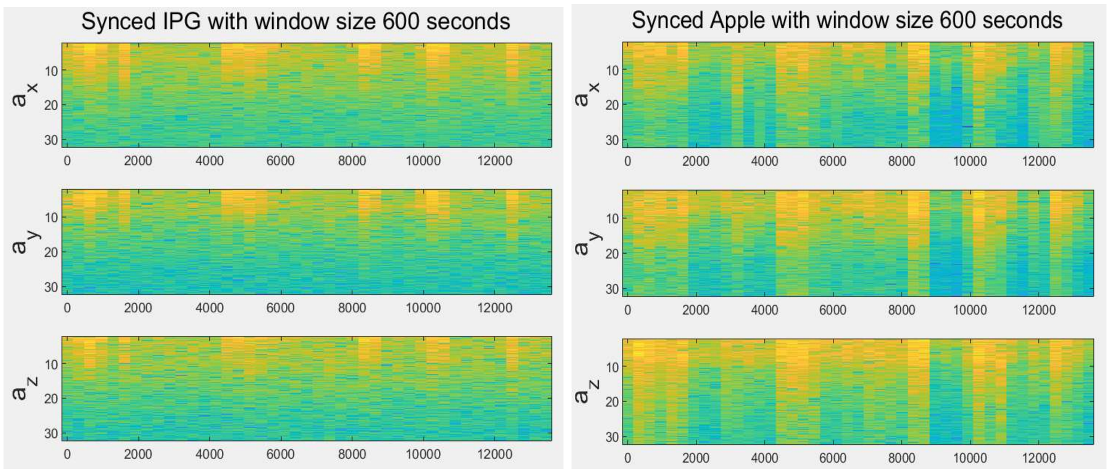


Figure 13. STFT of IPG and Apple Watch acceleration for subject 2 with window size of 600 sec, x-axis is segment windows in seconds and y-axis are the frequencies in which the activities occurred in logarithmic scale.

If we compare Fig. 10-13, we can see that with increasing window size we lose some information with higher window. After comparing the Apple Watch STFT plots with that of IPG STFT plots, we can say that the activities sensed by Apple Watch are more spread with wider frequency range than that of IPG. To obtain better frequency resolution, we chose a 10-second window for our analysis.

Then the integral power was calculated in four different frequency bands from the spectral power. 0-4 Hz band represents the typical normal movements, 4-7 Hz band represents Parkinson's tremor band, 7-12 Hz band represents the physiological tremor band, and 12-32 Hz band represents the beta oscillations which increases as the clinical symptoms worsens.

To find the power in each band at first, we found all the indices in each band (0-4 Hz, 4-7 Hz, 7-12 Hz, 12-32 Hz) and stored the indices of these frequency bands in an array. Then we calculated the absolute value of the spectral power (power we got using $stft$), $S(t, f)$. After that MATLAB command $trapz$ is applied to the absolute value of spectral power across each frequency band to compute power in each band using the following formula.

$$P_{fmin-fmax}(t) = \int_{fmin}^{fmax} S(t,f)df$$

Where, $fmin$ is the minimum frequency in each band and $fmax$ is the maximum frequency in each band.

We stored this result in a 12-dimensional array, where each row indicates integral power in different frequency band in each three axes. Table 3 shows how the integral power is stored in the 12-dimensional array.

Table 3. 12-dimensional integral power in four frequency bands.

	Dimension	Frequency band	Integral power in each column
Row 1	A-x	0-4 Hz	$P_{0\text{-}4,\,start} - P_{0\text{-}4,\,end}$
Row 2		4-7Hz	$P_{4\text{-}7,\,start} - P_{4\text{-}7,\,end}$
Row 3		7-12 Hz	$P_{7\text{-}12,\,start} - P_{7\text{-}12,\,end}$
Row 4		12-32 Hz	$P_{12\text{-}32,\,start} - P_{12\text{-}32,\,end}$
Row 5	A-y	0-4 Hz	$P_{0\text{-}4,\,start} - P_{0\text{-}4,\,end}$
Row 6		4-7Hz	$P_{4\text{-}7,\,start} - P_{4\text{-}7,\,end}$
Row 7		7-12 Hz	$P_{7\text{-}12,\,start} - P_{7\text{-}12,\,end}$
Row 8		12-32 Hz	$P_{12\text{-}32,\,start} - P_{12\text{-}32,\,end}$
Row 9	A-z	0-4 Hz	$P_{0\text{-}4,\,start} - P_{0\text{-}4,\,end}$
Row 10		4-7Hz	$P_{4\text{-}7,\,start} - P_{4\text{-}7,\,end}$
Row 11		7-12 Hz	$P_{7\text{-}12,\,start} - P_{7\text{-}12,\,end}$

Row 12		12-32 Hz	$P_{12\text{-}32,\ start} - P_{12\text{-}32,\ end}$

Here, $P_{fmin\text{-}fmax,\ start} - P_{fmin\text{-}fmax,\ end}$ represents the values in each column in a specific row for each time segment from start to the end.

After getting the integral power for each band, we plotted them. Fig. 14-17 shows the integral power in each band.

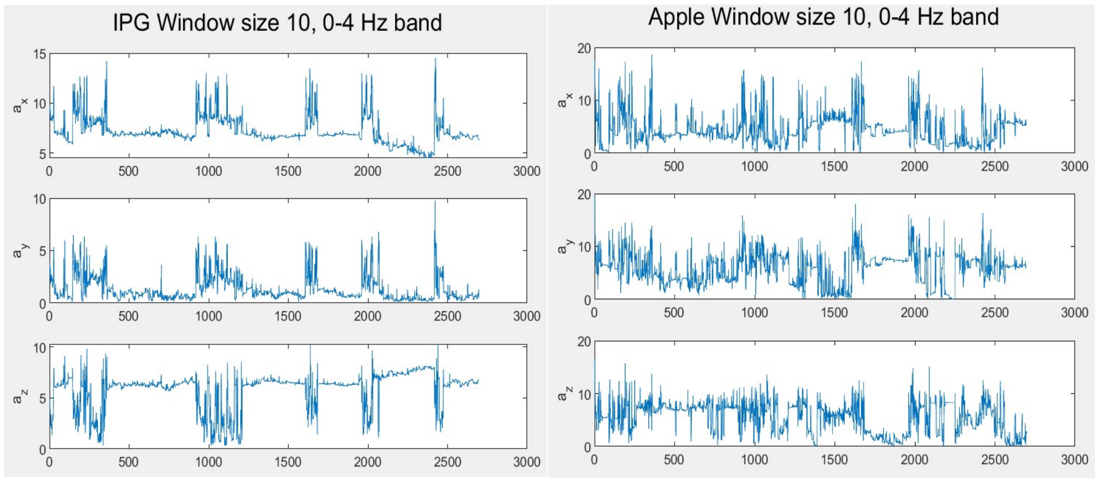


Figure 14. IPG and Apple Watch integral power in 0-4 Hz band for subject 2.

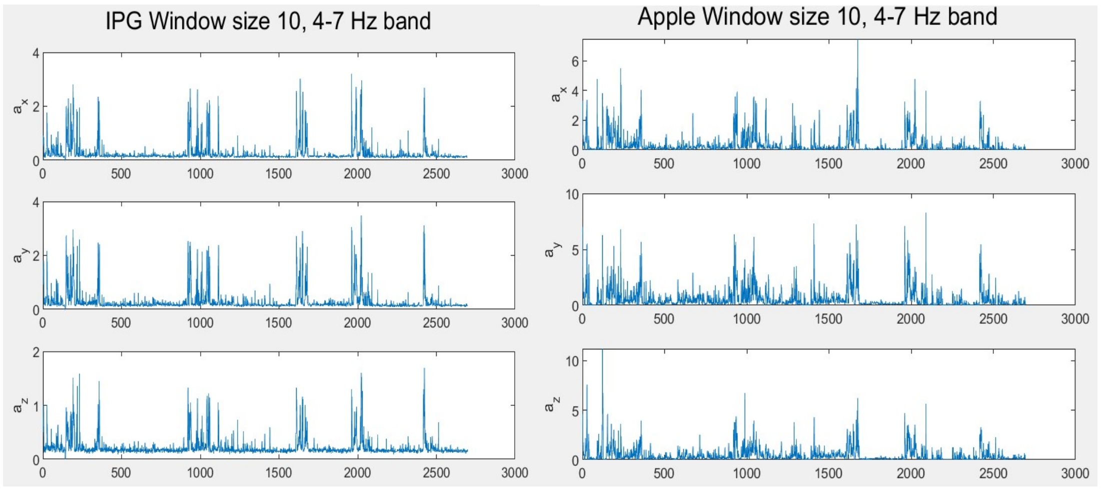


Figure 15. IPG and Apple Watch integral power in 4-7 Hz band for subject 2.

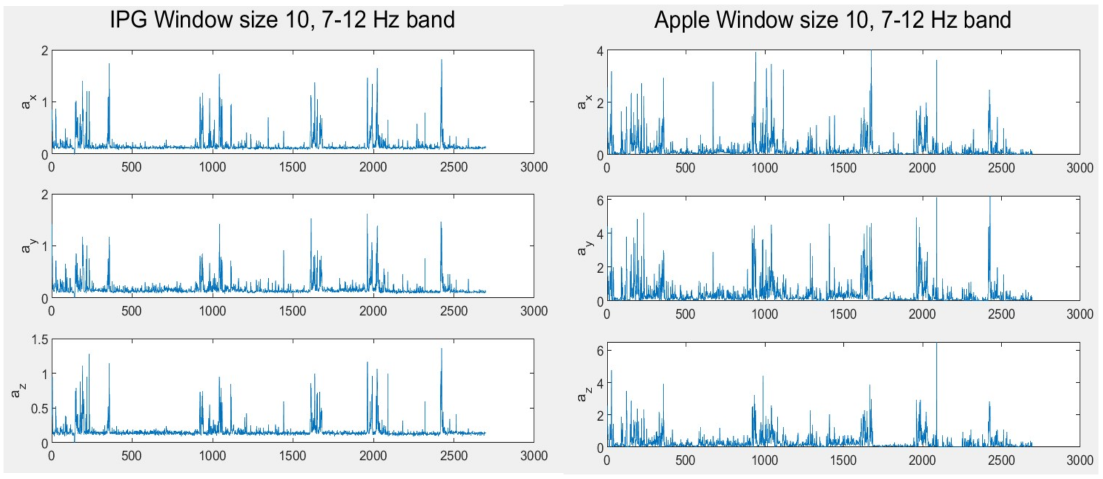


Figure 16. IPG and Apple Watch integral power in 7-12 Hz band for subject 2.

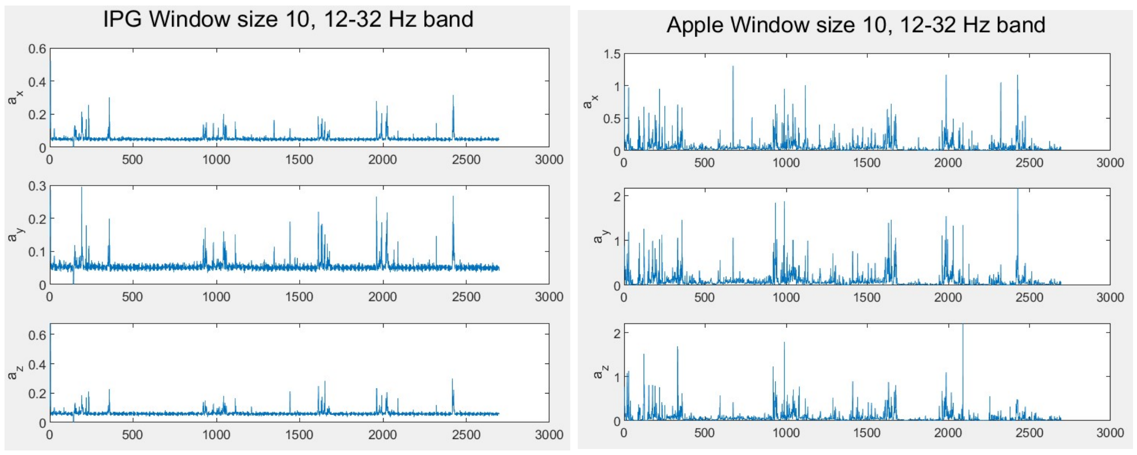


Figure 17. IPG and Apple Watch integral power in 12-32 Hz band for subject 2.

Fig. 14-17 shows that different plots have bursts at the same time we got bursts in the raw acceleration signals. Also, in between bursts there are rest or less amplitude oscillations. Looking at all the plots of all frequency bands it is visible that the 0-4 Hz plot (Fig. 14) has the highest amplitude, which indicates there is more general activity and power in this band for both IPG and Apple Watch. The next highest amplitude can be seen in the 4-7 Hz plots (Fig. 15), which is the Parkinson's tremor band. Apple Watch acceleration has more energy compared to that of IPG acceleration for all frequency bands. In places where IPG integral power seems to be rest (less amplitude variation), Apple Watch has more amplitude variations in the same place. We hypothesize that these amplitude oscillations might be noise. From the integral plot it is clear that IPG has more rest than Apple Watch. Since Apple Watch 4-7 Hz band (Fig. 15) has more power than that of IPG, it is expected that Apple Watch may detect more tremors than IPG. Parkinson's tremor appears mostly in hands. Because of the Apple Watch's location on the wrist, it may also happen that some of the high intensity voluntary active movement can be misclassified as tremor. How we used the integral power in different frequency bands to classify different activities will be discussed in the next chapter.

CHAPTER 3

Clustering Algorithm

3.1 Classification with K-means clustering

Classification is a machine learning algorithm which categorizes a set of data into discrete classes. There are many biomedical applications of classification methods. Classification can be used to diagnose diseases, to detect certain conditions for which alerts, or notifications would be needed, or to determine patient states for administering the appropriate therapy. For example, based on features such as blood glucose concentration, frequency of urination, presence of ketones in the urine, blurred vision, and levels of fatigue, individuals can be classified into two classes i.e., with diabetes and without diabetes. For the purpose of this thesis, we use classification to identify the type of physical activity of a participant at any given time. Knowing their physical activity may potentially allow for adaptive closed-loop control of their deep brain stimulation therapy. To create a successful classification algorithm, it is important to provide identifying feature sets to predict a class or category.

Clustering means classifying data points in feature-space, depending on similarities with or closeness with data points in the same cluster. K-means clustering is a popular clustering algorithm for unsupervised learning. K-means clustering picks out k centroids in an n-dimensional feature-space and assigns each data point to the nearest cluster based on identifying features. This method uses the Euclidean distance formula to determine the distance of a data point from each of the centroids. Then the data point is assigned to the cluster that minimizes the distance between the data point and the cluster's centroid. There are several steps in this algorithm, as outlined here:

Step 1: Selecting a value of *k* i.e., number of clusters and then selecting random centroids

Step 2: Allocating each data points to the nearest cluster by calculating the distance from the centroid

Step 3: Calculating the new centroid of all the data points assigned to each cluster

Step 4: Re-calculating the distances from each data point to every centroid and assigning the data points to the cluster with the nearest centroid

Step 5: Repeating step 3 and 4. If the centroid doesn't change from the previous iteration, these centroids will be the final centroids.

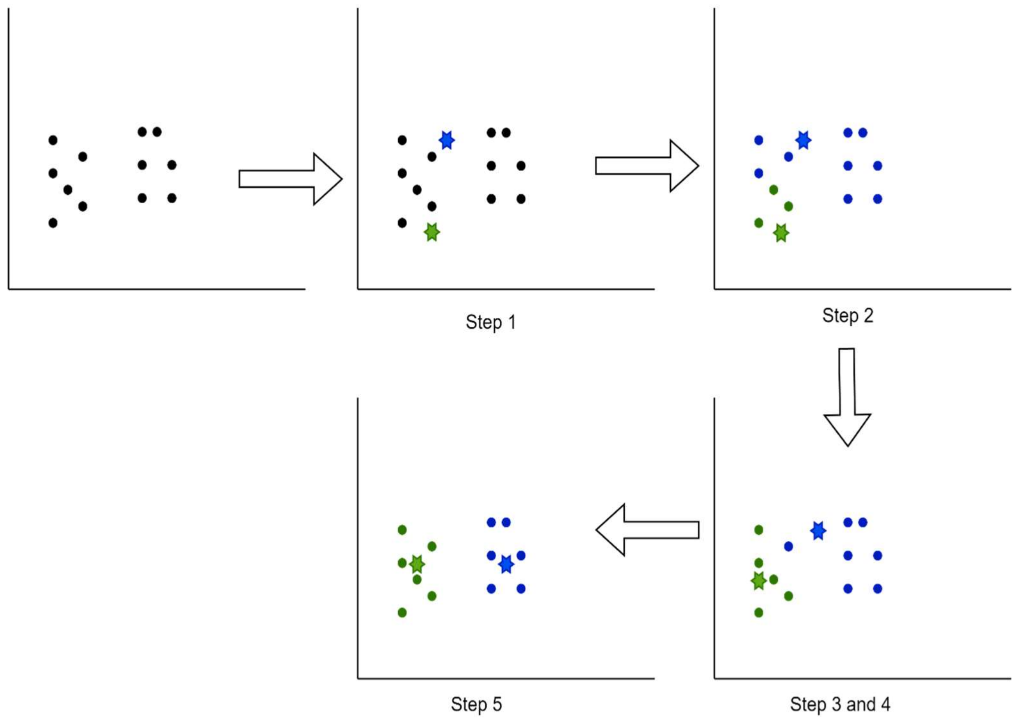


Figure 18. Steps of k-means clustering algorithm.

In this project we used k-means clustering algorithm to classify different physical activities of the patient as well as Parkinson's tremor so that the stimulation parameters of DBS can be automatically set according to the specific need at specific time.

3.2 Feature extraction from acceleration data

Feature extraction is the most important step in any machine learning model. In our case it is important to determine features from the acceleration signal which are distinct for different physical activity states of the patients. We extracted our features from the spectral power in four different frequency bands (0-4 Hz, 4-7 Hz, 7-12 Hz, 12-32 Hz) in each axis of the acceleration [32]. Thus, we extracted 12 total features from the acceleration signal.

We performed Principal Component Analysis (PCA) to visualize the data in feature space. Principal Component Analysis is a dimensionality reduction method in machine learning, used to reduce the number of features to avoid data redundancy. The first step in PCA is normalization. In this step the range of the continuous initial variables are being normalized so that each of them contributes equally. After normalization all the variables will be in the same scale which will prevent biased results. The second step is to compute the covariance matrix. In order to find the correlation among variables it is important to compute the covariance matrix. Covariance matrix can be calculated using the flowing formula.

$$Covariance\ matrix, C = \frac{X^T X}{N}$$

Where:

X = Normalized input matrix

N = Number of observations

The third step is to compute the eigen vectors and eigen values of the covariance matrix to find the principal components. From the rule of eigenvalue and eigenvector we can write the following equation.

$$Cx = \lambda x$$

$$(C - \lambda I)x = 0$$

Where:

C = Covariance matrix

x = Eigen vectors

λ = Eigen values

I = Identity matrix

By solving the above equation, we can get the eigen values and eigen vectors. The eigenvectors corresponding to the eigenvalues represent the principal component vectors. The first principal component vector is the eigenvector associated with the maximum eigenvalue which contains the highest amount of information. Similarly, the eigenvector associated with the second maximum eigenvalue is the second principal component vector and so on.

We used the `pca` command in MATLAB which returned the principal component (PC) coefficients and the PC scores. The PC scores are the representation of the observations in PC space. PC coefficients tell us how much of the original 12 features make up the new PC feature. The first two principal components have the highest variance and thus least redundancy and are typically used to represent the new features in PC space.

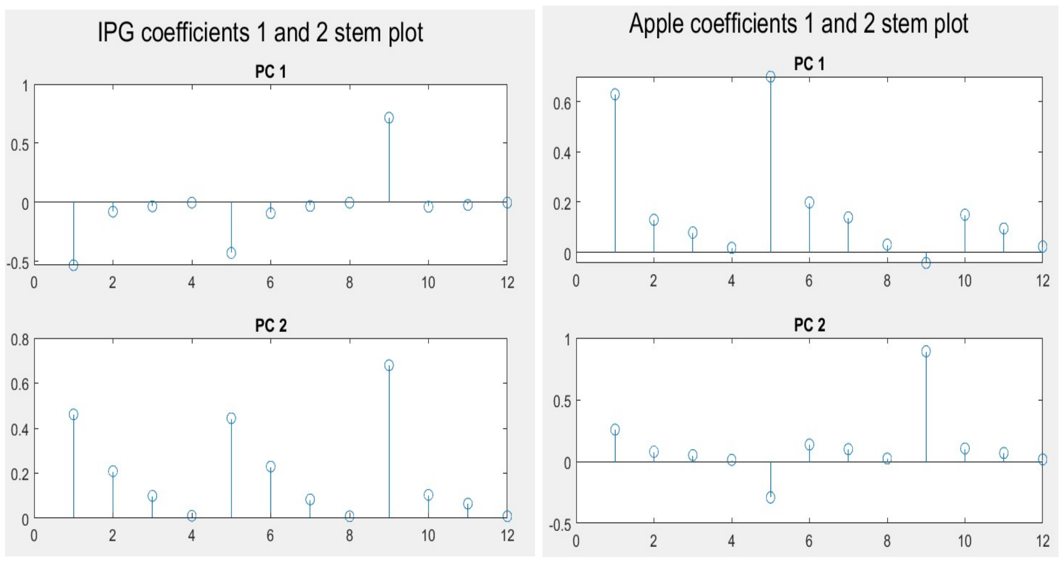


Figure 19. PC 1 and 2 derived from the 12 original features from the 10th dataset for subject 2.

In Fig. 19, index 1 is the x dimension of the 0-4 Hz band. Index 2 is the x dimension of the 4-7 Hz band. Index 3 is the x dimension of the 7-12 Hz band. Index 4 is the x dimension of the 12-32 Hz band. Index 5 is the y dimension of the 0-4 Hz band. Index 6 is the y dimension of the 4-7 Hz band. Index 7 is the y dimension of the 7-12 Hz band. Index 8 is the y dimension of the 12-32 Hz band. Index 9 is the z dimension of the 0-4 Hz band. Index 10 is the z dimension of the 4-7 Hz band. Index 11 is the z dimension of the 7-12 Hz band. Index 12 is the z dimension of the 12-32 Hz band.

From Fig. 19, we can see that for IPG, index 1, 5 and 9 indicates 0-4 Hz band plays most important role in PC1. Index 4, 8 and 12 i.e., 12-32 Hz band has the lowest role in PC1. For PC2 other bands play a significant role. Let us see how the points cluster in the first two PC spaces.

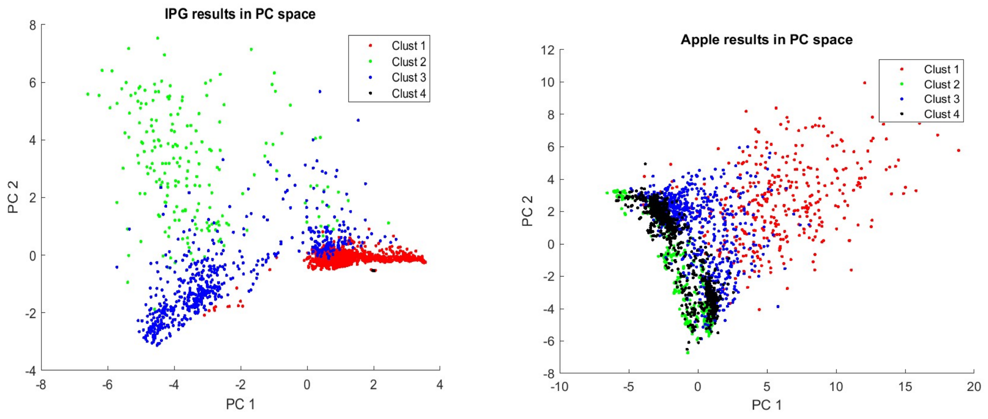


Figure 20. Integral power in PC space for subject 2 dataset 10.

3.3 Elbow method of clustering

A fundamental step of any unsupervised classification algorithm is to determine the optimum number of classes into which data will be clustered. We used the "elbow method" for determining the optimum number of clusters in our clustering algorithm [33]. The elbow method is a popular method of finding the optimum value of *k* in *k*-means clustering. We computed distortion, defined according to the following Equation. Distortion is the average squared distances from the cluster centers of the respective clusters.

$$Distortion = \frac{\sqrt{(c - x_1)^2 + (c - x_2)^2 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + (c - x_N)^2}}{N}$$

Where:

c = Location of cluster center

x_1, x_2…………, x_N = Locations of the corresponding samples

We calculate this distortion metric for varying numbers of clusters from *k* = 1 to *k* = 10. If we plot the distortion array with the clusters, we get the following plots in Fig. 21.

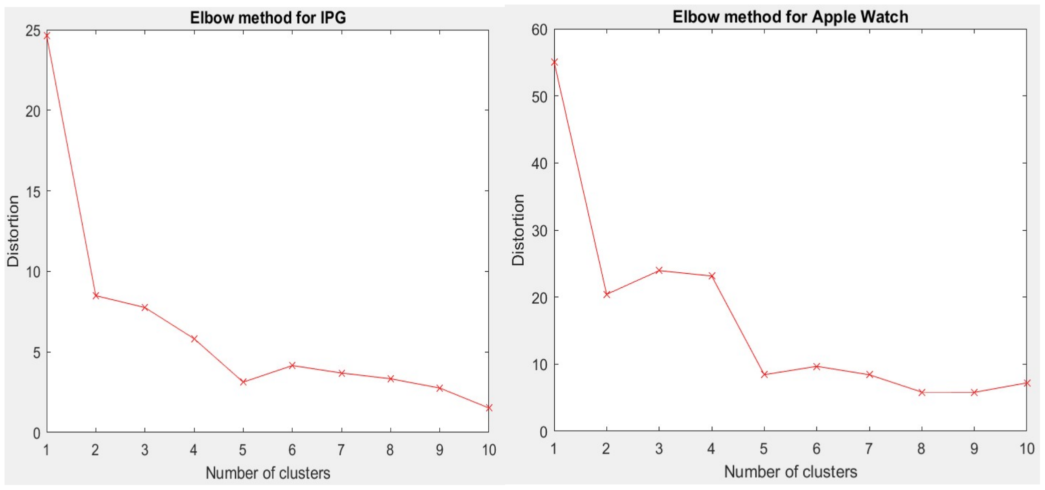


Figure 21. Elbow plot for subject 2 dataset 10.

From this figure, we can see that there is no clear elbow for both IPG and Apple Watch. Then we determined the exponential fit of the curve using the following Equation which is shown in Fig. 22.

$$y_{hat} = e^{c}e^{ax} + B$$

Where:

y_{hat} = Exponential fitted curve

c = Intercept of the linear regression model

a = Slope

x = Number of clusters from 1 to 10

B = Offset

We performed linear regression on the data (x, log(y)). Where:

$$y = Distortion\ array - \min(Distortion\ array) + Offset\ value$$

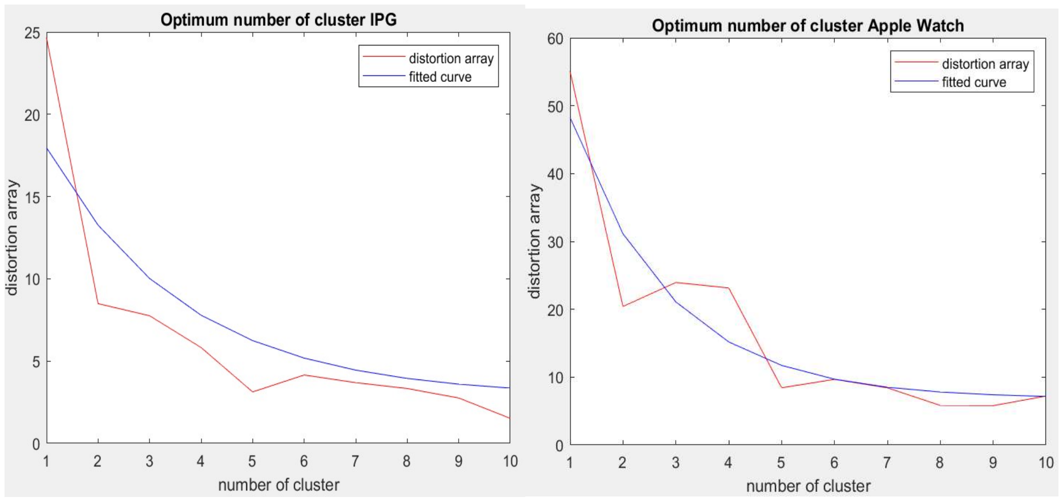


Figure 22. Exponential fit of the distortion array for dataset 10 subject 2.

Instead of manually trying to figure out the number of clusters, we attempted to automate it. We downloaded a function named knee_pt.m by Dmrity Kaplan from MATLAB file exchange which finds a location of "knee" of a curve and provides a consistent and mathematically justifiable answer when there is no obvious location along the curve where the curve "turns". The function uses as a definition of a "knee" the point where the curve is best approximated by a pair of lines. This function applied to our fitted exponential curve returned 4 as the optimal number of clusters for both the IPG and Apple Watch.

3.4 Classifying acceleration data

After getting the optimum number of clusters, we applied k-means clustering algorithm on the integral power in four frequency bands (0-4 Hz, 4-7 Hz, 7-12 Hz, 12-32 Hz) in each axis of the acceleration [32]. Using the MATLAB function `kmeans`, we got an array ipg_idx and apple_idx which gave information regarding which data point consisted of which cluster. For subject 2 dataset 10, we got ipg_idx and apple_idx of size

2694X1. This indicates there are 2694 data points (segment windows) and each of the data points are assigned to either cluster 1,2,3 or 4 based on the features.

To determine the spectral characteristics, we plotted the STFT of each cluster. It appears, as expected, that the spectral characteristics (relative distribution of power across the axes and frequency bands) were consistent within clusters and differ across clusters, as seen in Fig. 3.6.

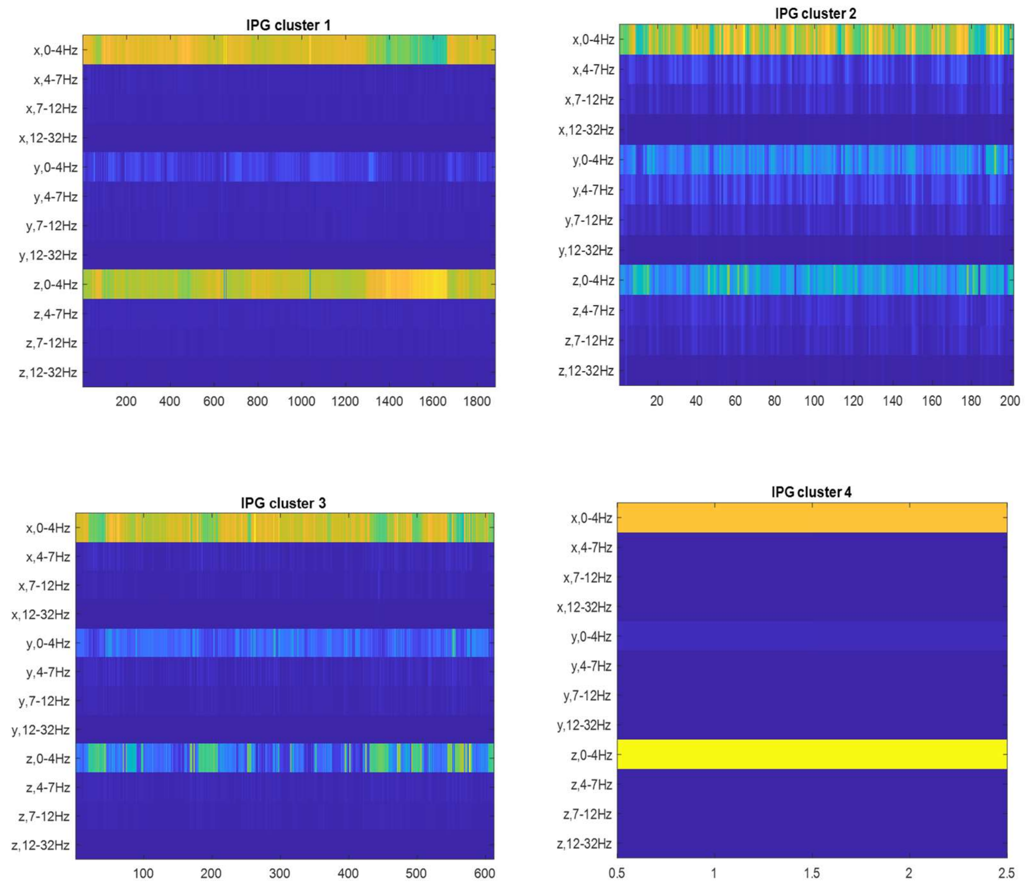


Figure 23. STFT of IPG cluster 1-4 for subject 2 dataset 10.

Starting from top of the plot, row 1 is the x dimension of the 0-4 Hz band, row 2 is the x dimension of the 4-7 Hz band, row 3 is the x dimension of the 7-12 Hz band, row 4 is the x dimension of the 12-32 Hz band. Row 5-8 is the y dimension of the four frequency bands and row 9-12 are the z dimension of the four frequency bands.

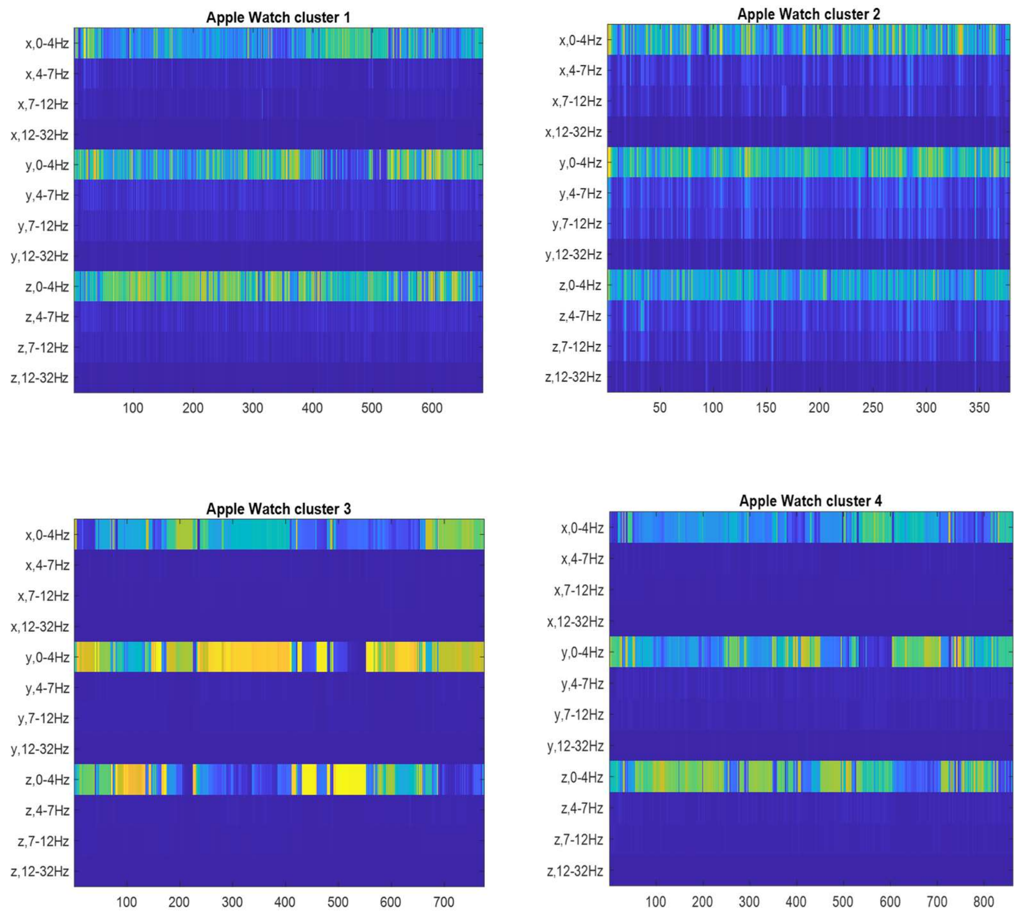


Figure 24. STFT of Apple Watch cluster 1-4 for subject 2 dataset 10.

From the STFTs of the clusters for IPG and Apple Watch, there is more power in the 0-4 Hz band (1st, 5th and 9th row) for both accelerometers. After getting the cluster numbers for each window segment, we can get our classification of the acceleration signal as Fig. 25 and 26. The final classification result is color coded according to the cluster number, cluster 1 = red, cluster 2 = green, cluster 3 = blue, cluster 4 = black, cluster 5 = magenta etc.

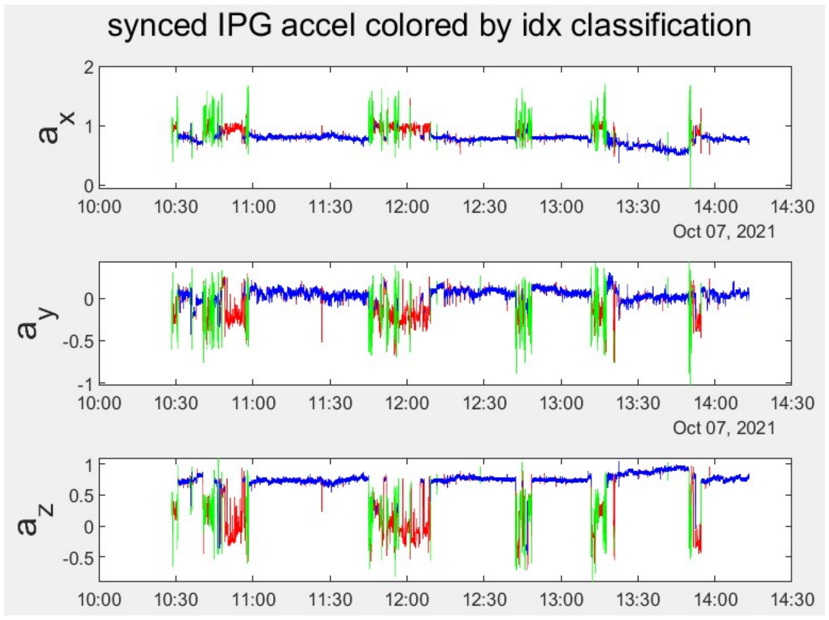


Figure 25. IPG Acceleration color coded classification, cluster 1 = red, cluster 2 = green, cluster 3 = blue.

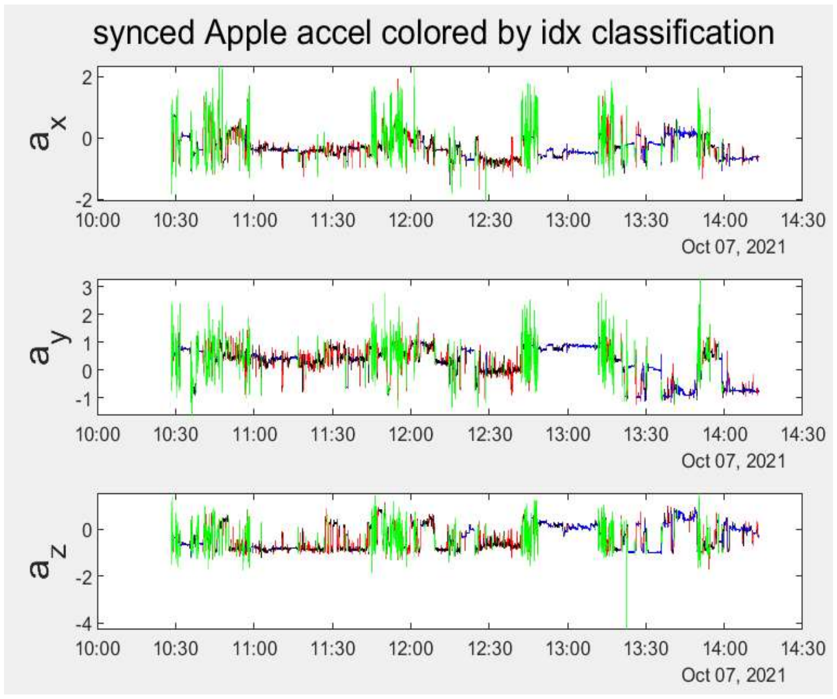


Figure 26. Apple Acceleration color coded classification, cluster 1 = red, cluster 2 = green, cluster 3 = blue, cluster 4 = black.

This is an unsupervised classification where we have no idea about the patients' activities and their timing. So, which cluster indicates which activity must be determined from the spectral power of the clusters and the signal characteristics. We will discuss how to determine which cluster determines which activity in Chapter 5.

CHAPTER 4

Validation Experiment

Data was collected from patient subjects while they were at home, and they were free to do any daily living activities. No information is provided regarding what activities were performed at any given time by the subjects. Thus, having no ground truth data posed a challenge to validating the proposed classification model.

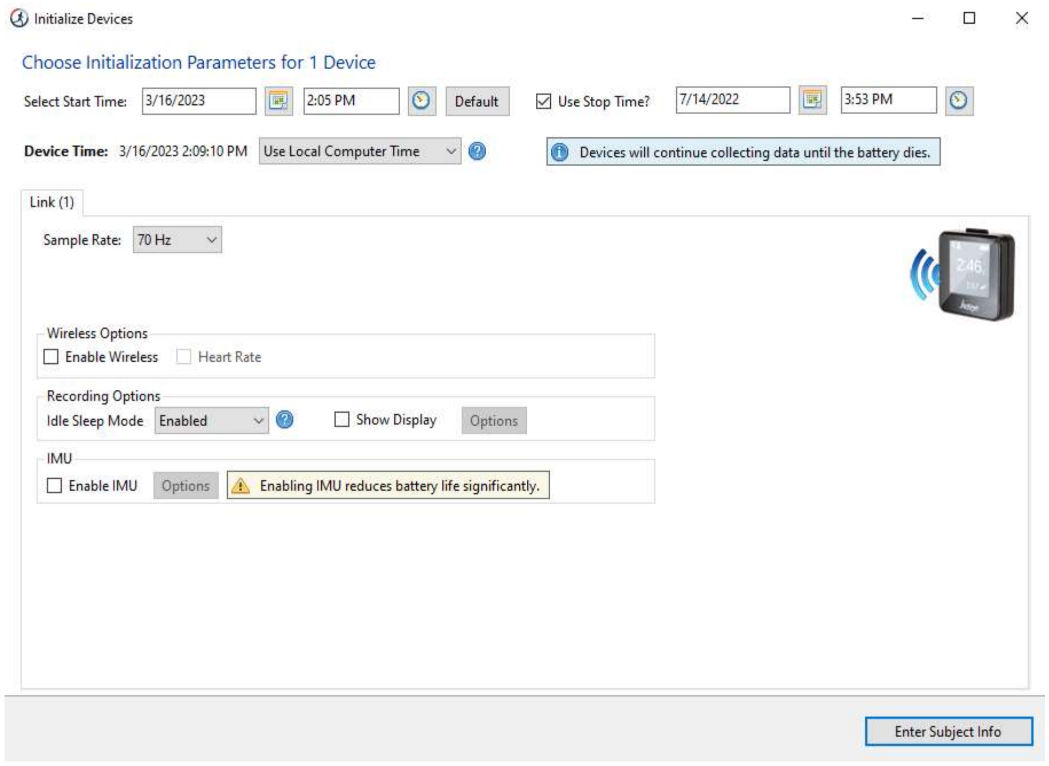
Therefore, we designed and conducted a controlled experiment to validate the classification algorithm. In our experiment, we acquired acceleration data from two commercial accelerometry devices: the ActiGraph GT9X link (ActiGraph Corp., Pensacola, FL), commonly used in research studies on acceleration, and the Apple SE Watch, which is previously was the chosen candidate for providing the accelerometry feedback in the proposed closed-loop DBS system. The subject wore both the ActiGraph GT9X link and Apple Watch on the same wrist and performed predefined activities, which included: 1) tapping fingers on a table at 4-7 Hz with to simulate tremor, 2) laying down, 3) jumping up and down, 4) standing, 5) writing, 6) sitting, 7) moving arm in circular motion to simulate cooking, 8) walking, and 9) typing. The duration, starting and ending time for each activity was recorded. We acquired the 3-axis acceleration data from the Apple Watch using the HemiPhysioData mobile application (HemiPhysio Apps), and data was then transferred to the PC via an iPhone. Acceleration data from ActiGraph was acquired using ActiLife software.

4.1 Actigraph - the hardware, software data acquisition

The ActiGraph GT9X Link captures and records three axes raw acceleration data. The ActiGraph GT9X Link consists of an inertial measurement unit (IMU) which contains a gyroscope, magnetometer, and 3-axis accelerometer, and a programmable LCD display [28].

Figure 27. Actigraph GT9X Link [28].

ActiLife is a software which is used for actigraphy data analysis ActiLife's robust screening and analysis toolkit allows users to extract, process, and score collected data [29]. ActiLife software is a licensed program, and we purchased an ActiLife license. After installing the software, ActiGraph needs to be initialized. For initializing we connected Actigraph to the computer through a USB and initialized ActiGraph. While initializing the device we set the sampling rate to 70 samples/s. After that we set the start time and stop time of the experiment.


Initialize Devices

Choose Initialization Parameters for 1 Device

Select Start Time: 3/16/2023 2:05 PM Default ☑ Use Stop Time? 7/14/2022 3:53 PM

Device Time: 3/16/2023 2:09:10 PM Use Local Computer Time

Devices will continue collecting data until the battery dies.

Link (1)

Sample Rate: 70 Hz

Wireless Options

☐ Enable Wireless ☐ Heart Rate

Recording Options

Idle Sleep Mode Enabled ☐ Show Display Options

IMU

☐ Enable IMU Options Enabling IMU reduces battery life significantly.

Enter Subject Info

Figure 28. ActiGraph initialization with ActiLife software.

Our subject wore ActiGraph on the wrist and performed some predefined activities. After the selected duration was over, ActiGraph automatically stopped recording. Then, the subject took off the ActiGraph from the wrist and set it to the ActiGraph case which is connected to the computer through USB for acquiring the recorded data. After that using ActiLife software we acquired a .csv file which contained 3-axis acceleration data.

4.2 Apple Watch - the hardware, software data acquisition

We used Apple Watch SE for acquiring the 3-axis acceleration signal. Apple Watch has gyroscope and accelerometer which can supply data about the movement in the physical world.

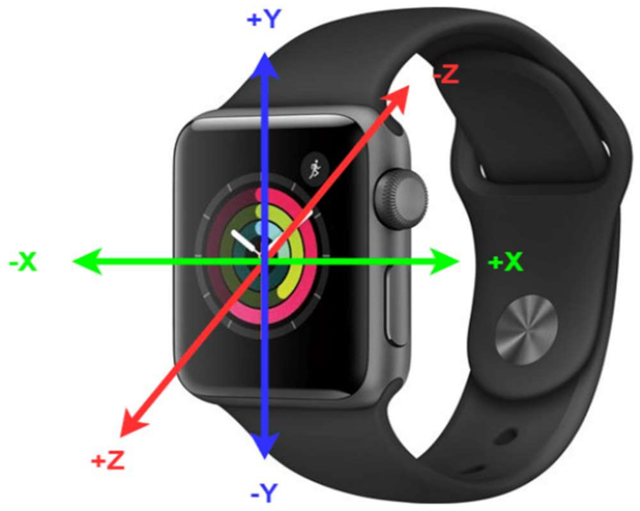


Figure 29. Apple Watch [30].

We used the HemiPhysioData Apple Watch mobile application for acquiring 3-axis acceleration data. HemiPhysioData is an application developed by Moez Ur Rehman for acquiring raw and processed data from built-in Apple Watch sensors. At first, we installed the HemiPhysioData app in Apple iPhone and then installed it in Apple Watch. There are several features to select before recording data such as sampling frequency, dominant side, to which wristwatch was worn, move type, start time, end time, haptic feedback etc. When haptic feedback is enabled it's possible to select the start in time and duration. We can also manually click on start and stop. We clicked start and stopped manually for our experiment. After recording an activity, it was logged in to the Watch. When we clicked on the logged files in the app there was a list of csv files which included the recorded acceleration data. After recording the data, we sent the file from the Apple Watch to the Apple iPhone which is connected to the Watch. Then for processing the data we transferred the data from iPhone to the computer.

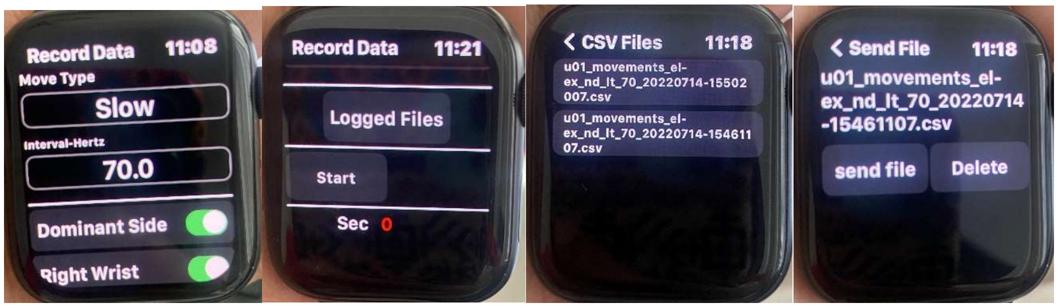
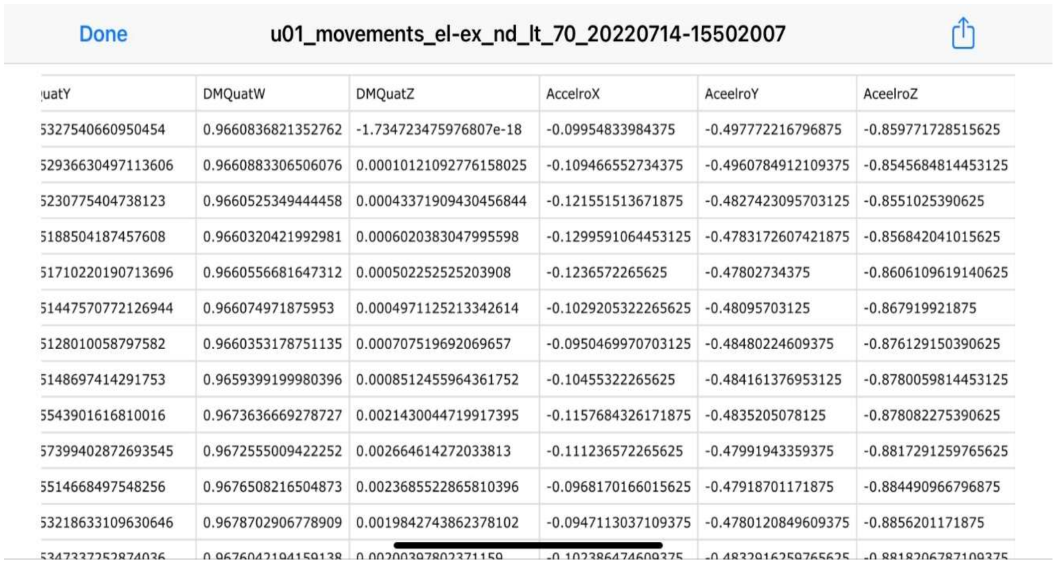


Figure 30. Acceleration data acquisition from Apple Watch.

The Actigraph output processed metrics, such as roll, pitch, yaw, rotation rate is 3 axes, gravity in 3 axes etc. and wrote them to the output .csv file in addition to the raw 3-axis acceleration data (AccelroX, AccelroY, AccelroZ).



Done u01_movements_el-ex_nd_lt_70_20220714-15502007

uatY	DMQuatW	DMQuatZ	AccelroX	AceelroY	AceelroZ
5327540660950454	0.9660836821352762	-1.734723475976807e-18	-0.09954833984375	-0.497772216796875	-0.859771728515625
5293663049711 3606	0.9660883306506076	0.00010121092776158025	-0.109466552734375	-0.4960784912109375	-0.8545684814453125
5230775404738123	0.9660525349444458	0.00043371909430456844	-0.121551513671875	-0.4827423095703125	-0.8551025390625
5188504187457608	0.9660320421992981	0.0006020383047995598	-0.1299591064453125	-0.4783172607421875	-0.856842041015625
51710220190713696	0.9660556681647312	0.000502252525203908	-0.1236572265625	-0.47802734375	-0.8606109619140625
51447570772126944	0.966074971875953	0.0004971125213342614	-0.1029205322265625	-0.48095703125	-0.867919921875
5128010058797582	0.9660353178751135	0.000707519692069657	-0.0950469970703125	-0.48480224609375	-0.876129150390625
5148697414291753	0.9659399199980396	0.0008512455964361752	-0.10455322265625	-0.484161376953125	-0.8780059814453125
5543901616810016	0.9673636669278727	0.0021430044719917395	-0.1157684326171875	-0.4835205078125	-0.878082275390625
57399402872693545	0.9672555009422252	0.002664614272033813	-0.111236572265625	-0.47991943359375	-0.8817291259765625
5514668497548256	0.9676508216504873	0.0023685522865810396	-0.0968170166015625	-0.47918701171875	-0.884490966796875
53218633109630646	0.9678702906778909	0.0019842743862378102	-0.0947113037109375	-0.4780120849609375	-0.8856201171875
5347337252874036	0.9676042194159138	0.002003978023711 59	-0.102386474609375	-0.4832916259765625	-0.8818206787109375

Figure 31. 3-axis acceleration data in the csv file.

After acquiring the csv file into the iPhone, we transferred the data from iPhone to computer and processed the data using MATLAB software.

4.3 Experimental protocol

For validating the classification algorithm, we selected activities which are commonly performed in daily living: lying down; jumping; standing; walking; typing; writing; pretending to cook; and simulated hand tremor. To simulate tremor at the 4-7 Hz frequency, the subject tapped her finger on the table at 4-7 Hz frequency range (at 300-450 beats per minute). The subject wore both the ActiGraph and Apple Watch on the same wrist and performed these activities sequentially according to table 4. The sampling frequency for both ActiGraph and Apple Watch was 70 Hz.

Table 4. Experimental setup.

Activity	Duration (Minutes)
400 BPM tapping	1
350 BPM tapping	1
300 BPM tapping	1
Lying Down	6
Jumping up &down	3
Standing	6
Writing	3
Sitting	6
Pretending to cook	3
Standing	6
Walking	3
Sitting	6
Typing	3

For the first minute, the activity was creating tremor in 6.67 Hz (400 BPM) frequency. The frequency of tremor for the second minute was 5.83 Hz (350 BPM) and finally for the third minute the tremor frequency was 5 HZ (300 BPM). Then other activities were also done according to the sequence given in table 4.

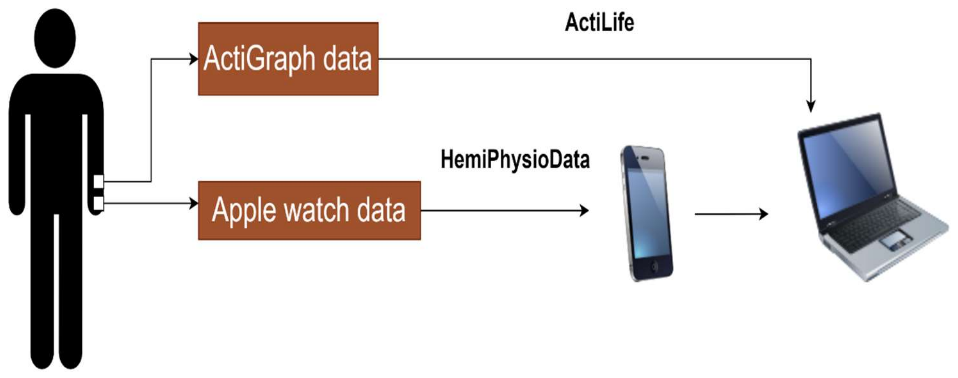


Figure 32. Experimental Protocol.

At first, we ran total 48 minutes of experiment at a time and acquired 48 minutes of data from ActiGraph. We noticed that there were a lot of data packet loss in Apple Watch data acquisition. Because of data packet loss during streaming with the Apple Watch for recordings in the order of an hour, we had to break the experiment into short 3- or 6-minute recordings according to each activity. The data for each activity was written to a separate csv file. Then, the csv files were serially appended to each other according to the sequence of the activities. On the contrary, ActiGraph recording was a continuous 48-minutes recording. For Apple Watch 3- or 6- minute recording we just clicked start and stop button of the HemiPhysioData app to get separate csv files for each activity. Data was recorded at the same time for both the accelerometers. By doing this data loss for Apple Watch decreased, but there was still some data packet loss. Instead of getting 48-minutes of data, we got 30 minutes of data from Apple Watch.

4.4 Results and Analysis

We applied our clustering model to the acquired data to validate the model. Fig. 33 shows the raw acceleration data from ActiGraph and Apple Watch.

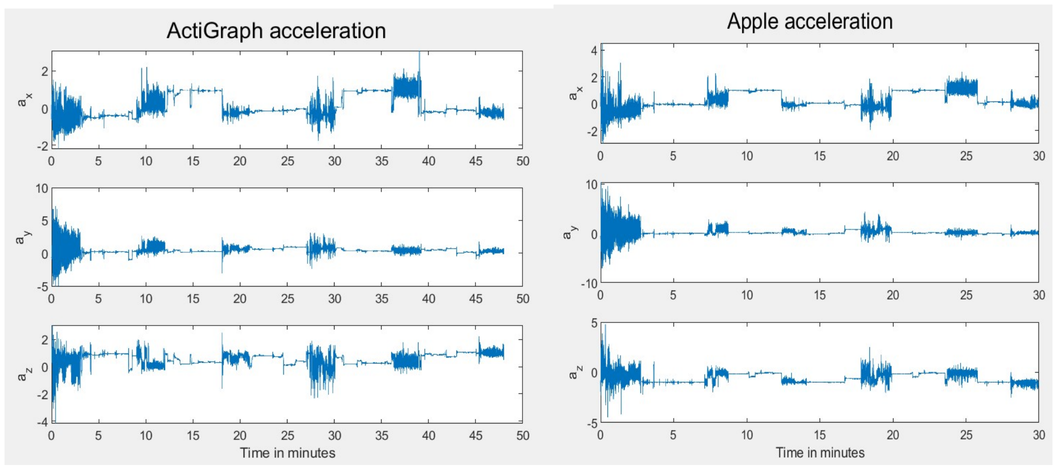


Figure 33. Raw acceleration signals from ActiGraph and Apple Watch.

With a 10-s window and 50% overlap we took the STFT of the signals and extracted the features by integrating the power in four different frequency bands (0-4 Hz, 4-7 Hz, 7-12 Hz and 12-32 Hz). We got 4 numbers of clusters for both the accelerometers using elbow methods of clustering.

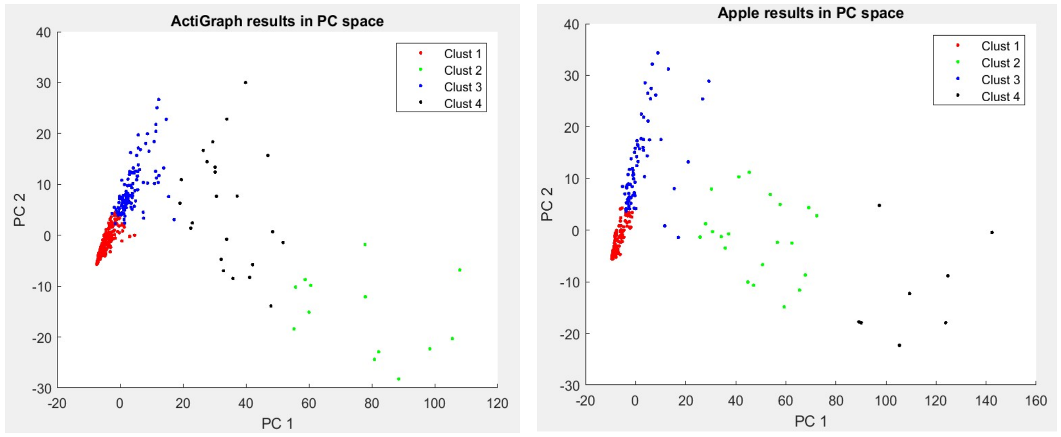


Figure 34. Clusters in PC space for ActiGraph and Apple Watch.

Let us analyze the spectral power for both the accelerometers.

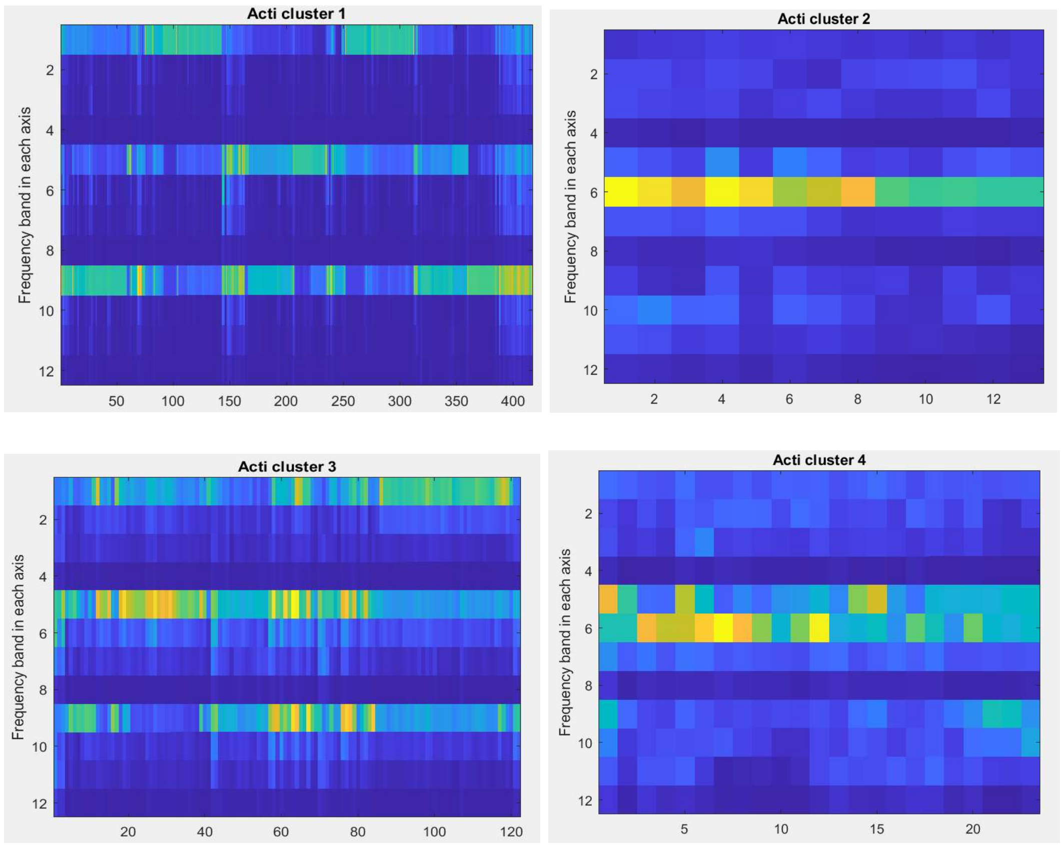


Figure 35. STFT of ActiGraph cluster 1-4.

Starting from the top of the plot, row 1 is the x dimension of the 0-4 Hz band, row 2 is the x dimension of the 4-7 Hz band, row 3 is the x dimension of the 7-12 Hz band, row 4 is the x dimension of the 12-32 Hz band. Row 5-8 is the y dimension of the four frequency bands and row 9-12 are the z dimension of the four frequency bands. Cluster 3's 0-4 Hz band is brighter than that of cluster 1. Therefore, we identified cluster 3 as corresponding to voluntary movement and cluster 4 corresponding to rest activity.

We can see from the 4-cluster plot in Fig. 4.8 that in cluster 2 and 4 the 6th row (y dimension 4-7 Hz) is the brightest of all which means cluster 2 and 4 has the highest power in 4-7 Hz band. Therefore, we determined cluster 2 and 4 to be the tremor class. By analyzing the signal characteristics, we observed that the difference between cluster 2 and cluster 4 is that cluster 2 contains higher frequencies within the tremor band than cluster 4; therefore, we identified cluster 2 as high frequency tremor and cluster 4 as low frequency tremor.

Let’s analyze the spectral power in each cluster for Apple Watch.

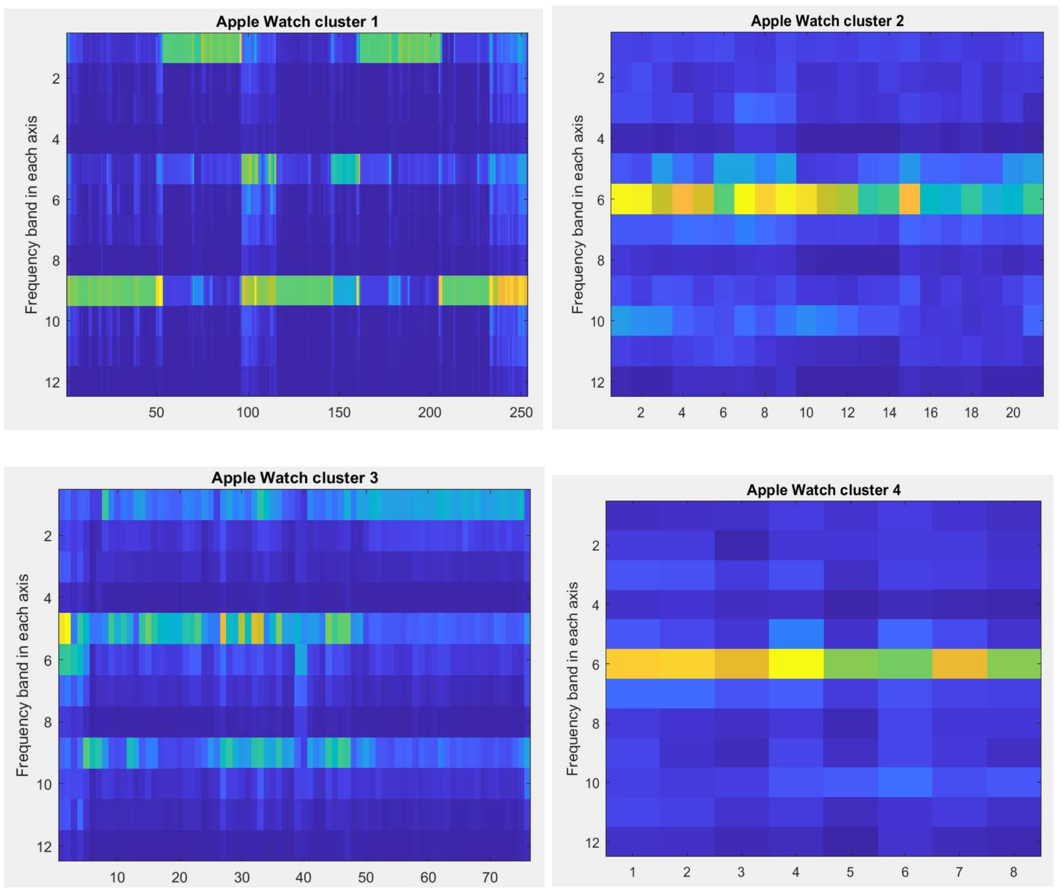


Figure 36. STFT of Apple Watch cluster 1-4.

Apple Walch shows similar characteristics as ActiGraph. Also, for Apple Watch cluster 2 and 4 indicates tremor, cluster 3 indicates voluntary activity and cluster 1 indicates rest.

The classification result for ActiGraph and Apple Watch is shown in Fig. 37.

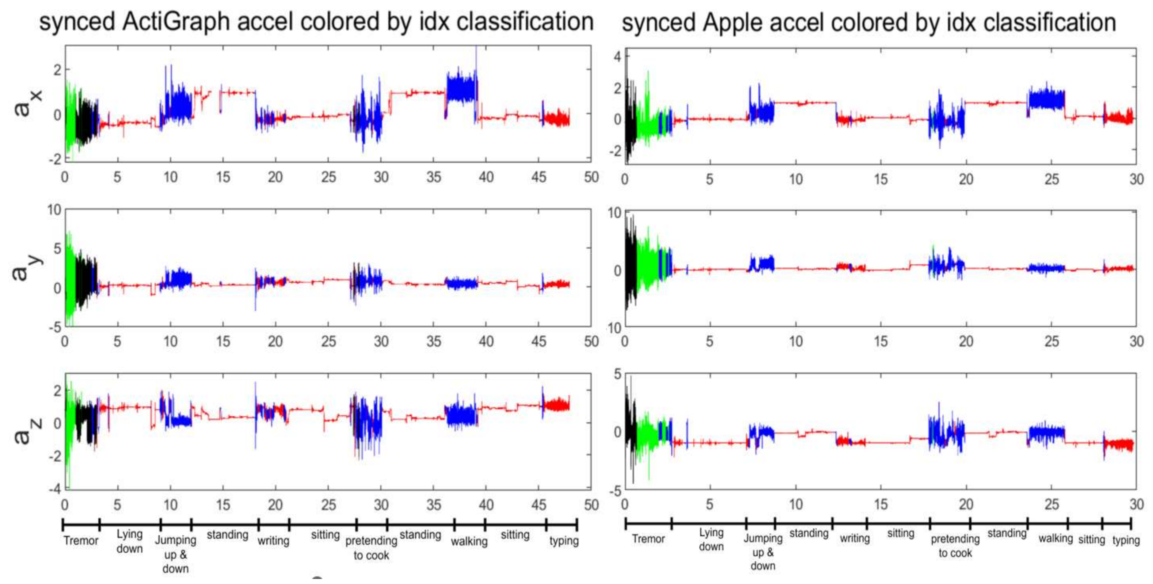


Figure 37. Classification result for ActiGraph and Apple Watch.

Classification results for ActiGraph and Apple Watch look similar. We compared the classification result with the ground truth data to compute the accuracy of our model. We noticed that rest activities, where there was no whole-body movement, were classified as a common category; these included lying down, standing, and sitting. Included in this category were writing and typing, in which the body is also stationary, although they require higher degrees of arm movement. Walking, jumping up and down, and pretending to cook, which all involve voluntary movement of the whole body to a much greater degree, tended to be categorized into a common class. High intensity tremor and lower intensity tremor tended to be categorized each in their own separate class. Therefore, we designated 3 major categories into which data could be classified by our algorithm: 1) rest; 2) voluntary movement; and 3) tremor, which is an involuntary movement that typically does not entail movements of the whole body. We used the elbow method (described in Section 3.3) to determine how many classes the data should be clustered into. The optimal number of clusters was consistently 3 or 4. When the number was determined to be 4 clusters, the tremor category was split into high intensity tremor and low intensity tremor; or sometimes high intensity voluntary movement (walking or jumping) and low intensity voluntary movement (typing and writing and pretending to cook).

Fig. 38 shows the ground truth signal for ActiGraph and Apple Watch where tremor = 1, rest = 2 and voluntary movement = 3. We prepared the ground truth signal with the start time and end time of each activity for both the accelerometers.

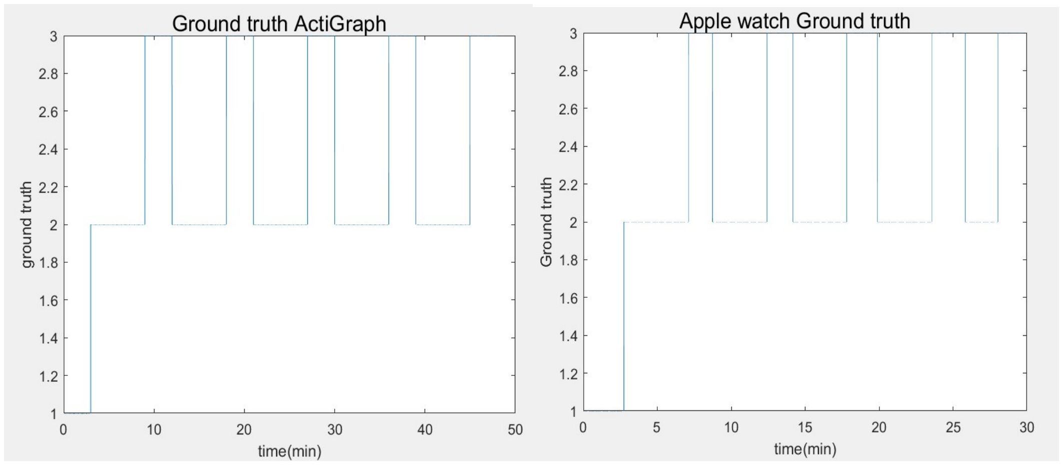


Figure 38. Ground Truth signal for ActiGraph and Apple Watch.

Following the ground truth data, we assigned a value of 1 for clusters which indicate tremor (cluster 2 and 4), value of 2 for cluster 1 which indicates rest and value of 3 which indicates voluntary movement. Then we calculated true positives, true negatives, false positives, and false negatives for both ActiGraph and Apple Watch.

Table 5. ActiGraph confusion matrix.

	TP	FN	FP	TN
Tremor	34	5	3	536
Rest	352	8	65	150
Voluntary Activity	111	68	13	386
Total	497	81	81	1072

Table 6. Apple Watch confusion matrix.

	TP	FN	FP	TN
Tremor	28	5	1	324
Rest	206	6	47	99
Voluntary Activity	111	48	11	234
Total	345	59	59	657

After that we calculated the true positive rate (TP rate), false positive rate (FP rate), and overall accuracy for both ActiGraph and Apple Watch.

$$True\ \ positive\ rate = \frac{Total\ true\ positives}{Total\ true\ positives + Total\ false\ negatives} \times 100\%$$

$$False\ positive\ rate = \frac{Total\ false\ positives}{Total\ false\ positives + Total\ true\ negatives} \times 100\%$$

$$Overall\ accuracy = \frac{Total\ true\ positives + Total\ true\ negatives}{Total\ indices} \times 100\%$$

We got true positive rate and false positive rates for ActiGraph and Apple Watch as follows-

Table 7. True positive and false positive rate of ActiGraph and Apple Watch.

Accelerometer	True positive rate	False positive rate
ActiGraph	86.43%	6.78%
Apple Watch	83.52%	8.24%

Overall accuracy for ActiGraph classification = (497+1072) / (497+81+81+1072) X 100% = 90.64%

Overall accuracy for Apple Watch classification = (345+657) / (345+59+59+657) X 100% = 89.46%

Therefore, we can say that our algorithm is validated to have 91% overall accuracy for ActiGraph with slightly less accuracy with Apple Watch. In a nutshell, all the rest activities were perfectly detected with both ActiGraph and Apple Watch. Tremor activity was also perfectly detected for both devices except some low frequency tremor data points were classified as voluntary movement. Some low voluntary movement such as typing, and writing was misclassified as rest activity for both the accelerometers. Overall, misclassification was slightly more in Apple Watch classification.

4.5 Change of classification performance with changing window size and number of clusters

We have checked the classification performance by changing the window size (10s, 20s and 60s) in calculating the short time Fourier transform and by changing the number of classes in the k-means clustering to be either 3 or 4. Window size and cluster number plays a significant role in the classification performance, especially with the Apple watch data.

Table 8. Comparison of classification performance with changing window size and number of clusters.

Device		10s win, 3 clusters	20s win, 3 clusters	60s win, 3 clusters	10s win, 4 clusters	20s win, 4 clusters	60s win, 4 clusters
ActiGraph	True positive rate	83.65%	85.02%	86.31%	86.43%	88.85%	87.36%
	False positive rate	8.17%	7.50%	6.84%	6.78%	5.58%	6.31%
Apple Watch	True positive rate	74.86%	76.65%	72.41%	83.52%	76.40%	75.86%
	False positive rate	12.57%	11.67%	13.80%	8.24%	11.80%	12.07%

For ActiGraph the highest true positive rate was achieved with 20 sec window and 4 number of clusters, and the True positive rate is 88.85%. On the contrary, Apple Watch True positive rate was highest for 10 sec window and 4 clusters in the classification. The performance was more consistent using the ActiGraph data and consistently higher than the results when using the Apple Watch data. For both cases, 4 clusters yielded greater accuracy. With three clusters, for both the devices, around 40% of tremor activity was detected as voluntary active movement. Also, some voluntary active movement is misclassified as rest. With 4 clusters, most of the tremor activity was classified perfectly.

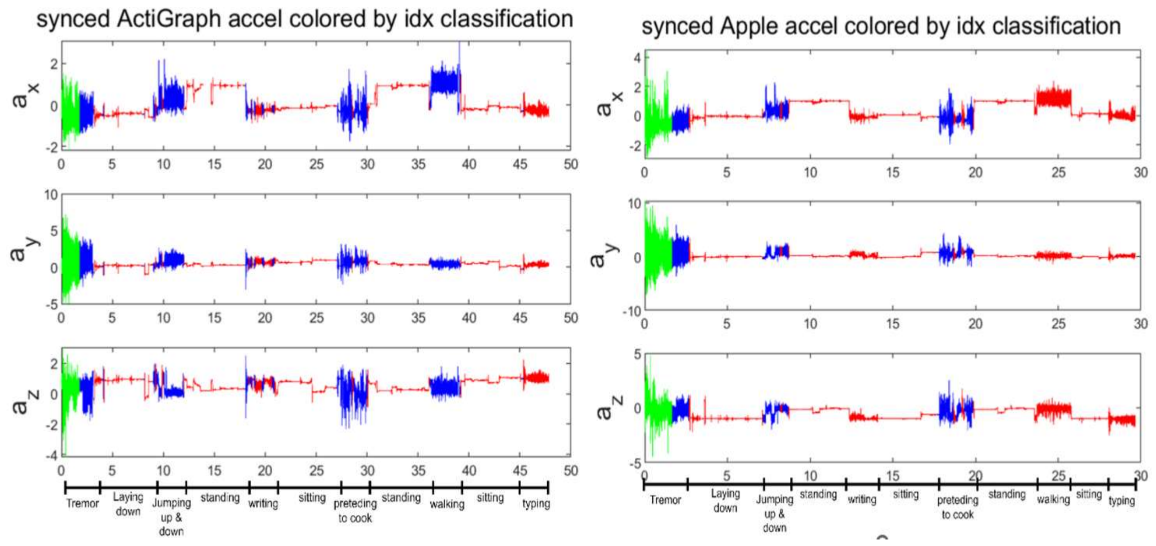


Figure 39. Classification result of ActiGraph Vs Apple Watch with 3 numbers of clusters, green = tremor, red = rest, blue = voluntary activity.

With the Apple Watch data, walking was misclassified as rest for all window sizes and cluster numbers except 10-second window with 4 clusters. We hypothesize that smaller window sizes yielded greater accuracy because it provided better frequency resolution.

CHAPTER 5

Application of Clustering to Patient at Home Data

5.1 Calculating percentage of each activity and percentage of classification match for IPG and Apple Watch

After validation of the algorithm, it was applied to the data streamed from Parkinson's disease patient. Six patient subjects who all have severe PD were enrolled in the study; data has been collected from only 5 of the 6 thus far. We applied our model to 48 different datasets from 5 subjects during their daily living. From this point forward, participants will be referred to by codes: NU5U for subject 1, E395 for subject 2, RZCH for subject 3, 6KOZ for subject 4 and AC27 for subject 5. There has been no data available for the sixth subject BOI0. The participants were free to do any daily living activities and were not required to log their activity; therefore, there is no ground truth data for the activities the participants were doing at the time of recording of data. Thus, before we applied our unsupervised classification algorithm to the patient subject data, we validated our algorithm with controlled experimental data, as described in Chapter 4.

We had applied our algorithm to the patient subject data previously, and based on those initial results, we identified 3-5 possible activity categories. The classifications from applying our algorithm were mapped onto the time-domain acceleration signals using a color-coding scheme (see Fig. 4.10 and 4.12, for example). Based on these results, it seemed clear that the acceleration data could be classified into 3 major activity categories: 1) tremor, 2) voluntary movement, and 3) rest. For finer resolution, there could be up to 5 major activity categories: 1) high intensity tremor, 2) low intensity tremor, 3) high intensity voluntary active movement, 4) low intensity voluntary active movement, or 5) rest. According to the distortion metric and knee method discussed in chapter 3, section 3.3, the number of clusters for k-means clustering is specified. An example of the distortion curve used to identify the number of classes for a specific data set is shown in Fig. 40. As can be seen for this particular case, the elbow of the distortion metric occurs between 3 and 4 clusters but closer to 4.

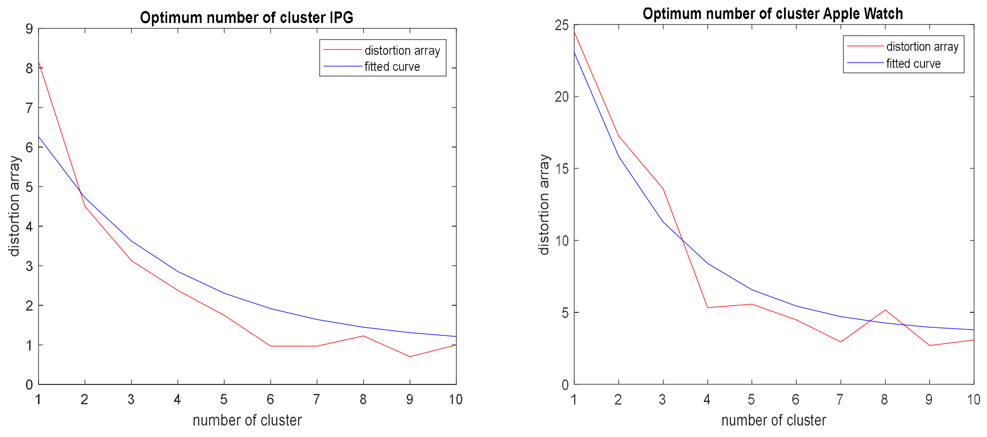


Figure 40. Elbow of distortion matric.

Our clustering algorithm classifies different activities based on the spectral power in four different frequency bands (0-4 Hz, 4-7 Hz, 7-12 Hz, 12-32 Hz) in each axis of acceleration. Each index in the acceleration signal is assigned to a cluster based on the spectral features. Fig. 41 shows an example how the clustering was done-

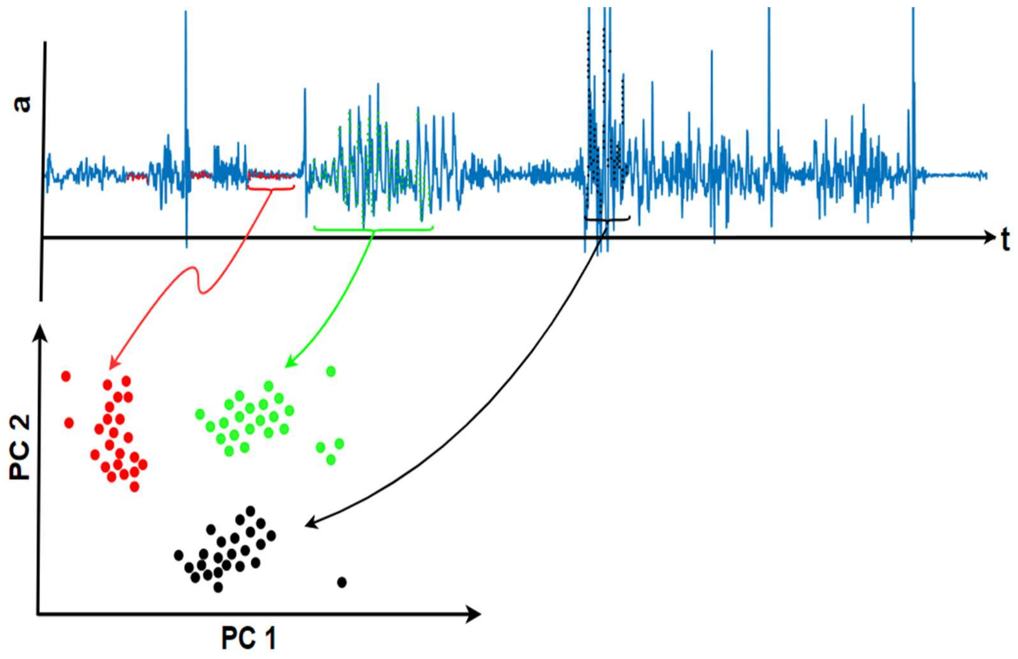


Figure 41. Assigning each acceleration indices into a cluster [32].

In Fig. 41 we can see there are three clusters where the black cluster indicates very high intensity movement as there are more oscillations in the acceleration signal. On the other hand, the portion of the acceleration signal which has lower oscillation than the previous one is clustered as a different cluster (green). The signal which has the lowest oscillation is assigned to red cluster as well. We calculated the total number of indices in each cluster, then divided that by the total number of indices to get the percentage of each activity cluster.

$$Percentage\ of\ each\ activity\ cluster$$
$$= \frac{Total\ number\ of\ indices\ in\ one\ cluster}{Sum\ of\ the\ number\ of\ indices\ in\ all\ clusters} \times 100\%$$

For instance, if we want to calculate the percentage of black cluster the equation will be-

$$Percentage\ of black\ cluster = \frac{Total\ number\ of\ indices\ in\ black\ cluster}{Indices\ in\ black\ cluster + Indices\ in\ green\ cluster + Indices\ in\ red\ cluster} \times 100\%$$

Following this procedure, we calculated the percentage of each activity cluster in the classification result for both the IPG and Apple Watch classification.

For getting the percentage match of the Apple watch classification with the IPG classification, we considered IPG classification as the reference and calculated true positive, true negative, false positive and false negative rates for all the activities. For a given activity A, table 9 describes how each case was accounted for in the confusion matrix. If IPG and Apple Watch both classify an activity as category A, it will be counted as a true positive. If IPG classifies an activity as *A*, but Apple Watch does not, it will be counted as a false negative. If IPG does not detect an activity as *A* (i.e., as *B* or *C)*, but Apple Watch classifies it as *A*, it will be counted as a false positive. Finally, if IPG and Apple watch both classify it as not *A*, it will be counted as a true negative. We calculated these for all the indices present in the acceleration signal.

Table 9. Confusion matrix.

		Apple watch	
		Activity *A*	Not Activity *A*
IPG	Activity *A*	TP	FN
	Not activity *A*	FP	TN

Next, we calculated the percentage of classification match of the Apple Watch with the IPG classification.

Table 10. Confusion matrix for three classes.

	TP	TN	FP	FN
Tremor	N_{TT}	$N_{\sim T\sim T}$	$N_{T\sim T}$	$N_{\sim TT}$
Rest	N_{RR}	$N_{\sim R\sim R}$	$N_{R\sim R}$	$N_{\sim RR}$
Voluntary active movement	N_{VV}	$N_{\sim V\sim V}$	$N_{V\sim V}$	$N_{\sim VV}$

Where:

N_{TT} = Number of datapoints for which Apple Watch and IPG both detected tremor

$N_{\sim T\sim T}$ = Number of datapoints for which Apple Watch and IPG both didn't detect tremor

$N_{T\sim T}$ = Number of datapoints for which Apple Watch detected tremor, but IPG didn't

$N_{\sim TT}$ = Number of datapoints for which Apple Watch didn't detect tremor, but IPG detected

N_{RR} = Number of datapoints for which Apple Watch and IPG both detected rest

$N_{\sim R\sim R}$ = Number of datapoints for which Apple Watch and IPG both didn't detect rest

$N_{R\sim R}$ = Number of datapoints for which Apple Watch detected rest, but IPG didn't

$N_{\sim RR}$ = Number of datapoints for which Apple Watch didn't detect rest, but IPG detected

N_{VV} = Number of datapoints for which Apple Watch and IPG both detected voluntary movement

$N_{\sim V \sim V}$ = Number of datapoints for which Apple Watch and IPG both didn't detect voluntary movement

$N_{V \sim V}$ = Number of datapoints for which Apple Watch detected voluntary movement, but IPG didn't

$N_{\sim VV}$ = Number of datapoints for which Apple Watch didn't detect rest, but IPG detected

We calculated Percentage of tremor match by adding all the N_{TT} and $N_{\sim T \sim T}$ for all 48 dataset, then dividing it by the sum of all N_{TT}, $N_{\sim T \sim T}$, $N_{T \sim T}$, $N_{\sim TT}$ for all 48 dataset. We also calculated rest match and voluntary activity match of Apple Watch-based classification with IPG-based classification for all 48 datasets as follows-

$$Tremor\ match = \frac{Total\ N_{TT} + Total\ N_{\sim T \sim T}}{Total\ N_{TT} + Total_{\sim T \sim T}\ + Total\ N_{T \sim T}\ + Total\ N_{\sim TT}} \times 100\%$$

$$Tremor\ mismatch = 100\% - Tremor\ match$$

$$Rest\ match = \frac{Total\ N_{RR} + Total\ N_{\sim R \sim R}}{Total\ N_{RR} + Total_{\sim R \sim R}\ + Total\ N_{R \sim R}\ + Total\ N_{\sim RR}} \times 100\%$$

$$Rest\ mismatch = 100\% - Rest\ match$$

$$Voluntary\ activity\ match$$
$$= \frac{Total\ N_{VV} + Total\ N_{\sim V \sim V}}{Total\ N_{VV} + Total_{\sim V \sim V}\ + Total\ N_{V \sim V} + Total\ N_{\sim VV}} \times 100\%$$

$$Tremor\ mismatch = 100\% - Voluntary\ activity\ match$$

Table 11 shows the percentage of classification matches and mismatches for each class.

Table 11. Percentage match and mismatch of Apple Watch-based classification with IPG-based classification.

	Percentage match	Percentage mismatch
Tremor	83.27%	15.93%
Rest	69.67%	30.32%
Voluntary active movement	68.36%	31.63%

From the table we can see that most of the mismatch of the Apple Watch-based classification with the IPG-based classification was for the rest and voluntary active movement. 83.27% of tremor activity of the Apple Watch classification got matched with the IPG classification.

5.2 Tremor vs. voluntary active movement

While investigating the classification result, we noticed there is 15.93% tremor mismatch between Apple Watch and IPG classification. In most cases, Apple Watch detected more tremors. In the following figure, Apple Watch detected continuous tremor (blue), while IPG detected some voluntary movement (red) during tremor activity.

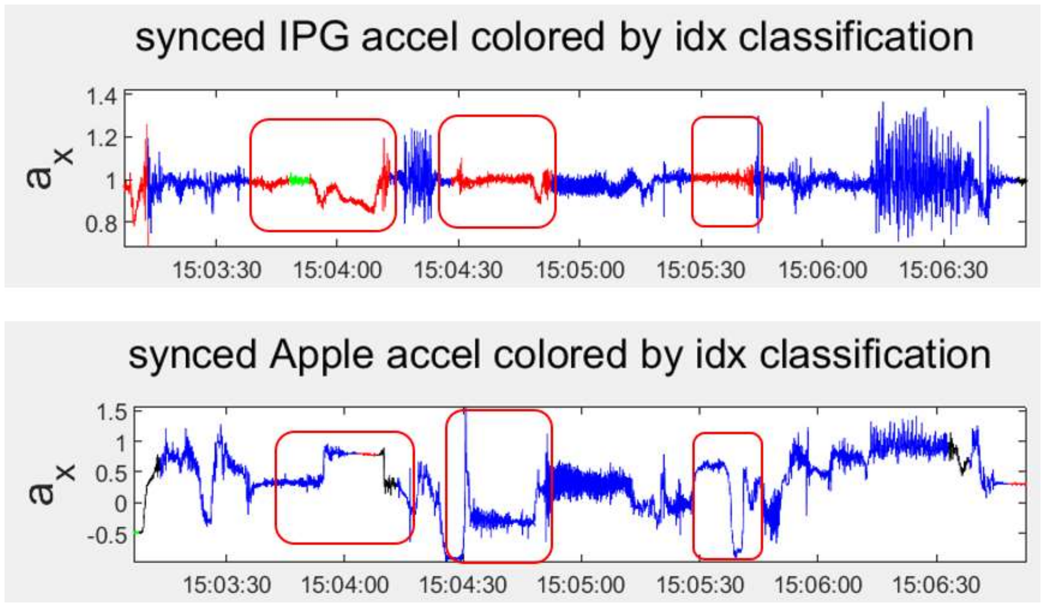
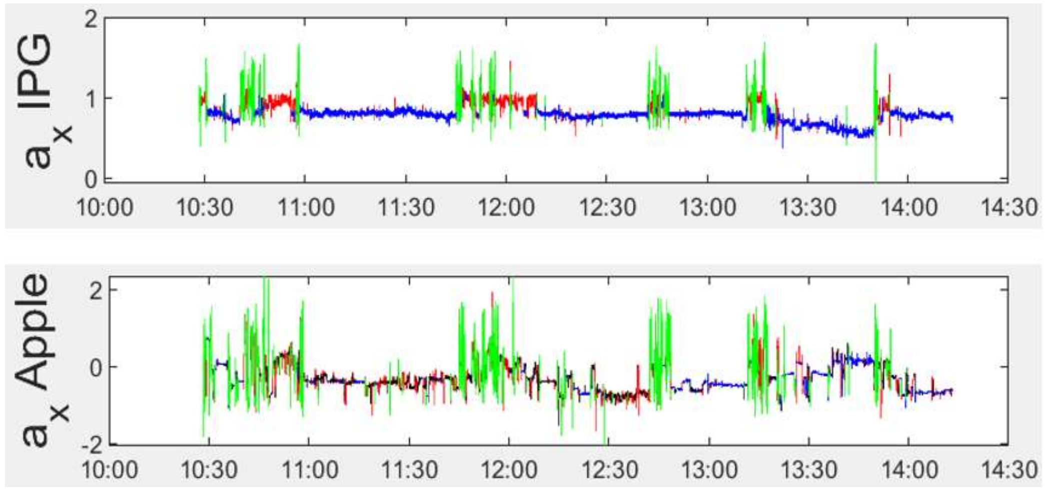


Figure 42. Apple Watch detected more tremor (blue) than IPG.

5.2.1 Apple and IPG both detect tremor

In cases where IPG classified tremor and the tremor amplitude was strong, Apple Watch also classified the activity as tremor. As an example of such a case, let us discuss the following dataset 10 for subject E395.



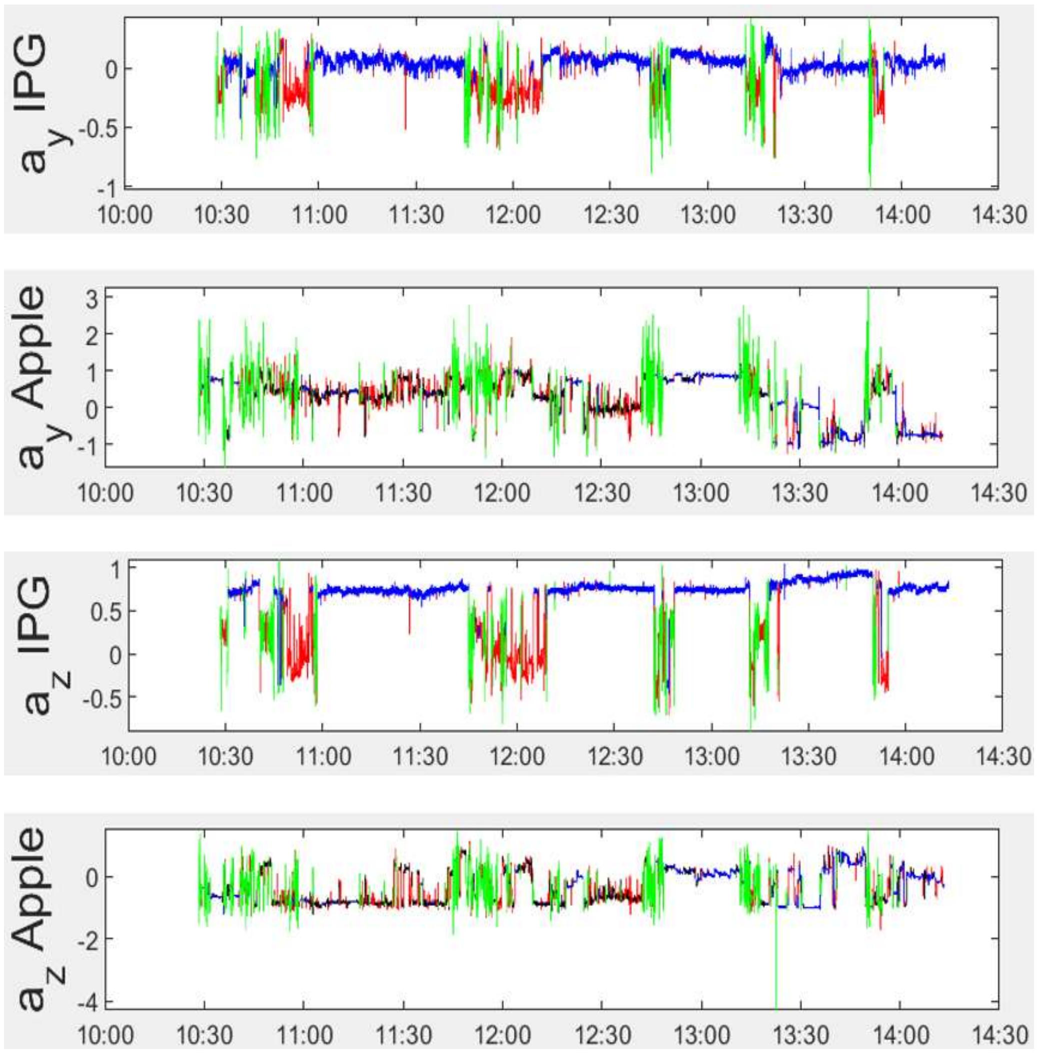


Figure 43. Comparison of IPG and Apple Watch classification.

According to the elbow method, we identified the optimal number of clusters to be four clusters both in the IPG classification and Apple Watch classification. As seen in Fig. 44, a couple outliers were selected as cluster 4 (black) in the IPG classification; thus, the meaningful categories in this instance were tremor, voluntary movement, and rest.

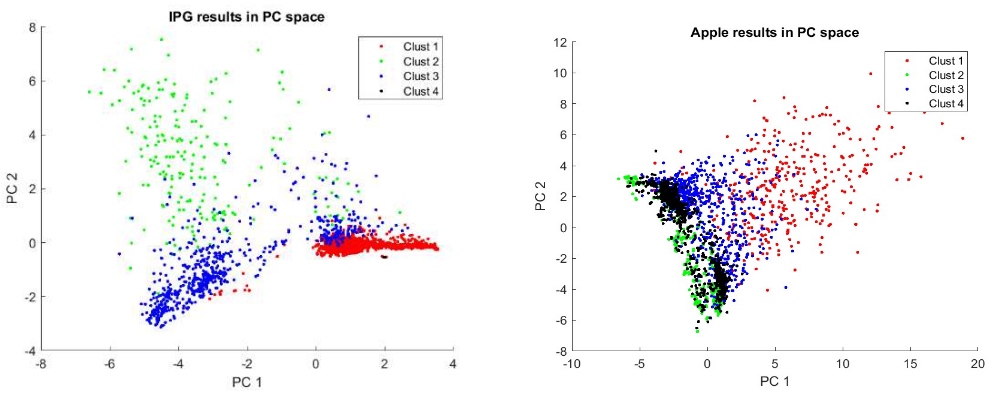


Figure 44. Clusters of IPG and Apple Watch Classification.

Let us determine which cluster indicates tremor, rest, and voluntary movement. To find that out let us analyze the power spectral density in four different frequency bands in each axis for IPG.

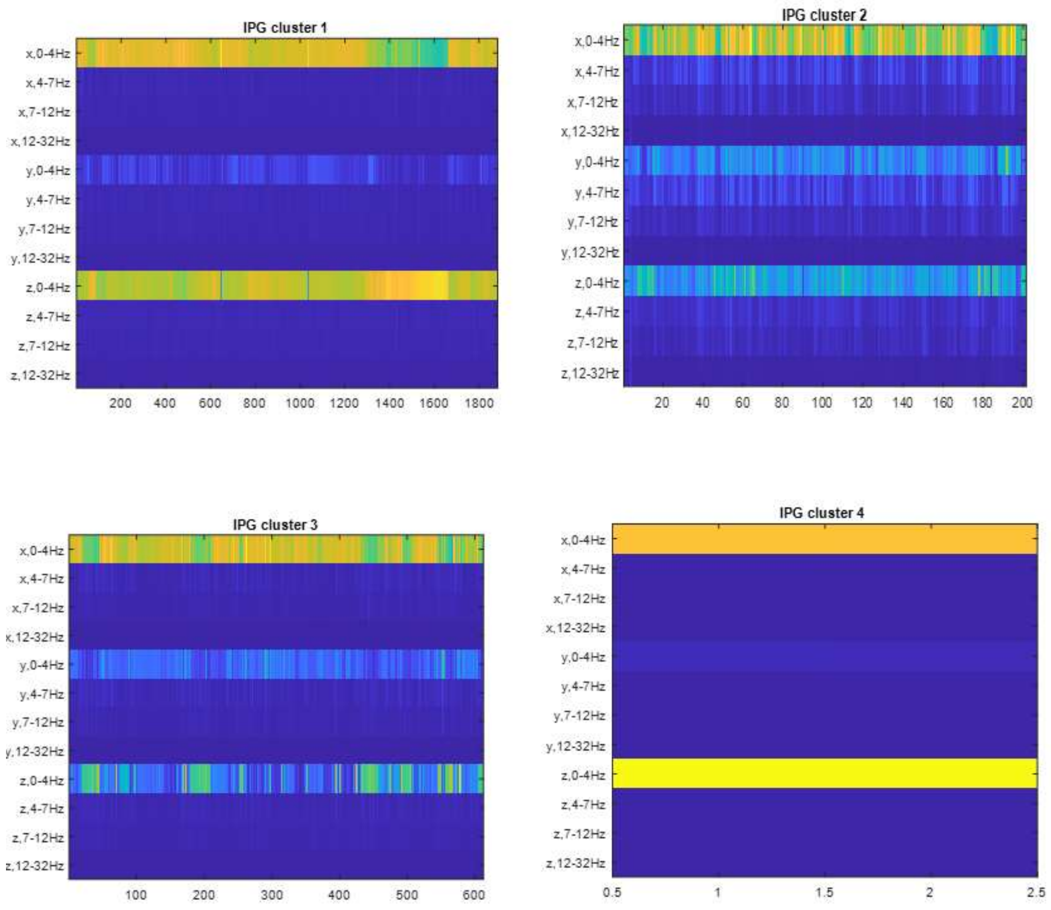


Figure 45. STFT of IPG cluster 1-4.

Starting from top of the plot, row 1 is the x dimension of the 0-4 Hz band, row 2 is the x dimension of the 4-7 Hz band, row 3 is the x dimension of the 7-12 Hz band, row 4 is the x dimension of the 12-32 Hz band. Row 5-8 are the y dimension of the four frequency bands and row 9-12 are the z dimension of the four frequency bands.

In all the clusters the first row which is 0-4 Hz band in x axis is the brightest row. That means the 0-4 Hz band has the highest energy. Among all the clusters, cluster 1 and cluster 3 has a brighter 0-4 Hz band which means that cluster 1 and cluster 3 can be voluntary movement or rest cluster. Referring to Fig. 43, according to the signal characteristics cluster 1 (red) has more oscillation and higher amplitude compared to cluster 3 (blue). So, we can consider cluster 3 (blue) as rest and cluster 1 (red) as voluntary active movement. The 4-7 Hz band (2nd, 6th and 10th row) has the highest power in cluster 2 as this band is brighter in cluster 2 than any other cluster. 4-7 Hz band is the tremor band, so cluster 2 (green) can be tremor.

Similarly, to detect the activities let us analyze the power spectral density in four different frequency band in each axis for Apple Watch.

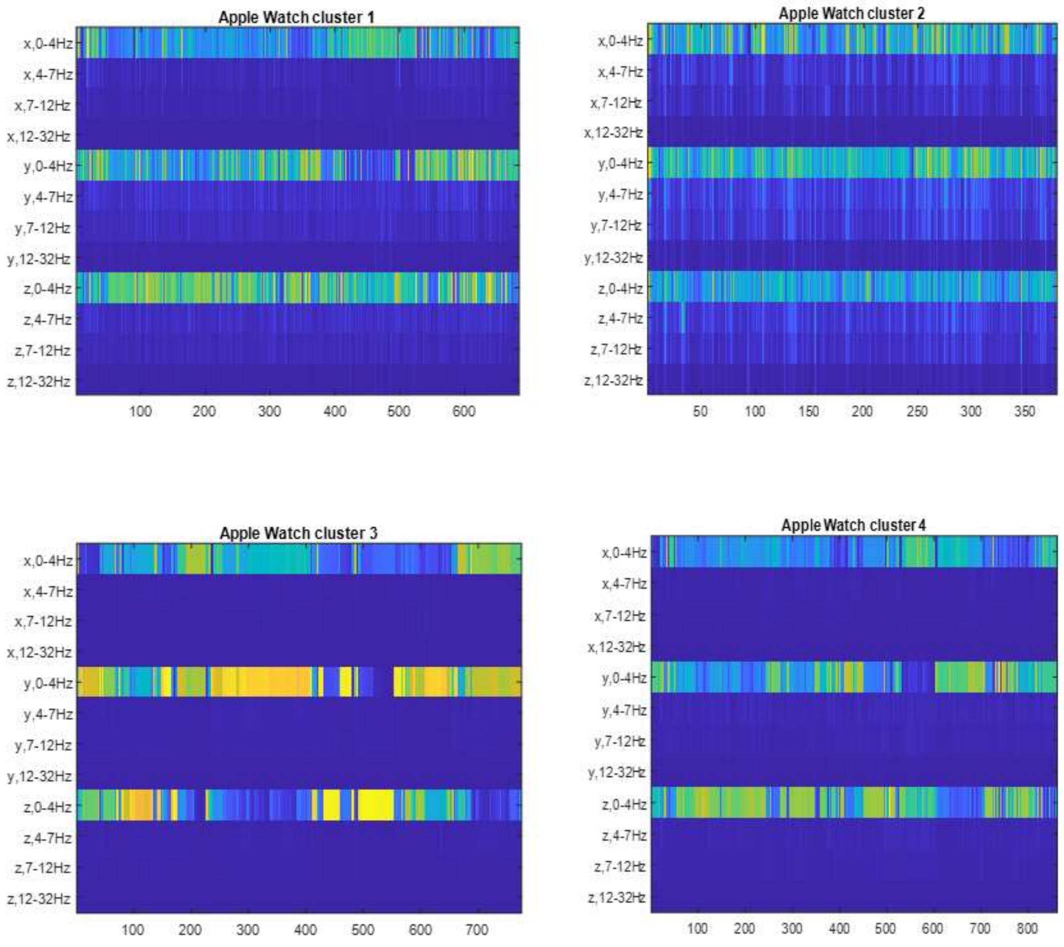


Figure 46. STFT of Apple Watch cluster 1-4.

Among the four clusters, cluster 2 has brighter 4-7 Hz (2nd, 6th and 10th row) band compared to other clusters. Cluster 2 (green) can be considered as tremor. Based on signal characteristics (Fig. 43), cluster 3 (blue) can be rest, cluster 1 (red) can be high intensity active movement and cluster 4 (black) can be some low intensity active movement in Apple Watch classification.

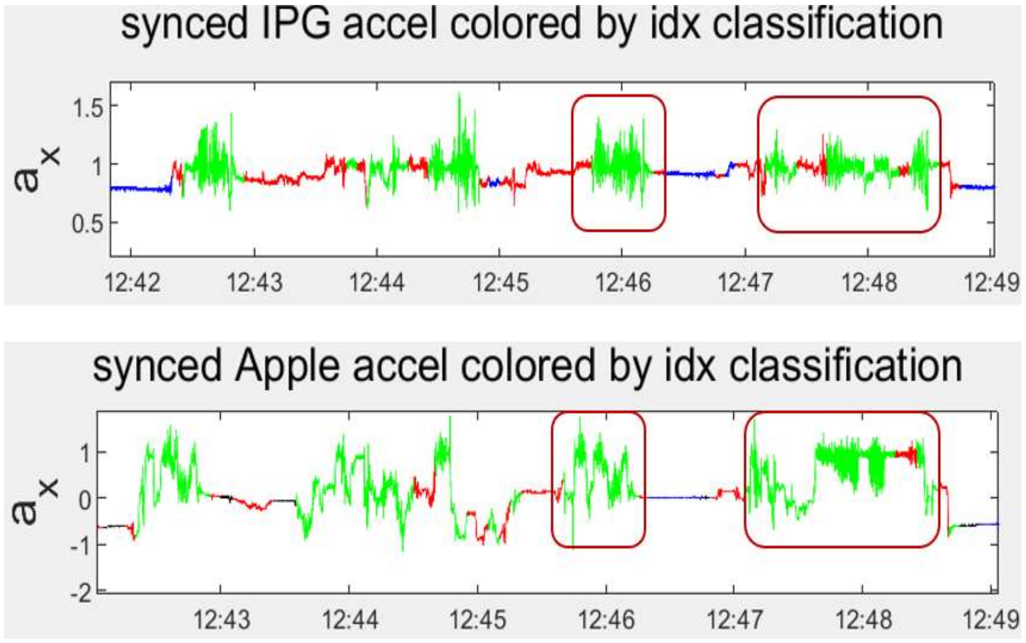


Figure 47. Tremor (green) detected in both IPG and Apple Watch.

In both IPG and Apple Watch green (cluster 2) is tremor. Fig. 47 illustrates an example of IPG, and Apple watch-based classifications generally agree on the occurrence of tremor. When IPG detects tremor (green) at any instance, Apple Watch also detects tremor (green) at that instance.

5.2.2 Apple detects tremor while IPG detects voluntary movement

There are cases where Apple Watch detected an activity as tremor while that activity was detected as voluntary active movement in IPG. Overall, tremor percentage was higher in Apple watch classification than IPG. For the same dataset analyzed above Apple Watch detected more tremors than IPG. Fig. 48 shows Apple Watch detected tremor while IPG detected voluntary movement. When the activity was classified as tremor (green) in Apple Watch, in some places the same activity was detected as voluntary movement (red) in IPG.

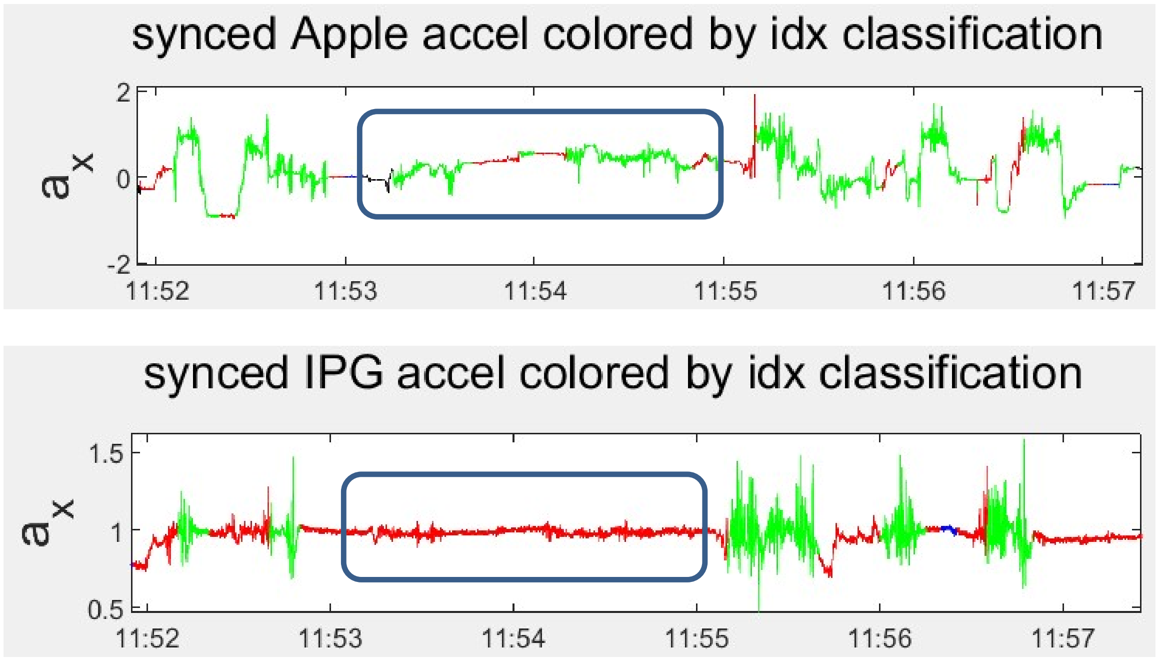


Figure 48. Apple Watch detects tremor (green) while IPG detects voluntary movement (red) for subject E395, dataset 10.

Let us analyze another dataset, dataset 2 for subject NU5U. According to elbow method of clustering, both IPG and Apple Watch have 4 numbers of clusters including one outlier in IPG.

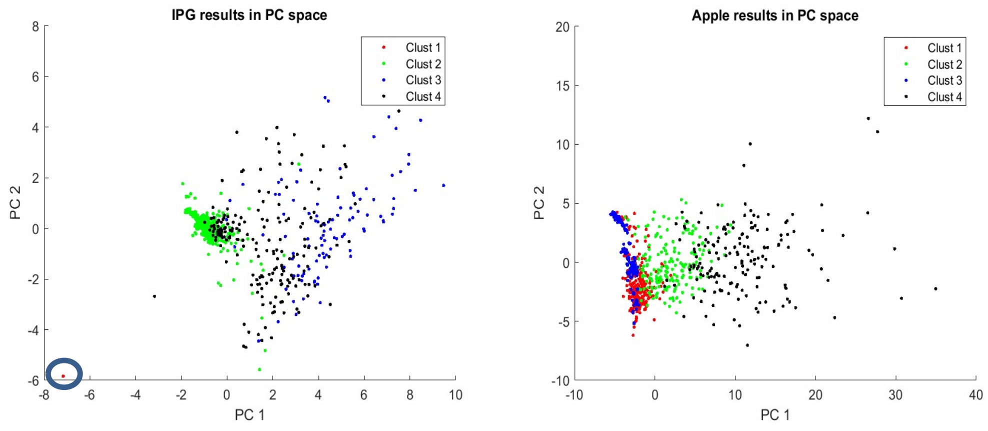


Figure 49. Clusters of IPG and Apple Watch Classification for subject NU5U, dataset 2.

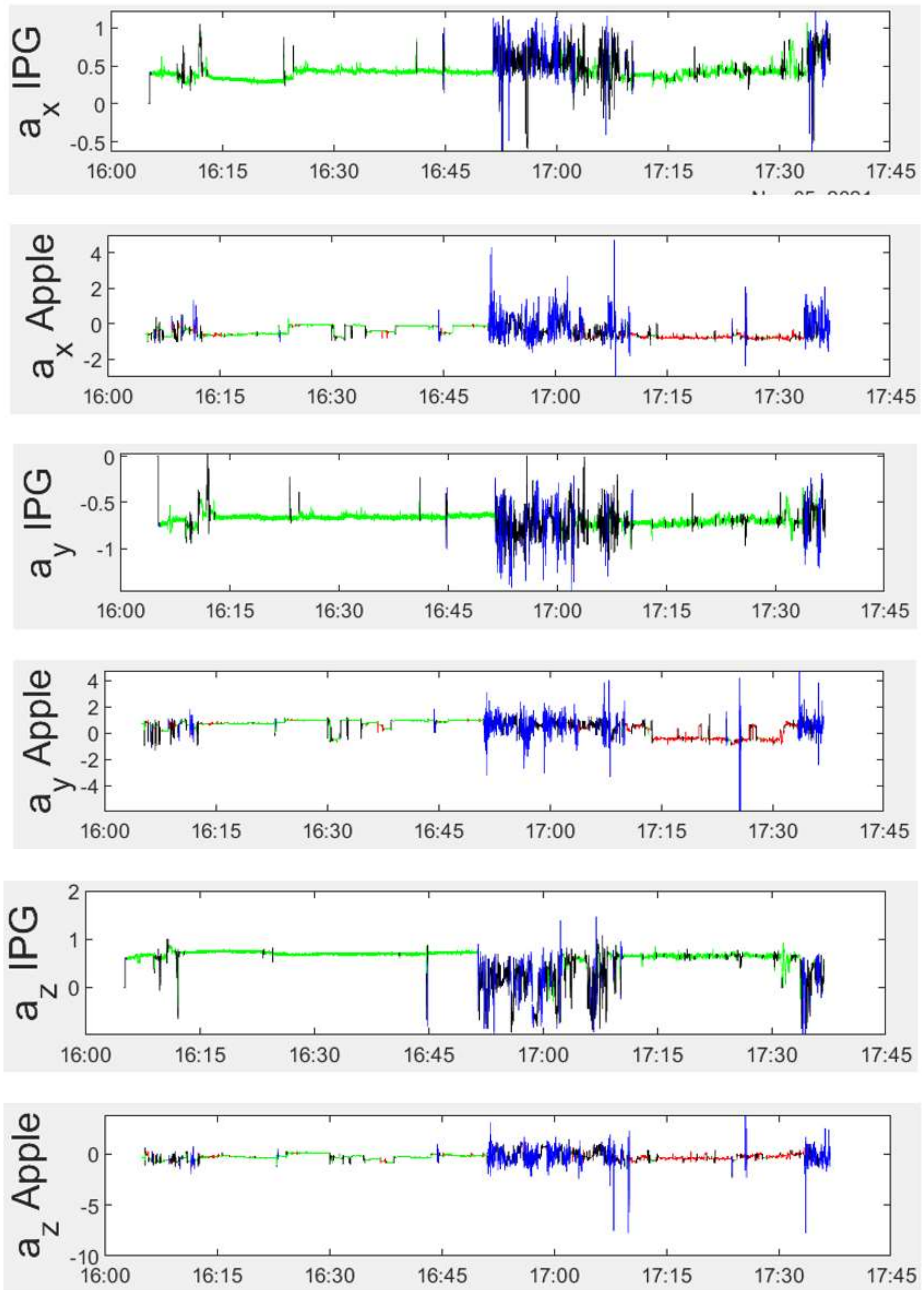


Figure 50. Comparison of IPG and Apple Watch classification for subject NU5U, dataset 2.

Based on the power spectral density and signal characteristics we can predict that that blue (cluster 3) indicates tremor, black (cluster 4) indicates voluntary active movement and green (cluster 2) indicates rest in both the IPG and Apple Watch classification. In Apple Watch classification red (cluster 1) indicates low intensity voluntary active movement. The percentage of tremor is higher in Apple Watch classification than IPG. IPG detected 9% of the activity as tremor while Apple Watch detected 15% of the activity as tremor. Fig. 51 shows Apple Watch detected tremor while IPG detected voluntary movement.

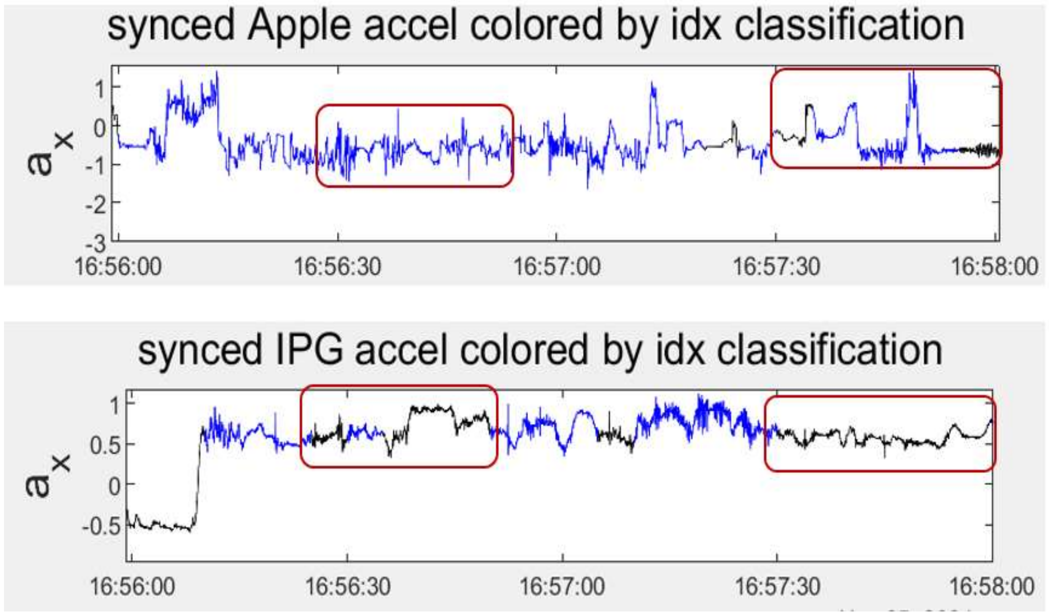
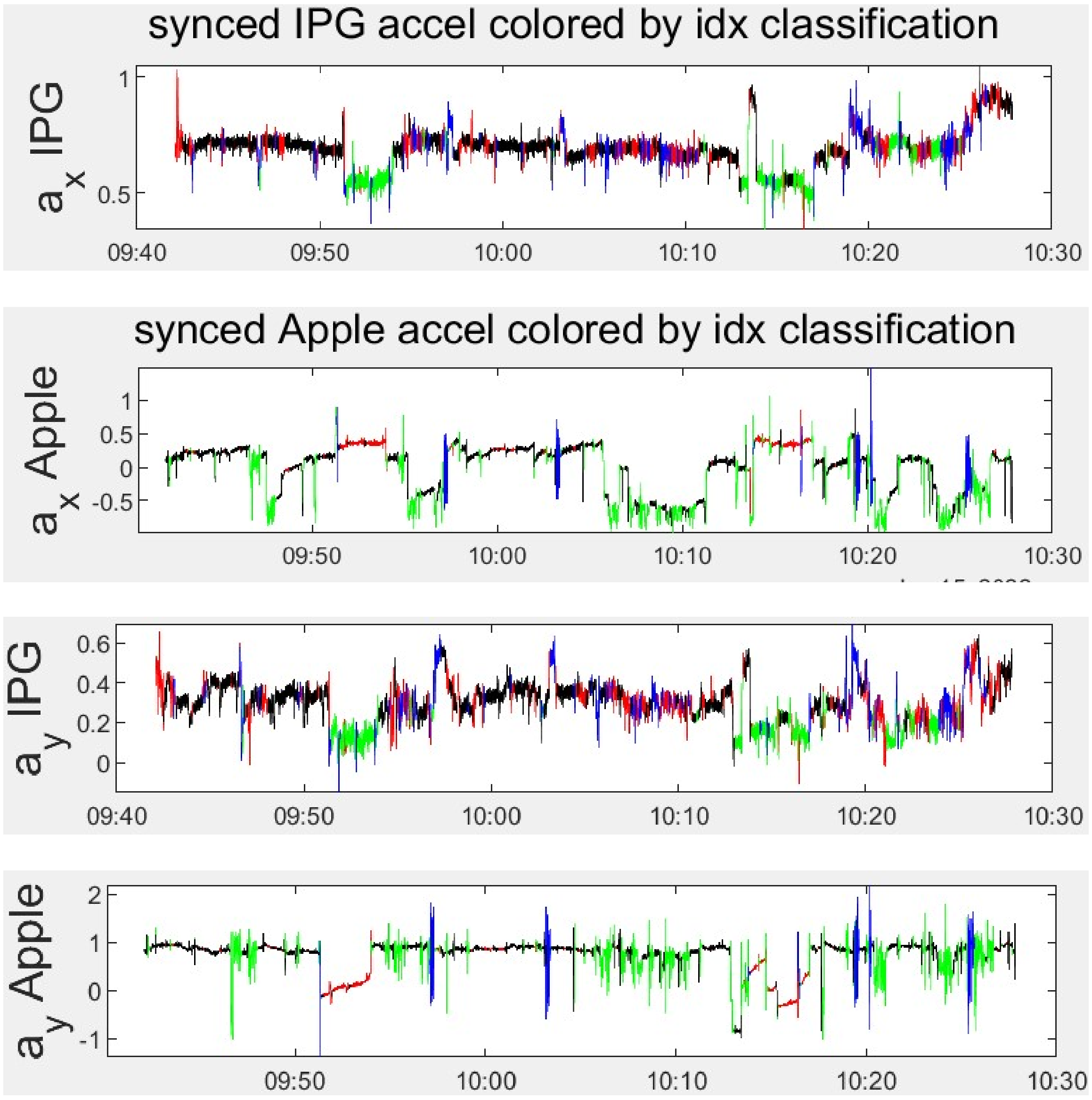


Figure 51. Apple Watch detects tremor (blue) while IPG detects voluntary movement (black).

For this dataset, blue (cluster 3) was tremor for both IPG and Apple Watch. We can see that from 16:56:00 to 16:57:30 Apple watch classified the activity as cluster 3 i.e., tremor. On the contrary, in IPG besides tremor (blue), voluntary activity (black) was also detected at that time frame. We hypothesize that the subject was having tremor while doing some active body movement.

From Fig 52 it can be seen that overall tremor detection is higher in Apple Watch classification for all subjects except AC27. For subject AC27 IPG detected more tremor.



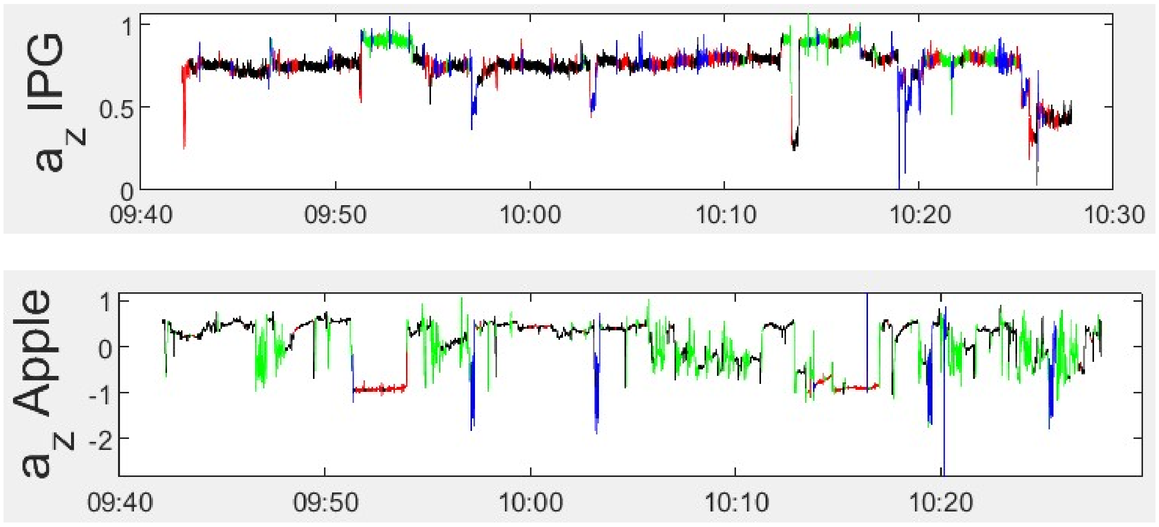


Figure 52. Classification result for subject 5 dataset 4 where tremor (blue) was detected.

After analyzing the power spectral density and signal characteristics we found that blue color (cluster 3) indicates tremor for both IPG and Apple Watch. After zooming in on the result, the tremor was more visible.

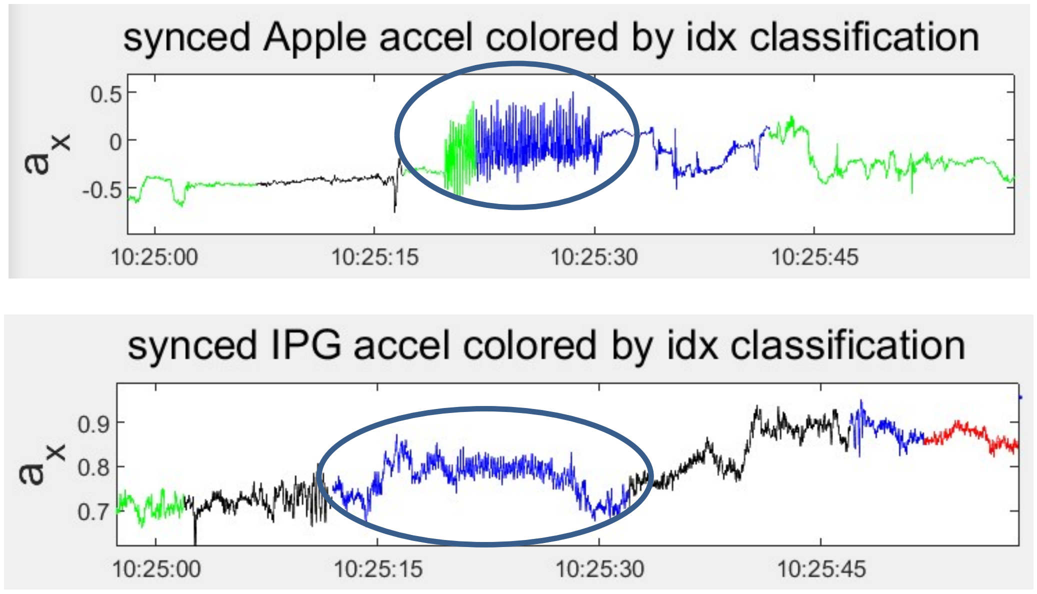


Figure 53. Tremor (blue) detected in subject 5 dataset 4.

From Fig. 54 we can see that tremor is detected in both IPG and Apple watch for only a few seconds. Tremor is more strongly visible in Apple Watch; yet the percentage of data classified as tremor was greater in IPG than Apple Watch for this particular subject.

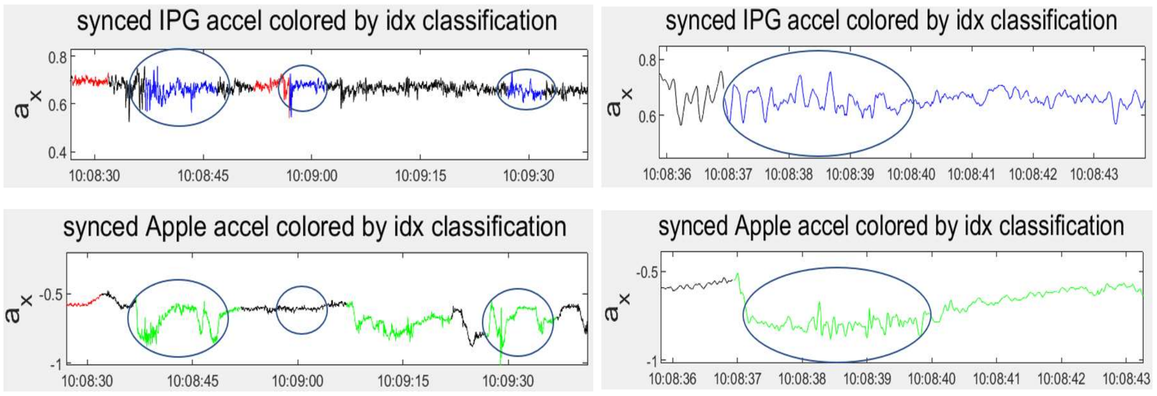


Figure 54. IPG detected tremor (blue) while Apple Watch detected voluntary movement (green and black), right image more zoomed in.

If we zoom in the signal of the first ellipse, we can see from the right figure that from 10:08:37 to 10:08:40 the signal could be tremor, since the signal corresponds to 4 Hz frequency. So, IPG can detect tremors which last only for 2-3 seconds. On the contrary, Apple Watch can't detect such tremors.

5.3 Voluntary active movement vs. rest

Overall Apple Watch detected more voluntary movement while IPG detected more rest. Some activities which are classified as rest in IPG are classified as rest in Apple Watch classification. Because of the external position of the Apple Watch, it detected arbitrary hand movement while resting as voluntary active movement.

5.3.1 Apple and IPG both detect rest

There were also cases in which detection of both voluntary movement and rest matched between IPG and Apple watch classification. Fig. 55 shows the classification result for subject NU5U, dataset 7 and illustrates one case where such a match for voluntary movement and rest occurred.

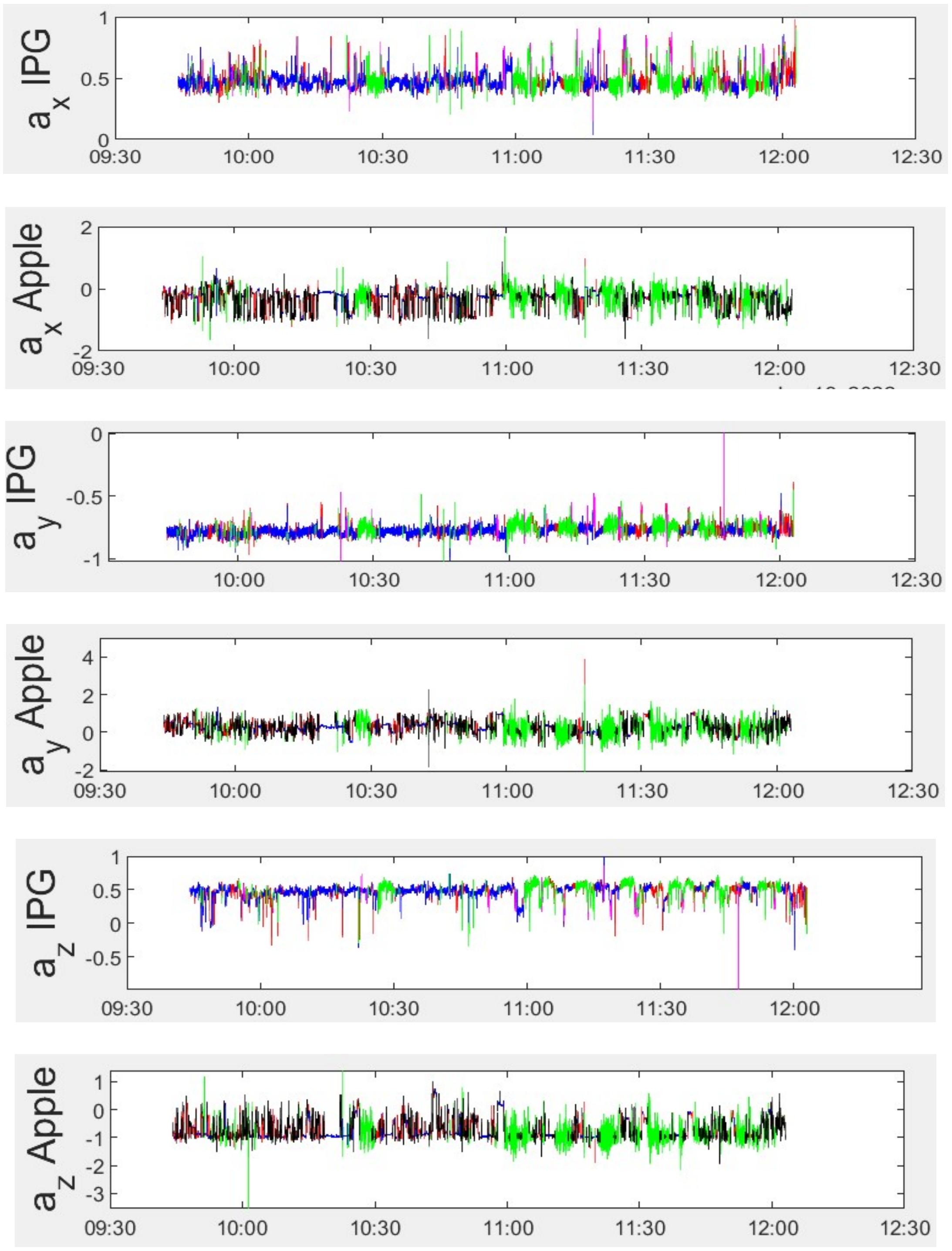


Figure 55. Comparison of IPG and Apple Watch classification for subject NU5U, dataset 7.

According to the elbow method of clustering, there are four clusters for both the Apple Watch and IPG classification. Let us analyze the STFTs of the clusters to figure out which cluster indicates which activity.

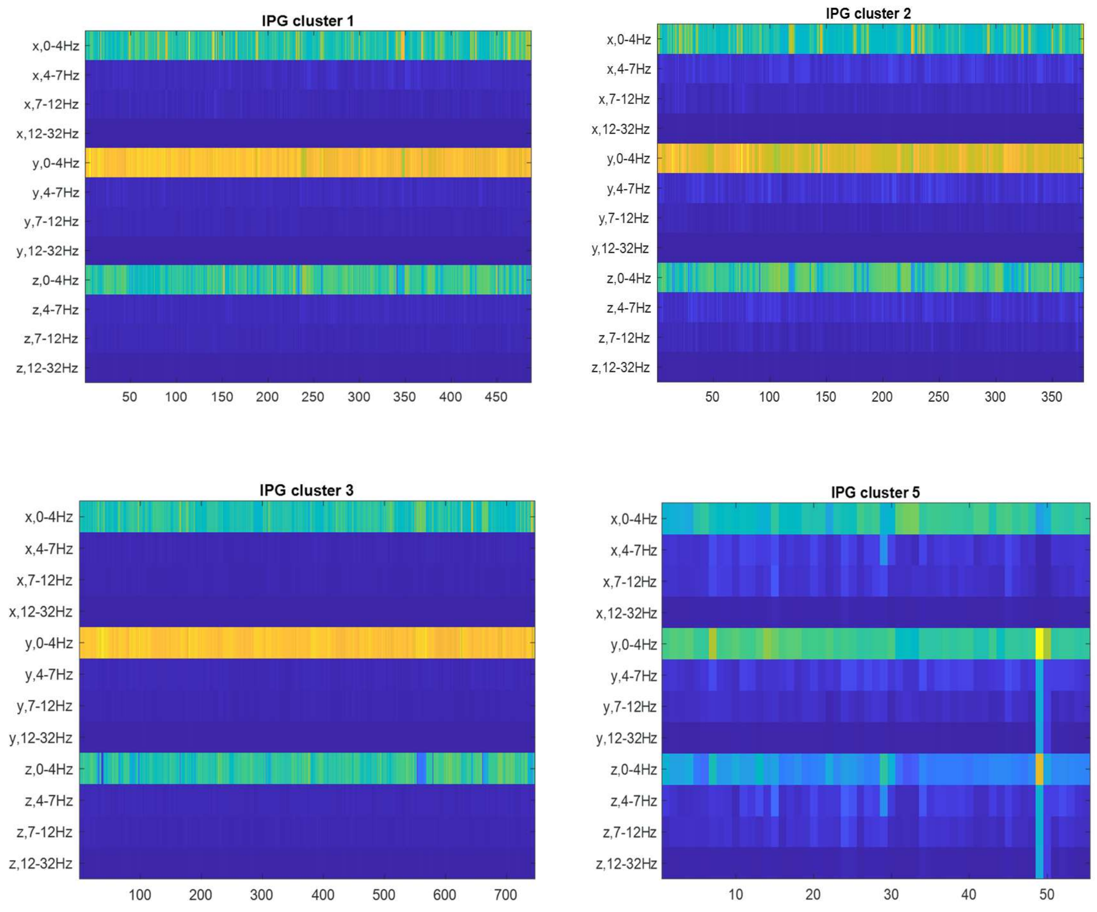


Figure 56. STFT of IPG cluster 1-4 for Subject NU5U, dataset 7.

From the STFT of the clusters it's visible that the 4-7 Hz band (2nd,6th and 10th row) is brighter in cluster 2 (green) and cluster 5 (magenta). That means these two clusters have higher power in the 4-7 Hz band. Cluster 2 (green) and cluster 5 (magenta) can be the tremor bands. Cluster 1 (red) and cluster 3 (blue) have brighter 0-4 Hz band (1st, 5th and 9th row). These two clusters can be rest or voluntary active movement. By analyzing the signal characteristics, we can hypothesize that cluster 3 (blue) can be rest and cluster 1 (red) can be voluntary movement.

Similarly, to detect the activities let us analyze the power spectral density in four different frequency band in each axis for Apple Watch.

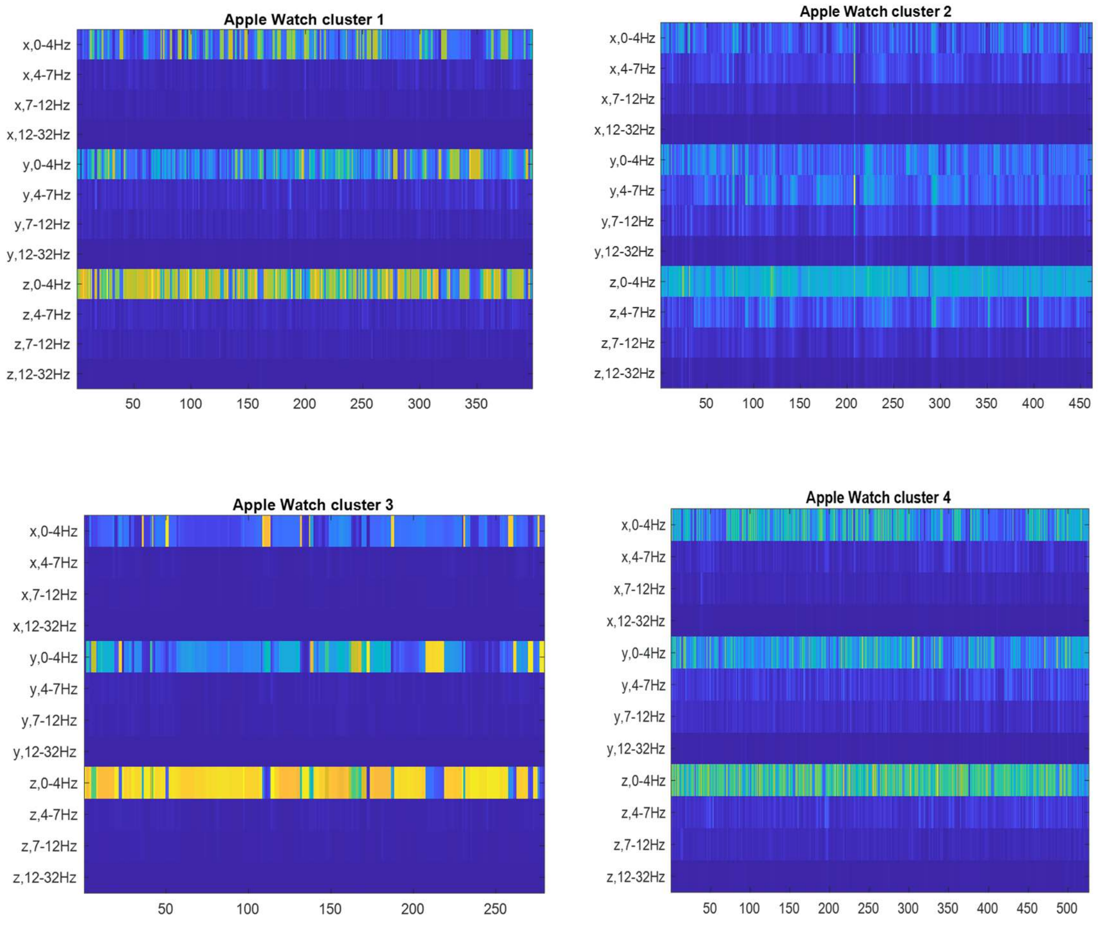


Figure 57. STFT of Apple Watch cluster 1-4 for Subject NU5U, dataset 7.

By analyzing the STFTs and the signal characteristics we can predict that for Apple Watch cluster 2 (green) is tremor, cluster 3 (blue) is rest, cluster 4 (black) is high intensity voluntary movement, cluster 1 (red) is low intensity voluntary movement. Fig. 58 shows the difference between red and black cluster-

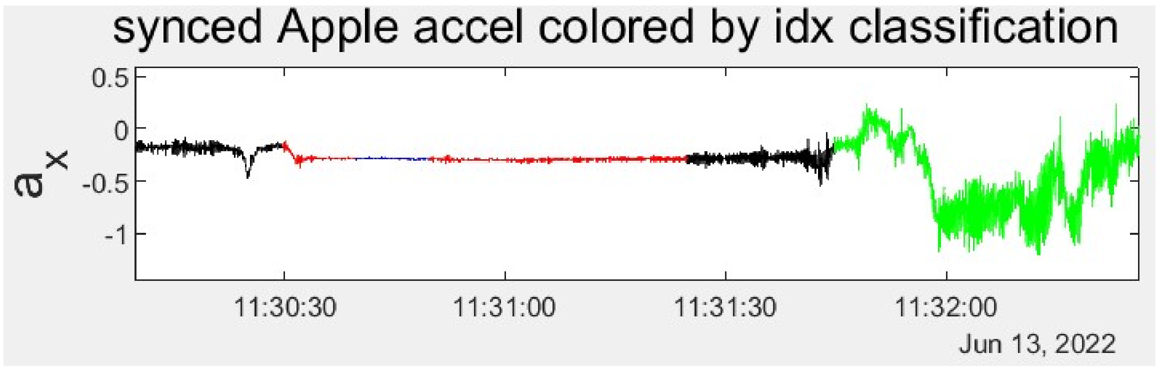


Figure 58. High and low intensity movement.

Fig. 59 shows the case when IPG and Apple Watch both detect rest. For both Apple Watch and IPG blue is rest. We can see the marked rectangles where both Apple Watch and IPG detects rest at the same time.

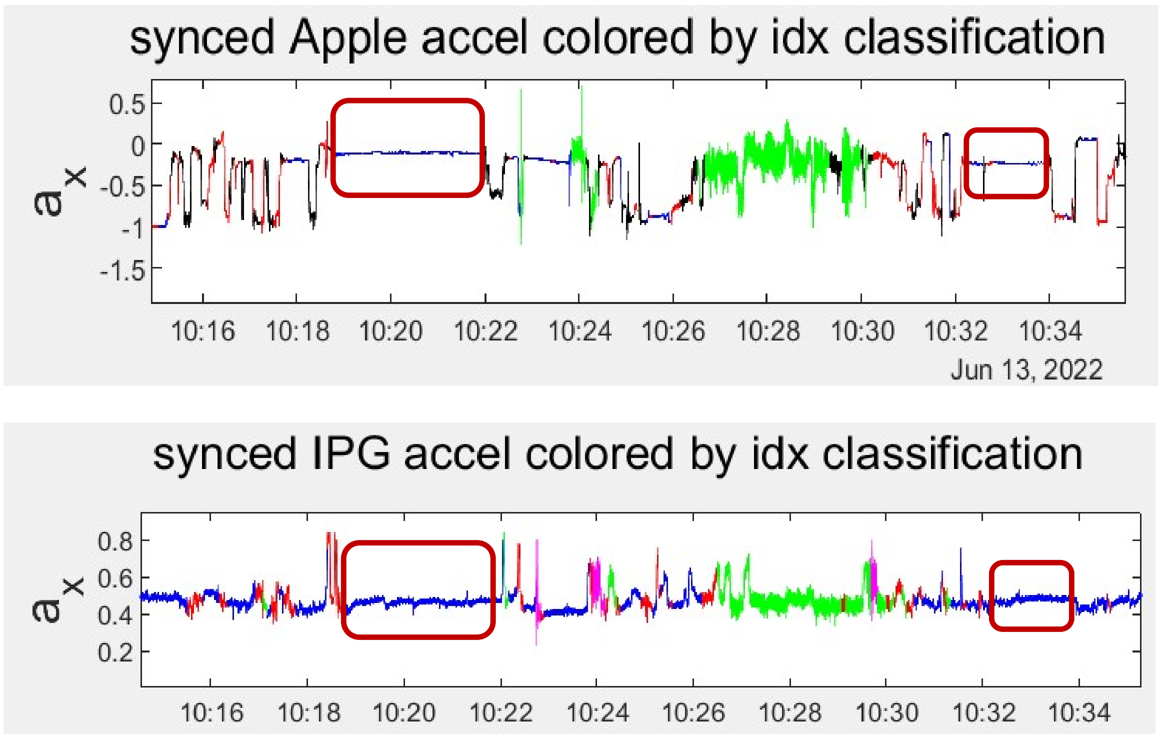


Figure 59. Apple Watch (blue) and IPG (blue) both detect rest.

In many cases we found that Apple Watch and IPG detect rest at the same time. We also found many cases where IPG detects rest while Apple Watch detects voluntary active movement.

5.3.1 Apple Watch detects voluntary movement while IPG detects rest

Apple Watch is worn on the wrist, while IPG is implanted on the chest. Because Apple Watch is placed on the hand, it's more sensitive to hand movement than IPG. Apple Watch detects the arbitrary hand movement while resting as voluntary movement. Let us analyze the same dataset, subject NU5U, dataset 7. Fig. 60 shows the case where IPG detects rest while Apple Watch detects voluntary movement.

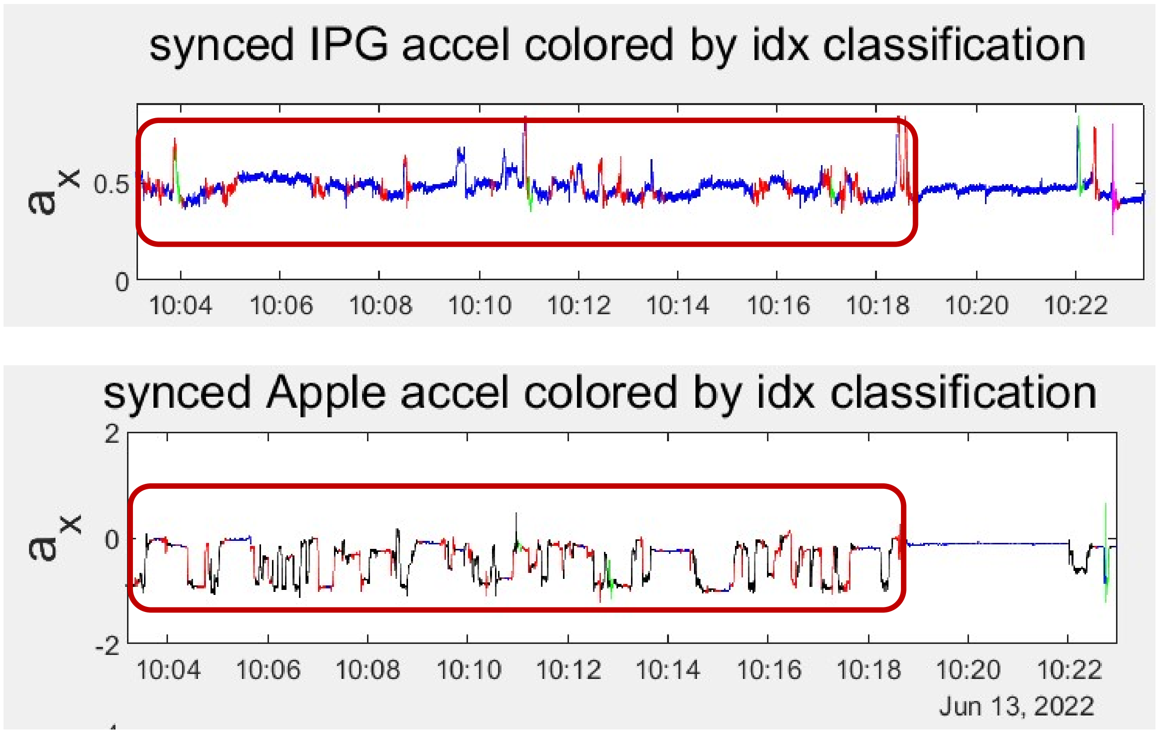


Figure 60. IPG detects rest (blue) while Apple Watch detects voluntary movement (black and red).

For subject NU5U, dataset 7 in IPG classification blue was rest, red was voluntary movement. In Apple Watch classification blue was also rest, black and red was voluntary movement. We can see in Fig. 60 that in the IPG classification from 10:04 to 10:18 most of the activity is classified as rest (blue). On the contrary, in Apple Watch classification for the same time frame most of the activity is classified as voluntary movement (black and red). We assume that during rest activity probably the subject was performing some arm movement which is classified as voluntary movement in Apple Watch classification. In that sense, IPG gives us more accurate classification than Apple Watch.

5.4 Comparing percentage of tremor classification to clinical assessment

After analyzing the classification result of IPG and Apple Watch acceleration, we compared our tremor classification with clinical assessment. Our clinical experts assessed each patient and according to our clinical experts subject NU5U has lots of hand tremor, subject E395 has intermittent tremor, subject RZCH has hand tremor, subject 6KOZ has tremor in the jaw and face and subject AC27 has never reported tremor. Table 12 shows the comparison of tremor detection with clinical assessment.

Table 12. Comparison of tremor detection with clinical assessment.

Subject		Clinical Assessment	Avg IPG Tremor %	Avg Apple Watch Tremor %
Subject 1	NU5U	Lots of hand tremor	26.09	26.6
Subject 2	E395	Intermittent tremor	16.68	17.16
Subject 3	RZCH	Hand tremor	15.7	27.5
Subject 4	6KOZ	Tremor in jaw and face	22.78	25.92
Subject 5	AC27	Never reported Tremor	2.82	0.81

It’s visible from the table that subject 1 has the highest percentage of tremor in IPG classification with slightly more tremor in Apple Watch. Since this subject has a lot of tremors in hand and due to the external location of Apple Watch, it detected more tremors. For subject AC27 we got 2.82% tremor in IPG and 0.81% tremor in Apple Watch classification. We ran a total of 10 datasets for subject AC27 and got tremor in only two datasets for a very small amount of time. Although the patient has never reported tremor, our algorithm detected tremor for a small amount of time which indicates that our algorithm is able to detect tremor even if tremor doesn’t continue for a long time. This proves the sensitivity of our algorithm.

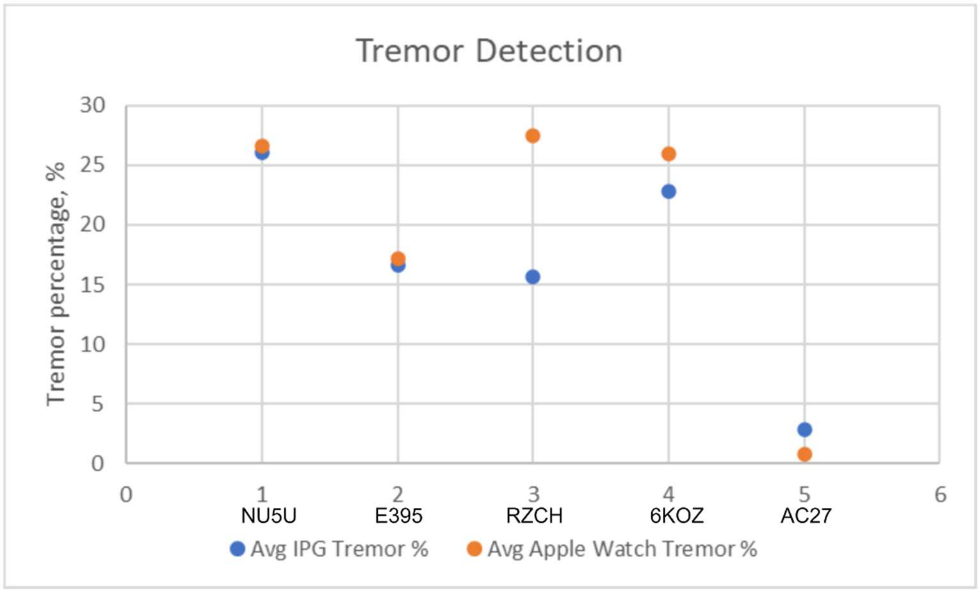


Figure 61. Average IPG and Apple Watch tremor detection.

Overall, our classification result coincides with the clinical assessment for all subjects.

CHAPTER 6

Summary of Results

The results in this thesis show that acceleration acquired by the Apple Watch and IPG accelerometers are largely consistent with each other, but also identify and illustrate differences between these two accelerometers' data. The differences in signal characteristics and performance of the activity classification are consistent with the differences in the location of the two accelerometers; namely, the accelerometer on the wrist (the Apple Watch) detects more power in the hand tremor but does not always classify the rest states accurately, whereas the implanted IPG accelerometer is best able to classify rest states when the body is stationary (whether sitting, standing, or lying down). We computed the Pearson's correlation coefficient between the two accelerometers to determine how linearly related acceleration from the two accelerometers are to each other. The accelerations recorded from these two accelerometers were correlated by an average of 0.43 among all the datasets when we used a window size of 10 second. We obtained the highest average correlation coefficient of 0.70 among all the datasets for a 10-minute window.

We then analyzed the signals in the frequency domain by performing the short time Fourier transform. We calculated the integral power in different frequency bands such as 0-4 Hz, 4-7 Hz, 7-12 Hz and 12-32 Hz. By analyzing the integral power plots of different frequency bands, we saw that Apple Watch collected more power in all the bands. In some instances, the IPG detected no activity even while the Apple Watch detected activity, and often detected less power in acceleration. Although Both Apple Watch and IPG detected activity at the Parkinson's tremor band (4-7 Hz), Apple Watch

4-7 Hz band had more power than IPG. This was consistent with our hypothesis that the Apple Watch would detect more tremors than the IPG, but we discovered that there were cases in which IPG detected tremors as strongly as the Apple Watch. We hypothesize based on these results that while tremor is most visible in the appendages, there may be tremor in the core of the body that is just as readily detected by the IPG.

From the classification result plots, it is visible that IPG classification was clearer than Apple Watch. IPG detected rest activity more accurately, while Apple Watch picked some low intensity movement at the time of rest. This was because of the external location of the Apple Watch. Because of the location, Apple Watch is more sensitive to movements and as a result it had more power than IPG. Apple Watch might misclassify some high intensity movements to be tremor, because of its sensitivity to movements. After getting the integral power in each band, we classified the acceleration signal into Parkinson's tremor, rest, and voluntary active movement according to the spectral features.

Since this was an unsupervised classification, for validating the model we performed a controlled experiment with ActiGraph and Apple Watch. We achieved an overall accuracy of 91% of our classification algorithm with ActiGraph and slightly lower accuracy with Apple Watch. Most of the misclassification was due to the voluntary active movement being classified as rest.

After validating the model, we applied our algorithm to 48 datasets from five subjects in their daily life. After analyzing all the results, we noticed that most of the tremor activity was detected in both the accelerometers at the same time, although Apple Watch detected more tremors. Some activities which were classified as voluntary movement in IPG-based classification, were classified as tremor in Apple Watch. While analyzing the integral power in different frequency bands we predicted that Apple Watch might detect more tremors than IPG. Average tremor detection among all the datasets in IPG was 16.81%, while in Apple Watch the average tremor detection was 19.6%. Tremor was also detected in two datasets out of ten datasets for a very small amount of time in subject AC27 who never reported tremor. This proves that our algorithm can detect tremors which don't even last for long periods. We also compared the tremor detection for each subject with the clinical assessment which coincides with our results.

In all the datasets percentage of rest was higher in IPG-based classification and percentage of voluntary activity was higher in Apple Watch classification. Some activities which were classified as rest in IPG were classified as voluntary movement in Apple Watch. Again, this was due to the external location of the Apple Watch. Apple watch detected the arbitrary hand movement at the time of rest as voluntary activity.

Then we calculated the percentage of each activity to match the Apple Watch-based classification with the IPG-based classification. 83.27% of the tremor detected at the same time in both the accelerometers. Most of the mismatches were during the rest and voluntary activity. Rest activity match was 69.67% and voluntary activity match was 68.36% among all the datasets.

CHAPTER 7

Discussion - Implications for Closed-Loop DBS

In this thesis we analyzed the possibility of using an implanted accelerometer for providing feedback to the closed loop deep brain stimulation device. Implanted accelerometers have some advantage over externally worn wristwatches. For instance, an implanted accelerometer that is already integrated into the DBS device reduces the required number of devices to record data from patients. An implanted accelerometer can be used instead of using a wristwatch for providing feedback to the closed loop control system.

From our analysis, the implanted IPG accelerometer appears to detect postural rest activity more accurately than Apple Watch. Also, Apple Watch has noisier data transmission. Because of the different physical location of these two accelerometers, the recorded signal had differences, and consequently there were discrepant classification results. The Apple Watch was more sensitive to arm movements, and thus, arm/hand movements tend to be classified with the active voluntary movements, whereas the IPG only classifies movements of the whole body as active voluntary movements. If all these classifications are relevant to effective DBS, then the feedback signal could be a hybrid acceleration signal that somehow combines the information from both the internal and external accelerometer. The results of our analyses indicate that overall IPG-based classification was more accurate than the conventional Apple Watch.

Since our classification algorithm can detect tremor, rest, and voluntary activity etc., this algorithm can be deployed to a microcontroller which is integrated with the IPG. According to the classified physical state the stimulation parameters of the stimulator would be set in the IPG. For example, if the classified physical state is tremor, the stimulator will be stimulating. Otherwise, it will be in sleep mode. As a result, the battery life of the IPG will be improved. As our algorithm can detect high and low intensity tremors, the stimulation parameter would be set differently according to the tremor severity. Fig. 62 shows a feasible hardware implementation of our proposed closed-loop DBS system.

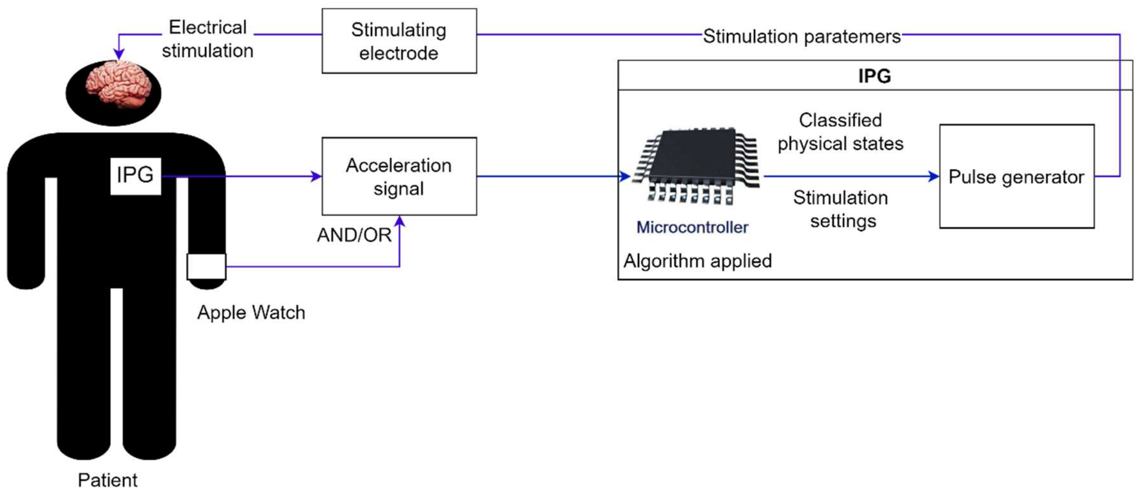


Figure 62. Hardware implementation of the classification algorithm.

In conclusion, we have validated our clustering algorithm to have 91% accuracy through the controlled experiment. After implementing the algorithm, patients don't need to go to the clinicians for manual settings of the stimulation parameters of the DBS device. Instead, they can be adaptively set according to the detected activity states of the patient. The patients can get rid of the tedious clinical visits and trial and error process of stimulation settings. The problems with the side effects for overstimulation or under stimulation can also be solved. Our adaptive closed-loop DBS algorithm can afford the maximal quality of life improvements of the Parkinson's disease patients.

According to the analysis, we can conclude that using hybrid acceleration from both Apple Watch and IPG accelerometer as feedback to the DBS device, the stimulation parameters could be set according to the physical states of the DBS users. Since IPG detects postural rest activity (when body is stationary, but there is some hand/arm movement) and active voluntary movement more accurately, DBS can take feedback from IPG classification to detect the activity as rest or voluntary activity and turn the DBS stimulation off. On the contrary, if the patient has tremor in hand, Apple Watch tremor detection could be used for setting the stimulation. We can use tremor detection from IPG either, since IPG tremor detection seems more accurate according to our analysis. To strongly say that IPG tremor detection is more accurate, we must wait for the controlled experiment on each patient. Our classification algorithm is also able to detect the intensity of tremor. According to the tremor intensity, the amount of stimulation to be supplied to the patient can also be set. Thus, instead of getting constant stimulation all the time regardless of the activity, the DBS user can get adaptive stimulation using our algorithm according to the physical state. For getting feedback, conventional Apple Watch can be completely replaced by the IPG, since IPG classification results seem more accurate than conventional Apple Watch.

Still there is potential for improvement of our algorithm, and we still hope to address a major limitation in our validation experiments. Although we have validated our model with a controlled experiment, the experiment was conducted on healthy subjects with no Parkinson's disease, and there was no use of IPG data. We planned controlled experiments to be conducted by our Duke collaborators on each of the five patient subjects. Each subject will be asked to perform activities like those used during our controlled experiments while at their clinic visits and while IPG and Apple watch accelerometry data are acquired. The Duke researcher will record the times of the different activities to provide ground truth data. With the control data from the IPG and Apple Watch, along with the timing information and the type of activities, we will apply our model to these data and compare with the ground truth. Then it would be possible to validate the model under much more comparable conditions to the condition under which DBS users would be using closed-loop DBS.

REFERENCES

[1] Prof Eduardo Tolosa MD, Gregor Wenning MD, Werner Poewe MD, "*The diagnosis of Parkinson's disease*", The Lancet Neurology, Volume 5, Issue 1, January 2006, Pages 75-86.

[2] Fasano, A.; Fung, V.S.C.; Lopiano, L.; Elibol, B.; Smolentseva, I.G.; Seppi, K.; Takáts, A.; Onuk, K.; Parra, J.C.; Bergmann, L.; et al. "*Characterizing Advanced Parkinson's Disease: OBSERVE-PD Observational Study Results of 2615 Patients"*. BMC Neurol. 2019, 19, 50.

[3] Kim, H.-J.; Mason, S.; Foltynie, T.; Winder-Rhodes, S.; Barker, R.A.; Williams Gray, "*C.H. Motor Complications in Parkinson's Disease: 13-Year Follow-up of the CamPaIGN Cohort. Mov. Disord"*. 2020, 35, 185–190.

[4] Nakamura Y. "*Problems of long-term levodopa therapy in Parkinson's disease"*. Nihon Rinsho. 1997 Jan;55(1):65-71. Japanese. PMID: 9014425.

[5] Joel S. Perlmutte and Jonathan W. Mink, "Deep brain stimulation", Annual Review of Neuroscience, Vol. 29:229-257 (Volume publication date: 21 July 2006)

[6] Martin Jakobs, Anton Fomenko, Andres M Lozano, Karl L Kiening, "*Cellular, molecular, and clinical mechanisms of action of deep brain stimulation—a systematic review on established indications and outlook on future developments*", EMBO Molecular Medicine, March 2019, PMID: 30862663.

[7] Lozano AM, Lipsman N, Bergman H, Brown P, Chabardes S, Chang JW, Matthews K, McIntyre CC, Schlaepfer TE, Schulder M, Temel Y, Volkmann J, Krauss JK. "*Deep brain stimulation: current challenges and future directions"*. Nat Rev Neurol. 2019 Mar;15(3):148-160. doi: 10.1038/s41582-018-0128-2. PMID:

30683913; PMCID: PMC6397644.

[8] NIH Research Matters, "*Deep Brain Stimulation Curbs Parkinson Symptoms*", National Institute of Health (NIH), January 12, 2009.

[9] Hariz M. "*Deep brain stimulation: new techniques. Parkinsonism Relat Disord*". 2014 Jan;20 Suppl 1: S192-6. doi: 10.1016/S1353-8020(13)70045-2. PMID: 24262179.

[10] Lozano AM, Lipsman N. "*Probing and regulating dysfunctional circuits using deep brain stimulation*". Neuron. 2013; 77:406–424.

[11] Odekerken VJ, van Laar T, Staal MJ, Mosch A, Hoffmann CF, Nijssen PC, et al. "*Subthalamic nucleus versus globus pallidus bilateral deep brain stimulation for advanced Parkinson's disease (NSTAPS study): a randomised controlled trial*". Lancet Neurol. 2013; 12:37–44. doi: 10.1016/S1474-4422(12)70264-8.

[12] Zrinzo L, van Hulzen AL, Gorgulho AA, Limousin P, Staal MJ, De Salles AA, et al. "*Avoiding the ventricle: a simple step to improve accuracy of anatomical targeting during deep brain stimulation*". J Neurosurg. 2009; 110:1283–90. doi: 10.3171/2008.12. JNS08885.

[13] de Koning PP, Figee M, van den Munckhof P, Schuurman PR, Denys D. "*Status of deep brain stimulation for obsessive-compulsive disorder: a clinical review of different targets*". Curr Psychiatry Rep. 2011; 13:274–82. doi: 10.1007/s11920 011-0200-8.

[14] S. Little, A. Pogosyan, S. Neal, B. Zavala, L. Zrinzo, M. Hariz, T. Foltynie, P. Limousin, K. Ashkan, J. Fitzgerald, A. L. Green, T. Z. Aziz and P. Brown, "*Adaptive Deep Brain Stimulation in advanced Parkinson Disease*," Annals of

Neurology, vol. 74, no. 3, pp. 449-457, 2013.

[15] Ghasemi P, Sahraee T, Mohammadi A. "*Closed- and Open-loop Deep Brain Stimulation: Methods, Challenges, Current and Future Aspects".* J Biomed Phys Eng. 2018 Jun 1;8(2):209-216. PMID: 29951448; PMCID: PMC6015649.

[16] Rosin B, Slovik M, Mitelman R, Rivlin-Etzion M, Haber SN, Israel Z, et al. "*Closed-loop deep brain stimulation is superior in ameliorating parkinsonism. Neuron"*. 2011; 72:370–84.

[17] Lettieri C, Rinaldo S, Devigili G, Pauletto G, Verriello L, Budai R, et al. "*Deep brain stimulation: subthalamic nucleus electrophysiological activity in awake and anesthetized patients"*. Clin Neurophysiol. 2012; 123:2406–13.

[18] P. Afshar, D. Moran, A. Rouse, W. Xuan, and T. Denison, "*Validation of chronic implantable neural sensing technology using electrocorticographic (ECoG) based brain machine interfaces,*" 2011 5th International IEEE/EMBS Conference on Neural Engineering, Cancun, 2011, pp. 704–707.

[19] S. E. Qasim, C. de Hemptinne, N. C. Swann, S. Miocinovic, J. L. Ostrem, and P. A. Starr, "*Electrocorticography reveals beta desynchronization in the basal ganglia cortical loop during rest tremor in Parkinson's disease*," Neurobiol Dis, vol. 86, pp. 177-186, 2//2016.

[20] R. Hyo-Gyuem, J. Jaehun, J. A. Fredenburg, S. Dodani, P. Patil, and M. P. Flynn, "*A wirelessly powered log-based closed-loop deep brain stimulation SoC with two way wireless telemetry for treatment of neurological disorders*," 2012 Symposium on VLSI Circuits (VLSIC), Honolulu, HI, 2012, pp. 70–71.

[21] Abosch A, Lanctin D, Onaran I, Eberly L, Spaniol M, Ince NF. "*Long-term*

recordings of local field potentials from implanted deep brain stimulation electrodes". Neurosurgery. 2012; 71:804–14.

[22] Swann N, Poizner H, Houser M, Gould S, Greenhouse I, Cai W, et al. "*Deep brain stimulation of the subthalamic nucleus alters the cortical profile of response inhibition in the beta frequency band: a scalp EEG study in Parkinson's disease"*. J Neurosci. 2011; 31:5721–9.

[23] Graupe D, Basu I, Tuninetti D, Vannemreddy P, Slavin KV. "*Adaptively controlling deep brain stimulation in essential tremor patient via surface electromyography"*. Neurol Res. 2010; 32:899–904.

[24] Chang SY, Kimble CJ, Kim I, Paek SB, Kressin KR, Boesche JB, et al. "*Development of the Mayo investigational Neuromodulation control system: toward a closed-loop electrochemical feedback system for deep brain stimulation"*. J Neurosurg. 2013; 119:1556–65.

[25] C.-H. Kuo, G. A. White-Dzuro and A. L. Ko, "*Approaches to closed-loop deep brain stimulation for movement disorders*," Neurosurgical Focus, vol. 45, no. 2, 2018.

[26] K. K. Sellers, R. Gilron, J. Anso, K. H. Louie, P. R. Shirvalkar, E. F. Chang, S. J. Little and P. A. Starr, "*Analysis-rcs-data: Open-Source Toolbox for the Ingestion, Time-Alignment, and Visualization of Sense and Stimulation Data from the Medtronic Summit RC+S System*", Frontiers in Human Neuroscience, vol. 15, 2021.

[27] J. V. D. Wardt, A. M. M. van der Stouwe, M. Dirkx et al., "*Systematic clinical approach for diagnosing upper limb tremor*," Journal of Neurology, Neurosurgery

& Psychiatry, vol. 91, no. 8, pp. 822–830, 2020.

[28] ActiGraph, LLC (2023). Retrieved from: https://actigraphcorp.com/actigraph-link/

[29] ActiGraph, LLC (2023). Retrieved from: https://actigraphcorp.com/actilife/

[30] Mostafa, Abeer & Barghash, Toka & Assaf, Asmaa & Gomaa, Walid. (2020). "*Multi-sensor Gait Analysis for Gender Recognition*". 629-636. 10.5220/0009792006290636.

[31] rune-labs (2021), jupyter notebook templates, Github repository: https://github.com/rune-labs/opensource/tree/master/jupyter-notebook-templates

[32] Erick A. Rojas-Torres (2022), Team flow: *"Adaptive deep brain stimulation for Parkinson's Disease application"* [Master's Thesis, California State University, Los Angeles].

[33] Pratap Dangeti. *Statistics for Machine Learning.* Packet Publishing, July 2017. ISBN: 9781788295758.

APPENDIX A

Main MATLAB Script file for Running the Data from the PD patients

This script was used to get all the figures and results in this thesis. It is separated into different sections.

Loading in and plotting raw data

```
clear all;
clc;

rng('default')
% data sets with gaps give sampling frequencies that are off, IPG
% should be at ~64 Hz and Apple Watch should  be ~50 Hz

path(path, 'E:\Thesis\DBSdata');

[ptPath] = uigetdir('Choose the patient data path to obtain the acceleration input
files)');
dateList = dir(ptPath);

nFiles = length(dateList) - 2
cd(ptPath);

% for i = 1:nFiles,
close all;
[filename, pathname] = uigetfile('*.csv', 'Choose the IPG acceleration data');
ius = strfind(filename, '_'); % index of underscores
ptID = filename(1:ius(1)-1)
date = filename(ius(1)+1: ius(2)-1)

IPGdata = datastore([pathname,filename],'Type','tabulartext');
IPGdata = readall(IPGdata);
IPGdata = IPGdata.Variables;

tIPG = IPGdata(:,2);
tConvIPG = datetime(tIPG,'ConvertFrom','epochtime','Epoch','1970-01-
01','TicksPerSecond',1,'TimeZone','UTC');
tConvIPG.TimeZone = 'America/New_York';
IPG_duration = tConvIPG(end) - tConvIPG(1)

IPGaccel = IPGdata(:,3:end);
nCh = size(IPGaccel, 2);

filename = [filename(1:ius(2)) 'apple_accel.csv']

dataApple= datastore([pathname,filename],'Type','tabulartext');
dataApple = readall(dataApple);
dataApple = dataApple.Variables;
```

```
tApp = dataApple(:,2);
tConvApp = datetime(tApp,'ConvertFrom','epochtime','Epoch','1970-01-
01','TicksPerSecond',1,'TimeZone','UTC');
tConvApp.TimeZone = 'America/New_York';
Apple_duration = tConvApp(end) - tConvApp(1)

Appleaccel= dataApple(:,3:end);
fsIPG = round(1/mean(seconds(diff(tConvIPG))))
fsApple = round(1/mean(seconds(diff(tConvApp))))

axis_labels = {'a_x' 'a_y' 'a_z'};
channels = size(IPGaccel,2);

figure('Name', 'Raw IPG acceleration')
for i = 1: channels
    subplot(channels,1,i)
    plot(tConvIPG, IPGaccel(:,i))
    ylabel([axis_labels{i}]);
end
sgtitle('IPG acceleration')

figure('Name', 'Raw Apple acceleration')
for i = 1: channels
    subplot(channels,1,i)
    plot(tConvApp, Appleaccel(:,i))
    ylabel([axis_labels{i}]);
end
sgtitle('Apple acceleration')

figure('Name', 'Raw acceleration overlaps')
for i = 1: channels
    subplot(channels,1,i)
    plot(tConvApp, Appleaccel(:,i));hold on
    plot(tConvIPG, IPGaccel(:,i))
    ylabel([axis_labels{i}]);
    legend('Apple Watch','IPG')
end
sgtitle('Raw overlapping acceleration data')
```

Data selection

If there is a gap in the datasets (IPG and Apple watch aren't 64 and 50 Hz respectively)

then select the longest data segment in the data set

```
data_select = input('Do you want to select the overlapping data segment? Y/N:\n','s')

if upper(data_select) == 'Y'
    close all
    figure;
    for i = 1: channels
        subplot(channels,1,i)
        plot(tApp, Appleaccel(:,i),'Color',[0,0,1,0.025]);hold on
        plot(tIPG, IPGaccel(:,i),'Color',[1,0,0,0.95])
        axis([-inf inf -1 1])
        legend('Apple Watch','IPG')
    end
    sgtitle('Raw overlapping acceleration data')

    disp('Select start time, then end time');
    [selectTimes, y] = ginput(2);
    selectTimes;

    IPGstart = sum(tIPG< selectTimes(1));
    IPGend = sum(tIPG< selectTimes(2));

    tIPG = tIPG(IPGstart+1:IPGend);
    IPGaccel = IPGaccel(IPGstart+1:IPGend,:);

    tConvIPG = datetime(tIPG,'ConvertFrom','epochtime','Epoch','1970-01-
01','TicksPerSecond',1,'TimeZone','UTC');
    tConvIPG.TimeZone = 'America/New_York';
    IPG_duration = tConvIPG(end) - tConvIPG(1)
    fsIPG = round(1/mean(seconds(diff(tConvIPG))))

    Applestart = sum(tConvApp< tConvIPG(1)) + 1;
    Appleend = sum(tConvApp< tConvIPG(end));

    tConvApp = tConvApp(Applestart:Appleend);
    fsApple = round(1/mean(seconds(diff(tConvApp))))
    Apple_duration = tConvApp(end) - tConvApp(1)
    Appleaccel = Appleaccel(Applestart:Appleend,:);

    close all
    figure('Name','New raw acceleration data segment');
    for i = 1: channels
        subplot(channels,1,i)
        plot(tConvApp, Appleaccel(:,i));hold on
        plot(tConvIPG, IPGaccel(:,i))
        ylabel([axis_labels{i}]);
        legend('Apple Watch','IPG')
    end
    sgtitle('New raw acceleration data segment')

else
    disp('You picked no')
```

```
%     return
end
```

Interpolating and plotting on synced axis

```
fs = round(1/mean(seconds(diff(tConvIPG))));
if tConvIPG(end)<= tConvApp(end)
    tSync = timeScaling(tConvIPG, fs);
else
    tSync = timeScaling(tConvApp, fs);
end
fs_synced = round(1/mean(seconds(diff(tSync))))

[accelApp, L] = syncData(Appleaccel, tConvApp, tSync, 1);
[accelIPG, L] = syncData(IPGaccel, tConvIPG, tSync, 0);
accelApp(isnan(accelApp))=0;
accelIPG(isnan(accelIPG))=0;

figure('Name', 'Overlapping data with synced axis')
for i = 1:3
    subplot(3,1,i)
    plot(tSync, accelApp(:,i))
    hold on;
    plot(tSync, accelIPG(:,i))
    ylabel(axis_labels{i},'FontSize',16);
    legend('apple','ipg')
end
sgtitle('Overlapping data with synced axis')
```

STFT of acceleration signal using different time windows

Calculate the STFT of the accel signals and plot, also find the acceleration integral power and plot its different bands (IPG/Apple_result)

```
twindows = [10]; % 20 seconds, 1 minute, 10 minutes
win_size = length(twindows);

band = [0 4 7 12 32];
titles = {'0-4 Hz', '4-7 Hz', '7-12 Hz', '12-32 Hz'};
IPG_results = [];
Apple_results = [];
% fIPG= (1:Nfft)*fsIPG/Nfft - fsIPG/2 ;
for window = 1:win_size;
    IPG_result = [];
    Apple_result = [];

    e = nextpow2(twindows(window)*fs); % changes nfft value acccording to time window
    Nfft = 2^e;
    twin = twindows(window); % window size
    Lwin = round(twin*fs);
    win = hanning(Lwin);
    Noverlap = round(0.5*Lwin);

    figure('Name', ['Synced IPG with window size ' num2str(twindows(window)) ' seconds'])
    for j = 1:channels;
        subplot(channels,1,j)
        [IPG_stft_synced, FIPG_synced] = stft(accelIPG(:,j),fs, 'Window', win,
'FFTLength', Nfft, 'OverlapLength', Noverlap);
        df = mean(diff(FIPG_synced)); % use with trapz, trapz*df
        start = sum(FIPG_synced<2);
        imagesc((1:(length(accelIPG(:,j))/fs)-1), FIPG_synced(start:end),
20*log10(abs(IPG_stft_synced(start:end,:))));
        ylabel([axis_labels{j}],'FontSize',16);
        IPG.stft{j} = IPG_stft_synced;

        for k = 1:length(band)-1;
            IPGband = find(FIPG_synced >=band(k) & FIPG_synced <=band(k+1));    % looks
for all the content in the specific freq. band
            BW = band(k+1)-band(k); % bandwidth of frequency band

            t_stft = abs((IPG_stft_synced));
            dist = trapz(t_stft(IPGband,:));
            IPG_result = [IPG_result; dist*df/BW];  % integral power of IPG acceleration

        end

    end

    IPG_results.window{window} = IPG_result;
    sgtitle(['Synced IPG with window size ' num2str(twindows(window)) ' seconds'])

    for j = 1:4 % 4 different bands, 1 figure for each band
        figure('Name', ['IPG Window size ' num2str(twindows(window)) ', ' titles{j} '
band'])
        for k = 1:channels  % 3 axis (x y z)
            subplot(channels,1,k);
            plot(IPG_results.window{window}((k-1)*4+j,:))
            ylabel(axis_labels{k})
        end
        sgtitle(['IPG Window size ' num2str(twindows(window)) ', ' titles{j} ' band'])
    end

    figure('Name', ['Synced Apple with window size ' num2str(twindows(window)) '
seconds'])
    for j = 1:channels;
        subplot(channels,1,j)
        [Apple_stft_synced, FApple_synced] = stft(accelApp(:,j),fs, 'Window', win,
'FFTLength', Nfft, 'OverlapLength', Noverlap);
        start = sum(FApple_synced<2);
        df = mean(diff(FApple_synced));
        imagesc((1:(length(accelApp(:,j))/fs)-1), FApple_synced(start:end),
20*log10(abs(Apple_stft_synced(start:end,:))));
```

```
        ylabel([axis_labels{j}],'FontSize',16);

        for k = 1:length(band)-1;
            Appleband = find(FApple_synced >=band(k) & FApple_synced <=band(k+1));
            BW = band(k+1)-band(k);

            t_stft = abs((Apple_stft_synced));
            y = trapz(t_stft(Appleband,:));
            Apple_result = [Apple_result; y*df/BW];

        end
    end
    Apple_results.window{window} = Apple_result;
    sgtitle(['Synced Apple with window size ' num2str(twindows(window)) ' seconds'])

    for j = 1:4,
        figure('Name', ['Apple Window size ' num2str(twindows(window)) ', ' titles{j} '
band'])
        for k = 1:channels,
            subplot(channels,1,k);
            plot(Apple_results.window{window}((k-1)*4+j,:))
            ylabel(axis_labels{k})
        end
        sgtitle(['Apple Window size ' num2str(twindows(window)) ', ' titles{j} ' band'])
    end

end
```

Plotting envelopes and calculating correlation coefficients

```
aIPG = sqrt(sum(accelIPG.^2, 2));   % acceleration magnitude
aIPG(find(isnan(aIPG))) = 0;

aApple = sqrt(sum(accelApp.^2, 2)); % acceleration magnitude
aApple(find(isnan(aApple))) = 0;

twindows = [10, 60, 600];   % 20 seconds, 1 minute, 10 minutes
titles = {'10 sec' '1 min' '10 min'};
rs = [];

figure('Name', 'Envelopes of acceleration signal');
for iEnv = 1:3;
    twin = twindows(iEnv); % window for moving average in seconds
    envIPG = env(aIPG-1, twin, fs);
    envApp = env(aApple-1, twin, fs);

    subplot(channels,1,iEnv);
    plot(tSync, envIPG);
    hold on;
    plot(tSync, envApp);
    ylabel('G', 'FontSize', 12);
    title(titles{iEnv},'FontSize',12);
    legend('apple','ipg');
    %pause;

    envIPG(isnan(envIPG))=0;
    envApp(isnan(envApp))=0;
    r = round(corrcoef([envIPG envApp]),2);
    rs = [rs r(2)];
end
rs
```

Performing elbow method to find optimal clusters

```
clusters = 10;
ipg_distortion_array = [];
for clust = 1:clusters
    [ipg_idx c] = kmeans((IPG_result'),clust,'Replicate',5);
    d = c';
    for j = 1:length(unique(ipg_idx))
        x_array = [];
        idx_array = IPG_result(:,find(ipg_idx==j));
        for k = 1:size(idx_array,2)
            x = idx_array(:,k)-d(:,j);
            x_array = [x_array x];
        end
        %          size(x_array)
        t = x_array.^2;
        y = sum(t,2);
        %          size(t)
        distortion = sqrt(y);
        distortion = mean(distortion);
    end
    ipg_distortion_array = [ipg_distortion_array distortion];
end
figure('Name', 'Elbow Method for IPG')
plot(1:clusters,ipg_distortion_array,'-xr')
xlabel('Number of clusters')
ylabel('Distortion')
title('Elbow method for IPG')

apple_distortion_array = [];
for clust = 1:clusters
    [apple_idx c] = kmeans((Apple_result'),clust,'Replicate',5);
    d = c';
    for j = 1:length(unique(apple_idx))
        x_array = [];
        idx_array = Apple_result(:,find(apple_idx==j));
        for k = 1:size(idx_array,2)
            x = idx_array(:,k)-d(:,j);
            x_array = [x_array x];
        end
        %          size(x_array)
        t = x_array.^2;
        y = sum(t,2);
        %         size(t)
        distortion = sqrt(y);
        distortion = mean(distortion);
    end
    apple_distortion_array = [apple_distortion_array distortion];
end

figure('Name', 'Elbow Method for Apple Watch')
plot(1:clusters,apple_distortion_array,'-xr')
xlabel('Number of clusters')
ylabel('Distortion')
title('Elbow method for Apple Watch')
```

Finding optimal number of clusters

```
y = ipg_distortion_array - min(ipg_distortion_array) + 0.2; % IPG
x = [1:10; ones([1, 10])]';
[coef,BINT,R,RINT,STATS] = regress(log(y'), x);
A = exp(coef(2));
clus = [1:10];
B = mean(ipg_distortion_array(7:end));
yhat = A*exp(coef(1)*clus) + B;
figure('Name', 'Optimum number of cluster IPG');
plot(clus, ipg_distortion_array, 'r')
hold on;
plot(clus, yhat, 'b');
xlabel('number of cluster'); ylabel('distortion array'); title('Optimum number of cluster
IPG');
legend('distortion_array', 'yhat')
cluster_IPG = knee_pt(yhat, clus)

% Apple Watch
y = apple_distortion_array - min(apple_distortion_array) + 0.2;
x = [1:10; ones([1, 10])]';
[coef,BINT,R,RINT,STATS] = regress(log(y'), x);
A = exp(coef(2));
clus = [1:10];
B = mean(apple_distortion_array(7:end));
yhat = A*exp(coef(1)*clus) + B;
figure('Name', 'Optimum number of cluster Apple Watch');
plot(clus, apple_distortion_array, 'r')
hold on;
plot(clus, yhat, 'b')
xlabel('number of cluster'); ylabel('distortion array'); title('Optimum number of cluster
Apple Watch');
legend('distortion array', 'yhat')
cluster_apple = knee_pt(yhat, clus)
```

Clustering using optimal k clusters

```
clusters = cluster_IPG;
[ipg_idx c] = kmeans(log(IPG_result'),clusters,'Replicate',5);
for clust = 1:clusters
    idx_array = IPG_result(:,find(ipg_idx==clust));
    figure('Name',['STFT of IPG cluster ' num2str(clust)])
    imagesc(idx_array)
    title(['IPG cluster ' num2str(clust)])
    set(gca, 'YTick', [1:12]);
    labels = {'x,0-4Hz', 'x,4-7Hz', 'x,7-12Hz', 'x,12-32Hz', 'y,0-4Hz', 'y,4-7Hz', 'y,7-
12Hz', 'y,12-32Hz', 'z,0-4Hz', 'z,4-7Hz', 'z,7-12Hz', 'z,12-32Hz'}
    set(gca,'YTickLabel', labels);
end
clusters = cluster_apple;
[apple_idx c] = kmeans(log(Apple_result'),clusters,'Replicate',5);
for clust = 1:clusters
    idx_array = Apple_result(:,find(apple_idx==clust));
    figure('Name',['STFT of Apple Watch cluster ' num2str(clust)])
    imagesc(idx_array)
    title(['Apple Watch cluster ' num2str(clust)])
    set(gca, 'YTick', [1:12]);
    labels = {'x,0-4Hz', 'x,4-7Hz', 'x,7-12Hz', 'x,12-32Hz', 'y,0-4Hz', 'y,4-7Hz', 'y,7-
12Hz', 'y,12-32Hz', 'z,0-4Hz', 'z,4-7Hz', 'z,7-12Hz', 'z,12-32Hz'}
    set(gca,'YTickLabel', labels);
end
```

Look at the clusters in pc space

```
pt_markers = ['r.'; 'g.'; 'b.'; 'k.';'m.';'y.'];

[IPG_coeff IPG_score] = pca(IPG_result');

figure('Name', 'IPG results in PC space')
plot(IPG_score(:,1),IPG_score(:,2),'.c')
title('IPG results in PC space')
xlabel('PC 1')
ylabel('PC 2')

figure('Name', 'Stem plot of IPG coefficients 1 and 2')
subplot(2,1,1)
stem(IPG_coeff(:,1))
title('PC 1')
subplot(2,1,2)
stem(IPG_coeff(:,2))
title('PC 2')
sgtitle('IPG coefficients 1 and 2 stem plot')
clusters = cluster_IPG;
figure('Name', 'IPG results in PC space color coded')
for iclust = 1:clusters,
    hold on;
    plot(IPG_score(find(ipg_idx==iclust), 1), IPG_score(find(ipg_idx==iclust), 2),
pt_markers(iclust,:), 'LineWidth', 1);
end
title('IPG results in PC space')
xlabel('PC 1')
ylabel('PC 2')
legend('Clust 1', 'Clust 2', 'Clust 3','Clust 4', 'clust 5', 'clust 6')

[Apple_coeff Apple_score] = pca(Apple_result');

figure('Name', 'Apple Watch results in PC space')
plot(Apple_score(:,1),Apple_score(:,2),'.c')
title('Apple results in PC space')
xlabel('PC 1')
ylabel('PC 2')

figure('Name', 'Stem plot of Apple Watch coefficients 1 and 2')
subplot(2,1,1)
stem(Apple_coeff(:,1))
title('PC 1')
subplot(2,1,2)
stem(Apple_coeff(:,2))
title('PC 2')
sgtitle('Apple coefficients 1 and 2 stem plot')

figure('Name', 'Apple Watch results in PC space color coded')
clusters = cluster_apple;
for iclust = 1:clusters,
    hold on;
    plot(Apple_score(find(apple_idx==iclust), 1), Apple_score(find(apple_idx==iclust),
2), pt_markers(iclust,:), 'LineWidth', 1);
end
title('Apple results in PC space')
xlabel('PC 1')
ylabel('PC 2')
legend('Clust 1', 'Clust 2', 'Clust 3','Clust 4', 'clust 5', 'clust 6')
```

Plot synced acceleration color coded by clusters

Synced acceleration data are named accelApp and accelIPG use ipg_idx and apple_idx, 3

Clusters

```
idx_color = ['r', 'g', 'b','k', 'm', 'y'];
Delta = Lwin - Noverlap; % number of samples between each time block of the stft

figure('Name', 'Synced IPG acceleration color coded')
nBlocks = floor(length(accelIPG)/Delta);
tSyncB = reshape(tSync(1:nBlocks*Delta), Delta, nBlocks); % organized in time blocks used
for stft and getting apple_idx
```

```
clusters = cluster_IPG;
for iclust = 1:clusters
    iThisClust  = find(ipg_idx == iclust)+1;
    iThisClust = iThisClust(iThisClust <= nBlocks);
    for j = 1:channels
        subplot(3,1,j)
        accelIPGB = reshape(accelIPG(1:nBlocks*Delta, j), Delta, nBlocks); % organized in
time blocks corresponding to stft and used for getting apple_idx
        plot(tSyncB(:, iThisClust),accelIPGB(:, iThisClust),idx_color(iclust));  hold on
        ylabel([axis_labels{j}],'FontSize',16);
    end
end
sgtitle('synced IPG accel colored by idx classification')

figure('Name','Synced Apple acceleration color coded')
nBlocks = floor(length(accelApp)/Delta);
tSyncB = reshape(tSync(1:nBlocks*Delta), Delta, nBlocks); % organized in time blocks used
for stft and getting apple_idx
clusters = cluster_apple;
for iclust = 1:clusters
    iThisClust  = find(apple_idx == iclust)+1;
    iThisClust = iThisClust(iThisClust <= nBlocks);
    for j = 1:channels
        subplot(3,1,j)
        accelAppB = reshape(accelApp(1:nBlocks*Delta, j), Delta, nBlocks); % organized in
time blocks corresponding to stft and used for getting apple_idx
        plot(tSyncB(:, iThisClust),accelAppB(:, iThisClust),idx_color(iclust));  hold on
        ylabel([axis_labels{j}],'FontSize',16);
    end
end
sgtitle('synced Apple accel colored by idx classification')
```

Plotting tremor band over classified acceleration signals

"tremor" band 4-7 Hz band are the 2nd,6th and 10th rows in the matrices "Apple_result" and "IPG_result"

```
IPG_tremor_band = IPG_result([2,6,10],:);    % IPG tremor band
Apple_tremor_band = Apple_result([2,6,10], :);   % Apple Watch tremor band

tTremor = tSync((Lwin-Noverlap):(Lwin-Noverlap):end);
tTremor = tTremor(1:end-1);
Delta = Lwin - Noverlap; % number of samples between each time block of the stft

figure('Name','Classified IPG acceleration with tremor band')
nBlocks = floor(length(accelIPG)/Delta);
tSyncB = reshape(tSync(1:nBlocks*Delta), Delta, nBlocks); % organized in time blocks used
for stft and getting ipg_idx
clusters = cluster_IPG;
for iclust = 1:clusters
```

```
    iThisClust  = find(ipg_idx == iclust)+1;
    iThisClust = iThisClust(iThisClust <= nBlocks);

    for j = 1:channels
        subplot(3,1,j)
        accelIPGB = reshape(accelIPG(1:nBlocks*Delta, j), Delta, nBlocks); % organized in
time blocks corresponding to stft and used for getting apple_idx
        plot(tSyncB(:, iThisClust),accelIPGB(:, iThisClust),idx_color(iclust));  hold on
        plot(tTremor,IPG_tremor_band(j,:),'Color',[0.9290 0.6940 0.1250]);
        ylabel([axis_labels{j}],'FontSize',16);
    end
end
sgtitle('Classified IPG acceleration w/ tremor band')
trem_clust_ipg = input('which cluster is "tremor" in the ipg classification? r(1), g(2),
b(3), k(4), m(5), y(6):\n ');
low_trem_clust_ipg = input('which cluster is "low tremor" in the ipg classification?
r(1), g(2), b(3), k(4), m(5), y(6):\n ');
rest_clust_ipg = input('which cluster is "rest" in the ipg classification? r(1), g(2),
b(3), k(4), m(5), y(6):\n ');
activity_clust_ipg = input('which cluster is "voluntary activity" in the ipg
classification? r(1), g(2), b(3), k(4), m(5), y(6):\n ');
low_activity_clust_ipg = input('which cluster is "low voluntary activity" in the ipg
classification? r(1), g(2), b(3), k(4), m(5), y(6):\n ');

figure('Name','Classified Apple acceleration with tremor band')
nBlocks = floor(length(accelApp)/Delta);
tSyncB = reshape(tSync(1:nBlocks*Delta), Delta, nBlocks); % organized in time blocks used
for stft and getting apple_idx
clusters = cluster_apple;
for iclust = 1:clusters
    iThisClust  = find(apple_idx == iclust)+1;
    iThisClust = iThisClust(iThisClust <= nBlocks);
    for j = 1:channels
        subplot(3,1,j)
        accelAppB = reshape(accelApp(1:nBlocks*Delta, j), Delta, nBlocks); % organized in
time blocks corresponding to stft and used for getting apple_idx
        plot(tSyncB(:, iThisClust),accelAppB(:, iThisClust),idx_color(iclust));  hold on
        plot(tTremor,Apple_tremor_band(j,:),'Color',[0.9290 0.6940 0.1250]);
        ylabel([axis_labels{j}],'FontSize',16);
    end
end
sgtitle('Classified Apple acceleration w/ tremor band')
trem_clust_apple = input('which cluster is "tremor" in the Apple classification? r(1),
g(2), b(3), k(4), m(5), y(6):\n ');
low_trem_clust_apple = input('which cluster is "low tremor" in the apple classification?
r(1), g(2), b(3), k(4), m(5), y(6):\n ');
rest_clust_apple = input('which cluster is "rest" in the Apple classification? r(1),
g(2), b(3), k(4), m(5), y(6):\n ');
activity_clust_apple = input('which cluster is "high voluntary activity" in the Apple
classification? r(1), g(2), b(3), k(4), m(5), y(6):\n ');
low_activity_clust_apple = input('which cluster is "low voluntary activity" in the Apple
classification? r(1), g(2), b(3), k(4), m(5), y(6):\n ');
```

Comparing IPG and Apple classification

Comparing Apple Watch based classification with IPG based classification

```
% finding matches for High tremor

TPt =0; % true matches for tremor
FPt=0; % False matches for tremor
FNt=0; % False matches for tremor
TNt=0; % true matches for tremor

apple_tremor_clust = zeros(length(apple_idx),1);
for j = 1:length(apple_idx)
    if apple_idx(j) == trem_clust_apple
        apple_tremor_clust(j) = 3;
    end
end
tremor_apple = size(nonzeros(apple_tremor_clust), 1);

IPG_tremor_clust = zeros(length(ipg_idx),1);
for j = 1:length(ipg_idx)
    if ipg_idx(j) == trem_clust_ipg
        IPG_tremor_clust(j) = 3;
    end
end
tremor_ipg = size(nonzeros(IPG_tremor_clust), 1);

for j = 1:length(ipg_idx)
     if (apple_tremor_clust(j)==3) && (IPG_tremor_clust(j)==3)
         TPt = TPt+1;
     elseif (apple_tremor_clust(j) ~=3) && (IPG_tremor_clust(j)==3)
         FNt = FNt+1;
     elseif (apple_tremor_clust(j)==3) && (IPG_tremor_clust(j)~=3)
         FPt = FPt+1;
     elseif (apple_tremor_clust(j)~=3) && (IPG_tremor_clust(j)~=3)
         TNt = TNt+1;
     end
end

% finding matches for Low tremor

TP_lt =0; % true matches for tremor
FP_lt=0; % False matches for tremor
FN_lt=0; % False matches for tremor
TN_lt=0; % true matches for tremor

apple_low_tremor_clust = zeros(length(apple_idx),1);
for j = 1:length(apple_idx)
    if apple_idx(j) == low_trem_clust_apple
        apple_low_tremor_clust(j) = 3;
    end
end
```

```
low_tremor_apple = size(nonzeros(apple_low_tremor_clust), 1);

IPG_low_tremor_clust = zeros(length(ipg_idx),1);
for j = 1:length(ipg_idx)
    if ipg_idx(j) == low_trem_clust_ipg
        IPG_low_tremor_clust(j) = 3;
    end
end
low_tremor_ipg = size(nonzeros(IPG_low_tremor_clust), 1);

for j = 1:length(ipg_idx)
     if (apple_low_tremor_clust(j)==3) && (IPG_low_tremor_clust(j)==3)
         TP_lt = TP_lt+1;
     elseif (apple_low_tremor_clust(j) ~=3) && (IPG_low_tremor_clust(j)==3)
         FN_lt = FN_lt+1;
     elseif (apple_low_tremor_clust(j)==3) && (IPG_low_tremor_clust(j)~=3)
         FP_lt = FP_lt+1;
     elseif (apple_low_tremor_clust(j)~=3) && (IPG_low_tremor_clust(j)~=3)
         TN_lt = TN_lt+1;
     end
end

% finding matches for voluntary activity

TPv=0; % true matches for voluntary activity
FPv=0; % False matches for voluntary activity
FNv=0; % False matches for voluntary activity
TNv=0; % true matches for voluntary activity

apple_activity_clust = zeros(length(apple_idx),1);
for j = 1:length(apple_idx)
    if apple_idx(j) == activity_clust_apple
        apple_activity_clust(j) = 2;
    end
end
activity_apple = size(nonzeros(apple_activity_clust), 1);

IPG_activity_clust = zeros(length(ipg_idx),1);
for j = 1:length(ipg_idx)
    if ipg_idx(j) == activity_clust_ipg
        IPG_activity_clust(j) = 2;
    end
end
activity_ipg = size(nonzeros(IPG_activity_clust), 1);

for j = 1:length(ipg_idx)
     if (apple_activity_clust(j)==2) && (IPG_activity_clust(j)==2)
         TPv = TPv+1;
     elseif (apple_activity_clust(j) ~=2) && (IPG_activity_clust(j)==2)
         FNv = FNv+1;
     elseif (apple_activity_clust(j)==2) && (IPG_activity_clust(j)~=2)
         FPv = FPv+1;
     elseif (apple_activity_clust(j)~=2) && (IPG_activity_clust(j)~=2)
         TNv = TNv+1;
     end
end

% finding matches for lower voluntary activity

TP_lv=0; % true matches for voluntary activity
FP_lv=0; % False matches for voluntary activity
FN_lv=0; % False matches for voluntary activity
TN_lv=0; % true matches for voluntary activity

apple_low_activity_clust = zeros(length(apple_idx),1);
for j = 1:length(apple_idx)
    if apple_idx(j) == low_activity_clust_apple
        apple_low_activity_clust(j) = 2;
    end
end
low_activity_apple = size(nonzeros(apple_low_activity_clust), 1);

IPG_low_activity_clust = zeros(length(ipg_idx),1);
for j = 1:length(ipg_idx)
    if ipg_idx(j) == low_activity_clust_ipg
        IPG_low_activity_clust(j) = 2;
    end
end
low_activity_ipg = size(nonzeros(IPG_low_activity_clust), 1);

for j = 1:length(ipg_idx)
     if (apple_low_activity_clust(j)==2) && (IPG_low_activity_clust(j)==2)
         TP_lv = TP_lv+1;
     elseif (apple_low_activity_clust(j) ~=2) && (IPG_low_activity_clust(j)==2)
         FN_lv = FN_lv+1;
     elseif (apple_low_activity_clust(j)==2) && (IPG_low_activity_clust(j)~=2)
         FP_lv = FP_lv+1;
     elseif (apple_low_activity_clust(j)~=2) && (IPG_low_activity_clust(j)~=2)
         TN_lv = TN_lv+1;
     end
end

% finding matches for rest

TPr=0; % true matches for rest
FPr=0; % False matches for rest
FNr=0; % False matches for rest
TNr=0; % true matches for rest

apple_rest_clust = zeros(length(apple_idx),1);
for j = 1:length(apple_idx)
    if apple_idx(j) == rest_clust_apple
        apple_rest_clust(j) = 1;
    end
end
rest_apple = size(nonzeros(apple_rest_clust), 1);

IPG_rest_clust = zeros(length(ipg_idx),1);
```

```
for j = 1:length(ipg_idx)
    if ipg_idx(j) == rest_clust_ipg
        IPG_rest_clust(j) = 1;
    end
end
rest_ipg = size(nonzeros(IPG_rest_clust), 1);

for j = 1:length(ipg_idx)
     if (apple_rest_clust(j)==1) && (IPG_rest_clust(j)==1)
         TPr = TPr+1;
     elseif (apple_rest_clust(j) ~=1) && (IPG_rest_clust(j)==1)
         FNr = FNr+1;
     elseif (apple_rest_clust(j)==1) && (IPG_rest_clust(j)~=1)
         FPr = FPr+1;
     elseif (apple_rest_clust(j)~=1) && (IPG_rest_clust(j)~=1)
         TNr = TNr+1;
     end
end
% Total true positive and false detection rate
total_TP = TPt+TP_lt+TPr+TPv+TP_lv;
total_FN = FNt+FN_lt+FNr+FNv+FN_lv;
total_FP = FPt+FP_lt+FPr+FPv+FP_lv;
total_TN = TNt+TN_lt+TNr+TNv+TN_lv;

% percentage of activities
percentage_tremor_ipg =
tremor_ipg/(tremor_ipg+low_tremor_ipg+rest_ipg+activity_ipg+low_activity_ipg) *100
percentage_low_tremor_ipg =
low_tremor_ipg/(tremor_ipg+low_tremor_ipg+rest_ipg+activity_ipg+low_activity_ipg) *100
percentage_rest_ipg =
rest_ipg/(tremor_ipg+low_tremor_ipg+rest_ipg+activity_ipg+low_activity_ipg) *100
percentage_activity_ipg =
activity_ipg/(tremor_ipg+low_tremor_ipg+rest_ipg+activity_ipg+low_activity_ipg) *100
percentage_low_activity_ipg =
low_activity_ipg/(tremor_ipg+low_tremor_ipg+rest_ipg+activity_ipg+low_activity_ipg) *100

percentage_tremor_apple =
tremor_apple/(tremor_apple+low_tremor_apple+rest_apple+activity_apple+low_activity_apple)
*100
percentage_low_tremor_apple =
low_tremor_apple/(tremor_apple+low_tremor_apple+rest_apple+activity_apple+low_activity_ap
ple) *100
percentage_rest_apple =
rest_apple/(tremor_apple+low_tremor_apple+rest_apple+activity_apple+low_activity_apple)
*100
percentage_activity_apple =
activity_apple/(tremor_apple+low_tremor_apple+rest_apple+activity_apple+low_activity_appl
e) *100
percentage_low_activity_apple =
low_activity_apple/(tremor_apple+low_tremor_apple+rest_apple+activity_apple+low_activity_
apple) *100
```

APPENDIX B

Main MATLAB Script file for Running the Validation Experimental Data

Loading and plotting raw data

```
clear;
clc;
close all;

%rng('default')
% data sets with gaps give sampling frequencies that are off, ActiGraph
% should be at ~70 Hz and Apple Watch should  be ~70 Hz

path(path,'E:\Thesis\DBSdata\REAL_EXPERIMENT_DATA');

[filename, pathname] = uigetfile('*.csv', 'Choose the Acti acceleration data');

Actidata = csvread([pathname filename], 11,0);

fsActi =70;
L_acti = size(Actidata,1);
tActi = [0:(L_acti-1)]/fsActi;

tConvActi =  tActi;
Actiaccel = Actidata(:,1:end);
nCh = size(Actiaccel, 2);

% read Apple watch data
[filename, pathname] = uigetfile('*.csv', 'Choose the Apple acceleration data');


dataApple= csvread([pathname filename], 1,26);

fs_apple = 70;
L_apple = size(dataApple,1);
tApp = [0:(L_apple-1)]/fs_apple;
tConvApp = tApp;
Appleaccel= dataApple(:,1:end);

axis_labels = {'a_x' 'a_y' 'a_z'};
channels = size(Actiaccel,2);

figure('Name', 'Raw ActiGraph acceleration')
for i = 1: channels
    subplot(channels,1,i)
    plot(tConvActi/60, Actiaccel(:,i))
    ylabel([axis_labels{i}]);
end
sgtitle('ActiGraph acceleration')

figure('Name', 'Raw Apple acceleration')
```

```
for i = 1: channels
    subplot(channels,1,i)
    plot(tConvApp/60, Appleaccel(:,i))
    ylabel([axis_labels{i}]);
end
sgtitle('Apple acceleration')

figure('Name', 'Raw acceleration overlaps')
for i = 1: channels
    subplot(channels,1,i)
    plot(tConvApp/60, Appleaccel(:,i));hold on
    plot(tConvActi/60, Actiaccel(:,i))
    ylabel([axis_labels{i}]);
    legend('Apple Watch','Acti')
end
sgtitle('Raw overlapping acceleration data')
```

STFT of acceleration signal using different time windows

Calculate the STFT of the accel signals and plot, also find the acceleration integral power and plot its different bands (Acti/Apple_result)

```
fs = 70;
twindows = [10]; % 10s, 20 seconds, 1 minute, 10 minutes
win_size = length(twindows);

band = [0 4 7 12 32];
titles = {'0-4 Hz', '4-7 Hz', '7-12 Hz', '12-32 Hz'};
Acti_results = [];
Apple_results = [];
% fActi= (1:Nfft)*fsActi/Nfft - fsActi/2 ;
for window = 1:win_size;
    Acti_result = [];
    Apple_result = [];

    e = nextpow2(twindows(window)*fs); % changes nfft value acccording to time window
    Nfft = 2^e;

    twin = twindows(window); % window size
    Lwin = round(twin*fs);
    win = hanning(Lwin);
    Noverlap = round(0.5*Lwin);

    figure('Name', ['Synced ActiGraph with window size ' num2str(twindows(window)) '
seconds'])
    for j = 1:channels;
        subplot(channels,1,j)
        [Acti_stft_synced, FActi_synced] = stft(Actiaccel(:,j),fs, 'Window', win,
'FFTLength', Nfft, 'OverlapLength', Noverlap);
        df = mean(diff(FActi_synced)); % use with trapz, trapz*df
        start = sum(FActi_synced<2);
        imagesc((1:(length(Actiaccel(:,j))/fs)-1)/60, FActi_synced(start:end),
20*log10(abs(Acti_stft_synced(start:end,:))));
        ylabel([axis_labels{j}],'FontSize',16);
        Acti.stft{j} = Acti_stft_synced;

        for k = 1:length(band)-1;
            Actiband = find(FActi_synced >=band(k) & FActi_synced <=band(k+1));    %
looks for all the content in the specific freq. band
            BW = band(k+1)-band(k); % bandwidth of frequency band

            t_stft = abs((Acti_stft_synced));
            dist = trapz(t_stft(Actiband,:));
            Acti_result = [Acti_result; dist*df/BW];  % integral power of Acti
acceleration

        end

    end

    Acti_results.window{window} = Acti_result;
    sgtitle(['Synced ActiGraph with window size ' num2str(twindows(window)) ' seconds'])

    for j = 1:4 % 4 different bands, 1 figure for each band
        figure('Name', ['ActiGraph Window size ' num2str(twindows(window)) ', ' titles{j}
' band'])
        for k = 1:channels  % 3 axis (x y z)
            subplot(channels,1,k);
            plot(Acti_results.window{window}((k-1)*4+j,:))
            ylabel(axis_labels{k})
        end
        sgtitle(['ActiGraph Window size ' num2str(twindows(window)) ', ' titles{j} '
band'])
    end

    figure('Name', ['Synced Apple with window size ' num2str(twindows(window)) '
seconds'])
    for j = 1:channels;
        subplot(channels,1,j)
        [Apple_stft_synced, FApple_synced] = stft(Appleaccel(:,j),fs, 'Window', win,
'FFTLength', Nfft, 'OverlapLength', Noverlap);
        start = sum(FApple_synced<2);
        df = mean(diff(FApple_synced));
        imagesc((1:(length(Appleaccel(:,j))/fs)-1)/60, FApple_synced(start:end),
20*log10(abs(Apple_stft_synced(start:end,:))));
        ylabel([axis_labels{j}],'FontSize',16);

        for k = 1:length(band)-1;
            Appleband = find(FApple_synced >=band(k) & FApple_synced <=band(k+1));
            BW = band(k+1)-band(k);

            t_stft = abs((Apple_stft_synced));
            y = trapz(t_stft(Appleband,:));
            Apple_result = [Apple_result; y*df/BW];
        end
    end
    Apple_results.window{window} = Apple_result;
    sgtitle(['Synced Apple with window size ' num2str(twindows(window)) ' seconds'])

    for j = 1:4,
        figure('Name', ['Apple Window size ' num2str(twindows(window)) ', ' titles{j} '
band'])
        for k = 1:channels,
            subplot(channels,1,k);
            plot(Apple_results.window{window}((k-1)*4+j,:))
            ylabel(axis_labels{k})
        end
        sgtitle(['Apple Window size ' num2str(twindows(window)) ', ' titles{j} ' band'])
    end

end
```

Performing elbow method to find optimal clusters

```
clusters = 10;
Acti_distortion_array = [];
for clust = 1:clusters
    [Acti_idx c] = kmeans(Acti_result',clust,'Replicate',5);
    d = c';
    for j = 1:length(unique(Acti_idx))
        x_array = [];
        idx_array = Acti_result(:,find(Acti_idx==j));
        for k = 1:size(idx_array,2)
            x = idx_array(:,k)-d(:,j);
            x_array = [x_array x];
        end
        %         size(x_array)
        t = x_array.^2;
        y = sum(t,2);
        %         size(t)
        distortion = sqrt(y);
        distortion = mean(distortion);
    end
    Acti_distortion_array = [Acti_distortion_array distortion];
end
figure('Name', 'Elbow Method for ActiGraph')
plot(1:clusters,Acti_distortion_array,'-xr')
xlabel('Number of clusters')
ylabel('Distortion')
title('Elbow method for ActiGraph')

apple_distortion_array = [];
for clust = 1:clusters
    [apple_idx c] = kmeans(Apple_result',clust,'Replicate',5);
    d = c';
    for j = 1:length(unique(apple_idx))
        x_array = [];
        idx_array = Apple_result(:,find(apple_idx==j));
        for k = 1:size(idx_array,2)
            x = idx_array(:,k)-d(:,j);
            x_array = [x_array x];
        end
        %          size(x_array)
        t = x_array.^2;
        y = sum(t,2);
        %          size(t)
        distortion = sqrt(y);
        distortion = mean(distortion);
    end
    apple_distortion_array = [apple_distortion_array distortion];
end

figure('Name', 'Elbow Method for Apple Watch')
plot(1:clusters,apple_distortion_array,'-xr')
xlabel('Number of clusters')
ylabel('Distortion')
title('Elbow method for Apple Watch')
```

Finding optimal number of clusters

```
y = Acti_distortion_array - min(Acti_distortion_array) + 0.2; % IPG
x = [1:10; ones([1, 10])]';
[coef,BINT,R,RINT,STATS] = regress(log(y'), x);
A = exp(coef(2));
clus = [1:10];
B = mean(Acti_distortion_array(7:end));
yhat = A*exp(coef(1)*clus) + B;
figure('Name', 'Optimum number of cluster IPG');
plot(clus, Acti_distortion_array, 'r')
hold on;
plot(clus, yhat, 'b');
xlabel('number of cluster'); ylabel('distortion array'); title('Optimum number of cluster
ActiGraph');
legend('distortion_array', 'yhat')
cluster_Acti = knee_pt(yhat, clus)

% line = (log2(A)+ coef(1)*clus) + log2(B);
% plot(clus, line)
% p = polyfit(yhat(1:5),clus(1:5),1)

% Apple Watch
y = apple_distortion_array - min(apple_distortion_array) + 0.2;
x = [1:10; ones([1, 10])]';
[coef,BINT,R,RINT,STATS] = regress(log(y'), x);
A = exp(coef(2));
clus = [1:10];
B = mean(apple_distortion_array(7:end));
```

```
yhat = A*exp(coef(1)*clus) + B;
figure('Name', 'Optimum number of cluster Apple Watch');
plot(clus, apple_distortion_array, 'r')
hold on;
plot(clus, yhat, 'b')
xlabel('number of cluster'); ylabel('distortion array'); title('Optimum number of cluster
Apple Watch');
legend('distortion array', 'yhat')
cluster_apple = knee_pt(yhat, clus)
```

Clustering using optimal k clusters

```
clusters = cluster_Acti;
[Acti_idx c] = kmeans(Acti_result',clusters,'Replicate',5);
for clust = 1:clusters
    idx_array = Acti_result(:,find(Acti_idx==clust));
    figure('Name',['STFT of ActiGraph cluster ' num2str(clust)])
    imagesc(idx_array)
    title(['Acti cluster ' num2str(clust)])
end
clusters = cluster_apple;
[apple_idx c] = kmeans(Apple_result',clusters,'Replicate',5);
for clust = 1:clusters
    idx_array = Apple_result(:,find(apple_idx==clust));
    figure('Name',['STFT of Apple Watch cluster ' num2str(clust)])
    imagesc(idx_array)
    title(['Apple Watch cluster ' num2str(clust)])
end

% for i = 1:5
% idx_array = abs(Acti_result(:,find(idx==i)));
% figure;imagesc(idx_array(1050:1200,:));size(idx_array)
% end
```

Look at the clusters in pc space

```
pt_markers = ['r.'; 'g.'; 'b.'; 'k.'];

[Acti_coeff Acti_score] = pca(Acti_result');

figure('Name', 'ActiGraph results in PC space')
plot(Acti_score(:,1),Acti_score(:,2),'.c')
title('Acti results in PC space')
xlabel('PC 1')
ylabel('PC 2')

figure('Name', 'Stem plot of ActiGraph coefficients 1 and 2')
subplot(2,1,1)
stem(Acti_coeff(:,1))
title('PC 1')
```

```
subplot(2,1,2)
stem(Acti_coeff(:,2))
title('PC 2')
sgtitle('ActiGraph coefficients 1 and 2 stem plot')

figure('Name', 'ActiGraph results in PC space color coded')
for iclust = 1:clusters,
    hold on;
    plot(Acti_score(find(Acti_idx==iclust), 1), Acti_score(find(Acti_idx==iclust), 2),
pt_markers(iclust,:), 'LineWidth', 1);
end
title('ActiGraph results in PC space')
xlabel('PC 1')
ylabel('PC 2')
legend('Clust 1', 'Clust 2', 'Clust 3','Clust 4')

[Apple_coeff Apple_score] = pca(Apple_result');

figure('Name', 'Apple Watch results in PC space')
plot(Apple_score(:,1),Apple_score(:,2),'.c')
title('Apple results in PC space')
xlabel('PC 1')
ylabel('PC 2')

figure('Name', 'Stem plot of Apple Watch coefficients 1 and 2')
subplot(2,1,1)
stem(Apple_coeff(:,1))
title('PC 1')
subplot(2,1,2)
stem(Apple_coeff(:,2))
title('PC 2')
sgtitle('Apple coefficients 1 and 2 stem plot')

figure('Name', 'Apple Watch results in PC space color coded')
for iclust = 1:clusters,
    hold on;
    plot(Apple_score(find(apple_idx==iclust), 1), Apple_score(find(apple_idx==iclust),
2), pt_markers(iclust,:), 'LineWidth', 1);
end
title('Apple results in PC space')
xlabel('PC 1')
ylabel('PC 2')
legend('Clust 1', 'Clust 2', 'Clust 3','Clust 4')
```

Plot synced acceleration color coded by clusters

synced acceleration data are named Appleaccel and Actiaccel use Acti_idx and

apple_idx

```
idx_color = ['r', 'g', 'b','k'];
Delta = Lwin - Noverlap; % number of samples between each time block of the stft

figure('Name', 'Synced ActiGraph acceleration color coded')
nBlocks = floor(length(Actiaccel)/Delta);
tSyncB = reshape(tConvActi(1:nBlocks*Delta), Delta, nBlocks); % organized in time blocks
used for stft and getting apple_idx
for iclust = 1:clusters
    iThisClust  = find(Acti_idx == iclust);
    for j = 1:channels
        subplot(3,1,j)
        ActiaccelB = reshape(Actiaccel(1:nBlocks*Delta, j), Delta, nBlocks); % organized
in time blocks corresponding to stft and used for getting apple_idx
        plot(tSyncB(:, iThisClust)/60,ActiaccelB(:, iThisClust),idx_color(iclust));  hold
on
        ylabel([axis_labels{j}],'FontSize',16);
    end
end
sgtitle('synced ActiGraph accel colored by idx classification')

figure('Name','Synced Apple acceleration color coded')
nBlocks = floor(length(Appleaccel)/Delta);
tSyncB = reshape(tConvApp(1:nBlocks*Delta), Delta, nBlocks); % organized in time blocks
used for stft and getting apple_idx
for iclust = 1:clusters
    iThisClust  = find(apple_idx == iclust);
    for j = 1:channels
        subplot(3,1,j)
        AppleaccelB = reshape(Appleaccel(1:nBlocks*Delta, j), Delta, nBlocks); %
organized in time blocks corresponding to stft and used for getting apple_idx
        plot(tSyncB(:, iThisClust)/60,AppleaccelB(:, iThisClust),idx_color(iclust));
hold on
        ylabel([axis_labels{j}],'FontSize',16);
    end
end
sgtitle('synced Apple accel colored by idx classification')
```

Plotting tremor band over classified acceleration signals

"tremor" band 4-7 Hz band are the 2nd,6th and 10th rows in the matrices "Apple_result" and "Acti_result"

```
Acti_tremor_band = Acti_result([2,6,10],:);    % Acti tremor band
Apple_tremor_band = Apple_result([2,6,10], :);   % Apple Watch tremor band

tTremor = tConvActi((Lwin-Noverlap):(Lwin-Noverlap):end);
tTremor = tTremor(1:end-1);
Delta = Lwin - Noverlap; % number of samples between each time block of the stft
```

```
figure('Name','Classified ActiGraph acceleration with tremor band')
nBlocks = floor(length(Actiaccel)/Delta);
tSyncB = reshape(tConvActi(1:nBlocks*Delta), Delta, nBlocks); % organized in time blocks
used for stft and getting Acti_idx
for iclust = 1:clusters
    iThisClust  = find(Acti_idx == iclust);
    for j = 1:channels
        subplot(3,1,j)
        ActiaccelB = reshape(Actiaccel(1:nBlocks*Delta, j), Delta, nBlocks); % organized
in time blocks corresponding to stft and used for getting apple_idx
        plot(tSyncB(:, iThisClust)/60,ActiaccelB(:, iThisClust),idx_color(iclust));  hold
on
        plot(tTremor/60,Acti_tremor_band(j,:),'Color',[0.9290 0.6940 0.1250]);
        ylabel([axis_labels{j}],'FontSize',16);
    end
end
sgtitle('Classified ActiGraph acceleration w/ tremor band')
trem_clust_Acti=zeros(2,1)
for k=1:2
trem_clust_Acti(k)=input('which cluster is "tremor" in the ActiGraph classification?
r(1), g(2), b(3), k(4):\n');
end
rest_clust_Acti = input('which cluster is "rest" in the ActiGraph classification? r(1),
g(2), b(3), k(4):\n ');
activity_clust_Acti = input('which cluster is "voluntary activity" in the ActiGraph
classification? r(1), g(2), b(3), k(4):\n ');


figure('Name','Classified Apple acceleration with tremor band')
tTremorA = tConvApp((Lwin-Noverlap):(Lwin-Noverlap):end);
tTremorA = tTremorA(1:end-1);

nBlocks = floor(length(Appleaccel)/Delta);
tSyncB = reshape(tConvApp(1:nBlocks*Delta), Delta, nBlocks); % organized in time blocks
used for stft and getting apple_idx
for iclust = 1:clusters
    iThisClust  = find(apple_idx == iclust);
    for j = 1:channels
        subplot(3,1,j)
        AppleaccelB = reshape(Appleaccel(1:nBlocks*Delta, j), Delta, nBlocks); %
organized in time blocks corresponding to stft and used for getting apple_idx
        plot(tSyncB(:, iThisClust)/60,AppleaccelB(:, iThisClust),idx_color(iclust));
hold on
        plot(tTremorA/60,Apple_tremor_band(j,:),'Color',[0.9290 0.6940 0.1250]);
        ylabel([axis_labels{j}],'FontSize',16);
    end
end
sgtitle('Classified Apple acceleration w/ tremor band')
trem_clust_apple=zeros(2,1)
for k=1:2
trem_clust_apple(k)=input('which cluster is "tremor" in the Apple classification? r(1),
g(2), b(3), k(4):\n');
end
%trem_clust_apple = input('which cluster is "tremor" in the Apple classification? r(1),
g(2), b(3), k(4):\n ');
%low_trem_clust_apple = input('which cluster is "low tremor" in the Apple classification?
r(1), g(2), b(3), k(4):\n ');
rest_clust_apple = input('which cluster is "rest" in the Apple classification? r(1),
g(2), b(3), k(4):\n ');
activity_clust_apple = input('which cluster is "voluntary activity" in the Apple
classification? r(1), g(2), b(3), k(4):\n ');
```

Actigraph correct detection

Creating ground truth signal based on activities tremor and comparing it with the classification result

```
t_acti = tConvActi/60;

gt = zeros(1, length(t_acti));

i_gt = round([0:3:48]*60*fs); % index of the time points of ground truth signal

gt(i_gt(1)+1:i_gt(2)) = 1; % let's make the code for tremor = 1
gt(i_gt(2)+1:i_gt(4)) = 2; % let's make the code for laying down = 2
gt(i_gt(4)+1:i_gt(5)) = 3; % let's make the code for voluntary activity(bouncing) = 3
gt(i_gt(5)+1:i_gt(7)) = 2; % let's make the code for standing = 2
gt(i_gt(7)+1:i_gt(8)) = 3; % let's make the code for voluntary activity(writing) = 3
gt(i_gt(8)+1:i_gt(10)) = 2; % let's make the code for sitting = 2
gt(i_gt(10)+1:i_gt(11)) = 3; % let's make the code for voluntary activity(cooking) = 3
gt(i_gt(11)+1:i_gt(13)) = 2; % let's make the code for standing = 2
gt(i_gt(13)+1:i_gt(14)) = 3; % let's make the code for voluntary activity(walking) = 3
gt(i_gt(14)+1:i_gt(16)) = 2; % let's make the code for sitting = 2
gt(i_gt(16)+1:i_gt(17)) = 3; % let's make the code for voluntary activity(typing) = 3


figure('Name','Ground truth ActiGraph');
plot(t_acti,gt); xlabel('time(min)'); ylabel('ground truth');
sgtitle('Ground truth ActiGraph')

% finding TP,FP,FN
% FOR TREMOR
nBlocks = floor(length(Actiaccel)/Delta);
tSyncB = reshape(tConvActi(1:nBlocks*Delta), Delta, nBlocks); % organized in time blocks
used for stft and getting Acti_idx
gt = reshape(gt(1:nBlocks*Delta), Delta, nBlocks);
gt = gt';

TPt =0; % true positive for tremor
FPt=0; % False positive for tremor
FNt=0; % False negative for tremor
TNt=0; % true negative for tremor

Acti_tremor_clust = zeros(length(Acti_idx),1);
```

```
for j = 1:length(Acti_idx)
    if Acti_idx(j) == trem_clust_Acti(1,1)
        Acti_tremor_clust(j) = 1;
    end
end
for j = 1:length(Acti_idx)
    if Acti_idx(j) == trem_clust_Acti(2,1)
        Acti_tremor_clust(j) = 1;
    end
end
for j = 1:length(Acti_idx)
     if (Acti_tremor_clust(j)==1) && (gt(j)==1)
         TPt = TPt+1;
     elseif (Acti_tremor_clust(j) ~=1) && (gt(j)==1)
         FNt = FNt+1;
     elseif (Acti_tremor_clust(j)==1) && (gt(j)~=1)
         FPt = FPt+1;
     elseif (Acti_tremor_clust(j)~=1) && (gt(j)~=1)
         TNt = TNt+1;
     end
end


% FOR REST

TPr =0; % true positive for rest
FPr=0; % False positive for rest
FNr=0; % False negative for rest
TNr=0; % True negative for rest

Acti_rest_clust = zeros(length(Acti_idx),1);
for j = 1:length(Acti_idx)
    if Acti_idx(j) == rest_clust_Acti
        Acti_rest_clust(j) = 2;
    end
end
for j = 1:length(Acti_idx)
     if (Acti_rest_clust(j)==2) && (gt(j)==2)
         TPr = TPr+1;
     elseif (Acti_rest_clust(j) ~=2) && (gt(j)==2)
         FNr = FNr+1;
     elseif (Acti_rest_clust(j)==2) && (gt(j)~=2)
         FPr = FPr+1;
     elseif (Acti_rest_clust(j)~=2) && (gt(j)~=2)
         TNr = TNr+1;
     end
end

% FOR VOLUNTARY ACTIVITY

TPv =0; % true positive for voluntary activity
FPv=0; % False positive for voluntary activity
FNv=0; % False negative for voluntary activity
TNv =0; % True negative for voluntary activity
```

```
Acti_activity_clust = zeros(length(Acti_idx),1);
for j = 1:length(Acti_idx)
    if Acti_idx(j) == activity_clust_Acti
        Acti_activity_clust(j) = 3;
    end
end
for j = 1:length(Acti_idx)
     if (Acti_activity_clust(j)==3) && (gt(j)==3)
         TPv = TPv+1;
     elseif (Acti_activity_clust(j) ~=3) && (gt(j)==3)
         FNv = FNv+1;
     elseif (Acti_activity_clust(j)==3) && (gt(j)~=3)
         FPv = FPv+1;
     elseif (Acti_activity_clust(j)~=3) && (gt(j)~=3)
         TNv = TNv+1;
     end
end
```

Apple Watch correct detection

Creating ground truth signal based on activities tremor and comparing it with the classification result

```
t_app = tConvApp/60;

% time according to activities

t1a= 0:1/fs:tConvApp(165*fs); %tremor
t2a=tConvApp(165*fs+1):1/fs:tConvApp(426*fs);  % laying down
t3a = tConvApp(426*fs+1):1/fs:tConvApp(523.8*fs); % voluntary activity
t4a= tConvApp(523.8*fs+1):1/fs:tConvApp(744.6*fs);  % standing
t5a = tConvApp(744.6*fs+1):1/fs:tConvApp(849.6*fs); % voluntary activity
t6a=tConvApp(849.6*fs+1):1/fs:tConvApp(1068*fs); %sitting
t7a=tConvApp(1068*fs+1):1/fs:tConvApp(1194*fs); % voluntary activity
t8a=tConvApp(1194*fs+1):1/fs:tConvApp(1413*fs); % standing
t9a=tConvApp(1413*fs+1):1/fs:tConvApp(1549.2*fs); % voluntary activity
t10a=tConvApp(1549.2*fs+1):1/fs:tConvApp(1683.6*fs); % sitting
t11a= tConvApp(1683.6*fs+1):1/fs:tConvApp(1797.4*fs); % voluntary activity

gta =
cat(2,(t1a>=0),2*(t2a>=0),3*(t3a>=0),2*(t4a>=0),3*(t5a>=0),2*(t6a>=0),3*(t7a>=0),2*(t8a>=
0),3*(t9a>=0),2*(t10a>=0),3*(t11a>=0));

figure('Name','Ground truth Apple watch');
plot(t_app,gta); xlabel('time(min)'); ylabel('Ground truth');
sgtitle('Apple watch Ground truth');
%
% finding TP,FP,FN
% FOR TREMOR
```

```
nBlocks = floor(length(Appleaccel)/Delta);
tSyncBa = reshape(tConvApp(1:nBlocks*Delta), Delta, nBlocks); % organized in time blocks
used for stft and getting Acti_idx
gta = reshape(gta(1:nBlocks*Delta), Delta, nBlocks);
gta = gta';

TPta =0; % true positive for tremor
FPta=0; % False positive for tremor
FNta=0; % False negative for tremor
TNta=0; % true negative for tremor

apple_tremor_clust = zeros(length(apple_idx),1);
for j = 1:length(apple_idx)
    if apple_idx(j) == trem_clust_apple(1,1)
        apple_tremor_clust(j) = 1;
    end
end
for j = 1:length(apple_idx)
    if apple_idx(j) == trem_clust_apple(2,1)
        apple_tremor_clust(j) = 1;
    end
end
for j = 1:length(apple_idx)
     if (apple_tremor_clust(j)==1) && (gta(j)==1)
         TPta = TPta+1;
     elseif (apple_tremor_clust(j) ~=1) && (gta(j)==1)
         FNta = FNta+1;
     elseif (apple_tremor_clust(j)==1) && (gta(j)~=1)
         FPta = FPta+1;
     elseif (apple_tremor_clust(j)~=1) && (gta(j)~=1)
         TNta = TNta+1;
     end
end

% FOR REST

TPra =0; % true positive for rest
FPra=0; % False positive for rest
FNra=0; % False negative for rest
TNra=0; % true negative for rest

apple_rest_clust = zeros(length(apple_idx),1);
for j = 1:length(apple_idx)
    if apple_idx(j) == rest_clust_apple
        apple_rest_clust(j) = 2;
    end
end
for j = 1:length(apple_idx)
     if (apple_rest_clust(j)==2) && (gta(j)==2)
         TPra = TPra+1;
     elseif (apple_rest_clust(j) ~=2) && (gta(j)==2)
         FNra = FNra+1;
     elseif (apple_rest_clust(j)==2) && (gta(j)~=2)
         FPra = FPra+1;
     elseif (apple_rest_clust(j)~=2) && (gta(j)~=2)
         TNra = TNra+1;
     end
end

% FOR VOLUNTARY ACTIVITY

TPva =0; % true positive for activity
FPva=0; % False positive for activity
FNva=0; % False negative for activity
TNva=0; % true negative for activity

apple_activity_clust = zeros(length(apple_idx),1);
for j = 1:length(apple_idx)
    if apple_idx(j) == activity_clust_apple
        apple_activity_clust(j) = 3;
    end
end
for j = 1:length(apple_idx)
     if (apple_activity_clust(j)==3) && (gta(j)==3)
         TPva = TPva+1;
     elseif (apple_activity_clust(j) ~=3) && (gta(j)==3)
         FNva = FNva+1;
     elseif (apple_activity_clust(j)==3) && (gta(j)~=3)
         FPva = FPva+1;
     elseif (apple_activity_clust(j)~=3) && (gta(j)~=3)
         TNva = TNva+1;
     end
end

% Actigrap true positive and false detection rate
total_TP_acti = TPt+TPr+TPv;
total_FN_acti = FNt+FNr+FNv;
total_FP_acti = FPt+FPr+FPv;
total_TN_acti = TNt+TNr+TNv;

correct_detection_acti = (total_TP_acti/(total_TP_acti+total_FN_acti))*100
false_detection_rate_acti = (total_FP_acti/(total_FP_acti+total_TN_acti))*100

% Apple watch true positive and false detection rate
total_TP_app = TPta+TPra+TPva;
total_FN_app = FNta+FNra+FNva;
total_FP_app = FPta+FPra+FPva;
total_TN_app = TNta+TNra+TNva;

correct_detection_apple = (total_TP_app/(total_TP_app+total_FN_app))*100
false_detection_rate_apple = (total_FP_app/(total_FP_app+total_TN_app))*100
```

APPENDIX C

MATLAB Helper Functions

Creates envelopes of the acceleration signals

```
function ma = env(s, twin, fs)
% display(twin)
    s = s - mean(s);
    Lwin = round(fs*twin);
    w = ones(1, Lwin);
    ma = conv(abs(s), w, 'same')/Lwin;
end
```

Sets the time vector on the same scale for both accelerometers

```
function tSync = timeScaling(t, fs)
% t = original time vector of asynchronously acquired signal
% fs = desired sampling frequency in samples per second
%
% tSync = output time vector, now with constant sampling interval

    N = round(seconds(t(end) - t(1))*fs);
    tSync = linspace(t(1), t(end), N);
end
```

Synchronizes the data of the accelerometers by using linear interpolation

```
Setfunction [a, L] = syncData(accel, tConv, tSync, Apple)
% a = raw acceleration from device (Apple watch or IPG)
% tConv = converted time from device (Apple watch or IPG)
% tSync = converted time from IPG, and resampled at fixed sampling rate

    if(tSync(end) < tConv(end))
        L = sum(tConv <= tSync(end));
    else
        L = length(tConv);
    end
    a = zeros(length(tSync), 3);
    [t, iunique] = unique(tConv(1:L));
    a(:,1) = interp1(t, accel(iunique,1), tSync);
    a(:,2) = interp1(t, accel(iunique,2), tSync);
    a(:,3) = interp1(t, accel(iunique,3), tSync);
end
```

Finds the knee point (optimum number of clusters) of an exponential curve by Dmrity Kaplan from MATLAB file exchange

```
function [res_x, idx_of_result] = knee_pt(y,x,just_return)
%function [res_x, idx_of_result] = knee_pt(y,x,just_return)
%Returns the x-location of a (single) knee of curve y=f(x)
%  (this is useful for e.g. figuring out where the eigenvalues peter out)
%
%Also returns the index of the x-coordinate at the knee
%
%Parameters:
% y (required) vector (>=3 elements)
% x (optional) vector of the same size as y
% just_return (optional) boolean
%
%If just_return is True, the function will not error out and simply return a Nan on
%detected error conditions
%
%Important:  The x and y  don't need to be sorted, they just have to
%correspond: knee_pt([1,2,3],[3,4,5]) = knee_pt([3,1,2],[5,3,4])
%
%Important: Because of the way the function operates y must be at least 3
%elements long and the function will never return either the first or the
%last point as the answer.
%
%Defaults:
%If x is not specified or is empty, it's assumed to be 1:length(y) -- in
%this case both returned values are the same.
%If just_return is not specified or is empty, it's assumed to be false (ie the
%function will error out)
%
%
%The function operates by walking along the curve one bisection point at a time and
%fitting two lines, one to all the points to left of the bisection point and one
%to all the points to the right of of the bisection point.
%The knee is judged to be at a bisection point which minimizes the
%sum of errors for the two fits.
%
%the errors being used are sum(abs(del_y)) or RMS depending on the
%(very obvious) internal switch.  Experiment with it if the point returned
%is not to your liking -- it gets pretty subjective...
%
%
%Example: drawing the curve for the submission
% x=.1:.1:3; y = exp(-x)./sqrt(x); [i,ix]=knee_pt(y,x);
% figure;plot(x,y);
% rectangle('curvature',[1,1],'position',[x(ix)-.1,y(ix)-.1,.2,.2])
% axis('square');
%

%Food for thought: In the best of possible worlds, per-point errors should
%be corrected with the confidence interval (i.e. a best-line fit to 2
%points has a zero per-point fit error which is kind-a wrong).
%Practially, I found that it doesn't make much difference.
%
%dk /2012



%{

% test vectors:

[i,ix]=knee_pt([30:-3:12,10:-2:0])  %should be 7 and 7
knee_pt([30:-3:12,10:-2:0]')  %should be 7
knee_pt(rand(3,3))  %should error out
knee_pt(rand(3,3),[],false)  %should error out
knee_pt(rand(3,3),[],true)  %should return Nan
knee_pt([30:-3:12,10:-2:0],[1:13])  %should be 7
knee_pt([30:-3:12,10:-2:0],[1:13]*20)  %should be 140
knee_pt([30:-3:12,10:-2:0]+rand(1,13)/10,[1:13]*20)  %should be 140
knee_pt([30:-3:12,10:-2:0]+rand(1,13)/10,[1:13]*20+rand(1,13)) %should be close to 140
x = 0:.01:pi/2; y = sin(x); [i,ix]=knee_pt(y,x)  %should be around .9 andaround 90
[~,reorder]=sort(rand(size(x)));xr = x(reorder); yr=y(reorder);[i,ix]=knee_pt(yr,xr)  %i
should be the same as above and xr(ix) should be .91
knee_pt([10:-1:1])  %degenerate condition -- returns location of the first "knee" error
minimum: 2

%}


%set internal operation flags
use_absolute_dev_p = true;  %ow quadratic

%deal with issuing or not not issuing errors
issue_errors_p = true;
if (nargin > 2 && ~isempty(just_return) && just_return)
    issue_errors_p = false;
end

%default answers
res_x = nan;
idx_of_result = nan;

%check...
if (isempty(y))
    if (issue_errors_p)
        error('knee_pt: y can not be an empty vector');
    end
    return;
end

%another check
if (sum(size(y)==1)~=1)
    if (issue_errors_p)
        error('knee_pt: y must be a vector');
    end

    return;
end

%make a vector
y = y(:);

%make or read x
if (nargin < 2 || isempty(x))
    x = (1:length(y))';
else
    x = x(:);
end

%more checking
if (ndims(x)~= ndims(y) || ~all(size(x) == size(y)))
    if (issue_errors_p)
        error('knee_pt: y and x must have the same dimensions');
    end

    return;
end

%and more checking
if (length(y) < 3)
    if (issue_errors_p)
        error('knee_pt: y must be at least 3 elements long');
    end
    return;
end

%make sure the x and y are sorted in increasing X-order
if (nargin > 1 && any(diff(x)<0))
    [~,idx]=sort(x);
    y = y(idx);
    x = x(idx);
else
    idx = 1:length(x);
end

%the code below "unwraps" the repeated regress(y,x) calls.  It's
%significantly faster than the former for longer y's
%
%figure out the m and b (in the y=mx+b sense) for the "left-of-knee"
sigma_xy = cumsum(x.*y);
sigma_x  = cumsum(x);
sigma_y  = cumsum(y);
sigma_xx = cumsum(x.*x);
n        = (1:length(y))';
det = n.*sigma_xx-sigma_x.*sigma_x;
mfwd = (n.*sigma_xy-sigma_x.*sigma_y)./det;
bfwd = -(sigma_x.*sigma_xy-sigma_xx.*sigma_y) ./det;
```

```
%figure out the m and b (in the y=mx+b sense) for the "right-of-knee"
sigma_xy = cumsum(x(end:-1:1).*y(end:-1:1));
sigma_x  = cumsum(x(end:-1:1));
sigma_y  = cumsum(y(end:-1:1));
sigma_xx = cumsum(x(end:-1:1).*x(end:-1:1));
n        = (1:length(y))';
det = n.*sigma_xx-sigma_x.*sigma_x;
mbck = flipud((n.*sigma_xy-sigma_x.*sigma_y)./det);
bbck = flipud(-(sigma_x.*sigma_xy-sigma_xx.*sigma_y) ./det);

%figure out the sum of per-point errors for left- and right- of-knee fits
error_curve = nan(size(y));
for breakpt = 2:length(y-1)
    delsfwd = (mfwd(breakpt).*x(1:breakpt)+bfwd(breakpt))-y(1:breakpt);
    delsbck = (mbck(breakpt).*x(breakpt:end)+bbck(breakpt))-y(breakpt:end);
    %disp([sum(abs(delsfwd))/length(delsfwd), sum(abs(delsbck))/length(delsbck)])
    if (use_absolute_dev_p)
        % error_curve(breakpt) = sum(abs(delsfwd))/sqrt(length(delsfwd)) +
sum(abs(delsbck))/sqrt(length(delsbck));
        error_curve(breakpt) = sum(abs(delsfwd))+ sum(abs(delsbck));
    else
        error_curve(breakpt) = sqrt(sum(delsfwd.*delsfwd)) + sqrt(sum(delsbck.*delsbck));
    end
end

%find location of the min of the error curve
[~,loc] = min(error_curve);
res_x = x(loc);
idx_of_result = idx(loc);
end
```