IS4250 PROJECT REPORT

AY2015/2015 Semester 2

Online Detection of Freezing of Gait with Smartphones and Machine Learning Techniques

Prepared by Team 19

Bui Do Phuong Tung	A0099838A
Lin Yu Ting	A0112896X
Gao Yiyang	A0100423E

Table of Contents

1. Introduction

Based on research, there are approximately 60,000 Americans are diagnosed with Parkinson's Disease every year, and an estimation of 7 to 10 million people are living with this disease worldwide. (Statistics on Parkinson's, 2016) Freezing of gait (FoG) is one of major motor symptoms suffered by advanced PD patients. Statistics shows FoG occurs in 53% of advanced PD patients. (Roneil Malkani, 2013) In a survey of 6620 patients suffering from PD, 47% of the subjects experienced regular freezing while 27% experienced daily episodes of FoG. (Macht, 2007) As most of FoG patients are resistant to Pharmacological treatment, the invention of most effective non-pharmacological treatment becomes more important.

This paper proposed a non-pharmacological treatment method for PD patient with FOG symptoms. The non-pharmacological treatment method is an online context-aware FoG detection system that comprised by a mobile device, several accelerometers, and an Android online FoG detection application. The system works to analyze the motion data collected from PD patients and analyze the frequency of body components in order to detect FoG events on time and provide a metronome ticking sound as a sense queue.

As the sensitivity and specificity of the system is affected by factors including sensor placement, Patient's walking style and algorithm parameters, a study involved 10 PD patients is conducted. Within 8 hours of experiments, 237 FoG events were identified by professional physiotherapists in retrospective way. In contrast, the system achieved 73.1% of sensitivity and 81.6% of specificity. By testing on different sensor placements, patient's walking style and algorithms used, the researchers aim to find the best feature sets in order to achieve the best sensitivity and specificity of the detection system.

1.1 Background Information

1.1.1 Parkinson's Disease

Parkinson's disease was first identified by a British apothecary named James Parkinson in 1817. James is the first person who has identified the main characteristics of Parkinson's disease in his medical essay - "An essay on the shaking palsy". Sixty years later, James' work was recognized by a French Neurologist, Jean Martin Charcot. He acknowledged the importance of his studies and named this medical condition discovered by him as Parkinson's Disease.

Figure 1. James Parkinson

PD is a systematic movement disorder, and the symptoms of this disease will be worsen over

time. It usually involves the malfunction and death of neurons. The patient will start to lose his control on movement when PD progresses. This is caused by decrease in dopamine production in brain. The common symptoms experienced by PD patients are tremor, bradykinesia, rigidity and postural instability. (*Parkinson's Disease Information,* 2009*)*

Figure 2. Parkinson's Disease Symptoms

1.1.2 Freezing of Gait (FoG)

Freezing of Gait is one of the secondary motor symptoms of advanced PD. It defined as a gait disturbance, interrupting walking with a sudden and transient nature. (Fahn, 1995) It is one of the most debilitating motor symptoms for PD patients. Patients with FoG will experience loss of mobility and unable to reinitiate his steps. It usually causes falls and a loss of independence. (Heremans E, 2013)

1.1.3 Current Treatment

Currently, there is no cure for this symptom. Patients have to either undergo through pharmacological treatment by taking medicine called Levodopa or choose the non-pharmacological treatment. For the non-pharmacological treatment, which called Rhythmic Auditory Stimulation (RAS), is to use a metronome to produce ticking sound t as cues for gait training of PD patients. Once the patient received this external signal, they will try to continue their movements. However, the effect of RAS treatment wears off overtimes. In addition, it is inconvenient to use in daily lives. (Zack Zhu, 2013)

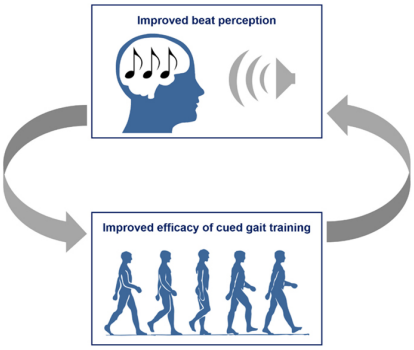


Figure 3. Rhythmic Auditory Stimulation

2. System Design

This paper has proposed a more mobile and user friendly non-pharmacological treatment method for advanced PD patients with FoG symptom. This online detection system consisted by a mobile device with real-time FoG detection app and three external sensors. (Referring to Figure 4. FoG Online Detection System)

Figure 4. FoG Online Detection System

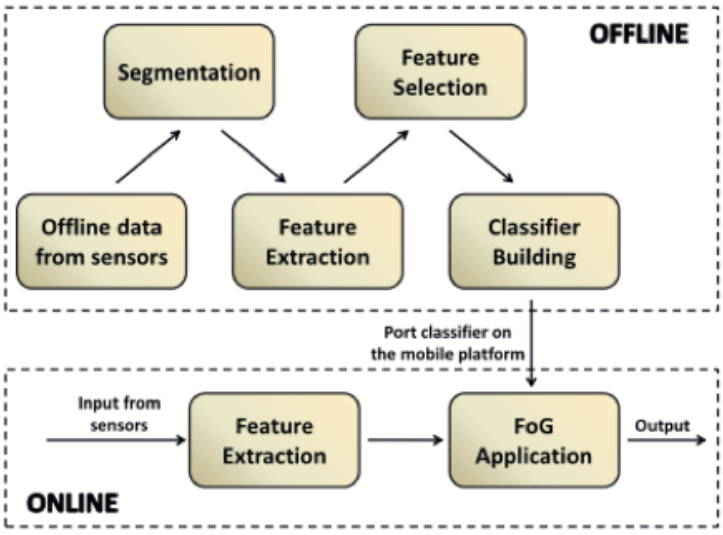


Figure 5. Classification Process

As referring to Figure 5, it shows the classification process of online FoG detection application. Firstly, the FOG classifier is trained offline with previously collected data. Data

will be segmented based on different patients. Feature selection is performed to choose the most discriminative features which used to distinguish FoG from normal gait. These selected features will be used for classifier training with machine learning techniques in Weka data mining suite. The classifier is then serialized and ported to smartphone using the Weka serialization API.

In the online phase of process, the android app using the deserialized classifier created in earlier process to detect FoG in real time. After the data collection from the wearable sensors, the FoG detection application notify the sensor communication module to collect the data, preprocess it to extract the necessary features and pass them to the classifier. When a FoG event is detected, the system will provide the feedbacks to the patients in the form of RAS signal or phone vibration.

3. System Evaluation

3.1 Dataset

The Daphnet Freezing of Gait Dataset in users with Parkinson's disease (Daphnet Freezing of Gait Dataset), created as a collaboration between the Laboratory for Gait and Neurodynamics, Tel Aviv Sourasky Medical Center, Israel and the Wearable Computing Laboratory, ETH Zurich, Switzerland in 2008. It is devised to benchmark automatic methods to recognize gait freezing of gait with wearable acceleration sensors placed on legs and hip.

3.1.1 Recording Scenario

The dataset was recorded in the lab with emphasis on generating many freeze events. Patients are to complete two 10-15 minutes sessions, consisting of three walking tasks:

1) Walking in a straight line, including several 180 degree turns
2) Random walking in a reception hall space, including a series of initiated stops and 360 degrees turns. Patient should stop or turn in different directions
3) Walking simulating activities of daily living (ADL), including entering and leaving rooms, walking to the kitchen, getting drink and returning to the starting room with a cup of water

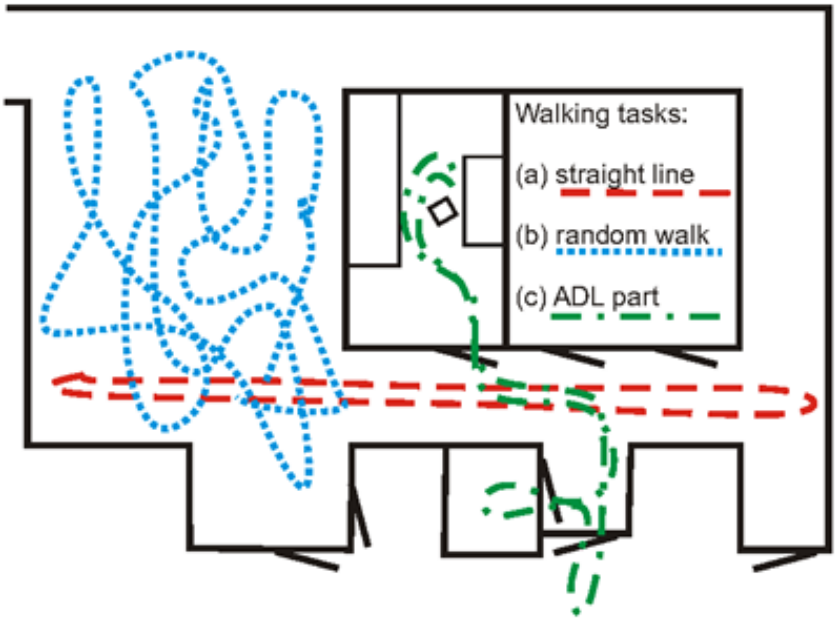


Figure 6. Recording Scenario

3.1.2 Sensors Details

The dataset comprises 3 wearable wireless acceleration sensors recording 3D acceleration at 64 Hz. The sensors are placed at 3 body locations: ankle (shank), upper leg, and hip.

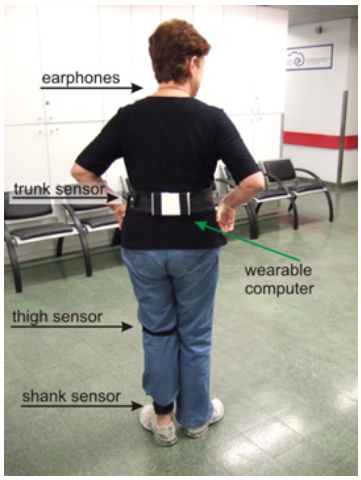


Figure 7. Placement of Wearable Sensors

3.1.3 Data Attributes

Each file in the dataset contains the following information:

1) Time of sample in millisecond
2) Ankle (shank) acceleration - horizontal forward acceleration [mg]
3) Ankle (shank) acceleration - vertical [mg]
4) Ankle (shank) acceleration - horizontal lateral [mg]
5) Upper leg (thigh) acceleration - horizontal forward acceleration [mg]
6) Upper leg (thigh) acceleration - vertical [mg]
7) Upper leg (thigh) acceleration - horizontal lateral [mg]
8) Trunk acceleration - horizontal forward acceleration [mg]
9) Trunk acceleration - vertical [mg]

10) Trunk acceleration - horizontal lateral [mg]
11) Annotations (annotation = 0: not part of the experiment; annotation = 1: experiment, no freezing of gait (can be standing, walking, turning); annotation = 2: freezing of gait)

3.2 Experiments

The experiment evaluates three main goals:
1) Effective machine learning algorithms in terms of detection accuracy and latency
2) Accuracy between different window lengths
3) Accuracy between different sensor placements

3.2.1 Machine Learning Algorithms

Experiment: The following machine learning algorithms are tested:

- Random Trees (RT)
- Random Forests (RF)
- decision trees and pruned decision trees (C4.5)
- Naive Bayes (NB)
- Bayes Nets (BN)
- k-nearest neighbor with one neighbor (KNN-1)
- k-nearest neighbor with two neighbors (KNN-2)
- Multilayer perceptron (MLP)
- Boosting (AdaBoost) with pruned C4.5 trees
- Bagging with pruned C4.5 trees.

For this experiment, both training and testing data are from the same patient. 10-fold cross validation is performed for each of the feature selection and classification methods listed for each patient data from the DAPHNet dataset.

The video annotation done by physiotherapists in the DAPHNet dataset is used as basis of evaluation for the accuracy checking of the algorithm. The detection performance is based on window evaluation: the result output of each window is compared to the annotation in the dataset for result checking.

The window lengths to be used for this experiment are 1 second and 4 second windows. The 4s-window results are presented to compare with the state-of-the-art FoG detection system, which also uses 4s windows. Furthermore, shorter windows such as 1s-window is also tested.

For each algorithm, the following parameters are calculated:

- True Positives (TP): Windows that are correctly labeled as FoG
- False Positives (FP): Windows that are wrongly labeled as FoG
- False Negatives (FN): Windows where the system failed to detect FoG during FoG episodes
- True Negatives (TN): The remaining windows (labeled as no FoG)
- Sensitivity = TP/(TP+FN)
- Specificity = TN/(TN + FP)
- F1-measure: takes into account the precision (precision = TP/(TP+FP)) and recall rate (identical to specificity) for each class

- Area under the curve (AUC) in the ROC space
- Latency: the delay between the moment where an FoG episode starts and the moment that an FoG episode is detected by the application

Results:

	1s window				4s window			
Classifier	**Sens (%)**	**Spec (%)**	**F1 (%)**	**AUC (%)**	**Sens (%)**	**Spec (%)**	**F1 (%)**	**AUC (%)**
Random Forest	**97.76**	**99.75**	**98.35**	**99.82**	**99.54**	**99.96**	**99.75**	**99.98**
C4.5	93.47	99.38	95.94	97.40	98.43	99.84	99.04	99.35
Näive Bayes	48.06	98.66	73.62	93.08	41.87	99.73	71.13	95.31
MLP	77.46	97.29	82.94	95.84	84.99	98.63	91.17	98.41
AdaBoost with C4.5	**98.35**	**99.72**	**98.37**	**99.85**	**99.69**	**99.96**	**99.78**	**99.98**
Bagging with C4.5	97.57	99.59	97.60	99.84	99.41	99.91	99.53	99.97

Figure 8. Average Sen, Spec, F1 and AUC for different classifiers, 1s-windows and 4s-windows

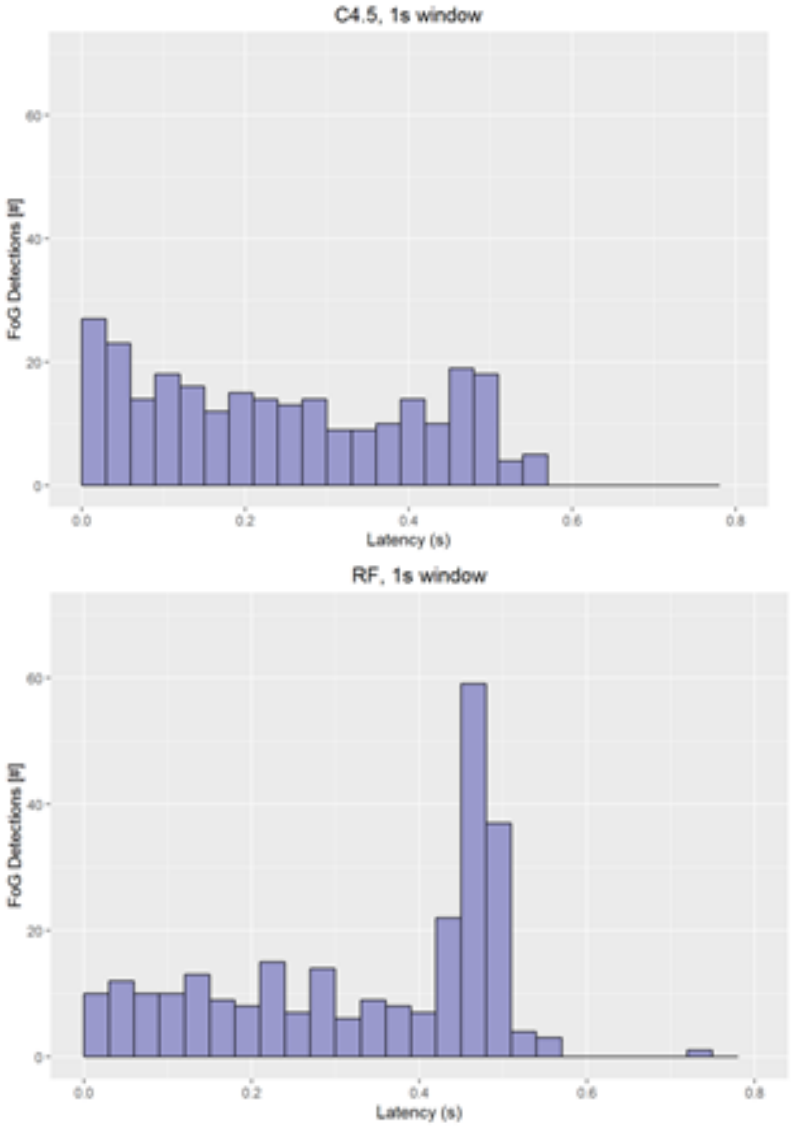
The result obtained is very good for RF, C4.5, NB, MLP, boosting and bagging methods, in both window lengths of 1s and 4s. However, RF and AdaBoost with C4.5 have the best result here both in 1s-window and 4s-window (more than 97% sensitivity, 99.7% specificity for 1s-window; more than 99.5% sensitivity, 99.9% specificity for 4s-window). Furthermore, 1s-window only has a slightly worse result compared to 4s-window.

Window	Classifier	Mean (s)	Std (s)	Max (s)
1s	C4.5	**0.235**	0.175	**0.578**
	RF	0.346	0.169	0.718
4s	C4.5	**1.085**	0.731	**2.016**
	RF	1.653	0.59	2.047

Figure 9: Latency Result

The latency is also determined along with the performance result, Here the 1s-window has much lower latency compared to 4s-window, with around 4 times shorter. As such, there is a trade off between latency and performance in the result, with long detection time but really accurate result in 4s-window while fast detection but not as accurate result in 1s-window.

In addition, between the algorithms, while RF obtains better performances for FoG detection accuracy compared to C4.5 trees, the detection latency is higher. Fig. 10 below shows the comparison between RF and C4.5 in latency. We can see that RF usually detects FoG around half the window length after the FoG episode starts. However, the differences between latency of C4.5 and RF are relatively small and the maximum latency of both algorithms in 1s-window is only 0.718s.


C4.5, 1s window
FoG Detections [#]
0
20
40
60
0.0
0.2
0.4
0.6
0.8
Latency (s)
RF, 1s window
FoG Detections [#]
0
20
40
60
0.0
0.2
0.4
0.6
0.8
Latency (s)

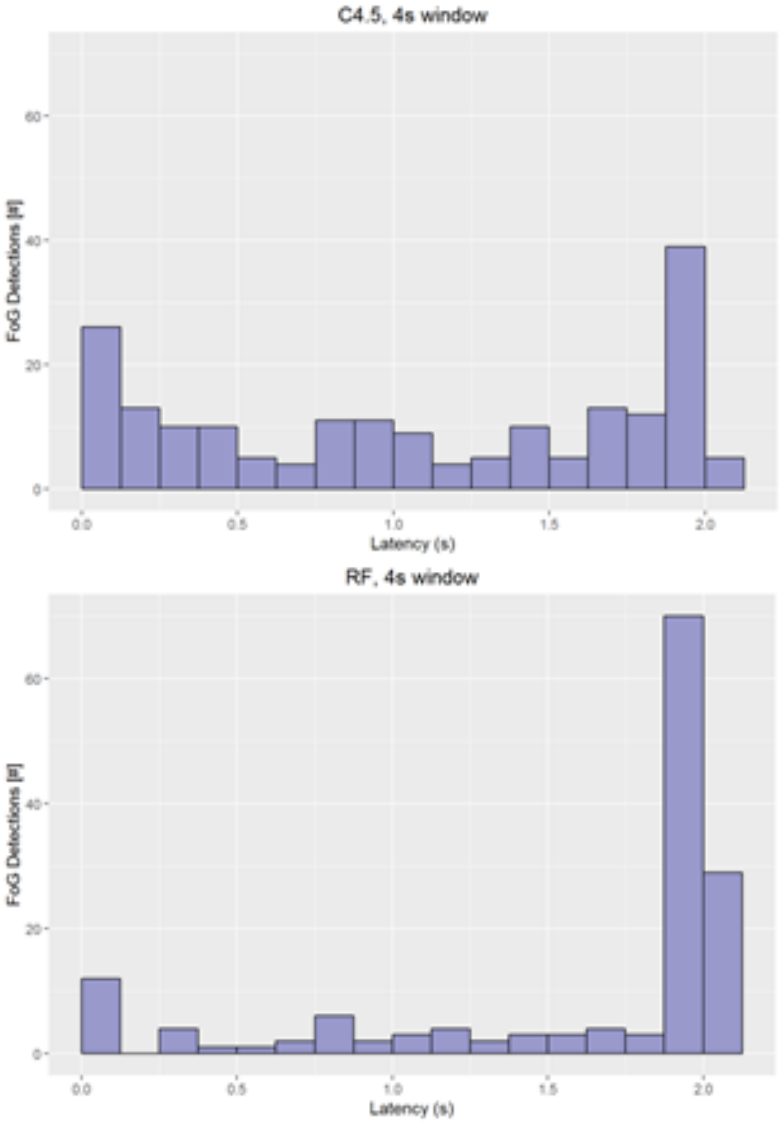


Figure 10. Simulated histogram on correlation of detection latency (Refer to Appendix A-1 for codes)

3.2.2 Window Lengths

For the analysis of window length, below is a graph plotted with the following data:

1) Performance (minimum value between sensitivity and specificity) vs window length

2) Detection latency versus window length.

These data are measured with window sizes from 0.5s to 8s and in steps of 0.5s.

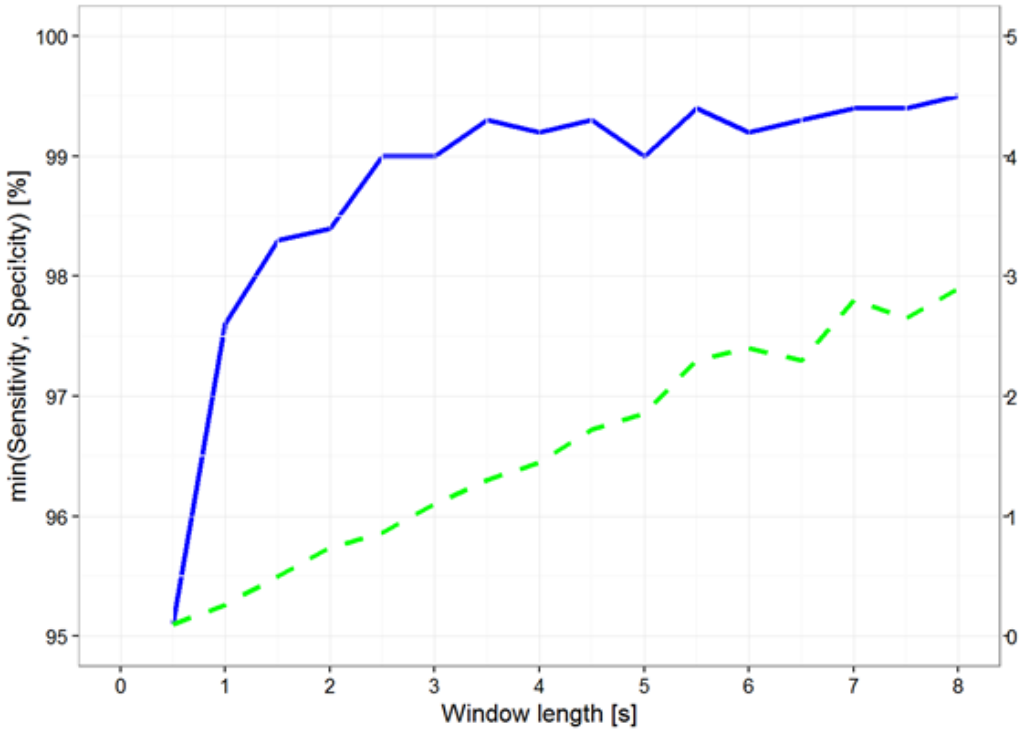


Figure 11. Simulated Plot on detection performance and latency vs. window length (RF classifier) (Refer to Appendix A-2 for codes)

We can see that when the window length increases, the detection performance increase as well as detection latency. However, the performance increases sharply from 0.5 to 3s and then only increases minimally afterwards. As such, window lengths that is larger than 3 will not affect the detection performance much. In the meanwhile, the latency increases constantly through the different window length. Therefore we can say that 1s-window length will give a better trade off between latency and performance compared to 4s-window length.

3.2.3 Sensor Placements

For sensor placement that would give the best result, three body locations was investigated which are ankle, upper leg and hips. The data collected from each location will be used for analysis with RF classifier. The sensitivity as well as specificity of the data in each location and direction (x, y, z) will be calculated for comparison. Below is the result:

	x	*y*	*z*	*x & y & z*
Ankle				
Sens	92.65%	89.18%	91.85%	98.21%
Spec	99.26%	98.84%	99.02%	99.76%
Knee				
Sens	89.36%	87.71%	88.58%	97.94%
Spec	98.86%	98.80%	98.80%	99.73%
Hip				
Sens	88.81%	83.68%	90.77%	98.63%
Spec	98.77%	98.26%	99.02%	99.83%

Figure 12. Sensor placement detection performance

From the result above, we can see that the hip sensors give the best results with 98.63% sensitivity and 99.83% specificity which is very high. However, there is not much difference between the result of the 3 placements. As a result, we can say that having one sensor is enough for FoG detection as each of the 3 places all give good result on its own. Also it can be placed at either of the 3 places depending on which is most convenient and comfortable for the patient.

3.2.4 Experiment Result

In general, the algorithm that could be chosen for smartphone application can be C4.5 and RF classifiers. It is because Pruned C4.5 classifiers are small, easy and quick to implement on smartphone while giving good detection result (in both latency and accuracy). RF however, is more complex to implement on smartphone but it gives a better detection accuracy compared to C4.5.

For window size, 1s-window should be implemented since it has a better trade-off between detection performance and latency compared to 4s-window.

For where the sensor should be put to gives the best result. Each of the 3 locations of ankle, upper leg and hips gives similar result. As such, it is possible to use only one sensor on one of these body part for detection.

3.3 Random Forest Simulation

Our group has also done an analysis on random forest using the same Daphnet dataset in simulation of the research paper.

3.3.1 Dataset

The dataset we used for this is the same dataset that was used in the research paper: Daphnet Freezing of Gait Dataset. However, we only used the data of the 2nd patient for this experiment since it this patient data is referenced in the research paper.

The dataset is in txt format. As such manual conversion to csv format was done for easier processing with R. Furthermore, since there is no attribute labels in the original text files, short label for each attribute was added for the csv file

Attribute name and description	**Label in csv file**
Time of sample in millisecond	time
Ankle (shank) acceleration - horizontal forward acceleration [mg]	AHF
Ankle (shank) acceleration - vertical [mg]	AV
Ankle (shank) acceleration - horizontal lateral [mg]	AHL
Upper leg (thigh) acceleration - horizontal forward acceleration [mg]	ULHF

Upper leg (thigh) acceleration - vertical [mg]	ULV
Upper leg (thigh) acceleration - horizontal lateral [mg]	ULHL
Trunk acceleration - horizontal forward acceleration [mg]	THF
Trunk acceleration - vertical [mg]	TV
Trunk acceleration - horizontal lateral [mg]	THL
Annotations (annotation = 0: not part of the experiment; annotation = 1: experiment, no freeze (can be any of stand, walk, turn); annotation = 2: freeze)	anno

Since each of the patient does the test two times in the dataset, two csv files were generated, patient2test1.csv and patient2test2.csv (from S02R01.txt and S02R02.txt files in the dataset)

We also make sure that this data was exactly what was used in the research paper. For this we compare our plots on period of freezing of gait of the 2nd patient against the plot in the paper.

Plots from the paper:

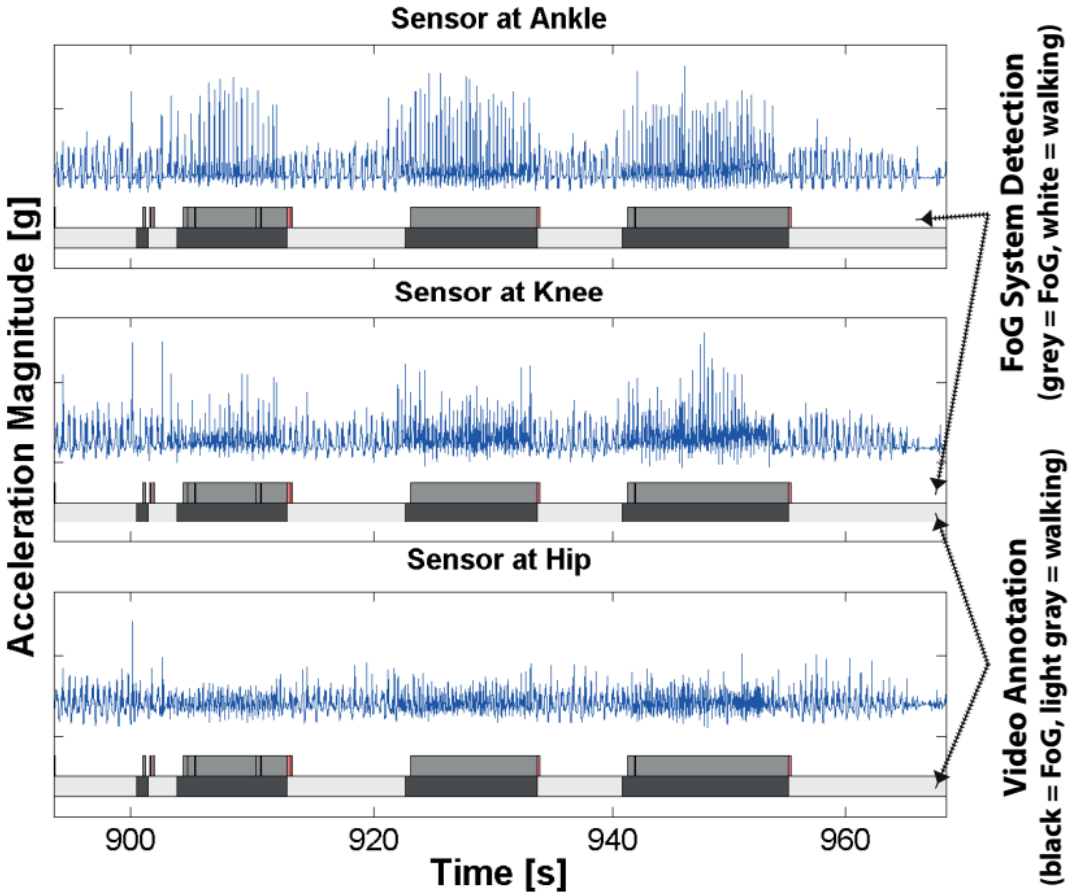
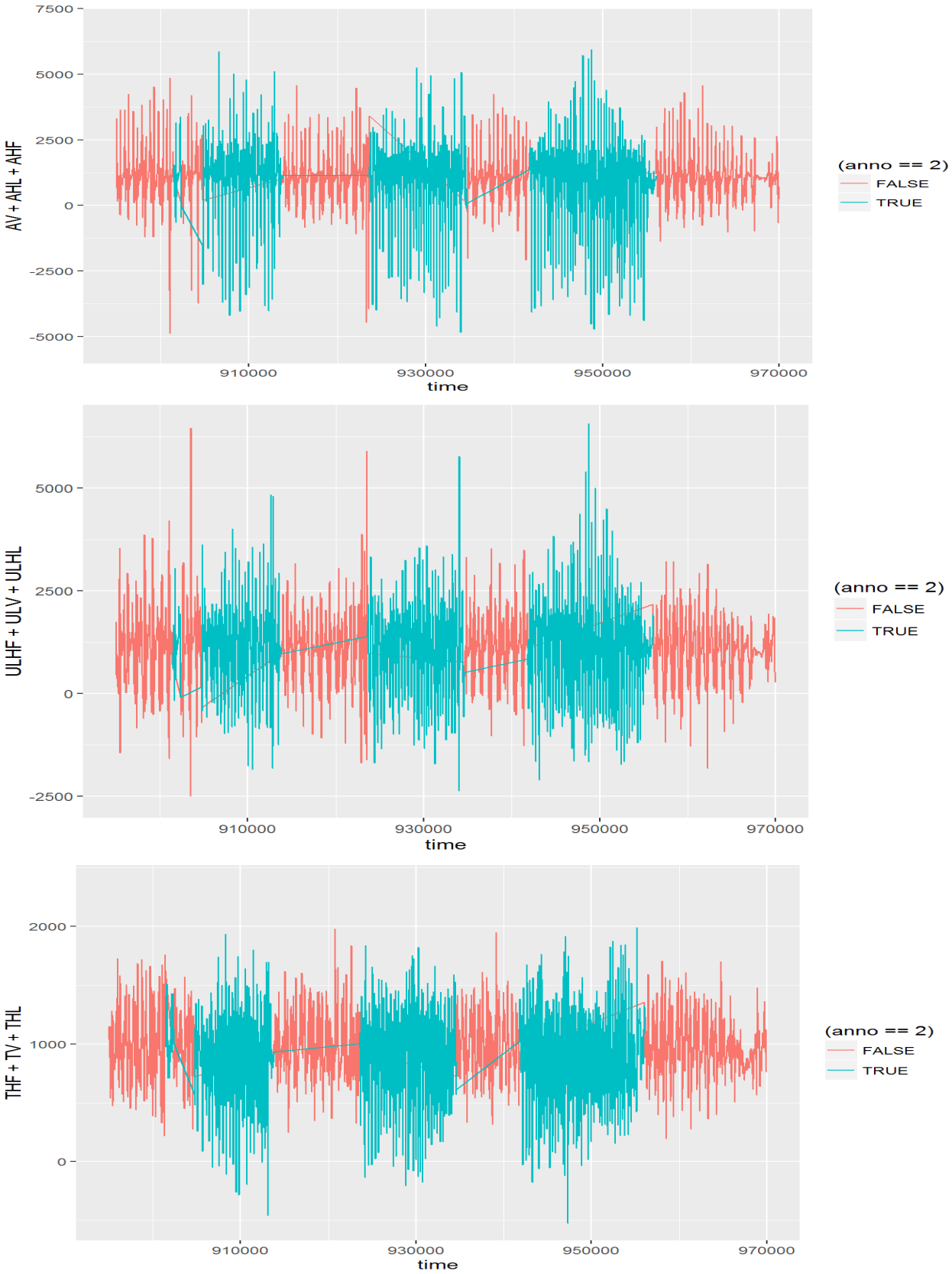


Figure 13. Signal extract from Patient 2 using data from ankle, knee and hip sensors

Simulated Plots:



Red line: no freezing of gait, blue line: freezing of gait happening (anno == 2 means "freezing of gait happening" from the dataset description)

Figure. 14. Acceleration vs Time for Patient 2 (Refer to Appendix A-3 for codes)

3.3.2 Simulation Procedure

Preprocessing:

Before Random Forest algorithm can be ran on the dataset, we have to filter out all the data that is not relevant to the study such as the part of the data when the patient is not doing the walking experiment. In the dataset. These parts of the data are marked with Annotation being 0 in the dataset. As such we only keep the data that have Annotation value 1 and 2 (bigger than 0). We put the filtered data of patient2test1.csv into train data frame and patient2test2.csv into test data frame. (Refer to Appendix B-1 for codes)

Experiment:

For this experiment, similar to the research paper, we will take one test of the patient as training data and the other test as test data. As such, the 2nd patient 1st test will be used as training data while the 2nd test will be used as test data.

However, our method was different from what described in the research paper. In the paper, the data were divided into a progressive array of data within 1 second window or 4 seconds window (for example 1s-window: the first window contains data from 0s to 1s, the 2nd window contains data from 0.005s to 1.005s and so on). Each of the window is then labeled whether it has freezing of gait happening inside or not. Said data are then used to perform machine learning algorithms such as Random Forest or C4.5 on to come up with a way to classify the data. Similar for testing, the data are divided into windows and apply the classification method that the machine learning algorithms come up with to classify the windows. Latency is also measured and taken into account after the classification is done to.

For our experiment, we do not put the data into windows and instead apply the algorithm (Random Forest) directly onto the data. It is because the research paper gives very limited information on how to implement window length. Also, it is possible that for the research paper, a different programming language was used for the analysis. Therefore, we may not be able to do the same with our Random Forest package in R. As such the acceleration at each exact moment is taken into account instead of a whole window of data. Similar for testing, the classification on the test data will classify each exact moment whether it has freezing of gait or not.

Method we used:

Step 1: run random forest on the training data and obtain result
Step 2: do prediction on test data
Step 3: compare the prediction with the original annotation
Step 4: write the result into csv files for later evaluation

* This process is performed four times, evaluating on Ankle acceleration, Upper Leg acceleration, Trunk acceleration as well as all of the acceleration respectively.
(Refer to Appendix B-2 for codes)

3.3.3 Simulation Result

After running random forest, we obtained the following result: (Refer to Appendix B-3 for evaluation code)

	Predict: No FoG	Predict: FoG	True Positives	True Negatives	False Positives	False Negatives
Ankle	56274	8687	55084	1190	1805	6882
Upper Leg	56393	8568	55778	615	1111	7457
Trunk	53486	11475	52657	829	4232	7243
All	57447	7514	55516	1931	1373	6141

We can then calculate Sensitivity (TP/(TP+FN)) and Specificity (TN/(TN+FP))

	Sensitivity (%)	Specificity (%)
Ankle	88.89	39.73
Upper Leg	88.21	35.63
Trunk	87.91	16.38
All	**90.04**	**58.44**

We can see that using all of the data from Ankle, Upper Leg and Trunk gives the best result with 90.04% sensitivity and 58.44% specificity. However, this is still much lower than what was proposed in the research paper with 97.76% sensitivity and 99.75% specificity for Random Forest.

3.3.4 Discussion

This difference as a result could be mainly due to the difference in our method where we do not divide the data into windows due to various reasons stated above. As such the prediction of our experiment is fundamentally different compared to the one done in the research paper. Furthermore, latency is also not taken into account. Thus, this will make our result not as accurate as it should be.

As a result, for any other study regarding this topic on machine learning on freezing of gait, we would recommend that our method should not be carried out since it will not provide a good result.

4. Contributions

As the paper was using a combination of machine learning techniques, wearable sensor and smartphone to build an online platform to detect FoG for PD patients. This is the first attempt to online detection of FoG system in 2009, which aims to replace the old treatment method with smartphones. The contribution made by this experiment would be emphasizing on the **mobility** and **ease of use** of the smartphone device to PD patients in the following several ways.

4.1 Smartphone's Mobility and Convenience

According to the research shown below, the smartphone penetration rate has increased from 16% in 2009 to 37% in 2014 (Global smartphone penetration from 2008 to 2014, 2012). In addition, results released by Google at the Mobile First World Event 2014 indicated that among most of the developed countries, Singapore had a leading penetration rate of 85% on smartphone adoption in 2014(Jimmy T., 2014). The figures have also suggested that the unobtrusive design and computational functions makes smartphone the most convenient wearable device for PD patients.

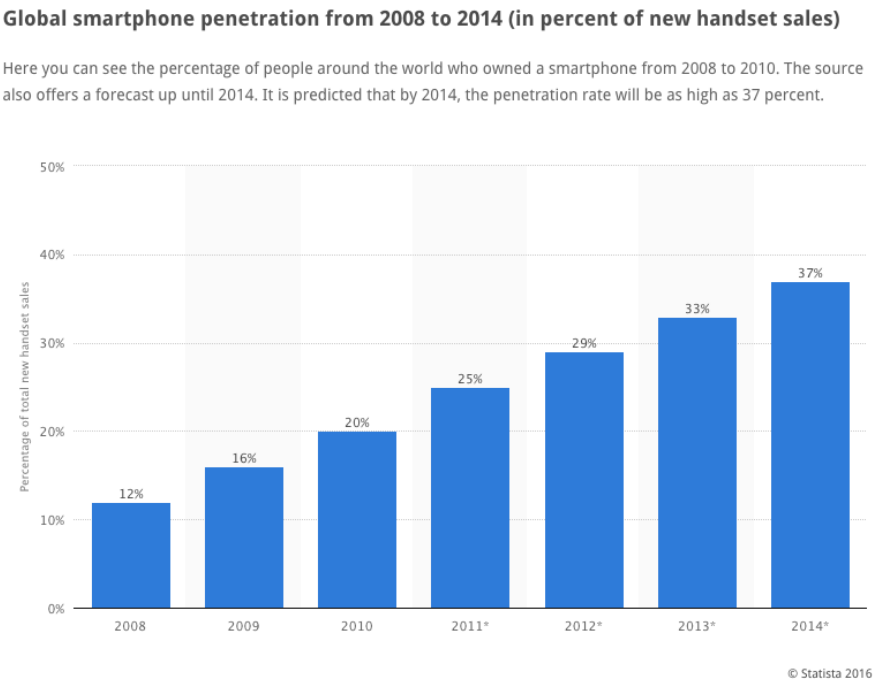


Figure 15. Mobile Penetration Rate Worldwide

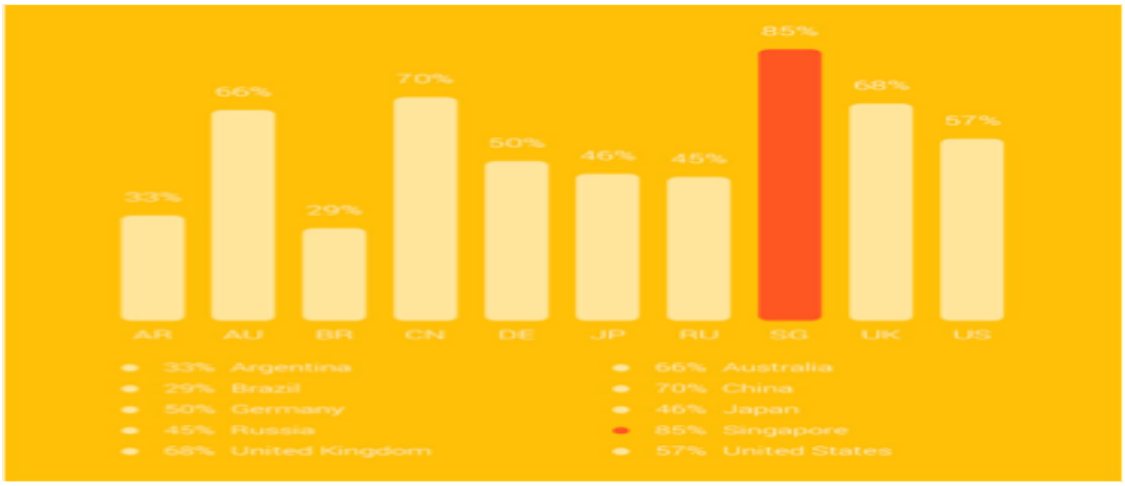


Figure 16. Research released by Google at the Mobile First World Event 2014

Moreover, smartphones enable simplified treatment procedures for PD patients. As the average age of PD patients is around 62 years old, who may face various levels of difficulties in using technical equipment such as the old dedicated wearable device. Leveraging from patient's own smartphone device, which they are more familiar with over their daily usage, it reduces the additional learning efforts for them as well.

Secondly, smartphone as wearable device brings much more convenience to PD patients. Comparing to the old device which was hard to carry and needed external assistance to make it work, smartphone is more portable and will not cause unusual attention from other people when smartphone is carried as.

4.2 Economical Savings

One of the contributions brought by this new treatment method to end users, the PD patients, is the economical savings. Successful detection of FoG also reducing significantly on the injury cost by PD patients if they were falling down accidentally. Reported by Global Burden of Diseases Study (GBD, 2010), back in 1998, the average sanitary cost incurred by a typical fall from an elderly including fees of nursing, hospital, ambulance) was more than $19,000 and the number was expected to exceed $20 billion in total in 2010 (Stevens J, 2006) for US elderly. The introduction and success of FoG detection by smartphone have obviously reduces the economical burden for PD patients with falling symptom and their families.

In the past, the dedicated device was not only difficult to carry, but also costs the PD patients an extra amount of money. However, by leveraging the smartphone online detection system, the PD patients no longer need to purchase the dedicated wearable device as they are able to use their smartphones instead, which means a more cost-saving choice for PD patients.

4.3 Improve on Motion Data Collection

The use of smartphone as wearable devices facilitates collection of motion data. According to Daniel R. (2014), "Motion data is a special feature that built by Nokia for their Lumia phones, based on their beta Motion Monitor app since late 2013. The system app uses GPS and the accelerometer, letting the phone collect your steps and location for use in third-party apps like Bing Health + Fitness through the SensorCore API." Motion data include motion patterns such as movement of walking, running or opening a door will be detected or recognized when sufficient training data are available. By using smartphone instead of dedicated wearable devices, the motion data can be collected consistently in the activities of gait detection. As more and more motion data collected, the level of accuracy for future FoG detection will be improved for PD patients.

4.4 Enabling Telemedicine with Smartphones

As smartphones replace dedicated wearable device, it enables telemedicine between patients and medical institutions to happen. According to American Telemedicine Association, telemedicine is defined as, by leveraging electronic communications including number of

applications and services provided from video, email, smartphones and other forms of telecommunications technology to enable medical information exchange from one site to another. It eliminates distance barriers and improves access to medical services, which would not be consistently available in some distant rural communities. Moreover, by leveraging smartphone as wearable device, the data can be easily transferred and monitored from patient's side to medical institution side.

5. Limitations for Experiments

5.1 Small Sample Size

In the experiment section above, it has been mentioned that there were only 10 PD patients chosen that is relatively small for the experiment. The small sample increases the probability of biased result.

5.2 Statistical Noises

Statistical noises existed in the experiment because PD patients are experiencing low levels of control over their behaviors/movements. This could increase the statistical noise during the experiment. For example, the participants may misunderstand the instruction, which result in invalid movements of patients to be recorded as experiment result. These statistical noises may also contribute to the biased result.

5.3 Limited Resources for Experiment

Although this experiment is carried out by Laboratory for Gait and Neurodynamics, Tel Aviv Sourasky Medical Center, Israel and the Wearable Computing Laboratory, ETH Zurich, Switzerland, the number of physicians who in charge of the experiment activities are limited. Due to the shortage of equipment and labor shortage, the experiment is both inefficient and time consuming, as a lot of time has spent to give instructions to PD patients and lead them complete the experiment tests in.

6. Future Work and Conclusion

Besides detecting Freezing of Gait (FoG) with smartphones and online detection system, we found another study done by a research team from Korea in 2014. They have conducted a similar experiment by placing an accelerometer inside the shoe. (Refer to Figure 17. Measurement System of FoG) This accelerometer will collect the motion data from PD patients and analyze it using custom Matlab program.

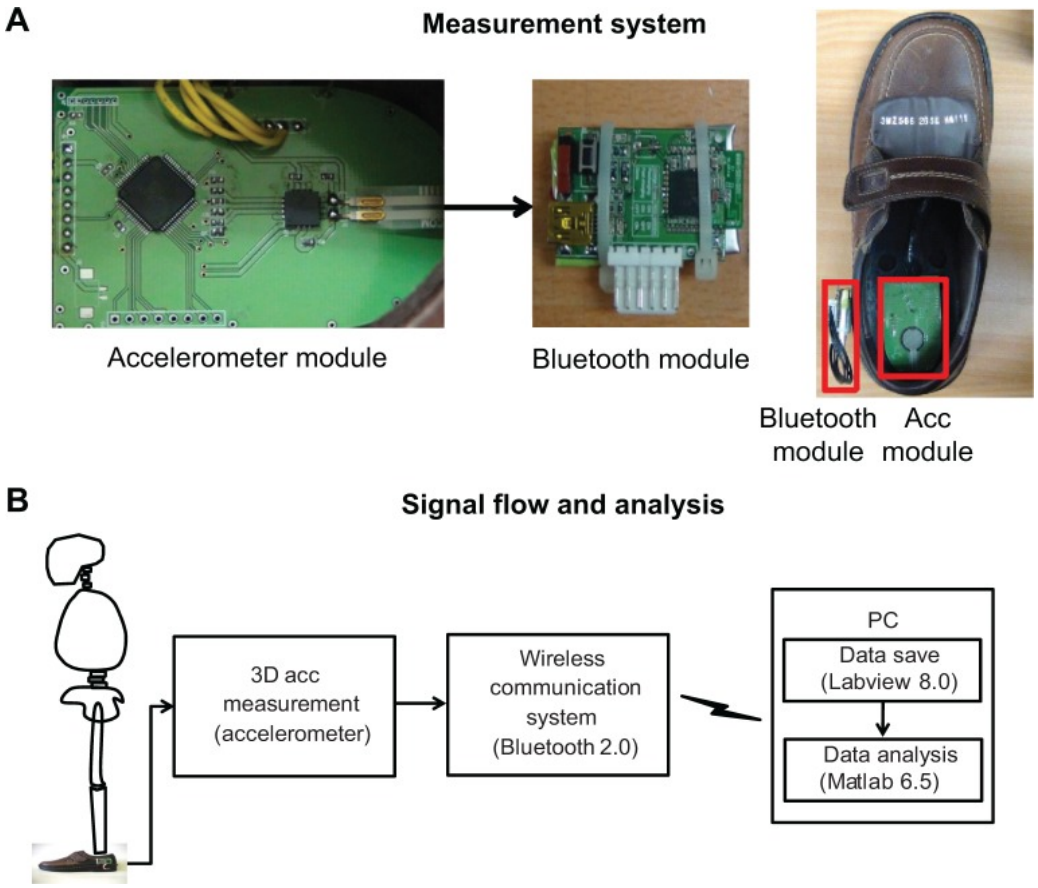


Figure 17. Measurement System of FoG

Both of the studies implies the change of treatment focus from Rhythmic Auditory Stimulation (RAS) to online detection of FoG. With the development of machine learning techniques, time is the only matter for researchers to find the most efficient machine learning techniques, which can detect FoG event accurately.

On the other hand, as discussed in the study, where a single body-worn accelerometer enables the online FoG detection system with a smartphone as wearable computer. The smartphone sensors can be used instead of accelerometer and collect the motion data from PD patients. As such, the online detection system can be supported by the sensor embedded in the smartphone. In 2015, there is a three-year project called CuPID which conducted by an 8 members European Union (EU) funded consortium with researchers at Tel Aviv University, works on a similar idea. They have designed a mobile app which aims to work as virtual physicians for PD patients, as well as treatment for patients with FoG syndrome. Although the project is still under pilot test with sensors embedded in the shoe, the result is promising. This group of researchers is working to recruit more physical therapist and PD patients in order to obtain more movement patterns, which will contribute to the accuracy of their experiment. (American Friends of Tel Aviv University, 2015) This innovative treatment mobile application is planning to be launched in 2018.

Moreover, except for the rapid adoption of smartphones worldwide (Refer to Section 4.1), the age group of smartphone users also shifted to an older generation. For example, in US, there are 61.5% of smartphone users among people who aged between 55 to 64 years old, and there

was almost half of mobile users aged 65 owned a smartphone in 2014. (Smartphone Penetration, Rising in All Age and Income Demos, Hits 75% of the US Mobile Market, 2015)

Since the majority of PD patients are 65+, and they are becoming native smartphone user, there would be high demand for this new FoG online detection system. And it is believed that the smartphone will be used solely to conduct FoG detection without any extra device for PD patients in the future.

7. Reference Lists

Fahn,S. The freezing phenomenon in parkinsonism. Adv. Neurol 67, 53-63 (1995)

Lamberti,P. et al. Freezing gait in Parkinson's disease. Eur. Neurol. 38, 297-301 (1997)

Macht, M., Kaussner, Y., M¨oller, J.C., Stiasny-Kolster, K., Eggert, K.M., Kr¨uger, H.P., Ellgring, H.: Predictors of freezing in Parkinson's disease: A survey of 6,620 patients. Movement Disorders 22 (2007) 953–956

Heremans E, N. A. (2013, 6 13). *Freezing of gait in Parkinson's disease: where are we now?* Retrieved 3 11, 2016, from PubMed: http://www.ncbi.nlm.nih.gov/pubmed/23625316

Parkinson's Disease Information. (23, 6 2009). Retrieved 2 18, 2016, from Parkinsons.org: http://www.parkinsons.org/parkinsons-history.html

Roneil Malkani, C. Z. (2013, 11 1). *Amantadine for freezing of gait in patients with Parkinson's disease*. Retrieved 3 11, 2016, from PWC: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3690306/

Statistics on Parkinson's. (2016). Retrieved 3 11, 2016, from Parkinson's Disease Foundation : http://www.pdf.org/en/parkinson_statistics

Zack Zhu, S. M. (2013). *Real-Time Detection of Freezing of Gait for Parkinson's Disease Patients via Smartphone.* Wearable Computing Laboratory, Swiss Federal Institute of Technology Zurich, Neurodynamics & Gait Research Laboratory, Tel Aviv Sourasky Medical Center, US.

Kwon, Y., Park, S. H., Kim, J.-W., Ho, Y., Jeon, H.-M., Bang, M.-J., … Jeon, H. S. (2014). A practical method for the detection of freezing of gait in patients with Parkinson's disease. *Clinical Interventions in Aging*, *9*, 1709–1719. http://doi.org/10.2147/CIA.S69773

Jimmy T. (2014, 11 5). *Singapore is first in the world for smartphone adoption.* Retrieved 3 12, 2016 from Singapore Hardware Zone.com:
http://www.hardwarezone.com.sg/tech-news-singapore-first-world-smartphone-adoption

Global Burden of Diseases, Injuries, and Risk Factors Study (GBD). Retrieved 7 21, 2010 from:
http://www.healthdata.org/gbd.

Stevens JA, Corso PS, Finkelstein EA, Miller TR.(Oct, 2006). *The costs of fatal and non-fatal falls among older adults.* Retrieved from PubMed:
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2563445/

Daniel R.(2014, 8 29). *Microsoft updates Motion Data step counter for newer Lumia Windows Phones.* Retrieved from Windows Central:
http://www.windowscentral.com/microsoft-updates-motion-data-step-counter-newer-lumia-windows-phones

American Telemedicine Association. *What is telemedicine?* Retrieved 3 15, 2016 from
http://www.americantelemed.org/about-telemedicine/what-is-telemedicine#.VwVNXxN95-U

American Friends of Tel Aviv University. "Smartphone app may prevent dangerous freezing of gait in Parkinson's patients." ScienceDaily. ScienceDaily, 25 June 2015. Retrieved 4 1, 2016 from HTTP://www.sciencedaily.com/releases/2015/06/150625131150.htm

Appendix A: Plot Simulation Codes

Each of the plots used inside this report is generated using R. For all these plots, the data for plotting is generated by observing the graphs that are included in the research paper and recorded down into different csv files for each plot. This process is done manually. After that, R is used to read the csv files and draw the respective plots with ggplot2.

1) 4 Histograms comparing C4.5 and RF in 1s-window and 4-window:

Code:

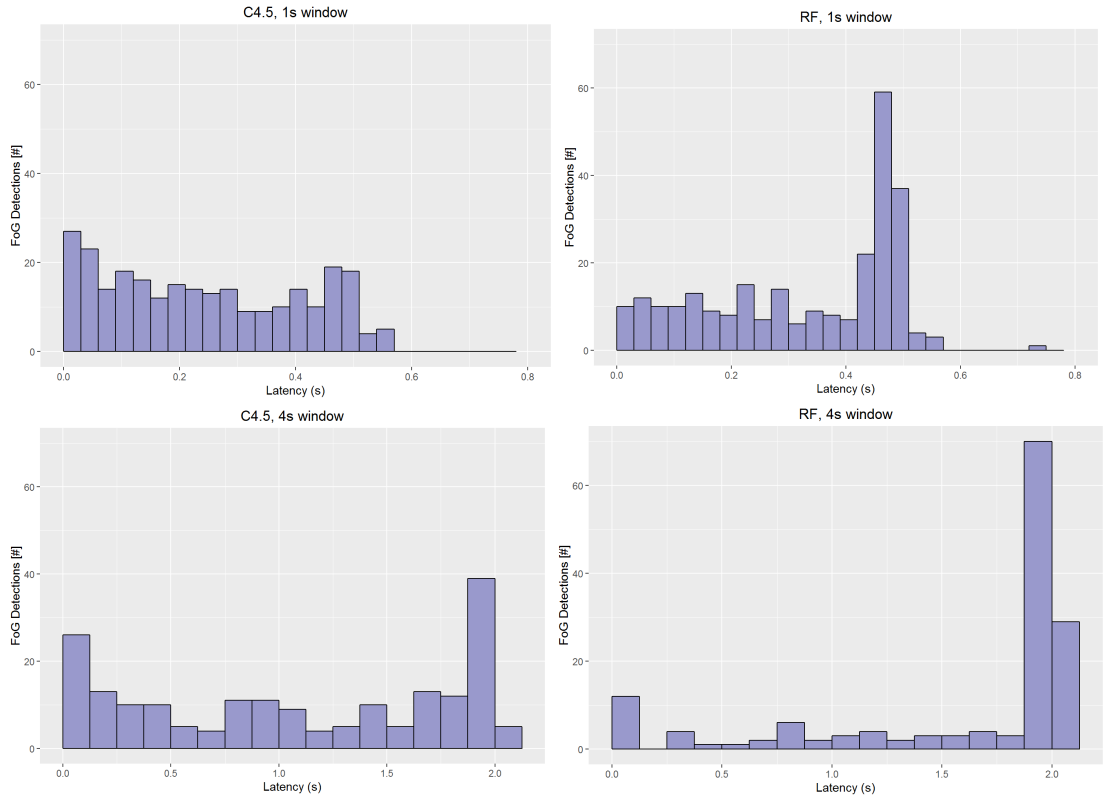
````
```{r, echo=FALSE}
c45_1s.data <- read.csv("c45_1s.csv")
ggplot(c45_1s.data, aes(Latency)) + geom_histogram(binwidth =
(0.03), fill="#9999CC",colour = "black", na.rm=TRUE) +
scale_x_continuous(limits = c(0, 0.8)) + ylim(0, 70) +
labs(x="Latency (s)", y="FoG Detections [#]", title = "C4.5, 1s
window")
```

```{r, echo=FALSE}
rf_1s.data <- read.csv("rf_1s.csv")
ggplot(rf_1s.data, aes(Latency)) + geom_histogram(binwidth =
(0.03), fill="#9999CC",colour = "black", na.rm=TRUE) +
scale_x_continuous(limits = c(0, 0.8)) + ylim(0, 70) +
labs(x="Latency (s)", y="FoG Detections [#]", title = "RF, 1s
window")
```

```{r, echo=FALSE}
c45_4s.data <- read.csv("c45_4s.csv")
ggplot(c45_4s.data, aes(Latency)) + geom_histogram(binwidth =
(0.125), fill="#9999CC",colour = "black", na.rm=TRUE) +
scale_x_continuous(limits = c(0, 2.125)) + ylim(0, 70) +
labs(x="Latency (s)", y="FoG Detections [#]", title = "C4.5, 4s
window")
```

```{r, echo=FALSE}
rf_4s.data <- read.csv("rf_4s.csv")
ggplot(rf_4s.data, aes(Latency)) + geom_histogram(binwidth =
(0.125), fill="#9999CC",colour = "black", na.rm=TRUE) +
scale_x_continuous(limits = c(0, 2.125)) + ylim(0, 70) +
labs(x="Latency (s)", y="FoG Detections [#]", title = "RF, 4s
window")
```
````

Result:



For these four graphs, geom_histogram was used to plot them with x-axis and y-axis limit (for 1s-window: x-axis: from 0 to 0.8; y-axis: from 0 to 70. 4s-window: x-axis: 0 to 2.125; y-axis: 0 to 70)

2) Performance and latency vs. window length plot

Code:

```
```{r, echo=FALSE}
rf_4s.data <- read.csv("sen-spe_latency.csv")

# two plots
p1 <- ggplot(rf_4s.data, aes(Time,minSenSpe)) + geom_line(colour =
"blue", size= 1.2) + labs(x="Window length [s]",
y="min(Sensitivity, Speci!city) [%]") + scale_x_continuous(limits
= c(0, 8), breaks=seq(0,8,1)) + ylim(95, 100) + theme_bw()
p2 <- ggplot(rf_4s.data, aes(Time,latency)) + geom_line(colour =
"green", linetype="dashed", size= 1.2) + labs(x="Window length
[s]", y="Mean latency [s]") + scale_x_continuous(limits = c(0, 8),
breaks=seq(0,8,1)) + ylim(0, 5)+ theme_bw() %+replace%
theme(panel.background = element_rect(fill = NA))
p2

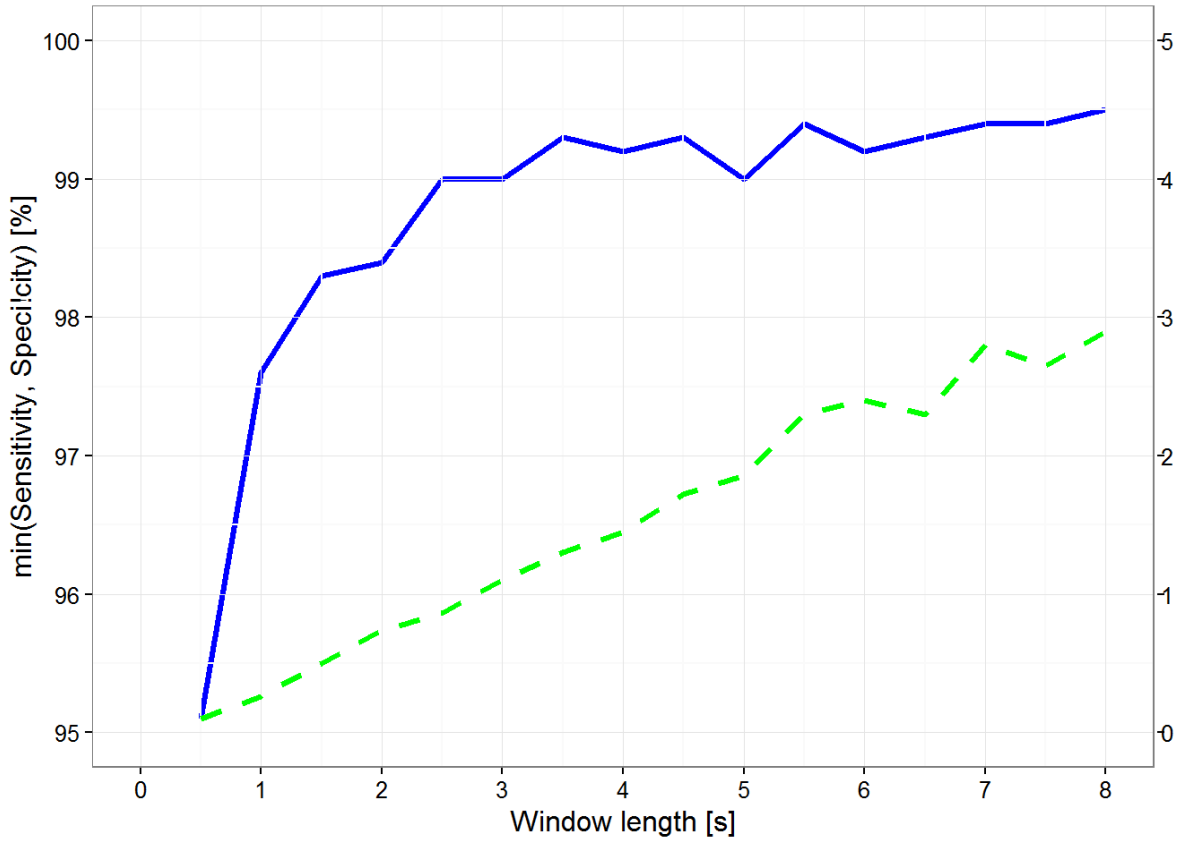
# extract gtable
g1 <- ggplot_gtable(ggplot_build(p1))
g2 <- ggplot_gtable(ggplot_build(p2))
```
```

```
# overlap the panel of 2nd plot on that of 1st plot
pp <- c(subset(g1$layout, name == "panel", se = t:r))
g <- gtable_add_grob(g1, g2$grobs[[which(g2$layout$name ==
"panel")]], pp$t,
   pp$l, pp$b, pp$l)

# axis tweaks
ia <- which(g2$layout$name == "axis-l")
ga <- g2$grobs[[ia]]
ax <- ga$children[[2]]
ax$widths <- rev(ax$widths)
ax$grobs <- rev(ax$grobs)
ax$grobs[[1]]$x <- ax$grobs[[1]]$x - unit(1, "npc") + unit(0.15,
"cm")
g <- gtable_add_cols(g, g2$widths[g2$layout[ia, ]$l],
length(g$widths)-1)
g <- gtable_add_grob(g, ax, pp$t, length(g$widths)-1, pp$b)

grid.draw(g)
```

Result:



Here the plot has 2 y-axes with the left one being the minimum between sensitivity and specificity and the right one being latency. Since ggplot2 does not support having plots with 2 y-axes, we need to extract the plot data from two separate plots and and

try to fit them into one. This causes the right y-axis to not have axis name. There is also no legend for what each of the line represent (blue line for min(Sensitivity, Specificity) vs window length. Green dotted line for latency vs window length).

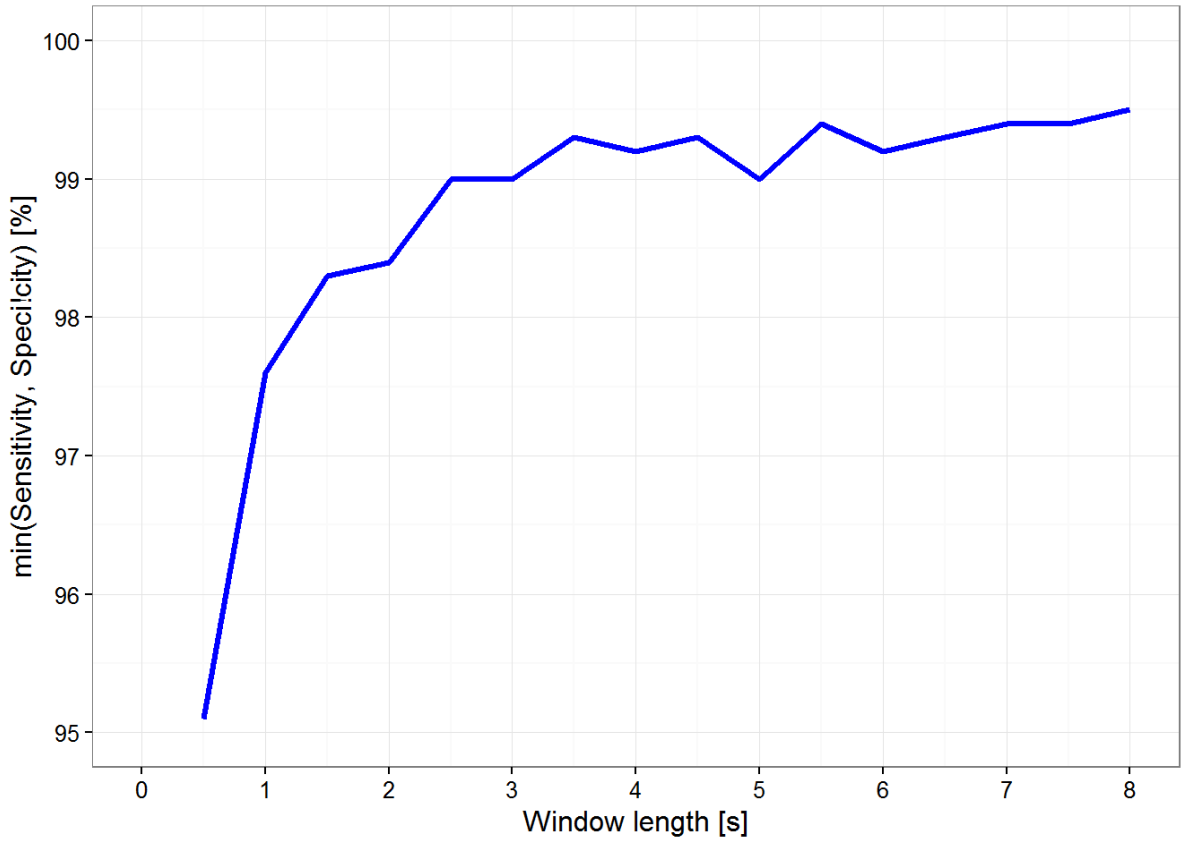
The 2 separate plots are as follow:
Code:

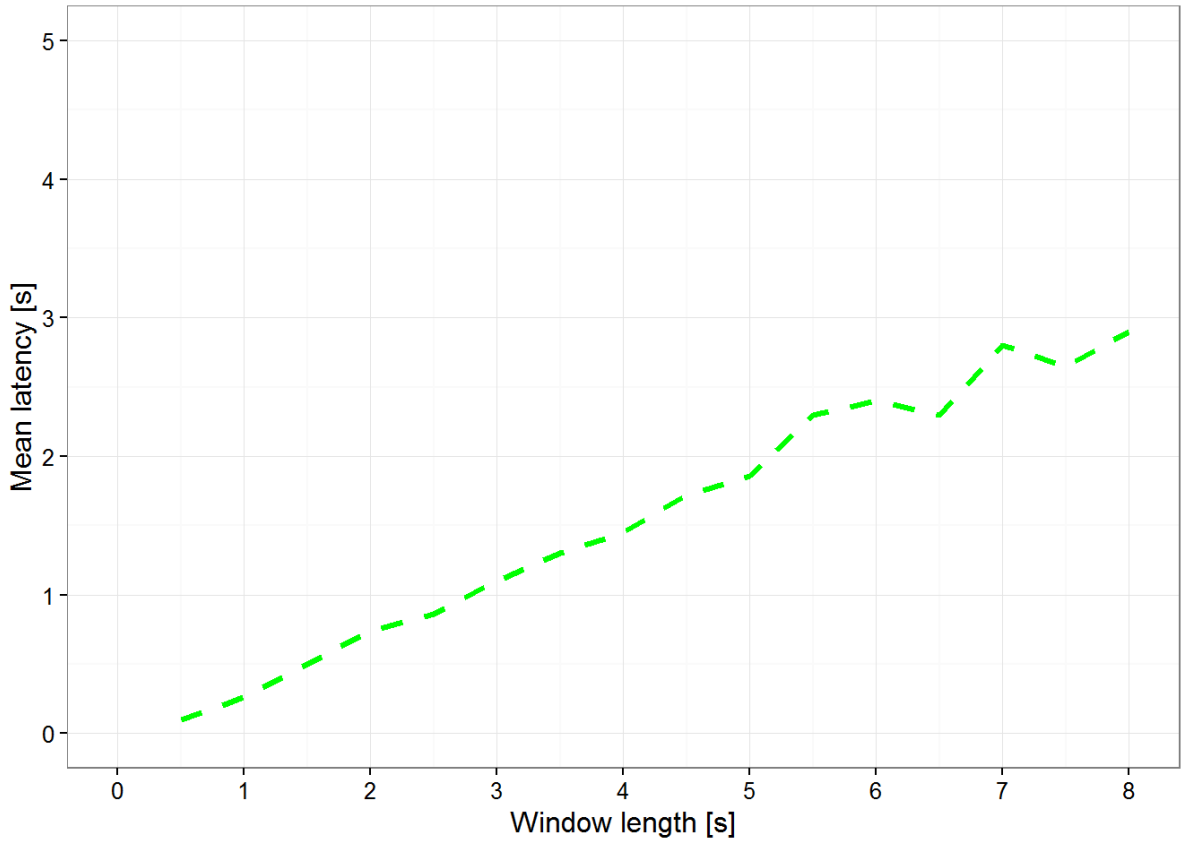
```
```{r, echo=FALSE}
rf_4s.data <- read.csv("sen-spe_latency.csv")

# two plots
ggplot(rf_4s.data, aes(Time,minSenSpe)) + geom_line(colour =
"blue", size= 1.2) + labs(x="Window length [s]",
y="min(Sensitivity, Speci!city) [%]") + scale_x_continuous(limits
= c(0, 8), breaks=seq(0,8,1)) + ylim(95, 100) + theme_bw()

ggplot(rf_4s.data, aes(Time,latency)) + geom_line(colour =
"green", linetype="dashed", size= 1.2) + labs(x="Window length
[s]", y="Mean latency [s]") + scale_x_continuous(limits = c(0, 8),
breaks=seq(0,8,1)) + ylim(0, 5)+ theme_bw() %+replace%
theme(panel.background = element_rect(fill = NA))
```
```

Result:

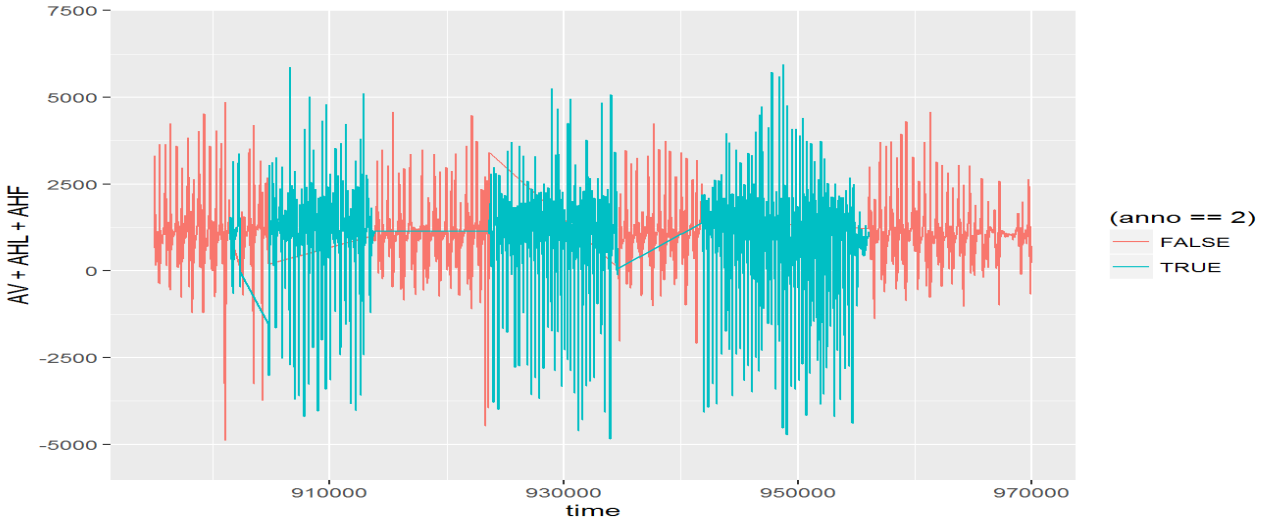

100
99
98
97
96
95
min(Sensitivity, Speci!city) [%]
0
1
2
3
4
5
6
7
8
Window length [s]

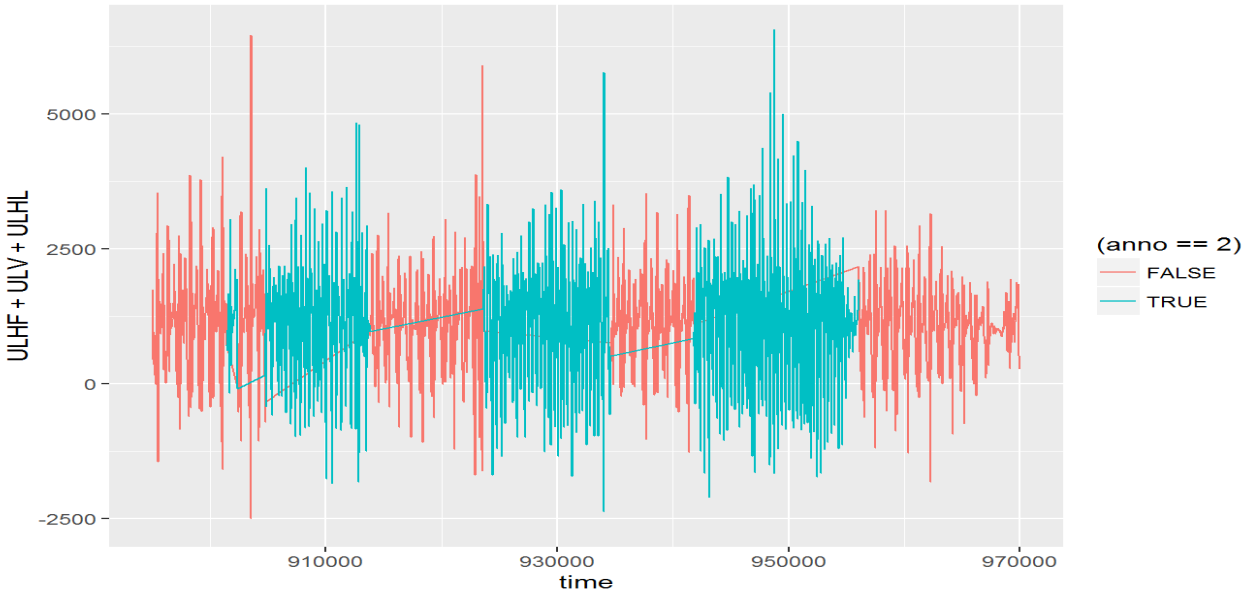

5
4
3
2
1
0
Mean latency [s]
0
1
2
3
4
5
6
7
8
Window length [s]

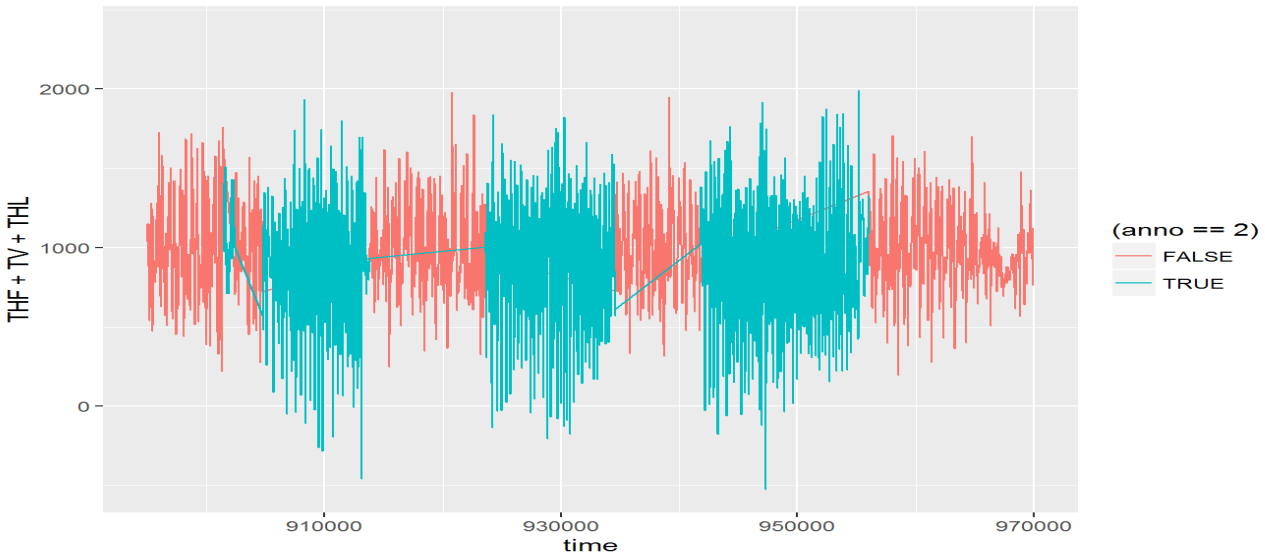
3) Acceleration vs time plots for patient 2

Code:

```
```{r, echo=FALSE}
testRaw <- read.csv("patient2test1.csv")
testRaw2 <- read.csv("patient2test2.csv")
test <- dplyr::filter(testRaw, anno > 0)
test2 <- dplyr::filter(testRaw2, anno > 0)
ggplot(test, aes(time, AV+AHL+AHF, colour=(anno==2))) +
geom_path()+ scale_x_continuous(limits = c(895000, 970000))
ggplot(test, aes(time, ULHF + ULV + ULHL, colour=(anno==2))) +
geom_path()+ scale_x_continuous(limits = c(895000, 970000))
ggplot(test, aes(time, THF + TV + THL, colour=(anno==2))) +
geom_path()+ scale_x_continuous(limits = c(895000, 970000))
```
```







**Red line: no freezing of gait, blue line: freezing of gait happening (anno == 2 means "freezing of gait happening" from the dataset description)*

For these 3 plots, the color of the line is differentiated by the value of anno (if anno = 2 then the color is blue, anno != 2 then the color is red). Here geom_path is used along with limiting x axis between 895000 and 970000

Appendix B: Random Forest Simulation Codes

1) Preprocessing

Before Random Forest algorithm can be run on the dataset, we have to filter out all the data that is not relevant to the study such as the part of the data when the patient is not doing the walking experiment. In the dataset. These parts of the data are marked with Annotation being 0 in the dataset. As such we only keep the data that have Annotation value 1 and 2 (bigger than 0). We put the filtered data of patient2test1.csv into train data frame and patient2test2.csv into test data frame.

Code:

```
trainRaw <- read.csv("patient2test1.csv")
train <- dplyr::filter(trainRaw, anno > 0)

testRaw <- read.csv("patient2test2.csv")
test <- dplyr::filter(testRaw, anno > 0)
```

2) Random Forest

Our method:

- Step 1: run random forest on the training data and obtain result
- Step 2: do prediction on test data
- Step 3: compare the prediction with the original annotation
- Step 4: write the result into csv files for later evaluation

* This process is performed four times, evaluating on Ankle acceleration, Upper Leg acceleration, Trunk acceleration as well as all of the acceleration respectively.

Code:

```
#-----------------
# Ankle
rf <- randomForest(as.factor(anno) ~ AHF + AV + AHL, data=train,
importance=TRUE)
varImpPlot(rf)

Prediction <- predict(rf, test)
test.predicted <- cbind(test, Prediction)
test.predicted <- dplyr::mutate(test.predicted, correct = anno ==
Prediction)
submit <- data.frame(time = test.predicted$time,anno =
test.predicted$anno, prediction = Prediction, Correct =
test.predicted$correct)
write.csv(submit, file = "ankle.csv", row.names = FALSE)

#-----------------
# Upper Leg
rf <- randomForest(as.factor(anno) ~ ULHF + ULV + ULHL,
data=train, importance=TRUE)
varImpPlot(rf)
```

```
Prediction <- predict(rf, test)
test.predicted <- cbind(test, Prediction)
test.predicted <- dplyr::mutate(test.predicted, correct = anno ==
Prediction)
submit <- data.frame(time = test.predicted$time,anno =
test.predicted$anno, prediction = Prediction, Correct =
test.predicted$correct)
write.csv(submit, file = "upperLeg.csv", row.names = FALSE)

#-----------------
# Trunk
rf <- randomForest(as.factor(anno) ~ THF + TV + THL, data=train,
importance=TRUE)
varImpPlot(rf)

Prediction <- predict(rf, test)
test.predicted <- cbind(test, Prediction)
test.predicted <- dplyr::mutate(test.predicted, correct = anno ==
Prediction)
submit <- data.frame(time = test.predicted$time,anno =
test.predicted$anno, prediction = Prediction, Correct =
test.predicted$correct)
write.csv(submit, file = "trunk.csv", row.names = FALSE)

#-----------------
# All
rf <- randomForest(as.factor(anno) ~ AHF + AV + AHL + ULHF + ULV +
ULHL + THF + TV + THL, data=train, importance=TRUE)
varImpPlot(rf)

Prediction <- predict(rf, test)
test.predicted <- cbind(test, Prediction)
test.predicted <- dplyr::mutate(test.predicted, correct = anno ==
Prediction)
submit <- data.frame(time = test.predicted$time,anno =
test.predicted$anno, prediction = Prediction, Correct =
test.predicted$correct)
write.csv(submit, file = "all.csv", row.names = FALSE)
```

3) Result Evaluation

After running the script, we have 4 csv files containing the prediction as well as whether the prediction is correct. We can check how many prediction is correct. How many True Positives, False Positives, True Negatives and False Negatives.

Code:

````
#Ankle
```{r, echo=FALSE}
result <- read.csv("ankle.csv")
table(result$Correct)

print('only no freezing of gait:')
resultFreezing <- dplyr::filter(result, anno == 1)
table(resultFreezing$Correct)
print("only freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 2)
table(resultFreezing$Correct)
```


#Upper Leg
```{r, echo=FALSE}
result <- read.csv("upperLeg.csv")
table(result$Correct)

print("only no freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 1)
table(resultFreezing$Correct)
print("only freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 2)
table(resultFreezing$Correct)
```


#Trunk
```{r, echo=FALSE}
result <- read.csv("trunk.csv")
table(result$Correct)

print("only no freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 1)
table(resultFreezing$Correct)
print("only freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 2)
table(resultFreezing$Correct)
```


#All
```{r, echo=FALSE}
result <- read.csv("all.csv")
table(result$Correct)

print("only no freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 1)
table(resultFreezing$Correct)
print("only freezing of gait:")
````
````

```
resultFreezing <- dplyr::filter(result, anno == 2)
table(resultFreezing$Correct)
```

Appendix C: List of Project Files

- Dataset_fog_release folder: the original Daphnet Dataset
- all.csv: contains the result of the test of random forest on test data using all data from ankle, upper leg and trunk for analysis
- ankle.csv:contains the result of the test of random forest on test data using only data from ankle for analysis
- c45_1s.csv: contains the data for plotting the histogram with c4.5 in 1s-window
- c45_4s.csv: contains the data for plotting the histogram with c4.5 in 4s-window
- evaluate.Rmd: script for *Result Evaluation* of the random forest experiment
- patient2test1.csv: data of 2nd patient, 1st test
- patient2test2.csv: data of 2nd patient, 2nd test
- plot.Rmd: script for plotting all the plots using R
- randomForest.Rmd: script for performing Random Forest on the data using R
- rf_1s.csv: contains the data for plotting the histogram with random forest in 1s-window
- rf_4s.csv: contains the data for plotting the histogram with random forest in 4s-window
- sen-spe_latency.csv: contains the data for plotting the line graph with 2 y-axes
- trunk.csv: contains the result of the test of random forest on test data using only data from trunk for analysis
- upperLeg.csv: contains the result of the test of random forest on test data using only data from upper leg for analysis

Appendix D: R Script Content

- plot.Rmd:

````
---
title: "Plots"
output: html_document
---

```{r,message=FALSE, echo=FALSE}
library(ggplot2)
library(dplyr)
library(gtable)
library(grid)
```

```{r, echo=FALSE}
c45_1s.data <- read.csv("c45_1s.csv")
ggplot(c45_1s.data, aes(Latency)) + geom_histogram(binwidth =
(0.03), fill="#9999CC",colour = "black", na.rm=TRUE) +
scale_x_continuous(limits = c(0, 0.8)) + ylim(0, 70) +
labs(x="Latency (s)", y="FoG Detections [#]", title = "C4.5, 1s
window")
```

```{r, echo=FALSE}
rf_1s.data <- read.csv("rf_1s.csv")
ggplot(rf_1s.data, aes(Latency)) + geom_histogram(binwidth =
(0.03), fill="#9999CC",colour = "black", na.rm=TRUE) +
scale_x_continuous(limits = c(0, 0.8)) + ylim(0, 70) +
labs(x="Latency (s)", y="FoG Detections [#]", title = "RF, 1s
window")
```

```{r, echo=FALSE}
c45_4s.data <- read.csv("c45_4s.csv")
ggplot(c45_4s.data, aes(Latency)) + geom_histogram(binwidth =
(0.125), fill="#9999CC",colour = "black", na.rm=TRUE) +
scale_x_continuous(limits = c(0, 2.125)) + ylim(0, 70) +
labs(x="Latency (s)", y="FoG Detections [#]", title = "C4.5, 4s
window")
```

```{r, echo=FALSE}
rf_4s.data <- read.csv("rf_4s.csv")
ggplot(rf_4s.data, aes(Latency)) + geom_histogram(binwidth =
(0.125), fill="#9999CC",colour = "black", na.rm=TRUE) +
scale_x_continuous(limits = c(0, 2.125)) + ylim(0, 70) +
labs(x="Latency (s)", y="FoG Detections [#]", title = "RF, 4s
window")
````
````

```
```

```{r, echo=FALSE}
rf_4s.data <- read.csv("sen-spe_latency.csv")

# two plots
p1 <- ggplot(rf_4s.data, aes(Time,minSenSpe)) + geom_line(colour =
"blue", size= 1.2) + labs(x="Window length [s]",
y="min(Sensitivity, Speci!city) [%]") + scale_x_continuous(limits
= c(0, 8), breaks=seq(0,8,1)) + ylim(95, 100) + theme_bw()
p2 <- ggplot(rf_4s.data, aes(Time,latency)) + geom_line(colour =
"green", linetype="dashed", size= 1.2) + labs(x="Window length
[s]", y="Mean latency [s]") + scale_x_continuous(limits = c(0, 8),
breaks=seq(0,8,1)) + ylim(0, 5)+ theme_bw() %+replace%
theme(panel.background = element_rect(fill = NA))
p2

# extract gtable
g1 <- ggplot_gtable(ggplot_build(p1))
g2 <- ggplot_gtable(ggplot_build(p2))

# overlap the panel of 2nd plot on that of 1st plot
pp <- c(subset(g1$layout, name == "panel", se = t:r))
g <- gtable_add_grob(g1, g2$grobs[[which(g2$layout$name ==
"panel")]], pp$t,
   pp$l, pp$b, pp$l)

# axis tweaks
ia <- which(g2$layout$name == "axis-l")
ga <- g2$grobs[[ia]]
ax <- ga$children[[2]]
ax$widths <- rev(ax$widths)
ax$grobs <- rev(ax$grobs)
ax$grobs[[1]]$x <- ax$grobs[[1]]$x - unit(1, "npc") + unit(0.15,
"cm")
g <- gtable_add_cols(g, g2$widths[g2$layout[ia, ]$l],
length(g$widths)-1)
g <- gtable_add_grob(g, ax, pp$t, length(g$widths)-1, pp$b)

grid.draw(g)

```

```{r, echo=FALSE}
rf_4s.data <- read.csv("sen-spe_latency.csv")

# two plots
ggplot(rf_4s.data, aes(Time,minSenSpe)) + geom_line(colour =
```

```
"blue", size= 1.2) + labs(x="Window length [s]",
y="min(Sensitivity, Speci!city) [%]") + scale_x_continuous(limits
= c(0, 8), breaks=seq(0,8,1)) + ylim(95, 100) + theme_bw()

ggplot(rf_4s.data, aes(Time,latency)) + geom_line(colour =
"green", linetype="dashed", size= 1.2) + labs(x="Window length
[s]", y="Mean latency [s]") + scale_x_continuous(limits = c(0, 8),
breaks=seq(0,8,1)) + ylim(0, 5)+ theme_bw() %+replace%
theme(panel.background = element_rect(fill = NA))
```

```{r, echo=FALSE}
testRaw <- read.csv("patient2test1.csv")
testRaw2 <- read.csv("patient2test2.csv")
test <- dplyr::filter(testRaw, anno > 0)
test2 <- dplyr::filter(testRaw2, anno > 0)
ggplot(test, aes(time, AV+AHL+AHF, colour=(anno==2))) +
geom_path()+ scale_x_continuous(limits = c(895000, 970000))
ggplot(test, aes(time, ULHF + ULV + ULHL, colour=(anno==2))) +
geom_path()+ scale_x_continuous(limits = c(895000, 970000))
ggplot(test, aes(time, THF + TV + THL, colour=(anno==2))) +
geom_path()+ scale_x_continuous(limits = c(895000, 970000))
```
```

- randomForest.Rmd:

```
---
title: "RandomForest"
output: html_document

```{r,message=FALSE, echo=FALSE}
library(ggplot2)
library(dplyr)
library(gtable)
library(grid)
library(randomForest)
```

```{r, echo=FALSE}
trainRaw <- read.csv("patient2test1.csv")
train <- dplyr::filter(trainRaw, anno > 0)

testRaw <- read.csv("patient2test2.csv")
test <- dplyr::filter(testRaw, anno > 0)

rf <- randomForest(as.factor(anno) ~ AHF + AV + AHL, data=train,
importance=TRUE)
varImpPlot(rf)
```

```
Prediction <- predict(rf, test)
test.predicted <- cbind(test, Prediction)
test.predicted <- dplyr::mutate(test.predicted, correct = anno ==
Prediction)
submit <- data.frame(time = test.predicted$time,anno =
test.predicted$anno, prediction = Prediction, Correct =
test.predicted$correct)
write.csv(submit, file = "ankle.csv", row.names = FALSE)

rf <- randomForest(as.factor(anno) ~ ULHF + ULV + ULHL,
data=train, importance=TRUE)
varImpPlot(rf)

Prediction <- predict(rf, test)
test.predicted <- cbind(test, Prediction)
test.predicted <- dplyr::mutate(test.predicted, correct = anno ==
Prediction)
submit <- data.frame(time = test.predicted$time,anno =
test.predicted$anno, prediction = Prediction, Correct =
test.predicted$correct)
write.csv(submit, file = "upperLeg.csv", row.names = FALSE)

rf <- randomForest(as.factor(anno) ~ THF + TV + THL, data=train,
importance=TRUE)
varImpPlot(rf)

Prediction <- predict(rf, test)
test.predicted <- cbind(test, Prediction)
test.predicted <- dplyr::mutate(test.predicted, correct = anno ==
Prediction)
submit <- data.frame(time = test.predicted$time,anno =
test.predicted$anno, prediction = Prediction, Correct =
test.predicted$correct)
write.csv(submit, file = "trunk.csv", row.names = FALSE)

rf <- randomForest(as.factor(anno) ~ AHF + AV + AHL + ULHF + ULV +
ULHL + THF + TV + THL, data=train, importance=TRUE)
varImpPlot(rf)

Prediction <- predict(rf, test)
test.predicted <- cbind(test, Prediction)
test.predicted <- dplyr::mutate(test.predicted, correct = anno ==
Prediction)
submit <- data.frame(time = test.predicted$time,anno =
test.predicted$anno, prediction = Prediction, Correct =
test.predicted$correct)
write.csv(submit, file = "all.csv", row.names = FALSE)
```

```
```

- evaluate.Rmd:

```
---
title: "evaluate"
output: html_document
---


```{r,message=FALSE, echo=FALSE}
library(ggplot2)
library(dplyr)
library(gtable)
library(grid)
library(randomForest)
```

Ankle
```{r, echo=FALSE}
result <- read.csv("ankle.csv")
table(result$Correct)

print('only no freezing of gait:')
resultFreezing <- dplyr::filter(result, anno == 1)
table(resultFreezing$Correct)
print("only freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 2)
table(resultFreezing$Correct)
```

Upper Leg
```{r, echo=FALSE}
result <- read.csv("upperLeg.csv")
table(result$Correct)

print("only no freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 1)
table(resultFreezing$Correct)
print("only freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 2)
table(resultFreezing$Correct)
```

Trunk
```{r, echo=FALSE}
result <- read.csv("trunk.csv")
table(result$Correct)
```
```

```
print("only no freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 1)
table(resultFreezing$Correct)
print("only freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 2)
table(resultFreezing$Correct)
```

All
```{r, echo=FALSE}
result <- read.csv("all.csv")
table(result$Correct)

print("only no freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 1)
table(resultFreezing$Correct)
print("only freezing of gait:")
resultFreezing <- dplyr::filter(result, anno == 2)
table(resultFreezing$Correct)
```