Faculty of Natural and Mathematical Sciences
Department of Informatics

Bush House, 30 Aldwych
Strand Campus
London WC2B 4BG
United Kingdom
https://www.kcl.ac.uk/nms/depts/informatics

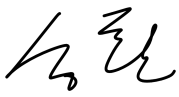
7CCSMDPJ
Individual Project Submission 2018/19

Name: Seunghwan Kim
Student Number: 1817642
Degree Programme: MSc Data science
Project Title: Optimal sensor placement for everyday motion tracking of a neurological patient
Supervisor: Dr Brendan Michael
Word count: 12316

RELEASE OF PROJECT
Following the submission of your project, the Department would like to make it publicly available via the library electronic resources. You will retain copyright of the project.

■ I **agree** to the release of my project

☐ I **do not** agree to the release of my project

Signature: Seunghwan Kim 

Date: 20/08/2019

Department of Informatics

King's College London

United Kingdom

7CCSMDPJMSc Project

OPTIMAL SENSOR PLACEMENT FOR EVERYDAY MOTION TRACKING OF A NEUROLOGICAL PATIENT

Student Name: Seunghwan Kim

Student Number: 1817642

Degree Programme: MSc Data science

Supervisor's Name: Dr Brendan Michael

This dissertation is submitted for the degree of MSc in Data science

ABSTRACT

Motion capture technology has made a lot of progress since Eadweard Muybridge's 19th-century motion-pictures of animal locomotion. Currently, it is possible to not only store images of an object in a two-dimensional plane as a picture or video but also to implement precise movements of the object in 3D space by motion tracking systems (MOCAP), also called motion capture. These technologies are used in modern society for medical purposes especially with great importance for neurological treatment by continuously monitoring the patient's movements. However, modern sensor-based motion-tracking technology has its limitations in using rigidly attached sensors. This can cause a great deal of discomfort in constantly observing patients' daily life. This study takes such problem seriously and attempts to find optimal sensor placements in everyday clothing to solve the problem without losing the motion-tracking capability of the existing sensor placement. To find optimal sensor placement to solve the problem, machine learning and mutual information are used. A series of processes have been described in this paper to produce results based on the two main theories with visualization. As a result, 4 suitable sensor positions have been derived. These results allow for new technologies in making a neuro-related patient free from rigidly attached sensors directly to clothes or skin by finding optimal sensor placement for everyday wear.

ACKNOWLEDGMENTS

I would like to express my special thanks of gratitude to my project supervisor “Dr Brenden Michael” for your continued consideration and guidance in completing individual project under difficult circumstance that I could not attend meeting for personal reason. And thank you again for providing an alternative to the circumstance.

TABLE OF CONTENTS

GLOSSARY

ANN	Artificial Neural Network	18
DFT	Discrete Fourier Transform	16
EEF	Electroencephalogram	11
EEG	Electroencephalography	11
FFT	Fast Fourier Transform	16
FT	Fourier Transform	16
IR-UWB	Impulse Radio Ultra-Wideband	20
ICA	Independent Component Analysis	11
KNN	K-Nearest Neighbors	19
MATLAB	Matrix Laboratory (software)	9
MI	Mutual Information	9
MMI	Multivariate Mutual Information	41
MOCAP	Motion Capture	10
MPIDE	Multi-Platform IDE	18
NN	Neural Network	18
PCA	Principal Component Analysis	9
RELIEF	Feature selection algorithm	19
RMS	Root Mean Square	20
SBS	Sequential Backward Selection	14
SFS	Sequential Feature Selection	9
SFS	Sequential Forward Selection	14
SSCP	Sums of Squares and Cross Products	12
SVM	Support Vector Machine	9
W	Work	16
mRMR	Minimum Redundancy Maximum Relevance	19

LIST OF FIGURES AND TABLES

List of Figures

List of Tables

1 INTRODUCTION

1.1 Motivation

Over the past few decades, continuous studies of neuro-related patients were conducted, and continuous patient observing was unavoidable for the treatment, for example monitoring continuous inspection of nerve-related patients' status (e.g. Parkinson's disease) [1]. For such motion-tracking technology, existing motion capture technologies can be divided into camera-based motion capture and sensor-based systems [2]. The camera-based motion capture is much more prevalent in many areas in the field of movie and sport to make computer graphic or analyze activities of athletes respectively as well as the sensor-based systems are using sensors affixed onto garment and trace to get the behaviours of the subject [2]. Both technologies are certainly state-of-the-art technologies currently. Both of systems, however, have critical issue to make use in continuous medical diagnosis since the first system using camera is constrained by narrow space where multiple cameras are shooting as well as the second technology using sensors are needed to be affixed tightly onto the body using elastic straps or sensor-built clothing [3]. Due to those reasons, those technologies are impractical in long-term tracking of user wearing daily wear who wants to get medical diagnosis to predict diseases (such as neurodegenerative diseases) at an early stage. A state-of-the-art measurement system is to use sensors embedded into items of clothing. This project will investigate the dataset made through those sensors using machine learning, as well as the role that loose-fitting clothing has on the measurable information obtained, with aim of creating understanding the optimal location of sensors for human motion recognition.

1.2 Aims and objectives

The purpose of this research is to find the optimum sensor locations of cloth for tracking human motion. The first condition to be the optimum sensor location is that the sensor locations should have the ability to classify most accurately human motion. For the second condition, the position of the sensor should be least affected by the flutter of the loose-fitting clothes with any slackness so that, at which location, the sensor can predict human motion accurately like it is on tight clothes. If a location satisfies the above two conditions, continuous analysis of the user's movements in daily life will be feasible. The objectives of this study are as follows:

- Rearranging large volume of raw data and reducing the dimensionality of 3D coordinates into 1D coordinates to be analyzed as frame domain signal or discrete random distribution.

- Finding the optimum parts of the body for placing sensors by computing accuracy for each part using a support vector machine (SVM)

- Computing mutual information between sensors in various conditions

- A comprehensive analysis based on the data produced above and deriving optimal sensor placement

1.3 Contributions

In this report, the theory of mutual information (MI) plays the most important role to find the optimal sensor placement. Assuming that one sensor is used, the mutual information is computed in different motions and in different fabrics. The optimal sensor placement is decided by the sensor having bigger mutual information of different fabric and lower mutual information of different motions at the same time. Support vector machine is used to compute accuracy for each part of the body after 3D coordinates data for a motion of a model is dimensionally reduced using principal component analysis (PCA). In order to achieve the objectives, each section suggested in previous section proceeded as follows:

- Frame-centric dataset of a motion-captured with 3D coordinates is rearranged to the dataset based on the sensor using Python, and 3D coordinates of motion tracking are projected into first principal component using PCA (i.e. reducing dimensions of dataset)

- The accuracy of each part of the body is computed using support vector machine (SVM) after feature selection using sequential feature selection (SFS) to analyse which part of sensors on cloth make the accurate motion prediction in 11 areas of the cloth.

- Mutual information is calculated in two different situations, different motions or different size of cloth using Python, and those are visualized using MATLAB

- Through data of the above analysis, a comprehensive analysis was performed to find the optimum sensor location

1.4 Brief description for report structure

Section 2 presents the summarised introduction for background theories used to complete this study
Section 3 introduction of related works where the optimum sensor placement found in previous researches will be introduced and the methods to derive them will be presented
Section 4 will show the procedures to find the optimum sensor placement and is largely divided into two parts. One is to find the optimum area on a cloth to put sensors using a support vector machine (SVM), and the other is to find a more detailed sensor location using mutual information (IM). The former is to find a larger area to plant the sensor, the latter is to find a detailed location for the size to fit a sensor.
Section 5&6 provides a comprehensive result and conclusion with its limitation

2 BACKGROUND

2.1 General background of human analysis

The history of human motion capture dates back to the mid of 19C by Eadweard Muybridge's galloping horse. It is a tool to show the locomotion of a horse using 20,000 photos, as well as his invention zoopraxiscope. After him, in France 1880s, Etienne-Jules Marey analysed the motion of human and animal's using film by creating chronophotographic gun and fixed plate camera. In 1915, Max Fleischer invented rotoscoping that allows animators to track cartoon characters over photographed frames of live performances. In the first half of the 20th century, electric stroboscope is invented by Harold Edgerton which allows precise observation of motion. Through those series of developments, motion capture systems (MOCAP) technologies are the most important of modern times. Table 1 shows the tabulated classification system of MOCAP [4]. What the table shows is how to collect data for motion capture, of which the methods are divided according to whether or not to use a marker, use a camera, or according to received input by sensor. In the following ways, a model's movement can be captured and visualised on three-dimensional coordinates using computer. Among those, sensor-based motion that will be addressed in this research uses accelerometer prevalently. One major problem with sensor-based motion capture, however, is that there are inevitable variations while attaching the sensor to the body or cloth. Especially if it is a sensor on loosing clothing for medical purposes that is needed to be tracked consistently.

Table 1 Classification of motion captures technologies[4]

MOCAP technologies	
Active	Passive
Electromechanical	Optical: retroreflective markers
Optical fibre	Acoustic
Optical: strobing LEDs	Optical markerless (video-based)
Acoustic	
Inertial	
Optical markerless based on structured light	
Optical markerless based on video	

2.2 Mutual information

Mutual information can be used to study the statistical dependence, and which was invented by Shannon (1948). Since then, mutual information has gone through many substantial developments and applications. To examine the role that sensors placed on loose clothing has on classifier accuracy, the mutual information of each sensor will be investigated. This method is suitable because the mutual information proves powerful extensions of classical coefficients of correlation [5] measuring amount of information that data contains about another data of a sensor that is made in different motions or various styles of clothing. In addition, mutual information has been utilized to compute the dependence between signals of a stationary time series [5]. It has the important characteristic that 0 mutual information always implies statistical independence, unlike the correlation coefficient, which can be zero for highly dependent non-Gaussian data. [6]. The measure is thus inherently more powerful than the linearly orthogonal error criterion of least squares that it can replace in nonlinear and non-Gaussian applications such as independent component analysis (ICA) [6].

One of major this research related application using mutual information is the electroencephalogram (EEF) or electroencephalography (EEG) for signal processing that generated from changes in electric potentials consisting of different brain waves [7]. Many pieces of existing researches about EEF used the amount of mutual information between waves. For those researches, mutual information used to measure the brain interact within each other to perform a given task, as well as to select the more representative features from EEGs (i.e. to find out features that have a high level of mutual information to minimize redundancy of features). That means, by measuring the amount of mutual information, how similar the two random variables (frame-domain signal in this research) can be shown. Apart from feature selection, mutual information is also used in signal processing to measure the functional connectivity between the frontal and parietal lobes in the brain by analyzing the waves [8]. As such, measuring the mutual information of a signal (wave) can measure the similarity between two signals and there are many applications already being used in many places.

Consider two random variables X and Y with a joint mass probability p (x, y) and marginal probability mass function p(x) and p(y). The mutual information I (X; Y) is the relative entropy between the joint distribution and the product distribution p(x)p(y) [5]:

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log(\frac{p(x,y)}{p(x),p(y)}) \tag{2.1}$$

$$= \mathrm{D}(\mathrm{p}(\mathrm{x},\mathrm{y})||\mathrm{p}(\mathrm{x})\mathrm{p}(\mathrm{y})) \tag{2.2}$$

$$= E_{p(x,y)} log \frac{p(X,Y)}{p(X)p(Y)} \tag{2.3}$$

The equation (2.1) shows above is the traditional formulation of the mutual information. Each event or object specified by (x, y) is weighted by the corresponding probability p (x, y). this assumes that all object or events are equivalent apart from their probability of occurrence [9].

2.3 Technical methods

2.3.1 Principal component analysis (PCA)

The principal component analysis (PCA) is a technique to reduce the dimensionality of data set (sample) composed of many interrelated variables while retaining the sample's information as much as possible [10]. The PCA analysis is performed on one of the square symmetric matrices which can be pure sums of squares and cross products (SSCP), Covariance matrix, scaled sums of squares and cross products, or correlation matrix, sums of squares and cross products from standardized data [10]. The results after analyzing using SSCP and Covariance will not be difference because those matrix types are only different in a global scaling factor as well as the results will be different in correlation matrix. This is because the correlation matrix is used if the variances of each variate are different, or the scale of measurement of each variate differs. In this project, the covariance matrix is used to get reduced dimensionality of data set.

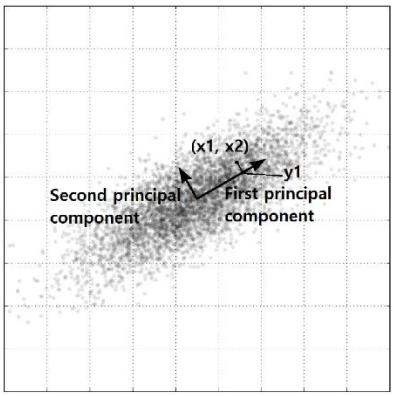


Figure 1 PCA of multivariate Gaussian distribution in the two-dimensional plane

Figure 1 above shows bi-variate scatter plotting with the first and second principal component. The point (x1, x2) is projected onto the first principal component which coincides with directions of maximum variation of the original observations to make projected point y1. All the points of X (X is set of all points in the figure) are projected onto the principal component to reduce the dimensionality of 2D original data set. The first principal component has the property that the variance of the projected points Y (Y is set of all projected points onto the axis) is greater than the variance of the points projected any other axis that passes through the centre of the elliptical of points. Likewise, the second principal component indicates the second biggest variance of any other possible axis or line.

Step 1: Zero-mean data

This step is to transform all variables on the same scale not to lead biased results. This step helps to reduce the difference of variance between initial variables and the variables of points projected onto principal component. If PCA is done without this step, points with a variance, ranges between 0 to 100, will dominate over a variance, ranges between 0 and 1. Normalization to corresponding scales will help to avoid this problem. This can be done by subtracting mean of variables from each variable.

Step 2: Covariance matrix

The covariance matrix is a symmetric matrix which is used to understand the difference between two variables. It means if there are variables highly correlated, the redundant information is contained. In this project, 3D coordinates will be projected onto the first principal

component. So, one vector will be composed of 3 variables which are x, y and z axis. 3×3 covariance matrix for a vector will be:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix} \quad (2.4)$$

The covariance matrix for N number of vectors will be the sum of squared divided by the number of the vectors.

$$C = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)(x_i - \mu)^T \quad (2.5)$$

Step 3: Feature vector

At this stage, eigenvalue and eigenvector are computed using the covariance matrix calculated in the previous step by simple equation. The x and λ indicate eigenvector and eigenvalue of covariance matrix respectively.

$$(C - \lambda I) = x \quad (2.6)$$

Each column of the eigenvector is principal component and the first principal component is computed using the biggest eigenvalue as well as the first principal component maximizes the possible variance. The second principal component is also decided by the second biggest eigenvalue and the rest principal components are taken in the same way (i.e. each column of eigenvector has eigenvalue as well as the larger the value eigenvalue is, the bigger the variance eigenvector has).

Step 4: Projection

To project vectors onto principle component, zero-mean vectors are used. The desired principle component is chosen, and projected vector will be simply computed by multiplication between zero-mean data and principle component. Needless to say, first principle component has to be used.

Projected data set = $Feature\ vector\ of\ first\ principle\ component^T \times Zero - mean\ data^T$

2.3.2 Correlation

The Pearson's coefficient r, a.k.a. the Pearson Product-Moment Correlation is a measure of association between two variables. The Pearson's r coefficient is defined as the percentage of the covariance of two vectors composed of numerical data after normalized using the square root of those variances. The coefficient is determined in the range between – 1 and 1. The coefficient value 1 indicates two variables are perfectly linear relationship as well as the value of -1 shows two variables are negatively related. The computation of the coefficient can be represented mathematically [11].

$$r = \frac{C_{xy}}{\sqrt{C_{xx}C_{yy}}} = \frac{C_{xy}}{\sigma_x \sigma_y} \quad (2.7)$$

The cross-correlation is a method to measure interrelation of series as a function of the displacement of one signal in relation to the other, and generally used to search similar pattern inside a long signal for a shorter signal. In the simplest example, there are signals labelled $x_1[n], x_2[n]$, n = frame number between identical time gap (i.e. number of samples). Mathematically, the cross-correlation can be computed using the equation [12].

$$r_{12} = \frac{1}{N}\sum_{n=0}^{N-1} x_1[n]x_2[n] \tag{2.8}$$

The correlation analysis is usually aimed to identify the relationship between variables, which can be between two variables (Pearson's correlation), one to multiple variables (Multicollinearity) and identical two variables (Autocorrelation function). The basic idea of the autocorrelation function is the cross-correlation function. Cross-correlation is a measure of similarity of two signals, or it can be numerical variables, as a function of a time-lag applied to one of them. In autocorrelation function, identical two variables are used (i.e. the variables used in previous section $x_1[n], x_2[n]$ become $x_1[n], x_1[n]$ on the same equation) based on cross-correlation method. Autocorrelation function generates correlation coefficient at every different time lagging. So, this can be utilized to assume the period (time-series) of signal which can be critical property of the signal.

2.3.3 Sequential Feature selection (SFS)

In general, there is a large amount of data for machine learning. In particular, high-dimensional features can produce errors (the curse of dimensionality) or lead to incorrect results. to solve these problems, feature selection can be used to reduce the dimensionality of features by selecting a subset of entire features. In general, SFS is characterized that there are two components. One is an objective function has to be set, called the criterion and the other is that candidate subset is used. The criterion is applied to seek to minimize over all feasible feature subset [13]. The most common ways to set the criteria are mean squared error and misclassification rate for regression models and classification models respectively. A candidate subset if used while the criterion is evaluated where the features are added or deleted by two methods, sequential forward selection (SFS) and sequential backward selection (SBS). The features are added into the candidate subset from empty subset during feature selection algorithm is running and adding is repeated until the criterion does not decrease for the sequential forward selection (SFS). But the method, sequential backward selection, starts with a full candidate subset and sequentially removes the items until removal of further features does not impact on decrease the criterion [13].

2.3.4 Support vector machine (SVM)

The goal of the SVM is to create a flat boundary called a hyperplane that divides the space and creates a homogeneous division on both clusters. In 2-dimensional space, the operation of the SVM algorithm is to find a straight line dividing the two classes (separating hyperplane) where one or more line of division can be selected between support vectors. In this case, the maximum margin hyperplane has to be computed and this is the ultimate goal of the SVM algorithm. This supervised learning will be used to classify two kinds of motions to compute accuracy for each body parts [14].

2.3.5 Properties of signal

- Mean

Mean value of the discrete signal can be computed in simple equation. Where f(t) is input signal, and T is a reciprocal number of fundamental frequencies

$$\text{Mean}\big(\text{f(t)}\big) = \frac{1}{T}\sum_{t-T}^{t} d(t)dt \tag{2.9}$$

- Variance

Variance is a distance between mean value and variable, which is like the amplitude of signal with offset 0.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{2.10}$$

- Estimated time series (periodic) of random discrete distribution by autocorrelation

The autocorrelation is to find out the correlation of between a variable and itself at different sequent phases. The autocorrelation can be defined using the values or r_{xx}. This can be computed using formula.

$$r_{xx}[k] = \sum_{n=-\infty}^{\infty} x[n-k]x^*[n] \tag{2.11}$$

Where n is the phase difference between a signal and itself. At each phase difference correlation is calculated to find the highest correlation point where becomes time series of random discrete signal.

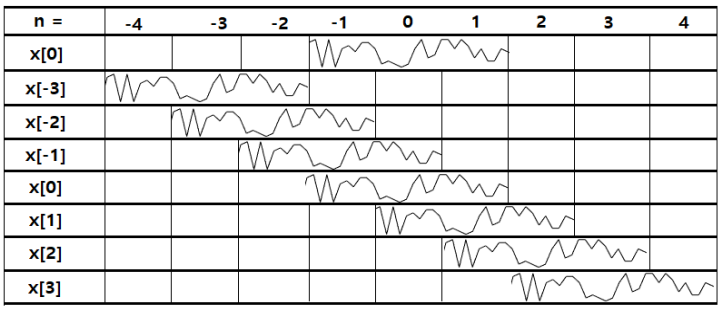


Figure 2 Procedure of autocorrelation from the time t = -∞ to t = ∞

- Skewness and Kurtosis

The measurements mentioned above that are mean, variance, max and min value are numerical measures of signal or distributions. However, the skewness and kurtosis can be used to estimate the shape of a distribution and give us more precise shape information [15]. Skewness shows how much and where to the distribution skews as well as the kurtosis indicates how tall and

acute the centre of the peak is [15]. The estimation of shape can be done based on normal distribution. The skewness and excess kurtosis of a normal distribution are 0. Based on a normal distribution, the values of skewness and excess kurtosis tells how different from normal distribution in terms of shape (i.e. if values are close to 0, the shape of measured distribution is close to a normal distribution). Both of those can be computed with simple equation [15].

$$\text{Skewness} = \frac{\sum_i^N (X_i - \bar{X})^3}{(N-1) \times \sigma^3} \tag{2.12}$$

$$\text{Excess Kurtosis} = \text{n}\frac{\sum_{i=1}^N (X_i - X_{avg})^4}{(\sum_{i=1}^N (X_i - X_{avg})^2)^2} - 3 \tag{2.13}$$

- Fast Fourier transform (FFT) algorithm and main frequency

The fast Fourier transform (FFT) algorithm is invented to compute the discrete Fourier transform (DFT), and it is one of the most important algorithms in the numerical analysis [16]. The code to execute FFT can be easily found on the internet and is used to process signals such as video and voice by analyzing frequency component using Fourier series or Fourier transform (FT). the frequency components are decomposed from the original signal by using FT which can be called the main frequency in relation to its magnitude (Amplitude). By recovering the function from those components, initial function can be expressed as a sum of periodic components for which Fourier analysis is a basic method. The time-domain signal is converted into frequency domain signal after FT. In this study, the frame domain sensor values will be transformed to discretized frequency domain with amplitude. Every signal has its specific amplitude and frequency components. These values will be used to express the feature of signal of a sensor for supervised learning using support vector machine (SVM).

- Entropy

Entropy is a term originally used in classical thermodynamics. Entropy is a function of temperature, which means the possibility of a given heat being converted into a corresponding work (W). As such, entropy generally means uncertainty. In information theory, the entropy can be applied to introduce information entropy, which expresses the potential for information to be obtained [5]. In short, it means the amount of meaningful information, not the amount of all information. This also can be used to measure the amount of information from the probability distribution. In this study, the signal will be assumed to be discrete random distribution and its entropy will be computed using Python. This can show the amount of information of the signal and will be used for one of signal features. The definition of entropy for a discrete random variable X is given by Pr ($X = x_k$) $=p_k$ for k = 1, 2…, then the entropy of X is defined as

$$\text{H(X)} = -\sum_{K \geq 1} p_k log p_k \tag{2.14}$$

3 RELATED WORK

3.1 Finding optimal sensor placement of a specific combination of body parts using a neural network (NN)

Sensor placement on the human body has previously been examined in [17], with accelerometers placed on the human body. In this, it was found that sensors placed on the right hand and right thigh gave the best accuracy when predicting the type of motion being before. These positions were optimal because the combination of accelerometers on the right hand and thigh have the lowest errors when a neural network built with those sensors. The most important part of this paper was the methodology that paper was not about finding the best sensor to find the best sensor but about identifying the least error sensor to remove. And the next thing noticed was that that paper limited the research subjects to a combination of approximate sensor positions without finding the location of the sensor very specific. As in this study, my research divided the sensor position of the model into 11 parts to see how the approximate sensor placements should be. The procedures for the experiment are outlined below. Among them, the most important thing to notice is which parts of the body were selected to build neural network (NN)

For the data collection in that paper, three accelerometer sensors are used to be placed on the human body (i.e. inertial measurement unit). A sensor consists of 3 axes accelerometer to measure acceleration at 3 directions x, y and z. The sensors are placed on the three-part of the body which are on the right hand (acc1), the right thigh (acc2) and the chest (acc3). The data made by the sensors are transmitted toward a PC using a serial port or to an SD card to be stored, after which data acquisition is implemented with MPIDE software platform (Multi-Platform Integrated Development Environment). The collected data then is imported into MATLAB for processing and analysis on which the recognition system is built. The recognition system is composed of the procedures that are designing, training and simulating using artificial neuronal network (ANN). 17 postures are used for labels to be recognized by ANN architecture with 20 neurons in the hidden layer such as standing, sitting, walking and so on. For the feature, the fast Fourier transform (FFT) algorithms are used to compute the discrete Fourier transform (DFT) made by the sensors and the recognition rate is used to find out the best position of sensor with the sensor combinations that are [(acc1), (acc2), (acc3), (acc1+acc2), (acc1+acc3), (acc2+acc3) and (acc1+acc2+acc3)]. For the results, the sensor on the right thigh and right hand (acc1+acc2) gave the best recognition rate in FFT analysis.

3.2 Feature extraction of a signal from an accelerometer before feature selection, and optimal sensor placement of existing researches

Sensor placement for activity detection was carried out in [18], with list of features extracted from accelerometer signal and introduces of feature selection. The features of raw accelerometer signals to maximize the precision of classification using KNN (K-nearest neighbour) in terms of sensor location on the body which are on ear, chest, wrist, knee, ankle, arm and waist. The below indicates the features of signal from the accelerometer, as well as it shows which features can be extracted from the signal.

Table 2 List of features extracted from raw accelerometer signals [18]

Feature number	Description
1	Averaged variance over 3 axes
2	RMS of signal derivative
3	Mean of signal derivative
4	Average entropy over 3 axes
5	Average cross-correlation between each 2 axes
6	The average range over 3 axes
7	The average main frequency of the FFT over 3 axes
8	Total signal energy averaged over 3 axes
9	The energy of 0.2 HZ window around the main frequency over total FFT energy (3 axis average)
10	Averaged skewness over 3 axes
11	Averaged kurtosis over 3 axes
12	Averaged range of cross-covariance between each 2 axes
13	Averaged mean of cross-covariance between each 2 axes

To investigate the importance of each type of feature above, three methods of feature selection were studied that are RELIEF feature selection, Simba feature selection and mRMR (minimum redundancy Maximum Relevance). The results of feature selecting shows similar results in the three methods such as, in terms of chest, the three features that are Average main frequency of the FFT over 3axes, energy of 0.2Hz window around the main frequency over total FFT energy (3 axis average) and average range of cross-covariance between each 2 axes are chosen for classification since these three features are selected as majority after feature selection. In this paper, sequential feature selection (SFS) will be used and be expected to be comparted with chosen features in that research.

The precision and recall are calculated to find optimal sensor location using KNN classifier with $K = 1$, 5 and 7. The results show waist and chest are the best sensor placement for transitional activity as well as waist, chest and wrist are the best sensor placement for medium-level activity with the highest precision calculated using Bayesian classifier with Gaussian priors. Although KNN classifier played a good role here because it used various activities. In my research, support vector machine (SVM), however, will be used to be trained with features since there are only two motions (SVM is the best classifier for binary label). And this journal put together the optimal sensor from other articles clearly in Table 3 with which the result of this paper can be conveniently compared with a well-organized result below.

Table 3 Optimal sensor placement of existing research [18]

Reference	Position
L. Atallah et al.	waist, chest and wrist
Bao et al.	Hip, wrist, ankle, arm and thigh
Mathie et al.	Waist
Karantonis et al.	Waist
Yang et al.	Wrist
Lo et al.	Ear-worn
Hester et al.	ankle

3.3 Human motion prediction using machine support vector machine (SVM) and signal processing

The closest paper of this study has already been examined in [19], in terms of sharing three keywords, Human motion prediction, machine learning, and signal. The paper suggested human motion prediction using machine learning, support vector machine (SVM) and neural networks (NN). The machine learning algorithms classify the signal features which are mean and root mean squared (RMS) of acceleration signal of 3 axes (X, Y and Z) from the smartphone using built-in accelerometer application as well as covariance of 3-dimensional signal. These features are shown very simple to compute in comparison to the research in [18], and those will be also be included in my experiment in feature extraction before selecting features using SFS (As to foretell the results, RMS and mean value are less relevant compared to the features introduced in [18]). Before that, data cleaning is done using high pass filter which can be used to amplify the signal that is higher than specified cut-off frequency and minimize the signal that is lower than the cut-off frequency. Human motions used for labels to be classified are 6 motions which are standing, sitting, lying, and walking on flat ground, downstairs and upstairs.

3.4 Human gesture recognition using IR-UWB radar sensor, and reducing the dimensionality of the signal data

It has been already examined in [20] that a human gesture recognition algorithm using Impulse radio ultra-wideband (IR-UWB) radar for data collection as an alternative of motion recognition using video technology since its drawback that is unavailable under poor environmental conditions such as fog. The streams of baseband pulses made by IR-UWB are extracted into useful features using principal component analysis (PCA). In this study, 5 hand gestures are used to set the labels, and classifiers are trained using 81 samples for each hand gestures. For classification of the gestures, neural network (NN) is used. The NN with backpropagation has one hidden layer with nine units and the output layer has 6 units. After training the NN, there are 500 trials to test and compute the accuracy which shows 100% for gesture recognition using backpropagate neural network for each gesture. This study shows neural network can classify amongst very similar hand gestures and the PCA does not exclude core information for hand gestures for gesture recognition. Using the advantage that core information is included after PCA, Attempts will be made to reduce the dimensions of the dataset in this paper.

4 APPROACH

This section can be divided into two sections. The first is to use machine learning to examine which sensors in the body parts show the good accuracy after training those sensors in support vector machine (SVM) for binary classification in section 4.2. Here, the sensors are clustered into 11 body parts, and the sensors for each body parts will be trained to make SVM. To build SVM, 15 features are extracted from a signal reduced to first dimension on frame domain using principal component analysis (PCA). The accuracies calculated for each body parts using SVM will be tabulated at the last of this section. In section 4.3, it will be shown that how the amount of mutual information was to describe how similar or different signals sensors produce between different situations.

4.1 Data structure

The data given for this analysis are three-dimensional coordinates for human wearing clothes with sensors that leave a trail of a model. The number of sensors is various according to the size of cloth the model is wearing from 8472 to 15041. The three kinds of clothes are used based on size to analyze the optimal sensor placement for loose-fitting jean and T-shirts (In this study, two kinds of clothes will be used which are tight-fitting (F1) and loose-fitting (F3), as well as the model moves in 2 different motions of walking, normal speed view (M1) and slow speed view (M2).

A dataset for an individual motion is given by a matrix $\mathbf{D}$ in $\mathbb{R}^{I\times S\times F}$, where I is the number of input measurements, S is the number of sensors, and F is the number of frames with 2 motion, M1 & M2, and 2 fabrics, F1 & F3 (F2 is not used in this study) such as M1F3, $I = 3$ (3D: x–axis position, y-axis position, and z-axis position), $S = 8472$, and $F = 554$ means a model is wearing loose cloth with walking in normal speed view, and the form of data are x-axis of 8472 sensors with 554 frame.

The data is initially given by the shape of $\mathbf{D} \in \{M1F1, M1F3, M2F1\}$ in $\mathbb{R}^{I\times S\times F}$ where $I = 3$, $S = 8472$ and 15041 for F1 and F3 respectively, and $F = 554$ and 932 for M1 and M2 respectively. The initial dataset is converted by Python into an easy-to-analysis from for motion prediction such as $\mathbf{D} \in \{M1F1, M1F3, M2F1\}$ in $\mathbb{R}^{I\times F\times S}$ where $I = 3$, $F = 554$, and $S = 8472$ to be analyzed based on the sensor with frame domain where the number of frame for M2 and the number of sensor for F3 is reduced for the convenience of analysis by leaving a sensor that matches the

small size of clothing. After that, PCA is used to project the 3D coordinates onto the first principal components (i.e. I become 1 from 3). In the end, $\mathbf{D} \in$ {M1F1, M1F3, M2F1} in $\mathbb{R}^{I\times F\times S,}$ $I = 1$, $F = 554$, and $S = 8472$ are used for supervised learning and mutual information calculation between those. To compute mutual information between tight and loose-fitting, $\mathbf{D}\in$ {M1F1, M1F3} in $\mathbb{R}^{I\times F\times S,}$ $I = 1$, $F = 554$, and $S = 8472$ is used and mutual information is computed between sensors of frame domain 1D values. For example, the 1D coordinates (projected onto first principal component using PCA) of 554 frames for 8472 sensors.

4.2 Finding optimal body parts to put a sensor using support vector machine (SVM), after feature extraction and selection

The 8472 number of sensors are attached on the cloth to collect data made by the user's movements as the form of 3D coordinates of trajectory for 2 motions in this study. To predict a motion between those, a sufficient quantity of samples shall be obtained such as, 600 samples were taken for each label to train the NN for the research in section 3.1 and 256 samples were taken to train hand gesture into SVM in section 3.4. Namely, to train machine learning for motion prediction, each motion needs enough number of samples. The data given, unfortunately, to analyze the motion tracking in this study is limited as 1 sample per motion. Instead, there is a huge number of sensors are densely located on the cloth for each motion as well as the information between adjacent sensors is shared a lot (i.e. close sensors produces considerably similar sensor traces) as can be seen in figure 3, which shows similarity of signal between adjacent sensors. And, it shows the data produced by PCA and randomly chosen, of 3 sensors for each body parts (right calf, left thigh, abs, left forearm and right arm. The overlapped sensors are removed to erase duplicates such as if sensor A and sensor B are on same location when sensor A and B are on shirt and trouser respectively, the sensor B is omitted. The sensors are clustered by 11 body parts which are abs, chest, back and arm, forearm, thigh and calf for right and left. Adjoining sensors will be clustered to be considered as samples as can be seen in figure 4. As for example, there is N number of sensors are affixed at left forearm in figure 4-(a) forearm can be considered as samples for data of a sensor for right forearm in figure 4-(b). Each set of sensors (11 sets for each body parts) are used to compute accuracy using support vector machine (SVM) as well as 11 support vector machines are made and accuracy of each set of sensors are compared to find the highest accuracy part of body. From one set of sensor data, information is computed (i.e. 15 features are computed using Python). The 15 properties of signal are chosen for features of a sensor (maximum value, minimum value, skewness, kurtosis, 3 major frequencies of Fourier transform (FT), 3 amplitude of three major frequencies of Fourier transform, autocorrelation, entropy, root mean square (RMS), mean value, variance. Before training for supervised learning, the initial 15 features are computed to be selected to form the best subset of features using sequential feature selection in order to remove features which are of little influence for training supervised learning (i.e. the most informative feature at each timestep is chosen). For feature selection, 10-fold cross-validation of training set is used so that the feature selection algorithm uses 1-fold as test set and the rest as validating set to produce the most accurate feature. The set of sensors of body parts are divided into 70% of test and 30% training set to put discriminative power between test in body parts otherwise all the body parts will show 100% accuracy since there are a lot of training set and only binary labels. Figure 5 shows visualization of a series of processes for this section where (a), (b), (c), and (d) corresponds to the section from 4.2.1 to 4.2.4.

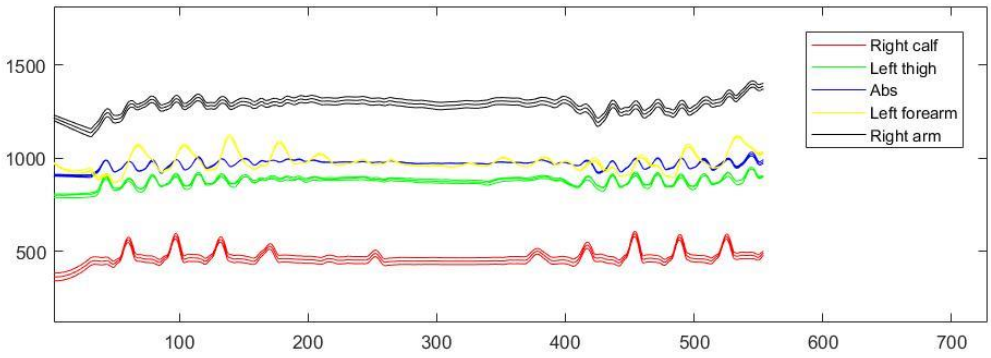


Figure 3 Similarity of the signal pattern of three sensors sampled at each part of the body

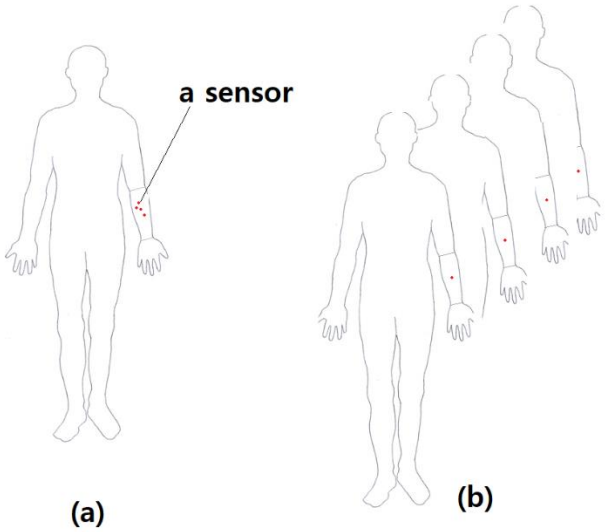


Figure 4 consideration of densely concentrated sensors as samples

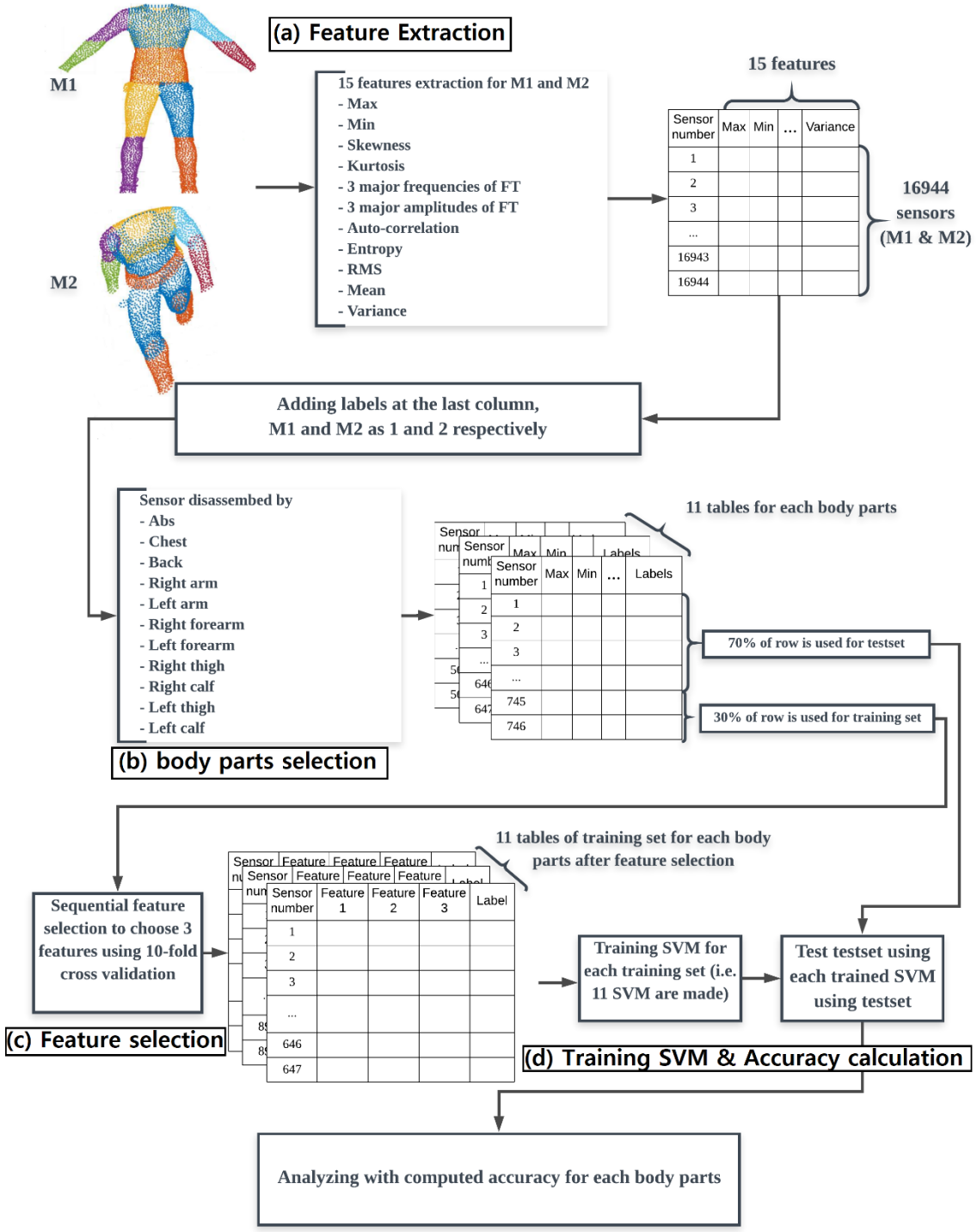


Figure 5 Procedure for computing accuracy for each body part using SVM through feature extraction and selection

4.2.1 Feature Extraction of a sensor, and principal component analysis (PCA)

Feature extraction starts from each sensor dataset including traces of sensor that are projected onto the first principal component using PCA following motion movements. This work is done for both sensor data from M1F1 and M2F1. Tally the 15 features are extracted from 1D sensor dataset with 16944 number of instances including M1F1 and M2F1. The features extracted are described in table 4. The window used for feature extraction is the same as the range of the sensor data given. The fast Fourier transform (FFT) is used to figure out what frequencies of signal are assembled to form the initial one-dimensional discrete signal of a sensor with inherent amplitude for each frequency. The Fourier analysis is basically a method to describe a function as a sum of periodic components, and for recovering the function from those components (here the function is initial 1D signal of a sensor, and component is decomposed signals from initial signal by unique frequency domain). Three amplitudes from the largest to the third are used as features as well as those' corresponding three frequencies are used. The maximum and minimum value of 1D sensor coordinates list are directly computed using max() and min() function using Python Skewness and Kurtosis is figured using kurtosis() and skewness() respectively. Kurtosis shows the sharpness of the maximum value of a discrete signal along with the measure of the asymmetry of the signal by skewness. Those four features indicate the characteristic of signal form and show how the signal looks like. Autocorrelation is computed by the function np.correlate() using Python. The function np.correlate() returns a series of crosscorrelation value between signal and itself in different phases from negative infinity to positive infinity. Amongst the series of correlation value, the biggest value is chosen to be used for a feature of the signal at which correlation between a signal and itself is the highest. The entropy of the sensor signal is calculated after the discrete values are sorted into the 10 bins. The range of the bins has the same range. The range is calculated by {Max(sensor)-Min(sensor)}/10 where sensor is the list of discrete values of a sensor. This shows the degree of disorder, or it can show the amount of information of the signal. As statistical figures, root mean squared value (RMS), mean and variance are computed with ease using built-in function, RMS(), MEAN() and variance() respectively as features. Likewise, the above mentioned 15 characteristics of a signal were used as features before feature extraction.

Table 4 Numbering 15 Features chosen of the signal made by a sensor

Feature number	Description
1	Maximum value
2	Minimum value
3	Skewness
4	Kurtosis
5	Amplitude of the first biggest frequency of fast Fourier transform (FFT)
6	Amplitude of the second biggest frequency of FFT
7	Amplitude of the third biggest frequency of FFT
8	The first major frequency of FFT
9	The second major frequency of FFT
10	The third major frequency of FFT
11	Autocorrelation
12	Entropy
13	Root mean squared (RMS) value
14	Mean value over the signal
15	Variance

4.2.2 Preparing features before training

The 8472 number of sensors are divided into 11 subsets according to body parts as can be seen in Table 5.

Table 5 How body parts divided for supervised learning and the number of sensors divided by each part of the body

Body parts with the number of sensors	
1. Abs (561)	7. Left forearm (283)
2. Chest (647)	8. Right thigh (853)
3. Back (1205)	9. Right calf (724)
4. Right arm (329)	10. Left thigh (891)
5. Right forearm (305)	11. Left calf (746)
6. Left arm (339)	

In the table 5 sensor locations are listed with how many sensors are in parentheses. As can be seen in the table, the sum of sensors in body parts is not the total number of sensors 8742. This is because there are many sensors duplicated affixed on same location of the body such as the model wears its shirt outside trouser around waist with belt. Around waist, the sensors from shirt, trouser and belt are overlapped. Those sensors are removed.

The subsets of sensors (i.e. body parts) are chosen based on joints. The joints under the cloth, shirt and trouser, are composed of two kinds that are ball and socket joint and hinge joint apart from the spine. Both shoulders and hip joints can be one of typical ball and socket joints. Those joints allow the greatest range of movement but, on the contrary, the movement of the arm is limited to the joint. The elbows and knees can be classified in hinge joint this only allows extensive flexion and extension with small amount of rotation. Figure 6 shows illustration of right leg with 6 locations from (1) to (6) where the sensors' trajectories are plotted on the graph (b) and (c). The trajectory of the two graphs (b) and (c) shows a clear difference in shape but the traces on each graph have a similar shape such as (1), (2) and (3) have similar traces. This is the reason why the sensors are grouped based on joints. The joints of two principles also can be considered as pendulum movement. A fixed point in the pendulum can be likened to a joint as can be seen in the figure 6-(a). the concatenated pendulums can revolve around a fixed point and this produces sine wave. This is the principal of signal decomposition using Fourier transform. For this reason, the 3 main frequencies and those amplitudes of fast (discrete) Fourier transform (FFT) will be used for feature extraction on the assumption that there are three large joints on the body which are spin, ball and socket joints (shoulders and hip joints) and hinge joints (knees and elbows).

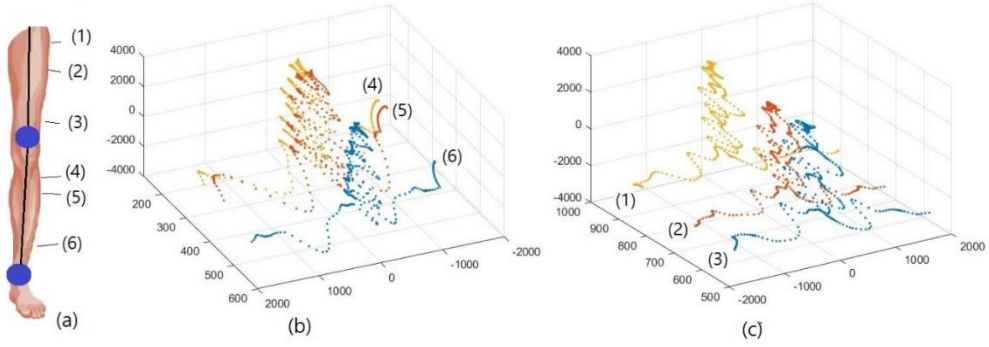


Figure 6 Sampling sensors from left leg to compare sensor traces between thigh and calf

4.2.3 Feature Selection

After extracting 15 features for every 16944 signals of sensor dataset of M1 and M2, the dataset is ready to be trained in the support vector machine (SVM) with 2 labels for motion 1 and 2. However, the number of instances is 16944 with 15-dimensional features can degrade performance of system and causes an unexpected result. One of main reason resulting bad consequence is the curse of dimensionality and overfitting. The performance of classifier increases until the number of features reach the optimal number of features but, after the classifier reaching the peak performance, it decreases rapidly. This can be tackled by using feature selection and much more accurate classification will be made. In regards overfitting, this can be minimized since the features will have a reduced variance with enhanced generalization by erasing irrelevant features or noisy. Additionally, the performance will be improved with smaller data size.

The sequential feature selection (SFS) method is used to find the desired number of features. Given a set of features Y of |Y|, the SFS finds the desired number of subsets with size d in X_d, X_d is the set of all possible subsets of Y. This method is exhaustive and evaluates each candidate subsets X using criterion function J(X), X is a possible subset of the desired number of subsets X_d. If the subset of X's evaluation is higher than the value of J, the X is considered as good a subset. The optimal subset of $\widetilde{X}_d$ can be found using the formula. To evaluate each possible subset, 10-fold cross-validation is used at each stage of evaluating criterion with J.

$$J(\tilde{X}_d) = \max_{X \in X_d} J(X) \tag{4.1}$$

The desired number of features is set as 3, $d = 3$. SFS is performed for 11 body parts separated after the table made after feature extraction (i.e. there are 11 tables with 15 features for each table of body parts).

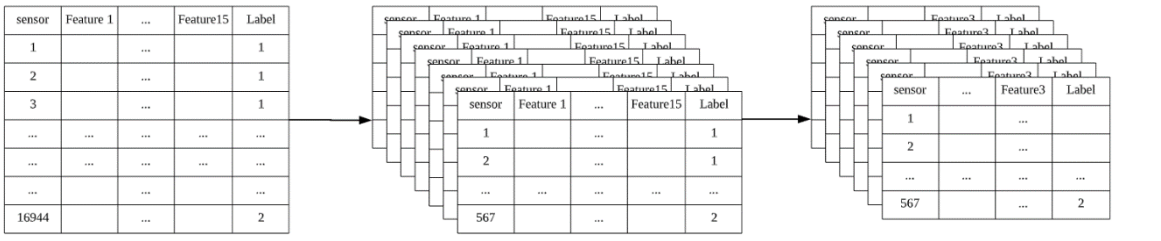


Figure 7 Visualization to show how initial dataset divided into subsets according to body parts and the shape of the matrix after feature selection

The feature selection is performed using MATLAB with the function sequentialfs() with 10-fold cross-validation and $d = 3$. Figure 8 shows the graph between 15 features and scores. The y-axis 'scores' indicates how much the feature is good to be used for the classification of two motions. 15 score show the best feature between 15 features and 1 score indicates the worst feature to be used for SVM training. As can be seen in the graph, skewness and kurtosis seem to best characterize the signal between M1 and M2 as well as the first and second frequencies of fast Fourier transform (FFT) are the next best features for training. On the contrary, Autocorrelation, RMS and mean value shows the poor scores. The reason why reduction of discriminative power of mean and RMS between M1 and M2 is because of principal component

analysis (PCA). PCA maximize the variance as the dimension decreases but the RMS and mean value are distorted (i.e. mean RMS error after PCA filtering). Entropy can be used to represent an informative quantity. However, the entropy stays in the middle rank and falls behind in comparison to skewness and kurtosis while entropy still maintains good scores compared to the values of FFT. Overall, skewness, kurtosis and first frequency of FFT are the best features to discriminate the motion M1 and M2.

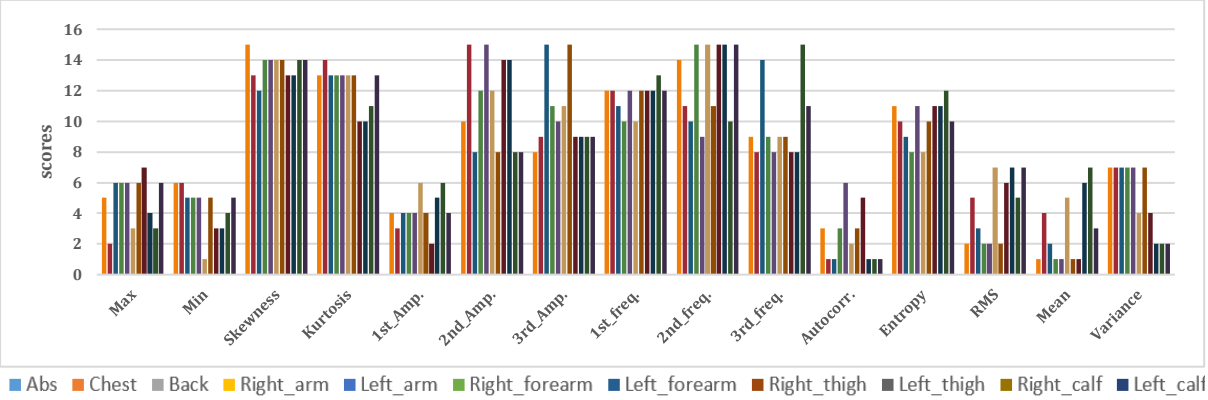


Figure 8 Feature scores by 11 body parts

The table below shows selected 3 features using sequential feature selection (SFS) for areas of the body. Those features will be used for training SVM (i.e. to build the 2D hyperplane for classifying the binary label (M1 and M2).

Table 6 The 3 most relevant feature by body area: Feature 1 indicates the most relevant feature among 3 features

Areas of the body	Feature 1, Feature 2, Feature 3
Abs	Skewness, 2nd frequency of FFT, Kurtosis
Chest	2nd amplitude of FFT, Kurtosis, Skewness
Back	3rd amplitude of FFT, 3rd frequency of FFT, Kurtosis
Right arm	2nd frequency of FFT, Skewness, Kurtosis
Left arm	2nd amplitude of FFT, Skewness, Kurtosis
Right forearm	2nd frequency of FFT, Skewness, Kurtosis
Left forearm	3rd amplitude of FFT, Skewness, Kurtosis
Right thigh	2nd frequency of FFT, 2nd amplitude of FFT, Skewness
Left thigh	2nd frequency of FFT, 2nd amplitude of FFT, Skewness
Right calf	3rd frequency of FFT, Skewness, 1st frequency of FFT
Left calf	2nd frequency of FFT, Skewness, Kurtosis

4.2.4 Training support vector machine (SVM)

For classification and regression, supervised learning is required and one of a significant machine is support vector machine (SVM) that follows theory of machine learning to maximize accuracy of prediction as well as avoiding over-fitting. The SVM as tool of classification and regression prediction uses hypothesis space of a linear function in high-dimensional feature space as a system which is trained by a learning algorithm.

11 SVM is made for each body parts to check the accuracy computed at each SVM. The SVM is trained using the most relevant 3 features calculated using feature selection are used. The fitcvsm() function in MATLAB is used to build SVM classifier using kernel functions for binary (M1 and M2) classification. In addition, it computes hyperparameters minimizing 5-fold cross-validation loss using automatic hyperparameter optimization. Figure 9 shows how fitcvsm() function finds out the best observed and estimated feasible point for abs SVM.

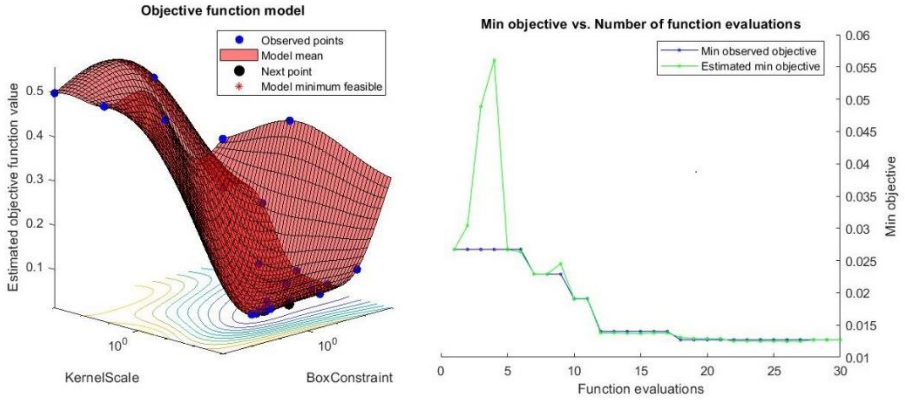


Figure 9 Procedure to find optimized hyperparameters using MATLAB

After training all the areas of the body, the SVMs are tested using a test set which is 70% sensors of single body parts.

Table 7 Accuracy for body parts tested using SVM with 30% and 70% of the training set and test set respectively

Parts of the body	Accuracy (%)
Abs	98.2143
Chest	100
Back	99.8617
Right arm	100
Left arm	100
Right forearm	99.5305
Left forearm	100
Right thigh	99.4137
Left thigh	99.6792
Right calf	99.4038
Left calf	99.9042

4.3 Analysis of sensor locations using mutual information of a model in various conditions, different motions and clothing

The datasets $\mathbf{D} \in \{M1F1, M2F1, M1F3\}$ in $\mathbb{R}^{I \times F \times S,}$ $I = 1$, $F = 554$, and $S = 8472$ are used to compute the mutual information between two individual sensors of M1F1, M2F1, and M1F3. Figure 10 shows as an example each entropy of M1F1 and M2F1, and it shows what the amount of mutual information about entropy means.

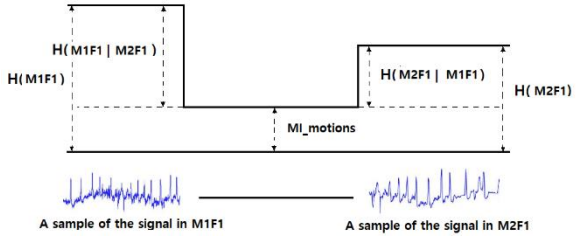


Figure 10 Description of mutual information between entropy H(M1F1) and H(M2F1)

Mutual information is computed using the package in Python called Sklearn which can be used for Mutual information between two discrete random variables. As a result of PCA calculation, the float values are calculated, and which are converted into integer values before mutual information is computed since the mutual information is based on probability distribution of variables. The decimal place of calculated data set after PCA was 13 decimal place which means individual sensors has 554 different values for 554 frames for both motions. For example, M1F1 and M2F1 having same probability distribution will result in same mutual information for 8472 sensors between M1F1 andM2F1. To prevent that problem, values are rounded at -1 decimal place, so that similar values can be considered as one value which can lead different probability distribution of M1F1, M2F1, and M1F3.

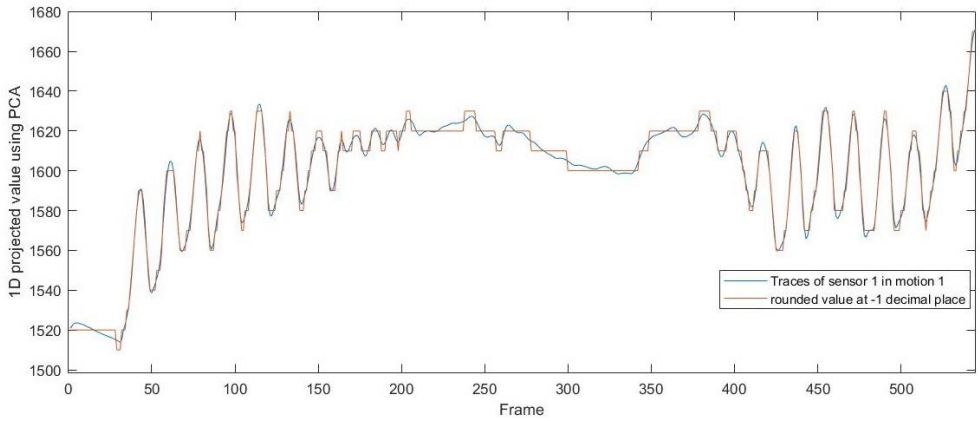


Figure 11 Trace of the randomly sampled sensor with its rounded value.

Figure 11 above plotted using MATLAB shows traces of first sensor of the first motion after dimensional reduction using PCA. Blue line indicates traces of original values over 554 frames, as well as the red line, indicates traces of marginal value after rounding at -1 decimal place.

With the above way, the MI is used to show how dependent or independent the data the sensors make in different movements or in different fluttering of clothes (here, the pair of signal made by a sensor for different motions are good if both have low MI, but for different slackness of cloth the sensor location is good if the signals have high MI).

5 RESULTS

In this section, the results obtained from the above approach are listed in 5.1. And the final four sensor positions will be shown in 5.2, by aggregating all the results.

5.1 Optimal sensor location for each condition using mutual information

This section will show as results how close or far the signal produced by a sensor between different situations. As different situations, the first different situation is to calculate mutual information of each sensor for a model moving in different motions in the same clothing (i.e. $MI_{motions}$ by M1F1 and M2F1). For the second condition, the mutual information in different styles of clothing for the same motion will be computed of each sensor (i.e. $MI_{fabrics}$ by M1F1 and M1F3). Where the former is to be addressed in section 5.1.1, the latter in section 5.1.2. In each section, the amount of mutual information for each sensor will be displayed as a colormap, as well as the optimal sensors in accordance with the condition of each section (i.e. The smallest amount of mutual information for different motions, and the biggest amount of mutual information for different size of clothing) will be displayed for each portion of the body.

At the end of this part in section 5.1.3, optimal sensor placement based on the value of the mutual information calculated above. The optimal sensor here is where the sensor shares the minimum amount of information between different motions and the maximum amount of information between the different sizes of clothing. Sensors meeting the above conditions were computed by subtraction mutual information between different clothing with mutual information between different motions as can be seen in Figure 12. A colourmap will be drawn using the calculated values ($MI_{difference}$) as follows in Figure 12, and the position of the sensor with the largest value will be displayed for each part of the body.

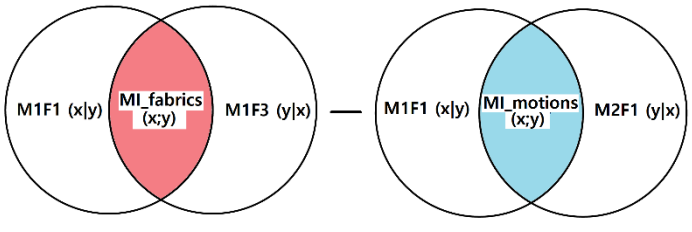


Figure 12 Calculating $MI_{difference}$ by subtracting $MI_{fabrics}$ with $MI_{motions}$ for each sensor

5.1.1 Result of sensor placement analysis using mutual information between different motions

After computing the mutual information values of a sensor between M1F1 and M2F1, sensors are plotted by colourmap. The dataset for mutual information among sensors of motions are $MI_{motions}$ which are computed using two datasets, $\mathbf{D} \in \{M1F1, M2F1\}$ in $\mathbb{R}^{I \times F \times S}$, where $I = 1$, $F = 554$, and $S = 8472$. The dataset $MI_{motions}$ is $\mathbf{D}$ in $\mathbb{R}^{S}$ where S is one-dimensional values of mutual information which is 8472.

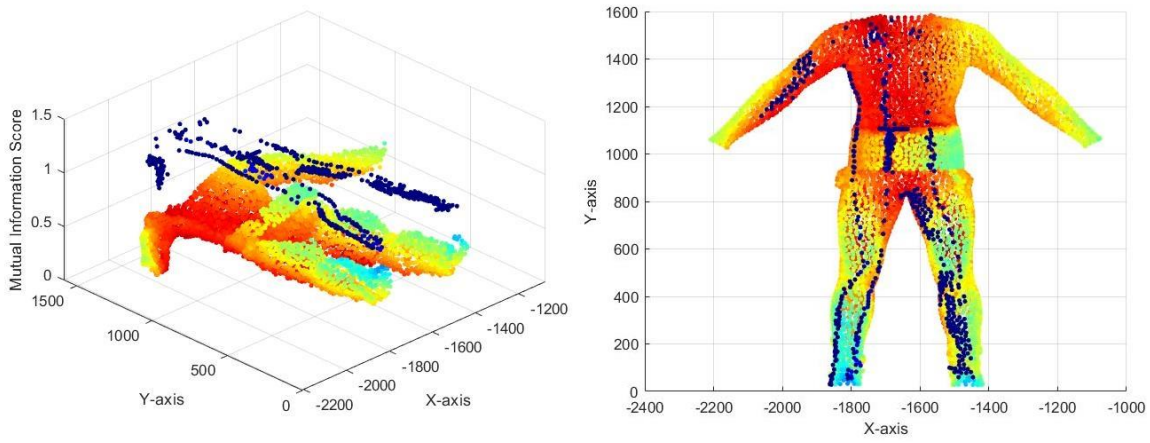


Figure 13 Colormap using the values of $MI_{motions}$ where the red area shows low $MI_{motions}$, and blue area show high $MI_{motions}$

Figure 13 shows colourmap of mutual information using MATLAB. The right side of figure 13 is side view of 3D space to show the value of mutual information ($MI_{motions}$) according to its colour, as well as the left is front view. Every point on the cloth indicates sensors, and there are 8472 sensors used. A red is used as reference colour for mutual information being 0, which means the deeper red, the closer to 0 the sensor is (i.e. the sensor with deeper red indicates the signal between M1F1 and M2F1 are more independent than other sensors). It seems to be shown that the more the sensor is placed in the centre of the right body, the more independent it shows to be between motions (Being independent, lower $MI_{motions}$, here means that the sensor in the area can be more distinguishable between two motions characteristics, as well as, with the sensor with high mutual information, the sensor barely distinguishes two motions in machine learning).

The previous section, mutual information is used to see overall mutual information of sensors affixed onto the cloth without concerning the parts of the body. However, this can miss an amount of information belonged in other parts of body because existing research shows the combination of sensors in right thigh and hand classifies motion prediction more accurately rather than only using sensor of right thigh or hand as can be seen in related work 3.1. the adjacent sensors, however, shared a lot of information proven in the previous section 4.3 showing each part shows almost 100% accuracy, by which body parts figure 14 shows the most independent sensors for each area of the body between M1F1 and M2F1. The figure is composed of 6 different views of model with optimal sensor placement for each body parts where (a)-frontal shot, (b)-rear shot, (c)-extremely low angle, (d)-left side shot, (e)-right side shot and (f)-right shoulder shot to appear sensors visually comfortable. In the figure 14-(a), one characteristic is seen that the optimal sensor for the abdominal region is located on the upper

left quadrant near the left rib as well as, close to the sensor, the sensor for chest is located. When it comes to the sensor on the back, the sensor that does not share the most information is located on the lower lumbar spine as can be seen in figure 14–(b). The sensor chosen for right and left forearm is on the inside of upper forearm which are in about the same position on both sides with respect to the centre of the body. In the figure 14-(c), the sensors for both right and left arm are observed clearly which are placed near the armpit of the inner arm but the sensor for the right arm is tilted toward the back, unlike the sensor for the left are is placed more near the chest. Regarding the sensors selected for legs, the sensor with the minimum mutual information for right thigh is positioned at the centre of crotch as described in the figure 14-(c) where the sensor at the centere is the sensor chosen for the right thigh. For the sensor on the left thigh, that is located near the groin on the innermost thigh that can be seen clearly in the figure 14-(e). for the body parts below the knee, both sensors are placed on the inside of both right and left calf near the knee, and these sensors are clearly visualized in figure 14-(d) and (e).

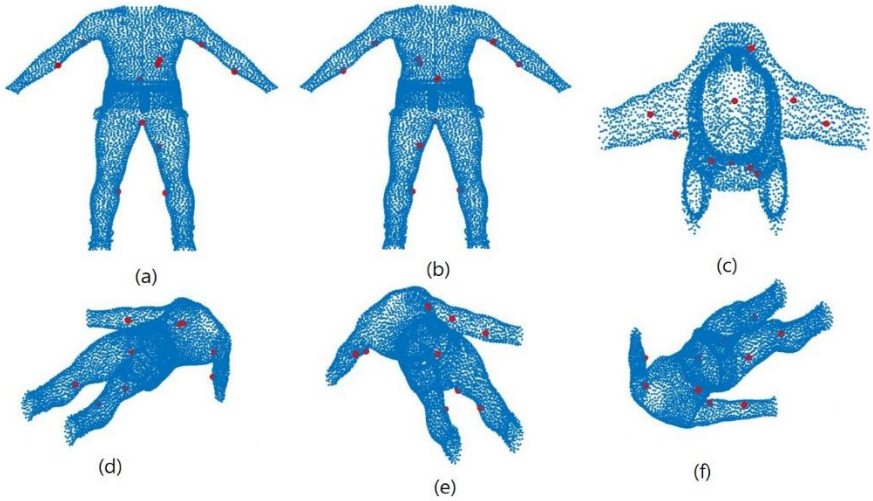


Figure 14 Sensors with lower $MI_{motions}$ for each body parts in the various angle of shooting

5.1.2 Result of sensor placement analysis using mutual information between different type of clothes

In order to maximize the sensor's usability, not only the placement of the sensor on the body but also the same effect should be achieved by attaching it to any slackness of fibre to track user movement continuously. As loose clothing is investigated, the daily life that is more realistic than when the tight clothing researched can be investigated since people do not wear tight clothes at home most of the time. The research in this section is focused to find out the location where the noise from the big clothes is minimized using mutual information between sensors affixed all over the body. The model used for expressing the loose-fit cloth is wearing a big dress as can be seen in figure 15. The analysis is performed on both data set M1F1 and M1F2 with $\mathbf{D} \in \{M1F1, M1F2\}$ in $\mathbb{R}^{I \times F \times S}$, $I = 1$, $F = 554$, and $S = 8472$.

Figure 15 loose-fitting used for M1F3

The mutual information in motion, but different styles of clothing, M1F1 and M1F3 with $\mathbf{D} \in \{M1F1, M1F3\}$ in $\mathbb{R}^{I\times F\times S}$, I = 1, F = 554, and S = 8472 is computed for every sensor trace with the same method in the previous section for calculating mutual information between M1F1 and M2F1. Figure 16 shows the mutual information between sensors in M1F1 and M1F3. In the following figure, the darker the red colour, the higher the amount of mutual information between the two sensors. The figure 16-(a) shows the layers of mutual information where almost mutual information is higher than 0.5 which means sensors on tight and loose fabric share a lot of common information in comparison to the mutual information between M1F1 and M2F1. The figure 16-(b) presents how the mutual information scores are distributed from the front. Many sensors display high levels of mutual information in the left half of the body which means the sensors on the left half of the body shares much more information between two clothes than the right half. The z-axis of figure 16-(a) is the dataset $MI_{fabrics}$ which is computed between datasets M1F1 and M1F3 with $\mathbf{D} \in \mathbb{R}^{I\times F\times S}$, I = 1, F = 554, and S = 8472.

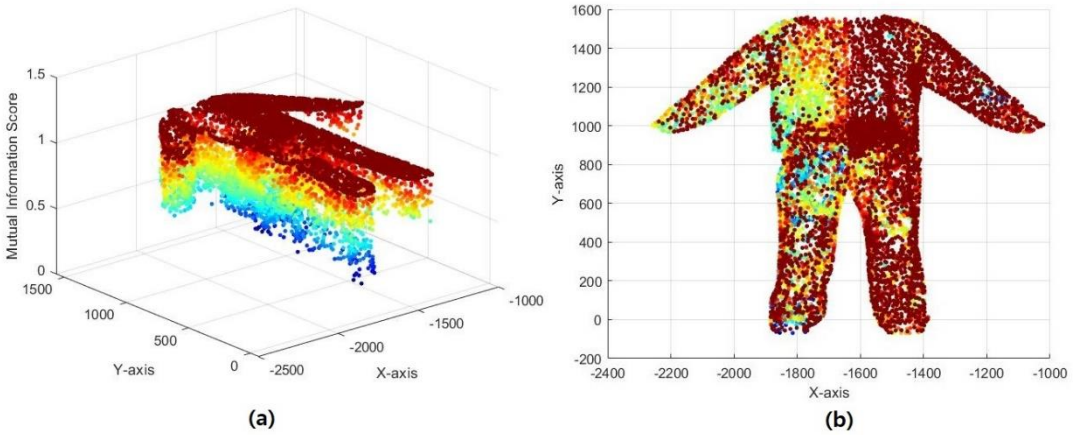


Figure 16 Colormap with $MI_{fabrics}$ for 8472 sensors where red-coloured sensor – higher MI and blue-coloured sensor – lower MI

Figure 17 shows the mutual information grouped by two clusters, M1F1 vs M2F1 and M1F1 vs M1F3. The red points are the mutual information from different clothes as well as the blue points are mutual information from different motions. As can be seen in figure 17, if motion is same, a lot of information is shared, but even if a model wore the same cloth, if the motion is different, the mutual information becomes lower.

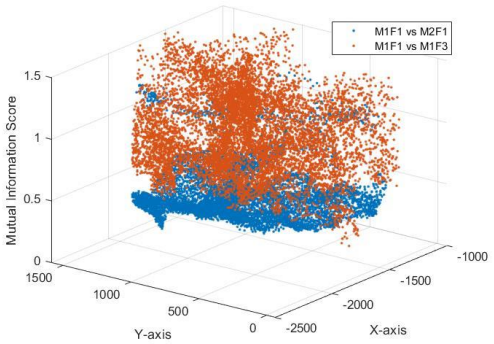


Figure 17 $MI_{motions}$ and $MI_{fabrics}$ in 3-dimensional space to show the trend that $MI_{fabrics}$ has higher mutual information

It is shown that how well the sensors on tight clothes work to discriminate two motions through computing accuracy after supervised learning using SVM. And, in this section, mutual information is used to calculate how much slack clothes on the same workout can make the difference in the amount of information on sensors between M1F1 and M1F3. Sensors in the same location on different clothes showed that those shared a large amount of information and are dependent on each other. Here, the sensors are chosen and coloured red on figure 18 that share the most information among the sensors attached to the two different types of cloth for each section of the body in table 1. Figure 18 shows 11 sensors with the biggest mutual information in each part of the body. In figure 18-(a), it is observed that the three sensors selected are concentrated in the right shoulder. This means that the signal produced by the sensors in the right shoulder between M1F1 and M1F3 are similar as well as the noise produced by loose clothing on the right shoulder is minimized compared to other sensors on back, chest and right arm. In figure 18-(c), the model is lying on her stomach and the camera was shot at the foot, where two sensors are placed on the elbow in the left arm. On the other arm, a sensor is located near the wrist. In both calves, two sensors are located at both sides of the malleolus. The sensor for the left thigh can be clearly observed in figure 18-(f) in which the sensor is positioned in the middle of the front thigh as well as the sensor for the right thigh is shown in the middle of the right thigh, slightly above the knee.

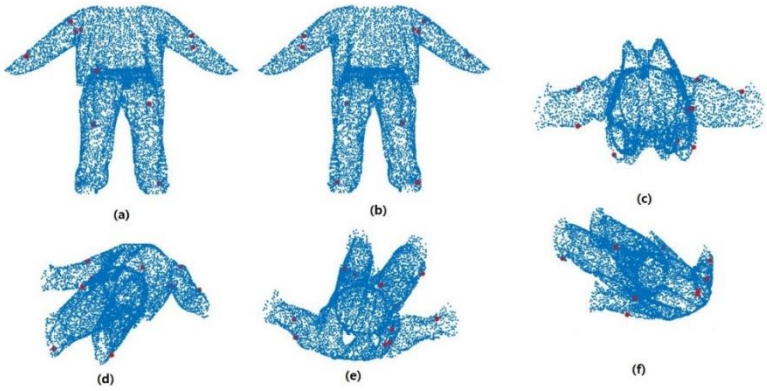


Figure 18 Sensors with higher $MI_{fabrics}$ for each body parts in various angle of shooting

5.1.3 Result of sensor placement analysis using difference of mutual information between $MI_{fabrics}$ and $MI_{motions}$

Two set of data, $MI_{motions}$ and $MI_{fabrics}$ (the shape of matrix for both is 8472×1 where 8472 is the number of sensors), were generated as a result of the mutual information study so far where $MI_{motions}$ is produced by calculating mutual information between two dataset $M1F1^{\{I\times F\times S\}}$ and $M2F2^{\{I\times F\times S\}}$ and $MI_{fabrics}$ is produced using $M1F1^{\{I\times F\times S\}}$ and $M2F2^{\{I\times F\times S\}}$ where I, F, and S for all the dataset are 1 dimensional values, 554 frames, and 8472 sensors respectively. For the $MI_{motions}$, the lower the amount of mutual information, the better the sensor since it means the sensor for two motions has big discriminative power with low overlapping amount of information. Conversely, in the dataset $MI_{fabrics}$, it was focused on finding sensors with high mutual information between two slackness of clothes since it is apparent that the place where the sensor can be affixed makes less affected by the rippling of the loose-fitting cloth (i.e. at the placement, a sensor shows large mutual information between F1 and F3). Using those data, it is evident that the optimal sensor location shows more discriminative between motion and less affected by noise generated by loose-fitting clothing (i.e. the larger the values in $MI_{fabrics}$ based on a specific sensor, the better, and the smaller the values in $MI_{motions}$, the more discriminative the sensor is). The values for higher $MI_{fabrics}$ and lower $MI_{motions}$ for each sensor are simply computed by subtraction of two matrices, $MI_{motions}$ and $MI_{fabrics}$, which is denoted $MI_{difference}$. Figure 19 illustrates colormap of $MI_{difference}$ for 8472 sensors where x and y-axis indicate the coordinates to show the shape of cloth and z–axis shows the values of $MI_{difference}$ for each sensor. The values are constrained between 0 to 1.5. As can be seen in figure 19-(a), the difference between $MI_{motions}$ and $MI_{fabrics}$ is uniformly distributed all over the body. From the side, it can be seen that the distribution of points is almost rectangular in figure 19-(b). As a feature, the sensors attached to the left side of the body appear to have a larger difference than the right side of the body in figure 19-(c). At this time, the smaller the amount of mutual information between different motion is, the more powerful to discriminate both motion the sensor is, as well as, for the sensor, the higher mutual information between different fabric means the larger information is shared at the sensor location between different fabric (i.e. at the sensor placement, the influx of uncertain information by loose clothing has been minimized).

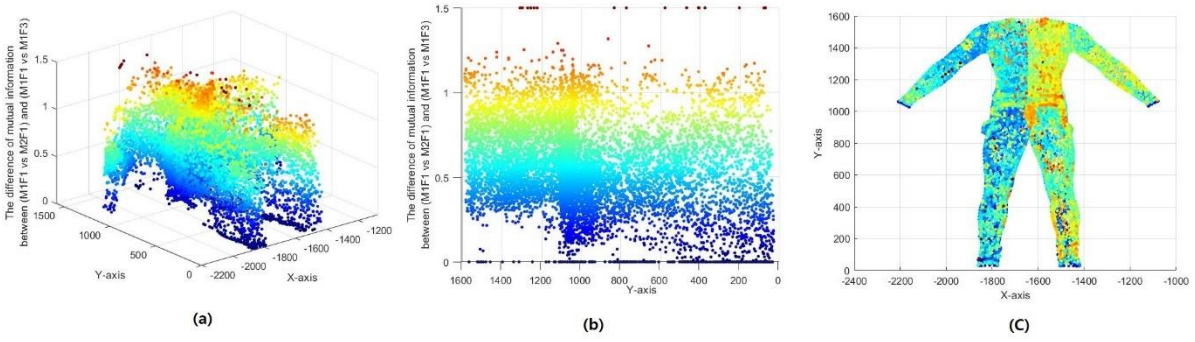


Figure 19 Colormap using the values of $MI_{difference}$ where red-coloured sensor shows larger gap between $MI_{fabrics}$ and $MI_{motions}$, and blue sensor shows smaller difference of MI

Figure 20 shows the optimal sensor placement based on the analysis of mutual information, $MI_{difference}$, at various angle of the shoot to show the exact location of optimal sensors. The red points present the biggest value of $MI_{difference}$ for each part of the body. For the chest, the optimal sensor is placed near the right rip as can be seen in figure 20-(d) as well as the optimal sensors for the abdominal region and back are located at right upper abdomen and right mitral muscle respectively in figure 20-(a) and (b). The optimal sensors for right arm and forearm are positioned near the outside elbow, and similarly the optimal sensor for left arm is located near the inside of elbow. In the middle of the left forearm, optimal sensor is placed as observed

clearly in figure 20-(c) (figure 20-(c) is the picture of lying model shoot from underfoot). The optimal sensor for right thigh is at outer thigh as can be seen in figure 20-(d) and the sensor for the left thigh is placed under the left bottom of buttock. The optimal sensors for both calves are placed near both ankles.

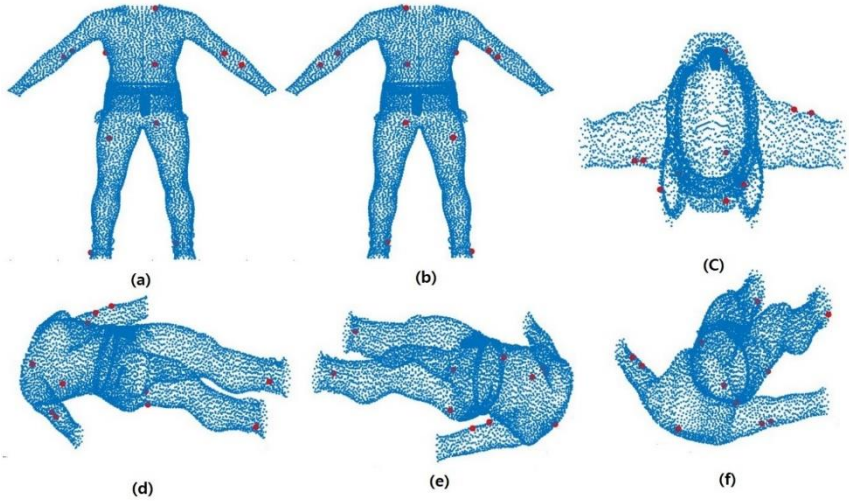


Figure 20 Sensors with higher $MI_{difference}$ for each body parts in the various angle of shooting

5.2 Final optimal sensor placement derived from comprehensive results

In this section, the final results based on the analysis using machine learning and mutual information in this research will be comprehensively presented. This section can be divided into three parts. In the first part, the results of the analysis of sensors attached to the body using machine learning will be summarized. In the second part, the results of analysis by location of sensors using mutual information will be reviewed. Finally, based on analysis using machine learning and mutual information above, the final optimal sensor placements will be finally derived.

5.2.1 Analysis of clothing sensor using machine learning for its optimal placement

The main challenges in section 4.2 were to find out which part of the body the sensors work well when the dataset the sensors make is trained using support vector machine (SVM), and which features should be used to make the machine to compute accuracy in every part of the body. The features are selected using sequential feature selection (SFS) for each part of the body and the three most significant features are selected after feature selection for every part of the body. How areas of the body are divided into parts can be seen in Table 5. Along with this, which three features are chosen for machine learning for each part can be found in Table 6. Figure 21 shows a clustered column chart that is reconstructed from Table 7 in section 4.2.4. Figure 21 indicates the motion predictability of sensors grouped by body parts which are computed by the number of correct predictions divided by total number of samples times time 100. In Figure 21, high accuracy is shown in all areas of the body. Among them, chest, right and left arm, and left forearm are seen as the most suitable parts of the body to place sensors as 100% of accuracy. Those places are written down to Table 8 for easy viewing with the best positions of sensors the research [18] has found.

Table 8 Comparison of optimal body parts to put a sensor with existing research

Reference	Position
L. Atallah et al. [18]	Chest, wrist and waist
This research	Chest, left arm and forearm, and right arm

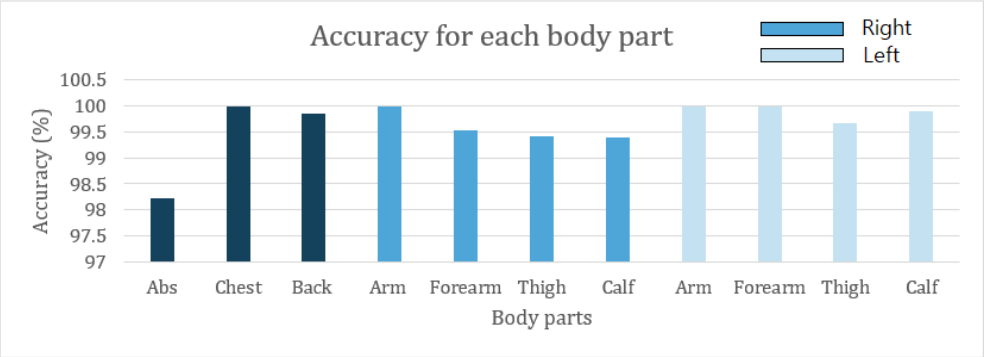


Figure 21 Accuracy of sensor prediction for each body parts tested using SVM

5.2.2 Analysis of clothing sensor using mutual information for its optimal placement

In this section, mutual information was used to find the optimum sensor placement. In here, precise optimal sensor locations are computed, unlike finding optimal sensor placement in body parts using machine learning. The calculation of mutual information was used under the following two conditions. First, it was used to calculate the difference in the amount of information when two motions of a model wearing the same clothes were taken. Next, the difference in mutual information between models wearing different clothes and taking the same motion was calculated to see how much difference a sensor would make when wearing loose clothes. Figure 22 shows the sensor positions obtained from the above two situations. In Figure 22-(a) and (b), sensors with the lowest amount of mutual information between the two motions are marked in each part of the body. And, in figure 22-(c) and (d), sensors with the biggest amount of mutual information between two different slackness of cloth are stamped by red colour point. (a), (c) and (b), (d) are views taken from the front and back respectively.

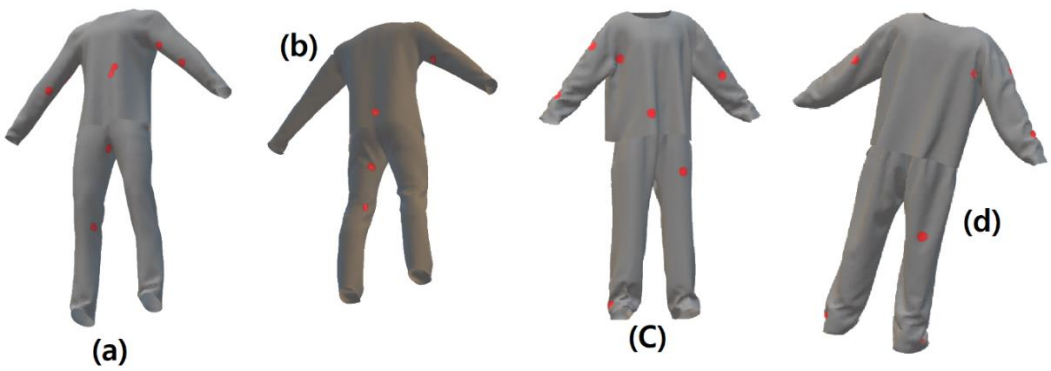


Figure 22 Optimal sensor placement for better motion recognition (a) and (b), and sensor position minimizing flutter of clothing (c) and (d)

5.2.3 Optimal sensor placement by comprehensive analysis of data produced by using mutual information and machine learning

So far, those have been analysed under three conditions to find the best sensor location: first to find the body parts with the highest accuracy using machine learning, second to find the sensors with the smallest amount of mutual information in the two motion, and finally to find out how much the sensors have shared mutual information in tight clothes and loose clothes. Finally, sensors that satisfy all of the above conditions are as follows in Figure 23. The placement of the sensors in Figure 23 is same to the sensor locations in Figure 20, but the figure below shows only the sensors shown among the sensors in Figure 20 that are 100% accurately classified through machine learning, that regions of the body can be reviewed in Table 7. As such, four optimal sensors are selected through a comprehensive analysis of all data made in this research. As a result, the optimal sensors are concentrated in the upper body rather than the lower body, as well as the 4 sensors are divided into two left and two right sides on the centre of the body. There are the two differences from the results of the previous research : first, the positions of these sensors are more precise than that of the sensor positions shown in existing studies (existing research shows only approximate body parts such as chest and right thigh etc. as can be seen in Table 1), and In regard the second difference the location of these sensors is where the amount of information changes due to the slackness of clothing is minimized.

Overall, the position of the sensors seen in figure 23 is the position with the least effect from the flutter of the garment and the best position for motion prediction at the same time.

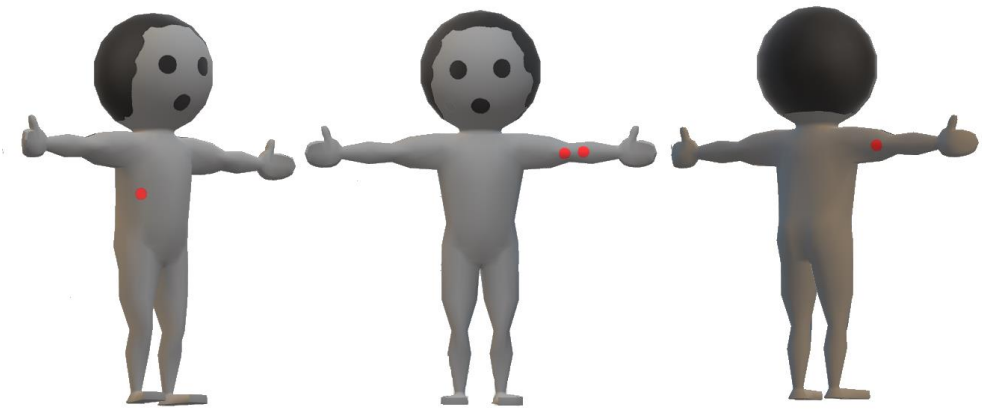


Figure 23 Final optimal sensor placements having the highest accuracy using SVM, having the least amount of mutual information between different motions, and minimizing the flutter of clothing at the same time

6 CONCLUSION

In this research, a framework for investigating the optimal sensor placement through machine learning and analysis of mutual information is presented. It is evident that all four optimal sensor positions found in this study are located in the upper body. One feature of my research differs from the previous study as can be seen in Table 3, in which wrist, ankle, or waist were classified as the primary optimal sensor placement for motion tracking. However, my analysis shows that optimal sensor placements are chosen near the mid-arm or the area between the upper and mid trunk of the body rather than near the sleeve of a garment. Intuitively, it can be seen that the optimal sensor placement avoids the area of the garment's wrist to minimize the flap of the sleeve as can be seen in figure 23. Thus, we looked for sensor locations that help patients with neurological diseases for continuous monitoring their motion sickness in everyday clothes by which placements neuro-related patients no longer have to wear tight clothes with sensors affixed.

In detail, the initial dataset was successfully transformed into one-dimensional data in the frame domain by the sensor through rearrangement. Through this work, one-dimensional data from each sensor can be analyzed using traditional signal and discrete random distribution theories. The part that uses machine learning to find the optimal area of sensor placement had satisfactory results, but there was limitation. It used a huge amount of sensor data, but only two motions were labelled for it, which greatly increased the accuracy as 100% of the classification using SVM, and this reduced the sensitivity of the classification test for body parts. To overcome this limitation, there was no choice but considerably extending test set as 70 presents to give the discriminative power to each body parts (i.e. to reduce the accuracy of classification by mean of manipulation to show the different accuracy for each body parts). In the analysis through mutual information, the optimal sensor location was found by measuring how close or far of a sensor in various situations. This has the limitation of measuring the distance between each motion or the type of clothing based on the amount of mutual information, not using the exact information contained in it. However, measuring distance using the amount of mutual information (i.e. how far the mutual information is between two different motion, as well as how close the mutual information is between two sizes of clothing) showed that it is possible to find the optimal sensor placement.

For future work, it is expected the classification to be carried out with various motions, as well as if the study of the amount of multivariate mutual information (MMI) has become clear in its interpretation and we can actually figure out what it is, it can be used to extract the multivariate mutual information among different motions and fabrics of sensors as can be seen in Figure 24

where the set I(y; z|x) with sky-blue shows the optimized sensor placement using multivariate mutual information rather than in figure 12.

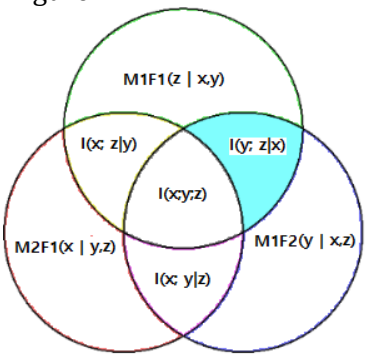


Figure 24 Venn diagram of multivariate mutual information for three variables x, y, and z

7 REFERENCES

[1] *Ciuti, G. et al., "MEMS Sensor Technologies for Human Centred Applications in Healthcare, Physical Activities, Safety and Environmental Sensing: A Review on Research Activities in Italy Sensors.", 2015.*

[2] *Xiao, X. and Zarar, S., "Machine Learning for Placement-insensitive Inertial Motion Capture." 2018 IEEE International Conference on Robotics and Automation, 2018.*

[3] *Bersotti, F. M. et al., "Human Motion Capture Via Inertial Sensors for Clinical And Sports Applications.", 2018.*

[4] *Berezina, B., "Motion Capture History, Technologies and Applications" [PowerPoint presentation], 2003.*

[5] *Cover, T. M. & Thomas, J. A., "Elements of Information Theory", John Wiley & Sons, Incorporated, Hoboken, 2006.*

[6] *John E. H., "Signal Processing Using Mutual Information, IEEE SIGNAL PROCESSING MAGAZINE" [Online pdf], 2006.*

[7] Li J. et al., "Permutation Conditional Mutual Information and Its Application to Epileptic EEG," 2013 International Conference on Computer Sciences and Applications, 2013.

[8] *Mert A. & Akan A., "Analysis of EEG signals by emprical mode decomposition and mutual information," 21st Signal Processing and Communications Applications Conference", 2013.*

[9] *Cover, T. M., & Thomas, J. A., "Elements of information theory." Retrieved from https://ebookcentral.proquest.com, 2006.*

[10] *Abdi, H. & Williams, L. J., "Principal component analysis." WIREs Comp Stat, 2: 433-459. doi:10.1002/wics.101, 2010.*

[11] *"Pearson's correlation coefficient " [online material] available at: http://www.hep.ph.ic.ac.uk/~hallg/UG_2015/Pearsons.pd, 2015.*

[12] *Huitema, B. & Laraway, S., "Autocorrelation.", 2006*

[13] *Rückstieß, T. et al., "Sequential Feature Selection for Classification."10.1007/978-3-642-25832-9_14, 2011*

[14] *Andrew N, Support Vector Machines [Lecture notes] Available at: http://cs229.stanford.edu/notes/cs229-notes3.pdf*

[15] *Brown, S., "Measures of Shape: Skewness and Kurtosis" Available at: http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/SkewStatSignif.pdf, 20018-2011*

[16] *Heckbert P " Fourier Transforms and the Fast Fourier Transform (FFT) Algorithm" Available at: http://www.cs.cmu.edu/afs/andrew/scs/cs/15463/2001/pu b/www/notes/fourier/fourier.pdf, 1995*

[17] *Orha, I., and S. Oniga. "Study regarding the optimal sensors placement on the body for human activity recognition." 2014IEEE 20th international Symposium for Design and Technology in Electronic Packaging (SIITME). IEEE, 2014.*

[18] *L. Atallah et al. "Sensor Placement for Activity Detecting using Wearable Accelerometers." Imperial College London. IEEE.*

[19] *Signh, V. K. et al. "Human Motion Prediction Using Machine Learning and Signal Processing." 2nd International Conference on Communication and Electronics Systems (ICCES 2017). IEEE. 2017.*

[20] *Park, J. and Cho, S. "IR-UWB Radar Sensor for Human Gesture recognition by Using Machine Learning." 18th International Conference on High Performance Computing and Communications, IEEE. 2016.*

Faculty of Natural and Mathematical Sciences
Department of Informatics

Bush House, 30 Aldwych
Strand Campus
London WC2B 4BG
United Kingdom
https://www.kcl.ac.uk/nms/depts/informatics

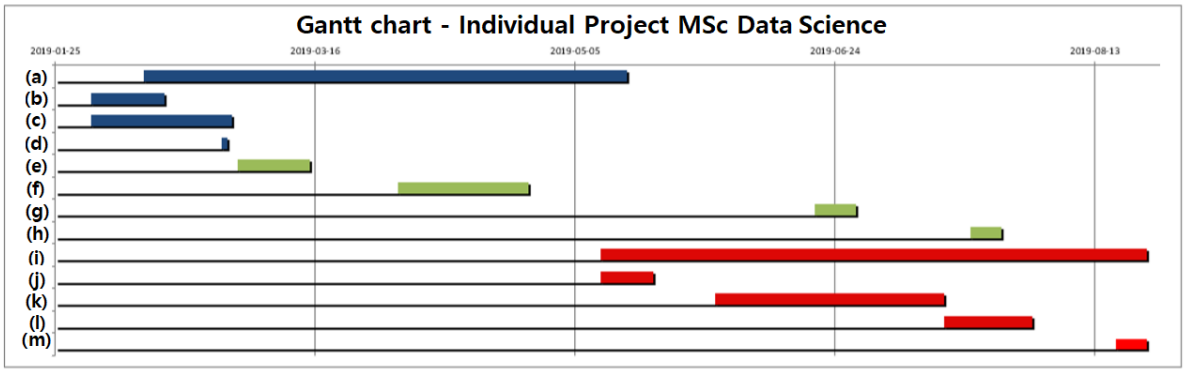
8 APPENDICES

8.1 Appendix: Project plan with Gantt chart

(i) Project plan

	Task Name	Start	End	Duration (days)
Literature Review	(a) Literature review	2019-02-11	2019-05-15	93
	(b) Set Project Direction	2019-02-01	2019-02-15	14
	(c) Background Research	2019-02-01	2019-02-28	27
	(d) Presentation and Plargiarism (Session)	2019-02-26	2019-02-27	1
Interim Report	(e) Data Acquisition Report	2019-03-01	2019-03-15	14
	(f) Preliminary Project Report	2019-04-01	2019-04-26	25
	(g) 1st MSc Project progress report	2019-06-20	2019-06-28	8
	(h) 2st MSc Project progress report	2019-07-20	2019-07-26	6
Final Report	(i) Technical Contribution	2019-05-10	2019-08-23	105
	(j) Background, Aims and Objectives	2019-05-10	2019-05-20	10
	(k) Analysis and Evaluation	2019-06-01	2019-07-15	44
	(l) Conclusion	2019-07-15	2019-08-01	17
	(m) Document work	2019-08-17	2019-08-23	6
Presentation	(n) Presentation Practice	2019-08-26	2019-09-13	18

(ii) Gantt chart



8.2 Appendix: Source Codes

8.2.1 Python codes

8.2.1.1 Data rearrangement – converting .obj into .txt file

```
import random

d = ["{0:03}".format(i) for i in range(555)]
del d[0]

sampling_index = list(range(0,15042))
sampling = random.sample(sampling_index,k= 8472)
sampling = sorted(sampling)

data = []
for i in range(len(d)):
    L = []
    f =
open("C:\Users\hwani\Desktop\Project\Dataset\P1M1C1F3\data_0000%s.txt"%d[i],"r")
    lines = file.readlines(f)
    file.close(f)

    for i in range(len(lines)):
        L.append(lines[i].split('\n')[0])

    del L[0]
    del L[0]

    data1 = []
    for i in range(len(L)):
        if 'n' not in L[i] and 't' not in L[i] and 'f' not in L[i] and 's' not in
L[i]:
            data1.append(L[i])
    data1 = data1[0:15041] ## 8472 -> Cloth

    sampled_data = []
    for i in range(len(sampling)):
        index = sampling[i]
        sampled_data.append(data1[index])

    data.append(sampled_data)

for i in range(len(data)):
    print i
    f =
open("C:\Users\hwani\Desktop\Project\Dataset\P1M1C1F3\databyframe\data%s.txt"%i,"w")
    for j in range(len(data[i])):
        a = str(data[i][j])
        a = a.strip("v ")
        f.write(a)
        f.write("\n")
f.close()
```

8.2.1.2 Converting frame-based files into sensor-based

```
import numpy as np

d = list(range(554))
trajectories = []
data = []

for i in range(len(d)):
    L = []
    f =
open("C:\Users\hwani\Desktop\Project\Dataset\P1M1C1F3\databyframe\data%s.txt"%d[i],
"r")
    lines = file.readlines(f)
    file.close(f)

    for i in range(len(lines)):
        L.append(lines[i].split('\n')[0])

    data.append(L)


data = np.array(data).T.tolist()

for i in range(len(data)):
    f =
open("C:\Users\hwani\Desktop\Project\Dataset\P1M1C1F3\datasensors\sensor%s.txt"%i,"
w")
    for j in range(len(data[i])):
        a = str(data[i][j])
        a = a.strip("v ")
        f.write(a)
        f.write("\n")
f.close()
```

8.2.1.3 Calculating principal component analysis (PCA) for each sensor

```
import numpy as np
from numpy import linalg as LA

d = list(range(8472))
final_data = []
final_data_1d = []
for i in range(len(d)):
    print i
    L = []
    f =
open("C:\Users\hwani\Desktop\Project\Dataset\P1M1C1F3\datasensors\sensor%s.txt"%d[i
],"r")
    lines = file.readlines(f)
    file.close(f)

    for j in range(len(lines)):
        L.append(lines[j].split('\n')[0])

    ### Use the Karhunen-Loeve Transform
    data = []
```

```
    for i in range(len(L)):
        l = L[i]
        b = l.split()
        pem = []
        for k in range(len(b)):
            a = float(b[k])
            pem.append(a)
        data.append(pem)

    data_min = []
    a = []
    b = []
    c = []
    for i in range(len(data)):
        a.append(data[i][0])
        b.append(data[i][1])
        c.append(data[i][2])
    data_min.append(sum(a)/len(a))
    data_min.append(sum(b)/len(b))
    data_min.append(sum(c)/len(c))

    ### Zero-mean data
    zeromean_data = []
    for i in range(len(data)):
        per = []
        x = data[i][0]-data_min[0]
        y = data[i][1]-data_min[1]
        z = data[i][2]-data_min[2]
        per = [x, y, z]
        zeromean_data.append(per)

    ### Covariance Matrix
    #   a   b   c
    #   d   e   f
    #   g   h   i
    covar_data = []
    cor_a = []
    cor_b = []
    cor_c = []
    cor_d = []
    cor_e = []
    cor_f = []
    cor_g = []
    cor_h = []
    cor_i = []
    for i in range(len(zeromean_data)):
        cor_a.append(zeromean_data[i][0] * zeromean_data[i][0])
        cor_b.append(zeromean_data[i][0] * zeromean_data[i][1])
        cor_c.append(zeromean_data[i][0] * zeromean_data[i][2])
        cor_d.append(zeromean_data[i][1] * zeromean_data[i][0])
        cor_e.append(zeromean_data[i][1] * zeromean_data[i][1])
        cor_f.append(zeromean_data[i][1] * zeromean_data[i][2])
        cor_g.append(zeromean_data[i][2] * zeromean_data[i][0])
        cor_h.append(zeromean_data[i][2] * zeromean_data[i][1])
        cor_i.append(zeromean_data[i][2] * zeromean_data[i][2])

    cor_a = sum(cor_a)/len(cor_a)
    cor_b = sum(cor_b)/len(cor_b)
    cor_c = sum(cor_c)/len(cor_c)
    cor_d = sum(cor_d)/len(cor_d)
    cor_e = sum(cor_e)/len(cor_e)
    cor_f = sum(cor_f)/len(cor_f)
    cor_g = sum(cor_g)/len(cor_g)
    cor_h = sum(cor_h)/len(cor_h)
    cor_i = sum(cor_i)/len(cor_i)

    covar_data = [[cor_a,cor_b,cor_c],[cor_d,cor_e,cor_f],[cor_g,cor_h,cor_i]]
```

```
    ##eig values and vectors
    A = np.array(covar_data)
    w, v = LA.eig(A)
    w_list = w.tolist()

    p = w_list[0]
    q = w_list[1]
    r = w_list[2]

    if isinstance(p,complex):
        w_list[0] = p.real
    if isinstance(q,complex):
        w_list[1] = q.real
    if isinstance(r,complex):
        w_list[2] = r.real

    min_index = w_list.index(min(w_list))
    max_index = w_list.index(max(w_list))

    v_reduced = np.delete(v,min_index,1)
    index_1d = []

    if max_index == 0:
        index_1d.append(1)
        index_1d.append(2)
    if max_index == 1:
        index_1d.append(0)
        index_1d.append(2)
    if max_index == 2:
        index_1d.append(0)
        index_1d.append(1)

    v_reduced_1d = np.delete(v,index_1d[0],1)
    v_reduced_1d = np.delete(v,index_1d[1],1)

    v_reduced_list = v_reduced.tolist()
    v_reduced_list_1d = v_reduced_1d.tolist()

    # examplars pernetration
    reduced_data = []

    for i in range(len(data)):
        y = []
        y_1 =
v_reduced_list[0][0]*data[i][0]+v_reduced_list[1][0]*data[i][1]+v_reduced_list[2][0
]*data[i][2]
        y_2 =
v_reduced_list[0][1]*data[i][0]+v_reduced_list[1][1]*data[i][1]+v_reduced_list[2][1
]*data[i][2]
        y.append(y_1)
        y.append(y_2)
        reduced_data.append(y)
    final_data.append(reduced_data)

    reduced_data_1d = []

    for i in range(len(data)):
        j = []
        j_1 =
v_reduced_list_1d[0][0]*data[i][0]+v_reduced_list_1d[1][0]*data[i][1]+v_reduced_lis
t_1d[2][0]*data[i][1]
        j.append(j_1)
        reduced_data_1d.append(j)
    final_data_1d.append(reduced_data_1d)

for i in range(len(final_data)):
    print i
    f =
```

```
open("C:\Users\hwani\Desktop\Project\Dataset\P1M1C1F3\PCA2D\sensor%s.txt"%d[i],"w")
    for k in range(len(final_data[i])):
        a = str(final_data[i][k])
        a = a.strip("[")
        a = a.strip("]")
        a = a.replace(',','')
        print a
        f.write(a)
        f.write("\n")
f.close()

for i in range(len(final_data_1d)):
    print i
    f =
open("C:\Users\hwani\Desktop\Project\Dataset\P1M1C1F3\PCA1D\sensor%s.txt"%d[i],"w")
    for k in range(len(final_data_1d[i])):
        a = str(final_data_1d[i][k])
        a = a.strip("[")
        a = a.strip("]")
        a = a.replace(',','')
        print a
        f.write(a)
        f.write("\n")
f.close()
```

8.2.1.4 Calculating mutual information and correlation for each sensor

```
from sklearn.metrics.cluster import mutual_info_score
import math
from math import pow

d = list(range(8472))
data_m1 = []
for i in range(len(d)):
    print i
    L = []
    f =
open("C:\Users\hwani\Desktop\Project\Dataset\data_m1_m2\PCA1D_m1\data%s.txt"%d[i],"
r")
    lines = file.readlines(f)
    file.close(f)

    for j in range(len(lines)):
        L.append(lines[j].split('\n')[0])
    data_m1.append(L)

data_m2 = []
for i in range(len(d)):
    print i
    L = []
    f =
open("C:\Users\hwani\Desktop\Project\Dataset\data_m1_m2\PCA1D_f3\sensor%s.txt"%d[i]
,"r")
    lines = file.readlines(f)
    file.close(f)

    for j in range(len(lines)):
        L.append(lines[j].split('\n')[0])
    data_m2.append(L)

data_float_m1 =[]
```

```
for i in range(len(data_m1)):
    print i
    per = []
    for j in range(len(data_m1[i])):
        a = float(data_m1[i][j])
        per.append(a)
    data_float_m1.append(per)

data_float_m2 =[]
for i in range(len(data_m2)):
    print i
    per = []
    for j in range(len(data_m2[i])):
        a = float(data_m2[i][j])
        per.append(a)
    data_float_m2.append(per)

corr = []
MI = []
for i in range(len(data_float_m1)):
    print i
    data_float_m1_rounded=[]
    data_float_m2_rounded=[]
    for k in range(len(data_float_m1[i])):
        a = round(data_float_m1[i][k],-1)
        data_float_m1_rounded.append(a)
        b = round(data_float_m2[i][k],-1)
        data_float_m2_rounded.append(b)
    c = mutual_info_score(data_float_m1_rounded,data_float_m2_rounded)
    MI.append(c)

    mean_m1 = sum(data_float_m1[i]) / len(data_float_m1[i])
    mean_m2 = sum(data_float_m2[i]) / len(data_float_m2[i])
    zero_mean = []
    zero_mean_x = []
    zero_mean_y = []
    for k in range(len(data_float_m1[i])):
        a = (data_float_m1[i][k] - mean_m1) * (data_float_m2[i][k] - mean_m2)
        b = pow(data_float_m1[i][k] - mean_m1, 2)
        c = pow(data_float_m2[i][k] - mean_m2, 2)
        zero_mean.append(a)
        zero_mean_x.append(b)
        zero_mean_y.append(c)
    d = sum(zero_mean) / len(data_float_m1[i])
    e = sum(zero_mean_x) / len(data_float_m1[i])
    f = sum(zero_mean_y) / len(data_float_m1[i])
    g = math.sqrt(e * f)
    h = d / g
    corr.append(h)

f =
open("C:\Users\hwani\Desktop\Project\Dataset\data_m1_m2\MIandCorr_loosingfit\corr.t
xt","w")
for k in range(len(corr)):
    print k
    a = str(corr[k])
    a = a.strip("[")
    a = a.strip("]")
    a = a.replace(',','')
    f.write(a)
    f.write("\n")
f.close()

f =
open("C:\Users\hwani\Desktop\Project\Dataset\data_m1_m2\MIandCorr_loosingfit\MI.txt
","w")
for k in range(len(MI)):
    print k
```

```
    a = str(MI[k])
    a = a.strip("[")
    a = a.strip("]")
    a = a.replace(',','')
    f.write(a)
    f.write("\n")
f.close()
```

8.2.1.5 Computing 15 features for each sensor

```
from __future__ import division
import numpy as np
from scipy.stats import entropy
from scipy.stats import kurtosis
from scipy.stats import skew
import math

d = list(range(8472))
sensors_m1 = []
for i in range(len(d)):
    print i
    L = []
    f =
open("C:\Users\hwani\Desktop\Project\Dataset\P1M2C1F1\PCA1D\sensor%s.txt"%d[i],"r")
    lines = file.readlines(f)
    file.close(f)
    for j in range(len(lines)):
        L.append(lines[j].split('\n')[0])
    sensors_m1.append(L)

sensor = []
for i in range(len(sensors_m1)):
    a = map(float, sensors_m1[i])
    sensor.append(a)

# variance
variance = []
for i in range(len(sensor)):
    a = np.var(sensor[i])
    variance.append(a)

#RMS value & MEAN
RMS = []
MEAN = []
for i in range(len(sensor)):
    a = np.asarray(sensor[i])
    mean = np.mean(a)
    b = np.mean(a**2)
    c = np.sqrt(b)
    RMS.append(c)
    MEAN.append(mean)

# entropy
ENTROPY_value = []
for i in range(len(sensor)):
    max_value = max(sensor[i])
    min_value = min(sensor[i])
    range_r = max_value-min_value
    range_bin = range_r/10
    probability = []
```

```
    bin0 = []
    bin1 = []
    bin2 = []
    bin3 = []
    bin4 = []
    bin5 = []
    bin6 = []
    bin7 = []
    bin8 = []
    bin9 = []

    for k in range(len(sensor[i])):
        if sensor[i][k] >= min_value and sensor[i][k] < min_value+range_bin:
            bin0.append(1)
        elif sensor[i][k] >= min_value+range_bin and sensor[i][k] <
min_value+range_bin*2:
            bin1.append(1)
        elif sensor[i][k] >= min_value+range_bin*2 and sensor[i][k] <
min_value+range_bin*3:
            bin2.append(1)
        elif sensor[i][k] >= min_value+range_bin*3 and sensor[i][k] <
min_value+range_bin*4:
            bin3.append(1)
        elif sensor[i][k] >= min_value+range_bin*4 and sensor[i][k] <
min_value+range_bin*5:
            bin4.append(1)
        elif sensor[i][k] >= min_value+range_bin*5 and sensor[i][k] <
min_value+range_bin*6:
            bin5.append(1)
        elif sensor[i][k] >= min_value+range_bin*6 and sensor[i][k] <
min_value+range_bin*7:
            bin6.append(1)
        elif sensor[i][k] >= min_value+range_bin*7 and sensor[i][k] <
min_value+range_bin*8:
            bin7.append(1)
        elif sensor[i][k] >= min_value+range_bin*8 and sensor[i][k] <
min_value+range_bin*9:
            bin8.append(1)
        elif sensor[i][k] >= min_value+range_bin*9 and sensor[i][k] <=
min_value+range_bin*10:
            bin9.append(1)

    a = bin0.count(1)
    b = bin1.count(1)
    c = bin2.count(1)
    d = bin3.count(1)
    e = bin4.count(1)
    f = bin5.count(1)
    g = bin6.count(1)
    h = bin7.count(1)
    i = bin8.count(1)
    j = bin9.count(1)

    overall = a+b+c+d+e+f+g+h+i+j

    probability.append(a / overall)
    probability.append(b / overall)
    probability.append(c / overall)
    probability.append(d / overall)
    probability.append(e / overall)
    probability.append(f / overall)
    probability.append(g / overall)
    probability.append(h / overall)
    probability.append(i / overall)
    probability.append(j / overall)

    u = entropy(probability)
    ENTROPY_value.append(u)
```

```
# Auto_correlation
Auto_correlation = []
for i in range(len(sensor)):
    y = sensor[i]-np.mean(sensor[i])
    correlated = np.correlate(y,y, mode= 'full')
    s = max(correlated)
    Auto_correlation.append(s)


# 3 major frequency using fft
first_amplitude = []
second_amplitude = []
third_amplitude = []
first_frequency = []
second_frequency = []
third_frequency = []
for i in range(len(sensor)):
    Fs = 2000
    T = 0.00005
    te = 10
    t = np.arange(0,te,T)
    y = sensor[i]
    n = len(y)  # Length of signal
    NFFT = n  # ?? NFFT=2^nextpow2(length(y))
    k = np.arange(NFFT)
    f0 = k * Fs / NFFT  # double sides frequency range
    f0 = f0[range(math.trunc(NFFT / 2))]  # single sided frequency range

    Y = np.fft.fft(y) / NFFT  # fft computing and normaliation
    Y = Y[range(math.trunc(NFFT / 2))]  # single sied frequency range
    amplitude_Hz = 2 * abs(Y)
    phase_ang = np.angle(Y) * 180 / np.pi

    amplitude = amplitude_Hz.tolist()
    sorted_amplitude = sorted(amplitude,reverse=True)
    the_biggest_amplitude = sorted_amplitude[0]
    the_second_amplitide = sorted_amplitude[1]
    the_third_amplitude = sorted_amplitude[2]

    first_amplitude.append(the_biggest_amplitude)
    second_amplitude.append(the_second_amplitide)
    third_amplitude.append(the_third_amplitude)

    first_index = amplitude.index(sorted_amplitude[0])
    second_index = amplitude.index(sorted_amplitude[1])
    third_index = amplitude.index(sorted_amplitude[2])

    first_frequency.append(f0[first_index])
    second_frequency.append(f0[second_index])
    third_frequency.append(f0[third_index])

#Skewness & Kurtosis
Skewness = []
Kurtosis = []
for i in range(len(sensor)):
    a = kurtosis(sensor[i])
    b = skew(sensor[i])
    Kurtosis.append(a)
    Skewness.append(b)

# Maximum & Minimum value
Maximum_value = []
Minimum_value = []
for i in range(len(sensor)):
    a = max(sensor[i])
    b = min(sensor[i])
    Maximum_value.append(a)
```

```
    Minimum_value.append(b)

Features = []
for i in range(len(Minimum_value)):
    attributes = []

    attributes.append(Minimum_value[i])
    attributes.append(Maximum_value[i])
    attributes.append(Skewness[i])
    attributes.append(Kurtosis[i])
    attributes.append(first_amplitude[i])

    attributes.append(second_amplitude[i])
    attributes.append(third_amplitude[i])
    attributes.append(first_frequency[i])
    attributes.append(second_frequency[i])
    attributes.append(third_frequency[i])

    attributes.append(Auto_correlation[i])
    attributes.append(ENTROPY_value[i])
    attributes.append(RMS[i])
    attributes.append(MEAN[i])
    attributes.append(variance[i])

    Features.append(attributes)

f =
open("C:\Users\hwani\Desktop\Dissertation\Approach\Features\Featuresdata\M2_15Featu
res\Features_M2.txt","w")
for i in range(len(Features)):
    a = str(Features[i])
    a = a.strip("[")
    a = a.strip("]")
    f.write(a)
    f.write("\n")
f.close()
```

8.2.2 MATLAB codes

8.2.2.1 Computing support vector machine (SVM) for each body parts

```
%M1 = table2array(a)
%M2 = table2array(b)
X = [M1;M2]
y = [1+ zeros(561,1); 2+zeros(561,1)]
data = [X,y]
data_shuffled = data(randperm(size(data,1)),:)

X_test = data_shuffled(1:785,1:15)
y_test = data_shuffled(1:785,16)

X_train = data_shuffled(786:end,1:15)
y_train = data_shuffled(786:end,16)

%% Feature Selection
```

```
 pts = statset('display','iter');
classf = @(train_data, train_labels, test_data, test_labels)...
   sum(predict(fitcsvm(train_data, train_labels,'KernelFunction','rbf'), test_data) ~=
test_labels);

[fs, history] = sequentialfs(classf, X_train, y_train, 'cv', c, 'options', opts,'nfeatures',2);

%% Best hyperparameter

X_train_w_best_feature = X_train(:,fs);
Md1 = fitcsvm(X_train_w_best_feature, y_train,'KernelFunction','rbf',...
   'OptimizeHyperparameters','auto',...
   'HyperparameterOptimizationOptions',struct('AcquisitionFunctionName',...
   'expected-improvement-plus','Showplots',false));

%% Test

X_test_w_best_features = X_test(:,fs);

test_accuracy = sum(predict(Md1,X_test_w_best_features) == y_test)/length(y_test)*100;

predicted = predict(Md1,X_test_w_best_features)

result_matrix = [predicted y_test]
[a b] = size(result_matrix)
TP = []
FP = []
FN = []
TN = []
for N = 1:a
   if result_matrix(N,1) == 1 && result_matrix(N,1) ==1
      TP = [TP 1]
   elseif result_matrix(N,1) == 1 && result_matrix(N,1) ==2
      FP = [FP 1]
   elseif result_matrix(N,1) == 2 && result_matrix(N,1) ==1
      FN = [FN 1]
   elseif result_matrix(N,1) == 2 && result_matrix(N,1) ==2
      TN = [TN 1]
   end
end
[z tp] = size(TP)
[z tn] = size(TN)
[z fp] = size(FP)
[z fn] = size(FN)

precisiion = tp /(tp+fp)
recall = tp/(tp+fn)
```

8.2.2.2 Displaying mutual information using colormap

```
A = table2array(data0)
```

```
B = table2array(MI)

B(B>1) = 1

X = [1:8472]
X = transpose(X)

MIwithIndex = [B X]
Zeros = zeros(8472,1)
scatter3(A(:,1), A(:,2),B,[10],B,'filled')

colormap(flipud(jet(500)))
zlabel('Mutual Information Score')
xlabel('X-axis')
ylabel('Y-axis')
zlim([0 1.5])
```

8.2.2.3 Plotting optimal sensor placement for each body parts

```
%% 1
A = table2array(data0)
B = Difference
Abs_array = table2array(Abs)
Bodyback_array = table2array(Bodyback)
Chest_array = table2array(Chest)
Leftcalf_array = table2array(leftlegbottom)
Leftthigh_array = table2array(leftlegtop)
Leftforearm_array = table2array(leftarmbottom)
Leftarm_array = table2array(leftarmtop)
Rightcalf_array = table2array(Rightlegbottom)
Rightthigh_array = table2array(Rightlegtop)
Rightforearm_array = table2array(Rightarmbottom)
Rightarm_array = table2array(Rightarmtop)

B = abs(B)
X = [1:8472]
X = transpose(X)

MIwithIndex = [B X]
%% 2
[a b] = size(Abs_array)
for N = 1: a
   values= MIwithIndex(Abs_array(N),:)
   Abs_MI_index(N,1) = values(1)
   Abs_MI_index(N,2) = values(2)

end

[a b] = size(Bodyback_array)
for N = 1: a
   values= MIwithIndex(Bodyback_array(N),:)
   Bodyback_MI_index(N,1) = values(1)
```

```
    Bodyback_MI_index(N,2) = values(2)

end

[a b] = size(Chest_array)
for N = 1: a
    values= MIwithIndex(Chest_array(N),:)
    Chest_MI_index(N,1) = values(1)
    Chest_MI_index(N,2) = values(2)

end
[a b] = size(Leftcalf_array)
for N = 1: a
    values= MIwithIndex(Leftcalf_array(N),:)
    leftcalf_MI_index(N,1) = values(1)
    leftcalf_MI_index(N,2) = values(2)

end
[a b] = size(Rightcalf_array)
for N = 1: a
    values= MIwithIndex(Rightcalf_array(N),:)
    rightcalf_MI_index(N,1) = values(1)
    rightcalf_MI_index(N,2) = values(2)

end

[a b] = size(Rightthigh_array)
for N = 1: a
    values= MIwithIndex(Rightthigh_array(N),:)
    rightthigh_MI_index(N,1) = values(1)
    rightthigh_MI_index(N,2) = values(2)

end

[a b] = size(Leftthigh_array)
for N = 1: a
    values= MIwithIndex(Leftthigh_array(N),:)
    leftthigh_MI_index(N,1) = values(1)
    leftthigh_MI_index(N,2) = values(2)

end
[a b] = size(Leftforearm_array)
for N = 1: a
    values= MIwithIndex(Leftforearm_array(N),:)
    leftforearm_MI_index(N,1) = values(1)
    leftforearm_MI_index(N,2) = values(2)

end
[a b] = size(Leftarm_array)
for N = 1: a
    values= MIwithIndex(Leftarm_array(N),:)
    leftarm_MI_index(N,1) = values(1)
    leftarm_MI_index(N,2) = values(2)
```

```
end

[a b] = size(Rightforearm_array)
for N = 1: a
   values= MIwithIndex(Rightforearm_array(N),:)
   rightforearm_MI_index(N,1) = values(1)
   rightforearm_MI_index(N,2) = values(2)

end
[a b] = size(Rightarm_array)
for N = 1: a
   values= MIwithIndex(Rightarm_array(N),:)
   rightarm_MI_index(N,1) = values(1)
   rightarm_MI_index(N,2) = values(2)

end
%% 3
sorted_MI_Abs = sortrows(Abs_MI_index,-1)
sorted_MI_Bodyback = sortrows(Bodyback_MI_index,-1)
sorted_MI_Chest = sortrows(Chest_MI_index,-1)
sorted_MI_leftcalf = sortrows(leftcalf_MI_index,-1)
sorted_MI_rightcalf = sortrows(rightcalf_MI_index,-1)
sorted_MI_leftthigh = sortrows(leftthigh_MI_index,-1)
sorted_MI_rightthigh = sortrows(rightthigh_MI_index,-1)
sorted_MI_leftforearm = sortrows(leftforearm_MI_index,-1)
sorted_MI_leftarm = sortrows(leftarm_MI_index,-1)
sorted_MI_rightforearm = sortrows(rightforearm_MI_index,-1)
sorted_MI_rightarm = sortrows(rightarm_MI_index,-1)
%% 4
scatter3(A(:,1),A(:,2),A(:,3),'.')
hold on


for N = 1  Index = sorted_MI_Abs(N,2)
   x = A(Index,1)
   y = A(Index,2)
   z = A(Index,3)
   scatter3(x,y,z,'fill','r')
   hold on
end

for N = 1  Index = sorted_MI_Bodyback(N,2)
   x = A(Index,1)
   y = A(Index,2)
   z = A(Index,3)
   scatter3(x,y,z,'fill','r')
   hold on
end

for N = 1  Index = sorted_MI_Chest(N,2)
   x = A(Index,1)
   y = A(Index,2)
   z = A(Index,3)
   scatter3(x,y,z,'fill','r')
```

```
    hold on
end

for N = 1  Index = sorted_MI_leftcalf(N,2)
    x = A(Index,1)
    y = A(Index,2)
    z = A(Index,3)
    scatter3(x,y,z,'fill','r')
    hold on
end
for N = 1  Index = sorted_MI_rightcalf(N,2)
    x = A(Index,1)
    y = A(Index,2)
    z = A(Index,3)
    scatter3(x,y,z,'fill','r')
    hold on
end
for N = 1  Index = sorted_MI_leftthigh(N,2)
    x = A(Index,1)
    y = A(Index,2)
    z = A(Index,3)
    scatter3(x,y,z,'fill','r')
    hold on
end
for N = 1  Index = sorted_MI_rightthigh(N,2)
    x = A(Index,1)
    y = A(Index,2)
    z = A(Index,3)
    scatter3(x,y,z,'fill','r')
    hold on
end
for N = 1  Index = sorted_MI_leftforearm(N,2)
    x = A(Index,1)
    y = A(Index,2)
    z = A(Index,3)
    scatter3(x,y,z,'fill','r')
    hold on
end
for N = 1  Index = sorted_MI_leftarm(N,2)
    x = A(Index,1)
    y = A(Index,2)
    z = A(Index,3)
    scatter3(x,y,z,'fill','r')
    hold on
end
for N = 1  Index = sorted_MI_rightforearm(N,2)
    x = A(Index,1)
    y = A(Index,2)
    z = A(Index,3)
    scatter3(x,y,z,'fill','r')
    hold on
end

for N = 1  Index = sorted_MI_rightarm(N,2)
```

```
    x = A(Index,1)
    y = A(Index,2)
    z = A(Index,3)
    scatter3(x,y,z,'fill','r')
    hold on
end
grid off
```