Signal Information Processing Tools for Healthcare Diagnostics

Minxiang Ye

Centre for Intelligent Dynamic Communications (CIDCOM)

Department of Electronic and Electrical Engineering

University of Strathclyde

This thesis is submitted for the degree of

Doctor of Philosophy

March 13, 2020

Declaration

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed: Minxiang Ye

Date: March 13, 2020

Acknowledgements

I would like to express my special appreciation and thank my supervisors, Dr. Lina Stankovic and Dr. Vladimir Stankovic, for the patient guidance, encouragement and advice they have provided throughout my PhD life. I would like to acknowledge Dr. Andrew Kerr, Mrs. Sylvia Moss and all volunteers involved in the experiments conducted at the University of Strathclyde and Brain and Spinal Injury Center (BASIC, Manchester, UK), for their thoughtful advice and help.

I would especially like to thank Prof. Samuel Cheung and Dr. Gene Cheung, with grateful collaboration experience in terms of both thoughtful research methodology and comments during my first secondment at Tongji University, Shanghai, China (Jul.-Oct. 2017) and second secondment at National Institute of Informatics, Tokyo, Japan (Mar.-Apr. 2018).

I wound also acknowledge the supports for my secondments in China and Japan by the Sensors and Intelligence in Built Environment (SENSIBLE) project funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734331.

I would gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

A special thanks to my family for all support and encouragement. Finally, I would also like to thank all friends for the support and friendship during both my life at University of Strathclyde and the secondments at Tongji University and National Institute of Informatics.

Abstract

The smart healthcare monitoring service has been given more attention in recent decades. With rising healthcare demand and progress in image processing, video-based gait assessment becomes a good alternative solution to assess the physical recovery progress for post-stroke survivors. However, most video-based assessment systems, commercially and in the literature, usually requires large laboratory space, are of high cost, and not portable, thus are impractical for in-home use. Accurate, low-cost, portable motion capture systems are growing in popularity, especially those that do not require expert knowledge to operate. This research proposes an alternative single depth camera based OPTIcal Kinematics Analysis system (named 'OPTIKA'). Novel signal processing and computer vision algorithms are proposed to determine motion patterns of interest from infrared and depth data, and enable real-time simultaneous tracking of joints based on attached retro-reflective ball markers. Specifically, an accurate trajectory-based gait phase classification system is proposed to facilitate the diagnostics of muscle activities during gait, using readings from low-cost motion capture systems 'OPTIKA'. Feature selection/extraction methods are proposed to enable an automatic segmentation of motion records into individual gait cycles with nine gait phases slice, which provides a more intuitive diagnostics experience for clinical therapists to analyze the rehabilitation progress associated to the kinematics in particular gait periods. This research also analyzes the sensitivity of feature selection/extraction methods against the classification performance in two healthcare monitoring applications. To overcome the limitations of high-cost training data labeling work and when parts of the training labels are noisy, a robust semi-supervised binary classifier is proposed to combine deep learning and graph based signal processing methods. The experiments demonstrate that given an acceptable proportion of noisy training labels, the proposed classifier outperforms several state-of-the-art classifiers. The overall concepts and systems presented in this thesis form an underlying approach for further video-based healthcare monitoring service that assists the diagnostics of physical rehabilitation.

Table of Contents

List of Abbreviations

ANFIS	Artificial Neural Fuzzy Inference Systems
ARX	Autoregressive Models with Exogenous Variables
ASIS	Anterior Superior Iliac Spine
AUC	Area under the Curve
BASIC	Brain and Spinal Injury Center
CNN	Convolution Neural Network
CRF	Conditional Random Fields
DBA	Dynamic Time Warping Barycenter Averaging
DBSCAN	Density-based Spatial Clustering of Applications with Noise
DML	Deep Metric Learning
DNN	Deep Neural Network
DT	Decision Tree
DTW	Dynamic Time Warping
DynGLR	Dynamic Graph Laplacian Regularization
ECG	Electrocardiogram
EEG	Electroencephalogram
EM	Expectation-Maximization
EMG	Electromyography
ERF	Enhanced Randomized Decision Forest
FFT	Fast Fourier Transform
FPR	False Positive Rate
GFT	Graph Fourier Transform

GLR	Graph Laplacian Regularization
GRU	Gated Recurrent Unit
GSP	Graph Signal Processing
HMM	Hidden Markov Model
HOS	Higher Order Statistics
IMUs	Inertial Measurement Units
IR	Infrared
KEEL	Knowledge Extraction based on Evolutionary Learning
KNN	K Nearest Neighbor
LDA	Linear Discriminant Analysis
LSTM	Long Short Term Memory
MEMM	Maximum Entropy Markov Model
MLP	Multilayer Perceptron
MS	Microsoft
NARX	Non-linear Autoregressive with Exogenous Inputs
NHS	National Health Service
NN	Neural Network
OE	Output Error Models
PCA	Principal Component Analysis
pHMM	Population Hidden Markov Model
PSIS	Posterior Superior Iliac Spine
QRS	Combination of Q,R,S Graphical Deflections Seen on Typical Electrocardiogram
RF	Random Decision Forest
RGB	Red-Green-Blue
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RRI	Interval of Two R-Wave Peaks
SDK	Software Development Kit

SHO	Shoulder
SVM	Support Vector Machines
SVM-RBF	Support Vector Machines with Radial Basis Function Kernel
TPR	True Positive Rate
ULBP	Uniform Local Binary Pattern

List of Symbols

S	Infrared image sequence
w	Blob detection threshold
r_b	Blob base radius
n	Number of markers used
b	Scan window length of the previous frames
S_d	Captured current infrared image
D_d	Captured current depth image
(p, q)	Marker centroid coordinates
r_m	Marker radius
r_r	Marker region radius
W	Max-Min width
D_0	Recovery resolution
D_1	Depth resolution (Sensor accuracy)
D_2	Distance resolution
m	Cluster mode
C	Number of calibration markers
γ	Level resolution
ξ	Range left clip rate
ϕ	Range right clip rate
τ	Local range length boundary
K	Number of gait phases
P	Gait phase label

List of Symbols

L	Length of frames in standardized gait cycle curve
λ_i	The i-th gait parameter
$V_j^z(f)$	Gait parameter value at frame f in j-th gait cycle curve for gait parameter z
$S_j^z(f)$	Gait parameter value at frame f in j-th standardized gait cycle curve for gait parameter z
G^z	The gait cycle curve cluster for gait parameter z
$\mathring{S}_m^z(f)$	Gait parameter value at frame f in the m-th gait cycle curve of cluster G^z for gait parameter z
$\bar{S}^z(f)$	Gait parameter value at frame f in the averaged curve $\bar{S}$ of cluster G^z for gait parameter z
$\bar{\omega}_m^z(x,y)$	Shared weight for frames x and y in m-th gait cycle curve of cluster G^z for gait parameter z
$P_{\bar{S}^z(f)}$	Gait phase label at frame f in the averaged curve $\bar{S}^z(f)$ of cluster G^z for gait parameter z
$P_{\mathring{S}_m^z(f)}$	Gait phase label at frame f in the m-th gait cycle curve $\mathring{S}$ of cluster G^z for gait parameter z
(u,v)	Feature pair that links two frames offsets by $l+uL$ and $l+vL$ from centroid frame f within an L-length sliding window
$\Re_l(u,v)$	The feature value linked by (u,v)
ϖ_f	Length of frames in sliding window centered at frame f
$Q_\varepsilon^z(P_a,P_b)$	Quality of ε-th feature candidate pair for gait parameter z to classify gait phase labels P_a, P_b
Ω	Total desired number of feature pairs for all gait parameters
$\mathcal{H}$	Combination of any two gait phase labels
r_z	Neighbor radius of two gait pattern curves for gait parameter z
ρ_f	Class probability vector at frame f
$\eta_f(a,b)$	Correlation coefficient of adjacent gait phase labels P_a, P_b at frame f
$\ddot{T}_p$	Median duration of gait phase in the training set

List of Symbols

$\mathbf{X}$	A set of observations
r	GLR iteration number
$\mathbf{Y}^r$	Labels corresponding to $\mathbf{X}$ at r-th GLR iteration
$\dot{\mathbf{X}}, \dot{\mathbf{Y}}^r$	A subset of $\mathbf{Y}^r$ corresponding to the training samples in $\dot{\mathbf{X}}$
$\mathbf{E} = \{e_{i,j}\}$	Binary matrix that represents the edge connectivity with each entry $e_{i,j}$ corresponding the edge connecting node i to node j
$\mathbf{A}$	Adjacency matrix
$\mathbf{D}$	Degree matrix
d_{max}	Maximum degree of node in $\mathbf{G}$
$\mathcal{V}^r(\cdot), \mathcal{C}^r(\cdot)$	Deep feature maps associated with r-th GLR iteration
$\mathcal{Z}_D(\cdot), \mathcal{Z}_{\mathcal{H}_\mathbf{U}}(\cdot)$	Shallow feature maps in $\mathcal{D}(\cdot)$ and $\mathcal{H}_\mathbf{U}(\cdot)$
$f^r(x)$	Observations associated with r-th GLR iteration
$g(x)$	Observations associated with graph update
x_a	A random node a selected from $\mathbf{X}$
x_p	A random node p selected from $\mathbf{X}$, with same label as x_a
x_n	A random node n selected from $\mathbf{X}$, with opposite label as x_a
$\mathbf{Q}$	A set of edges linked between nodes with opposite label
$\alpha_\mathbf{E}, \alpha_\mathbf{W}$	Minimum margin between deep metric based distances of $\mathbf{P}$ and $\mathbf{Q}$ edges for $Loss_\mathbf{E}$ and $Loss_\mathbf{W}$
$\mathbf{y}_i^1$	Encoded vector corresponding to the label y_i for node i
$\mathbf{y}_\mathcal{U}^1$	Encoded matrix corresponding to the labels for neighboring nodes
Θ	Activation function that estimates how much attention is paid on each edge
$\mathit{\Pi} = \{\pi_{i,j}\}$	Attention matrix with each entry $\pi_{i,j}$ corresponds to edge loss
$\pi_{a,p}, \pi_{a,n}$	Attentions on $\mathbf{P}$ and $\mathbf{Q}$ edges
κ	Conditional number
μ^r	Smoothness prior factor
β	Soft edge connectivity for graph update

List of Symbols

Γ	Maximum number of frames that are mislabeled around the gait event

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This chapter first overviews the state-of-the-art multimedia health monitoring applications in terms of complexity, portability, cost and highlights the ongoing challenges in enabling multimedia health monitoring for tele-rehabilitation and home-based solution, without the need for manual analysis by a clinical expert. Specifically, the focus of this thesis in designing a motion capture acquisition framework, feature selection and extraction of kinematics for gait phase classification, investigating the value of sequence labeling via hand-crafted feature selection vs machine-based feature selection as in deep learning networks, and finally designing a generic algorithmic framework for tackling the problem of noisy labels that can occur during pre-processing/labeling. The remaining parts outline the aims and objectives of this research, list contributions of this research, and introduce the organization of this thesis.

1.1 Overview

The radical shift in health monitoring services from expensive and large equipment ideal for large healthcare facilities, to portable and affordable technology that can be easily operated in the home setting becomes possible. For example, following a stroke, the recovery of physical functions such as walking, could be greatly enhanced by the intervention of a rehabilitation team focused on the identification and resolution of movement problems, typically through the practice of exercise tasks. A range of move-

ment abnormalities are periodically assessed to track and design rehabilitation progress for each individual patient [1]. As reported in [2], post-stroke survivors/patients are asked to complete a series of exercises and training modules after their conditions have been stabilized. The outcome of rehabilitation is generally improved if the patient receives a high intensity of practice combined with feedback on their movement to correct errors [3]. During their rehabilitation program, the recover progress is periodically assessed by evaluating their walking patterns based on kinematics measurements [4] which is called gait analysis. However, this ideal type of therapy is restricted by access to professional rehabilitation staff and equipment, a situation which has led to the growing importance of self-management strategies, including the use of home-use rehabilitation. Therefore, this research mainly focuses on the study of modern multimedia health monitoring techniques involved in gait assessment as they form the basis of further cost-effective home-use rehabilitation services.

Recent advances in the use of video-based motion capture methodologies, muscle actions have been described as the joint movements that are usually captured by motion capture systems [5] using spherical objects attached on the body called "markers". However, these systems, such as [6], [7], are usually based on multiple cameras/sensors, used to construct a 3D scene and track the high-precision positions of all markers in the 3D space, at the cost of high expense, require operational expertise and large laboratory space. To provide the service to patients who do not have access to these facilities, simple marker-based or markerless tracking systems [8] using multiple or single Red-Green-Blue (RGB) camera have been investigated in the literature [9]. However, these systems have limitations, such as requiring operational expertise and time-consuming processing [10], and requirement for a specific color of the underlying cloths, such as the single RGB camera systems of [11], [12] and multiple RGB camera systems, such as [13]. An attractive alternative is to use Microsoft (MS) Kinect sensor [14] with its own software development kit (SDK) capable of tracking 25 skeleton joints. However, as demonstrated in [15], Kinect's skeleton results tend to be too noisy and not reliable for clinical applications. Therefore, this research investigates the possibility of incorporating the marker-based motion tracking approaches [6] into a single MS Kinect

sensor to mitigate the limitations of gait analysis with low portability, complex setup procedure but still is capable of measuring high accuracy kinematics compared to the state-of-the-art VICON system.

In gait analysis literature [16], based on those captured joint trajectories that record the gait motion by the motion capture system, the physiotherapist perform further diagnostics of muscle activates based on those calculated kinematics measurements such as step/stride length and knee angle changes in each gait phase. To facilitate these diagnostics, this research investigates to possibility of developing a method that automatically extracts the gait phase information given joint trajectories in the gait record. This task in the gait analysis literature [17] is usually formulated as classification problem that assign a categorical label (gait phase information) to each frame within the gait record given joint trajectories data. The quality of feature selection and effectiveness of feature selection for specific pathological classification in distinguishing whether a subject is abnormal, have a significant impact on gait assessment methods [18]. Therefore, this is especially critical for systems relying on inexpensive motion capture systems, which need to demonstrate a comparable level of accuracy to clinical systems [19]. Lie group features, investigated in many action recognition applications [20], [21], have proved to be a strong feature for classifying actions. However, only long-time actions are investigated in the literature (see review paper [22] and references therein), whereas the proposed frame-wise solution often suffers from temporal misalignment of features. Population Hidden Markov Model (pHMM) is proposed in [23] to obtain a fixed length of silhouettes within a gait cycle for aligning temporal features; however, the algorithm requires high-quality silhouette extraction, reconstruction, and additional training. Therefore, this research aims to provide an insight of alternative feature selection/extraction methods and classification systems to achieve a high-precision gait phase extraction performance.

The results of using kinematics parameters like the knee angle used in [4] proves that feature selection method is effective to extract the gait phase information which is one of the sequence labeling problem that assigns a categorical label to a given sequential data. Therefore, from a research point of view, the question of “whether feature

selection/extraction method is sensitivity to other healthcare sequence labeling applications, such as Electrocardiogram (ECG)-based arrhythmia classification [24]". The motivation of conducting this case study for detecting cardiac disease is that there is an emerging trend of deep learning methods like [25] that enables an automatic feature/selection process through its classifier learning to assign a heartbeat type given a segment of ECG record. Similar to gait phase classification, several state-of-the-art feature extraction methods that extract linear and nonlinear features [26] are proposed to train a machine learning model for classifying beats. However, it is hard to fairly distinguish/evaluate/analyze the importance and sensitivities of feature selection/extraction methods used in both gait assessment and cardiac disease detection applications as different data settings are used in the experiments. However, before we can definitely conclude whether feature selection/extraction is a must for healthcare signal classification, we attempt to answer whether feature selection/extraction is sensitive to the healthcare application signal labeling sequence by analyzing the value of our proposed feature selection/extraction approach for another popular healthcare application, namely Electrocardiogram signal based arrhythmia classification.

Another aspect towards practical classification performance is the quality of labels in the training data, which usually is time-consuming to generate and requires professional knowledge, especially for labeling gait phase information which has a potential risk of mislabeling. Supervised and semi-supervised deep learning techniques have shown excellent performance for feature extraction and classification tasks [27], but are particularly sensitive to the quality of the training dataset, since they tend to overfit the models when learning from incorrect labels [28]–[30]. Conventional approaches to overcome model overfitting, based on various regularization techniques, e.g., l_1- or l_2-norm penalty on weights [31], dropout [32], batch normalization [33], skip-connections [34], [35] etc., are not effective in mitigating the effects of incorrect labels. These approaches can be grouped into methods based on: (a) inserting additional "trusted" labels, e.g., [36], [37], and (b) loss function correction to penalize classifier learning from training samples with relative low weights, e.g., [38], [39], [40]. In [36], a probabilistic model is integrated into the DNN to correct noisy labels. Similarly, in [37], a loss correction tech-

nique is introduced to mitigate the unreliability of noisy training labels. However, these methods require clean data to prevent the models from drifting away. A robust loss function that is less sensitive to label outliers is introduced in [38]. The results indicate that it achieves the state-of-the-art predictive performance against noisy training labels by reweighting the training samples that trades convexity for boundedness and remains constant for strongly negative values of its argument. A combination of training labels and predicted labels is used in [39] to avoid directly modeling the noisy training labels, but requires pre-training to achieve good results. [40] adds another softmax layer to further augment the correction model via a noise transition matrix, which is hard to estimate in practice, especially in multi-class classification problems. Graph signal processing (GSP) provides an alternative way to restore the noisy training labels using a graph signal smoothness prior [41]. The advances in using deep learning and GSP have been making this problem possible to tackle, however can only be performed and evaluated given a known fixed graph structured data [42]. Therefore, in this research, alternative approach that incorporates GSP into deep learning based classifier learning is investigated to mitigate the over-fitting problem without extra clean training labels.

The aims and objectives of this research are listed in the following section to overcome all main drawbacks of current healthcare monitoring techniques and classification methods for gait assessment and cardiac disease detection.

1.2 Aims and Objectives

The main aim of this research is to explore alternative video-based and home-use health monitoring techniques to facilitate physical rehabilitation monitoring for post-stroke survivors, that are cost-effective, portable and easy-to-use. Specifically, the post-stroke survivors are usually asked to do the walking tests systematically during the physiotherapy, their recovery progression is assessed based on the correlations between muscle activities and kinematic measurements during different gait phases within their gait records, such as knee angle [16], step length [43]. Therefore, the extraction of gait phase information for a gait record is one of the major tasks for this assessment which

is also an aim of this research. To facilitate the automation of gait phase information extraction, this research aims to develop effective feature selection and extraction methods based on the joint trajectories collected in the gait records. Following an emerging trend of deep learning techniques for sequence labeling problem, this research evaluates the importance and sensitivity of feature selection and extraction methods in gait assessment. This research also aims to expand the above evaluation in a wider application field, such as cardiac disease detection. To tackle the "noisy label" that occurred in the training dataset caused by the mislabeling of gait phases, this research also aims to improve the robustness of classification.

The main research objectives are listed below:

1. Develop a cost-effective, portable and easy-to-use motion capture framework for gait assessment to facilitate rehabilitation for post-stroke survivors, suitable for local clinics and home use, that do not require any clinical expertise to operate. The proposed high accuracy imaging methodology should achieve comparable kinematics measurements with the state-of-the-art optical motion capture system VICON [6].This comparison is performed by focusing on the following metrics, common for gait analysis, namely stride length and stance, swing duration and finally ensuring a less than 5% statistically significant difference.

2. Develop effective feature selection and extraction methods to improve the predictive performance of gait phase classification, so as to facilitate the kinematics analysis of muscle activation in individual phases during walking.

3. Investigate the importance and sensitivity of feature selection and extraction methods in health monitoring applications, using the state-of-the art feature selection/extraction methods and classifiers.

4. Develop a generic deep learning based classification method that is robust to noisy training labels, without providing extra clean data and prior knowledge; investigate the importance of each component in the architecture via an ablation study and evaluate the effects of its hyper-parameters.

1.3 Contributions of the Thesis

The major contributions of this thesis to achieve the research objectives can be summarized as:

1. To reduce the cost and operation space to assess the gait during the physical rehabilitation for post-stroke survivors, this research develops a *high accuracy marker-based single-camera imaging methodology* based motion capture system (see "OPTIKA" in Appendix. B) that enables real-time tracking of joint movements, where only single infrared-based (IR-based) depth camera is used (30 frame/sec for Kinect v2 [14]).

    1.1. To simplify the system setup, this research proposes an *automatic scene calibration procedure* to collect necessary subjective information that can be used multiple times, without requiring operational expertise.

    1.2. To obtain comparable kinematics measurements with the state-of-the-art VICON system [6], this research proposes a *depth maps recovery methodology* that simultaneously detects and tracks the attached retro-reflective markers in 3D space using adaptive thresholding.

2. To facilitate the diagnostic of post-stroke survivors' muscle activities in different gait phases, this research proposes a trajectory-based gait phase classification method to slice the gait sequence into different slices that enables the kinematics analysis with gait phase information.

    2.1. To better characterize gait patterns, this research proposes a *gait pattern extraction methodology* to reduce the amount of duplicated/similar gait patterns in the training data without sacrificing the classification accuracy. This research uses gait phase information in addition to kinematics, including gait trajectories, as is usual practice.

    2.2. To address the problem of feature mis-alignment in time, *e.g.*, across frames, this research proposes a trajectory based *gait feature extraction* method to

obtain a global optimal set of feature pairs as inputs for the gait phase classification problem.

2.3. To reduce the high computational complexity used in the filtering-based feature extraction method without sacrificing the classification performance, this research proposes an optimized feature extraction method that is computationally efficient.

2.4. To solve mis-classification of the periodic sequential gait phase labels, this research proposes a *gait phase reconstruction* method to refine the predictions at post-processing stage.

3. To investigate the question of "if a good classification performance can be obtained by feeding the raw features into robust classifiers or handcrafted feature generation/selection is necessary", this research fairly analyzes the importance and sensitivities of feature selection and extractions used in sequence labeling problem for healthcare monitoring applications, such as gait phase and arrhythmic classification tasks. Using the state-of-the-art feature selection/extraction methods with an appropriate experimental settings, this first-person research in the literature provides a perspective of the importance and cost trade-off between time and accuracy in using feature selection and extraction methods in the above two classification tasks.

4. To further improve the robustness of classifier against the "noisy" training labels caused by mislabeled the ground-truth data, this thesis provides a fundamental research of developing a framework to perform robust classifier learning without using extra effort in providing clean training data. The validation results prove that the proposed framework is capable of learning robust classifier for three binary classification datasets and a multi-class gait phase classification dataset.

4.1. To avoid over-fitting the classifier, this research proposes a *graph-based regularized loss function* that incorporates attention mechanism to regularize the convolution neural network (CNN) for deep feature learning.

4.2. To improve the reliability of classifier learning against the effects of noisy

training labels, this research develops a *graph-based semi-supervised classifier* to iteratively perform both online denoising and classification at the same time.

4.3. To assign the degree of freedom for graph connectivity learning that is robust to noisy label, this research introduces a *graph update procedure* to better reflect the node-to-node correlation based on convolution of iteratively updated edges, without prior knowledge of the underlying graph structure.

1.4 Organization of the Thesis

The rest of this thesis consists of four contribution chapters and one conclusion chapter. Each contribution chapter is structured with a related work pertaining to that chapter, methodology, results, discussion and a summary of research contributions. The contribution chapters 2-5 present contributions 1-4, respectively.

Chapter 2 - this chapter first reviews state-of-the-art motion capture methodologies used for gait analysis. Specifically, this thesis focuses on developing an alternative cost-effective, portable and easy-use video-based motion capture system. This chapter also highlights the challenges of using a single depth camera to accurately detect, locate and track the retro-reflective markers in 3D space and how these challenges are addressed by the proposed algorithms. The developed software system 'OPTIKA' is then used for data acquisition in "4-meter walking test" conducted during two different physical rehabilitation sessions for both post-stroke survivors and healthy volunteers. This chapter is based on the work that appeared in 2015 International Conference on Image Analysis and Processing [44], 2015 IEEE Global Conference on Signal and Information Processing [45] and IEEE Journal of Selected Topics in Signal Processing [46]. This chapter address contributions 1-4.

Chapter 3 - this chapter proposes features for better representing gait motions during each gait phases. A feature extraction method is proposed to further improve the predictive performance for accurately segment gait records into gait cycles of individual gait phases, based on the data acquired by 'OPTIKA'. This chapter is based on the

work that appeared in 2017 IEEE International Conference on Multimedia and Expo [47] and the work to appear in IEEE Transactions on Multimedia [48]. This chapter address contributions 5-7.

Chapter 4 - this chapter gives a review of state-of-the-art feature selection and extraction methods for sequence labeling in two healthcare monitoring applications, gait phase and arrhythmic classification. Using different classification methods, the importance and sensitivity of these feature engineering techniques are evaluated, with respect to the predictive performance and complexity of the classifier models. This chapter address contribution 8.

Chapter 5 - this chapter presents a review of state-of-the-art machine learning techniques to learn robust classification model from noisy labeled data. To migrate the effects of noisy labels, this chapter proposes an end-to-end trainable network for semi-supervised binary classification. Incorporating the graph signal processing with convolutional neural network, an iterative graph update procedure is proposed to better reflect the correlation between different samples. This chapter is based on the work that appeared in 2019 IEEE International Conference on Acoustics, Speech and Signal Processing [49]. This chapter address contributions 9-11.

Chapter 6 - this chapter presents conclusions of this research, highlights the remaining challenges and gives insights into the future research directions for home-use healthcare monitoring techniques and classification methods.

1.5 Ethics Approval

The data collection procedure performed using single imaging sensor for gait analysis application has passed ethical committee in National Health Service, UK and University of Strathclyde. See Appendix A for details of the pilot protocol.

1.6 Associated Publications

1.6.1 Book chapter

1. M. Ye, V. Stankovic, L. Stankovic and G. Cheung "Graph Spectral Image Classification", book chapter in "Graph Spectral Image Processing", submitted to ISTE, Feb. 2020. *Role: data interpretation, development of graph signal processing and deep learning based robust classifier learning for image data, chapter writing*

1.6.2 Journal articles

1. M. Ye, C. Yang, V. Stankovic, L. Stankovic and A. Kerr, "A depth camera motion analysis framework for tele-rehabilitation: Motion capture and person-centric kinematics analysis", IEEE Journal of Selected Topics in Signal Processing, vol.10, no. 5, pp. 877-887, Aug. 2016.
   Role: main concept design for the single depth camera motion capture system, data acquisition for stroke survivors during two different rehabilitation sessions at Brain and Spinal Injury Center (BASIC), Manchester, data acquisition for healthy volunteers during two different sessions at University of Strathclyde, system development of software system 'OPTIKA', data analysis and interpretation, paper writing

2. M. Ye, V. Stankovic, L. Stankovic and S. Cheng, "Distinct Feature Extraction for Video-based Gait Phase Classification", IEEE Transactions on Multimedia. doi: 10.1109/TMM.2019.2942479
   Role: main concept design of gait pattern extraction and distinct feature extraction methods, data analysis and interpretation of gait data and proposed algorithms, paper writing

3. M. Ye, V. Stankovic, L. Stankovic, S. Lulic, A. Anderla and S. Sladojevic, "Feature Selection and Extraction in Sequence Labeling for Arrhythmia Detection", Manuscript was submitted to Elsevier Pattern Recognition Letters, Mar. 2020.

Role: data analysis and interpretation of deep learning architectures, paper writing

4. M. Ye, V. Stankovic, L. Stankovic and G. Cheung, "Robust Deep Graph Based Learning For Binary Classification". Manuscript was submitted to IEEE Transactions on Signal and Information Processing over Networks, Mar. 2020.
   Role: main concept design of joint graph signal processing and deep learning for robust classifier learning, data analysis and interpretation of proposed concept and deep learning architectures, paper writing

1.6.3 Conference papers

1. M. Ye, C. Yang, V. Stankovic, L. Stankovic, and A. Kerr, "Gait analysis using a single depth camera", IEEE Global Conference on Signal and Information Processing, Orlando, FL, Dec. 2015.
   Role: main concept design of single camera gait analysis, data acquisition for stroke survivors during two different rehabilitation sessions at BASIC, Manchester, system development of software system 'OPTIKA', data analysis and interpretation for evaluating the tracking accuracy of 'OPTIKA', paper writing

2. M. Ye, C. Yang, V. Stankovic, L. Stankovic, and A. Kerr, "Kinematics analysis multimedia system for rehabilitation", Workshop on Image and Video Processing for Quality of Multimedia Experience, Genova, Italy, Sep. 2015.
   Role: main idea to develop a single depth camera based motion capture system for kinematics analysis, data acquisition during two different rehabilitation sessions at BASIC, Manchester, system development of 'OPTIKA', data analysis and interpretation for evaluating the tracking accuracy of 'OPTIKA', paper writing

3. M. Ye, C. Yang, V. Stankovic, L. Stankovic and S. Cheng, "Gait phase classification for in-home gait assessment", IEEE International Conference on Multimedia and Expo, Hong Kong, China, Jul. 2017. (Finalist of World's First 10K Best Paper Award)

Role: main concept design of gait phase classification algorithms, data analysis and interpretation, paper writing

4. M. Ye, V. Stankovic, L. Stankovic and G. Cheung, "Deep graph regularized learning for binary classification", International Conference on Acoustics, Speech, and Signal Processing, Brighton, United Kingdom, May. 2019.
   Role: main concept design of using graph signal processing to regularize deep learning based classifier learning, data interpretation, paper writing

Chapter 2

Motion Capture in Depth Videos

2.1 Review of Motion Capture Methodologies for Rehabilitation

Most motion analysis systems used for rehabilitation consist of multiple wearable sensors (e.g., passive/active optical markers, Electromyography (EMG) /Electroencephalogram (EEG) / ECG, inertial sensors, force plates), and require large laboratory space, are of high cost, and not portable, thus are impractical for flexible, mobile clinical and home-use rehabilitation programs [50]. Optical motion analysis systems are attractive; however, current marker-based and marker-less, single or multiple IR/RGB camera motion analysis systems have limitations, such as dependency on the underlying fabric color, time-consuming process, lack of portability and/or high price, such as VICON [6], single RGB camera systems of [4], [8], [11], [51] and multiple RGB camera systems, such as [13]. Inertial tracker-based systems, like Xsens MVN BIOMECH [52] and M3D force plate [53], are options for large clinics or hospitals, but are not suitable for small clinics and home use.

Alternatively, single RGB-depth camera systems, such as [14], [54], [55], after significant technological advances, have become cheap and popular options. For example, MS Kinect enables tracking of human joints in 3D space using a single camera and its SDK via skeleton tracking [14]. However, Kinect's skeleton data are too noisy (see,

e.g., Fig.1 in [56],[57]), and do not provide sufficient accuracy [5], [9], [15], [56]. Using two Kinect sensors, as in [58], can potentially improve the accuracy, but at the expense of portability, required expertise, and ease of setup. To investigate the reliability of lower limb motion tracking using MS Kinect v2 and its SDK, a simple case study is conducted by placing a single MS Kinect v2 camera in the sagittal view and asking a subject to walk normally from left to right within the camera view (mirrored). Using the same inputs of infrared and depth images acquired by the camera, the first column is obtained by the proposed system 'OPTIKA' and the second column is obtained by using MS Kinect SDK. It is shown that using only MS Kinect v2 and its SDK are not accurate enough to perform the motion tracking task, such as potential miss-tracking the positions of feet, knee and waist shown in Fig. 2.1.

The marker-less Kinect-based approach of [59], for performing the 'Get Up and Go Test', which is part of the larger Tinetti test to identify subjects at risk of falling, is based on the construction of the background depth frame, which enables background removal, followed by frontal pose analysis to get body structure parameters and the sagittal view joint trajectory estimation. The method does not achieve clinical accuracy showing an error of up to 15 pixels compared to the reference trajectory. Six joints are tracked in the sagittal plane; the foot joint was not tracked, and it is not expected to work well due to interference with the floor. A similar approach [60] uses RGB and depth images of MS Kinect for semi-automatic postural and spinal analysis using Dynamic Time Warping, pose estimation and gesture recognition. The algorithm requires substantial manual effort, operation expertise and is time-consuming, hence not suitable for real-time application. [61] uses Kinect's depth images to perform 3D pose estimation with high computational complexity and is unsuitable for real-time processing (30 frames/second, same as MS Kinect v2). [62] relies on Kinect SDK's virtual skeleton of the body and supervised learning to extract positions of the joints of interest in a gait analysis application, but is limited by high computational complexity, need for training data, and presents no scientific evidence that the proposed methods are clinically accurate.

This research develops a general framework to facilitate the next generation of

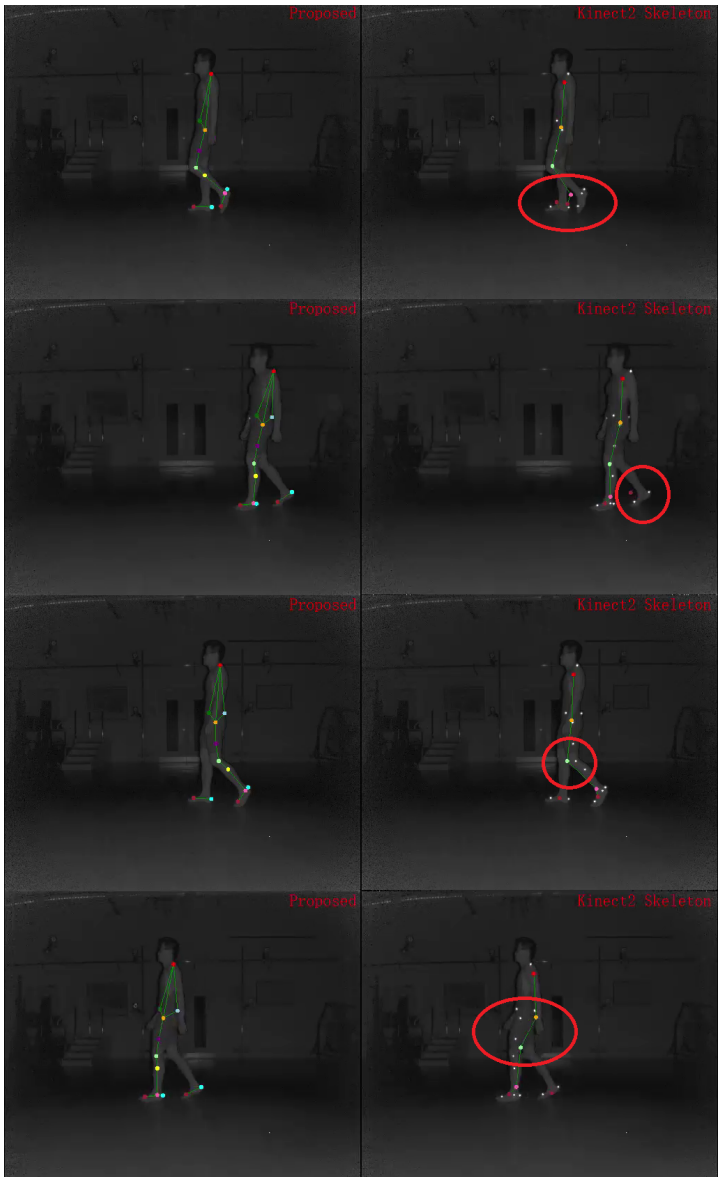


Figure 2.1: Sagittal View Motion Capture using 'OPTIKA' and MS Kinect v2. The captured joints in infrared images are shown in unique colors. Comparing the results of the tracking results from 'OPTIKA' and Kinect v2 shown in the first and second columns, this figure demonstrates the tracking failures of **front panel foot**, **back panel foot**, **knee joint**, **waist** from the first to last row, respectively.

portable and cost-effective tele-rehabilitation applications, suitable for local clinics and home use, that do not require any clinical expertise to operate. The proposed framework combines high accuracy marker-based tracking methodology based on IR sensing and portability and affordability of range imaging methodology using structured light or

Time-of-Flight sensors. The proposed kinematics framework is capable of building various motion analysis assessment tools that target different rehabilitation applications. In contrast to previous work [59] and [60], the proposed framework is benchmarked against the state-of-the-art optical motion system VICON [6] for gait analysis using the walk forward and back test, with 6 markers on each sagittal plane (left and right) to capture both sagittal planes during the walking test in one go, and most importantly, create a person-centric subject model to define the geometric relationship between different markers. Since 2D measurements are nonlinear due to the fish eye effect from the sensor lens, as opposed to [4], [51], [59], the proposed framework maps markers in 3D space that enables 3D motion tracking. Note that as reported in [63], camera calibration for a single MS Kinect v2 is insignificant in terms of depth measurement accuracy.

2.2 Overview of the Proposed Framework

The proposed framework comprises an optical motion capture system and kinematics analysis tools that enable secondary development for solution enhancement. The interconnection among the underlying algorithms and key parameters used in the algorithms are listed in this section.

The optical motion capture system (described in Sec.2.3) consists of a single depth camera (both IR and depth images are used) that enables creating 3D optical motion reconstruction. It is designed to capture human motion in real time by detecting retroreflective markers attached to joints of interest, and comprises three modules: (1) Data cleaning - for cleaning IR and depth images (described in Sec.2.3.1). (2) Detection - for tracking markers in image space (Sec.2.3.2). (3) Mapping - for recovering the markers' position in camera space through the proposed cluster location algorithm (Sec.2.3.3).

The proposed kinematics analysis tools are developed as an application solution that sits on the proposed motion capture system, facilitating portable, indoor rehabilitation diagnosis, as demonstrated by the gait analysis application in Fig. 2.2 and Sec.2.4. Autonomously located markers attached to subject's joints during the straight-line

walking exercise, are used to automatically calculate gait associated parameters commonly used for clinical assessment, such as joint angles, velocity, movement patterns, gait cycle phase, step and stride length, swing and stance phase, etc. [64]. In particular, the gait analysis tools (Sec.2.4) comprise: (1) Scene calibration (Sec.2.4.1), (2) Subject modeling (Sec.2.4.2) for building a person-specific body segmentation model, (3) Kinematics analysis module (Sec.2.4.3) for calculating gait analysis parameters based on the proposed analytics.

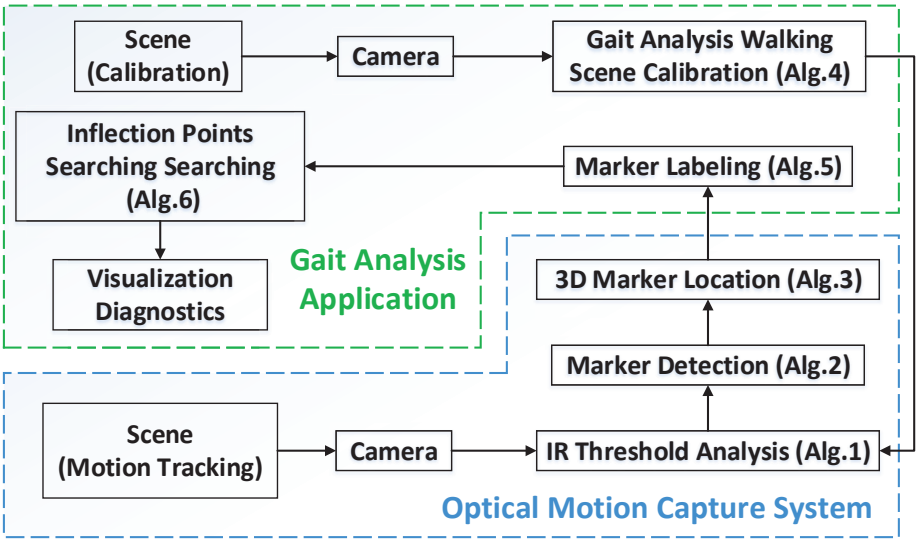


Figure 2.2: Overall proposed system structure diagram. It consists of an *optical motion capture system* that can be independent from applications, such as *gait analysis application* investigated in this research. Both optical motion capture system and gait analysis application rely on MS Kinect v2 camera which captures infrared and depth images, from the scene where human motion tracking will be performed and the same scene with only calibration markers, respectively. The output of an initial IR threshold for detecting the blob(marker) in the infrared image is used by optical motion capture system to adaptively locate the marker. After optical motion capture system locates the markers, gait analysis application labels and visualizes the markers, enables further kinematics diagnostics for gait analysis task, such as measuring step/stride length and stance/swing duration.

During processing of the IR images, it is observed that motion blur and light conditions strongly influence the speed and accuracy of marker tracking. MS Kinect v2 determines the depth by measuring the time emitted light takes from the camera to the object and back [65]. Since the retro-reflective markers interrupt the emitted light, the

depth measurements of those targets are missing. The only way to recover the depth value for each marker is to use their depth values. Three algorithms are proposed to address the above problems: (1) Threshold analysis (Alg.1) − extending previous work in [44] to solve fast motion and camera noise during marker detection, (2) Marker detection (Alg.2) − the idea is to improve the marker centroid location accuracy and speed which are attached to joints of interest, in image space, (3) Depth recovery and mapping (Alg.3) − the 3D texture is partially missing in the marker region and it is possible to use the point cloud histograms for restoring the depth value of the marker centroid. By looking at the point cloud histograms, one can get a kernel that has higher weight inside according to their Euclidean distance to the marker centroid and frequency of occurrence.

Once the coordinates of the markers in 3D space have been obtained above, the aim of the gait analysis application is to label or associate the markers to joints on the human subject, so that joint angles can be calculated during kinematics analysis. In order to do so, the first step is Scene calibration (Alg.4) whose purpose is to map the physical measurements of the physical experimental environment into a virtual environment, recreating a geometric relationship between the camera, calibration markers, and walking start/end points. This enables Marker labeling (Alg.5), where a person-centric subject model is constructed to map the subject's physical dimensions to virtual 3D space building a geometric relationship among markers on the body and hence markers can be accurately labeled as belonging to the foot joint, hip, etc. Once all the markers have been labeled, kinematics analysis commences in Gait event detection (Alg.6) by examining the relative trajectories of knee, ankle, and heel markers to the floor (obtained during scene calibration and marker labeling) to find inflection points and local peaks for gait events detection without pre-smoothing the data, in order to get the best accuracy, including addressing occluded markers.

Table 2.1 lists key parameters used in the proposed algorithms, and which values were observed to give the best results when trading accuracy and execution time. These pre-set values were selected based on the MS Kinect 2 sensor, a walking line distance to the camera of 2.5m-3m and sensor height of about 0.8m from the floor, resulting in

Table 2.1: Parameters used in Algs. 1–6

Param.	Alg.	Description	How is it set?
S	1,4	Captured infrared image sequence	Measured by the sensor
w	1	Blob detection threshold	initialized in Alg. 4 and updated in Alg. 1
r_b		Blob base radius	Calculated by Alg. 4
n		Number of markers used	Application specific (12 in the experiments)
b		Scan window length of the previous frames	3 frames (heuristically)
S_d	2	Captured current infrared image	Measured by the sensor
D_d		Captured current depth image	Measured by the sensor
(p, q)	3,5	Marker centroid coordinates	Calculated by Alg. 2
r_m		Marker radius	
r_r		Marker region radius	
W	3	Max-Min width	50 mm (heuristically)
D_0		Recovery resolution	2 pixels (heuristically)
D_1		Depth resolution (Sensor accuracy)	5 mm (heuristically)
D_2		Distance resolution	0.5 mm (heuristically)
m		Cluster mode	Application specific
C	4	Number of calibration markers	Application specific
γ	6	Level resolution	0.05 (heuristically)
ξ		Range left clip rate	0.03 (heuristically)
ϕ		Range right clip rate	0.03 (heuristically)
τ		Local range length boundary	3 (heuristically)

an approximate 4m walking line (see examples in Appendix. B). If the camera sensor and the latter distances are changed, a standard calibration procedure (e.g., [4], [66]) can be used to find optimal values for D_0, D_1 and D_2.

2.3 Optical Motion Capture System

The task of the proposed optical motion capture system is to simultaneously track multiple retro-reflective markers using a single IR depth camera, irrespective of the overlying motion analysis application. Retro-reflective materials were chosen since they introduce high intensity regions into IR images and blank holes into depth images. Therefore, the markers are detected on IR images, after which the marker location is recovered in the depth image and mapped to camera space via the following key steps: (1) Data cleaning - cleaning invalid data and reducing sensor noise, (2) Marker

detection - detecting markers in IR image space using connected component algorithm [67] with scene dependent adaptive thresholds, (3) 3D marker location - recovering marker depth values using cluster location algorithm (in Sec.2.3.3) and mapping depth space coordinates to the camera space using the depth map projection method in [68]. Each step is elaborated in the following subsections.

2.3.1 Video Acquisition

The primary source of noise affecting the captured IR images is from the camera lens of either IR transmitters or receivers and interfering sources such as metallic materials, retro-reflective materials, etc. Reflective materials other than markers will influence the measurements and constitute interference while recovering depth values. Fig.2.3 shows the noise, originating from the imaging sensor and reflective material, typically encountered in an acquired frame.

Figure 2.3: Noise from the Kinect sensor (yellow) and reflective material (red)

Approaches for denoising include depth map denoising, either spatially with, e.g., adaptive total variation [69], nonlocal graph-based transform with group sparsity [70] and layer-based depth correction and completion [71], or temporally with, e.g., parametric model-based nonlocal means [72] and joint-bilateral filter [73]. Such data cleaning approaches would potentially preserve sharp edges without over-smoothing, and improve the accuracy of marker tracking. Aiming for real-time applications, a simpler, intuitive and less complex, but effective approach is used based on Kalman filtering

[74]. Namely, since the initial locations of interfering materials can be detected in the first frame and corresponding pixel values, it is easy to predict their next state using Kalman filter, and exclude them from further processing.

2.3.2 Detection

After cleaning the frame from unwanted noise, IR images are converted to binary format in order to detect and identify markers via blob detection on a frame-by-frame basis. Since all retro-reflective marker regions have clearly distingushable pixel values in IR images from surounding regions, blob detection is a natural object detection choice.

There are several approaches to detect and identify blobs, such as matched filters / template matching [44], watershed detection [75], structure tensor analysis followed by hypothesis testing of gradient directions [76], [77], scale-space analysis [78]. All these approaches are limited by their sensitivity to noise, structure restriction and complexity [79]. In [44], a concentric cycle-based method (template matching) is proposed to perform the shape fitting test for each potential blob in order to locate all markers in image space (2D); however, this method is time consuming and requires expertise to determine associated parameters for the shape fitter and the kernel cluster filter, and cannot locate the center of the marker correctly when motion blur occurs and the marker is out of the sagittal plane, which leads to center deviation on those markers with circular distributed IR values.

To solve this problem and satisfy the real-time processing constraints, an enhanced heuristic IR analysis algorithm is proposed in Alg.1, where the threshold value is adaptively acquired for blob detection in the next frame. A sequence of b previous IR images and an initial threshold for blob detection w and blob base radius r_b, obtained by Alg.4 during the scene calibration process (which is application and scene dependent), are fed to Alg.1. Note that the initial threshold w in Alg.1 has little influence on the accuracy of the adaptive IR threshold. As expected, the further away the value of w from the optimal value, the higher the number of iterations to find a suitable threshold, resulting in longer execution time. Note that it stops iterating when the number of detected blobs f reaches convergence, i.e., the value of f between iterations is unchanged.

The main idea behind Alg.1 is to first assign w, for the current frame S_d to that used in one of the b previous frames, which results in the number of blobs in S_d closest to the actual number of markers n present in the scene. If this threshold detects more than n blobs (that is, some detected blobs are not markers), w is calculated by averaging the pixels from the n most significant blobs weighted by the their radius. Otherwise, if some markers were missed, a weighted average is taken over all detected blobs in the current and b previous frames.

After blob detection threshold is set, the connected component labeling algorithm [67], a classic blob extraction method used to detected connected regions in a binary image, is used to detect markers from the located blobs. Then, for each detected marker it finds a centroid, radius and region radius using simple pixel-based geometry. The overall proposed algorithm is detailed in Alg.2.

2.3.3 3D marker location

Once all blobs have been detected as valid markers, the next step is to obtain the coordinates of the markers in 3D space. In general, a depth camera has intrinsic parameters to perform spatial mapping from image space to camera space.

The depth-map projection method of [68] is adopted to acquire undistorted camera space coordinates of the tracked markers after marker centroids have been located. However, depth information within the marker region is empty due to the retro-reflective nature of the attached markers. Therefore, to recover the sensitive pixels around each marker region in the depth images, Alg.3 calculates the image histograms with respect to pixel intensity (Steps 21 to 28 in Alg.3) and distance to the marker centroid in IR images (Steps 29-32). The algorithm is executed for each detected marker. The following parameters are assigned heuristically to improve the recovery accuracy and are constant for all frames: Max-Min width, $W = 50$, recovery resolution, $D_0 = 2$, histogram depth resolution, $D_1 = 5$, histogram distance resolution, $D_2 = 0.5$.

Alg.3 tackles the problem of partial occlusion: the input to the algorithm is a cluster mode variable, m that can take 3 possible discrete values: (1) Normal - no occlusion for the marker, (2) Top - occlusion present in front of the marker, (3) Bottom - occlusion

Algorithm 1: Adaptive blob detection threshold setting for the next frame

Input: Captured image sequence from the sensor, S;
Initial blob detection threshold, w from Alg.4;
Blob base radius, r_b from Alg.4;
Number of markers used, n;
Scan window of b previous frames $S_{d-b}, \ldots, S_{d-1}$, with their blob detection thresholds $e_j, j = d-b, \ldots, d-1$;

Output: Blob detection threshold for the next frame, e_d;

1 **acquire** next IR frame S_d from S;
2 **set** $e_d = e_{j^*}$, where $j^* = \arg\min_{j=d-b,\ldots,d-1} |n - f_j|$, where f_j is the number of blobs detected in Frame S_d when blob threshold e_j from Frame S_j is used;
3 **set** f as the number of detected blobs when e_d is used on Frame S_d;
4 **order** the detected blobs into the descending order of the blob radius: $i_1^d, \ldots, i_n^d, \ldots, i_f^d$, where d denotes the frame number;
5 **set** $J_q^d, q = 1, \ldots, f$ as a matrix of all IR pixel values in Blob i_q^d, k_q^d their mean value, l_q^d and u_q^d as radius and blob region radius of Blob i_q^d in S_d, respectively;
6 **if** $f > n$ **then**
7 **calculate** new e_d by averaging IR pixel values from $J_1^d, \ldots, J_n^d$ weighted by $l_1^d, \ldots, l_n^d$;
8 **else if** $f < n$ **then**
9 **set** h_0 by averaging IR pixel values from $J_1^d, \ldots, J_f^d$ weighted by $l_1^d/r_b, \ldots, l_f^d/r_b$
10 **set** h_1 by averaging IR pixel values from $J_1^q, q = d-b, \ldots, d-1$ weighted by e_q/w
11 **set** h_2 by averaging blob radius from u_1^q, $q = d-b, \ldots, d-1$ weighted by e_q/w $e_d \Leftarrow (h_0 * f/n + h_1 * h_2/r_b + w)/(f/n + h_2/r_b + 1)$;
12 **if** $f_{last} \neq f_{current} \neq n$ **then**
13 **add** S_d to scan window when using e_d **and goto** 4;
14 **return** e_d;

at the back of the marker. Assume that markers are placed on the body as shown in Fig. 2.4, partial occlusion takes place on markers attached to the anterior superior iliac spine (ASIS), posterior superior iliac spine (PSIS), hip and femur during arm swing. Those markers are in the bottom mode, while heel, toe, shoulder markers are in the top mode and the remaining markers are always in the normal mode in Alg.3 since they are never occluded.

Algorithm 2: Marker detection

Input: Captured IR image frame, S_d;
IR blob detection threshold, e_d obtained by Alg.1;

Output: Marker centroid, $(p,q)_1,\ldots,(p,q)_n$ where n is the number of detected markers;
Marker radius, $r_{m_1},\ldots,r_{m_n}$;
Marker region radius, $r_{r_1},\ldots,r_{r_n}$;

1 Use connected component labeling [67] on S_d with e_d for IR-to-binary image conversion and obtain labeled markers $M_1,\ldots,M_n$;
2 **foreach** *marker* M_i *in* $M_1,\ldots,M_n$ **do**
3 **set** ϱ as the number of pixels in M_i;
4 **set** g as the sum of all IR pixel values in M_i;
5 **set** $v = g - \varrho * e_d$ as normalized sum of IR values;
6 **let** $(p_i,q_i) = (0,0)$, $r_{m_i} = 0, r_{r_i} = 0$ be M_i's centroid, radius and region radius, respectively;
7 **foreach** *pixel* $P_{x,y}$ *in* M_i **do**
8 $p_i = p_i + x * P_{x,y}/v$; $q_i = q_i + y * P_{x,y}/v$;
9 **foreach** *pixel* $P_{x,y}$ *in* M_i *with coordinates* (x,y) **do**
10 **set** pixel distance $l = \sqrt{((x-p_i)^2 + (y-q_i)^2)}$;
11 $r_{m_i} = r_{m_i} + l * P_{x,y}/v$;
12 **if** $r_{m_i} > r_{r_i}$ **then** $r_{r_i} = r_{m_i}$;
13 **return** $(p,q)_1,\ldots,(p,q)_n$, $r_{m_1},\ldots,r_{m_n}$, $r_{r_1},\ldots,r_{r_n}$;

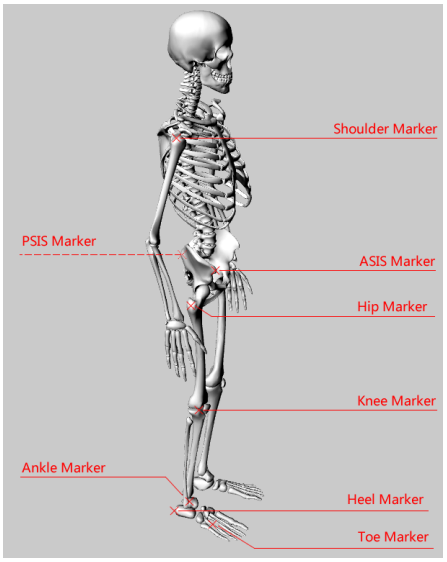


Figure 2.4: Example of Marker Placement for Capturing Lower-limb Motion.

Algorithm 3: 3D marker location

Input: Captured depth image frame, D_d;
Marker centroid, (p, q), Marker radius r_m and region radius r_r obtained by Alg.2;
Max-Min width, W;
Recovery resolution, D_0; Depth resolution, D_1; Distance resolution, D_2;
Cluster mode, m [defined in Sec. 2.3.3];

Output: Marker position in frame D_d, (x, y, z);

acquire depth values V^d at rectangle region of {left:$p - r_r - D_0$, top: $q - r_r - D_0$, right: $p + r_r + D_0$, bottom: $q + r_r + D_0$} in D_d;
order pixels in V^d in the increasing order $v^d(1), \ldots, v^d(N)$, and **set** $\kappa = 0$ and $z = 0$;
set $\Lambda_0 = v^d(1) + W$;
set V_0 as a vector of all depth values in V^d smaller than Λ_0 and the remaining values as V_2 , and **set** $V_1 = V_0$;
if ***sizeof***$(V_0) > 2$ **then**
    let $\kappa = \kappa + 1$, $V_0 = V^d \setminus V_0$ and **goto** 5;
else if ***sizeof*** $(V_0) = 1$ **then**
    set $\Lambda_1 = v^d(N) + W$; **set** V_1 as all depth values in V^d smaller than Λ_1 and the remaining values as V_2; **goto** 6 when **sizeof**(V_2) >**sizeof**$(V_1) + \kappa$, otherwise **goto** 13;
else
    if $m =$ *normal* **then**
        set $T_0 =$**min**(V_0) and $T_1 = T_0 + W$;
    else if $m =$*top* **then**
        set $T_0 =$**min**(V_1) and $T_1 = T_0 + W$;
    else if $m =$*bottom* **then**
        set $T_1 =$**max**(V_2) and $T_0 = T_1 - W$;
set H_0 as the histogram of pixels in V^d that fall between T_0 and T_1, with depth resolution D_1;
foreach *bin* h *in* H_0 **do**
    if ***sizeof***$(h) <$***min***$((r + D_0)^2, d^2/$***sizeof***$(H_0) + D_0)$ **then**
        if ***sizeof***$(h) <$***min***$(D_0,$***sizeof***$(v^d)/$***sizeof***$(H_0))$ **then**
            remove h from H_0;
foreach h *in* H_0 **do**
    histogram all pixels in h w.r.t their distance to centroid (p, q), with bin resolution D_2;
    set $\epsilon(h)$ as the mean value of the bin in h that has the highest count;
foreach h *in* H_0 **do**
    set $z = z + \epsilon(h)*$**sizeof**$(h)/$**sizeof**(H_0);
return (x, y, z) mapped from (p, q, z) using [68];

2.4 Gait Analysis Application

This section describes the proposed application-specific algorithms that interface with the proposed motion capture system (see Sec.2.3). The proposed gait analysis application, comprising scene dependent calibration, person-centric modeling, and kinematics analysis, enables autonomous, high-accuracy processing of gait associated data. Each of the three algorithms are explained next.

2.4.1 Straight-line Walking Scene Calibration

The purpose of scene calibration is to collect scene dimensions to build a geometric relationship between the camera, calibration markers, and walking start/end points.

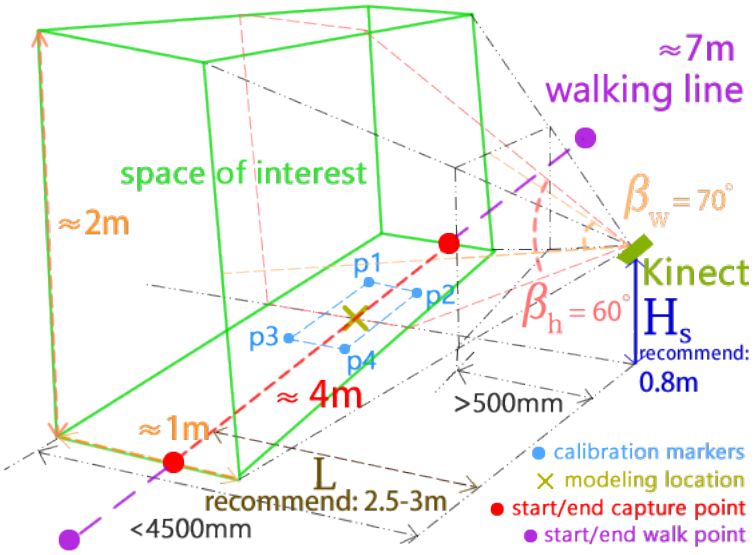


Figure 2.5: Virtual straight walking exercise scene [44].

A typical straight-line walking exercise scene captured by the camera, is represented as a virtual trapezoidal cylindrical model in Fig.2.5. The plane defined by 4 optical (calibration) markers, shown as blue dots in Fig.2.5, placed on the ground is perpendicular to the plane defined by the camera and the ground. An example of an IR image captured during the calibration is shown in Fig.2.6, where the start and end of walking are shown as red dots. Following [66] which reports an acceptable average depth

measurements error of 2-4 mm when objects are $2-3$m away from the MS Kinect v2 camera, in this research, a walking line distance of $2.5-3$m to the camera and the sensor height of 0.8m from the floor is used to ensure an approximate 4m walking line for the 4-meter walking test.

Figure 2.6: Scene calibration: Calibration markers in green are labeled (sagittal view). Walking line is defined between the start and end points, shown in red.

The overall walking scene calibration process is summarized in Alg.4. $C = 4$ calibration markers are placed on the ground one-by-one. The scene calibration process continuously searches and analyzes the status of the calibration marker plane in relation to the camera to ensure perpendicularity, and reports marker status as: (1) Uninitialized - stop mode, (2) Move Left/Right - camera needs to be moved to the left or right, (3) Tilt Down/Up, (4) Pan Left/Right, (5) Replace Markers - critical noise detected or marker placement error, (6) Done - calibration completed. Steps 12-15 perform manual adjustment of the camera pose.

Threshold w for blob detection (used in Alg.1) is calculated by first forming a histogram of edge pixels for each detected blob, and then finding the minimum (over all four marker blobs) of the largest histogram bin (Step 12). Alg.4 relies on subtracting the background to label the calibration markers and calls Alg.2, with updated w set to the minimal pixel value in the detected blob, to obtain the calibration marker's centroid and corresponding blob radius. Base blob radius r_b is set as the mean radius

of all calibration markers. Alg.4 determines the start and end points of the walking exercise, which are then physically marked on the floor using a tape to help the post-stroke for identification of the walking direction (see Appendix. B).

Algorithm 4: Gait Analysis Walking Scene Calibration

Input: Captured IR image sequence from the sensor, S;
Number of calibration markers, C;
Output: Blob centroids, $(p,q,z)_1,\ldots,(p,q,z)_C$;
Start and end walking point, $(r,s)_0,(r,s)_1$;
Initial IR blob threshold, w;
Blob base radius, r_b;
Walking line length, L;

1 **set** the number of labeled markers $c=0$;
2 **while** $c \leq C$ **do**
3 **repeat acquire** the next IR image from S;
4 **apply** frame subtraction detection;
5 **until** no significant motion detected;
6 **apply** frame subtraction detection using as background the previous frame with no motion detected;
7 **if** *blob detected* **then**
8 **update** markers' state using marker labeling (call Alg.2 with e_d set to the min IR value in the marker blob), and **let** $c=c+1$;
9 **calculate** a histogram of edge pixel values for each blob, and **set** w as the minimum, over all blobs, of the most significant bin.
10 **check** diagonal connection condition for $(p,q,z)_1,\ldots,(p,q,z)_C$ mapped using normal mode Alg.3 with current depth image from S;
11 **if** *connection is intersectant* **then**
12 **report** plane status defined by $(p,q)_1,\ldots,(p,q)_C$ relative to camera;
13 **else**
14 **report** critical error and **goto** 1;
15 **adjust** camera's pose according to the reported status;
16 **set** r_b =**mean**$\{r_1,\ldots,r_C\}$, where $\{r_1,\ldots,r_C\}$ are obtained by Alg.2 called in Step 8 above;
17 **def** start/end points $(r,s)_0,(r,s)_1$ relative to center of $(p,q)_1,\ldots,(p,q)_C$ during streaming with guideline tool;
18 **calculate** the distance between $(r,s)_0$ and $(r,s)_1$ in camera space as L;
19 **return** IR base threshold w, blob base radius, r_b walking line length L and visualize start and end points $(r,s)_0,(r,s)_1$ in IR/RGB stream.

2.4.2 Gait Model

Following calibration of the experimental environment, a unique complete subject model for sagittal gait analysis is constructed for every individual subject by physically measuring the subject standing at the location shown as X in Fig.2.5, specifically measuring $H0^*$, $H7^*$, and $W3^*$ to $W8^*$ (as shown in Fig.2.7) after all markers have been mapped in 3D space. The model is clustered into three parts: upper body, limb and foot models shown in Fig.2.7. For each frame, the model comprises the following: (i) position of all detected markers, (2) geometric relationship between markers, (3) virtual lines L13-L16 relative to the marker positions.

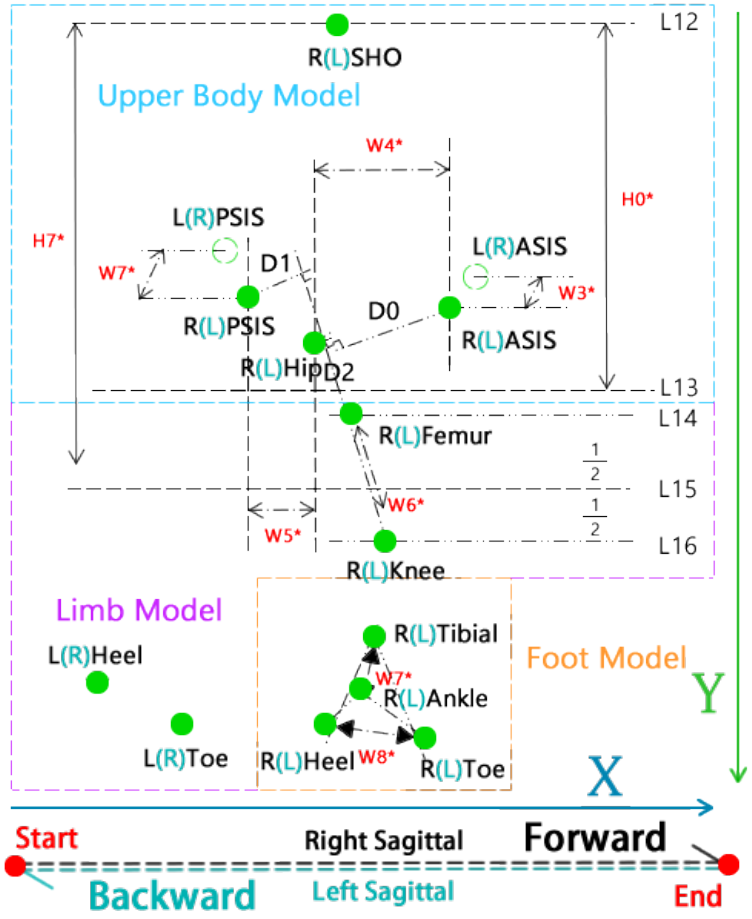


Figure 2.7: Sagittal Model. 12 visible markers are marked with green circles. 2 partial invisible markers are shown in circle outlines. 'R' ('L') denotes right (left) marker. For example, RPSIS is the right posterior superior iliac spine marker [44].

Each marker is labeled by examining all potential marker groups for upper body, limb and foot models using Alg.5, scanning each IR frame from left-top to right-bottom. In particular, the shoulder (SHO) marker is first chosen as the top marker in the first frame that shows the subject in the sagittal plane and is labeled within the region around virtual line $L12$, predicted by Kalman filtering [74] using the marker position in the previous frames and its velocity. Then, all relevant distances (see Fig.2.7) are updated using the subject model of the previous/reference frame in order to solve model matching errors due to complete occlusion occurences on hip and femur markers. Once geometric relationships (distances, locations) relative to virtual lines $L12 - L16$ are determined, geometric relationships between all marker combinations will be checked. For example, the upper body model marker group should satisfy $D0 > D1 > D2$, $X_{LPSIS} < X_{RPSIS} < X_{RHIP} < X_{RASIS} < X_{LASIS}$, and the ankle marker of the foot model should be inside the triangle region defined by tibial, toe, heel markers. Potential clusters are formed by calculating the distances between the markers in the cluster and comparing them with the updated distances $W3^*$ to $W8^*$.

Since the geometric location relationship of the limb model markers is changing along the Y axis during leg swing, six markers (Tibial, Ankle, two Heels, two Toes) are selected on the bottom of the model along the Y axis and determine the two heel and toe markers attached to the occluded body side, by their relative position to the knee and other visible markers. Finally, the marker name/position is determined by comparing the distances between the markers in each cluster relative to the updated distances $W3^*$ to $W8^*$ for each validated marker cluster across the upper body, limb and foot models.

2.4.3 Kinematics Analysis

Once all the markers have been labeled, kinematics analysis commences, closely following the relative joint angle and gait cycle definitions from [64]. The trajectories of the heel markers to the floor are examined to detect the stance and swing gait phases as the boundary event is when the heel marker contract the floor.

If a marker is occluded (full marker occlusion happens occasionally on the hip and

Algorithm 5: Marker labeling

Input: From Alg.2:
Centroids for n markers, $(p,q)_1,\ldots,(p,q)_n$;
Markers' radius, $r_{m_1},\ldots,r_{m_n}$;
Markers' region radius, $r_{r_1},\ldots,r_{r_n}$;
Marker positions in the previous frame, F;

Output: labeled/named markers

predict SHO marker from F using Kalman Filter [74];
if *SHO not found* **then**
  set centroid of the predicted region as SHO marker with radius and region radius as in F;
calculate all W's and L's values shown in Fig.2.7 using the current model (see Subsection 2.4.2);
order all markers in the region of L12 and L13 by X-coordinate;
determine the most-likely marker cluster for upper body based on D0, D1, D2 (see Subsection 2.4.2);
order markers under L13 by Y, and X afterwards.
divide lower limb markers into two clusters by evaluating 6 markers nearest to the ground by testing all possible clusters for the triangle foot model.
combine markers on the other side of the body into the triangle foot model in the upper limb region according to Y-coordinates;
determine the other side's foot position by checking its relative position with knee and foot marker;
map labeled $(p,q)_1,\ldots,(p,q)_n$ using Alg.3 with $r_{m_1},\ldots,r_{m_n}$ and $r_{r_1},\ldots,r_{r_n}$;
return labeled/named markers;

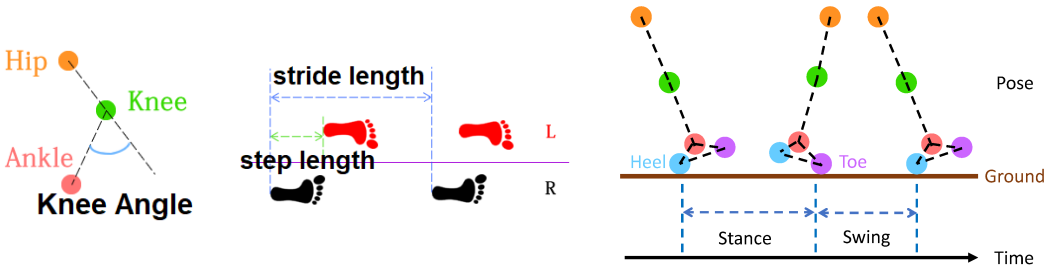


Figure 2.8: Knee angle, step and stride length, stance and swing duration. Stance and swing phases consist of the period that a foot is on the ground and in the air, respectively.

femur markers caused by the swing arm), the 2nd or 4th cubic Bezier curve interpolation [80] are adopted according to the occlusion length. The same curve interpolation is also used for marker trajectory resampling (from 30 fps to 100 fps) to obtain more samples for measuring gait associated data (and also for benchmarking with the 100fps state-

of-the-art VICON). The knee angle, step and stride length, stance and swing phase as shown in Fig. 2.8 are measured based on the resampled trajectories of heel, ankle, knee and hip markers as explained next.

Step and Stride Length

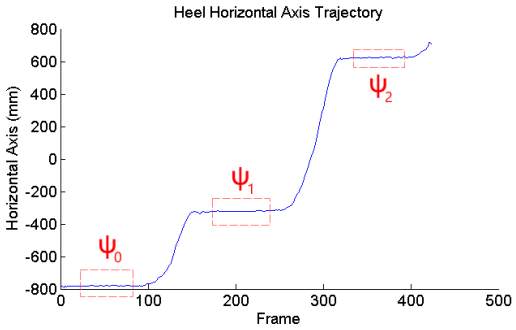


Figure 2.9: Heel Horizontal Axis [44]

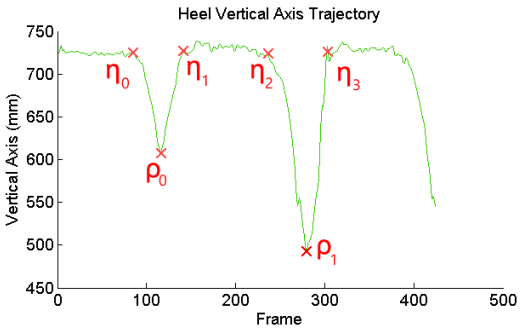


Figure 2.10: Heel Vertical Axis [44]

This task can be simplified into extracting stable values, where the heel marker trajectory horizontal axis value does not change over a window of frames (see an example in Fig.2.9, where ψ_0, ψ_1, ψ_2 denote three windows with no change detected using window matching between the inflection points). These points correspond to heel strikes to the floor. Once the left and right heel's horizontal stable values are found, the step and stride length can be calculated using the adjacent stable values over time, i.e., as $\psi_{i+1} - \psi_i$.

Gait Phases Detection

Gait events of heel strike and toe off are used to measure the stance and swing duration using heel marker trajectory vertical axis values. An example is shown in Fig.2.10, where η_0, η_1, η_2, η_3 denote inflection points, ρ_0, ρ_1 refer to local extremes. The proposed algorithm quantizes the heel marker vertical axis trajectory and then searches each quantization region between the inflection points from the global minimum to the maximum by iteratively regrouping the scanned points. The quantization step-size (level resolution), $\gamma = 0.05$, the range left and right clip rate, $\xi = \phi = 0.33$, and the local range length boundary, $\tau = 3$ are heuristically set for extracting the inflection

points and local peaks in order to obtain the relative time of heel strike and toe off through angle variation between the floor line and the line from toe to heel. A boolean variable 'locked' is used in Alg.6 as a flag for each range between two consecutive inflection points. For a given range p, in Step 18, $\min(p)$ and $\max(p)$'s lock levels denote the (local) minima (maxima) below (above) range p.

Algorithm 6: Inflection Points Searching

Input: Vertical coordinate of heel marker trajectory T (e.g., see y-axis on Fig.2.10)
Level resolution, γ;
Range left clip rate, ξ;
Range right clip rate, ϕ;
Local range length boundary, τ;

Output: Inflection points ηs;
Local peaks ρs;

1 **quantize** T using a step size γ into quantization levels Ω;
2 **def** range pool P as an empty set;
3 **set** $\sigma = \min(T)$
4 **for** $k = \min(\Omega)$ *to* $\max(\Omega)$ **do**
5 **find** all ranges, i.e., differences between two values in T that fall within the quantization bin k and are smaller than σ; **foreach** *found range* $f = [f_l, f_r]$ **do**
6 **set** locked(f_l) = false, locked(f_r) = false;
7 **if** f *insides* P **then**
8 **set** $T = T \setminus f$ and **goto** 5;
9 **if** *locked(f_l) == true* ***or*** *locked(f_r) == true* **then**
10 **update** P with f;
11 **else**
12 **if** *f_l's rate change* $< \xi$ **then**
13 **set** locked(f_l) = true and **goto** 6;
14 **if** *f_r's rate change* $< \phi$ **then**
15 **set** locked(f_r) = true and **goto** 6;
16 **foreach** p *in* P **do**
17 **if** *(locked(p_l) == true* ***and*** *locked(p_r) == true)* ***or*** *p is start* ***or*** *end range with one side locked* **then**
18 **if** *length(p)* $> \tau$ ***and*** $\min$(*p's lock levels*) $< 2 * \max$(*p's lock levels)−p's initial local peak* **then**
19 **return** p's boundary values as inflection points η and initial local peak as relevant local peak;

2.5 Visualization & Results

The proposed framework is demonstrated using an MS Kinect v2 sensor [14], though other similar sensors can be investigated as further case studies, e.g., commercially available MS Kinect v1 [81], Intel RealSense R200 [55], SoftKinetic DepthSense Cameras [54]. The MS Kinect v2 sensor outputs 16-bit 512x424 pixel resolution of IR and depth images at 30fps. A user-friendly interface is developed as part of 'OPTIKA' software in this work for the proposed underlying framework (see Sec.B).

Fig.2.11 shows the snapshot of the software, which shows how convenient it is to access the recorded experiments by selecting the tracker tool. Users can also view the automatic reconstruction process within the proposed multimedia application or manually playback the whole experiment. Autonomous analysis is performed and gait associated parameters are generated afterwards. These data (including joint angles, movement patterns, gait phases, step and stride length, swing and stance duration) can be easily accessed within the analyzer toolbox. For the rehabilitation application, a diagnostics interface is developed to report the patient's condition.

The proposed gait analysis application, and inherently the proposed framework and its six algorithms, was tested using 92 experiments with 14 subjects (11 males and 3 females), including 9 stroke survivors, and 25633 frames. The percentage of difference between the proposed system and VICON system in measuring knee angle α, step length ζ, stride length ξ, stance and swing duration is used to evaluate the motion tracking accuracy.

Evaluation of the proposed adaptive threshold analysis algorithm, Alg. 1, on an Intel i7-4710HQ 2.5GHz CPU, Windows10 operating system, implemented by Visual C++, against the static-threshold marker detection algorithm of [44], [45] indicates higher detection accuracy as shown in Table 2.2. Note that the results are averaged over all 92 experiments based on the ground truth of marker positions that are manually labeled clearly. However, it introduces an extra preprocessing step which increases the processing time by roughly 1ms per frame. In addition, as the distance between the camera and marker increases, the number of pixels in the captured IR images increases

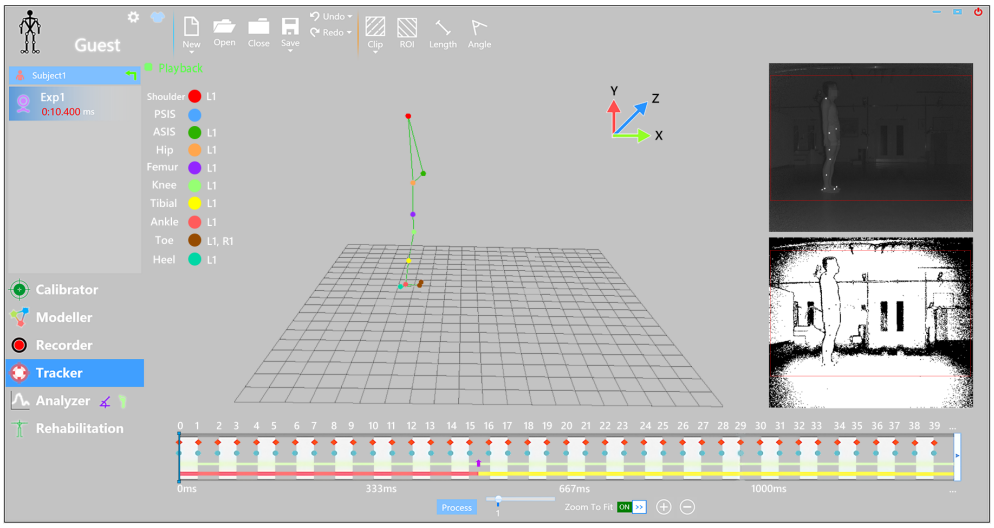


Figure 2.11: Multi-view tracker snapshot.

which requires more time to search for the regions of retro-reflective marker in Alg.1. This distance is also dependent on subject body dimensions and their walking direction. However, as will be shown next, the proposed blob detection threshold analysis algorithm simplifies the following processing steps, making the overall processing faster.

Table 2.2: Performance of the proposed dynamic vs static thresholding for marker detection, showing percentage of successfully detected markers and mean execution time per frame.

Algorithms	Detected (%)	Time (ms/frame)
Static Threshold [44]	91.44 ± 3.52	0.15 ± 0.04
Adaptive Threshold	98.08 ± 1.08	1.21 ± 0.33

The performance of the proposed marker identification algorithm, Alg.2, is evaluated for each marker using recall rate of marker centroid's distance error (within error reference $\beta = 0.5, 1.5, 2.5$ pixels), that is, the average number of frames where the distance between the detected marker centroid and its true position is within β. Fig.2.12 shows the recall rate increments of 8 experiments for 12 markers using the proposed adaptive threshold based algorithm and the algorithm of [44] when $\beta = 0.5$ pixel. It can be seen that the recall rate has been improved by about 3∼9% especially for those markers that are attached to feet (ankle, toe, heel, another foot's toe and heel) where out-of-plane, motion blur are most likely to occur.

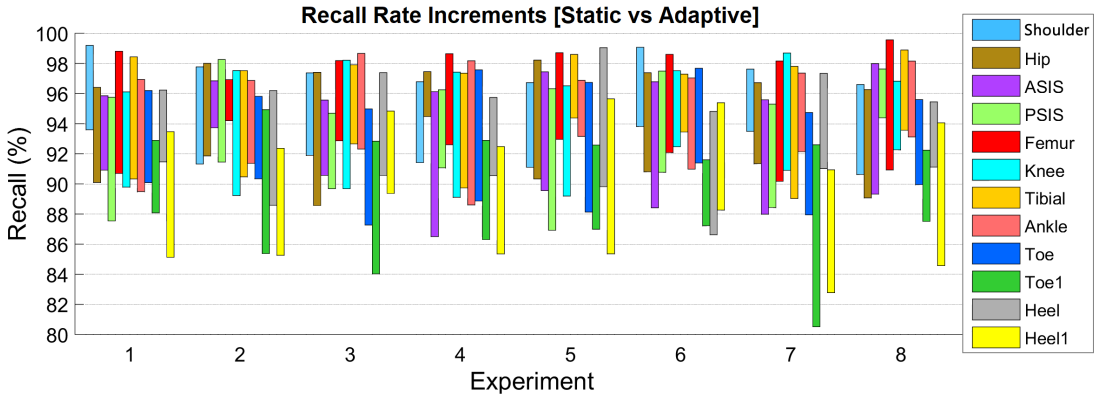


Figure 2.12: Recall Rate Increments when $\beta = 0.5$ pixel for 8 of experiments. Top boundary of each incremental rectangle is the recall rate using the proposed adaptive threshold based algorithm and bottom boundary is the one using the algorithm of [44].

Averaged results over all experiments and all markers are shown in Tab.2.3. It can be seen that the proposed Alg.2 significantly outperforms the static threshold approach in terms of both accuracy and the overall execution time.

Table 2.3: Performance of the proposed adaptive threshold based algorithm to locate center of a marker averaged over all markers and 92 experiments.

Characteristics	Adaptive	Static[44]
Aver. callback (%), $\beta = 0.5$	95.86 ± 1.64	88.12 ± 2.61
Aver. callback (%), $\beta = 1.5$	97.32 ± 1.75	90.84 ± 2.67
Aver. callback (%), $\beta = 2.5$	98.04 ± 1.73	92.10 ± 2.45
Time (ms/frame)	7.72 ± 1.16	117.53 ± 17.94

Evaluation of accuracy of detecting gait events is performed by manually selecting the key frames and examining (with expert knowledge) the whole IR image sequence with corresponding static point clouds captured during the experiments, which are used as reference (i.e., ground truth), for validating the step and stride length, stance and swing duration. In order to evaluate the performance of swing phase detection, results were averaged to obtain the mean percentage error and percentage standard deviation in Table 2.4. It can be seen from the table, that the mean and standard deviation of the error are very small and slightly decreased with the proposed adaptive threshold based system compared to that of [44], attributed to the proposed Alg.6.

The evaluation of the overall proposed system using VICON as a benchmark is

Table 2.4: Performance of the two methods for measuring step and stride length, and stance and swing phase

Metric	Step	Stride	Stance	Swing
Mean(%)[44]	1.06	1.16	1.81	1.12
Std(%)[44]	5.31	4.72	5.78	4.24
Mean(%)	0.98	1.08	1.73	1.09
Std(%)	4.12	4.23	4.56	3.56

discussed next. 5 subjects were simultaneously recorded using VICON and the proposed motion capture system. Each subject walked from left to right and back 8 times, hence a total of 40 experiments are used for comparison. The knee angle results of 4 experiments from 4 different healthy subjects are shown in Fig.2.13. Following [82] that uses statistical significance (5%) to evaluate the difference between VICON systems with different camera setups, in this research, it can be seen that the angles from the proposed system are well within the 5% error margin compared to VICON.

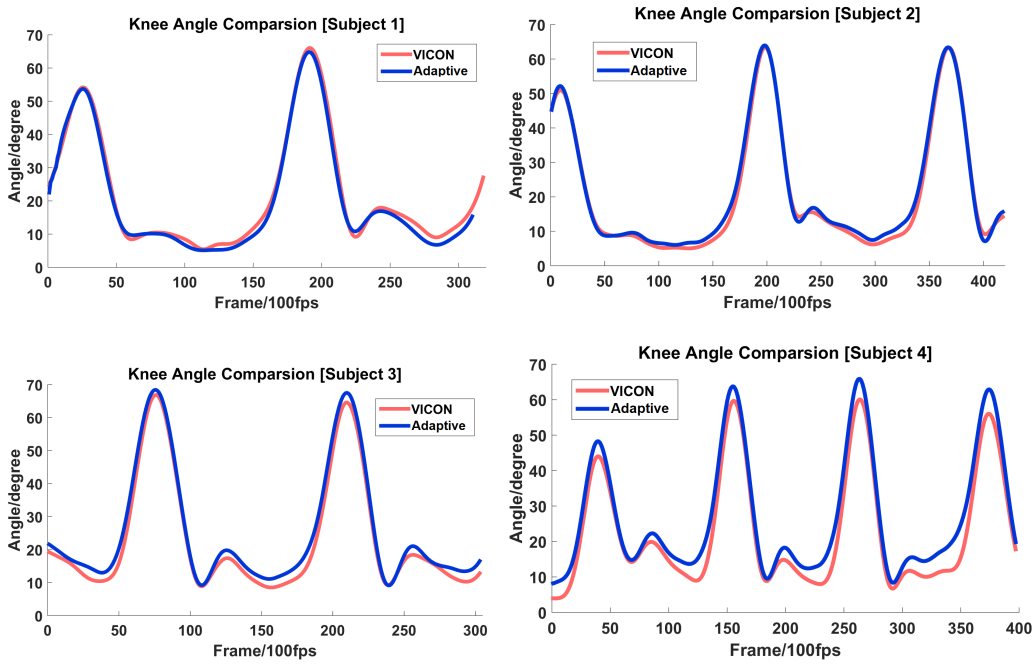


Figure 2.13: Knee angle comparison with VICON.

The knee angle measurement performance is next evaluated by calculating root-mean-square error (RMSE) for each of the 40 experiments between the proposed frame-

work and VICON. This is compared with the RMSE calculated from the system of [44] with VICON. The RMSE results are shown in Fig.2.14 as RMSE per experiment, where the effect of the algorithmic improvements over the static threshold based system is clearly illustrated by reduced RMSE for all 40 experiments. Lower RMSE for the proposed system is attributed to the adaptive thresholding and the improved marker detection/labeling method. The maximum RMSE with the proposed system was under 6 degrees. Note that VICON returns joint trajectories instead of marker trajectories, thus a potential error comes from the misalignment between marker positions and actual joints.

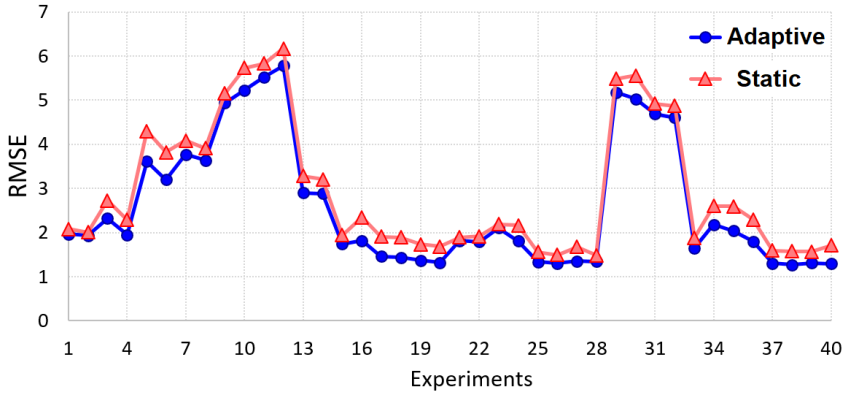


Figure 2.14: Knee angle RMSE over 40 experiments.

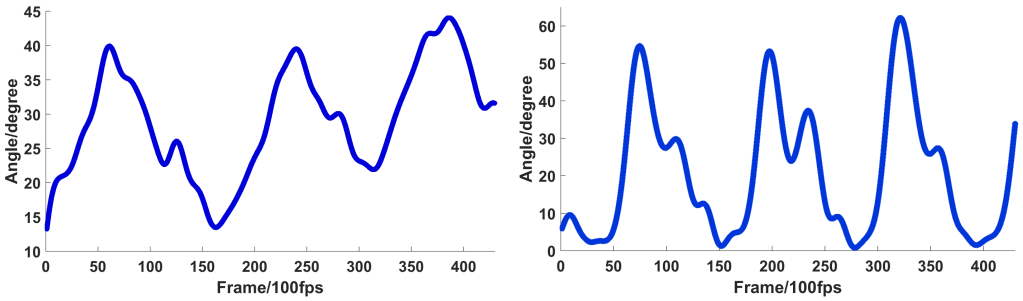


Figure 2.15: Knee angle for two stroke survivors.

Fig.2.15 shows two examples of knee joint angle during three walking cycles for two stroke survivors obtained by the proposed system. Comparing these results with those of the 4 healthy subjects shown in Fig.2.13, the effect of stroke is noticeable indicating

movement abnormalities unique to each individual. This clearly shows the need for a person-centric framework, as proposed in this chapter.

2.6 Summary

A video-based motion capture framework is proposed for motion analysis using a single MS Kinect sensor based on simultaneous marker detection and identification in 3D space and model-based kinematics analysis. Both the optical motion capture system and gait analysis application are evaluated over close to 100 sequences, involving 9 stroke survivors and 5 healthy subjects, and benchmarked against the 12 camera state-of-the-art VICON system following the system configuration in [82]. The acquired trajectory data by the proposed system is further used in the rest chapters as pre-processed data.

In contrast to VICON and similar commercial systems like Qualisys, the proposed framework, which supports a portable sensor for capturing experiments, is suitable for tele-rehabilitation programs through visualization, presentation and rehabilitation interfaces built in the proposed application. Validation results indicate high accuracy for sagittal plane gait analysis in measuring kinematics measurements, such as step and stride length, stance and swing duration. This makes the system practical in clinical tests for different rehabilitation studies. Furthermore, the application-specific results clearly show the need for a person-centric framework, as proposed in this chapter. A novel software package ("OPTIKA") is developed as a major contribution of this chapter which had been evaluated during two different rehabilitation sessions at Brain and Spinal Injury Center, Manchester, UK.

However, the proposed system still has some drawbacks: (1) still needs expertise to manually place the markers on the body. (2) requires extra time in scene calibration to capture the sensitive parameters associated to the environment and application, such as threshold for blob detection, blob base radius, start and end walking points (see Alg. 4). (3) application specific model, such as skeleton model of walking exercise in sagittal view (see Fig. 2.7), is required to perform the marker labeling. This results in low

practical value and high complexity in deploying the system for various motion capture based healthcare monitoring applications, such as gait analysis from different views, timed-up-and-go test [58]. (4) can only be operated in indoor environment which is lack of portability for outdoor motion capture applications. Approaches to solve these drawbacks can be a research direction in the future work (see Sec. 6.2).

In the proposed framework, algorithms associated with optical motion capture are generic to any application while only Algs. 4, 6 and 5 are application specific, Hence, only the latter three need modification for different rehabilitation exercises that require motion analysis. While the results are presented for the rehabilitation walking exercise in the sagittal plane view only, the frontal view gait analysis (see for example [51] for assessing upper limb movement) can be a research direction in the future work (see Sec. 6.2).

Chapter 3

Trajectory-based Motion Assessment

3.1 Review of Gait Phase Classification Methodologies

High-level features, such as step through length, step height speed, step interval, are extracted in [83] via a MS Kinect, during 360°-turn analysis. A simple statistical feature thresholding method further performs the classification of normal/abnormal gait. However, no numerical classification results are presented and lack of abnormal data is a limitation of this preliminary study to determe empirical thresholds. A point-of-care gait assessment framework in [84] quantifies several gait indices and evaluates limb impairment for patients with multiple sclerosis, involving Dynamic Time Warping (DTW), Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). However, [84] reports that the inaccuracy of the captured joint trajectories limits the validation of this preliminary study; furthermore, no benchmarking is included. In [85], kinematic parameters are extracted from a unified representation via a generic full-body kinematic model to segment motion sequences into repetitive action sequences, based on the zero-velocity crossing of the feature selection. The proposed unsupervised temporal segmentation method in [85] requires manual parameter tuning of the involved unscented Kalman filter, frequency analysis and adaptive k-means

clustering to achieve good performance. However, full-body actions involved in the experiments are easily-distinguishable and such unsupervised segmentation requires prior knowledge of the number and order of actions occurred in the motion sequences. Moreover, the method fails when a full sequence is not measured correctly. A similar video-based gait phase classification system in [4] first acquires joint trajectories by tracking 2D bullseye paper markers using a single high-speed expensive RGB camera. Heuristic thresholding criteria performs further automatic gait phase labeling once six gait events are extracted. Motivated by [4], a 3D motion capture system is proposed in [46] to facilitate the kinematics representation by tracking 3D trajectories of retro-reflective ball markers together with image processing algorithms. Based on the high precision joint trajectories, a heuristic method [46] is proposed to extract stance and swing phases. Both [4] and [46] require adaptive parameter tuning to achieve acceptable low classification error rate (the proportion of patterns that have been incorrectly classified).

In [17], a lower limb exoskeleton robot ROBIN-H1, a walking rehabilitation service for stroke patients, first acquires pitch orientations and angular velocities of the robot legs. The task of classifying stance and swing phases by the captured kinematics are then investigated by training a Multilayer Perceptron (MLP) neural network (NN) and an NN-based non-linear autoregressive with exogenous inputs (NARX). Results show that NARX-NN outperforms MLP-NN, but the classification error rate is 5.7% worse than its offline version. [17] suggests that the further research direction is to use an autoencoder for feature extraction and acquiring more data that comprises walking pattern to improve the ACC against data from stroke patients.

[18] acquire knee joint angle and foot switches as features to detect gait events. The experimental results of the classification fit percentages comparison between artificial neural fuzzy inference systems (ANFIS), autoregressive models with exogenous variables (ARX), output error models (OE), NARX and other NN-based models demonstrate the best model is NARX with a 11.41% classification error rate.

In [23], pHMM is proposed to extract a dynamics-normalized, averaged, gait cycle by observing silhouettes of gait stances, leading to the state-of-the-art identification

of the human subject. Various-length silhouettes are projected into a fixed-length sequence by training pHMM on manual silhouettes created from a subset of the Gait Challenge dataset [86]. However, human silhouettes vary with respect to subject shape, gait speed and walking direction, and it is impractical to generate a dataset manually to cover all possible silhouettes, especially for physical rehabilitation programs. Thus, a more robust and practical high-level motion representation approach is to rely on joint trajectories.

In [87], a commercial motion capture system 'Visual3D' is employed to acquire knee angle parameters to perform four-class gait phase classification by J48 Decision Tree, Random Decision Forest (RF), MLP, and Support Vector Machines (SVM) [88]. It is demonstrated that RF achieves the lowest classification error rate.

This chapter provides the following improvements on the prior state-of-the-art work reviewed above: (1) a comprehensive set of gait parameters, based on 3D joint trajectories, to generalize high-level kinematics across all gait phases, and (2) given a gait sequence, unlike [4], this research better slices the sequence into individual gait phase segments by frame-wise classification instead of detecting gait event frames.

Sec. 3.2 optimally extracts distinct features by observing transitions of 12 gait parameters within a sliding window at varying gait speed and direction, unlike [17] and [18] that only rely on a single kinematics parameter at a fixed gait speed and direction. As a result, temporal feature misalignment is addressed by extracting distinct feature transitions via gait cycle standardization and clustering, mapping spatio-temporal feature transitions by RF and reconstructing adjacent gait phases by fine-tuning gait events. Section IV employs SVM, a two-layer NN, and a NARX-NN model [17] as benchmarks for the evaluation of the classification performance.

3.2 Gait Phase Classification

Gait assessment reveals significant factors of abnormal gait, guiding the rehabilitation assessment and treatment, supporting clinical diagnosis and therapeutic effect evaluation. *Gait phase analysis*, as an important part of gait assessment [89], facilitates

medical diagnosis [18] by capturing movement of joints of interest during individual gait phase, analyzing precisely the status of a patient's muscle activation for effective gait rehabilitation treatments [90].

This section proposes a gait phase analysis method, which aims to accurately identify the start and end of each gait phase, that is, the time period between two consecutive gait events. This is achieved by classifying each frame of the captured video into one of $K = 9$ different gait phases (see Fig. 3.1), labeled as $\{P_1, \cdots, P_K\}$, in order to locate K *gait events* such as heel strike, toe off, etc.

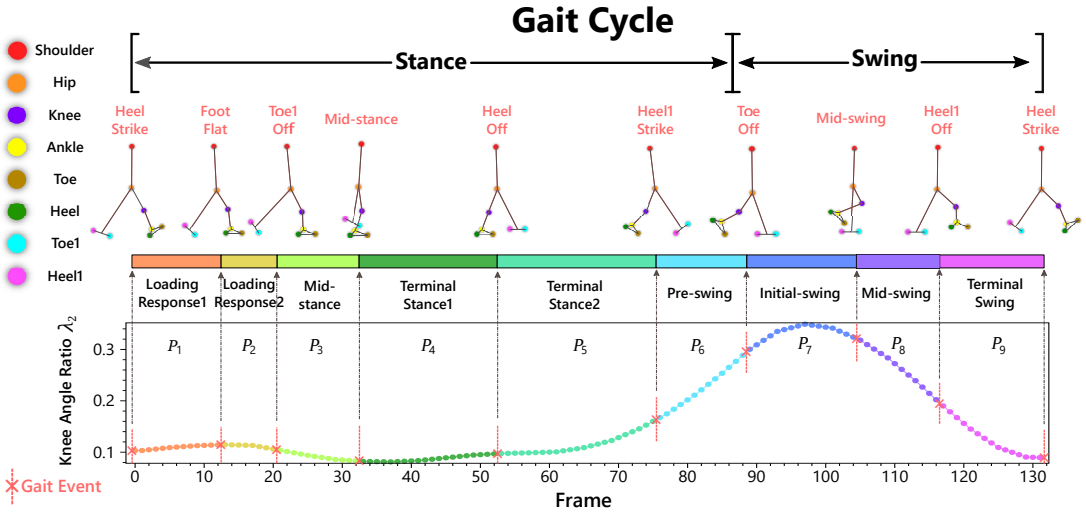


Figure 3.1: A complete typical gait cycle, from a right foot heel strike until the next right foot heel strike. Circles with different colours annotate eight joints of interest listed on the left. Nine specific skeleton poses are described with their corresponding gait events, shown in red text, starting from the Heel Strike event. Colour marking of nine gait phases $P_1 - P_9$, where P_1 denotes Loading Response1, and so on. Gait phase denotes a period between two consecutive gait events. Bottom panel demonstrates a typical curve of knee angle ratio λ_2 through the whole gait cycle. Table 3.1 defines the remaining gait parameters, targeting significant gait kinematics.

12 gait parameters $(\lambda_1, \cdots, \lambda_{12})$, defined in Table 3.1, are used to characterize the gait motion during a 4-meter walking test. These *gait parameters* and their kinematics observations, are defined following [91] and [4]. Fig. 3.2 demonstrates the definitions of the proposed *gait parameters*. A typical visual representation of each proposed gait parameters are shown in Fig. 3.3, regarding the gait phase information shown in individual color. The change of these gait parameters during the walking test is used as

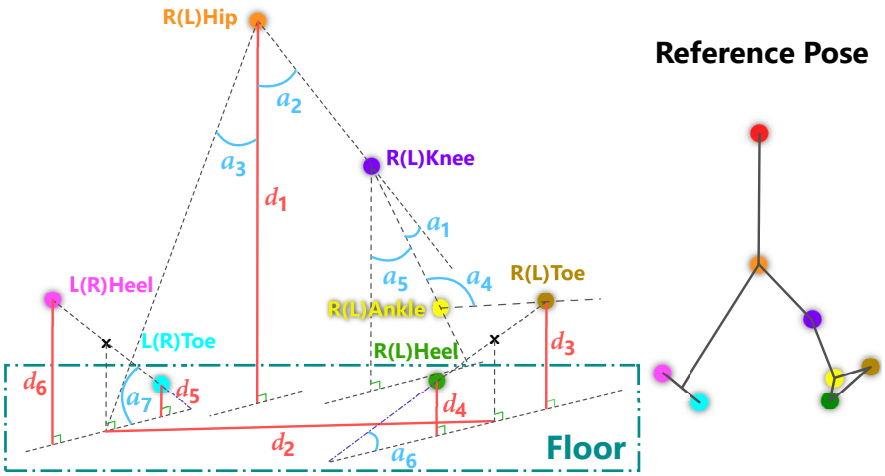
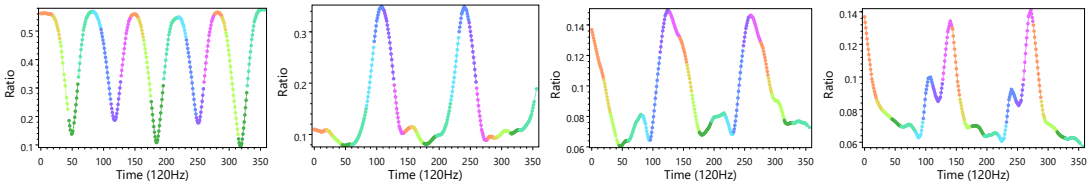
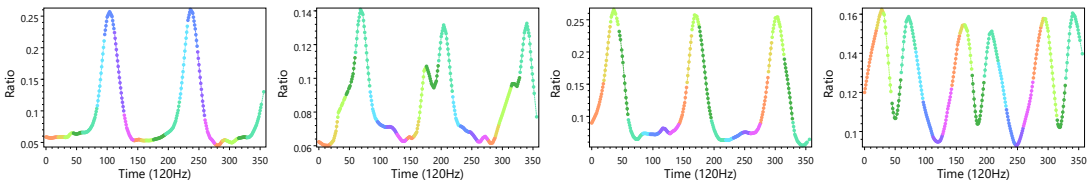


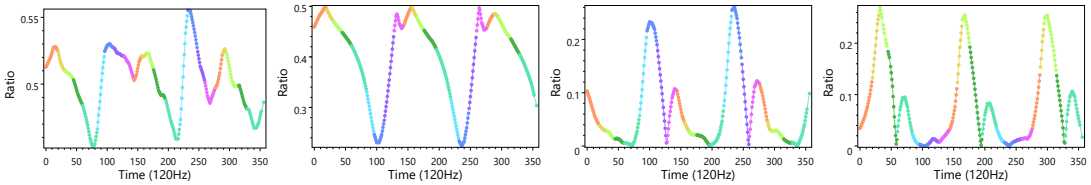
Figure 3.2: Visual representation of distances d_i and angles a_i that are used to define gait parameters in Table 3.1. 'x' denotes the 3D position at a segment centroid of two relevant joints represented by the same color-coded cycles as in Fig. 3.1. Note that in this figure, the camera is placed at the subject's right side. Thus R (right) ankle is captured while L(left) ankle is only tracked when the camera is placed on the left side.



(a) Foot distance ratio λ_1 (b) Knee angle ratio λ_2 (c) Thigh plane angle ratio λ_3 (d) Toe raise ratio λ_4



(e) Heel raise ratio λ_5 (f) Toe 1 raise ratio λ_6 (g) Heel 1 raise ratio λ_7 (h) Leg plane 1 angle ratio λ_8



(i) Ankle angle ratio λ_9 (j) Shank plane angle ratio λ_{10} (k) Foot angle ratio λ_{11} (l) Foot 1 angle ratio λ_{12}

Figure 3.3: Visualization of proposed 12 gait parameters during almost three gait cycles. The colors that refer to each gait phase are the same as shown in Fig. 3.1.

Table 3.1: Definitions and observations of the proposed 12 gait parameters. See Fig. 3.2 for the definition of parameters d_i and a_i. Note that parameters named with/without "1" are associated the back/frontal leg seen in sagittal view.

Parameter	Value	Observation
foot distance ratio λ_1	$\frac{d_2}{d_1}$	foot switch
knee angle ratio λ_2	$\frac{a_1}{180^\circ}$	limb support
thigh plane angle ratio λ_3	$\frac{a_2}{180^\circ}$	femur swing
toe raise ratio λ_4	$\frac{d_3}{d_1}$	toe contact
heel raise ratio λ_5	$\frac{d_4}{d_1}$	heel contact
toe 1 raise ratio λ_6	$\frac{d_5}{d_1}$	toe 1 contact
heel 1 raise ratio λ_7	$\frac{d_6}{d_1}$	heel 1 contact
leg plane 1 angle ratio λ_8	$\frac{a_3}{180^\circ}$	leg 1 swing
ankle angle ratio λ_9	$\frac{a_4}{180^\circ}$	limb support
shank plane angle ratio λ_{10}	$\frac{a_5}{180^\circ}$	tibia swing
foot angle ratio λ_{11}	$\frac{a_6}{180^\circ}$	foot support
foot 1 angle ratio λ_{12}	$\frac{a_7}{180^\circ}$	foot 1 support

the input features to perform gait phase classification (see Sec. 3.2.2). Joint trajectories [46] are used to calculate gait parameters per frame, which will, in turn, be used to perform feature extraction and classification to label the gait phases.

The block diagram of the overall proposed system is shown in Fig. 3.4, comprising 3 major steps: (1) gait pattern extraction, (2) gait phase feature extraction, and (3) gait phase reconstruction. These are described in the following three subsections, respectively.

3.2.1 Gait Pattern Extraction

Gait pattern, *i.e.*, the sequence of limb movements during walking, can be characterized by the defined gait parameters (see Table 3.1). While gait patterns are periodic as shown in Fig. 3.1, they vary among subjects due to differences in age, activity type, gender, proportion and health status. Like [92] that uses kinematic data to describe different gait patterns, this research characterizes a gait pattern as the joint movement (here referred to as $\lambda_1, \cdots, \lambda_{12}$ changes) occurring on a complete normalized gait cycle, invariant of walking speed and direction.

Gait cycle detection: Before characterizing the gait patterns, each gait cycle is

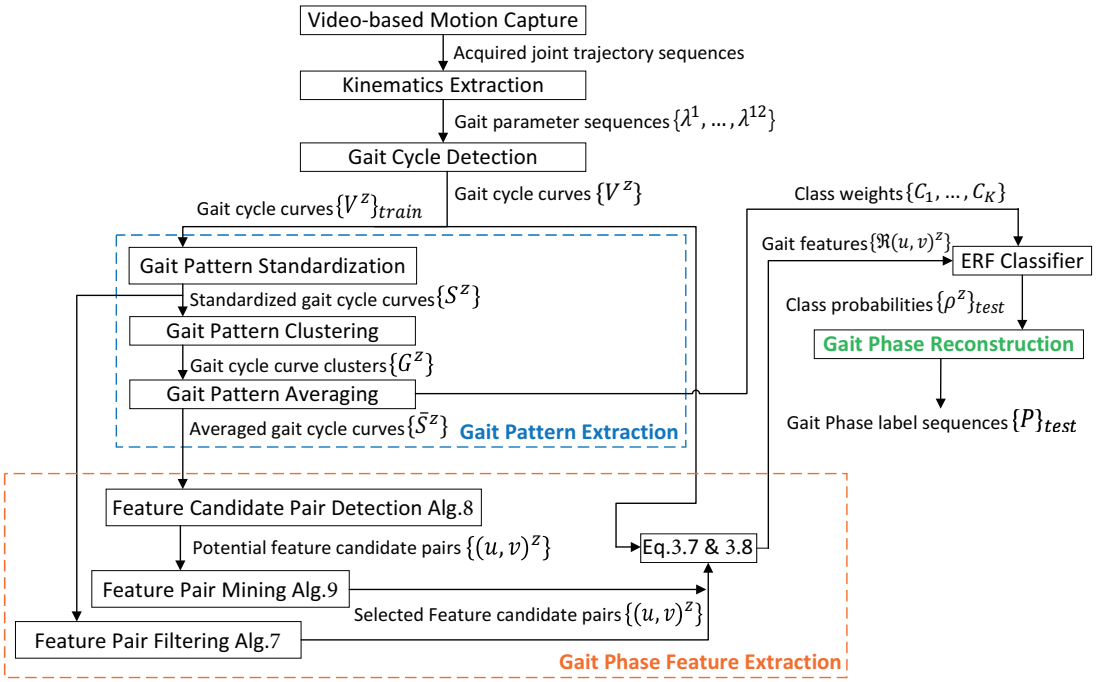


Figure 3.4: Overall system diagram. Note that either Alg. 7 (proposed Filtering method) or Algs. 8 and 9 (proposed Optimized method) are used.

detected by slicing the gait sequences into individual gait cycle (subsequence) based on heel strike event detection. The heel strike event occurs when the foot makes contact with the ground, heel-first; thus, it can be extracted by detecting change points of distances between heel joint and the ground, via the inflection points search method of [46].

Based on trajectories of joints of interest, obtained, for example, by tracking the markers placed on the joints in the recorded video, as in [46], gait parameters (see Table 3.1) are calculated in each frame. For the j-th gait cycle, let $V_j^{\lambda_i}(f)$, $i = 1, \cdots, 12$, denote the value of gait parameter λ_i in Frame f given a gait parameter sequence. For example, $V_1^{\lambda_2}(10)$ denotes the value of knee angle ratio of gait cycle 1 at Frame 10 within a gait parameter sequence.

To *extract distinctive gait patterns* after gait cycle detection, this research adopts the following three steps: (Step1) resample complete gait cycles into a fixed length of L samples, called standardized gait patterns; (Step2) cluster similar gait patterns into groups per gait parameter using density-based spatial clustering of applications with

noise (E-DBSCAN) [93]; (Step3) generalize gait patterns through DTW-Barycenter Averaging (DBA) [94].

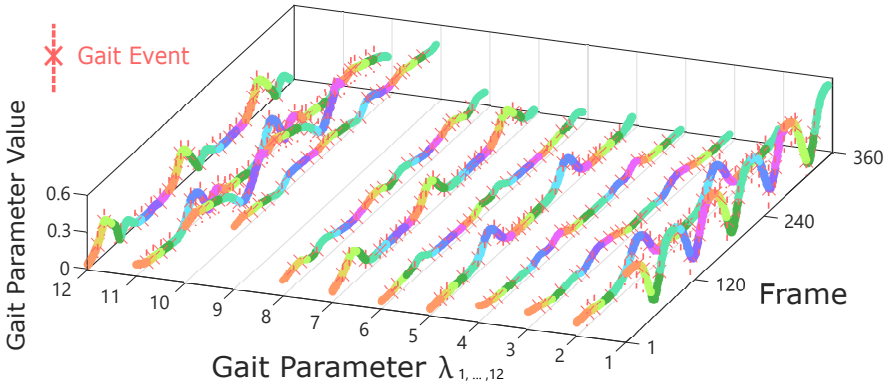


Figure 3.5: An example of the gait pattern representation (360 frames are used in this gait sequence). Note that all 12 gait parameter curves and their gait phase progress jointly characterize gait patterns.

Step1: Gait pattern standardization. First, gait pattern standardization is used to mitigate the influence of varying gait speed. Fig. 3.5 shows an example of gait phase periodicity for the proposed 12 gait parameters. Since, in general, $V_j^{\lambda_i}$ varies for different gait cycles, *i.e.*, different j, *2nd-cubic Bezier curve interpolation [80]* is used to resample $V_j^{\lambda_i}$ to a fixed length of L samples leading to a standardized gait parameter curve $S_j^{\lambda_i}(x = 1, \cdots L)$, denoted in the following by S_j^z, $z \in \{\lambda_1, \cdots, \lambda_{12}\}$.

Step2: Gait pattern clustering. Clustering analysis is usually used to group similar gait patterns in order to further perform abnormal detection, person recognition, etc. Gait phases are usually ignored in the literature for those tasks. However, similar standardized kinematics might have different gait phases across subjects and this variability is important to capture key gait events. For example, in [95], a wearable sensor-based gait phase detection system uses gait phase duration to classify level-walking and walking upstairs and downstairs by learning a decision tree model. This research takes both gait parameter change and gait phase transition into account to capture the differences between standardized gait patterns. The function $\mathbf{\Phi}_{i,j}^z(x, y)$ that measures the distance between sample x in i-th gait pattern S_i^z and sample y in

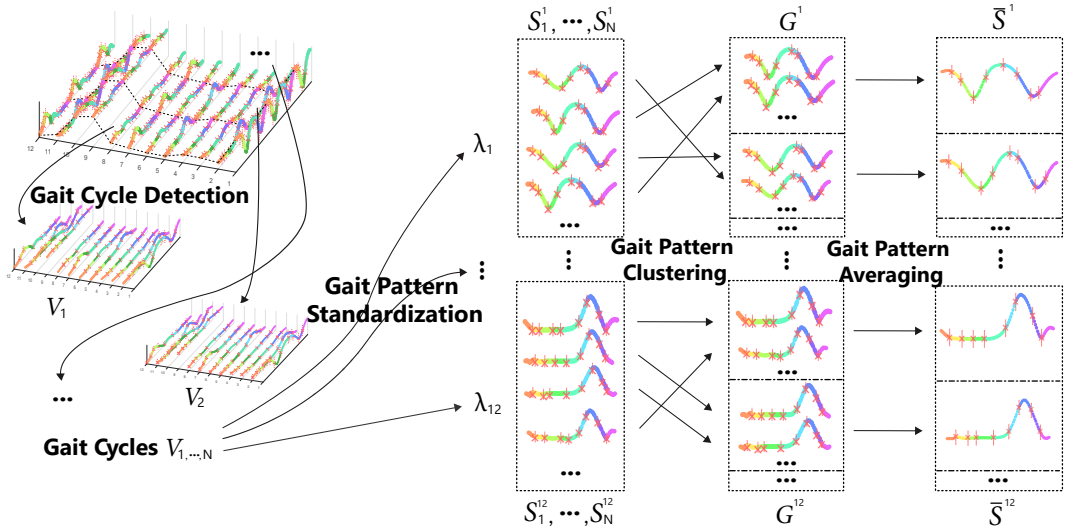


Figure 3.6: Proposed gait pattern extraction, showing steps from left to right: extraction of gait parameter curves V^{λ_i} via gait cycle detection, then standardizing to S^z (Step1), clustering to G^z (Step2) and averaging as $\bar{S}^z$ (Step3).

j-th gait pattern S_j^z is defined as:

$$\mathbf{\Phi}_{i,j}^z(x,y) = \big(S_i^z(x) - S_j^z(y)\big) \cdot \exp\big(\phi(P_{S_i^z(x)}, P_{S_j^z(y)})\big), \tag{3.1}$$

where

$$\begin{aligned}\phi(P_a, P_b) = &\lceil \frac{K-1}{2} \rceil - \Big| \mathbb{1}\big(\lfloor \frac{2|P_a - P_b|}{K+1} \rfloor \bmod 2 = 0\big) \cdot \\ &\lceil \frac{K-1}{2} \rceil - \Big(|P_a - P_b| \bmod \lceil \frac{K+1}{2} \rceil \Big) \Big|.\end{aligned} \tag{3.2}$$

Note that $P_{S_i^z(x)} \in \{1, 2, \cdots, K\}$ refers to the corresponding gait phase label at sample x in i-th the gait pattern S_i^z. $\mathbb{1}(p)$ is an operator that returns 1 if a Boolean expression p is true, and 0, otherwise. Eq. (3.1) includes both the numerical distance $(S_i^z(x) - S_j^z(y))$ and exponential label distance $\exp\big(\phi(P_{S_i^z(x)}, P_{S_j^z(y)})\big)$, instead of other conventional distance functions that are commonly used in DTW. Furthermore, parameter-free E-DBSCAN is used to obtain groups of gait patterns via DTW with distance function $\mathbf{\Phi}_{i,j}^z(x,y)$. Each cluster group $G^z = \{\mathring{S}_1^z, \cdots, \mathring{S}_M^z\}$ represents M similar gait patterns.

Step3: Gait pattern averaging. After E-DBSCAN clustering, DBA is operated

to extract labeled gait parameter curves to represent distinctive gait patterns. For each cluster, distance function $\mathbf{\Psi}^z_m(x, y)$ is the conventional DTW distance function between the sample x in the averaged curve $\bar{S}^z$ (a curve that averages all curves in the same cluster and is initialized as the curve with minimum Euclidean distance compared with all other curves) and the sample y at any $\mathring{S}^z_m$ in the m-th cluster, $m = 1, \cdots, M$, given by:

$$\mathbf{\Psi}^z_m(x, y) = \exp\Big(- \bar{\omega}^z_m(x, y) \cdot \big(\mathring{S}^z_m(y) - \bar{S}^z(x)\big)\Big), \tag{3.3}$$

where a shared weight $\bar{\omega}^z_m(x, y)$ is computed as:

$$\bar{\omega}^z_m(x, y) = \frac{\sum_{k=1}^{K} \Big(\mathbb{1}\big(P_{\mathring{S}^z_m(y)} = k\big)\Big)}{\sum_{k=1}^{K} \sum_{y'=1}^{m_y} \Big(\mathbb{1}\big(P_{\mathring{S}^z_m(y')} = k\big)\Big)} \cdot \exp\Big(\phi\big(P_{\bar{S}^z(x)}, P_{\mathring{S}^z_m(y)}\big)\Big). \tag{3.4}$$

Note that the number of occurrences of gait phase label k, is counted in all $\mathring{S}^z_{1,\cdots,M}$ within the same cluster. The exponential label distance is used as metric to weight the distance $\mathring{S}^z_m(y) - \bar{S}^z(x)$ in Eq. 3.3. The following two measurements are used to tradeoff: 1) between $\mathring{S}^z_m(y)$ and cluster mean $\bar{S}^z(x)$; 2) among gait phase labels $P_{\bar{S}^z(x)}$ and $P_{\mathring{S}^z_m(y)}$ for better gait pattern generalization.

To obtain the corresponding gait phase label for each sample at the averaged curve $\bar{S}^z$, this research first extracts all warped paths between the averaged curve and all curves within a cluster via DTW using distance function Eq.3.3. Then, it adopts min-max standardization to limit the path cost at each iteration of DBA. Finally, for each sample x, it measures the summed path cost for each possible label across all warped paths at all iterations and then label sample x as the label with minimum-sum path cost. Therefore, both numerical gait parameter value averaging and gait phase progress averaging are taken into account to obtain a good representation of a set of similar gait patterns.

Fig. 3.6 demonstrates the overall gait pattern extraction. The resulting series of the standardized curves $\bar{S}^z$ are obtained by performing Steps 1-3 for the purpose of data reduction from a large number of S^z curves. For example, 2 to 7 clusters can be

obtained in $\bar{S}^z$ for each $z \in \{\lambda_1, \cdots, \lambda_{12}\}$, from a total 205 gait patterns S^z during experiments.

Since the duration of each gait phase is different, the different numbers of frames in each class (gait phase) result in imbalanced class distribution for the classification task. This results in categorizing the samples into the majority class, in particular the insufficient identification of minority class. Following the validated work in [96] that assigns individual weights for each class to address this bias problem for medical data, the class weights $C_{k=1,\cdots,K}$ of each gait phase label in this work are used during training as:

$$C_{k=1,\cdots,K} = \frac{12 \cdot L}{\sum_{z=\lambda_1}^{\lambda_{12}} \sum_{l=1}^{L} \left(\mathbb{1}\left(P_{\bar{S}^z(l)} = k\right)\right)}. \tag{3.5}$$

3.2.2 Gait Phase Feature Extraction

Recall that the task is to classify each frame into one of $K = 9$ gait phases. Two main challenges in this multi-class gait phase classification problem are: (1) partial gait parameter curves due to both incomplete gait cycles and missing values due to the occlusion at joints of interest; (2) subject-sensitive gait patterns with varying gait speeds. The missing values result in corrupted input features for classification task. RF as one of the widely used machine learning algorithm enables the input missing observations by replacing the missing values with the following two ways: (a) the most frequent non-missing values of each class, (b) an average over the non-missing values of the m-th variables weighted by the proximities between the n-th case and the non-missing value case, given a missing feature value x(m,n) as proposed in [97]. Motivated by this, in this work, RF is used to handle the missing observations for the first challenge. In order to address the second challenge, inspired by image classification study [98], this research introduces the feature candidate pair and sliding window to mitigate feature alignment.

In general, feature engineering of gait phase classification comprises: (1) Feature Alignment: extracting time-varying features to solve temporal misalignment that causes feature mismatch, such that extracted abstract features are invariant to gait speed and

subject-dependent patterns. (2) Feature Mining: mine distinct feature pairs, reducing feature candidate pair set without sacrificing the performance.

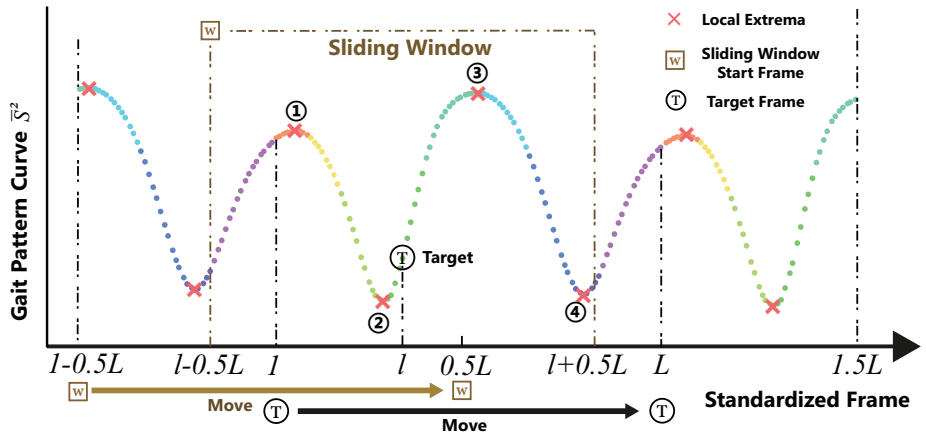


Figure 3.7: Feature candidate pair extraction in standardized frame domain: for target frame $l \in 1, \ldots, L$, different candidate pairs (u, v), that link frames with local extrema, give different value responses in terms of knee angle ratio λ_2. Same frames with local extrema give different (u, v) values for different target frames from 1 to L and its corresponding L-length sliding window starting from frame $1 - 0.5L$ to $0.5L$.

Proposed Feature Alignment

To extract the transition information of adjacent gait phases, gait pattern curves $\bar{S}^z$ are used which can be obtained as explained in Sec.3.2.1 (Step 3). A sliding window across the frames is selected in order to extract time-varying and linearly separable temporal features, where the extracted full gait cycle length L is set to be the length of the sliding window. Since some gait cycles are incomplete, this research estimates the length of each incomplete gait cycle from full gait cycles based on the speed of the hip marker.

Considering the continuity of two adjacent gait pattern curves, $\bar{S}^z$ becomes periodic

by connecting its starting frame to the end frame (see Fig. 3.7) of its replica by:

$$\tilde{S}^z = \frac{\bar{S}^z(1) + \bar{S}^z(L)}{2}$$

$$\bar{S}^z(x) = \begin{cases} \bar{S}^z(x+L) & x \in 1-0.5L, \cdots, 1 \\ \bar{S}^z(x) + \frac{(\tau L - x)(\tilde{S}^z - \bar{S}^z(1))}{\tau L} & x \in 1, \cdots, \tau L \\ \bar{S}^z(x) & x \in \tau L, \cdots, L - \tau L \\ \bar{S}^z(x) + \frac{(\tau L + x - L)(\tilde{S}^z - \bar{S}^z(L))}{\tau L} & x \in L - \tau L, \cdots, L \\ \bar{S}^z(x-L) & x \in L, \cdots, 1.5L, \end{cases} \tag{3.6}$$

where continuous edge ratio is set to $\tau = 0.1$ to make sure that the boundaries between adjacent $\bar{S}^z$'s are smooth, *e.g.*, the target point $\bar{S}^z(x)$ at any position x of the gait pattern curve is smooth within an L-length sliding window. Since the standardized curve is obtained by its replicas, the gait phase labels are the same as the ones extracted during gait cycle averaging. The standardized time stamp at sample x becomes x/L given a fixed length of L samples of a sliding window, centered at x. In order to address misalignment of features, this research introduces feature pair (u, v), where u and v take values in the range $[-0.5, 0.5]$ with a minimum resolution step of $1/L$ to describe any two points near x within the sliding window.

To capture the time-varying features for each sample x within an L-length sliding window on the periodic gait pattern curve $\bar{S}^z(x)$, $z \in \{\lambda_1, \cdots, \lambda_{12}\}$, given a feature pair $(u, v)^z$ for gait parameter z, the feature value are calculated by:

$$\Re_l(u, v) = \frac{\|\bar{S}^z(l + uL) - \bar{S}^z(l + vL)\|}{|u - v|}, \tag{3.7}$$

Since the standardized gait cycle/gait pattern length is heuristically set to 100 for matching the resampling ratio in Chapter 2 (benchmark kinematic measurements with the state-of-the-art motion capture system VICON). To generate features for classify 9 gait phases for each frame, $L = 100$ is chosen as the fixed length of sliding window such that a full gait cycle can be observed in the sliding window. As a result, the feature candidate pairs can be located in any two different gait phases within a full gait cycle

in order to provide features across any two gait phases for each frame. This results in $\binom{L}{2} = 4950$ feature candidate pairs for each gait parameter, including $(-0.5, -0.49)$, $(-0.5, -0.48), \cdots, (-0.5, 0.5), (-0.49, -0.48), \cdots, (0.49, 0.5)$ as $(u, v)^z$.

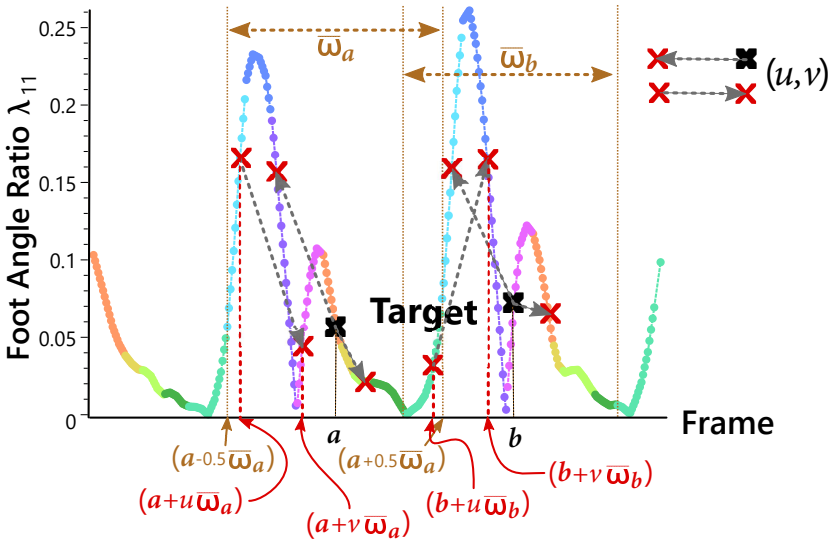


Figure 3.8: Feature pairs (u, v) at target frames a and b. Different values of foot angle ratio parameter λ_{11} for the same (u, v) pair can be observed at the target frames a and b.

In order to ensure the same time-varying feature statistics for the same (u, v) at the testing stage, this research introduces the following equation to calculate the feature value:

$$\Re_f(u, v) = \frac{\|V^z(f + u\varpi_f) - V^z(f + v\varpi_f)\|}{|t^z(f + u\varpi_f) - t^z(f + v\varpi_f)|}, \tag{3.8}$$

where for a frame f, $t(f+u\varpi_f)$ denotes the u shifted timestamp from the time at which target frame f is acquired given a ϖ_f-frame long sliding window centered at frame f. Thus, a gait sequence sampled from frame $f + u\varpi_f$ to frame $f + v\varpi_f$ represents a complete gait cycle. Fig. 3.8 demonstrates the feature pairs (u, v) in the time domain.

Proposed Feature Mining

To reduce the size of the feature candidate pairs set Ω, only the most informative pairs is selected. Since missing feature values often occur near the boundary of the gait parameter curves, the missing features for those conditions are limited. To deal with

the missing feature issues and further reduce the classification error rate, enhanced randomized decision forest (ERF) [99] is used to perform this task by keeping only uncorrelated high performing trees in random forests, reducing the occurrence of bad trees caused by randomization.

Two mining approaches are proposed to extract the feature pairs: (1) filtering method: enumerate all (u, v) link combinations in terms of standardized length $L = 100$ such that the quality of $12\binom{L}{2}$ feature candidate pairs for all gait parameters is evaluated during mining; (2) optimized method: find all (u, v) link combinations of local extrema frames via Persistence1D [100] that finds paired local extrema indices based on a certain threshold of absolute data difference for all local edges given a set of one-dimensional numerical values.

2.1) **Filtering method (Alg. 7)**: It enumerates all possible (u, v) link combinations of any two frames within an L-length sliding window centered on the target frame. Then it computes the feature values using (3.7) from the standardized curve $\bar{S}^z$ extracted in Sec. 3.2.1. To ensure the feature values are in the same feature space as the ERF model, [101] uses Gini impurity $= 1 - \sum_{k=1}^{K}(p_k)^2$ (where p_k is the probability of a frame being classified to a particular gait phase class k) as evaluation metric to quantify the information content of a feature candidate pair. To avoid overfitting the ERF model, it selects $\sum_{z=\lambda_1}^{\lambda_{12}} num^z$ top feature candidates with lowest Gini impurity from a total of $12\binom{L}{2}$ candidate pairs, where num^z is the resulting feature count for each gait parameter z. The distinct feature pairs are chosen via the following *Criteria* and Alg. 7:

(1) Given a finite set of feature candidate pairs $\{(u, v)^z_\varepsilon\}$ for each gait parameter z, it computes Gini impurity $\mathcal{G}^z_\varepsilon(P_a, P_b)$ (see Alg. 7) to separate samples with gait phase labels P_a and P_b, $a, b \in \{1, \cdots, K\}$ as a measure of the quality $Q^z_\varepsilon(P_a, P_b)$ for the ε-th feature candidate pair $(u, v)_\varepsilon$ for the gait parameter z.

(2) For each z, it calculates the quality Q^z for all possible gait phase label combinations

$\mathcal{H} = \{(P_1, P_2), (P_1, P_3), \cdots, (P_2, P_3), \cdots, (P_{k-1}, P_k)\}$ by:

$$Q^z = \sum_{\varepsilon=1}^{\binom{L}{2}} \sum_{H \in \mathcal{H}} Q_\varepsilon^z(H). \tag{3.9}$$

(3) To maximize the total quality, only num^z distinct pairs with the highest quality are considered, where

$$num^z = \frac{\Omega \sum_{\varepsilon=1}^{\binom{L}{2}} \sum_{H \in \mathcal{H}} Q_\varepsilon^z(H)}{\sum_{z=\lambda_1}^{\lambda_{12}} \sum_{H \in \mathcal{H}} Q^z(H)}. \tag{3.10}$$

Algorithm 7: Feature candidate pair filtering.

Input: Standardized gait pattern curves $\{S^z\}$ (Sec.3.2.1);
Potential feature candidate pair list $\{(u, v)^z\}$;
Output: Feature pair list $\{(u, v)^z\}$;

update standardized gait pattern curves $\{\bar{S}^z\}$ via Eq. 3.6;
foreach $z \in \{\lambda_1, \cdots, \lambda_{12}\}$ **do**
  foreach $H = (P_a, P_b) \in \mathcal{H}$ **do**
    foreach $(u, v) \in \{(u, v)^z\}$ **do**
      compute $\{\Re_l(u, v)^z\}$ for all $\{\bar{S}^z\}$ via Eq. 3.7;
      find best split for P_a and P_b that minimizes $\mathcal{G}_\varepsilon^z(P_a, P_b) = 1 - p_{P_a}^2 - p_{P_b}^2$ from $\{\Re_l(u, v)^z\}$;
      update $\{\Re_l(u, v)^z\}$ for all $\{S^z\}$ via Eq. 3.7;
      update $\mathcal{G}_\varepsilon^z(P_a, P_b)$ using the found best split;
      set $Q_\varepsilon^z(H) = 1 - \mathcal{G}_\varepsilon^z(H)$;
  update Q^z via Eq. 3.9;
estimate num^z using Eq. 3.10;
foreach $z \in \{\lambda_1, \cdots, \lambda_{12}\}$ **do**
  sort $\{(u, v)^z\}$ by Q^z;
  keep top num^z in $\{(u, v)^z\}$;

2.2) **Optimized method (Algs. 8 and 9)**: It observes heuristically that most of the distinct feature pairs link two frames where at least one frame has a local extreme value. Given the large time-varying feature value response and availability of a more distinguishable difference for classifying two gait phases, in order to reduce the computational complexity of feature mining, a more efficient way to extract feature candidate pairs (u, v) is to find those pairs that link local extrema. To detect the local extrema,

Persistence1D [100] is used to filter out those extrema with relatively large persistence.

Algorithm 8: Feature candidate pair detection.

Input: Standardized gait pattern curves $\{\bar{S}^z\}$ (Sec.3.2.1);
Neighbor radius r_z;
Output: Potential feature candidate pair list $\{(u,v)^z\}$;

update standardized gait pattern curves $\{\bar{S}^z\}$ via Eq. 3.6;
foreach *frame* $l \in \{1, \cdots, L\}$ **do**
  find frames $\{f_{i=1:n}\}$ from frames $l - 0.5L + 1, \cdots, l + 0.5L - 1$ with local extrema and persistences $\beta = \{\beta_{i=1:n}\}$ via [100];
  if $n < 2\sqrt{L}$ **then**
    $\beta_{threshold} = median(\beta)$;
  else
    $\beta_{threshold} = 0$;
  keep frames $\{f_{j=1:\mathring{n}}\}$ with corresponding persistence $\beta_j < \beta_{threshold}$;
  for $U = \{f_{j=1:\mathring{n}}\}$ **do**
    for $V = \{U, \cdots, f_{j=\mathring{n}}\}$ **do**
      if $V - U > r_z$ **then**
        add $(u,v) = (\frac{U-l}{L}, \frac{V-l}{L})$ to feature candidate pair list $\{(u,v)^z\}$;

For a standardized gait pattern curve $\bar{S}^z$, Alg. 8 extracts feature candidate pairs $\{(u,v)^z\}$ where neighbor radius r_z is designed to remove duplicate feature candidate pairs with similar feature values during adjacent frames. A good value for r_z is determined as follows: find the minimum frame length for which any two distinguishable frames are separated, and calculate the median duration $\ddot{T}_{p=1,\cdots,K}$ for each gait phase in the training sets. For example, it is observed that $\min(\ddot{T}_{p=1,\cdots,K}) \approx 6\%$ of a gait cycle; therefore, at least $r_z = 6\%L = 6$ frames are required to observe two distinguishable frames within the same gait phase. Note that a standardized gait pattern curve $\bar{S}^{z=2}$ for knee angle ratio λ_2 is first extracted as explained in Sec. 3.2.1 (see Fig. 3.7).

As shown in Fig. 3.7, the standardized frames from 1 to L are used to generate periodic gait pattern curve frames from $1 - 0.5L$ to $1.5L$ using Eq. 3.6. Next, an L-length sliding window moves from frame $1 - 0.5L$ to $0.5L$ with its corresponding target frame l from 1 to L for full enumeration of standardized gait pattern frames in terms of all potential feature candidate pairs. For each enumerated frame inside the sliding window, Persistence1D [100] extracts local extrema and its persistence. In order to

Algorithm 9: Feature pair mining.

Input: Potential feature candidate pair list $\{(u,v)_1^z, \cdots, \{(u,v)_{N_z}^z\}$ for each gait parameter $z \in \{\lambda_1, \cdots, \lambda_{12}\}$;
Desired Feature Count Ω;

Output: Feature pair list $\{(u,v)^z\}$ for each gait parameter z;

foreach z *in* $\{\lambda_1, \cdots, \lambda_{12}\}$ **do**
  for $n_z = \{1, \cdots, N_z\}$ **do**
    estimate probability of each candidate pair by $O_{n_z} = \frac{num((u,v)_{n_z})}{num((u,v)_1, \cdots, (u,v)_{N_z})}$;
  let $O_{median^z} = median(\{O_1, \cdots, O_{N_z}\})$;
foreach $(u,v)^z$ *in* $\{(u,v)^z\}$ **do**
  extract gait phase probabilities $\{p^1_{(u,v)^z}, \cdots, p^K_{(u,v)^z}\}$ when $(u,v)^z$ is detected at its target frame l;
foreach z *in* $\{\lambda_1, \cdots, \lambda_{12}\}$ **do**
  set feature count $\Omega^z = \frac{\Omega \cdot O_{median^z}}{\sum_{z=\lambda_1}^{\lambda_{12}} O_{median^z}}$;
  sort feature candidate pairs $\{(u,v)^z\}$ descending by probability O_{n_z};
  group $\{(u,v)^z\}$ by its top 2 gait phase labels with relative high gait phase probabilities;
  add Ω^z $(u,v)^z$ in total of groups to feature pair list $\{(u,v)^z\}$ balancedly based on class weights $C_{k=1,\cdots,K}$;

filter out those extrema with a relatively small value change, median persistence is heuristically use set as the threshold. For the remaining frames with local extrema, feature candidate pair (u,v) links any two frames based on their normalized offsets to the target frame l.

In order to reduce the computation time, Alg. 9 selects the potential feature candidate pairs based on their probabilities of occurrence in terms of gait parameter $z \in \{\lambda_1, \cdots, \lambda_{12}\}$. This reduces the number of input feature candidate pairs when globally optimal feature selection criteria is applied, such that significant computation time is saved to maximize the factor of variation between gait phases.

Given a total of $12\binom{L}{2} = 59,400$ feature candidate pairs for all gait parameters $z \in \{\lambda_1, \cdots, \lambda_{12}\}$, the filtering method examines each (u,v) by Gini impurity and selects those candidate pairs as per Eq. 3.10 on all extracted standardized gait patterns $\bar{S}$ within the training samples. Instead of this exhaustive search, the proposed optimized method first detects potential feature candidate pairs via Alg. 8 where those pairs with

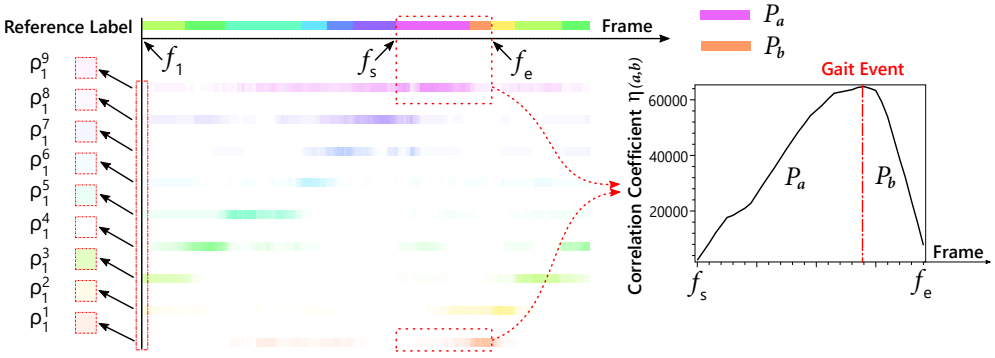


Figure 3.9: Class probability representation in a complete gait cycle and correlation coefficient for adjacent gait phase periods. Unique color is assigned to each gait phase label where low transparency means low probability of the label occurrence at specified frame.

low feature response are dropped; afterwards, the pairs will be selected according to their probabilities of occurrence in the training samples, via Alg. 9.

3.2.3 Gait Phase Reconstruction

After the ERF-based frame-wise classification, this research re-segments each gait phase period using the proposed gait phase reconstruction approach described next.

Since the gait phase is defined as the period between two adjacent gait events, the gait phase label sequence is a periodic piece-wise smooth signal. The gait phase label sequence obtained by the gait phase classifier, described in the previous subsection, sometimes contains classification errors and could even be non-periodic.

In particular, this research first locates a gait event as a time-stamp within two adjacent gait phases. Then, it obtains a class probability vector $\rho_f = \{\rho_f^1, \cdots, \rho_f^K\}$, given feature values $\Re^z(u, v)$ for each frame f from the trained ERF model as discussed in Sec. 3.2.2. Since, as shown in Fig. 3.9, mis-classification often occurs near the boundary of two adjacent gait phases, a correlation coefficient η_f in Eq.3.11 is proposed to capture similarity between adjacent gait events P_a and P_b, within the frames $f_s, \cdots, f_e$:

$$\eta_f(a,b) = \sum_{i=f_s}^{f} \rho_i^a \sum_{i=f}^{f_e} \rho_i^b - \sum_{i=f_s}^{f} \rho_i^b \sum_{i=f}^{f_e} \rho_i^a. \tag{3.11}$$

Fig. 3.9 shows the refinement of gait phase labels which is done by detecting the gait event at the moment when the correlation coefficient reaches the global maximum.

3.3 System Validation

This section reports the experimental results. Each step proposed in the methodology is assessed in terms of classification error rate and complexity in order to show its importance towards the final result, benchmarking each step with relevant algorithms in the literature.

Joint trajectories are obtained using the MS Kinect v2- based motion capture system of [46], which does not rely on Kinect skeleton data and shows close accuracy to commercial 12-camera VICON system [6]. A biomechanical model "Plug-in-Gait model" [102] is implemented by VICON to describe limb motion by placing retro-reflective markers along anatomical landmarks. As the marker placement defined in the model is widely used in the literature across different conventional gait model [103], in this work, the maker placement of "Plug-in-Gait model" is used in the experiments in order to benchmark with the commercial motion capture system VICON. Note that, the proposed algorithms are applicable to other motion capture systems, requiring only 10 joint trajectories defined in the widely used conventional gait model for gait assessment as input. However, the overall gait phase classification performance will depend on the tracking accuracy of the employed motion capture system.

9 stroke survivors and 6 healthy volunteers are asked to walk for 6 meters while the middle 4-meter motion is recorded at two different rehabilitation sessions. All 15 participants read the participant information sheets and completed the consent forms before data collection. The study covered in this research is conducted with the ethical approval of both National Health Service (NHS) and University of Strathclyde, and healthy volunteers with the ethical approval of University of Strathclyde (see Appendix A). As per similar experiments in [17], [18], [104] where 15 subjects, 10 subjects (116 strides), and 25 strides are used, respectively, in the experiments, 126 records (613 strides) are captured from 15 subjects with various walking speeds, directions and

patterns. The trajectories of hip, knee, ankle, toes, heels joints are acquired during the 4-meter walking test. To form ground-truth for classifying $K = 9$ gait events, this research first manually identifies all frames that separate adjacent gait phases by finding specified gait poses defined in [91]. Then, the timestamps are carefully refined by pose template matching on interpolated gait parameters (see Table 3.1). Finally, a gait phase label is assigned to each frame by slicing the extracted gait events.

To evaluate the predictive performance, classification error rate (%), average receiver operating characteristic (ROC) curve [105], [106] and area under the curve (AUC) [105] are used as evaluation metrics. ACC is defined as the percentage of correctly classified frames across all sequences. The average ROC is plotted by measuring the true positive rate (TPR) and false positive rate (FPR) based on one-versus-all binary classification across all sequences and all gait phases. For each gait phase k, the corresponding TPR measures the proportion of frames at phase k that are correctly classified. FPR calculates the proportion of frames not at gait phase k that are classified as gait phase k frames. This research divides the dataset into training and testing set, where the training set comprises stroke survivors 1-5 and healthy volunteers 1-3, and testing set comprises stroke survivors 6-9 and healthy volunteers 4-6. Grid search is adopted to tune hyper-parameters.

The importance of different steps in the proposed system (Fig. 3.4) is evaluated in the following ways. First, to justify the selection of ERF as classifier, this research tests One vs One multi-class SVM and classical two-layer softmax NN against ERF. To test the reliability and validity of the defined 12 gait parameters at frame-wise level, this research feeds normalized trajectories to the classifiers and compares performance when the proposed 12 gait parameters are used as features instead. To assess the value of gait phase reconstruction, schemes with and without this step are assessed.

Evaluations and benchmarking are grouped as follows: (1) Schemes denoted with SVM^1, NN^1, and ERF^1 use normalized joint trajectories as input (as discussed in [107]), *e.g.*, acquired joint trajectory sequences from the output of video-based motion capture are fed directly into the classifier (after normalization); (2) Schemes denoted with the classifier name without any superscripts, *e.g.*, SVM, NN, and ERF, perform

Kinematics Extraction and the proposed Gait Cycle Detection (see Fig. 3.4) and feed the resulting gait cycle curves $V^z(\lambda)$ directly into the classifier; (3) The scheme denoted by NARX-NN uses the state-of-the-art NARX-NN model [18] (Matlab 2016a Neural Time Series toolbox) on the standardized gait cycle curves S^z; since the NARX-NN performs worse when data with incomplete gait cycle is used, frames with complete gait cycles are used only. (4) NARX-NN^2, **Filtering**^{2} (Alg. 7) and **Optimized**^{2} (Alg. 8 and Alg. 9), denote NARX-NN, the proposed system with feature candidate pair filtering, and the proposed system with feature candidate pair detection and feature pair mining, respectively, without the Gait Phase Reconstruction block. (5) The proposed **Filtering** and **Optimized** schemes with all the steps, *e.g.*, including Gait Phase Reconstruction.

Note that for the NARX-NN model, given a regression result Γ_f at frame f [18], its corresponding gait phase label is $P(f) = \mathbb{1}\{\lfloor\Gamma_f\rfloor \mod K = 0\} \cdot K + \lfloor\Gamma_f\rfloor \mod K$, where $\mathbb{1}$ all ones matrix, $\lfloor\cdot\rfloor$ returns the first smaller integer, and mod returns a remainder, and the corresponding class probability is calculated by:

$$\rho_f^k = \begin{cases} 1 - |k - \Gamma_f| & k \in \{\lfloor\Gamma_f\rfloor, \lceil\Gamma_f\rceil\} \\ 0 & k \notin \{\lfloor\Gamma_f\rfloor, \lceil\Gamma_f\rceil\}. \end{cases} \tag{3.12}$$

Levenberg-Marquardt method is used to train a two-layered NARX-NN. In the experiments, data is randomly spitted into 70% data used for training, 15 used for validation and 15% used for testing. All 9×10 trained networks are evaluated for the 15% testing data for time delays from 1-9 frames with 10 network per time delay. The mean ACC of the NARX-NN model is evaluated in [47], showing that the best result is obtained for the input time delay of 8 frames, and this will be used in the following experiments.

The proposed gait phase classification system randomly chooses approximately 80% of the training data as training set and the rest as validation to tune hyper parameters. Note that both sets include frames with incomplete gait cycles, to match the training configurations using NARX-NN. 20 sets of ERF models with depth 20 and 30 trees are trained to evaluate the ACC for various sizes of feature candidate pair set, Ω.

The results are shown in Table 3.2 and Fig. 3.10 that are averaged over 126 gait

records and 20 experiments for each scheme. Benefits from a better characterized gait presentation using the proposed kinematics extraction method can be observed from Table. 3.2 and Fig. 3.10, since the methods that rely on the proposed Kinematics Extraction (SVM, NN, ERF) show a significant ACC and AUC improvement over feeding normalized joint trajectories to the classifiers (schemes with superscript ^{1}). Indeed, SVM consistently significantly outperforms SVM^1 with both higher ACC and AUC, while ERF, NN, ERF^1 and NN^1 show similar results, but all being significantly worse than the proposed (Filtering and Optimized) methods. NARX-NN generally outperforms SVM and NN classifiers, but it is consistently outperformed by the proposed method.

Table 3.2: Mean Classification Error Rate (%) for all $K = 9$ gait phase classes for all tested schemes.

Gait Phase Class	1	2	3	4	5	6	7	8	9	Mean
SVM^1	47.7	61.3	27.2	44.8	33.7	31.3	13.6	16.4	24.3	32.3
NN^1	38.0	50.9	19.2	19.4	21.3	18.4	9.2	12.6	19.2	22.3
ERF^1	48.4	45.3	16.7	21.4	26.9	29.4	7.8	16.8	28.8	25.9
SVM	18.6	38.1	25.9	33.6	20.4	12.5	11.5	12.1	17.9	20.7
NN	20.3	39.5	22.7	24.4	19.6	16.3	12.4	10.1	20.4	20.0
ERF	30.3	38.1	21.2	25.0	19.1	15.8	11.5	10.8	22.7	18.9
NARX-NN^2	20.3	17.6	18.2	19.6	20.6	17.1	15.5	12.7	13.8	16.7
Filtering^2	18.3	14.5	17.6	16.8	17.9	15.7	18.0	12.2	13.7	15.5
Optimized^2	16.9	12.8	13.3	14.7	15.7	14.8	16.8	11.5	13.7	13.9
NARX-NN	17.7	8.8	9.7	9.7	18.1	11.8	7.1	11.5	10.9	11.2
Filtering	1.8	0.9	0.9	1.4	2.4	**1.3**	1.6	**1.8**	2.2	1.1
Optimized	**1.5**	**0.7**	**0.8**	**1.1**	**2.0**	**1.3**	**1.3**	**1.8**	**2.0**	**0.8**

NARX-NN^2, Filtering^2 and Optimized^2 algorithms all significantly outperform ERF classifier. When comparing NARX-NN^2, Filtering^2 and Optimized^2 algorithms with NARX-NN, Filtering and Optimized algorithms, it is observed that the proposed Gait Phase Reconstruction method used in the latter algorithms leads to significantly reduce the classification error rate, of up to 15%. Finally, the proposed Optimized and Filtering methods consistently outperforms all prior benchmarks.

Next, Figs. 3.11 and 3.12, compares more closely the two proposed approaches of feature candidate pair mining, Filtering and Optimized. Since the Filtering method

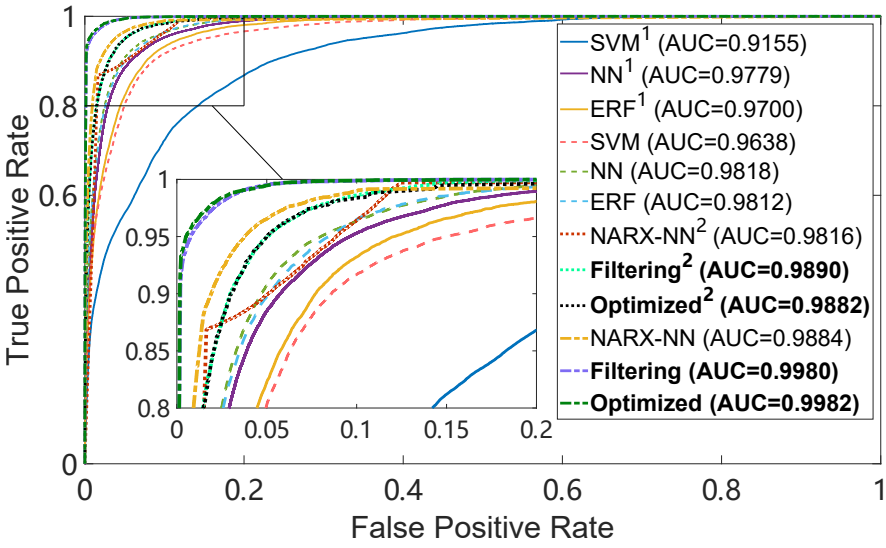


Figure 3.10: Average ROC curves for all tested schemes.

enumerates $\binom{L}{2}$ feature candidate pairs for each gait parameter during feature extraction and mining, it is obvious that the ERF model is over-fitted when the feature number reaches approximately 630 with the lowest final classification error rate. On the other hand, for the Optimized method that extracts all feature candidate pairs with local extrema linkage, a smaller feature number can achieve lower classification error rate. Both validating and testing classification error rates at the classifier stage are improved by using the optimized feature candidate pair detection and mining method (Algs.8 and 9).

This research defines the **computational efficiency** as $Speedup = T_{baseline}/T_{evaluate}$, where $T_{baseline}$ is the average total execution time required to train a classifier using the proposed filtering method, while $T_{evaluate}$ refers to the average total execution time for obtaining the results using the evaluated methods (*e.g.*, Alg. 8 + Alg. 9, or NARX-NN). For instance, the *Speedup* of optimized mining approach during testing equals to the execution time ratio between optimized and filtering mining approach on testing sets using corresponding trained models. The experiments were performed on an Intel i7-4710HQ 2.5GHz CPU, Windows10 operating system, implemented using Visual C++ and Matlab 2016a.

Fig. 3.13 confirms that there is a large computation performance boost using the

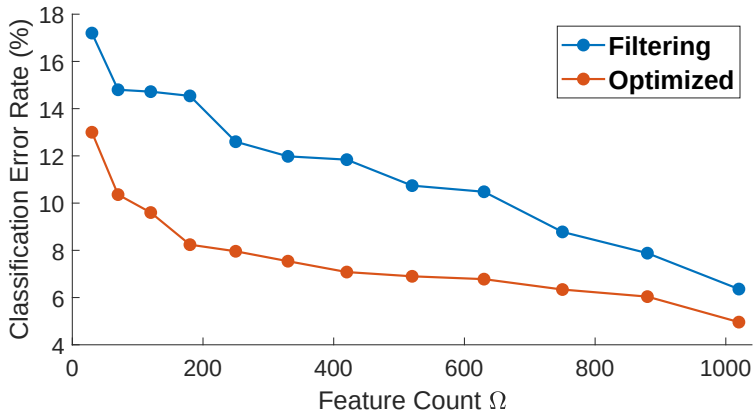


Figure 3.11: Validating classification error rate of the two mining methods during classifier stage using ERF.

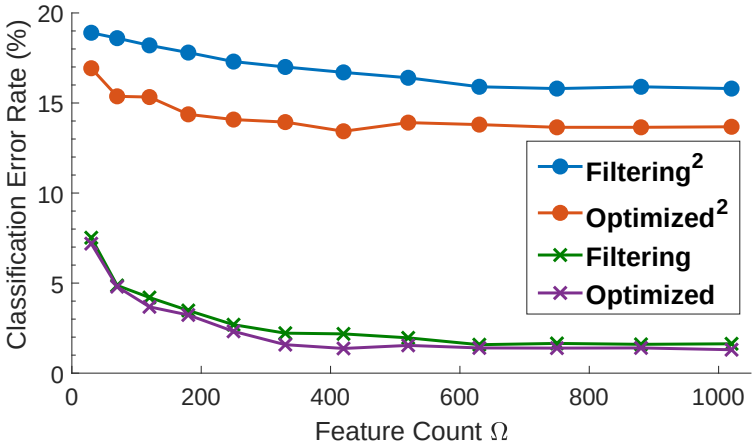


Figure 3.12: Testing classification error rate of the four mining methods before and after gait phase reconstruction using ERF.

proposed optimized feature candidate mining approach compared to the filtering method. The reduction in computation time comes from detecting feature candidate pairs with local extrema linkage before selection instead of using Gini impurity calculations.

Since the filtering feature extraction firstly computes qualities of all potential feature candidate pairs for each gait parameter, the computation time of feature extraction does not depend on the feature count Ω. Unlike the filtering approach, the proposed optimized feature extraction firstly detects the feature candidate pairs by detecting local extrema, and then narrows the potential candidates based on probabilities of

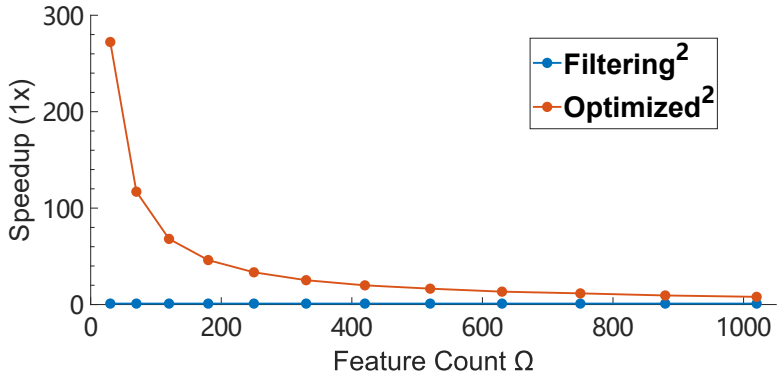


Figure 3.13: Computation performance (1x times of baseline) of the proposed feature extraction methods: the base line computation time is measured using the filtering method with 1020 feature candidate pairs.

different gait phase label combinations. This filters out potential feature candidate pairs before the feature selection process. A slight improvement is also observed in the final classification error rate after the gait phase reconstruction process, as shown in Fig. 3.12.

Finally, this research also compares the computational efficiency with the NARX-NN model-based method, where for fair comparison, the frames with full gait cycles are used only. The parameters for each method are set to achieve their best final classification error rates. The *Speedup* of the three approaches are listed in Table 3.3. Note that the corresponding best results for these three methods are shown in Table 3.2. It can be seen that Optimized method is the fastest method, 2 to 6 times faster than NARX-NN model.

Table 3.3: *Speedup* (1x times of baseline) of the proposed feature extraction methods and NARX-NN model corresponding to their lowest classification error rate.

Method	Filtering	Optimized	NARX-NN
Train	1	18.2	3.1
Test	253	1286	1110

3.4 Summary

This chapter proposes a classification system suitable for clinical diagnostics during rehabilitation program, enabling the possibility to reflect the muscle activities in each phase of gait. Given a gait record/sequence, this research distinguishes which gait phase is a frame associated with. With gait phase information (classes) of each frame, this research facilitates the diagnostics of individual gait phase for physical therapy, thus improving the quality of the overall gait assessment.

To better characterize gait motion and its patterns, 12 gait features (gait parameters) are first proposed drawing from the medical literature based on observations of lower limb movement focusing on foot switch and leg swings. These proposed parameters are then used as input to the proposed multi-channel time-series classification methods that fully utilizes temporal information of gait parameters improving the final classification accuracy. Optimized feature candidate pair detection and mining algorithms (Algs. 8, 9) are proposed to reduce the computational complexity without sacrificing the classification performance. In order to refine the gait phase labels, this work develops a classification approach to relocate the gait events between adjacent gait phases. The validation, conducted using 126 experiments, with 6 healthy volunteers and 9 stroke survivors with manually-labeled gait phases, achieves state-of-art classification accuracy of gait phase with lower computational complexity compared to static threshold based solutions. The overall proposed, frame-wise, multi-channel, time-series classification algorithm demonstrates a significant low classification error rate with respect to the state-of-the-art NARX-NN model.

However, this research still have some drawbacks, such as: (1) a more careful study of "sensitivity of each gait parameter to the classification problem" and "importance of each gait parameter in distinguishing whether a gait frame is in two adjacent frames" which will help to understand how important the associated observation of gait movement is associated with the extraction of gait phase information. (2) more simulations are required to investigate the sensitivity of neighbor radius r_z (input parameter of Alg. 8) in determining whether two given frame is adjacent or not in resulting optimal

feature sets to improve the predictive performance. (3) this research needs to consider the sensitivity of the proposed gait phase features in classifier learning to study whether the features are universal to every state-of-the-art machine learning methods. All these drawbacks can be taken into consideration as further research direction to gait phase classification task.

Chapter 4

Feature Selection and Extraction in Sequence Labeling for Trajectory-based Gait Analysis and Arrhythmia Detection

4.1 Review of Sequence Labeling

Sequence labeling or sequence classification assigns categorical/classification labels to a sequence of data points based either on a set of previous timestamped observations or based on a specific time segment of the signal. Sequence labeling is a pattern recognition task that in general can be: (1) one-to-one: predict an individual label in a sequence using a previous observation. (2) one-to-many: similar to the one-to-one method but with multiple predictions by retaining states from their previous outputs and observations as inputs for the next one. (3) many-to-one: predict a data label for a set of observations from multiple previous observations. (4) many-to-many: assign labels to each member of a subsequence given observations over a period of time.

In this thesis, two sequence labeling applications are studied, including gait phase classification and arrhythmia detection. In the gait analysis application, the objective

of gait phase classification (see Chapter 3) is to assign gait phase label to each frame of a given gait record (joint trajectory sequence). The resulting labeled sequence is then used to assess the kinematics in each gait phase for tracking the rehabilitation progress. In the arrhythmia detection application, each detected segment of ECG signals is classified into different beat classes after detecting the combination of Q,R,S graphical deflections seen on a typical ECG (QRS) complex [24].

The most commonly used types of algorithmic architectures for gait analysis and arrhythmia detection applications are one-to-one and many-to-one model, such as in [108]. In the following sections, the focus is therefore based on one-to-one and many-to-one algorithmic approaches.

In the literature, two types of labeling approaches are commonly investigated: (1) Model based Labeling: Probabilistic model that finds the best matched label via statistical inference. The most commonly used model sequence labeling relies on Markov assumption, that is, the target label is dependent only on the immediately adjacent labels/observations, such as Hidden Markov Model (HMM) and Conditional Random Fields (CRF). (2) Feature Based Labeling: Classification mapping problem that relates the input observations (features) to a categorical label, using techniques such as SVM, RF, DNN.

In the next sections, model-based and feature-based labeling approaches are reviewed in the context of one-to-one and many-to-one architectures, followed by feature selection and extraction in those state-of-the-art methods that are commonly used to solve sequence labeling problems.

4.1.1 Model Based Labeling

Hidden Markov Models

Hidden Markov Model is a statistical model for time-series classification introduced in [109]. Under the assumption that the conditional probability distribution of the future states of the process depends only on the current state [110], the model describes the state transitions and stochastic processes with an underlying set of hidden parameters.

[111] proposed a distributed classifier to assign four gait phases (Flat Foot, Heel Off, Swing, Heel Strike) given a sequence of sagittal lower limb angular velocities during a gait cycle. The angular velocity is measured via three uni-axial gyroscopes embedded in inertial measurement units (IMUs) attached on the three lower limb segments. Similarly, [112], [113] investigate the HMM-based classifier based on motion data captured from gyroscope. The evaluation metric for successful detection of a gait phase is based on walking speed relative tolerance window (30-60 ms) of the labeled gait phases. The results indicated that HMM-based classifier outperforms the threshold-based method [114] using relative high sampling rate gyroscope motion data.

HMM-based classifier is also investigated in cardiac arrhythmia classification [115], [116] to detect and classify three beat categories (normal beats, ventricular arrhythmia, supraventricular arrhythmia) based on a parallel combination of separate models for each beat class. [116] reports that wavelet transform leads to a better characterization than raw signals of ECG records using HMM-based classifier.

Conditional Random Fields

[117] proposed a discriminant probability model Conditional Random Fields (CRF) to tackle the potential victims of label bias problem in Maximum Entropy Markov Model (MEMM). The most commonly used CRF model to solve the sequence labeling task is linear chain CRF model (many-to-one), such as in gait recognition [118] and arrhythmia classification [119].

As in gait recognition [118] task, two-layer MLP classifiers and CRF are trained given two different sets of input features to model dependencies between adjacent frames that finally assign a categorical label to a video. Similarly, this chapter investigates the linear chain CRF-based classifier for gait phase classification given a set of observations from adjacent frames near the target frame to be classified.

[119] proposed a weighted CRF-based classifier to overcome the class unbalance issue via a cost-sensitive objective function and l_1-norm penalty on weights. However, no evaluation is performed using unweighted approaches to compare with the proposed cost-sensitive classifiers. Feature analysis is not involved in [119], instead a combi-

nation of 249 state-of-art features are selected to perform classification. The results indicate that the CRF-based classifier outperforms SVM and LDA classifiers for the intra-subject MIT-BIH Arrhythmia Dataset [120], which is also used in this study.

4.1.2 Feature based Labeling

This section first reviews the widely-used supervised machine learning algorithms for gait phase classification and arrhythmia classification and then reviews the associated feature selection and extraction approaches.

Support Vector Machine

Support Vector Machine is widely used in the literature for the classification task. The basic binary classification model projects the input observations into a metric space, usually specified by a kernel function, and finds the hyperplane that maximizes the margin between the two classes [121]. As in most binary classifiers, to solve the multi-class classification problem, one-vs-all or one-vs-one criteria [122] are used to perform inference during testing.

Random Decision Forest

Random decision forest [97] has been reported in the literature as a practical, high performance classifier that comprises ensemble, bootstrap and bagging to reduce the effects of over-fitting introduced by decision trees [123]. Its performance has also been shown and discussed in Chapter 3 in relation to the gait analysis application.

Convolution Neural Network

Convolution Neural Network [27] is one of the state-of-the-art deep learning neural networks that uses shared-weight filters to extract translation invariant features. By only receiving a restricted subarea of input observations for each neuron, it typically has lower complexity and connectedness compared to the fully-connected design in classical MLP network. Leveraging big data, CNN replaces the hand-crafted feature selection

and extraction into an automatic approach by learning the robust feature maps that contribute to its prediction accuracy during its training process [124].

Recurrent Neural Network

Recurrent Neural Network (RNN) is one of the state-of-the-art methods for modeling sequential data. Each layer of RNN shares the weights across time that iteratively capture dependencies between targets and past observations. However, classical RNN is usually hard to train and often suffers gradient vanishing problem as the number of matrix multiplications increases with the number of time steps (observations). Long short term memory (LSTM) [125] is proposed to solve the gradient vanishing problem by introducing gating mechanisms to allow self-loop flow for an indefinite amount of time. The gated recurrent unit (GRU) [126] makes less restrictive flow inside the unit by using only two gating units, and typically performs comparable performance compared to LSTM. Compared to classical RNN, bidirectional RNN [127] often performs better by considering both past and future of a specified time step.

4.1.3 Feature Selection and Extraction

Regardless of the classifier algorithm, feature selection is one of the most important pre-processing approaches to conduct dimensionality reduction addressing the following aspects in terms of: (1) selecting the most important features from high-dimensional datasets to mitigate the curse of dimensionality, (2) removing features that does not contribute to the prediction accuracy to make interpretation easier and training a simple model in less time, (3) reducing the risk of over-fitting and hence improve generalization, trading-off the bias and variance. The selected subset of original features achieves best performance with respect to some criteria, such as the proposed feature selection method in Chapter 3 for gait phase classification, or QRS complex detection [24] performed to detect the relevant ECG segments of interest in arrhythmia classification. Specifically for trajectory based gait phase classification, to select features, this research only focuses on lower limb joint trajectories considering the prior knowledge from the widely used plug-in-gait model [102]. In other sequence labeling task like “arrhythmia

detection", segments of two leads ECG records are selected by QRS complex detection to detect cardiac diseases [128].

Usually following feature selection, feature extraction is another powerful method to extract the most discriminatory information, transforming the original features into an optimal set of new features. It usually improves the final predictive performance and is conducted either before applying machine learning algorithm or inside the algorithm in feature engineering, such as gait features proposed in Chapter 3, or linear and nonlinear features used in [26], [129]–[131].

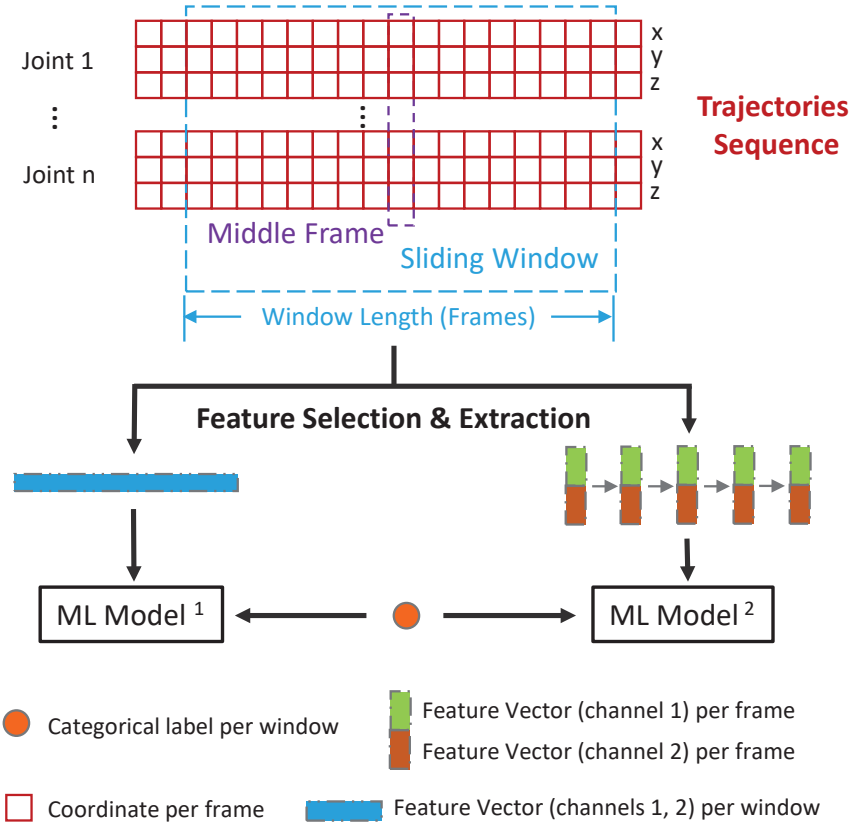


Figure 4.1: Many-to-One Architecture: One label is assigned to each sliding window. A frame refers to seven 3D joint positions, three coordinates (x,y,z) from each joint. ML Model [1] refers to RF, DNN, or CNN classifiers. ML Model [2] refers to HMM, CRF or Bidirectional GRU classifiers.

For time-series features, sliding window is a vital step to represent the sequential pattern and feature selection [132]. In the many-to-one model, the sliding window

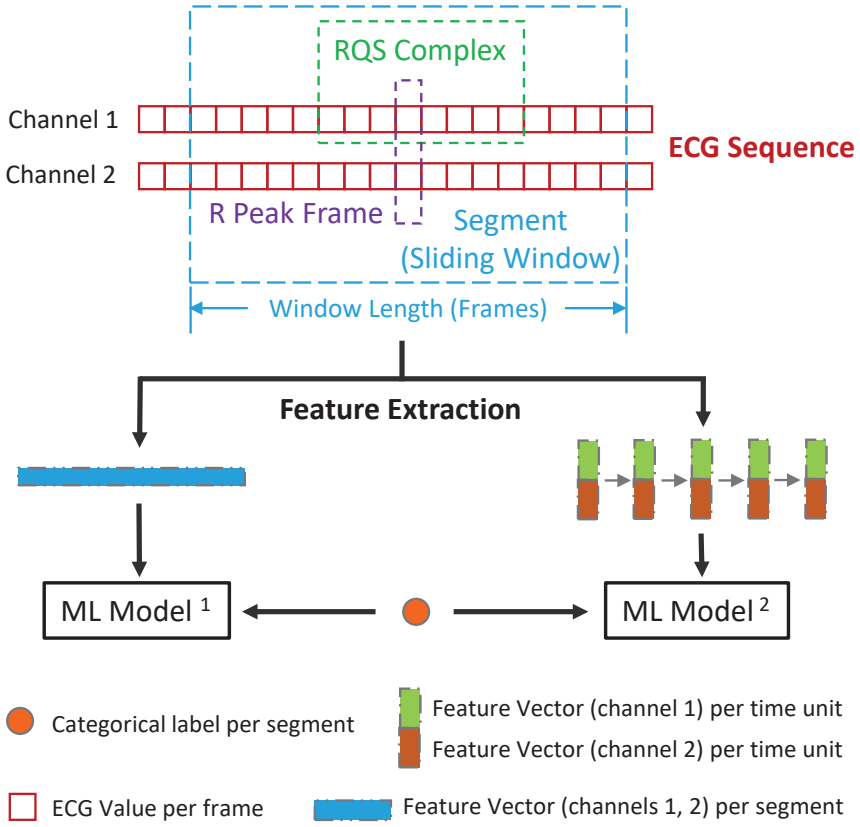


Figure 4.2: Many-to-One Architecture: One label is assigned to each ECG window. A frame refers to two samples, one from each channel. ML Model [1] refers to SVM, RF, DNN, or CNN classifier. ML Model [2] refers to Bidirectional GRU classifier. The time unit of time-series feature vectors that are feed into ML Model [2] depends on the feature extraction algorithms, i.e., time resolution in wavelet transform.

method reconstructs the input observations as a window of observations around the target frame instead of a single set of observations at the target frame. For gait phase classification, the sliding window is chosen based on the target frame that sits in the middle of the window as shown in Fig. 4.1. For arrhythmia classification, the target frame refers to be the R peak frame extracted by QRS detection as shown in Fig. 4.2. The length of the window controls the dimensionality of the input observations into machine learning algorithms, thus directly affecting the performance. Chapter 3 proposes a feature extraction method to extract the distinct observation changes (sequential patterns) within the sliding window. The proposed method offers the advantages of both dimensionality reduction and better classification performance, resulting in simplified

machine learning models. The signal representations of the gait trajectories vary with walking speed thus the sliding window length of a similar sequential pattern is often different.

Unlike gait trajectories features that have varying window length, ECG signals represent biomedical signs showing the working activity of the heart muscle which is often detectable through QRS complex detection with a stable wave/window length. Therefore, in this chapter, distinct feature extraction is performed with variable window length for gait phase classification and fixed window length for arrhythmia classification and the resulting predictive performance evaluated for both classification tasks. The amount of input information associated with the target frame in sequence labeling task varies with the window length prior to the sliding window technique. In this research, the sensitivity to window length for gait phase classification is evaluated, based on the classification performance of different classifier algorithms which has not been investigated in the literature for gait phase classification problem.

4.2 Simulations

This section evaluates the sensitivity of state-of-the-art feature engineering approaches using different sequence labeling classifiers, using inference speed and classification error rate as performance measures. The following two sections describe the simulation setup for gait phase classification and ECG arrhythmia classification, respectively.

4.2.1 Gait Phase Classification

As described in Chapter 3, data is collected for lower limb joint trajectories from 9 stroke survivors and 6 healthy volunteers during normal 4-meter walking test at two different rehabilitation sessions. Each frame within a total of 126 gait records is manually labeled with a categorical label (one of the nine gait phases) by experts which is organized into a dataset, named as ***Gait***. In order to analyze the muscle recovery progression in the rehabilitation program, gait phase classification is performed to automatically label the sequence into gait cycles and 9 gait phases. This facilitates the decision

making when physical therapy evaluation are conducted on kinematics measurements in different gait phases. The feature selection in gait phase classification requires prior domain knowledge of gait phases defined in [64]. Similarly, in [87], the importance of knee angle measurements is analyzed in 4 gait phases classification task based on the classification performance using RF, J-48 Decision Tree (DT) [133], SVM and MLP classifiers. The results demonstrate that the classifier performance increases if knee angle measurements are provided and RF outperforms other classifiers, achieving a lower mean classification error rate. The simulation setup for ***Gait*** dataset is list in Table. 4.1.

Table 4.1: Simulation Setup for ***Gait*** Dataset. There are 9 classes and class distribution shows the number of samples in each class.

Configuration	Description	
Data Scale	126 Records (613 Strides) with 83898 Frames	
Train Test Distribution	Training: 41924, Testing: 41974 Frames, Inter Subjects	
Target	9 Gait Phases (9 Classes)	
Class Distribution	Loading Response 1: 5806, Loading Response 2: 6474, Mid-Stance: 11271, Terminal Stance 1: 10201, Terminal Stance 2: 13915, Pre-swing: 7241, Initial-swing: 11656, Mid-swing: 7290, Terminal Swing: 10024 Frames	
Raw Feature	7 (Joints)×3(x,y,z) Joint Coordinates per Frame	
Feature Map	Normalized Joint Trajectories [107]	7×3 Real Attributes per Frame
	12 Gait Parameters (see Chapter 3)	12 Real Attributes per Frame
Many-to-one Scheme	Sliding Window	Concatenated Feature Maps from Adjacent $1, 10, 15, 20$ Frames per Frame
	Distinct Features (see Sec. 3.2.2 in Chapter 3)	420 Real Attributes per Frame and Vary with the Length of Sliding Window

4.2.2 ECG Arrhythmia Classification

The widely-used MIT-BIH Arrhythmia Dataset [120] (denoted as ***Arrhythmia***) is used to analyze the sensitivity of different state-of-the-art features among classical sequence labeling approaches and deep learning based approaches. The dataset contains 48 half-

hour excerpts of two channel ambulatory ECG recordings from 47 subjects in 1979. The recordings are digitized at 360 samples per second. Experts are asked to manually annotate each beat (approximately 11000 annotated beats). A ten seconds of data and five typical beat types from record 208 of this dataset are illustrated in Fig. 4.3.

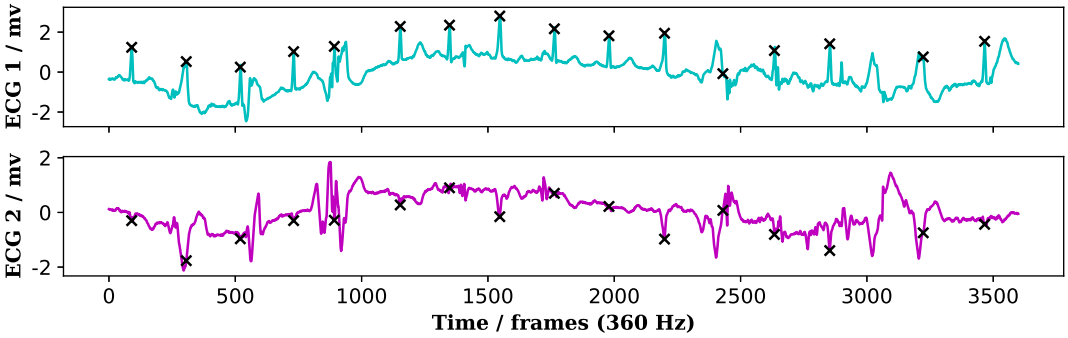
To investigate the intra-subject variability in arrhythmia classification with less attention on the class distribution in intra-subject variability, the whole dataset is split into training, validation, and testing sets with proportions 0.5, 0.15, 0.35, respectively. The simulation setup for ***Arrhythmia*** dataset is listed in Table. 4.2.

Table 4.2: Simulation Setup for ***Arrhythmia*** Dataset

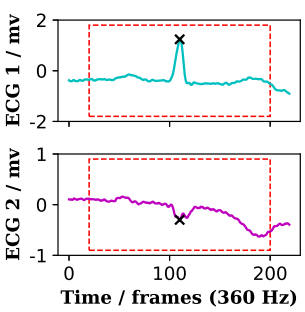
Configuration	Description	
Data Scale	126 Records (613 Strides) with 73854 Segments	
Class Distribution	Training: 44179, Validation: 13242, Testing: 31002 Segments, Intra Subjects	
Target	5 Classes	
Class Distribution	"N": 74683, "V": 6605, "Q": 3880, "S": 2469, "F": 786 Segments	
Raw Feature	2 ECG Channels × 180 (Frames) per Detected QRS Segment	
Feature Map	RR-Intervals (RRI) [129]	4 Real Attributes per Segment
	Fast Fourier Transform (FFT)	35 × 2 Real Attributes per Segment
	Wavelet	23 × 2 Real Attributes per Segment
	Wavelet with Uniform Local Binary Pattern (WaveletULBP) [134]	59 × 2 Real Attributes per Segment
	Higher order statistics (HOS) [26]	4 × 5 × 2 Real Attributes per Segment
	Hermite [135]	22 × 2 Real Attributes per Segment
Many-to-one Scheme	Sliding Window	Concatenated Feature Maps from Adjacent 180 Frames around Segment

4.2.3 Classifier Setup

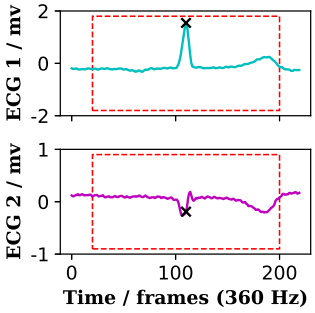
Following the aforementioned simulation settings for ***Gait*** and ***Arrhythmia*** Dataset, the sensitivity of sliding window length and different feature selection methods via HMM, Bidirectional GRU, CRF, RF, DNN, CNN classifiers is evaluated. The deep learning architectures used in this study are shown in Figs. 4.4–4.6 for Bidirectional



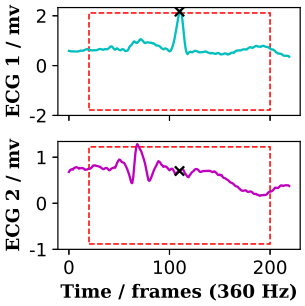
(a) Ten seconds from record 208 of the MIT-BIH Arrhythmia Dataset



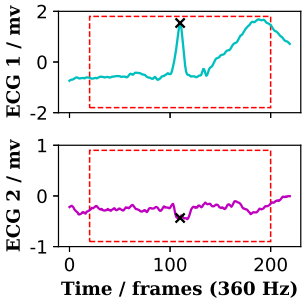
(b) "N": normal sinus rhythm



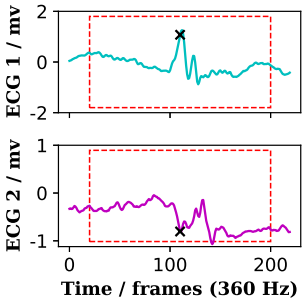
(c) "S": supraventricular ectopic beats



(d) "V": ventricular ectopic beats



(e) "F": ventricular fusion beats



(f) "Q": unknown beats

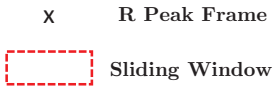


Figure 4.3: Illustration of typical ECG record with two ECG lead signals (denoted ECG 1 and ECG 2) and a subsequence of ECG lead signals for each beat type. QRS detection is first performed to extract the R peak frame shown as 'x'. The two subsequence of ECG lead signals within the 'red dashed' sliding window of 180 frames (the mean R peak interval is 0.5 second which is close to the suggested value in [26], [131]) are selected as the input features for feature extraction and then to be classified as one of the five beat types.

GRU, DNN, CNN classifiers, respectively, in relation to the dimensions of the input observations. The parameters for each architecture is heuristically set based on the findings from [136].

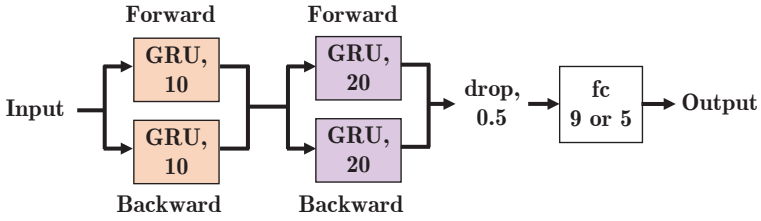


Figure 4.4: Deep learning architecture for bidirectional GRU based classifier. 'GRU' refers to the gated recurrent unit (GRU) based recurrent neural network layer with parameters of hidden state size. 'drop' refers to the dropout layer with keeping ratio. 'fc' means the fully connected layer with number of neurons.

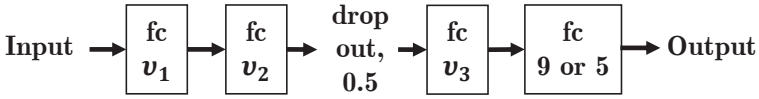
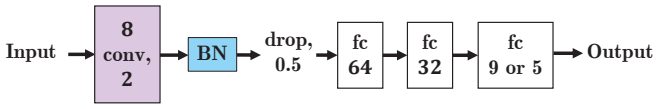


Figure 4.5: Deep learning architecture for DNN based classifier. The number of neurons v_1, v_2, v_3 in fully-connected layers vary dependent on the dimension of the input data (see details in Table. 4.3).

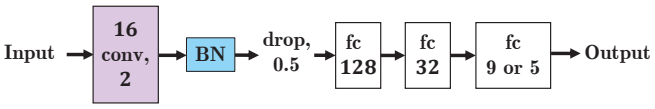
Table 4.3: Hyper-parameters used in DNN-based classifier for both ***Gait*** dataset and ***Arrhythmia*** dataset.

Dimensions of Observation	v_1	v_2	v_3
<8	16	32	None
≥8 and <16	32	64	32
≥16 and <32	64	128	32
≥32 and <64	128	256	64
≥64 and <128	256	512	64
≥128	512	1024	128

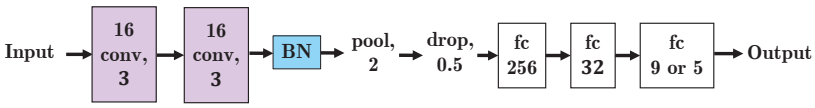
The hyper-parameters used in DNN-based classifier are listed in Tab. 4.3. There are 9 output neurons for gait phase classification and 5 for arrhythmic classifications. Similarly to Sec. 3.3, 30 decision trees with depth of 20 are used for RF-based classifier.



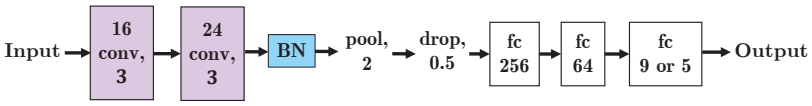
(a) Block diagram of the architecture when number of features <8



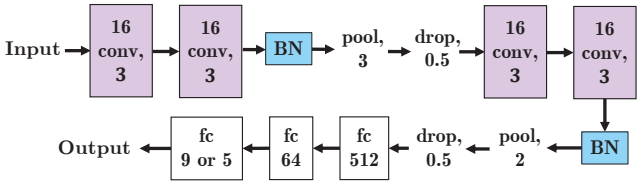
(b) Block diagram of the architecture when number of features ≥8 and <16



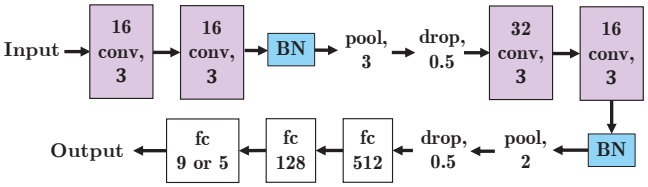
(c) Block diagram of the architecture when number of features ≥16 and <32



(d) Block diagram of the architecture when number of features ≥32 and <64



(e) Block diagram of the architecture when number of features ≥64 and <128



(f) Block diagram of the architecture when number of features ≥128

Figure 4.6: Deep learning architecture for CNN based classifier. 'a conv b' refers to the 1D convolutional layer with parameters of kernel size=b, filter number=a. A constant stride size of 1 is used for 'conv'. 'BN' refers to the batch normalization layer. Hyper-parameters of different layers vary dependent on the dimensions of the input observation.

4.3 Results and discussion

This section first compares the predictive performance of different classifiers using the state-of-the-art feature selection/extraction methods reviewed in Sec. 4.1. Three feature maps are compared, namely: trajectories (raw data as features), gait parameters and distinct features. Following the simulation setup in Sec. 4.2, this section demonstrates the classification error rates, inference speed for ***Gait*** and ***Arrhythmia***. 'HMM' classifier, as a one-to-one algorithmic architecture, predicts categorical label for an individual frame based on its previous observations. Therefore, the length of the sliding window used by HMM classifier is restricted to one frame results. Unlike HMM, the single frame sliding window case is not considered for many-to-one Bidirectional GRU and CRF architectures since they are designed to explore the relation across multiple frames in the sequence. In addition, the sliding window method is used on the resulting distinct features using gait phase feature extraction method proposed in Chapter 3. Those distinct features within the sliding window are selected as the input observations for the target frame. Therefore, this research focuses on the results for multiple frame sliding window cases using distinct features. Note that, distinct features are extracted by feature extraction method proposed in Chapter 3 for gait classification task, thus not applicable to arrhythmic classification task.

For ***Gait*** dataset, as demonstrated in Tab. 4.4, with feature selection, namely the proposed 12 gait parameters in Chapter 3, all classifiers are observed to constantly outperform the ones using normalized trajectories (raw data without feature selection), with respect to when using the mean classification error rate as metric. With increasing window size, more input observations/features are fed into the classifiers, resulting in better predictions. Similar results can also be observed in Figs. 4.7–4.10 using F-Measure as evaluation metrics. Especially for those classifiers (HMM, CRF) with much worse predictive performance, the proposed feature selection method facilitates the classification task. Comparing deep learning based feature extraction approaches, i.e., Bidirectional GRU, DNN, CNN, it can be observed that Bidirectional GRU performs better in modeling gait features (gait parameters). Similar to the feature selection ap-

Table 4.4: Classification Error Rate (%) for ***Gait*** Dataset. * refers to automatic feature extraction approach. **Bold** font is used to present the best metric value across different methods.

Method	Feature Map	Window Size (Frames)			
		1	**10**	**15**	**20**
HMM	Trajectories	51.8	—		
	Gait Params	26.4	—		
Bidirectional GRU*	Trajectories	—	20.0	19.6	18.9
	Gait Params	—	19.2	18.5	18.5
CRF	Trajectories	—	43.4	43.5	44.3
	Gait Params	—	24.9	26.8	27.8
Random Decision Forest	Trajectories	25.9	23.7	23.4	23.0
	Gait Params	20.4	19.6	19.4	19.0
	Distinct Features	—	20.3	18.8	18.2
DNN*	Trajectories	24.2	22.3	21.7	20.4
	Gait Params	23.9	21.4	19.8	19.1
	Distinct Features	—	17.1	16.3	16.1
CNN*	Trajectories	22.8	20.6	19.0	18.5
	Gait Params	21.7	20.5	19.0	18.5
	Distinct Features	—	**16.4**	**15.9**	**15.7**

proach proposed in Chapter 3, the bidirectional GRU model enables learning from both forward and backward temporal information, resulting in better predictive performance. With distinct features, DNN and CNN further learn the relations of temporal features with significant performance improvements by deep learning based feature extraction.

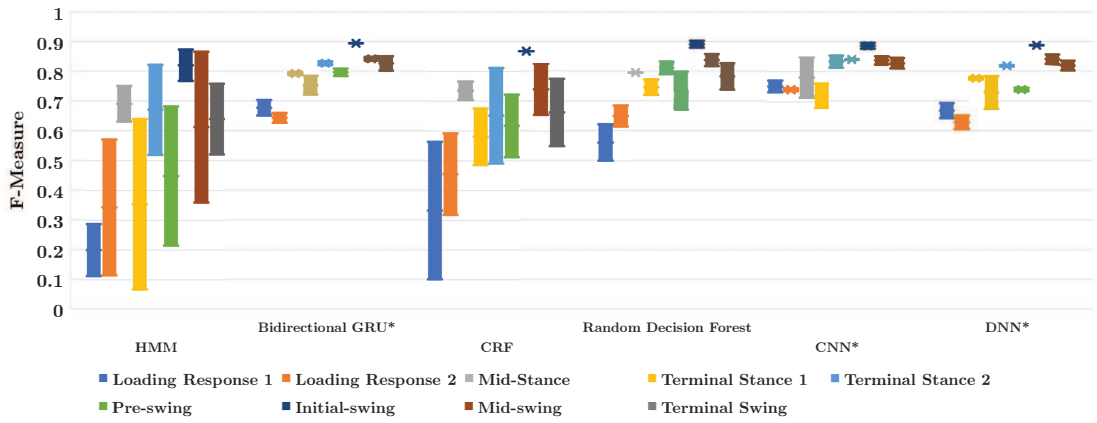


Figure 4.7: F-Measure Increments for Each Class in ***Gait*** Dataset using Feature Selection without sliding window. * refers to deep learning based feature extraction approach.

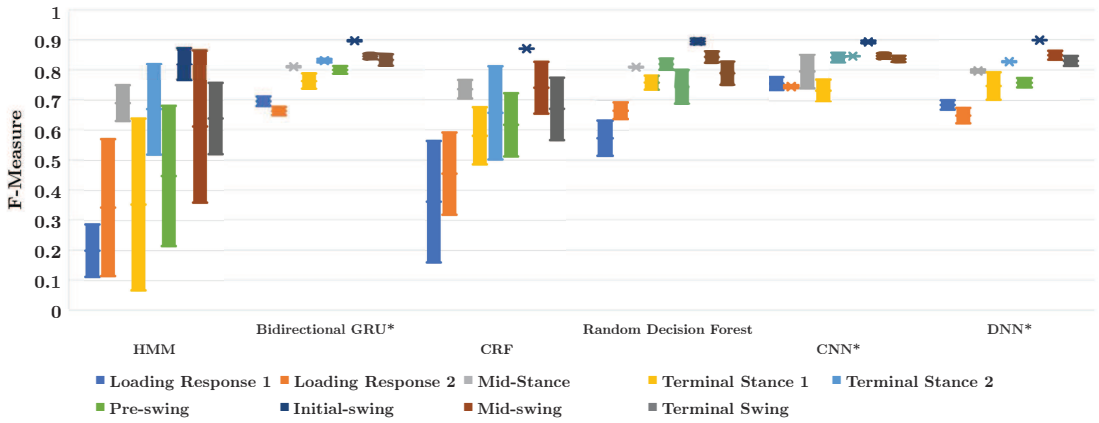


Figure 4.8: F-Measure Increments for Each Class in ***Gait*** Dataset using 10-Frame Sliding Window and Feature Selection. * refers to deep learning based feature extraction approach.

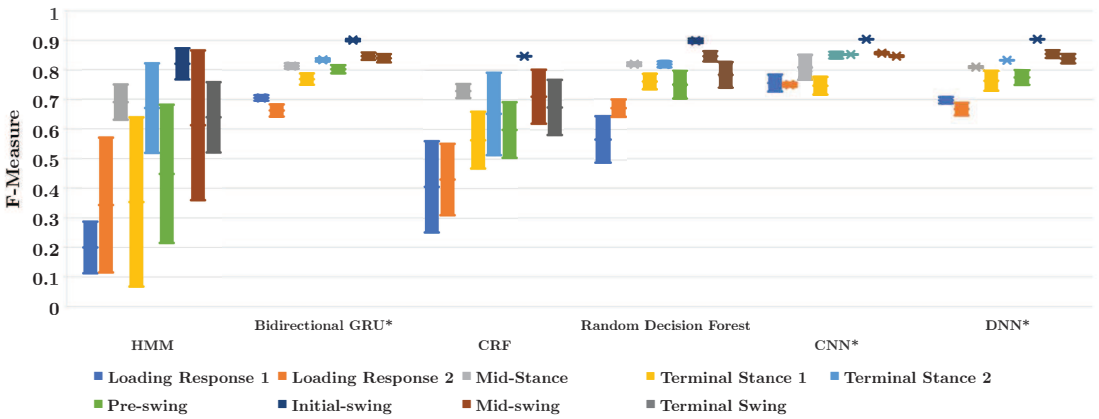


Figure 4.9: F-Measure Increments for Each Class in ***Gait*** Dataset using 15-Frame Sliding Window and Feature Selection. * refers to deep learning based feature extraction approach.

To further analyze the sensitivity of feature extraction, F-measure demonstrates the improvements as shown in Fig. 4.11, by comparing the results before and after feature extraction method proposed in Chapter 3. A significant improvement is observed for relatively rare gait phase classes, such as loading response 1, loading response 2 and pre-swing.

Unlike gait analysis application whose task is to classify individual frame into a categorical label/class, arrhythmia classification is performed on a sub-sequence of time-series data. For ***Arrhythmia*** dataset, this research shows the classification error

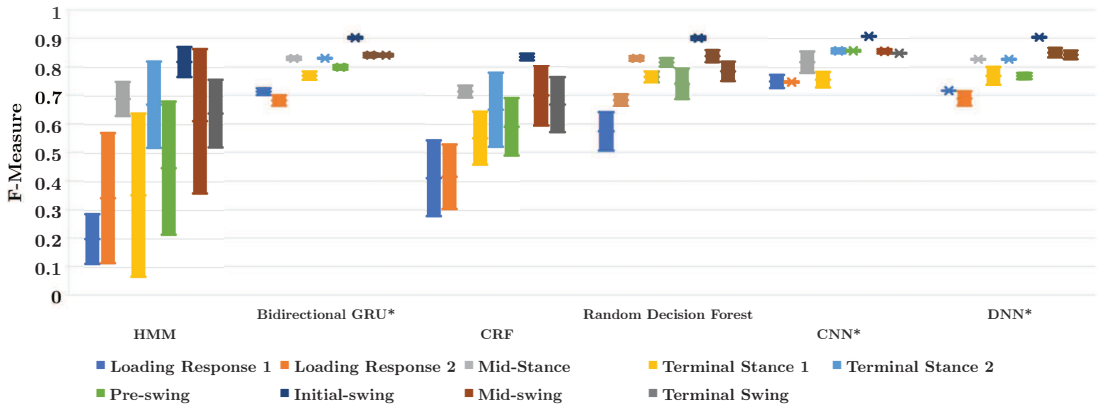


Figure 4.10: F-Measure Increments for Each Class in ***Gait*** Dataset using 20-Frame Sliding Window and Feature Selection. * refers to deep learning based feature extraction approach.

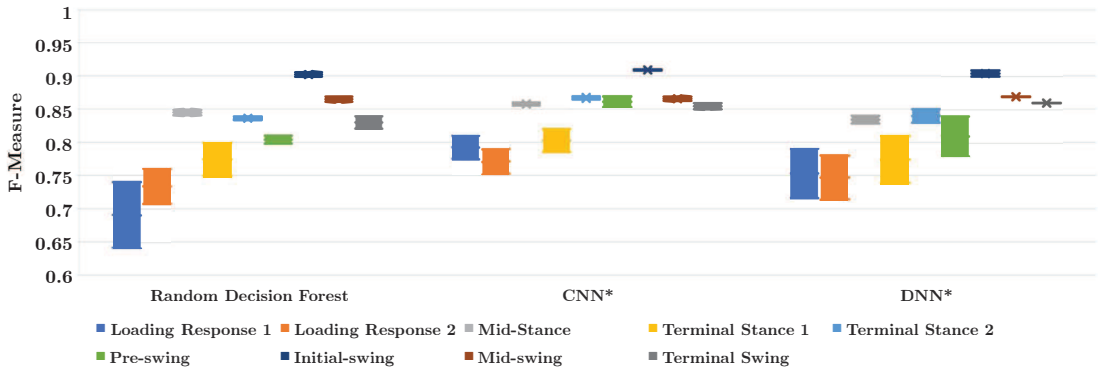


Figure 4.11: F-Measure Increments for Each Class in ***Gait*** Dataset using Feature Selection and Distinct Feature Extraction. The increments are visualized based on the F-measure using proposed gait parameters and distinct feature extraction methods. * refers to deep learning based feature extraction approach.

rates and F-measure for each class in Tabs. 4.5 and 4.6, respectively. It is clear that zero-mean standardized ECG signals outperform sensitive feature extractions across all benchmarked classifiers. Both SVM classifier and deep learning based classifiers achieve high predictive performance. Deep learning based classifiers that automatically extract deep features achieve comparable mean classification error rate and F-measure using Wavelet and HOS feature extraction methods.

Table 4.5: Classification Error Rate (%) for Each Class in ***Arrhythmia*** Dataset. * refers to deep learning based feature extraction approach.

Method	Feature Map	Arrhythmia Classes					
		N	S	V	F	Q	Mean
Bidirectional GRU*	Raw	0.6	25.6	5.6	25.2	0.5	11.5
	Wavelet	0.5	33.8	5.1	24.7	0.6	13.0
SVM	Raw	5.1	12.1	3.8	8.9	0.4	**6.0**
	RRI	1.3	98.6	92.4	85.1	56.8	66.8
	FFT	0.0	100	100	100	100	80.0
	Wavelet	0.0	60.9	96.5	96.8	97.6	70.4
	WaveletULBP	35.8	37.2	39.1	27.7	19.9	31.9
	HOS	1.0	15.7	3.3	22.7	4.0	9.3
	Hermite	36.6	15.5	27.2	21.6	5.4	21.3
Random Decision Forest	Raw	0.1	40.2	6.1	38.8	1.0	17.3
	RRI	1.6	31.1	20.4	77.0	14.2	28.9
	FFT	0.6	59.8	51.8	96.3	84.7	58.6
	Wavelet	0.1	38.9	5.7	37.1	0.9	16.5
	WaveletULBP	0.4	58.4	32.9	48.0	28.9	33.7
	HOS	0.1	35.1	5.7	35.6	2.2	15.7
	Hermite	0.3	36.6	6.6	30.1	0.73	14.8
DNN*	Raw	0.4	12.3	1.9	19.1	0.4	**6.8**
	RRI	0.4	95.2	74.5	100	100	74.0
	FFT	1.7	66.2	54.3	86.4	70.5	55.8
	Wavelet	0.4	17.3	3.9	19.7	0.4	8.4
	WaveletULBP	1.4	45.3	21.9	39.9	15.7	24.8
	HOS	0.5	18.3	2.8	20.1	0.4	8.4
	Hermite	0.6	35.7	5.1	32.5	1.2	15.0
CNN*	Raw	0.3	10.6	3.0	16.7	0.4	**6.8**
	RRI	3.2	56.5	39.9	100	60.4	52.0
	FFT	1.4	57.7	56.7	83.1	82.1	56.2
	Wavelet	0.4	13.4	2.7	17.7	0.4	**6.9**
	WaveletULBP	1.1	56.3	46.8	41.2	24.3	33.9
	HOS	0.4	12.1	2.7	16.0	0.3	**6.3**
	Hermite	0.6	24.6	4.4	29.6	0.6	12.0

Table 4.6: F-Measure for Each Class in ***Arrhythmia*** Dataset. * refers to deep learning based feature extraction approach.

Method	Feature Map	Arrhythmia Classes					
		N	S	V	F	Q	Mean
Bidirectional GRU*	Raw	0.99	0.80	0.95	0.80	0.99	0.91
	Wavelet	0.99	0.77	0.95	0.81	0.99	0.90
SVM	Raw	0.97	0.57	0.95	0.74	0.99	0.84
	RRI	0.92	0.03	0.14	0.17	0.56	0.36
	FFT	0.92	0.00	0.00	0.00	0.00	0.18
	Wavelet	0.92	0.56	0.07	0.06	0.05	0.33
	WaveletULBP	0.77	0.29	0.44	0.10	0.51	0.42
	HOS	0.99	0.85	0.94	0.84	0.98	0.92
	Hermite	0.77	0.16	0.80	0.15	0.97	0.57
Random Decision Forest	Raw	0.99	0.75	0.96	0.74	0.99	0.89
	RRI	0.97	0.79	0.82	0.32	0.86	0.75
	FFT	0.94	0.57	0.61	0.07	0.26	0.49
	Wavelet	0.99	0.76	0.96	0.76	0.99	0.89
	WaveletULBP	0.97	0.58	0.77	0.66	0.80	0.76
	HOS	0.99	0.78	0.96	0.77	0.99	0.90
	Hermite	0.99	0.77	0.95	0.80	**1.00**	0.90
DNN*	Raw	0.99	0.90	**0.98**	0.86	**1.00**	**0.95**
	RRI	0.92	0.09	0.38	0.00	0.00	0.28
	FFT	0.94	0.47	0.57	0.21	0.41	0.52
	Wavelet	0.99	0.86	0.97	0.86	0.99	0.94
	WaveletULBP	0.97	0.67	0.82	0.68	0.86	0.80
	HOS	0.99	0.86	0.97	0.86	1.00	0.94
	Hermite	0.99	0.75	0.95	0.79	0.99	0.89
CNN*	Raw	**1.00**	**0.91**	0.97	**0.87**	**1.00**	**0.95**
	RRI	0.94	0.59	0.67	0.00	0.44	0.53
	FFT	0.94	0.55	0.55	0.24	0.28	0.51
	Wavelet	0.99	0.89	0.97	**0.87**	0.99	**0.95**
	WaveletULBP	0.96	0.59	0.67	0.59	0.79	0.72
	HOS	0.99	0.89	0.97	**0.87**	**1.00**	**0.95**
	Hermite	0.99	0.81	0.96	0.80	0.99	0.91

Note that, the inference speed decreases and model size grows rapidly with the increasing dimensionality of input observations/features, such as the increasing window size for ***Gait*** dataset. Especially for deep learning based classifiers, a deeper architecture that automatically extracts deeper features suffers with significant longer inference time shown in Tabs. 4.7 and 4.8, limited by the size of model and computation capacity. Given sufficient training data, deep learning classifiers with higher model capability achieve better generalization and inference performance without considering domain knowledge.

Table 4.7: Inference Speed(microseconds/frame) for ***Gait*** Dataset. * refers to automatic feature extraction approach.

Method	Feature Map	Window Size (Frames)			
		1	**10**	**15**	**20**
HMM	Trajectories	1.2	—		
	Gait Params	**0.7**	—		
Bidirectional GRU*	Trajectories	—	2467.8	4522.2	6288.9
	Gait Params	—	2407.6	4351.7	5951.3
CRF	Trajectories	—	85.6	141.9	186.0
	Gait Params	—	84.1	137.1	181.1
Random Decision Forest	Trajectories	1.3	4.6	5.5	8.3
	Gait Params	1.0	4.3	5.1	7.5
	Distinct Features	—	6.3	12.5	18.3
DNN*	Trajectories	123.2	1361.7	1891.3	3002.5
	Gait Params	111.0	1102.9	1665.3	2541.4
	Distinct Features	—	2010.5	3023.7	5914.8
CNN*	Trajectories	82.2	747.4	1254.4	2521.7
	Gait Params	62.6	649.8	1027.3	2109.5
	Distinct Features	—	1223.2	2301.8	4633.1

Table 4.8: Inference Speed(microseconds/frame) for ***Arrhythmia*** Dataset. * refers to automatic feature extraction approach.

Feature Map	Method				
	Bidirectional GRU*	SVM	RF	DNN	CNN
Raw	33258.9	8275.1	8.0	838.4	1366.3
RRI	—	2803.2	3.6	428.5	555.5
FFT	—	5375.5	3.9	513.8	935.6
Wavelet	3673.4	2810.5	3.7	505.6	832.8
WaveletULBP	—	6632.8	4.0	575.1	1132.9
HOS	—	539.7	3.6	536.9	854.4
Hermite	—	2944.3	3.6	567.5	942.7

4.4 Summary

This chapter reviews the state-of-the-art feature selection and extraction methods in sequence labeling tasks of trajectory based gait analysis and cardiac diseases detection. The main contribution of this chapter is a rigorous evaluation, in terms of classification error rate, F-measure and inference speed, the sensitivity of feature selection and feature selection including sliding windows on model-based learning and feature-based learning classifiers with one-to-one and many-to-one architectures. The simulation results indicate that the influence of feature selection is dependent on the data or application, for instance, gait phase classification is strongly influenced by the feature selection/extraction, sliding window length which achieves lower classification error rate and higher F-measure. However this observation is not true for arrhythmia classification task, the state-of-the-art selection/extraction results in losing discriminative features to achieve comparable machine learning model in terms of classification performance. It is found that automatic feature extraction approaches, i.e., Bidirectional GRU, DNN and CNN, outperform HMM, RF and SVM. However, those deep learning-based classifiers often operate slower at inference, thus require to be accelerated by modern computing devices such as GPUs. Although a rigorous evaluation is conducted for these two applications, the question of how feature selection and extraction methods contribute to the sequence labeling accuracy is still open for other applications, such as gait recognition, natural language processing, etc.

Chapter 5

Robust Graph-based Classifier Learning

5.1 Review of Robust Classifier Learning

This section first provides an overview of the related work on robust graph-based classifier learning and robust DNN-based classifier learning, in the presence of "noisy labels". Then it discusses graph-based methods integrated with DNN that do not consider noisy data. Finally, the weaknesses of the state-of-the-art classifiers are discussed to motivate the present work.

5.1.1 Robust Graph-based Learning

A label propagation method is proposed in [137] to evenly spread, throughout the graph, label distributions from selected labeled nodes, which are usually noisy and with heuristic information. A KNN-sparse graph-based semi-supervised learning approach is proposed in [138] to remove most of the semantically-unrelated edges and adopt a refinement strategy to handle noisy labels.

To achieve more robust binary classification, in [139], negative edge weights are introduced into the graph to separate the nodes in two different clusters. A perturbation matrix is found to perform generalized GLR for binary classification via iterative re-

weighted least squares strategy [140]. The results demonstrate the applicability of negative edge weights for robust graph-based classifier learning for small amount of data without learning feature representation. This research evaluates this approach when sufficient but noisy training labels are provided.

5.1.2 Robust DNN-based Classifier Learning

Many studies investigate methods to accommodate a wide range of label noise levels and types, which often focus on data augmentation, network design-based regularization and loss correction. Data augmentation techniques have been successfully used in [141]–[143] to automatically annotate unlabeled samples and use these samples for retraining. In [144], training examples are assigned weights by a proposed meta-learning algorithm to minimize the loss on a clean unbiased validation set based on gradient direction.

The effectiveness of dropout regularization for cleaning noisy labels is shown in [145]. For image classification, the annotations obtained from web sources when building a very large image classification dataset are usually partial or unreliable [146], [147]. Based on these findings, [148] indicate that increasing the batch size and downscaling the learning rate is a practical approach to mitigate the effects of label noise, as noisy labels roughly cancel out and only a small learning signal remains.

The loss correction approach of [38] proposes a boosting algorithm 'SavageBoost' that is less sensitive to outliers and converges faster than conventional methods, such as Ada, Real, or LogitBoost. A dimensionality-driven learning strategy is discussed in [30] to avoid overfitting by identifying the transition from an early learning stage of dimensionality compression to an overfitting learning stage when the local intrinsic dimensionality steadily increases. Unlike the above loss correction studies to handle noisy and incomplete labeling, [39] use a combination of training labels and the prediction from the current model to update the training targets and perform weakly-supervised learning. Similarly, [149] integrate the Expectation-Maximization (EM) algorithm into CNN to detect and correct noisy labels, but require a properly pre-trained model. [150] propose an iterative learning framework to facilitate 'robustness to label noise' classifier learning by jointly performing iterative label detection, discriminative feature learning

and re-weighting.

5.1.3 Graph-based classifier learning with DNN

Recent years have seen integration of graph-based learning with deep learning. Given a fixed graph structure, [151]–[153] design CNNs for feature learning by feeding a polynomial of the graph Laplacian. [154] adopt edge convolution to learn combinational spatial features from neighboring nodes given a fixed skeleton graph. Based on the ideas of edge convolution, [155] propose a deeper CNN model to learn the underlying KNN graph structure of point cloud data by iteratively updating the graph. The results demonstrate the capability of edge convolution for feature generalization on point-cloud data.

The problem of insufficient data or incorrect training labels has not been investigated in the above graph-based hybrid methods for classifier learning. For incomplete or imprecise categories of tags (observations) in the training samples, [156] combine CNN and GLR using the sum of the cross-entropy loss and the GLR term for multi-label image annotation, where CNN is used to construct the fully-connected similarity-based graph. Unlike [156], [49] integrates GLR into CNN with a graph-based loss correction function to tackle the problem of insufficient training samples through semi-supervised graph learning.

5.1.4 Novelty with respect to reviewed literature

To enhance the classifier learning against the potential noisy training labels in gait phase classification task caused by mislabeling the gait event frames, this chapter proposes a generalized end-to-end CNN-based approach to learn a robust classifier. Based on the work in [49] that introduces the GLR operator to regularize a CNN given insufficient training data, the proposed approach performs iteratively GLR (similar to [157]) as a classifier signal restoration operator, update the underlying graph to better reflect the node-to-node correlation and regularize CNNs by introducing a graph-based loss correction function to pay less attention on those relatively less reliable training samples. Compared to the reviewed graph-based classifiers and robust DNN-based

classifiers, the proposed approach learns deep feature representations by: (1) adopting edge convolution to aggregate features from neighbors that are labeled in the same class, resulting in smoothed edge weights. (2) assigning the degree of freedom for learning the underlying data structure that better reflects the node-to-node correlation. (3) iteratively updating graph and operating GLR to perform label restoration and semi-supervised classification at the same time that helps to regularize the CNN training without an extra clean dataset. As a result, the proposed approach brings together the regularization benefits of GLR and the proposed loss functions to perform more robust deep metric learning. This study also adopts a rank-sampling strategy to find those training samples with high predictive performance, resulting in lower label noise level in the training set which further benefits the inference.

5.2 Robust Deep Graph Based Classifier Learning

This section first introduces notation and formulates the robust classifier learning problem following the related work [41], [139], [157]–[159]. Then, it describes the main concept behind the proposed Dynamic Graph Laplacian Regularization (DynGLR) neural network that learns robust deep feature map to effectively perform GLR when parts of the labeled data available to train the model are noisy.

5.2.1 Problem Formulation and Notation

Given a set of observations $\mathbf{X} = \{x_1, \ldots, x_N\}$, where $x_i \in \mathcal{R}^n$, $i = 1, \ldots N$, the task of a binary classifier is to learn an approximate mapping function that maps each observation $x \in \mathbf{X}$ into a corresponding binary discrete variable $y \in \mathbf{Y} = \{y_1, \ldots, y_N\}$, called classification label, where $y_i \in \{-1, +1\}$, $i = 1, \ldots, N$.

Let $\dot{\mathbf{Y}}^0 = \{-1, 1\}^M = \{y_1, \ldots, y_M\} \subset \mathbf{Y}$, $0 < M < N$, be a set of known (possibly noisy) labels that correspond to instances $\dot{\mathbf{X}} = \{x_1, \ldots, x_M\} \subset \mathbf{X}$ used for training. Let $\mathbf{Y}^0 = \{\dot{\mathbf{Y}}^0, \mathbf{0}^{N-M}\}$, where all $N - M$ unknown labels are set to zero (to be estimated during testing).

Given $\mathbf{X}$, the problem addressed in this study, is to learn the robust mapping func-

tion to assign a classification label to each observation $x \in \mathbf{X}$ when some classification labels $y \in \dot{\mathbf{Y}}^0$, used for training the model, are incorrect.

Let $\mathbf{G} = (\mathbf{\Psi}, \mathbf{E}, \mathbf{W})$ be an undirected graph, where $\mathbf{\Psi} = \{\psi_1, \ldots, \psi_N\}$ is a set of nodes, each corresponding to one instance in $\mathbf{X}$, $\mathbf{E} = \{e_{i,j}\}, i, j \in \{1, \ldots, N\}$, is a matrix representing the edge connectivity of $\mathbf{G}$; that is, $e_{i,j} = 1$ if there is an edge connecting vertices i and j and $e_{i,j} = 0$ otherwise; and each entry $w_{i,j}$ in the weight matrix $\mathbf{W} = \{w_{i,j}\}, i, j \in \{1, \ldots, N\}$ corresponds to the weight associated with edge $e_{i,j}$. Then, $\mathbf{Y}^0$ can be seen as a graph signal that indexes the graph $\mathbf{G}$. The combinatorial graph Laplacian matrix is given by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{A}$ is a symmetric $N \times N$ adjacency matrix with each entry $a_{i,j} = max(w_{i,j} \cdot e_{i,j}, w_{j,i} \cdot e_{j,i})$, and $\mathbf{D}$ is a degree matrix with entries $d_{i,i} = \sum_{j=1}^{N} a_{i,j}$, and $d_{i,j} = 0$ for $i \neq j$.

Similarly to [159], *triplets* are defined as observations (x_a, x_p, x_n), $x_a, x_p, x_n \in \mathbf{X}$ corresponding to vertices $\psi_a, \psi_p, \psi_n \in \mathbf{\Psi}$, respectively, such that $y_a = y_p \neq y_n$, and $y_a, y_p, y_n \in \dot{\mathbf{Y}}$. Let $\mathbf{P}$ be a set of all edges $e_{a,p}$, such that $y_a = y_p$, and $\mathbf{Q}$ a set of all edges $e_{a,n}$, for which $y_a \neq y_n$, that is, $\mathbf{P}$ and $\mathbf{Q}$ are sets of all edges that connect nodes with the same and opposite labels, respectively.

Motivated by CNNs ability to extract discriminative features and GLRs to 'clean' unreliable labels, this research formulates graph-based classifier learning as a two-stage learning process: (1) *graph learning* - extract deep feature maps, i.e., find a deep metric function that returns the most discriminative feature maps, and then generate an initial graph by learning the underlying $\mathbf{E}$ to maximize/minimize similarity between any two nodes in $\mathbf{G}$ that are indexed by the same/opposite labels. (2) *classifier learning* - iteratively refine the graph and effectively performing GLR to restore the corrupted classifier signal.

5.2.2 Initialization

Given the observation sets $\mathbf{X}$ and corresponding, potentially noisy labels, $\dot{\mathbf{Y}}^{\mathbf{0}}$, the first task is to learn a discriminative feature map $\mathcal{V}^0(\cdot)$ and generate an initial underlying graph for the learnt feature map.

Let

$$d_{i,j}(\mathcal{V}^0) = \|\mathcal{V}^0(x_i) - \mathcal{V}^0(x_j)\|_2^2,$$

be the Euclidean distance between the corresponding feature maps. For a node $\psi_i \in \mathbf{\Psi}$, let $\mathcal{E}_i$ be a set containing all vertices except ψ_i in ascending order with respect to the metric $d_{i,k}(\mathcal{V}^0)$, $k = 1, \ldots, N-1, k \neq i$. Let $\mathcal{S}_i$ be a subset of $\mathcal{E}_i$ containing the first γ_i elements of $\mathcal{E}_i$, that is, the set $\mathcal{S}_i$ contains γ_i most correlated vertices to vertex ψ_i according to metric $d_{i,k}(\mathcal{V}^0)$.

To effectively perform GLR, as in [139], [157], the underlying graph should be a sparsely connected graph. To control the sparsity of the resulting graph whilst maintaining connectivity, the proposed model uses an indicator operator to minimize the number of $\mathbf{Q}$ edges. A typical option is a KNN indicator that keeps only a maximum of γ_i edges for each individual node i, and sets others to zero. That is, each graph edge $e_{i,j}$ is set to:

$$e_{i,j} = \begin{cases} 1, & \text{if } \psi_i \in \mathcal{S}_j \text{ or } \psi_j \in \mathcal{S}_i \\ 0, & \textit{otherwise.} \end{cases} \tag{5.1}$$

Once an optimal edge matrix $\mathbf{E}^0 = \{e^0_{i,j}\}$ is computed through $\mathcal{V}^0(\cdot)$ and Eq. 5.1, an initial undirected and unweighted graph $\mathbf{G}^0 = (\mathbf{\Psi}, \mathbf{E}^0, \mathbf{W}^0 = 1)$ can be obtained. The block diagram is shown in Fig. 5.1. Note that, in the implementation, $\gamma_1 = \cdots = \gamma_N = \gamma^0$ is the initial condition, that is learnt as explained in Sec. 5.3.1.

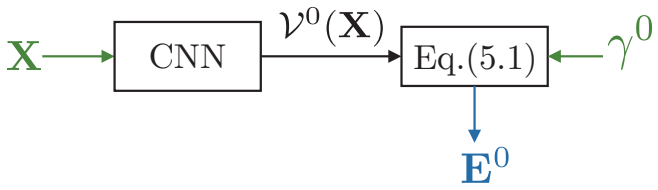


Figure 5.1: The block diagram of the unweighted graph generation scheme. $\mathcal{V}^0(\cdot)$ is a CNN-based feature map learnt by minimizing a loss function in order to reflect the node-to-node correlation. The implementation of the proposed CNN and loss function is described in Sec. 5.3.1.

5.2.3 Proposed Classifier Learning with Iterative Graph Update

If the noisy training labels are seen as a piece-wise smooth graph signal, $\mathbf{Y}^0$, then one can iteratively perform GLR for denoising the labels and performing semi-supervised classification, while refining the set of deep feature maps and the underlying graph.

Let $r > 0$ be the iteration index, initialized to 1, and let $\mathbf{G}^r = (\mathbf{\Psi}, \mathbf{E}^{r-1}, \mathbf{W}^r)$ be the graph, with $\mathbf{W}^r$ to be learnt, and $\mathbf{Y}^r$ the noisy labels in the r-th iteration. Thus, $\mathbf{G}^1$ is an N-node graph with edges set by Eq. 5.1. In the r-th iteration, each vertex ψ_i is indexed by a label $y_i^{r-1} \in \mathbf{Y}^{r-1}$ (graph signal), and is associated to a feature vector $\mathcal{V}^r(x_i)$.

Typically, the edge weight is computed using a Gaussian kernel function with a fixed scaling factor σ, i.e., $\exp\Big(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\Big)$, to quantify the node-to-node correlation. Instead of using a fixed σ as in [41], [139], [158], motivated by [160], the proposed model introduce an auto-sigma Gaussian kernel function to assign edge weight $w_{i,j}^r$ in $\mathbf{G}^r$ by maximizing the margin between the edge weights assigned to **P**-edges and **Q**-edges, as:

$$\begin{aligned} \sigma^* &= \arg\max_{\sigma} \Big[\exp\Big(-\frac{\omega^2_{\{\psi_a,\psi_p\}}}{2\sigma^2}\Big) - \exp\Big(-\frac{\omega^2_{\{\psi_a,\psi_n\}}}{2\sigma^2}\Big)\Big] \\ w_{i,j}^r &= \exp\Big(-\frac{\|\mathcal{V}^r(x_i) - \mathcal{V}^r(x_j)\|_2^2}{2\sigma^{*2}}\Big) \end{aligned} \tag{5.2}$$

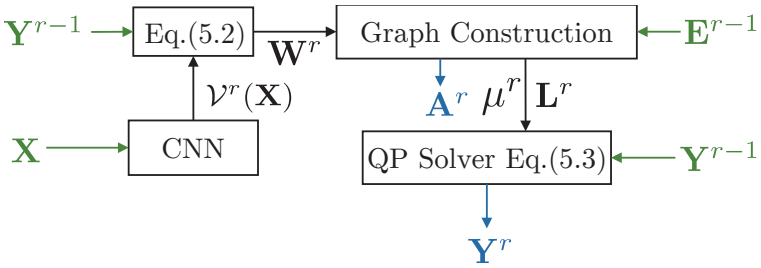
where $\omega_{\{\psi_a,\psi_p\}}$ and $\omega_{\{\psi_a,\psi_n\}}$ compute the mean Euclidean distances between nodes connected by **P**-edges and **Q**-edges, respectively. By setting the first derivative to zero, one can obtain the resulting optimal $\sigma^* = \sqrt{\frac{\omega^2_{\{\psi_a,\psi_n\}} - \omega^2_{\{\psi_a,\psi_p\}}}{2\log(\omega^2_{\{\psi_a,\psi_n\}}/\omega^2_{\{\psi_a,\psi_p\}})}}$, which is used to assign edge weights of the graph.

Closely following related work [139], [157], one can obtain the restored classifier signal by finding the smoothest graph signal $\mathbf{Y}^r$ as:

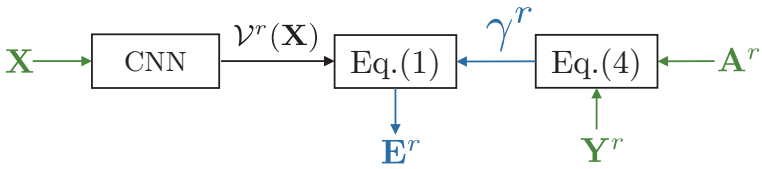
$$\mathbf{Y}^r = \arg\min_{\mathbf{B}} \ (\|\mathbf{Y}^{r-1} - \mathbf{B}\|_2^2 + \mu^r \mathbf{B}\mathbf{L}^r\mathbf{B}^T). \tag{5.3}$$

The minimization above finds a solution that is close to the observed set of labels in the previous iteration, $\mathbf{Y}^{r-1}$, while preserving piece-wise smoothness. To guarantee that the solution $\mathbf{Y}^r$ to the quadratic programming (QP) problem Eq. 5.3 is numerically

stable, Theorem 1 ([157] is adopted by setting an appropriate conditional number κ. The maximum value of the smoothness prior factor μ^r is then calculated as: $\mu^r_{max} = (\kappa - 1)/(2d^r_{max})$, where d^r_{max} is the maximum degree of the vertices in graph $\mathbf{G}^r$. See Fig. 5.2(a).



(a) The block diagram of the proposed classifier scheme. CNN is learnt by minimising the loss function to better reflect the node-to-node correlation. The edge matrix $\mathbf{E}^{r-1}$ is used as a mask when assigning edge weights to construct adjacency matrix $\mathbf{A}^r$. The proposed scheme performs GLR to restore the corrupted classifier signal $\mathbf{Y}^{r-1}$ given the resulting sparse graph Laplacian $\mathbf{L}^r$ and apply the constrained smoothness prior factor μ^r to ensure the numerical stability of QP solver. The implementation of $\mathcal{V}^r(\cdot)$ varies depending on the data scale and the dimension of the input observations. The output is the new set of 'denoised' labels $\mathbf{Y}^r$.



(b) The block diagram of the proposed graph update scheme. Based on the adjacency matrix $\mathbf{A}^r$ and restored classifier signal $\mathbf{Y}^r$, the proposed scheme learns a CNN to better refine the graph structure. The edge matrix $\mathbf{E}^r$ is updated via Eq. 5.4 and Eq. 5.1 based on both the previous restored classifier signal and the regularised deep feature map. The output of this block is the new edge matrix $\mathbf{E}^r$ that will be used in the next iteration.

Figure 5.2: The proposed graph-based classifier and graph update scheme. The green and blue colors denote input and output, respectively. The implementation details are given in Sec. 5.3.2.

Between each two GLR iterations, this research uses CNN to refine the feature map based on the denoised label signal, $\mathbf{Y}^{r-1}$ obtained in the previous GLR iteration. See an illustration in Fig. 5.2(b) for the graph update after r-th GLR iteration. The

individual degree of Vertex i is updated as:

$$\begin{aligned} \mathring{e}^r_{i,j} &= \begin{cases} 1, & \text{if } e^r_{i,j} \in \mathbf{P}^r \ \& \ a^r_{i,j} > \beta \\ 0, & \text{if } e^r_{i,j} \in \mathbf{Q}^r \ \& \ a^r_{i,j} \leq \beta, \end{cases} \\ \gamma^r_i &= \sum_{j=1}^{N} \mathring{e}^r_{i,j}, \end{aligned} \tag{5.4}$$

where $\mathbf{P}^r$ and $\mathbf{Q}^r$ sets are formed based on the denoised classifier signal $\mathbf{Y}^r$. The edge $e^r_{i,j}$ is removed if it connects vertices with opposite labels or the corresponding entry to adjacency matrix is less than β, which is heuristically set to 0.1.

5.3 Proposed Network

Based on the concepts described in the previous section, this section presents the algorithmic flow and describe the architecture used to implement the proposed DynGLR network.

The block diagram of the proposed DynGLR-Net is presented in Fig. 5.3. The overall network consists of three sub-networks: (1) **G-Net** (graph generator network) used to learn a deep metric function to construct an undirected and unweighted KNN graph $\mathbf{G}^0 = (\mathbf{\Psi}, \mathbf{E}^0, \mathbf{W}^0 = \mathbf{1})$. (2) **W-Net** (graph weighting and classifier network) used to assign edge weights $\mathbf{W}^r$ for effectively performing GLR to restore the corrupted classifier signal $\mathbf{Y}^r$. (3) **U-Net** (graph update network) used to refine $\mathbf{E}^r$ to better reflect the node-to-node correlation based on the restored classifier signal in the previous iteration $\mathbf{Y}^{r-1}$.

5.3.1 G-Net

In order to learn the optimal metric space, as in [159], this research uses a CNN, denoted by $\text{CNN}_{\mathcal{D}}$, to learn a mapping function $\mathcal{D}(\cdot)$. The detailed architecture of $\text{CNN}_{\mathcal{D}}$ is shown in Fig. 5.4.

For a random observation triplet (x_a, x_p, x_n), such that $e_{a,p} \in \mathbf{P}$ and $e_{a,n} \in \mathbf{Q}$, the

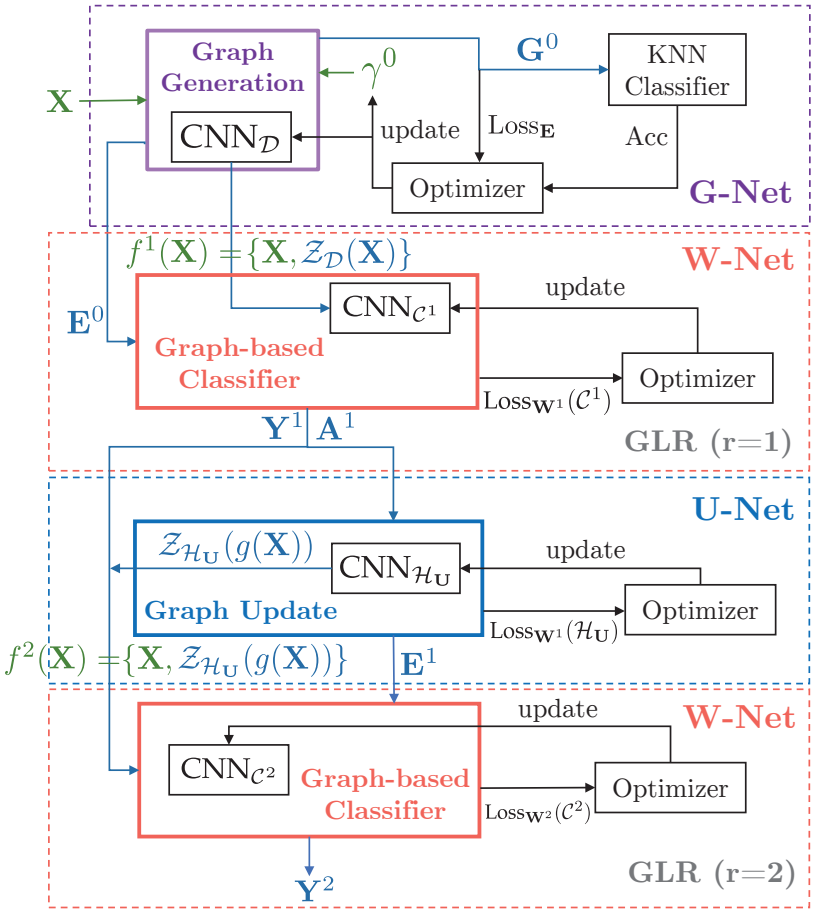


Figure 5.3: The overall block diagram of the proposed DynGLR-Net for $r = 2$. Given observations $\mathbf{X}$, G-Net (see Subsec. 5.3.1) first learns an initial undirected and unweighted KNN-graph by minimizing $\text{Loss}_{\mathbf{E}}$. The resulting edge matrix $\mathbf{E}^0$ is then used in the following, first, GLR iteration. The learnt shallow feature map $f^1(X) = \{X, \mathcal{Z}_{\mathcal{D}}(X)\}$ is then used as input to learn a $\text{CNN}_{\mathcal{C}^1}$ network for assigning weights to the initial graph edges. Given a subset of, potentially noisy labels, $\dot{\mathbf{Y}}$, GLR is operated on the constructed undirected and weighted graph to restore the labels. The resulting restored labels are used in the following GLR iterations (see Subsec. 5.3.2). To assign the degree of freedom for refining graph connectivity, the proposed network updates the graph edge sets by minimizing $\text{Loss}_{\mathbf{W}^1}(\mathcal{H}_\mathbf{U})$ given neighbor information for each node based on the resulting denoised classifier signal from the first GLR iteration. The proposed network then reassigns edge weights to the updated graph edge sets to perform better node classification in the second GLR iteration (see Subsec. 5.3.3).

proposed model minimizes the following loss function to learn the feature map:

$$Loss_{\mathbf{E}} = \sum_{a,p,n} \left[\alpha_{\mathbf{E}} - \|\mathcal{D}(x_a) - \mathcal{D}(x_n)\|_2^2 + \|\mathcal{D}(x_a) - \mathcal{D}(x_p)\|_2^2\right]_+, \tag{5.5}$$

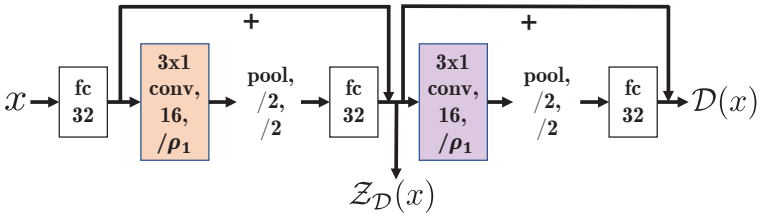


Figure 5.4: $\text{CNN}_{\mathcal{D}}$ neural network: 'pool/q/w' refers to a max-pooling layer with q=pool size and w=stride size. 'x conv y/ρ_1' refers to a 2D convolutional layer with y filters each with kernel size x and stride size ρ_1. 'fc x' means the fully connected layer with x=number of neurons. The stride size ρ_1 varies depending on the input data (see details in Sec. 5.4.1).

where $\mathcal{D}(\cdot)$ is a CNN-based feature map function, to be learnt, that returns a feature vector corresponding to the input observation, $\alpha_{\mathbf{E}}$ is the minimum margin, and operator $\left[\,\cdot\,\right]_+$ is a Rectified Linear Units (ReLU) activation function which is equivalent to $\max(\cdot, 0)$. Let $\mathcal{Z}_{\mathcal{D}}(x)$ be the learnt feature map output at the second to the last layer of $\text{CNN}_{\mathcal{D}}$ (see Fig. 5.4) obtained by minimizing the loss Eq. 5.5.

The loss function Eq. 5.5 promotes a community structure graph that has relatively small Euclidean distance between the feature maps of vertices connected by the edges in $\mathbf{P}$, and a large distance between the vertices connected by the edges in $\mathbf{Q}$, while keeping a minimum margin $\alpha_{\mathbf{E}}$ between these two distances.

Since the lack of a priori knowledge of the connectivity of the nodes, this research generates the initial graph as a fully connected graph; justification for starting with a fully connected graph is provided in [161]. A sparse $\mathbf{E}^0$ minimizes the number of $\mathbf{Q}$ edges by keeping only the connections with γ^0 neighbors per individual node. KNN-graph is constructed based on Eq. 5.1, where optimal maximum number of neighbors γ^0 is obtained via grid-search by evaluating classification accuracy of the KNN classifier (denoted by Acc in Fig. 5.3) using the validation data with the same amount of noisy labels as the training dataset. Note that, as one do not have any prior knowledge of the optimal maximum degree of each individual node, this research initially sets all $\gamma^0 = \gamma_1 = \ldots = \gamma_N$. Once the optimal number of neighbors γ^0 is obtained, the resulting graph edges $\mathbf{E}^0$ are used in the following section for pruning edge weights during edge weighting and are updated based on the regularized metric function and

the difference between the classifier signal, before and after GLR.

5.3.2 W-Net

For assigning edge weights $\mathbf{W}^r$ to the graph $\mathbf{G}^r$, this research first employs a CNN, denoted by $\text{CNN}_{\mathcal{C}^r}$, to learn a deep metric function. A robust graph-based triplet loss function is proposed to better learn feature map $\mathcal{V}^r$, as:

$$\begin{aligned} Loss_{\mathbf{W}^r}(\mathcal{V}) = \sum_{\psi_a,\psi_p,\psi_n} & \big[\alpha_{\mathbf{W}} - \|\mathcal{V}^r(f^r(x_a)) - \mathcal{V}^r(f^r(x_n))\|_2^2 \\ & \cdot\pi_{(\psi_a,\psi_n|e_{a,n}\in\mathbf{Q})} + \|\mathcal{V}^r(f^r(x_a)) - \mathcal{V}^r(f^r(x_p))\|_2^2 \\ & \cdot\pi_{(\psi_a,\psi_p|e_{a,p}\in\mathbf{P})}\big]_+ \\ \Pi^r = & \{\pi_{\psi_i,\psi_j}\} = \{\Theta(\dot{y}_i^r, \dot{y}_i^{r-1}, \dot{y}_j^r, \dot{y}_j^{r-1})\}. \end{aligned} \tag{5.6}$$

Θ is an edge attention activation function (see Eq. 5.7) for the particular function) that estimates how much attention should be given to each edge and $\mathbf{Y}^r = \{\dot{\mathbf{Y}}^r = [-1,1]^M, [-1,1]^{N-M}\}$ is the restored classifier signal obtained via Eq. 5.3 starting from the classifier signal in the previous iteration, $\mathbf{Y}^{r-1}$. π_{ψ_i,ψ_j} is the amount of attention, i.e., edge loss weights, assigned to the edge connecting vertices ψ_i and ψ_j. Note that $\mathcal{C}^r(\cdot)$ is the feature map learnt by minimizing Eq. 5.6.

The architectures for $r = 1$ and $r = 2$ are shown in Fig. 5.5. Since, at the first iteration $r = 1$, many noisy labels are expected, the residual network architecture will be different to the $r > 1$ case [162].

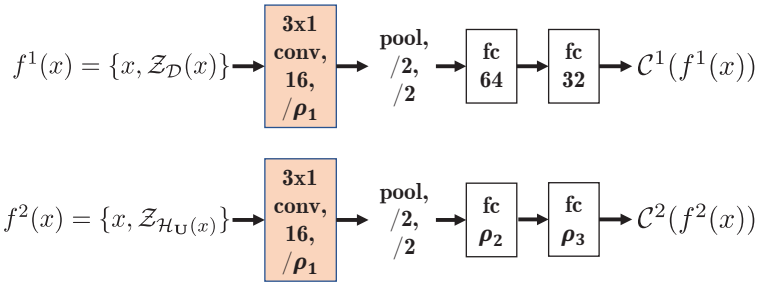


Figure 5.5: $\text{CNN}_{\mathcal{C}^r}$ neural nets. The stride ρ_1 and the number of neurons ρ_2, ρ_3 vary depending on the input data (see details in Sec. 5.4.1).

The architecture presented in Fig. 5.5 (top) is used as the feature map $\mathcal{C}^1(\cdot)$, after

G-Net, to construct the graph $\mathbf{G^1} = (\mathbf{\Psi}, \mathbf{E}^0, \mathbf{W}^1)$ by minimizing $\text{Loss}_{\mathbf{W}^1}(\mathcal{C}^1)$ taking as input undirected graph $\mathbf{G^0}$ learned via G-Net. The input to $\text{CNN}_{\mathcal{C}^1}$ is the concatenated observations $\mathbf{X}$ and "shallow feature maps" learned via G-Net, i.e., the output of the second to last layer of $\text{CNN}_{\mathcal{D}}$ as presented in Fig. 5.4, denoted by $\mathcal{Z}_{\mathcal{D}}(\mathbf{X})$.

The $r = 2$ architecture is shown in Fig. 5.5 (bottom), with observations $\mathbf{X}$ and "shallow feature maps" learned via U-Net (described in the next subsection) to facilitate the regularization of $\text{CNN}_{\mathcal{C}^2}$ by minimizing $\text{Loss}_{\mathbf{W}^2}(\mathcal{C}^2)$ based on the denoised labels, convolution on both feature maps, denoised classifier signal and their differences across neighbors.

Unlike [159], this research introduces edge attention activation Θ in Eq. 5.6 to dropout some edges with relatively large changes between $\dot{\mathbf{Y}}^r$ and $\dot{\mathbf{Y}}^{r-1}$ via GLR. This helps to focus learning on edges with high confidence given noisy training labels. Therefore, the overall training performance is better than the standard dropout layer approach, which drops out random neuron units in the network. The edge attention activation Θ and Φ for this study is implemented as:

$$\Phi(\dot{y}_i^{r-1}, \dot{y}_i^r) = \begin{cases} 1, & \text{if } |\dot{y}_i^{r-1} - \dot{y}_i^r| \leq \varepsilon^r \\ 0, & \text{if } |\dot{y}_i^{r-1} - \dot{y}_i^r| > \varepsilon^r \end{cases} \tag{5.7}$$

$$\Theta(\dot{y}_i^{r-1}, \dot{y}_i^r, \dot{y}_j^{r-1}, \dot{y}_j^r) = min(\Phi(\dot{y}_i^{r-1}, \dot{y}_i^r), \Phi(\dot{y}_j^{r-1}, \dot{y}_j^r)),$$

where threshold ε^r is used to determine whether a node's label can be trusted and also helps to control the sparsity of edge attention matrix $\mathit{\Pi}^r$. That is, if the difference between the signal label in the previous and current iteration is large, this means that the label most likely changed sign (from -1 to +1 or vice versa) and is unreliable in this iteration. To reflect the fact that there might be many noisy (unreliable) labels at the start, this research heuristically sets $\varepsilon^1 = 0.6$ for the first GLR iteration (see the results in Sec. 5.4). Since after applying GLR the classification signal is expected to be cleaner, this research heuristically sets threshold $\varepsilon^2 = 0.15$ for the second stacked W-Net during training to ensure that it regularizes CNNs with less concern about the over-fitting issue introduced by noisy labels.

5.3.3 U-Net

Edge convolution has been proven recently to be a rich feature representation method [154], [155]. This research adopts edge convolution in deeper feature map learning, i.e., after GLR $r = 1$. That is, given $\mathbf{A}^1$, from the first GLR iteration, each node's feature representation is enhanced by considering observations of both $\mathbf{X}$ and classifier signal $\mathbf{Y}^1$ from its six nearest neighbors (set heuristically), which are most-likely to have the same label.

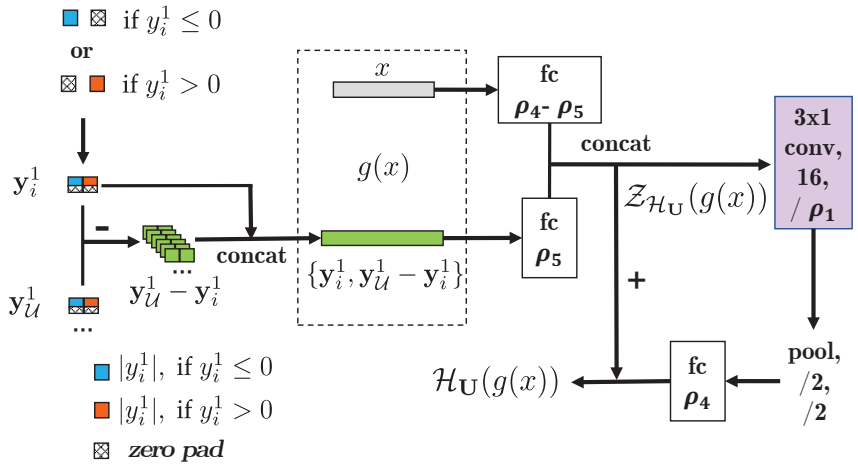


Figure 5.6: $\text{CNN}_{\mathcal{H}_\mathbf{U}}$ neural nets. The stride size ρ_1 and the number of neurons ρ_4, ρ_5 vary depending on the input data (see details in Sec. 5.4.1).

Incorporating an additional CNN, denoted by $\text{CNN}_{\mathcal{H}_\mathbf{U}}$, shown in Fig. 5.6, this research constructs a richer feature representation $\mathcal{H}_\mathbf{U}(g(x))$ to enhance the graph-based classifier learning with a single input to the network, $g(x)$, comprising x_i and $\{\mathbf{y}_i^1, \mathbf{y}_{\mathcal{U}}^1 - \mathbf{y}_i^1\}$, where $\mathbf{y}_i^1$ denotes a tuple $(y_i, 0)$, if $y_i > 0$ or $(0, y_i)$ otherwise. $\mathbf{y}_{\mathcal{U}}^1$ is a 6×2 matrix formed by concatenating $\mathbf{y}_i^1$ with the nearest six neighboring nodes. Finally, $\mathbf{y}_{\mathcal{U}}^1 - \mathbf{y}_i^1$ is obtained by subtracting each row of $\mathbf{y}_{\mathcal{U}}^1$ by $\mathbf{y}_i^1$.

Graph edge $\mathbf{E}^{r-1}$ is updated by Eq. 5.4 based on the learnt regularized feature map $\mathcal{H}_\mathbf{U}(\cdot)$ in order to better reflect the node-to-node correlation. The new edge matrix $\mathbf{E}^1$ and the denoised classifier signal $\mathbf{Y}^1$ are then used in the second graph-based classifier iteration.

Though one can continue iterating between W-Net and U-Net, in the practical

implementation, only two iterations are performed, to reduce computation complexity, since heuristically no improvement can be observed after $r = 2$ iterations.

5.4 Simulations

This section presents the simulation results, including the ablation study, visualization results, and comparison of the performance against different, classic and state-of-the-art classifiers, under different label noise levels.

5.4.1 Simulation setup: Datasets, Benchmarks, Parameters and Performance Measure

Datasets

Three corrected labeled binary-class datasets and a multi-class dataset are selected from Knowledge Extraction based on Evolutionary Learning dataset (KEEL) [163] and this thesis that vary in the number and type of features; these sets are, from low dimensional feature sets to higher ones: (1) ***Phoneme***: contains nasal (class 0) and oral sounds (class 1), with 5404 instances (frames) described by 5 phonemes of digitized speech. (2) ***Magic***: contains images generated by primary gammas (class 0) from the images of hadronic showers initiated by cosmic rays in the upper atmosphere (class 1), where 19020 instances are generated for simulation using the imaging technique, with each instance containing 10 attributes to characterize simulated images. (3) ***Spambase***: to determine whether an email is spam (class 0) or not (class 1), with 4597 email messages summarized by 57 particular words or characters. (4) ***Gait***: contains 15 subjects' 126 gait sequence records of middle 4-meter motion at two different rehabilitation sessions collected in Chapter 2. Each frame of the gait sequence is categorized as one of the 9 gait phases as shown in Fig. 3.1 and is described by 420 distinct features extracted in Chapter 3. The inputs and output (inputs $\Rightarrow$ output) of the classification task for the above datasets are: (1) ***Phoneme***: 5 real values $\Rightarrow$ 1 boolean value (zero or one). (2) ***Magic***: 10 real values $\Rightarrow$ 1 boolean value. (3) ***Spambase***: 47 real values $\Rightarrow$ 1 boolean value. (4) ***Gait***: 420 real values $\Rightarrow$ 1 categorical value (one of $\{0,1,\ldots,8\}$). The mo-

tivations of selecting the above datasets are: (1) To evaluate the binary classification performance of the proposed robust classifier against label noise across different applications, such as speech recognition (***Phoneme***), image classification (***Magic***), spam detection (***Spambase***). (2) To investigate the ability of using the proposed network for multi-class classification task and evaluate its robustness against the potential risks of mislabeling in gait phase classification application which is one of the major research tasks in this thesis.

Benchmark classifiers

This section compares the proposed network against 10 different classification methods: (1) SVM with radial basis function kernel (SVM-RBF) (2) a classical CNN, consisting of two CNN blocks and two fully connected layers afterwards, where each CNN block has a convolution layer, a max pooling layer and one dropout layer (3) a graph CNN with multiple graph construction blocks, where each block constructs a KNN graph based on multiple graph structures learnt via edge convolution; batch normalization with decay is used (called DynGraph-CNN [155]) (4) a KNN classifier using CNN-based deep metric learning (used CNN is the same as $\text{CNN}_{\mathcal{D}}$) [159] (called DML-KNN) (5) a rank-sampling [164] based KNN classifier using CNN-based deep metric learning (CNN used is same as in DynGraph-CNN), where sampling is performed on the training set by calculating the resulting classification accuracy using randomly sampled samples. For each dataset, the top 480 training samples with relatively high classification accuracy in the validation set are used. During inference, 480 selected training samples are divided into 6 equal-size batches by stratified random sampling and the predictions using each batch are averaged to obtain the final decision (6) label noise robust SVM with RBF kernel [165] (LN-Robust-SVM-RBF) (7) a graph-based classifier with negative edge weights assigned between the centroid sample pairs and between the boundary sample pairs (named Graph-Hybrid [139]) (8) a CNN network (same as in DynGraph-CNN) trained by savage loss (called CNN-Savage [38]) (9) a CNN network (same as in DynGraph-CNN) trained by bootstrap-hard loss (called CNN-BootStrapHard [39]) (10) a CNN network (same as in DynGraph-CNN) trained via

dimensionality-driven learning strategy (called CNN-D2L [30]). Note that Classifiers (1)-(4) are classical methods for normal, 'noise-free', conditions, and methods (5)-(9) are proposed to avoid overfitting under noisy training labels. All CNN-based methods adopt l_2 regularization for each layer. Similar to Benchmark (5), this research also adopts rank-sampling technique on the training set to select trusted samples that will further facilitate the predictive performance and consistency of the proposed model, denoted by 's' appended to the model name. Note that one-vs-all (OVA [166]) scheme is adopted to perform multi-class classification for ***Gait*** dataset based on the above classifiers (1)(6)(7).

Ablation study

To understand how different components of the proposed architecture affect the results, an ablation study is investigated by removing some components. The resulting architectures are denoted by DynGLR-G-number, where 'G' refers to Graph generation and '1' refers to edge weighting, '2' to GLR, and '3' graph update. That is, the following variants of the proposed scheme are compared: (1) DynGLR-G-2: importing the unweighted graph $\mathbf{G}^0$ generated by G-Net (see Fig. 5.1) into GLR for classification. (2) DynGLR-G-12: assigning weights to the unweighted graph $\mathbf{G}^0$ via an adaptive Gaussian kernel function (see Eq. 5.2); the resulting undirected and weighted graph is then used to perform node classification via GLR. (3) DynGLR-G-1232: updating the graph edge sets by considering the neighbors of each node with denoised classifier signal and observed feature maps (see Fig. 5.2); the resulting unweighted graph is then used for classification. (4) DynGLR-G-12312: reassigning weights to the updated unweighted graph to effectively perform classification; this research performs rank-sampling for all architectures to evaluate the benefits, denoted by 's' appended to the name of each proposed architecture. Note that for ***Gait*** dataset, OVA scheme is adopted to perform multi-class classification for all proposed architectures.

Simulation setup, performance measure and parameters

The experiments are conducted on ***Phoneme***,***Magic***,***Spambase*** datasets for binary classification task and ***Gait*** dataset to further investigate the multi-class classification performance and evaluate the robustness against the mislabeling of gait event frames in gait phase classification task. For the first three binary classification datasets, each dataset is randomly divided into training, validation and testing sets with 40%, 20%, 40% of instances with balanced class distribution, ensuring that there are enough training and validation samples without considering the under fitting problem caused by training models on the insufficient training data and inappropriate hyper-parameters found using insufficient validation samples. In gait phase classification task, training and testing split is used as the same as the experiments conducted in Chapter 3.

To evaluate the robustness of different classification methods against label noise, this research randomly samples subsets of instances from both training and validation sets and reverse their labels. Since there is no record of how the samples are labeled in the first three binary datasets, for each dataset, a subset of samples is first randomly sampled from training/validation sets with uniform distribution. Then the binary labels that correspond to the selected samples are reversed. To build a gait phase classification dataset, experts like therapist usually first manually extract the gait event frames that are located on the boundary of two adjacent gait phases by searching for a specified human pose introduced in [91]. The associated human pose to each gait event frame is drawn in Fig. 4.1 as skeleton that is labeled in red texts. Once all gait event frames are found, gait phase labels are assigned to their adjacent frames sequentially. Based on this record, in the chapter, the simulation of potential label noise for the ***Gait*** dataset is conducted by randomly shifting the timestamp of gait phase event within in the training set, resulting in shifted gait event frames and incorrect categorical labels. In detail, the number of frames shifted to the correctly extracted gait event frame is randomly selected from $\{-\Gamma, \ldots, \Gamma\}$, where Γ is defined as the maximum number of frames that could be mis-shifted by a data annotator when extracting the gait event frames.

Mean classification error rates are measured by running 20 experiments per label noise level (0% to 25%) or mislabeling level (0 to 8 frames). The same random seed settings are used across all classification methods for reproducing the simulation results and ensuring a same split of training, validation and testing sets. All duplicated samples of same features are removed to ensure a fair comparison considering the potential prediction failures made by semi-supervised classifiers such as DML-KNN, Graph-Hybrid and the proposed DynGLRs.

Hyper-parameters used for each experiment are obtained from the validation sets by grid search. All used parameters are listed in Table 5.1.

Table 5.1: Parameters for the proposed architectures. $x \Rightarrow y$ means that the learning rate decreases linearly from x to y with the epoch number.

Hyper-parameters	***Phoneme***	***Magic***	***Spambase***	***Gait***
$\rho_1, \rho_2, \rho_3, \rho_4, \rho_5$	1,256,64,256,6	1,128,32,128,4	2,32,32,64,6	2,128,32,128,6
G-Net learning rates	0.02⇒0.01	0.02⇒0.01	0.02⇒0.01	0.02⇒0.01
G-Net epochs	160	160	60	180
W-Net(r=1) learning rates	0.02⇒0.01	0.02⇒0.01	0.02⇒0.012	0.02⇒0.01
W-Net(r=1) epochs	320	320	80	240
U-Net learning rates	0.002⇒0.001	0.002⇒0.001	0.002⇒0.001	0.002⇒0.001
U-Net epochs	120	180	100	60
W-Net(r=2) learning rates	0.01⇒0.002	0.01⇒0.002	0.02⇒0.01	0.02⇒0.01
W-Net(r=2) epochs	60	40	40	180

To guarantee the solution $\mathbf{Y}^r$ to Eq. 5.3 is numerically stable, the conditional number $\kappa = 60$ and $\mu^r = 0.67\mu^r_{max}$ are used in all experiments. Eq. 5.5 and Eq. 5.6 use the distance margin $\alpha_{\mathbf{E}} = \alpha_{\mathbf{W}} = 10$. For each epoch, the batch size is 16, each batch comprising 80 labeled instances from training set and 20 unlabeled instances from validation set; thus $16 \cdot 100$ instances are randomly selected. This results in 16 graphs to regularize training per epoch.

All CNNs are learnt by Adam optimizer [167], classification accuracy and classifier signal changes are used as metric for rank-sampling for further improving the predictive performance.

5.4.2 Results and Discussion

This section is organized as follows. As part of the ablation study, Subsecs. 5.4.2 and 5.4.2 evaluate the ability of the proposed schemes to clear the noisy labels and observation samples during the training phase, respectively. This research analyzes the sensitivity of hyper-parameters that affect the regularization performance in Subsec. 5.4.2. It shows the effectiveness of the proposed iterative graph update scheme in Subsec. 5.4.3 by visualizing the learnt underlying graph in spectral domain. To analyze the impact of different components on the classification accuracy, it shows the classification error rates in Subsec. 5.4.3 and discuss the findings of the ablation study during the testing phase and show comparison with state-of-the-art schemes.

Evaluating graph update block

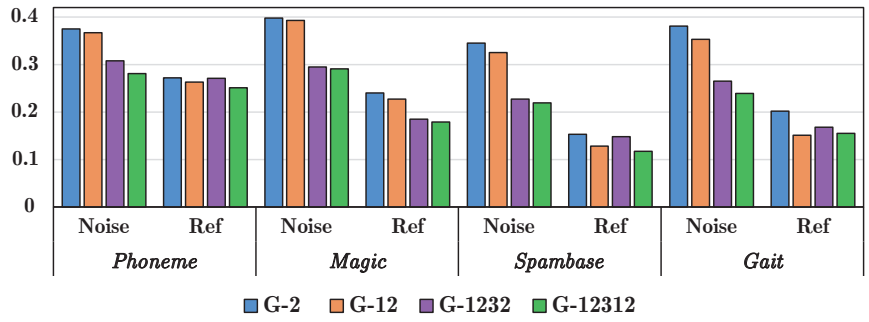


Figure 5.7: Mean Edge Weight Proportion ϱ for the proposed DynGLR-Nets for the three binary class datasets when 25% labels used for training are wrong (denoted by 'Noise') or when all training labels are correct (denoted by 'Ref' without use of GLR). G-2, G-12, G-1232, and G-12312 schemes are described in the previous subsection. Note that, for ***Gait*** dataset, the results are averaged over all binary classifiers based on OVA scheme when Γ=8 frames.

First, this research evaluates ability of the graph update block to clear noisy labels. The measure of mean edge weight proportions are used, which is defined as:

$$\varrho = \frac{\sum_{p,n}^{N} w_{p,n}}{\sum_{i,j}^{N} \mathbb{1}(w_{i,j} > 0)}, \tag{5.8}$$

where ψ_p and ψ_n are two nodes with the opposite labels, $w_{p,n}$ is the weight of the edge

$e_{p,n}$ and $\mathbb{1}(c)$ is an operator that returns 1 if condition c is fulfilled, and 0 otherwise. The results are shown in Fig. 5.7, which shows that, for all datasets, the number of connections between nodes with opposite labels decreases with iterations and becomes similar to that without any noise, indicating that the graph update manages to restore the noisy labels.

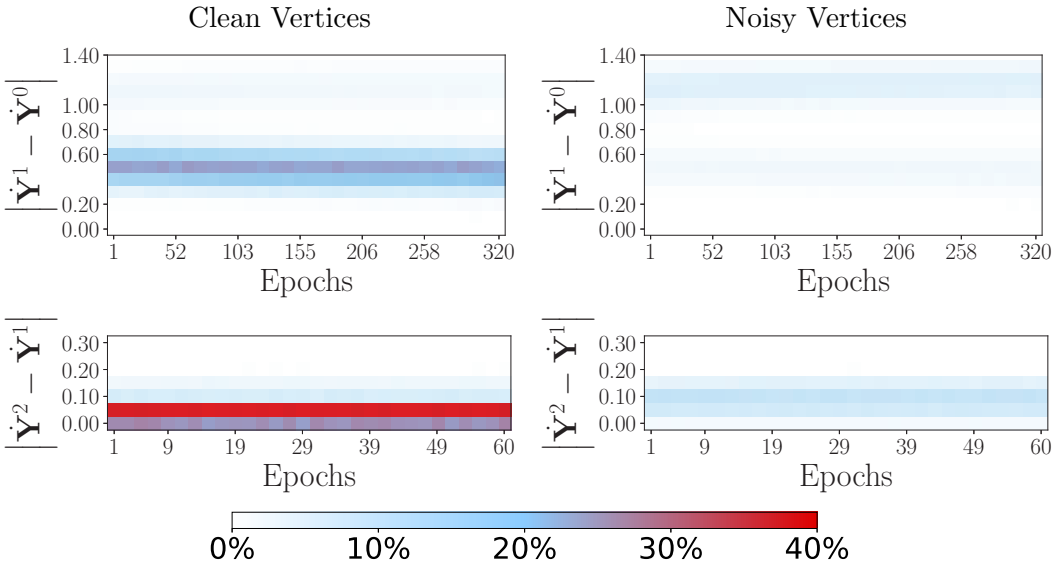
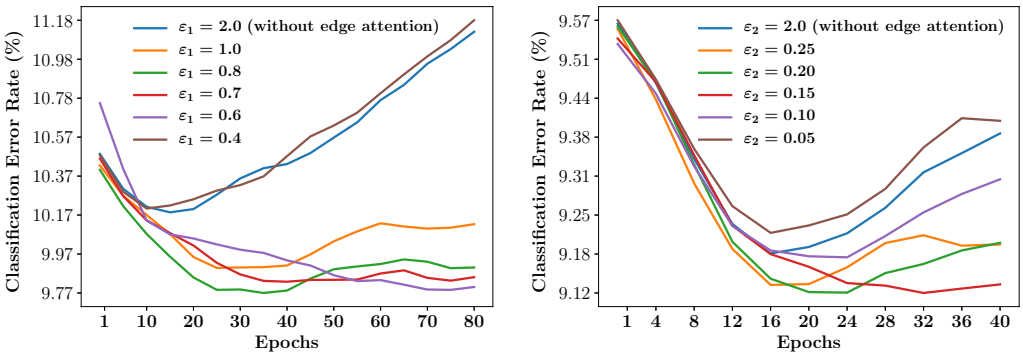


Figure 5.8: Classifier signal changes $|\dot{\mathbf{Y}}^{r-1} - \dot{\mathbf{Y}}^r|$ density visualization for ***Phoneme*** dataset after the first (top row) and second (bottom row) GLR iteration during the training period. Vertices with clean/incorrect labels are shown in the first/second column. The intensity of the classifier signal changes across all experiments are represented through colormaps.

Eq. 5.7 uses a threshold to distinguish reliable nodes from unreliable nodes. In order to evaluate the used approach and to set thresholds, the changes of the classifier signal $|\dot{\mathbf{Y}}^{r-1} - \dot{\mathbf{Y}}^r|$ during the first two iterations are shown in Fig. 5.8. One can see that when all the labels are clean (left column) the difference between the signals before and after GLR is mainly below 0.6 and 0.15, in the first and the second iteration, respectively. Thus, by setting the thresholds at $\varepsilon^1 \approx 0.6$ and $\varepsilon^2 \approx 0.15$ for the first and the second GLR iteration, respectively, it is easy to distinguish the vertices with potentially noisy labels. Similar observations are made for the ***Spambase*** and ***Magic*** datasets.

Heuristically, it can be observed that as more GLR iterations are performed, the overlap between the clean and noisy vertices distribution of classifier signal changes

is high, resulting in reduced ability to use thresholding for distinguishing if a node is sufficiently cleaned. That is why in all experiments, to reduce complexity, the graph update is performed only after the first iteration, i.e., for r=1.



(a) GLR (r=1) (b) GLR (r=2)

Figure 5.9: Classification Error Rate (%) for ***Spambase*** dataset using different $\varepsilon^{1,2}$ in DynGLR-G-12312.

Hyper-parameter sensitivity

This research uses attention activation Eq. 5.7 to detect the label of vertex ψ_i as possibly noisy if $\Phi(\dot{y_i}^{r-1}, \dot{y_i}^{r}) = 1$. To analyze the sensitivity of hyper-parameters ε^1 and ε^2 (thresholds) in Eq. 5.7, Fig. 5.9 shows the classification error rate for the ***Spambase*** dataset during training using different values of ε^1 and ε^2. It can be observed that the thresholds $\varepsilon^1 = 0.6, \varepsilon^2 = 0.15$, from the density visualization of classifier signal changes $|\dot{\mathbf{Y}}^{r-1} - \dot{\mathbf{Y}}^{r}|$ of Fig. 5.8, are appropriate to improve the regularization of CNNs. Note that the classification error rate reduces from first GLR iteration to second GLR iteration.

Evaluating Rank-sampling

To evaluate the denoising effects of GLR, Fig. 5.10 shows the mean noise level of the training labels after GLR is performed across all proposed architectures with and without sampling. It is clear from Fig. 5.10, for all three datasets, that the mean

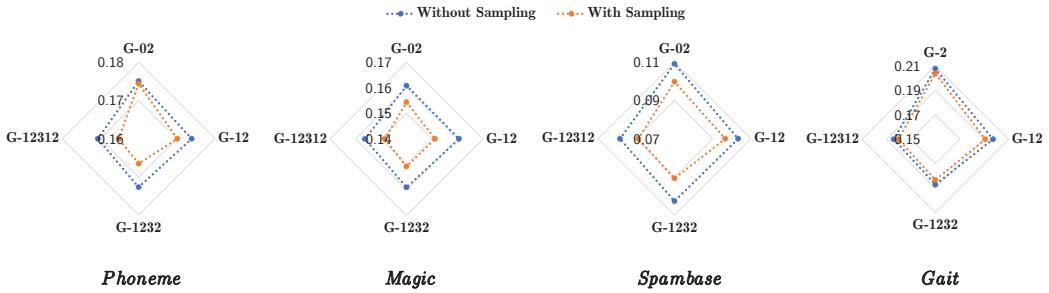


Figure 5.10: For the three binary class datasets, the mean noise level of the training labels after GLR is performed in all proposed DynGLR-Nets when 25% labels used for training are incorrect. For ***Gait*** dataset, the mean noise levels are measured over all binary classifiers based on OVA scheme when Γ=8 frames.

noise level is lower with rank-sampling than without. This confirms that with rank-sampling, one can further reduce the effects of noisy training labels by dropping out the less reliable training samples.

5.4.3 Feature map visualization

To demonstrate the effectiveness of denoising effects in feature-space domain, this research visualizes the underlying data structure learned by the proposed DynGLR-G-12312 for ***Magic*** dataset by Principal component analysis (PCA) of the learned deep feature maps at various stages of the algorithm. The results are shown in Figs. 5.11 and 5.12 using a single batch with 80 training and 20 testing samples. The vertical and horizontal axis values correspond to the values of PCA projected observation feature maps and the color (red or blue) corresponds to one of the two classes.

Fig. 5.11 show all observations (training and testing) for the case of no noise in the labels (left plots) and when 25% of labels are wrong (right plots). The first row shows the input observations, while the remaining rows present outputs of r-th iteration. By comparing the positions of red and blue dots in the left and right plots, one can see the way feature-space is changing due to iterative application of GLRs for denoising the labels.

For ***Magic*** dataset, it can be observed that the number of noisy+known labels drops from 41 (at the input of the network), to 24 (after the first iteration) to 21

after the second iteration. After $\mathcal{C}^1(X)$ is learnt, one can see from the second row of the figure that a decision boundary has been made, but there are several small clusters located at the boundary of the two major clusters contributing towards 24 miss-labeled nodes. After graph update and the second GLR iteration, it is obvious that a significant improvement of underlying structure where clean decision boundary can be made, leading to reduction on the number of wrongly restored labels.

Fig. 5.12 shows only unlabeled and miss-labeled samples for both datasets, where left and right columns present training and testing samples, respectively. It can be observed that a decreasing number of miss-labeled samples in both training and testing sets as GLR is effectively performing label denoising.

Graph spectrum visualization

Graph Fourier Transform (GFT) is another approach to represent the smoothness and connectivity of an underlying graph. As in [168], this research visualizes the magnitude of the GFT coefficients in Figs. 5.13 and 5.14 along the graph update iterations.

In accordance with [168], where it is shown that the magnitude of GFT coefficients decay rapidly for a smooth signal, in the results, one can clearly observe that the magnitude of GFT coefficients is decaying more rapidly along spectral frequencies once the graph is updated (in iteration $r = 1$). Furthermore, comparing the visualization results in the first and the second column, it can be seen that the graph weighting (W-Net) smooths the graph data, with low frequency components becoming more prominent.

Classification Error Rate Comparison

Tabs. 5.2–5.5 show comparison between the proposed DynGLR networks and all the benchmarks in terms of classification error rate. From the tables, it can be seen that the proposed DynGLR networks outperform all the benchmarks. The results also indicate that the proposed robust classifier can effectively mitigate the effects of noisy training labels caused by mislabeling of gait event frames for gait phase classification.

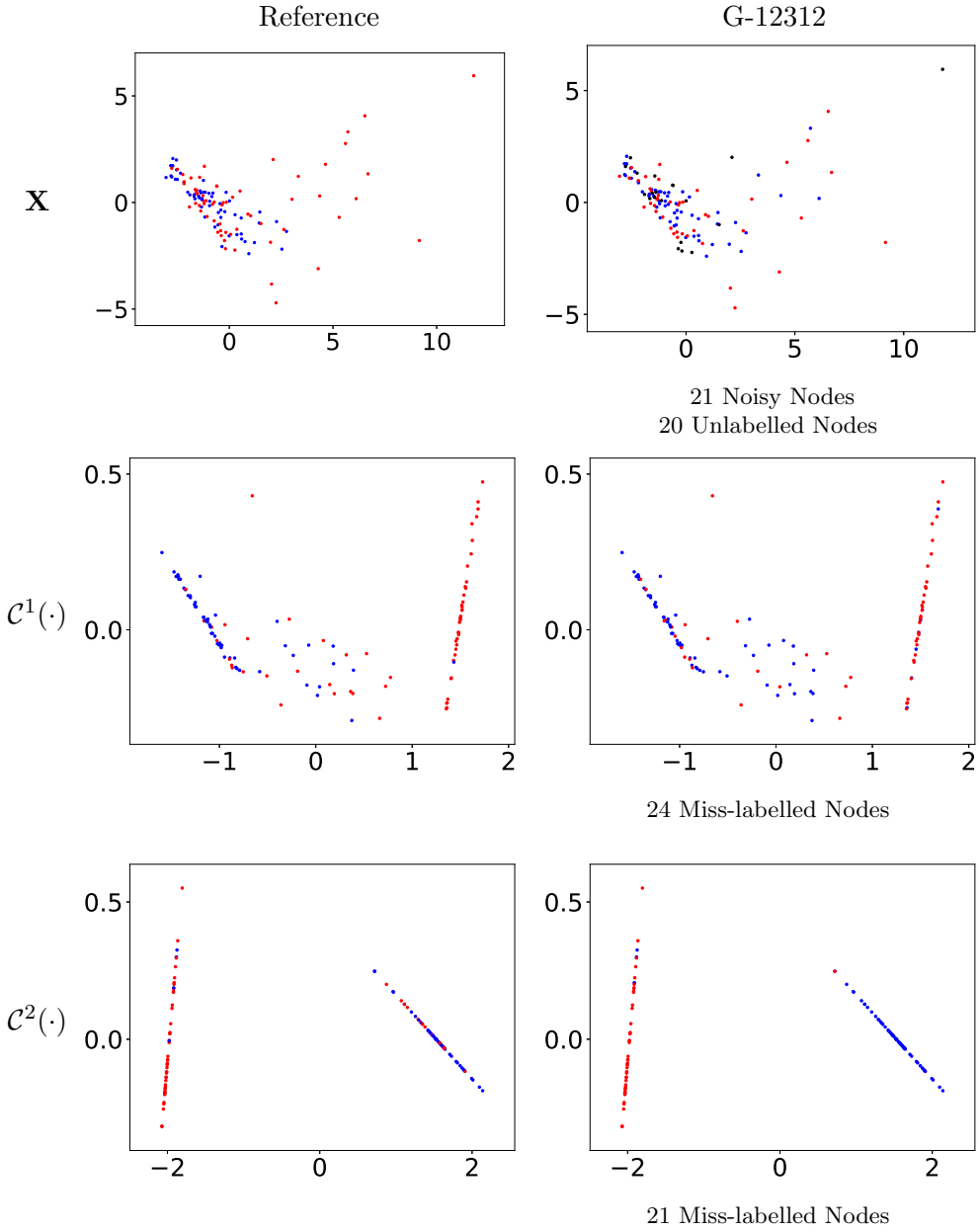


Figure 5.11: PCA-projected feature-space structure of input observations for ***Magic*** Dataset for noise-free input labels (left); 25% of mislabeled observations (right column). The top row shows the input observations, while the two rows below show outputs of W-Net iterations. Red, blue, black crosses correspond to Class 1, 2 and unlabeled nodes, respectively.

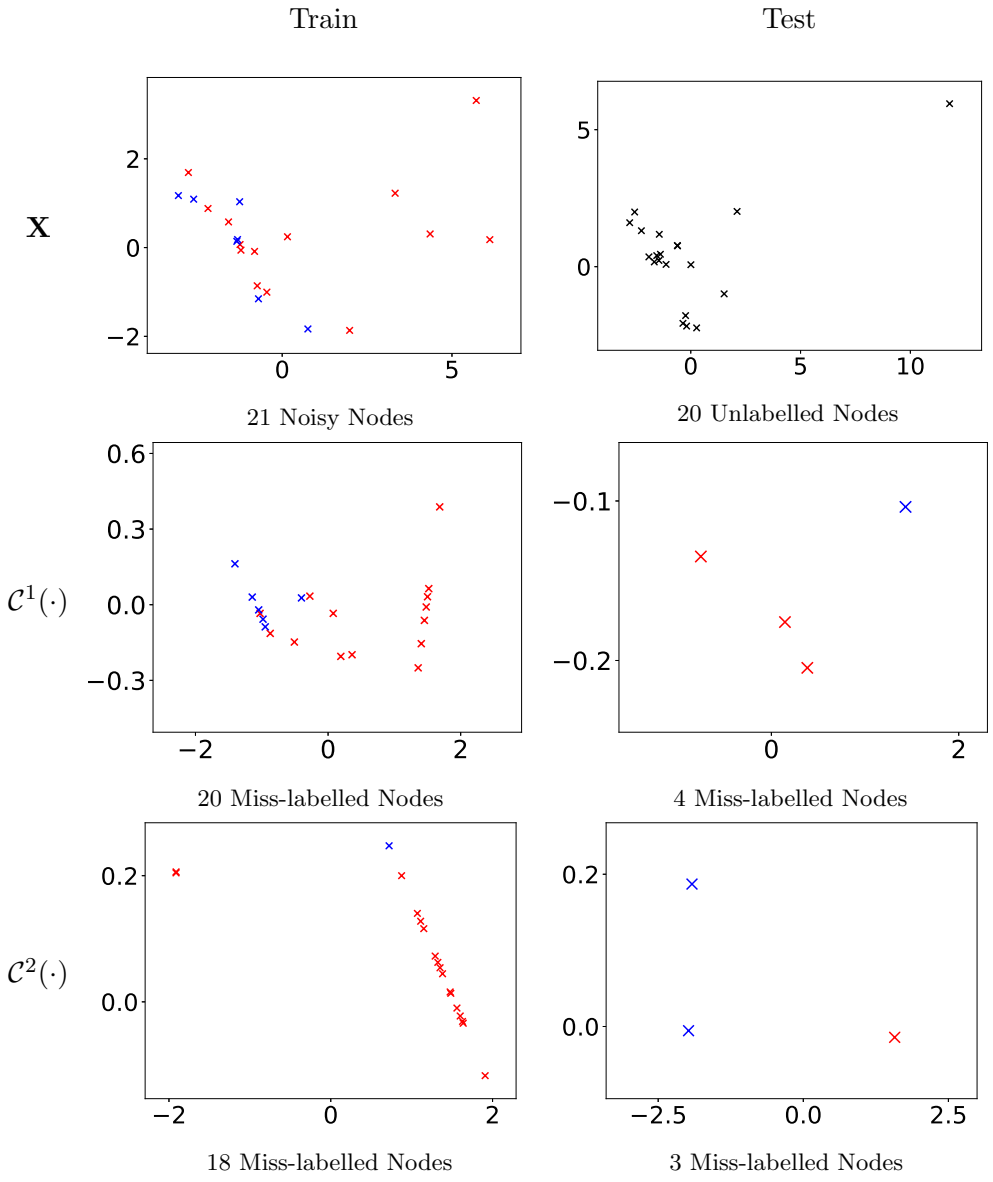


Figure 5.12: PCA-projected feature-space structure of input observations for ***Magic*** Dataset under 25% of mislabeled observations: training nodes (left) and testing nodes (right column). The top row shows the input observations, while the two rows below show outputs of W-Net iterations. Red, blue, black crosses correspond to Class 1, 2 and unlabeled nodes, respectively.

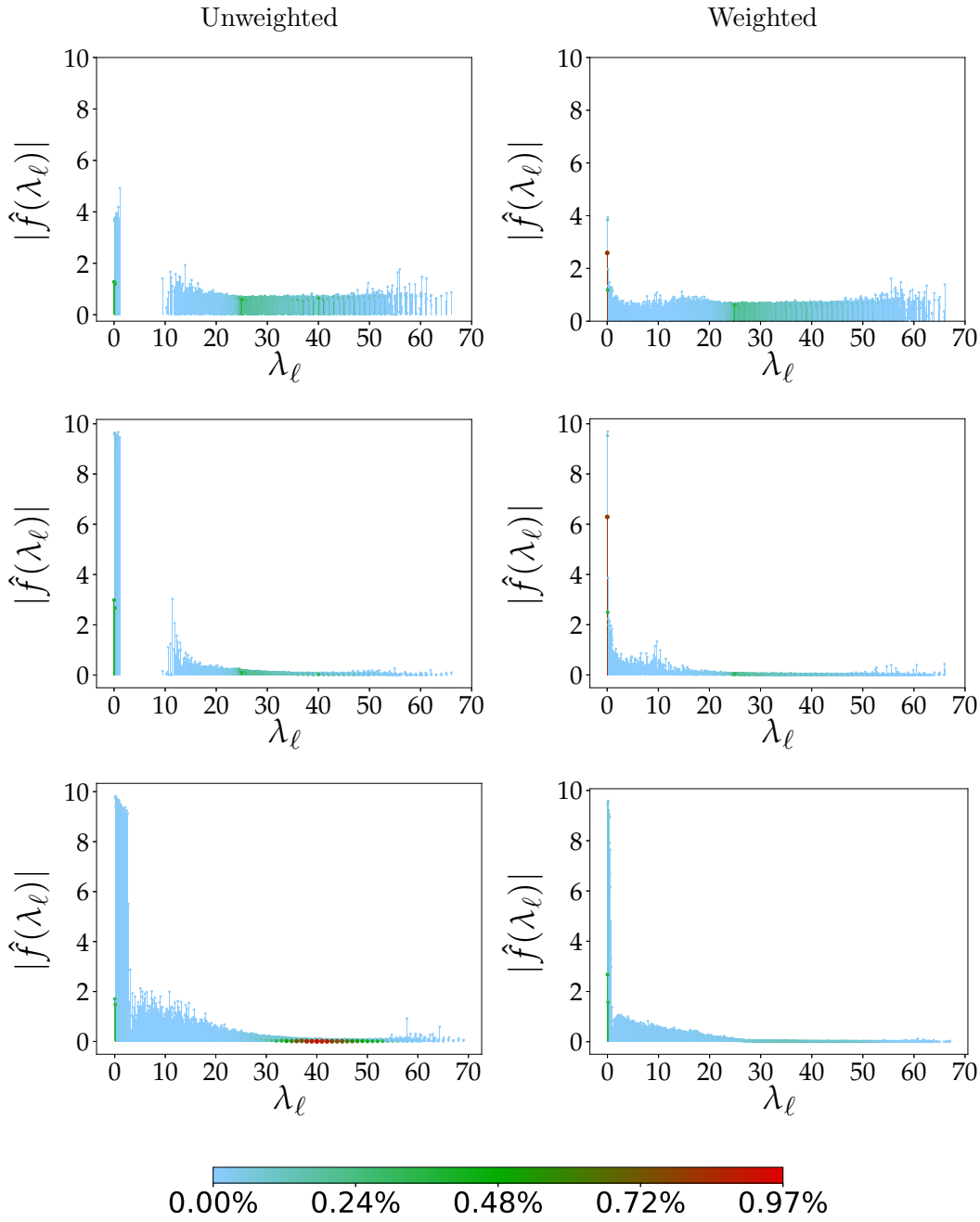


Figure 5.13: The magnitude of the Graph Fourier Transform coefficients for ***Phoneme*** Dataset. The density of each Eigenvalue λ_ℓ across all experiments on the testing sets is represented through colormaps. Top row shows the result after initialization and before GLR (G-Net output) and the second and third row show the result after the first ($r = 1$) and the second iteration ($r = 2$), respectively.

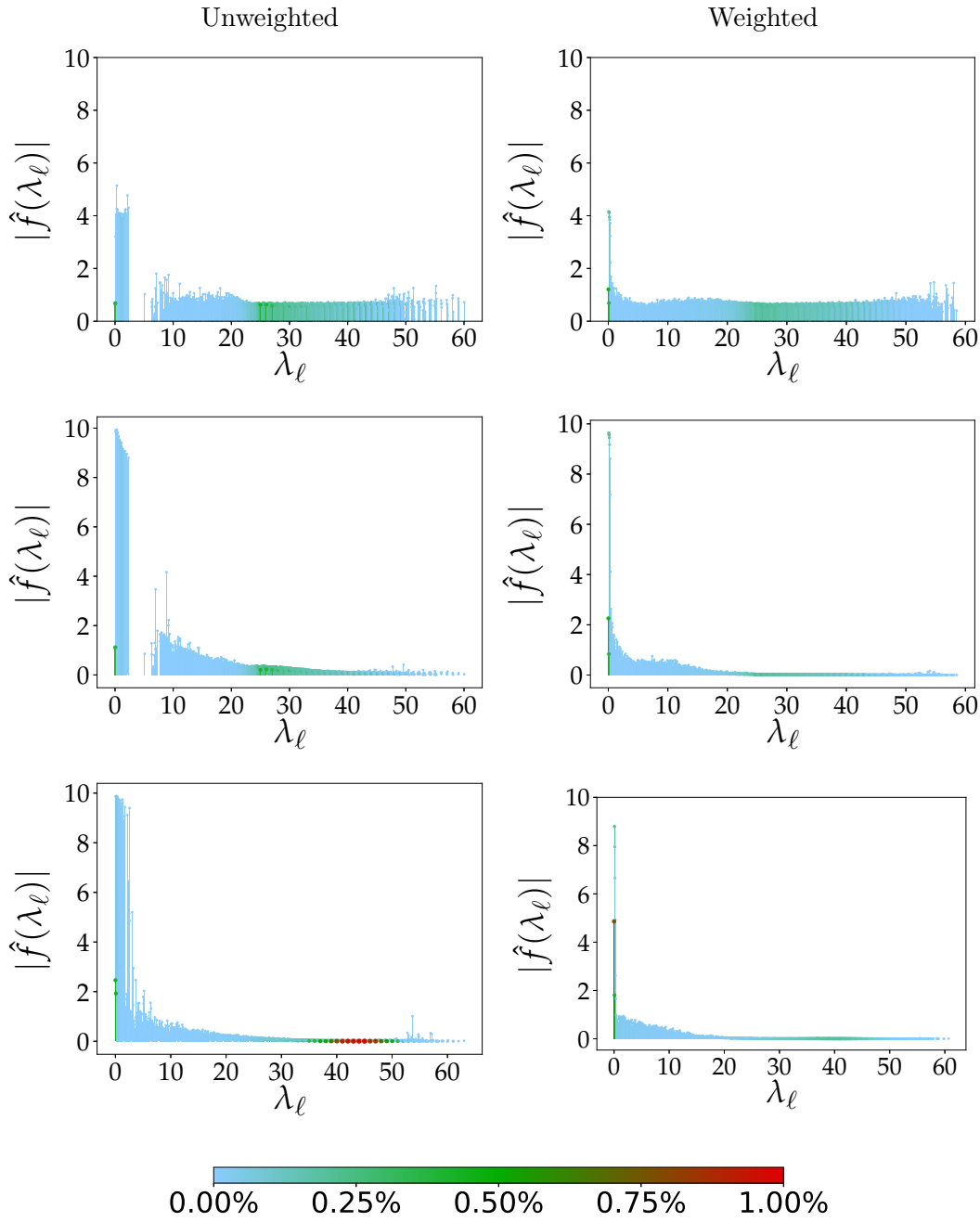


Figure 5.14: The magnitude of the Graph Fourier Transform Coefficients for ***Spambase*** Dataset. The density of each Eigenvalue λ_ℓ across all experiments on the testing sets is represented through colormaps. Top row shows the result after initialization and before GLR (G-Net output) and the second and third row show the result after the first ($r = 1$) and the second iteration ($r = 2$), respectively.

Table 5.2: Classification Error Rate (%) For ***Phoneme*** Dataset

% label noise	0	5	10	15	20	25
SVM-RBF	18.33	18.75	19.13	19.57	20.07	20.87
CNN	17.58	17.77	18.00	18.57	19.01	20.00
DynGraph-CNN	17.66	19.04	20.80	22.44	25.20	28.84
DML-KNN	17.04	17.54	17.82	18.58	19.64	21.00
DML-KNN-s	17.01	17.49	17.71	18.43	19.24	20.41
LN-Robust-SVM-RBF*	18.57	19.42	19.65	19.70	20.03	20.28
Graph-Hybrid*	22.01	23.77	25.58	27.97	30.33	33.39
CNN-Savage*	17.52	17.72	18.04	18.51	19.02	19.87
CNN-BootStrapHard*	17.46	17.72	18.00	18.31	18.84	20.15
CNN-D2L*	17.47	17.80	17.96	18.41	18.91	20.04
DynGLR-G-2*	17.04	17.50	17.70	18.34	18.81	20.03
DynGLR-G-12*	16.93	17.36	17.64	18.23	18.52	19.59
DynGLR-G-12s*	16.89	17.36	17.62	18.21	18.52	19.54
DynGLR-G-1232*	16.90	17.29	17.36	18.16	18.48	19.47
DynGLR-G-12312*	16.87	17.19	17.34	18.03	18.38	19.43
DynGLR-G-12312s*	**16.87**	**17.18**	**17.32**	**17.91**	**18.24**	**19.18**

Table 5.3: Classification Error Rate (%) For ***Magic*** Dataset

% label noise	0	5	10	15	20	25
SVM-RBF	18.42	19.07	19.63	20.18	20.61	21.13
CNN	16.45	16.87	16.91	17.62	18.09	18.86
DynGraph-CNN	17.74	18.69	19.33	21.05	24.15	27.40
DML-KNN	15.33	15.51	15.80	15.94	16.83	18.29
DML-KNN-s	15.33	15.51	15.78	15.89	16.58	17.08
LN-Robust-SVM-RBF*	18.57	18.70	18.80	19.05	19.39	19.82
Graph-Hybrid*	24.82	25.92	27.23	28.84	30.79	33.23
CNN-Savage*	16.31	16.74	16.99	17.41	18.10	18.87
CNN-BootStrapHard*	16.34	16.89	17.02	17.46	18.13	18.65
CNN-D2L*	16.34	16.79	17.21	17.48	18.20	18.75
DynGLR-G-2*	15.35	15.51	15.77	15.94	16.75	18.03
DynGLR-G-12*	15.22	15.47	15.68	15.85	16.60	17.33
DynGLR-G-12s*	15.22	15.47	15.68	15.83	16.52	16.91
DynGLR-G-1232*	15.22	15.46	15.66	15.85	16.58	17.18
DynGLR-G-12312*	15.22	15.46	15.66	15.85	16.55	17.17
DynGLR-G-12312s*	**15.22**	**15.45**	**15.65**	**15.83**	**16.49**	**16.85**

Table 5.4: Classification Error Rate (%) For ***Spambase*** Dataset

% label noise	0	5	10	15	20	25
SVM-RBF	8.09	8.50	8.98	9.75	10.68	11.49
CNN	7.69	8.27	8.89	9.85	10.8	12.47
DynGraph-CNN	8.33	9.01	10.45	12.68	16.78	22.34
DML-KNN	7.84	8.41	8.49	8.98	9.83	11.02
DML-KNN-s	7.81	7.35	8.42	8.86	9.74	10.34
LN-Robust-SVM-RBF*	7.84	8.38	8.89	9.68	10.56	11.32
Graph-Hybrid*	18.34	19.19	20.37	21.85	24.17	26.68
CNN-Savage*	8.04	8.36	8.90	9.80	10.42	12.13
CNN-BootStrapHard*	7.69	8.30	9.24	9.68	10.05	12.01
CNN-D2L*	7.73	8.46	9.05	9.87	10.96	12.17
DynGLR-G-2*	7.84	8.37	8.42	8.95	9.85	10.83
DynGLR-G-12*	7.73	8.13	8.37	8.78	9.44	9.82
DynGLR-G-12s*	7.72	8.11	8.35	8.75	9.35	9.63
DynGLR-G-1232*	7.65	8.05	8.22	8.67	9.15	9.56
DynGLR-G-12312*	7.55	7.99	8.21	8.64	9.01	9.18
DynGLR-G-12312s*	**7.55**	**7.94**	**8.18**	**8.61**	**8.96**	**9.13**

Table 5.5: Classification Error Rate (%) For ***Gait*** Dataset

Γ frames (% approx. label noise)	0 (0)	2 (9)	4 (18)	6 (26)	8 (33)
SVM-RBF	15.50	15.79	16.87	18.24	20.10
CNN	14.90	16.12	17.88	20.61	24.34
DynGraph-CNN	14.98	16.45	18.33	23.10	26.36
DML-KNN	14.77	16.20	18.01	20.71	22.32
DML-KNN-s	14.76	15.85	16.83	19.26	20.89
LN-Robust-SVM-RBF*	15.52	15.74	16.66	17.89	18.91
Graph-Hybrid*	24.38	25.84	26.17	27.68	29.20
CNN-Savage*	14.90	16.10	17.77	20.37	22.11
CNN-BootStrapHard*	14.88	16.09	17.71	20.12	22.04
CNN-D2L*	14.89	16.09	17.72	19.90	21.89
DynGLR-G-2*	14.78	15.81	16.78	18.87	20.45
DynGLR-G-12*	14.77	15.74	16.52	18.12	19.38
DynGLR-G-12s*	14.73	15.55	16.44	17.67	18.25
DynGLR-G-1232*	14.71	15.58	16.49	17.43	17.94
DynGLR-G-12312*	14.68	15.50	16.42	17.04	17.42
DynGLR-G-12312s*	**14.68**	**15.45**	**16.34**	**16.86**	**17.20**

5.4.4 Summary of findings

The DynGLR-G networks at the bottom of performance tables Tabs. 5.2–5.5 show the outcomes of the ablation study. Specifically:

- DynGLR-G-12 consistently outperforms DynGLR-G-2, showing the effect of edge-weighting.
- The improvement due to graph update can be observed between DynGLR-G-12 and DynGLR-G-1232
- By comparing DynGLR-G-1232 and DynGLR-G12312, small gains are observed, except for low noise in Spambase dataset, due to iterative design, incorporating edge weighting.
- By comparing DynGLR-G-2 and DML-KNN, performance improvements can be observed due to replacing KNN-based classification with GLR; larger gains can be observed as noise level increases.
- Semi-supervised classifiers DML-KNN and all DynGLRs with sampling (DynGLR-G-12s and DynGLR-G-12312s) benefit from rank-sampling, which also reduces the scale of training set without sacrificing the performance.

Furthermore, findings are that the importance of the following algorithmic steps, in order of largest to least importance, to the performance can be summarized as: (I) iterative graph update Eq. 5.4 - disconnect **Q**-edges and connect/reconnect **P**-edges based on the restored labels after each GLR iteration to refine the graph structure, (II) edge convolution operation - performing feature and denoised label aggregation on neighboring nodes provides richer and smoother inputs and results in a spatially sparse graph, (III) edge attention function Eq. 5.7 - to better reflect node-to-node correlation, regularizing CNN training by weighting the edge loss based on classifier signal changes before and after GLR.

5.5 Summary

This chapter introduces an end-to-end iterative graph-based deep learning architecture design to tackle the overfitting problem caused by the effects of noisy training labels. It first proposes a CNN-based graph generator G-Net to build an initial graph. Relying on the proposed graph-based regularized loss functions, it then proposes a graph-based classifier W-Net to perform online label denoising of the training samples that potentially have noisy labels. Based on the denoised training labels, the proposed model updates the underlying graph structure by learning the proposed graph update U-Net. Finally, the model learns a refined graph-based classifier W-Net to perform classification using the updated underlying graph structure. The validation on three different binary classification datasets demonstrate that the proposed architecture outperforms the state-of-the-art classification methods when uniform noise are added to the clean training labels. In further evaluation of the robustness against structured label noise, the results of multi-class dataset indicates that the proposed approach is capable of learning a robust classifier against the potential mislabeling of gait event frames for gait phase classification dataset, contributing the classifier learning that are previously investigated in Chapters 3 and 4. The rank-sampling method is adopted to achieve another enhancement for this semi-supervised classification problem. However, the methodologies to enhance such a robust classifier to achieve better predictive performance is still a open question which can be a future research direction.

There are still several drawbacks in the proposed architecture that can be further studied, such as: (1) the K nearest neighbor based graph construction process limits the maximum of node degree and requires manual tuning the hyper-parameter K to achieve a good practice. (2) the selection of the values for thresholds ϵ is critical in distinguishing whether the label of the sample is reliable or not which requires to be investigated for more datasets. (3) train multiple W-Nets for each GLR iteration is time consuming which might results in significantly time waste for large-scale image classification dataset. To mitigate these drawbacks, two main aspects can be investigated: (1) G-Net: during graph construction process, different graph edge generation methods can

be investigated to study the impact the graph construction in the proposed architecture, such as [169] that proposes a simple switching based graph generator to generate graph preserving features of graph structure satisfying given feature constraints. (2) W-Net: refining a single W-Net across all GLR iteration can be an alternative solution to reduce the model size and boost the training process.

Chapter 6

Conclusions and Future Work

This chapter gives a brief summary and conclusion of this work. It also provides several insights into the future research.

6.1 Conclusions

Driven by recent advances in imaging and multimedia signal processing technology, healthcare monitoring services based on multimedia processing are emerging. Gait analysis as one of the diagnostics tools in physical rehabilitation programs, being, for example, periodically performed in the post-stroke recovery assessment. Video-based gait analysis that enables cost-effective visual tele-rehabilitation for home-use attracts more attentions from the community in recent decades. However, current optical diagnostic and patient assessment tools tend to be expensive and not portable, such as VICON [6], Qualisys [7] and etc. This research proposed an alternative video-based smart healthcare monitoring solution that is cost-effective, portable and requires much less expertise knowledge. This thesis achieves this goal via: (1) automatic real-time motion capture with autonomous scene calibration using a portable single depth camera (MS Kinect v2) to reduce the system complexity in system setup and manual efforts in measuring kinematics parameters (2) gait analysis by automatic and effective gait phase segmentation using the proposed feature selection and extraction methods to facilitate the diagnostics of muscle activities occurred in each different gait phase. Fol-

lowing the recent advances in automatic feature extraction approach using deep learning techniques, to study whether those hand-crafted features is sensitivity to those state-the-of-art classification methods in sequence labeling problem, this research analyzes the importance and sensitivity of feature selection and extraction methods in health-care monitoring applications like gait assessment and cardiac diseases detection. To improve the robustness of classifier learning when partial of training data are incorrectly labeled caused by manual mislabeling of the ground truth data, this research introduces an iterative graph-based deep learning architecture design to learn robust classifier model, reducing the risk of overfitting problem.

Specially, the following methodologies are proposed to address the related challenges. Chapter 2 proposes: (1) adaptive IR thresholding algorithm to reduce the complexity of determining an appropriate threshold to correctly detect retro-reflective markers without leading to center deviation on those markers with circular distributed IR values. (2) histogram-based 3D marker location algorithm to map the detected 2D marker into 3D space by recovering the depth maps even when partial occlusion occurs. (3) automatic scene calibration procedure to setup the initial parameters that can be reuse multiple times, without requiring expertise knowledge compared to the state-of-the-art motion capture systems. (4) model-based automatic marker labeling to perform robust classification of markers attached on the body joint of interests. Facilitating the diagnostics of gait motion in individual gait phases, Chapter 3 includes: (1) 12 gait features to better characterize the joint movements in different gait phases. (2) gait pattern extraction method to prevent overfitting the classifier learning and reduce the amount of training samples without sacrificing the classification performance. (3) optimal feature extraction method to tackle the problem of temporal feature misalignment. (4) gait phase reconstruction method to improve the segmentation of gait phases by iteratively smoothing the predictions results globally. To further analyze the importance and sensitivity of feature selection and extractions, this research evaluates the predictive performance and complexity of the classifier models using the state-the-art classification methodologies for both gait phase and arrhythmia classification applications in Chapter 4. Improving the robustness of classifiers against the noisy training

labels, Chapter 5 proposes: (1) graph-based regularized loss functions to regularize the CNN networks, considering the reliability of training samples. (2) iterative graph-based semi-supervised classifier to perform both online denoising of training labels and classification of unlabeled testing samples. (3) graph update procedure to better reflect the correlation between samples using denoised training labels resulting from graph Laplacian regularization.

However, besides the drawbacks discussed in each chapter's summary, there are some limitations, drawbacks and considerations for this work, such as: (1) In Chapter 2, this work does not provide the analysis of marker tracking accuracy in terms of 3D marker position comparison with the benchmarked VICON system. An alternative way to investigate this can be done by using both static and dynamic testing similar to [170] that uses the distance measurements of two markers (with constant distance in real-world) to evaluate the linear tracking accuracy between two motion capture system for gait analysis. (2) Alg. 3 relies on the identification of marker position first which is not universal to unknown marker positions. An alternative method can be used to replace it with a machine learning based regression model to improve the robustness of marker location in 3D space, such as [171] that uses convolutional neural network to automatically label the retro-reflective markers and reconstruct the hand motion. (3) In Chapter 2, more IR-based marker detection algorithms can be used to benchmark the detection accuracy instead of using only two proposed thresholding methods (static threshold and adaptive threshold). Alternative methods with small number of tunable parameters include: (a) [172] proposes a generalized Laplacian of Gaussian filter to detect elliptical bob structure by locating the local maxima of an intermediate map obtained from aggregating the log-scale-normalized convolution responses of each individual filter. (b) [173] finds the maximas in the matrix of the Determinant of Hessian of the input image. (4) In Chapter 3, this work propose 12 gait parameters to characterize the gait motion. However, no other state-of-the-art approach is investigated to benchmark with the proposed gait parameters to conduct the gait phase classification task. More parameters as investigated in the literature can also be considered in the future work, such as sagittal plane velocity, relative anterior acceleration in [174]. In

addition, there is a open question "if less number of gait parameters is still effective to be selected as input features to perform the classification task". Therefore, different combinations of the proposed gait parameters can be selected as input features to investigate this question. (5) The overall video based motion capture system and gait phase classification method still requires expertise in accurate marker placement on the human body. This limitation should be addressed to finally deploy this kind of system for in-home use without manual efforts from human. Future research direction to reduce the manual efforts is discussed in the next section.

In summary, this thesis not only provides an alternative cost-effective, portable motion capture solution, but also investigates the feature selection & extraction methods and robust classifier learning to facilitate the visual diagnostics for home-use healthcare monitoring application.

6.2 Challenges and Future Work

Emerging healthcare monitoring techniques facilitate the cost reduction, effectiveness improvement for traditional diagnostics which is often conducted in hospitals and local clinics. For gait assessment in physical rehabilitation programs, although many challenges are addressed in this research, there still remain open challenges where this section provides several suggestions for future research:

1. **gait analysis using a single depth camera in different views**
   Gait analysis is a clinical tool to perform a detailed assessment of a patient with a walking disorder [175]. One of the elements in typical clinical gait analysis is kinematic analysis, video-based motion capture system as one of the widely used solutions to capture kinematics measurements, like state-of-the-art VICON system often requires multiple cameras to capture the full body movements to perform gait analysis [176]. To reduce the complexity and cost of such video-based motion capture in gait analysis, this work propose a motion capture framework in Chapter 2. However, only sagittal view gait analysis is investigated in this work. Gait analysis from other view can be a future research direction. Experiments

can be conducted similar to [62] which place the sensor to the ceiling in front of the walking path line. Furthermore, accuracy of more kinematics measurements are suggested to validate the robustness of system in wider range of gait analysis like [177].

2. **robust marker labeling without pre-defined model**

   The model-based marker labeling method in Chapter 2 requires automatic body measurements before performing motion capture which is lack of robustness across different motion capture based healthcare monitoring applications, such as gait analysis from different views, timed-up-and-go test [58]. An application specific pre-defined model like Plug-in-Gait is required, which is not universal for capturing human motion in more generic tasks, such as when additional markers are required during the session. A more generic approach like [178] that incorporates deep learning techniques to automatic label the retro-reflective markers can be a potential research direction to tackle this challenge. To conduct the experiments for using a universal model to label the marker, the following steps can be one of the potential approaches: (1) project the joint trajectory data in public available motion capture dataset [179] (consists of 2600 recordings with 144 different human subjects performing a variety of motions), into random views in order to generate enough synthetic joint trajectory data. (2) for each frame, assume that the marker label (joint name) is unknown to the project joint positions and use the marker positions as input features and the corresponding joint as output categorical label to build a multi-class classification dataset. (3) develop a machine learning model to assign individual categorical label (joint name) to a given set of marker positions in each frame. (4) split the dataset into training, validation, testing sets and train a model to evaluate its predictive performance in the testing set. (5) evaluate the transferability of the model trained on [179] dataset to the acquired trajectory data in Chapter 2. The motivation of the suggested study is to investigate the possibility of using machine learning technique to automatically assign joint names to a set of marker positions.

3. **feature fusion for multi-lead ECG classification**

   In this research, the sensitivity analysis of the feature selection/extraction method in classification task focuses on the fixed two-lead ECG signals using a single feature extraction method. [180] investigates the deep feature fusion based methods for classifying normal/abnormal samples using multi-lead ECG signals. The results show that the proposed multi-lead feature fusion based method [180] can improve the classification accuracy compared to the one only with the single-lead features. However, the target task in [180] is binary classification, which requires a more board study in multi-class case, such as arrhythmia classification. Since currently the method of 12-lead simultaneous recording ECG is very popular in clinical practice, the study of analyzing feature fusion based feature selection/extraction methods on 12-lead ECG signals can be a potential research direction.

4. **graph structure generation learning**

   Without prior knowledge of the underlying graph structure of partially labeled data, in Chapter 5 proposes a CNN-based graph generator to build an initial graph structure to reflect the node-to-node similarity. However, this method requires to heuristically set the maximum degree for each node, which in this thesis is estimated by grid search using a K nearest neighbor classifier. An alternative method can be joining the random walk algorithm and graph sampling algorithm to decrease the number of edges to an appropriate number without sacrificing the underlying graph structure to represent the relationship between samples.

Bibliography

[1] I. Vourganas, V. Stankovic, L. Stankovic, and A. Kerr, "Factors that contribute to the use of stroke self-rehabilitation technologies: A review," *JMIR Biomedical Engineering*, vol. 4, no. 1, e13732, Aug. 2019.

[2] N. Byl, W. Zhang, S. Coo, and M. Tomizuka, "Clinical impact of gait training enhanced with visual kinematic biofeedback: Patients with parkinson's disease and patients stable post stroke," *Neuropsychologia*, vol. 79, pp. 332–343, 2015.

[3] P. Langhorne, F. Coupar, and A. Pollock, "Motor recovery after stroke: A systematic review," *The Lancet Neurology*, vol. 8, no. 8, pp. 741–754, 2009.

[4] C. Yang, U. Ugbolue, A. Kerr, V. Stankovic, L. Stankovic, B. Carse, K. Kaliarntas, and P. Rowe, "Autonomous gait event detection with portable single-camera gait kinematics analysis system," *Journal of Sensors*, vol. 2016, 2015.

[5] S. kook Jun, X. Zhou, D. K. Ramsey, and V. N. Krovi, *A comparative study of human motion capture and analysis tools.*

[6] VICON, *Motion capture*, `http://www.vicon.com`, Mar. 2019.

[7] Qualisys, *Motion analysis*, `http://www.qualisys.com`, Mar. 2019.

[8] T.-Y. Liao, S.-G. Miaou, and Y.-R. Li, "A vision-based walking posture analysis system without markers," in *International Conference on Signal Processing Systems (ICSPS)*, vol. 3, Jul. 2010, pp. 254–258.

[9] H. Nguyen and J. Meunier, "Gait analysis from video: Camcorders vs. kinect," in *Image Analysis and Recognition*, A. Campilho and M. Kamel, Eds., ser. Lecture Notes in Computer Science. Springer, 2014, vol. 8815, ch. 8, pp. 66–73.

[10] U. C. Ugbolue, E. Papi, K. T. Kaliarntas, A. Kerr, L. Earl, V. M. Pomeroy, and P. J. Rowe, "The evaluation of an inexpensive, 2d, video based gait assessment system for clinical use," *Gait & Posture*, vol. 38, pp. 483–489, Jul. 2013.

[11] A. Leu, D Ristić-Durrant, and A. Gräser, "A robust markerless vision-based human gait analysis system," in *IEEE international symposium on applied computational intelligence and informatics (SACI)*, May 2011, pp. 415–420.

[12] C. Yang, U. Ugbolue, B. Carse, V. Stankovic, L. Stankovic, and P. Rowe, "Multiple marker tracking in a single-camera system for gait analysis," in *IEEE International Conference on Image Processing (ICIP)*, Oct. 2013.

[13] Y.-R. Li, S.-G. Miaou, C. K. Hung, and J. T. Sese, "A gait analysis system using two cameras with orthogonal view," in *2011 International Conference on Multimedia Technology (ICMT)*, IEEE, Jul. 2011, pp. 2841–2844.

[14] *Developing with kinect for windows*, `https://developer.microsoft.com/en-us/windows/kinect`, Mar. 2019.

[15] R. Clark, S. Vernon, B. Mentiplay, K. Miller, J. McGinley, Y. Pua, K. Paterson, and K. Bower, "Instrumenting gait assessment using the kinect in people living with stroke: Reliability and association with balance tests," *Journal of NeuroEngineering and Rehabilitation*, vol. 12, no. 1, p. 15, 2015.

[16] W. Wang, K. Li, S. Yue, C. Yin, and N. Wei, "Associations between lower-limb muscle activation and knee flexion in post-stroke individuals: A study on the stance-to-swing phases of gait," *PloS one*, vol. 12, no. 9, 2017.

[17] J.-Y. Jung, W. Heo, H. Yang, and H. Park, "A neural network-based gait phase classification method using sensors equipped on lower limb exoskeleton robots," *Sensors*, vol. 15, no. 11, 27738—27 759, Oct. 2015.

[18] C. Galván-Duque, R. Zavala-Yoé, G. Rodríguez-Reyes, and F. Mendoza-Cruz, "Comparison between classical and intelligent identification systems for classification of gait events," *Journal of Control Science and Engineering*, vol. 1, pp. 21–34, 2015.

[19] B. Carse, B. Meadows, R. Bowers, and P. Rowe, "Affordable clinical gait analysis: An assessment of the marker tracking accuracy of a new low-cost optical 3d motion analysis system," *Physiotherapy*, vol. 99, no. 4, pp. 347–351, 2013.

[20] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3d skeletons as points in a lie group," in *IEEE conference on computer vision and pattern recognition (CVPR)*, Jun. 2014, pp. 588–595.

[21] R. Vemulapalli and R. Chellapa, "Rolling rotations for recognizing human actions from 3d skeletal data," in *IEEE CVPR*, Jun. 2016, pp. 4471–4479.

[22] S. Zhang, Z. Wei, J. Nie, L. Huang, S. Wang, and Z. Li, "A review on human activity recognition using vision-based method," *Journal of Healthcare Engineering*, vol. 2017, Jul. 2017.

[23] Z. Liu and S. Sarkar, "Improved gait recognition by gait dynamics normalization," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 28, no. 6, pp. 863–876, 2006.

[24] A. Ligtenberg and M. Kunt, "A robust-digital qrs-detection algorithm for arrhythmia monitoring," *Computers and Biomedical Research*, vol. 16, no. 3, pp. 273–286, 1983.

[25] M. Alfaras, M. C. Soriano, and S. Ortín, "A fast machine learning model for ecg-based heartbeat classification and arrhythmia detection," *Frontiers in Physics*, vol. 7, p. 103, 2019.

[26] Y. Kaya, H. Pehlivan, and M. E. Tenekeci, "Effective ecg beat classification using higher order statistic features and genetic feature selection.," *Biomedical Research*, vol. 28, no. 17, pp. 7594–7603, 2017.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[28] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," 2014. arXiv: 1406.2080.

[29] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *IEEE CVPR*, 2017, pp. 2233–2241.

[30] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S.-T. Xia, S. Wijewickrema, and J. Bailey, "Dimensionality-driven learning with noisy labels," in *ICML*, 2018, pp. 3361–3370.

[31] P. Lemberger, "On generalization and regularization in deep learning," 2017. arXiv: 1704.01312.

[32] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012. arXiv: 1207.0580.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, pp. 448–456.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.

[35] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE CVPR*, vol. 1, 2017, p. 3.

[36] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *IEEE CVPR*, 2015, pp. 2691–2699.

[37] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," 2018. arXiv: 1802.05300.

[38] H. Masnadi-Shirazi and N. Vasconcelos, "On the design of loss functions for classification: Theory, robustness to outliers, and savageboost," in *NIPS*, 2009, pp. 1049–1056.

[39] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," 2014. arXiv: 1412.6596.

[40] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," 2016.

[41] A. Sandryhaila and J. M. F. Moura, "Classification via regularization on graphs," in *IEEE GlobalSIP*, Dec. 2013, pp. 495–498.

[42] B. Jiang and D. Lin, "Graph laplacian regularized graph convolutional networks for semi-supervised learning," 2018. arXiv: `1809.09839`.

[43] S. J. Olney and C. Richards, "Hemiparetic gait following stroke. part i: Characteristics," *Gait & posture*, vol. 4, no. 2, pp. 136–148, 1996.

[44] M. Ye, C. Yang, V. Stankovic, L. Stankovic, and A. Kerr, "Kinematics analysis multimedia system for rehabilitation," in *International Conference on Image Analysis and Processing (ICIAP)*, Springer, 2015, pp. 571–579.

[45] M. Ye, C. Yang, V. Stankovic, L. Stankovic, and A. Kerr, "Gait analysis using a single depth camera," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec. 2015, pp. 285–289.

[46] M. Ye, C. Yang, V. Stankovic, L. Stankovic, and A. Kerr, "A depth camera motion analysis framework for tele-rehabilitation: Motion capture and person-centric kinematics analysis," *IEEE Journal of Selected Topics in Signal Processing (JSTSP)*, vol. 10, pp. 877–887, 5 Aug. 2016.

[47] M. Ye, C. Yang, V. Stankovic, L. Stankovic, and S. Cheng, "Gait phase classification for in-home gait assessment," in *IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, Jul. 2017, pp. 1524–1529.

[48] M. Ye, C. Yang, V. Stankovic, L. Stankovic, and S. Cheng, "Distinct feature extraction for video-based gait phase classification," in *IEEE Transactions on Multimedia (TMM)*, IEEE, in press 2019.

[49] M. Ye, V. Stankovic, L. Stankovic, and G. Cheung, "Deep graph regularized learning for binary classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 3537–3541.

[50] A. M. de-la Herran, B. Garcia-Zapirain, and A. Mendez-Zorrilla, "Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications," *Sensors*, vol. 14, no. 2, pp. 3362–3394, Feb. 2014.

[51] C. Yang, A. Kerr, V. Stankovic, L. Stankovic, P. Rowe, and S. Cheng, "Human upper limb motion analysis for post-stroke impairment assessment using video analytics," *IEEE Access*, vol. 4, pp. 650–659, 2016.

[52] J.-T. Zhang, A. C. Novak, B. Brouwer, and Q. Li, "Concurrent validation of xsens mvn measurement of lower limb joint angular kinematics," *Physiological Measurement*, vol. 34, no. 8, pp. 63–69, 2013.

[53] *M3d force plate*, `http://www.tecgihan.co.jp/en/products/m3d_forceplate/`, Mar. 2019.

[54] *Softkinetic depthsense camera*, Mar. 2016. [Online]. Available: `http://www.softkinetic.com/Products/DepthSenseCameras`.

[55] *Intel® realsense™ camera (r200)*, Mar. 2016. [Online]. Available: `https://software.intel.com/en-us/RealSense/R200Camera`.

[56] S. Sarkar, L. Stankovic, A. Kerr, and P. Rowe, "Kinect-based lower limb motion analysis," in *2015 XXV Congress International Society of Biomechanics*, Jul. 2015.

[57] L. Wiedemann, R Planinc, I Nemec, and M Kampel, "Performance evaluation of joint angles obtained by the kinect v2," in *IET International Conference on Technologies for Active and Assisted Living (TechAAL)*, IET, Nov. 2015, pp. 1–6.

[58] O. Lohmann, T. Luhmann, and A. Hein, "Skeleton timed up and go," in *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Oct. 2012, pp. 1–5.

[59] E. Cippitelli, S. Gasparrini, S. Spinsante, and E. Gambi, "Kinect as a tool for gait analysis: Validation of a real-time joint extraction algorithm working in side view," *Sensors*, vol. 15, no. 1, pp. 1417–1434, 2015.

[60] M. Reyes, A. Clapés, J. Ramírez, J. R. Revilla, and S. Escalera, "Automatic digital biometry analysis based on depth maps," *Computers in Industry*, vol. 64, no. 9, pp. 1316–1325, 2013.

[61] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys, "Accurate 3d pose estimation from a single depth image," in *International Conference on Computer Vision (ICCV)*, IEEE, Nov. 2011, pp. 731–738.

[62] M. Gabel, E. Renshaw, A. Schuster, and R. Gilad-Bachrach, "Full body gait analysis with kinect," in *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Aug. 2012, pp. 1964–1967.

[63] J. Steward, D. Lichti, J. Chow, and R. Ferber, "Performance assessment and calibration of the kinect 2.0 time-of-flight range camera for use in motion capture applications,"

[64] C. Kerrigan, M. Schaufele, and M. Wen, "Gait analysis," in *Rehabilitation Medicine: Principles and Practice*, 1998, pp. 167–174.

[65] J. Sell and P. O'Connor, "The xbox one system on a chip and kinect sensor," *IEEE Micro*, vol. 34, no. 2, pp. 44–53, 2014.

[66] L. Yang, L. Zhang, H. Dong, A. Alelaiwi, and A. El Saddik, "Evaluating and improving the depth accuracy of kinect for windows v2," *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4275–4285, Aug. 2015.

[67] H. Samet and M. Tamminen, "Efficient component labeling of images of arbitrary dimension represented by linear bintrees," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 10, no. 4, pp. 579–586, Jul. 1988.

[68] E Lachat, H Macher, M. Mittet, T Landes, and P Grussenmeyer, "First experiences with kinect v2 sensor for close range 3d modelling," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, vol. 40, no. 5, p. 93, 2015.

[69] F. Lenzen, H. Schäfer, and C. Garbe, "Denoising time-of-flight data with adaptive total variation," in *International Symposium on Visual Computing*, Springer, 2011, pp. 337–346.

[70] W. Hu, X. Li, G. Cheung, and O. Au, "Depth map denoising using graph-based transform and group sparsity," in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, Sep. 2013, pp. 1–6.

[71] J. Shen and S.-C. S. Cheung, "Layer depth denoising and completion for structured-light rgb-d cameras," in *IEEE CVPR*, Jun. 2013, pp. 1187–1194.

[72] Y. S. Kim, B. Kang, H. Lim, O. Choi, K. Lee, J. D. Kim, and C. Kim, "Parametric model-based noise reduction for tof depth sensors," in *Three-Dimensional Image Processing (3DIP) and Applications II*, International Society for Optics and Photonics, vol. 8290, 2012.

[73] M. Camplani and L. Salgado, "Adaptive spatio-temporal filter for low-cost camera depth maps," in *IEEE International Conference on Emerging Signal Processing Applications (ESPA)*, Jan. 2012, pp. 33–36.

[74] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[75] C. Steger, "Subpixel-precise extraction of watersheds," in *IEEE ICCV*, vol. 2, 1999, pp. 884–890.

[76] W. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct points, corners and centres of circular features," in *Intercommission conference on fast processing of photogrammetric data*, Interlaken, 1987, pp. 281–305.

[77] U. Köthe, "Edge and junction detection with an improved structure tensor," in *Pattern Recognition*, Springer, 2003, pp. 25–32.

[78] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision (IJCV)*, vol. 30, pp. 79–116, 1998.

[79] S. Hinz, "Fast and subpixel precise blob detection and attribution," in *IEEE International Conference on Image Processing*, IEEE, vol. 3, 2005, pp. 457–460.

[80] M. K. Agosto, "Bezier curves," in *Computer Graphics and Geometric Modelling: Implementation & Algorithms*, 2005, pp. 396–404.

[81] *Kinect for xbox 360*, Mar. 2016. [Online]. Available: `http://www.xbox.com/en-US/xbox-360/accessories/kinect`.

[82] E. Papi, U. C. Ugbolue, and P. J. Rowe, "System dependent differences in the accuracy of gait kinematics: A comparison of a modern 12 camera vicon nexus system with an older 8 camera vicon 612," in *XXIV Congress of the International Society of Biomechanics (ISB 2013)*, 2013.

[83] J. Zhao, F. E. Bunn, J. M. Perron, E. Shen, and R. S. Allison, "Gait assessment using the kinect rgb-d sensor," in *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2015, pp. 6679–6683.

[84] F. Gholami, D. A. Trojan, J. Kövecses, W. M. Haddad, and B. Gholami, "A microsoft kinect-based point-of-care gait assessment framework for multiple sclerosis patients," *IEEE journal of biomedical and health informatics (JBHI)*, vol. 21, no. 5, pp. 1376–1385, 2017.

[85] Q. Wang, G. Kurillo, F. Ofli, and R. Bajcsy, "Unsupervised temporal segmentation of repetitive human actions based on kinematic modeling and frequency analysis," in *International conference on 3D vision (3DV)*, IEEE, 2015, pp. 562–570.

[86] R. Gross and J. Shi, "The cmu motion of body (mobo) database," Tech. Rep., Jul. 2001.

[87] J. D. Farah, N. Baddour, and E. D. Lemaire, "Gait phase detection from thigh kinematics using machine learning techniques," in *IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, May 2017, pp. 263–268.

[88] C. wei Hsu, C. chung Chang, and C. jen Lin, *A practical guide to support vector classification*, `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`, Mar. 2019.

[89] S. Mulroy, J. Gronley, W. Weiss, C. Newsam, and J. Perry, "Use of cluster analysis for gait pattern classification of patients in the early and late recovery phases following stroke," *Gait & posture*, vol. 18, no. 1, pp. 114–125, 2003.

[90] J. Bae, "Gait analysis based on a hidden markov model," in *2012 12th International Conference on Control, Automation and Systems*, IEEE, 2012, pp. 1025–1029.

[91] "Delisa's physical medicine & rehabilitation: Principles and practice," in, W. R. Frontera, Ed., 5th ed. Lippincott Williams & Wilkins, 2010, vol. 1, ch. 5, pp. 121–137.

[92] B. Pietraszewski, S. Winiarski, and S. Jaroszczuk, "Threedimensional human gait pattern–reference data for normal men," *Acta of Bioengineering and Biomechanics*, vol. 14, no. 3, pp. 9–16, 2012.

[93] P. Sharma and Y. Rathi, "Efficient density-based clustering using automatic parameter detection," in *International Congress on Information and Communication Technology (ICICT)*, Springer, 2016, pp. 433–441.

[94] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[95] J.-S. Wang, C.-W. Lin, Y.-T. C. Yang, and Y.-J. Ho, "Walking pattern classification and walking distance estimation algorithms using gait phase information," *IEEE Transactions on Biomedical Engineering (TBME)*, vol. 59, no. 10, pp. 2884–2892, Oct. 2012.

[96] M. Zhu, J. Xia, X. Jin, M. Yan, G. Cai, J. Yan, and G. Ning, "Class weights random forest algorithm for processing class imbalanced medical data," *IEEE Access*, vol. 6, pp. 4641–4652, 2018.

[97] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[98] J Shotton, R Girshick, A Fitzgibbon, T Sharp, M Cook, M Finocchio, R Moore, P Kohli, A Criminisi, A Kipman, *et al.*, "Efficient human pose estimation from single depth images," in *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013, pp. 175–192.

[99] S Bharathidason and C. J. Venkataeswaran, "Improving classification accuracy based on random forest model with uncorrelated high performing trees," *International Journal of Computer Applications (IJCA)*, vol. 101, no. 13, pp. 26–30, Sep. 2014.

[100] Y. Kozlov and T. Weinkauf, *Persistence1d: Extracting and filtering minima and maxima of 1d functions*, `https://www.csc.kth.se/~weinkauf/notes/persistence1d.html`, Mar. 2019.

[101] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[102] VICON, *Plug-in-gait model*, `https://www.vicon.com/downloads/documentation/plug-in-gait-model-details`, Mar. 2019.

[103] M. Schwartz and P. C. Dixon, "The effect of subject measurement error on joint kinematics in the conventional gait model: Insights from the open-source pycgm tool using high performance computing methods," *PloS one*, vol. 13, no. 1, 2018.

[104] Y. Yang, F. Pu, Y. Li, S. Li, Y. Fan, and D. Li, "Reliability and validity of kinect rgb-d sensor for assessing standing balance," *IEEE Sensors Journal*, vol. 14, no. 5, pp. 1633–1638, May 2014.

[105] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[106] Q. Zou, L. Ni, Q. Wang, Q. Li, and S. Wang, "Robust gait recognition by integrating inertial and rgbd sensors," *IEEE Transactions on Cybernetics*, vol. 48, no. 4, pp. 1136–1150, 2018.

[107] L. Lo Presti and M. La Cascia, "3d skeleton-based human action classification," *Pattern Recognition*, vol. 53, no. C, pp. 130–147, 2016.

[108] C. Zhang, J. Zhang, and J. Hong, "Classification of eeg signals using multiple gait features based on small-world neural network," in *International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, IEEE, Aug. 2016, pp. 61–66.

[109] L. R. Rabiner and B.-H. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[110] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[111] J. Taborri, S. Rossi, E. Palermo, F. Patanè, and P. Cappa, "A novel hmm distributed classifier for the detection of gait phases by means of a wearable inertial sensor network," *Sensors*, vol. 14, no. 9, 16212—16 234, Sep. 2014.

[112] A. Mannini and A. M. Sabatini, "Gait phase detection and discrimination between walking–jogging activities using hidden markov models applied to foot motion data from a gyroscope," *Gait & Posture*, vol. 36, no. 4, pp. 657–661, 2012.

[113] N. Abaid, P. Cappa, E. Palermo, M. Petrarca, and M. Porfiri, "Gait detection in children with and without hemiplegia using single-axis wearable gyroscopes," *PLOS ONE*, vol. 8, no. 9, pp. 1–8, Sep. 2013.

[114] R. C. González, A. M. López, J. Rodriguez-Uría, D. Alvarez, and J. C. Alvarez, "Real-time gait event detection for normal subjects from lower trunk accelerations," *Gait & posture*, vol. 31, no. 3, pp. 322–325, 2010.

[115] D. A. Coast, R. M. Stern, G. G. Cano, and S. A. Briller, "An approach to cardiac arrhythmia analysis using hidden markov models," *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 9, pp. 826–836, Sep. 1990.

[116] P. R. Gomes, F. O. Soares, J. H. Correia, and C. S. Lima, "Cardiac arrhythmia classification using wavelets and hidden markov models – a comparative approach," in *International Conference of the IEEE Engineering in Medicine and Biology Society*, Sep. 2009, pp. 4727–4730.

[117] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[118] M. Hagui and M. A. Mahjoub, "Hidden conditional random fields for gait recognition," in *International Image Processing, Applications and Systems (IPAS)*, Nov. 2016, pp. 1–6.

[119] G. de Lannoy, D. Francois, J. Delbeke, and M. Verleysen, "Weighted conditional random fields for supervised interpatient heartbeat classification," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 1, pp. 241–247, Jan. 2012.

[120] G. B. Moody and R. G. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, May 2001.

[121] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[122] K.-B. Duan and S. S. Keerthi, "Which is the best multiclass svm method? an empirical study," in *International workshop on multiple classifier systems*, Springer, 2005, pp. 278–285.

[123] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[124] L. Nanni, S. Ghidoni, and S. Brahnam, "Handcrafted vs. non-handcrafted features for computer vision classification," *Pattern Recognition*, vol. 71, pp. 158–172, 2017.

[125] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[126] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[127] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[128] S Gayathri, M Suchetha, and V Latha, "Ecg arrhythmia detection and classification using relevance vector machine," *Procedia engineering*, vol. 38, pp. 1333–1339, 2012.

[129] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ecg morphology and heartbeat interval features," *IEEE transactions on biomedical engineering*, vol. 51, no. 7, pp. 1196–1206, 2004.

[130] F. A. Elhaj, N. Salim, A. R. Harris, T. T. Swee, and T. Ahmed, "Arrhythmia recognition and classification using combined linear and nonlinear features of ecg signals," *Computer Methods and Programs in Biomedicine*, vol. 127, no. C, pp. 52–63, 2016.

[131] T. Teijeiro, P. Félix, J. Presedo, and D. Castro, "Heartbeat classification using abstract features from the abductive interpretation of the ecg," *IEEE journal of biomedical and health informatics*, vol. 22, no. 2, pp. 409–420, 2016.

[132] T. G. Dietterich, "Machine learning for sequential data: A review," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, Springer, 2002, pp. 15–30.

[133] G. Kaur and A. Chhabra, "Improved j48 classification algorithm for the prediction of diabetes," *International Journal of Computer Applications*, vol. 98, no. 22, 2014.

[134] S. Nikan, F. Gwadry-Sridhar, and M. Bauer, "Pattern recognition application in ecg arrhythmia classification," *BIOSTEC 2017*, p. 48, 2017.

[135] M. Lagerholm, C. Peterson, G. Braccini, L. Edenbrandt, and L. Sornmo, "Clustering ecg complexes using hermite functions and self-organizing maps," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 7, pp. 838–848, 2000.

[136] D. Weimer, B. Scholz-Reiter, and M. Shpitalni, "Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection," *CIRP Annals*, vol. 65, no. 1, pp. 417–420, 2016.

[137] M. Speriosu, N. Sudan, S. Upadhyay, and J. Baldridge, "Twitter polarity classification with label propagation over lexical links and the follower graph," in *Proceedings of the First workshop on Unsupervised Learning in NLP*, 2011, pp. 53–63.

[138] J. Tang, R. Hong, S. Yan, T.-S. Chua, G.-J. Qi, and R. Jain, "Image annotation by knn-sparse graph-based label propagation over noisily tagged web images," *ACM TIST*, vol. 2, no. 2, p. 14, 2011.

[139] G. Cheung, W. Su, Y. Mao, and C. Lin, "Robust semisupervised graph classifier learning with negative edge weights," *IEEE T-SIPN*, vol. 4, no. 4, pp. 712–726, Nov. 2018.

[140] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk, "Iteratively reweighted least squares minimization for sparse recovery," *Communications on Pure and Applied Mathematics*, vol. 63, no. 1, pp. 1–38, 2010.

[141] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari, "Learning object class detectors from weakly annotated video," in *IEEE CVPR*, 2012, pp. 3282–3289.

[142] I. Misra, A. Shrivastava, and M. Hebert, "Watch and learn: Semi-supervised learning for object detectors from video," in *IEEE CVPR*, 2015, pp. 3593–3602.

[143] A. Kuznetsova, S. Ju Hwang, B. Rosenhahn, and L. Sigal, "Expanding object detector's horizon: Incremental learning framework for object detection in videos," in *IEEE CVPR*, 2015, pp. 28–36.

[144] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," 2018. arXiv: 1803.09050.

[145] I. Jindal, M. Nokleby, and X. Chen, "Learning deep networks from noisy labels with dropout regularization," in *IEEE ICDM*, 2016, pp. 967–972.

[146] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit, *et al.*, "Openimages: A public dataset for large-scale multi-label and multi-class image classification," *Dataset available from https://github. com/openimages*, vol. 2, no. 3, p. 18, 2017.

[147] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "Yfcc100m: The new data in multimedia research," *Communications of the ACM*, vol. 59, no. 2, pp. 64–73, 2016.

[148] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, "Deep learning is robust to massive label noise," 2017. arXiv: 1705.10694.

[149] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *IEEE CVPR*, 2015, pp. 2691–2699.

[150] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia, "Iterative learning with open-set noisy labels," 2018. arXiv: 1804.00092.

[151] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013. arXiv: 1312.6203.

[152] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016. arXiv: 1609.02907.

[153] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 3844–3852.

[154] X. Zhang, C. Xu, and D. Tao, "Graph edge convolutional neural networks for skeleton based action recognition," 2018. arXiv: 1805.06184.

[155] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," 2018. arXiv: 1801.07829.

[156] J. Mojoo, K. Kurosawa, and T. Kurita, "Deep cnn with graph laplacian regularization for multi-label image annotation," in *ICIAR*, Springer, 2017, pp. 19–26.

[157] J. Zeng, J. Pang, W. Sun, and G. Cheung, "Deep graph laplacian regularization for robust denoising of real images," 2019.

[158] C. Yang, G. Cheung, and V. Stankovic, "Alternating binary classifier and graph learning from partial labels," in *APSIPA*, Nov. 2018.

[159] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*, Springer, 2015, pp. 84–92.

[160] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *International Conference on Neural Information Processing Systems*, ser. NIPS, Vancouver, British Columbia, Canada: MIT Press, 2004, pp. 1601–1608.

[161] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under laplacian and structural constraints," *IEEE JSTSP*, vol. 11, no. 6, pp. 825–841, 2017.

[162] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.

[163] J. Alcala-Fdez, A. Fernández, J. Luengo, J Derrac, S Garc'ia, L. Sanchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, pp. 255–287, Jan. 2010.

[164] Y Wang, Z Pan, and Y Pan, "A training data set cleaning method by classification ability ranking for the k-nearest neighbor classifier.," *IEEE transactions on neural networks and learning systems*, 2019.

[165] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *ACML*, vol. 20, South Garden Hotels and Resorts, Taoyuan, Taiwain, Nov. 2011, pp. 97–112.

[166] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of machine learning research*, vol. 5, no. Jan, pp. 101–141, 2004.

[167] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. arXiv: 1412.6980.

[168] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[169] X. Ying and X. Wu, "Graph generation with prescribed feature constraints," in *Proceedings of the 2009 SIAM International Conference on Data Mining*, SIAM, 2009, pp. 966–977.

[170] D. Thewlis, C. Bishop, N. Daniell, and G. Paul, *A comparison of two commercially available motion capture systems for gait analysis: High end vs low-cost.* 2011.

[171] S. Han, B. Liu, R. Wang, Y. Ye, C. D. Twigg, and K. Kin, "Online optical marker-based hand tracking with deep labels," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–10, 2018.

[172] H. Kong, H. C. Akakin, and S. E. Sarma, "A generalized laplacian of gaussian filter for blob detection and its applications," *IEEE transactions on cybernetics*, vol. 43, no. 6, pp. 1719–1733, 2013.

[173] T. Lindeberg, "Feature detection with automatic scale selection," *International journal of computer vision*, vol. 30, no. 2, pp. 79–116, 1998.

[174] B. Ulrich, A. N. Santos, B. M. Jolles, D. H. Benninger, and J. Favre, "Gait events during turning can be detected using kinematic features originally proposed for the analysis of straight-line walking," *Journal of biomechanics*, vol. 91, pp. 69–78, 2019.

[175] M. W. Whittle, "Clinical gait analysis: A review," *Human movement science*, vol. 15, no. 3, pp. 369–387, 1996.

[176] M. Greenberg, J. Gronley, J Perry, and R Lewthwaite, "Concurrent validity of observational gait analysis using the vicon motion analysis system," *Gait & Posture*, vol. 2, no. 4, pp. 167–168, 1996.

[177] J. Latorre, C. Colomer, M. Alcañiz, and R. Llorens, "Gait analysis with the kinect v2: Normative study with healthy individuals and comprehensive study of its sensitivity, validity, and reliability in individuals with stroke," *Journal of neuroengineering and rehabilitation*, vol. 16, no. 1, p. 97, 2019.

[178] A. Chatzitofis, D. Zarpalas, S. Kollias, and P. Daras, "Deepmocap: Deep optical motion capture using multiple depth sensors and retro-reflectors," *Sensors*, vol. 19, no. 2, p. 282, 2019.

[179] *Cmu graphics lab motion capture database*, `http://mocap.cs.cmu.edu`, Jan. 2020.

[180] B. Chen, W. Guo, B. Li, R. K. F. Teng, M. Dai, J. Luo, and H. Wang, "A study of deep feature fusion based methods for classifying multi-lead ecg," 2018. arXiv: `1808.01721`.

[181] *Kinect for windows features: Kinect sensor key features and benefits*, Mar. 2016. [Online]. Available: `https://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx`.

[182] *Kinect hardware requirements and sensor setup*, `https://dev.windows.com/en-us/kinect/hardware-setup`, Mar. 2017.

Appendix A

Pilot Study Protocol for Gait Analysis

Research Group

Philip Rowe, Biomechanics Unit, Biomedical Engineering, University of Strathclyde;
Andrew Kerr, Biomechanics Unit, Biomedical Engineering, University of Strathclyde;
Vladimir Stankovic, Electronic and Electrical Engineering, University of Strathclyde;
Lina Stankovic, Electronic and Electrical Engineering, University of Strathclyde;
Shikha Sarkar, Electronic and Electrical Engineering, University of Strathclyde;
Cheng Yang, Electronic and Electrical Engineering, University of Strathclyde.
Minxiang Ye, Electronic and Electrical Engineering, University of Strathclyde.

Protocol Title

Validation of a 2D single camera video system and a 3D single camera video system to measure limb motor control: A pilot study

Design

Pilot validation of a two and three dimensional video system to measure metrics of limb motor control. Concurrent comparison will be made with a state of the art motion analysis (VICON, Oxford, UK).

Sample

A pilot sample of 5-10 healthy subjects.

The small sample size (5-10) is considered sufficient to provide the mean and variance data for a power calculation of sample size for a future larger study as well a testing the feasibility of procedures.

Recruitment

Individuals will be recruited from the staff and student community of the University of Strathclyde. Interested individuals being given information sheets.

Data Collection Protocol

Interested individuals will be offered a two hour appointment in the large biomechanics laboratory (Wolfson Building, Biomedical Engineering, University of Strathclyde) with dates and times arranged at the convenience of the interested individual.

Taking Consent

On arrival at the biomechanics laboratory one of the research team will provide a short introduction to the biomechanics unit as well as the motion analysis and video camera systems. The objectives of the study will be explained again to the participants and they will be given an opportunity to ask any questions. If the individual is still happy to proceed they will be asked to sign a consent form. At this point they will be assigned a study number, *e.g.* PVT001, and referred to thereafter as study participant number PVT00[No.].

Participant Preparation

Each participant will then be helped to put on lycra clothing, if the clothes they are wearing are considered too loose fitting. To help construct the three dimensional model each participant will then have the following dimensions measured and recorded in their individual case report file in addition to their age and gender:

Height – cm;

Weight – Kg;

Arm length - cm;

Shoulder width – cm;

Elbow width – cm;

Wrist width – cm;

Leg length – cm;

Pelvis width – cm;

Knee width – cm;

Ankle width - cm;

Retro-reflective markers will then be attached to the skin (or clothes) overlying the following anatomical points:

- Upper limb:

posterior superior iliac spine;

anterior superior iliac spine;

mid iliac crest;

spinous process of 7th cervical vertebra;

tragus of ear;

most lateral border of the acromion process;

mid humerus;

lateral and medial epicondyles of humerus;

radial styloid process;

head of ulna.

- Lower limb:

anterior superior iliac spine;

front waist;

back waist;

tip of the big toe;

outside of the thigh below hand swing;

outside of the knee joint;

outside of the lower leg;

bony prominence on the outside of the ankle;

back of the foot;

outside of the foot at the base of the little toe;

tip of the big toe.

For the 2D and 3D video analysis, circular sticky paper labels marked with a concentric black and white pattern will be attached to the wrist [radial styloid process], elbow [lateral epicondyles of humerus], shoulder [most lateral border of the acromion process], pelvis [mid iliac crest), and head (tragus of ear), for upper limb; pelvis, knee, and ankle, for lower limb. Where these landmarks already have the retro-reflective markers attached the paper labels will be placed directly under the reflective marker which will be attached to the center of the paper label. This has worked well in a previous validation study for the lower limb (Ugbolue et al 2012).

For upper limb movement analysis, participants will sit in a standard sized armless chair in front of an adjustable table in the middle of the laboratory with their affected arm resting on the table. For lower limb movement analysis, participants will walk on a scaled mat (6 meters long, 0.7 meter wide) in one direction, and the other way around.

For the 2D video analysis, a high speed camera (Camera A) will be mounted on a tripod and position approximately 4m from the participant in line with their elbow. For the 3D video analysis, a conventional camera with a distance sensor will be mounted above Camera A.

Movement Tasks

For upper limb movement analysis, when the starting position has been achieved and the participant is happy to continue they will be asked to perform the following movements three times each giving a total of 15 movements.

1. Reach forward to touch a plastic cup positioned on the table directly in front of them at a distance equivalent to 80;

2. Reach forward to lift the same plastic cup towards their mouth. Instruction: bring the cup to your mouth;

3. Reach forward to lift the same plastic cup and turn it. Instruction: reach and turn cup over;
4. Reach forward to touch a plastic cup positioned on the table toward their unaffected at a distance equivalent to 20;
5. Lift their hand to touch the back of their head.

For lower limb movement analysis, again, when the starting position has been achieved and the participant is happy to continue they will be asked to perform the following movements three times each giving a total of 6 movements.

1. Walk from one end (Point A) of the scaled mat to the other (Point B);
2. Walk from Point B to Point A.

Participants will be asked to perform each movement as naturally as they can.

Once the movements have all been attempted the participant will be thanked for their participation and markers removed from their body. This will conclude their participation in the study.

Data Storage

The case report file (using participants study number) will be stored in a locked cabinet for the duration of the study *i.e.* until September 2014. Video files will be stored on an encrypted external hard drive and locked in the same cabinet. After the study has ended the video files will be permanently deleted and the paper records of the data kept in storage (locked cabinet in the Biomedical Engineering, Department) for a period of 5 years. Processed results will be made available for publication and to inform a grant application for a larger study.

Statistical Analysis

Data from the three systems (3D motion analysis and 2D and 3D video analysis) will be compared for each movement using analysis of variance and intra class correlation coefficients. The variables for comparison will be:

1. Movement duration;
2. Maximum forward tilt of trunk;

3. Magnitude of angular displacement at shoulder and elbow;
4. Relative timing of trunk, shoulder and elbow movements;
5. Magnitude of angular displacement at knee;
6. Relative timing of hip, knee, and ankle movements.

This data will be used to inform a power calculation of sample size for a second larger study planned for next year which will investigate validity, reliability and responsiveness of the video system with a larger sample including stroke survivors as participants.

Appendix B

Optical Kinematics Analysis (OPTIKA) Software

B.1 Introduction

OPTIKA is a software package for analyzing motions(gait) in a easy way via a single MS Kinect camera. It is written by me using Visual C++ and C# during my PhD. This appendix provide an operation manual for the OPTIKA software which autonomously tracks the movement of markers attached to the body and measures kinematics parameters related to joint movement and velocity. OPTIKA software is equipped with the latest state-of-the-art signal processing and computer vision algorithms to determine motion patterns of interest from infrared and depth data. OPTIKA is portable and easy to set up, thus suitable for local clinics and home use, without requiring any clinical expertise to operate. It is independent of the color of the fabric of subjects' clothing or lighting conditions. Motion capture is enabled using only a single camera (infrared + depth), which is portable and cost effective compared to most industry-standard optical systems, without compromising on accuracy.

A user-friendly interface to the proposed underlying framework of OPTIKA is provided for a gait analysis application. It supports the following features:

1. Recording of infrared and depth images, for example, using MS Kinect SDK [14].

2. Real-time camera/scene calibration.

3. Real-time subject modeling for sagittal plane.

4. Multi-view tracking and 3D trajectory reconstruction.

5. Kinematics analysis using customized analyzers.

6. Rehabilitation diagnostics interface using local or cloud database.

Figure B.1: GUI Interface for Multi-view Tracking

Fig. B.1 shows the snapshot of the graphical user interface (GUI), which shows how convenient it is to access the recorded experiments by selecting the walking component. Users can also view the automatic reconstruction process within the multimedia application or manually playback the whole experiment. Autonomous analysis is performed and gait associated parameters are generated afterwards. These data (including joint angles, movement patterns, gait phases, step and stride length, swing and stance duration) can be easily accessed within the analyzer toolbox. For the rehabilitation applications, a diagnostics interface is developed to report the patient's condition. The following sections describe which components are needed, including sensor and markers, how to set-up the sensor camera, place the markers and use the software. The kinematics engine currently calculates the following parameters:

1. Step and stride length.
2. Swing and stance duration.
3. Ankle, knee and hip angle during gait - in the sagittal plane.
4. Movement of the pelvis around the horizontal axis.
5. Velocity and acceleration of the ankle, knee and hip joint.

B.2 Equipment

1. MS Kinect v2 for Windows or MS Kinect v2 for Xbox One and Adapter for Windows [181].
2. A Kinect compatible computer/laptop. Please use Kinect Configuration Verifier tool to check compatibility [182].
3. Retro-Reflective Markers (19mm, see Table. B.1 for the number of markers needed).
4. Tape, scissors or marked walking mat.

B.3 Environment Setup

This setup is only performed if the camera location is changed before measurements. There are three steps as below:

1. Place walking mat or walking line (tape) 4 meters in length. Mark the center (indicated by 'Model Point') - see Fig. B.2.
2. Place Kinect camera on a tripod 2.8 meters away from the center – see Fig. B.3.
3. Place the Floor Tool on the model point and use OPTIKA Scene Scanner for scanning the scene.

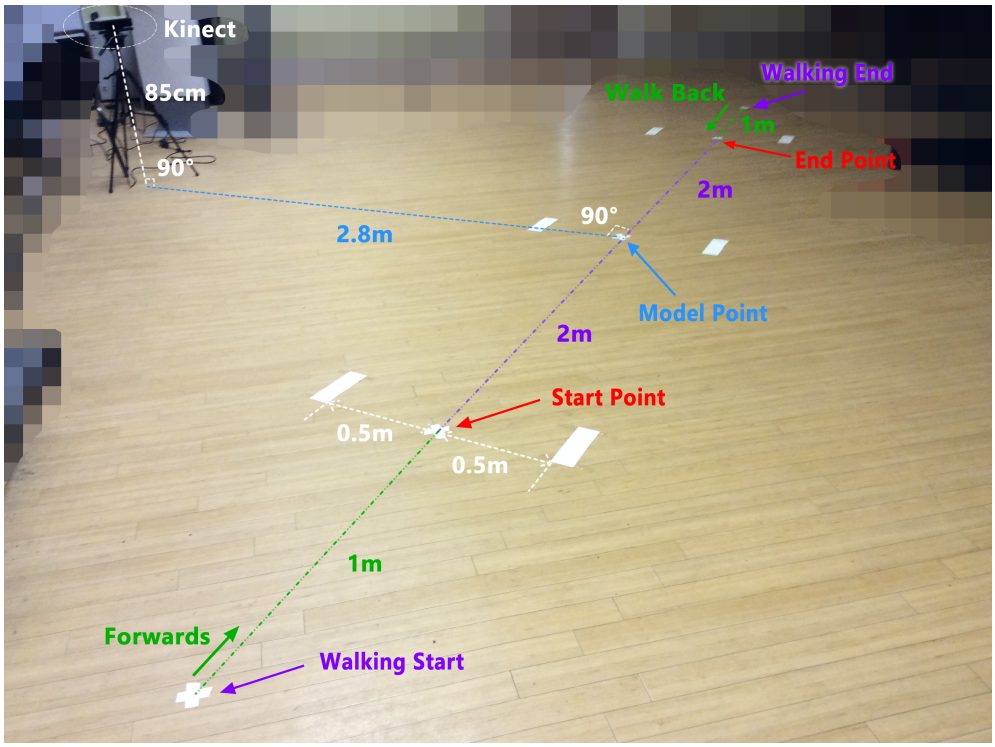


Figure B.2: Physical environment setup 1

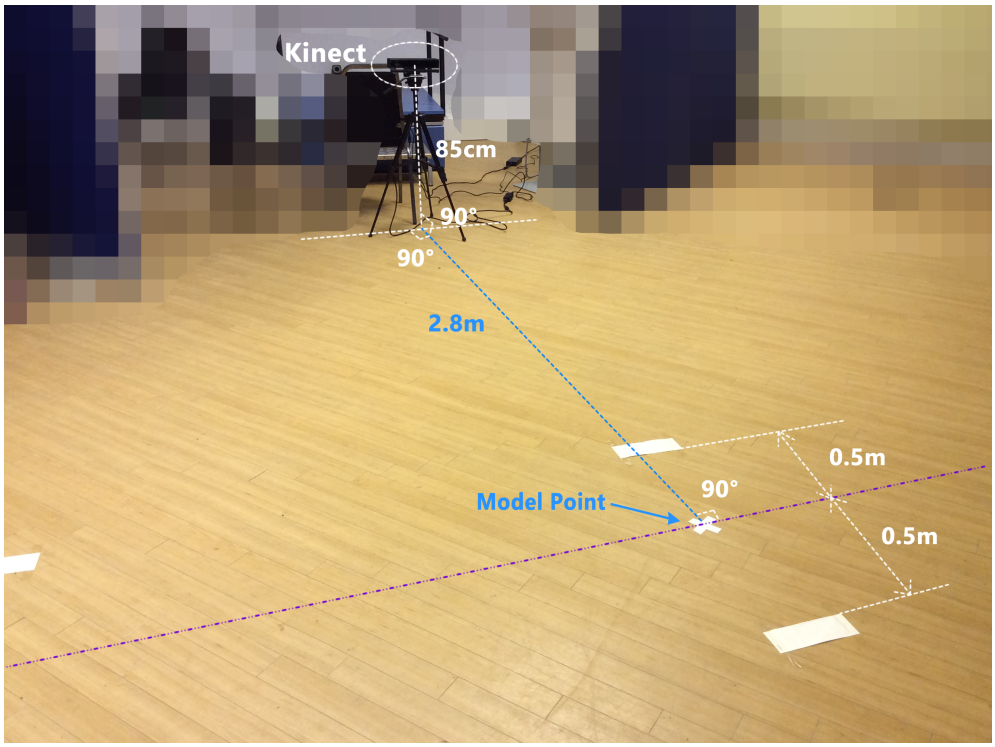


Figure B.3: Physical environment setup 2

B.4 Subject Setup

The following two steps should be executed for each new subject before measurements take place.

Marker location	Knee Angle	Ankle Angle	Hip Angle	Pelvis Tilt	Step	Stride	Swing	Stance	Gait Velocity
Shoulder	1	1	1	1	1	1	1	1	1
Hip	1	1	1	1	1	1	1	1	1
ASIS				1					
PSIS				1					
Knee	1	1	1	1	1	1	1	1	1
Ankle	1	1			1	1	1	1	1
Toe					2	2	2	2	2
Heel					2	2	2	2	2

Table B.1: Marker Placement. 1: Marker always present during all experiments. 2: Marker should be placed on both left and right feet.

1. Place markers on subject's body depending on the required measurements – see Table. B.1. The first column lists the joint position. The first row names the required parameters to be calculated. For example, if step length is needed, markers should be placed on the shoulder, hip, knee, ankle joint, as well as toe and heel of both left and right foot.

2. Use OPTIKA Subject Calibrator and follow the instructions/prompts. a) Ask subject to stand at the model point such that their left sagittal body is facing the camera. b) Ask subject to turn the opposite way such that the right sagittal faces the camera. Note: Ensure all markers are visible on your screen during the modeling process.

B.5 Measurement

1. Use OPTIKA Tracker tool and follow the prompts. a) Ask subject to walk from walking start to walking end or from walking end to walking start. b) Select

desired mode (e.g. Left Sagittal Full Mode is walking from right to left when looking from camera and all markers in Table. B.1 are attached). c) Press start button when subject starts walking. d) Press stop button when subject returns.

2. Press save button to store the results for reviewing.

3. Review previous video and correct marker labels through marker manager.

4. Select desired analyzer for processing the kinematics and the results will be exported as different formats.