GOA COLLEGE OF ENGINEERING
FARMAGUDI, GOA

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION
ENGINEERING

2016 - 2017

HUMAN IDENTIFICATION USING
GAIT RECOGNITION

by

Azfar Khoja (P.R.No.: 201305838)
John George (P.R.No.: 201305821)
Mrinal Shinde(P.R.No.: 201305759)
Neviya Prakash (P.R.No.: 201305942)

A project submitted
in partial fulfilment of the requirements
for the degree of
Bachelor of Engineering
in
Electronics and Telecommunication Engineering
GOA UNIVERSITY

under the guidance of

Prof. Milind Fernandes

Assistant Professor,

Electronics & Telecommunication Department
Goa College of Engineering

CERTIFICATE

This is to certify that the project entitled

"HUMAN IDENTIFICATION USING GAIT RECOGNITION"

submitted by

Azfar Khoja	**P.R. No.: 201305838**
John George	**P.R. No.: 201305821**
Mrinal Shinde	**P.R. No.: 201305759**
Neviya Prakash	**P.R. No.: 201305942**

has been successfully completed in the academic year 2016-2017 as a partial fulfilment of the requirement for the degree of BACHELOR OF ENGINEERING in Electronics and Telecommunication Department, at Goa College of Engineering, Farmagudi.

_____

(Internal Examiner)   (External Examiner)

Place: Farmagudi, Ponda, Goa
Date:

PROJECT APPROVAL SHEET

The project entitled

"HUMAN IDENTIFICATION USING GAIT RECOGNITION"

by

Azfar Khoja	**P.R. No.: 201305838**
John George	**P.R. No.: 201305821**
Mrinal Shinde	**P.R. No.: 201305759**
Neviya Prakash	**P.R. No.: 201305942**

completed in the year 2016-2017 is approved as a partial fulfilment of the requirements for the degree of **BACHELOR OF ENGINEERING in Electronics and Telecommunication Engineering** and is a record of bonafide work carried out successfully under our guidance.

_____

(Project Guide)
Milind Fernandes
Assistant Professor,
ETC Dept.

_____   _____

(Head of Department)   (Principal)
Dr. H. G. Virani   Dr. R. B. Lohani
Professor, ETC Dept.   Goa College of Engineering

Place: Farmagudi, Ponda, Goa
Date:

Dedication

This thesis is dedicated to our parents, teachers, friends and other acquaintances, who have been there for us in the thick and thin of the implementation of this project.

Acknowledgement

Contents

List of Figures

List of Tables

Abstract

A biometric system can be defined as a pattern-recognition system that can recognize individuals based on the characteristics of their physiology or behaviour. One biometric technique for unintrusive identification is gait recognition which basically identifies people by the way they walk. In former work, gait recognition is mainly achieved with camera systems. In this study, we present an approach for gait recognition using Microsoft Kinect V2, a peripheral for the gaming console XBOX One, which provides us with marker less tracking of human motion in real time. We extract and evaluate a number of static and kinematic features and present the results of various classification algorithms for person identification.

Chapter 1

Introduction

1.1 Preamble

Compared to other biometric features such as iris and fingerprint detection. There are inherent advantages of gait such as it is perceivable at a distance, it need not be in contact and it is non-invasive. It works on itself without any user cooperation. Another important aspect is that it gives readings under low light conditions which is fairly accurate. Also disguising one's gait or imitating a person's walk is practically impossible. As a result, it has fascinated several security-sensitive environments such as classified research and nuclear labs, military, banks etc. Gait recognition is particularly useful in crime scenes where other biometric traits (such as face or fingerprint) might be obscured intentionally.

Two common categories of gait recognition are appearance-based and model-based approaches. Among the two, the appearance-based approaches suffer from changes in the appearance owing to the change of the viewing or walking direction. Model based ones are view and scale invariant and reflect in the kinematic characteristics of the walking manner. In this study we present a model based approach using the Microsoft Kinect sensor which offers us a 3D model of the human skeleton with capability to track upto 25 joints of the human body.

1.2 Motivation

The initial motivation to build a gait recognition system was developed after watching the Gait Analysis scene in the movie Mission Impossible – Rogue Nation (2015) where it was used for authentication for security purposes. Although at that time it felt as a futuristic technology, through research we realised that existing gait recognition approaches mostly use standard video cameras for capturing and recording the movement of walking persons. Here, the main difficulty lies in the extraction of characteristic features that can be used for identification. The challenges of existing gait recognition system and the possibilities Kinect offers lead to the assumption that the problem of gait recognition could be simplified using the Kinect sensor.

1.3 Proposed Idea

In this project we propose a skeleton model based approach (provided by Microsoft Kinect Sensor) for gait recognition and person identification. Our system consists of three components: The first component records the skeletal data offered by Kinect using the SDK provided by Microsoft. The second component processes on this data using MATLAB for feature extraction. Finally we use different classification algorithms available in MATLAB classification toolbox to identify a person using previously recorded training data and compare their performance and accuracy. This is a prototypic implementation of a gait recognition system, where we evaluate the possibilities of gait recognition using the Microsoft Kinect.
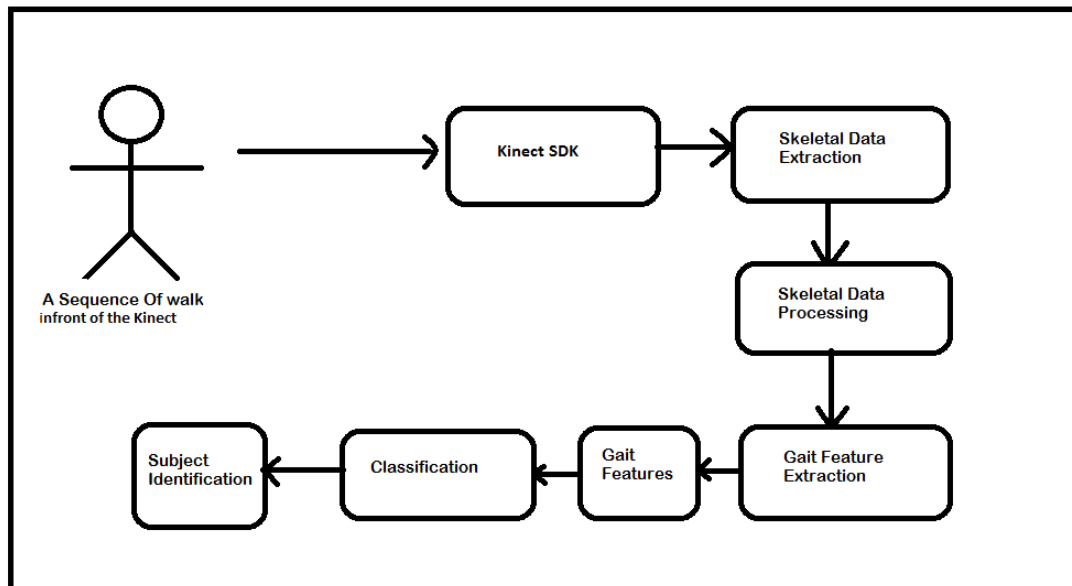
Figure 1.1: Process flow diagram

1.4 Problem Statement

Chapter 2

Literature Review

2.1 Full Body Gait Analysis with Kinect

1. Authors: Moshe Gabel, Ran Gilad-Bachrach, Erin Renshaw and Assaf Schuster

2. Abstract: Human gait is an important indicator of health, with applications ranging from diagnosis, monitoring, and rehabilitation. In practice, the use of gait analysis has been limited. Existing gait analysis systems are either expensive, intrusive, or require well-controlled environments such as a clinic or a laboratory. We present an accurate gait analysis system that is economical and non-intrusive. Our system is based on the Kinect sensor and thus can extract comprehensive gait information from all parts of the body. Beyond standard stride information, we also measure arm kinematics, demonstrating the wide range of parameters that can be extracted. We further improve over existing work by using information from the entire body to more accurately measure stride intervals. Our system requires no markers or battery-powered sensors, and instead relies on a single, inexpensive commodity 3D sensor with a large preexisting install base. We suggest that the proposed technique can be used for continuous gait tracking at home.

3. Hardware used : Microsoft's Kinect Xbox 360 console.

4. Methodology : Our technique uses a "virtual skeleton" produced by the Kinect sensor

and software. The skeleton information is converted into a large set of features which are fed to a model that predicts the values of interest. For example, inorder to measure stride duration, the model detects whether the foot is touching the ground. The outcome of this model is fed to a state machine that detects the current state from which the measurements are derived.

5. Conclusion In this work we have presented a novel method for full body gait analysis using the Kinect sensor. Using the virtual skeleton as the input to a learned model, we demonstrated accurate and robust measurements of a rich set of gait features. We showed that our method improves on prior art both in terms of having smaller bias and in having smaller variance. Moreover, our method can be extended to measuring other properties, including lower limb angular velocities and core posture. The sensor used is affordable and small, thus allowing installation in domestic environments. Since the sensor does not require maintenance, it allows for continuous and long term tracking of gait and its trends. These properties enable many applications for diagnosis, monitoring and adjustments of treatment. However measuring the utility of the methods presented here for medical applications is a subject for further research.

2.2 Motion Analysis using Kinect sensor

1. Authors: Richa D'Costa, Manpreet Kaur, Nishtha D. Wanchoo. Under the guidance of PROF. Milind Fernandes, Asst. Professor at Goa College of Engineering

2. Abstract: With the advent of Microsoft Kinect sensor, a flexible low cost tool has been made available that enables marker less tracking of human motion in real time. The study explores the possibility of utilizing Microsoft's Kinect sensor to analyse the biomechanics of the shot put throw. It presents a software prototype capable of capturing, recording, analyzing and comparing movement patterns using three-dimensional vector angles. The goal of the present work is to ease the analysis of a shot put game for overcoming the difficulty of visual error detection in shotput game using a software prototype which compares an amateur's game to that of a professional to yield results, which is implemented by using the Kinect sensor. It combines both the biomechanics analysis and Kinect motion capturing and develops a shot put game improvement solution with coaching evaluation.

3. Tools used : Microsoft Kinectv2 sensor, Visual studio 2013, Excel 2013

4. Methodology : Shotput is a game involving many complex motions simultaneously which includes rotational, translational and lateral motions. For any beginner, it gets very difficult to detect the stage of the throw which needs an improvement. Existing methods involve a coach trying to evaluate the stage of flaw in the game by observation. This has 2 major drawbacks: The presence of a skilled trainer is inevitable for every throw of the beginners practice session. And the accuracy level would be lower than desired, since the correction of flaw in the game is manual method. To overcome the above difficulties ,we have developed a software prototype for which could detect the flaw in the beginners game without constant physical presence for the busy trainer. Our prototype software takes care of mainly 3 important parameters in the game: 1. The time duration at each phase: It must be well within the range of our reference for maximum release velocity. 2. The following angles at the start of the initial phase: Right knee, Left Knee, Right hip, Right elbow If these angles are well within our reference angles, it results in higher build up energy to have an increased release velocity 3. The release angle of the player: When release angle is within the range of 38°-42°, maximum range is attained.

5. Conclusion : The Kinect system being a marker-less system, is able to capture the human motion with a reduction in time, whereas in a marker-based system attaching markers on the skin of the subject is a time consuming process (which can take 15 to 20 minutes). Although our software prototype could be used in a shot put game to provide coaching evaluation, the results cannot be completely relied upon, as the accuracy was limited due to decreased resolution of Kinect sensor. A few drawbacks of this sensing technology were the fixed location of the sensor with a range of capture of only roughly ten meters, a difficulty in fine movement capture, and shoulder joint biomechanical accuracy. The better reliable results refer to the hip flexion and knee angles and the results of hip adduction and ankle angles are not accurate enough to be relied on. This suggests that Kinect is better for capturing the rotation pattern of the joints with a large range of motion. In conclusion, Kinect system is a reliable system which permits to obtain acceptable kinematics results.

2.3 Gait Recognition with Kinect

1. Authors: Johannes Preis, Moritz Kessel, Martin Werner and Claudia Linnhoff-Popien from Ludwig Maximilians University, Munich, Germany

2. Abstract: The prominence of systems for automatic person identifcation has risen increasingly during the past years. One biometric technique for unintrusive identifcation is gait recognition which ofers the possibility to recognize and identify movement patterns of persons from some distance away. In former work, gait recognition is mainly achieved with camera systems. In this paper, we present an approach for gait recognition based on Microsoft Kinect, a peripheral for the gaming console XBOX 360, with an integrated depth sensor alowing for skeleton detection and tracking in realtime. We evaluate a number of body features together with steplength and speed, their relevance for person identifcation, and present the results of an empirical evaluation of our system, where we were able to accomplish a success rate of more than 90% with nine test persons.

3. Tools used : KinectV1,WEKA.

4. Methodology : We propose a model-based approach for gait recognition based on the skeleton provided by Mircosoft Kinect. As said before, Kinect provides a high quality skeletal model of up to two users in front of the Kinect sensor in a Cartesian coordinate system. We decided to use this skeletal data for recognition and did not use the depth and color images directly. Our system consists of three components: The first component records the skeletal information offered by Kinect which is then processed by the second component for feature extraction. Finally, we use the machine learning framework WEKA to identify a person on the basis of previously recorded training data.

5. Conclusion : In this paper, we presented a model based approach to gait recognition based on Microsoft Kinect. We use 13 biometric features such as the height, the length of limbs, and the steplength which are computed from the skeleton frames generated by Kinect. Based on testdata from 9 different persons, the three basic classifers Naive Bayes, 1R, and C4.5 were trained and evaluated concerning the success rate of their classifcation. Based on the features used of the decision tree C4.5, we found out that only four features, namely

height, length of legs, length of torso, and length of the left upper arm, were suffcient to correctly identify a person in 91% of all cases using the complete video from the specific experiment and the Naive Bayes classifer. Classification based solely on steplength and speed still yielded 55.2% success rate using either Naive Bayes or the decision tree.

Chapter 3

Introduction to Kinect Sensor

3.1 Kinect v2 Sensor Specifications

Microsoft Kinect Sensor was initially marketed to add motion control to games and is used as a peripheral for Xbox 360. Based on a webcam-style add-on peripheral, it enabled users to control and interact with their console/computer without the need for a game controller, through a natural user interface using gestures and spoken commands.
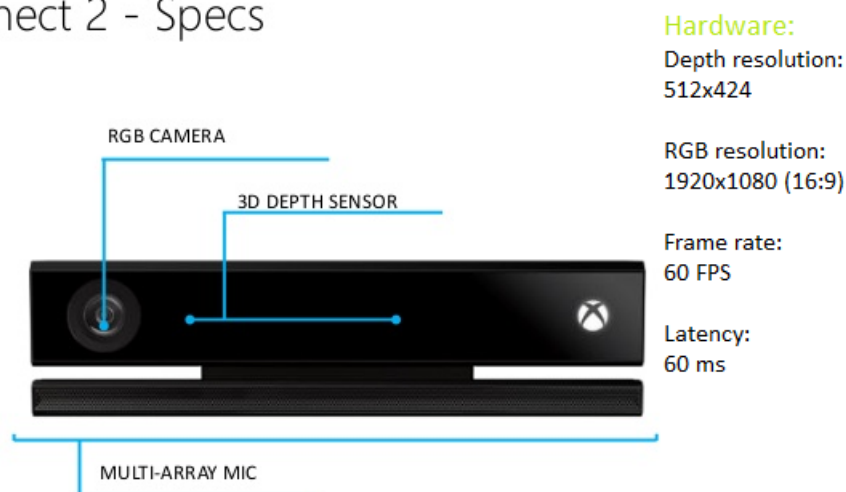
Figure 3.1: Microsoft Kinect V2 Sensor

The second version of the Kinect that is used in this study was released with Xbox One on November 22, 2013. This new hardware and software provided many improvements, including:

1. Improved body tracking - The enhanced fidelity of the depth camera, combined with improvements in the software, have led to a number of body tracking developments. The latest sensor tracks as many as six complete skeletons (compared to two with the V1), and 25 joints per person (compared to 20 with the V1). The tracked positions are more anatomically correct and stable and the range of tracking is broader.

2. Improved depth sensing - With higher depth fidelity and a significantly improved noise floor, the sensor gives you improved 3D visualization, improved ability to see smaller objects and all objects more clearly, and improves the stability of body tracking.

3. 1080p colour camera (30 Hz, 15 Hz in low light) - The colour camera captures full, beautiful 1080p videos that can be displayed in the same resolution as the viewing screen, allowing for a broad range of powerful scenarios. In addition to improving video communication and video analytic applications, this provides a stable input on which high quality, interactive applications are built.

4. New active infrared (IR) capabilities (512 x 424 30 Hz) - In addition to allowing the sensor to see in the dark, the new IR capabilities produce a lighting-independent view—and you can now use IR and colour at the same time.

5. The Kinect V2 has a range of 0.5 to 4.5m

	Version 1	Version 2
Depth Range	0.4m to 4m	0.5m to 4.5m
Colour Stream	640x480 @30fps	1920x1080 @30fps
Depth Stream	320x240	512x424
Infrared Stream	None	512x424
Type Of Light	Light Coding	ToF
Audio Stream	4-mic array 16khz	4-mic array 48khz
USB	2.0	3.0
Bodies Tracked	2(+4)	6
Joints	20	25
Hand Tracking	External Tools	Yes
Face Tracking	Yes	Yes+Expressions
FOV	57°H 43°V	70°H 60°V
Tilt	Motorised	Manual

Table 3.1: Differences between Kinect v1 and v2

3.2   Skeleton Tracking

By using the depth stream, the Kinect SDK is able to detect the presence of the person in front of the sensor. Kinect is capable of simultaneously tracking upto six people, including two active individuals for gait analysis with a feature extraction of 25 joints per person. The number of people the device can "see" (but not process as individuals) is only limited by how many will fit in the field-of-view of the camera. For each tracked person, Kinect SDK API provides a "skeleton" as a set of motion data. A skeleton contains 25 position sets, one for each "joint" of the body.
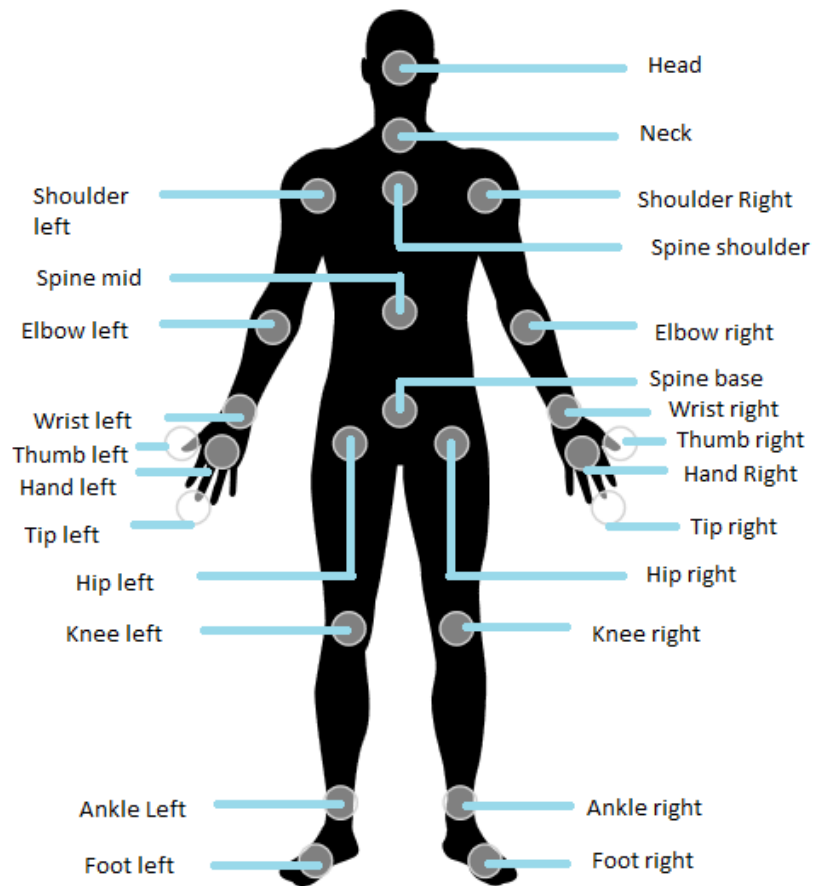
Figure 3.2: Joints Represented by Skeleton Tracking

3.3 Measuring Distances Using Kinect

Kinect integrates an infrared sensor, along with a depth processor. The visible area is called "field of view". The depth processor produces depth frames. Each depth frame is a grid of points. The Kinect SDK is then feeding the depth frames to a powerful body-detection algorithm. The algorithm identifies 25 human body joints and calculates their coordinates in the 3D space. Every single joint has 3 values: X, Y, and Z. It is projected in a Cartesian coordinate system. The point is the position of the sensor. Every other point is measured in terms of the position of the sensor. Check the graph below,It is viewing the sensor and the scene from above.

- X is the position in the horizontal axis.

- Y is the position in the vertical axis.

- Z is the position in the depth axis.

Figure 3.3: Measurement of distances using Kinect

As seen from the figure, Z value is not the linear distance between the point and the sensor. Instead, it's the distance between the point and the plane of the sensor

Chapter 4

Study of Classification Algorithms

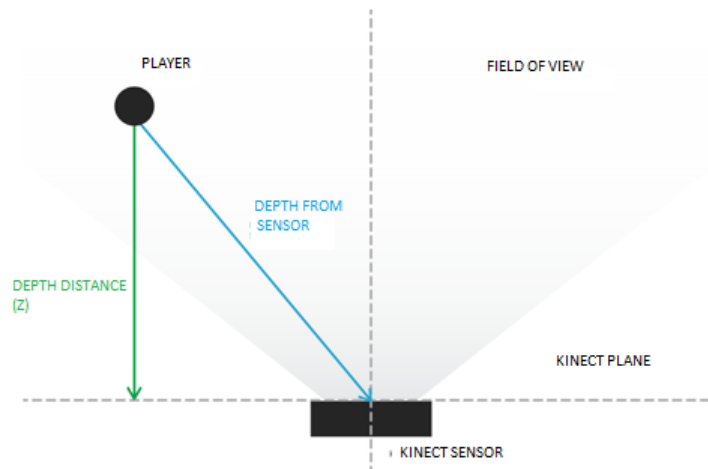Classification is a branch of supervised learning whose aim is to identify which set of categories a new observation belongs, on the basis of a set of training data which contains observations whose category membership is known. We would briefly like to discuss the classification algorithms we have used in our study.

4.1 Support Vector Machine (SVM)

SVMs are supervised learning algorithms used for classification and regression analysis. A support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which is then used to separate out and classify the data. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class, since in general the larger the margin, the lower the generalization error of the classifier. SVM is able to discriminate data that is not linearly separable by using the kernel trick. The trick stipulates the original finite-dimensional space to be mapped into a much higher-dimensional space, presumably making the separation easier in that space. This is done by selecting a kernel function to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant.
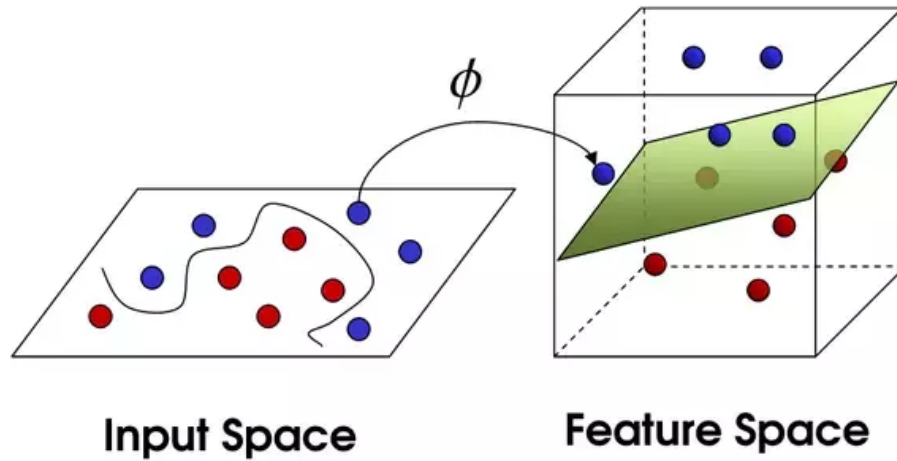
Figure 4.1: Kernel trick representation

4.1.1 Pros and Cons associated with SVM

If we compare, there are quiet a few positives than the negatives of SVM like it is very effective in high dimensional spaces. Even where the number of dimensions is greater than the number of samples it yet has its effect. It uses a subset of training points in the decision function (called support vectors) that makes the memory more efficient. The algorithm is versatile where different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

There are very also a few negatives, some of them are, it does not perform well when we have a large data set because the required training time is higher. When the data set is noisy the performance is not good enough, that is the target classes are overlapping. And lastly SVM does not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

4.2 k-Nearest Neighbors algorithm (k-NN)

The k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. The closest neighbor (NN) rule distinguishes the classification of unknown data point on the basis of its closest neighbor whose class is already known. The value 'k' indicates how many nearest neighbors are to be considered to characterize class of a sample data point. It makes utilization of more than one closest neighbor to determine the class in which the given data point belongs to and consequently it is called as KNN. If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
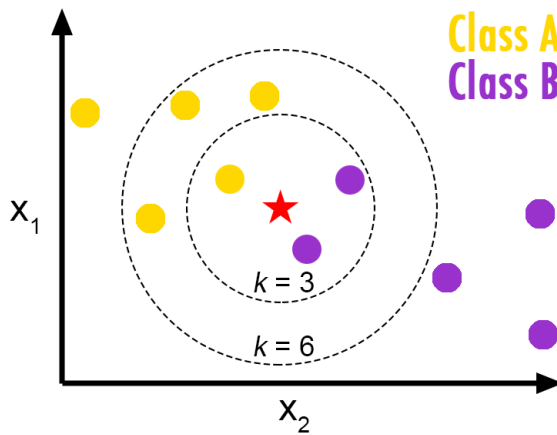
Figure 4.2: Nearest Neighbours for different values of k

4.2.1 Pros and Cons associated with k-NN

This algorithm is simpler to implement. k-NN employs lazy learning and generalizes it during testing, which makes it a real time algorithm. It also handles multi-class cases naturally and can do well in practice with enough representative data.

On the other hand when the data sets are large, lazy learning requires that, most of k-NN's computation be done during testing, rather than during training which increases the computational time. Searching for the nearest neighbour is a time consuming task. The set back over here is inputs can commonly be "close" to many data points. This reduces the effectiveness of k-NN, since the algorithm relies on a correlation between closeness and similarity.

4.3 Decision tree learning

Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. To predict a response, follow the decisions in the tree from the root (beginning) node down to a leaf node. The leaf node contains the response.
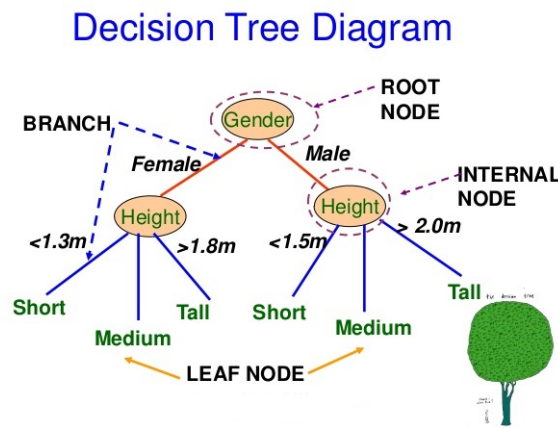
Figure 4.3: Decision tree diagram

MATLAB provides 3 types of decision trees depending on the maximum number of splits possible.

Classifier Type	Prediction Speed	Memory Usage	Interpretability	Model Flexibility
Simple Tree	Fast	Small	Easy	Low. Few leaves to make coarse distinctions between classes (Maximum number of splits is 4).
Medium Tree	Fast	Small	Easy	Medium Medium number of leaves for finer distinctions between classes (Maximum number of splits is 20).
Complex Tree	Fast	Small	Easy	High Many leaves to make many fine distinctions between classes (Maximum number of splits is 100).

Figure 4.4: Types of Decision Trees in MATLAB

4.3.1 Pros and Cons associated with Decision tree

They are very fast to build and test. Another strong behavior is that they work on highly non-linear data, easy to understand, visualise and require minimum assumption. Even the nonlinear relationships between parameters do not affect tree performance.

Whereas the weekness here is, even a small change in input data can at times, cause large changes in the tree. Changing variables, excluding duplication information, or altering the sequence midway can lead to major change and might possibly require redrawing the tree. Decision Trees do not work well if you have smooth boundaries, that is they work best when you have discontinuous piece wise constant model. If you truly have a linear target function decision trees are not the best. They do not work best if you have a lot of un-correlated variables and a tree with many levels and branches can lead to an over optimized result, or many false positives due to multiple comparison.

Chapter 5

Implementation and Methodology

5.1 Tools Used

5.1.1 Microsoft Kinect v2 sensor

Specifications of the Kinect are discussed in Chapter 3.

5.1.2 Visual Studio 2015 Community Edition

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code. Visual Studio supports different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++and C++/CLI (via VisualC++), VB.NET (via VisualBasic.NET), C# (via Visual C#), and F# (as of Visual Studio 2010). Support for other languages such as Python, Ruby, Node.js, and M among others is available via language services installed separately.

Microsoft provides the Kinect SDK and Kinect specific libraries to be used on Visual Studio. This is the reason we have chosen Visual Studio as our development platform as it makes interfacing the Kinect easy.

5.1.3 Vitruvius

Vitruvius is an advanced kinect framework which provides a set of easy to use kinect utilities on Visual Studio and unity that simplify and speed up many aspects of Kinect for Windows App development.The product also alleviates alot of the heavy math associated with implementing motion tracking in advanced kinect apps.

Vitruvius lets you easily calculate joint angles and find the length of specified segments in 3D space, and the rotation of the body in the X, Y, Z axis. Vitruvius also provides simplified bitmap manipulation and lets you take full advantage of the latest Kinect for Windows facial tracking and gesture detection for Windows human-computing interactions.

We used the Vitruvius Library on top of the SDK to simplify the maesuremnt of length and angles.

5.1.4 Notepad++

Notepad++ is a text editor and source code editor for use with Microsoft Windows. It supports tabbed editing, which allows working with multiple open files in a single window. Notepad++ is distributed as free software. It supports syntax highlighting and code folding for over 50 programming, scripting, and markup languages.

The skeletal data obtained from the SDK is stored in CSV format in a Notepad++.

5.1.5 MATLAB R2015a

MATLAB is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

MATLAB provides special toolboxes for applications such as Signal Processing and Communications, Control System Design and Analysis, Pattern Recognition, Classification Learner and other statistical modelling. The CSV format file is processed in MATLAB for gait feature extraction and classification.

5.2 Gait Features

The Kinect SDK offers the detection and tracking of 25 different skeletal joints. The features that can be generated from these joints can broadly be classified as static and kinematic features. The static features are the ones which do not change in time as a person is walking, for e.g. height. We define a set of 7 static features used in our study.

SR No.	Static Features
1	Height
2	Length of Full Arms
3	Length of Fore Arms
4	Length of Upper Arms
5	Length of Full Legs
6	Length of Thighs
7	Length of Lower Legs

Table 5.1: The set of static features

Kinematic features basically involve distance and angle between joints while a person is walking. The value of these features constantly change during the walk and are shown to be periodic in nature. We define a set of 8 kinematic features involving distance between joints.

SR No.	Features
1	Distance between ankles
2	Distance between knees
3	Distance between elbows
4	Distance between hands
5	variance of head (along x)
6	variance of head (along y)
7	variance of left knee (along y)
8	variance of right knee (along y)

Table 5.2: kinematic features involving distance between joints

When we talk about the features such as variance of head and knees along X and Y. We are trying to map the vertical and lateral oscillations of these skeleton joints. Features involving angles are extracted during the walk but are not used in this study and will be discussed later. Here we would like to divide our implementations into two parts

• Part I : This will deal with the experimental setup, design of the application using the Kinect SDK, logic behind measurements of static and kinematic features and their extraction. The tasks discussed in Part I are implemented on Visual Studio.

• Part II : This involves the overhead processing on the gait features, classification and comparison on the performance of different classification algorithms. These tasks are implemented on MATLAB.

5.3   Experimental Set up

The Kinect was placed at height of 1.5m and the person was made to walk in front of the Kinect. The figure5.1 gives us a nice visual representation of the setup.
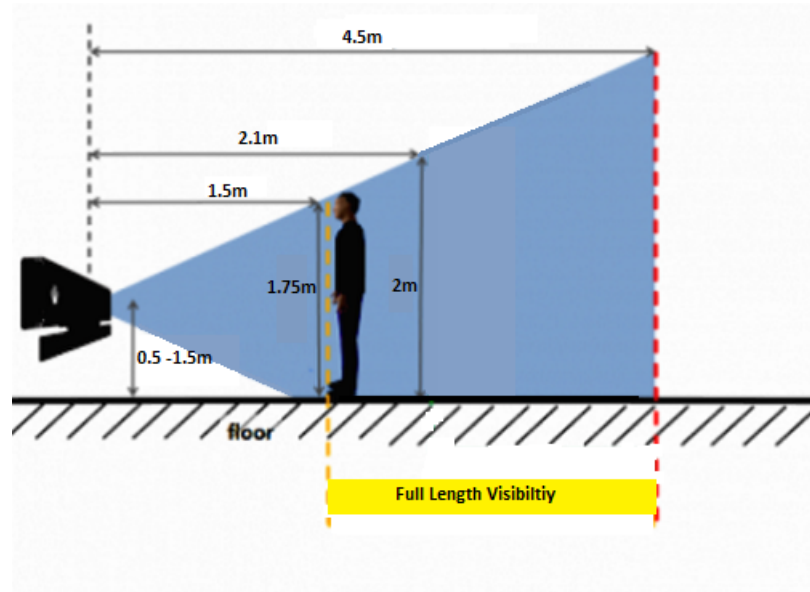

Figure 5.1: Experimental Setup

Initially, the person is made to stand in a straight posture at the maximum range of Kinect (close to 4.5m). Here the static feature readings are taken for a period of about 3 seconds.

Then the person is asked to walk straight towards the Kinect. We are interested in the kinematic features of the walk upto a distance of 1.5m from the Kinect, because within this range we are able to capture the motion of the entire body. Beyond this,some part of the joints always goes out of the frame.

Although the range of 3m (4.5-1.5 m) appears small. It is more than adequate to extract one complete gait cycle. The representation of this gait cycle is shown in Figure 5.1.

5.4   Logic behind the measurement of static and kinematic features

5.4.1   Measurement of length between joints

We refer back to the figure depicting all the body joints represented by the Kinect.

For explanatory purposes we will show how height is measured. In principle same logic is
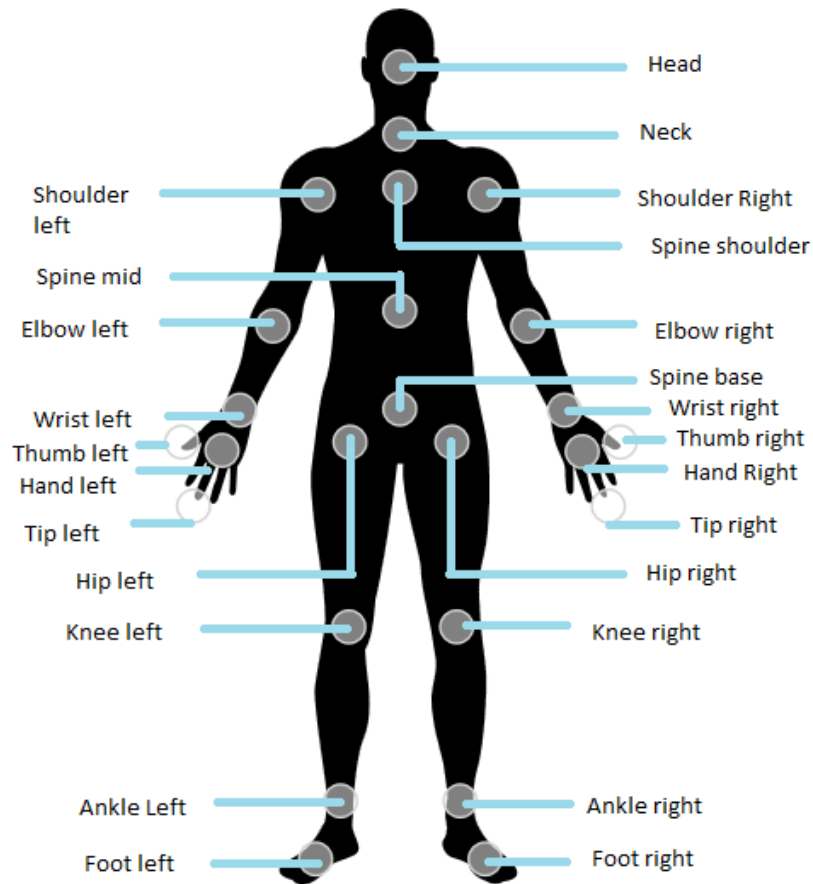

Figure 5.2: 25 joints offered by Kinect

applied to get other measurements and distances. Kinect provides you with the coordinates (X, Y and Z) of 25 skeleton joints. We might think that a person's height is the distance from the head joint to a foot joint, right? This is fundamentally incorrect because users might stand with a bended knee, they might lean a bit to the left or to the right. If you try to calculate the distance defined from the head joint to one of the foot joints, you'll get a far from accurate result.

If we examine the skeleton joints as seen in figure 5.2 we'll notice that the height is the sum of the lengths of the following line segments:

- Head – Neck

- Neck – Spine Shoulder

- Spine Shoulder – Spine Mid

- Spine Mid – Spine Base

- Hip Left/Hip Right – Knee Left / Knee Right

- Knee Left / Knee Right – Ankle Left / Ankle Right

- Ankle Left / Ankle Right – Foot Left / Foot Right

Using (X,Y,Z) coordinates of each joint we can calculate the length of the line defined by two joints in the 3D space, simply by using the distance formula:

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Pretty straightforward till here. But, how will we determine which leg corresponds to the most accurate user height? Hence we take the average of both the legs and then add this to the sum of the rest of the joints. It should be noted that the height calculated in this case does not represent the true height because Kinect only gives you the centre of the head joint. It does not detect the end of the head for accurate measurements. Also height measured in this study excluded the foot left and right joints, as they being close to the floor gave readings prone to error.

Hence for all intent and purposes, whenever we talk about height in this study it will be the sum of the lengths of the line segments:

Head -> Neck -> Spine Shoulder -> Spine Mid -> Spine Base -> HipLeft/HipRight -> KneeLeft/KneeRight -> AnkleLeft/AnkleRight
Where distances from the hip joint to the ankle joint gets averaged for the two legs.

5.5   Measurement of angles between joints

We present a new gait feature that provides us relative motion between two joints by computing the joint relative angles (JRA) over a complete gait cycle. JRA between two joints p1 and p2 can be defined as the angle formed by p1 and p2 with respect to a reference point r. Given the coordinates of 3 points p1, p2, and r in a 3-D space the angle p1, p2 formed by p1 –> r –> p2 can be calculated as

$$\theta_{p_1,p_2} = \cos^{-1} \frac{p_1 r . r p_2}{\|p_1 r\| . \|r p_2\|}$$

The SPINE_ BASE joint was selected as the reference point, since it remains almost stationary during walking. For the 25 skeletal joints, there is a total of 300 possible joint-pair combinations, which is a high-dimensional feature space. In our study we left out JRA involving 5 joints: Hand Left/Right, Tip Left/Right and SpineBase. The position of the 4 former joints are usually irrelevant to gait and the latter if included would have given us just 2 joints(a line) instead of 3,which is required for an angle. So we are left with 20 joints, which gives us a total of 190 possible combinations.

In addition, out of these 190 not all joint-pair is relevant in gait feature representation. For example, JRAs between the SpineShoulder and the SpineMid joints does not represent any information related to human gait, since both these joints remain almost stationary when a person walks. Therefore, identifying the skeletal joint pairs that are relevant to human gait motion is imperative. The following figure gives an example of a JRA.
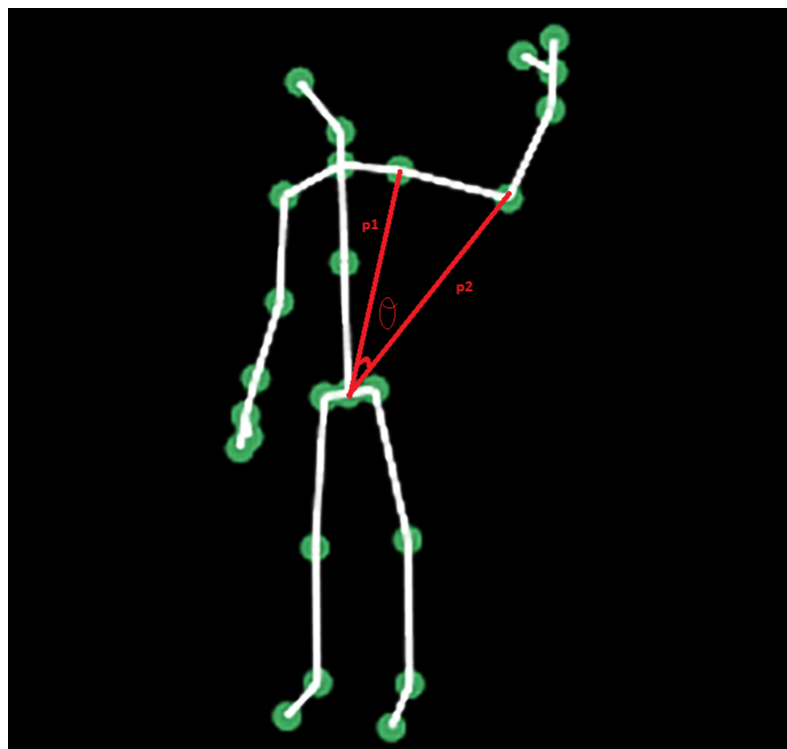
Figure 5.3: JRA between Shoulder Left and Elbow Left

5.6   Design of the App

We designed a windows application on Visual Studio using the Kinect SDK for the purpose of extracting skeletal data and gait features which were discussed in the preceding sections. We named our App "TRACE".

Here is the front page of TRACE. It has only one circular button. When pressed leads to the next page where the actual extraction happens.
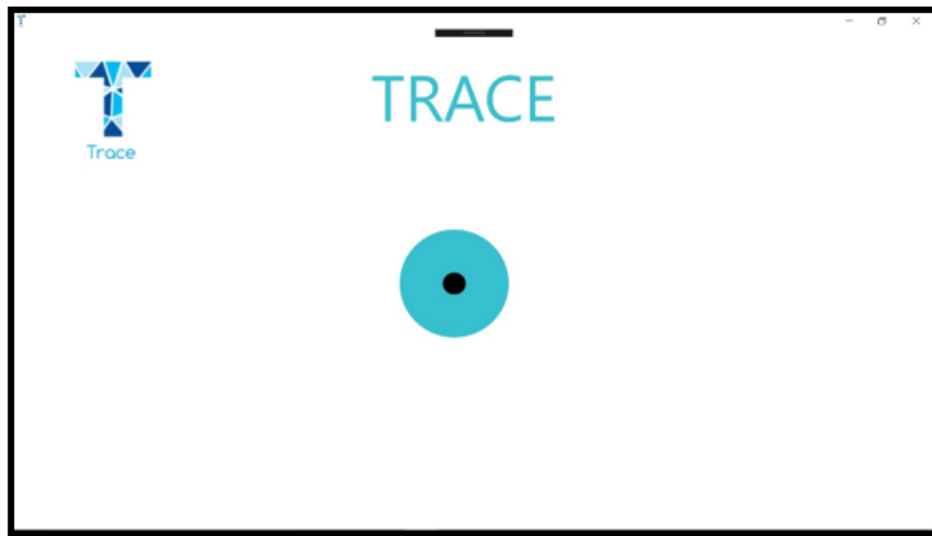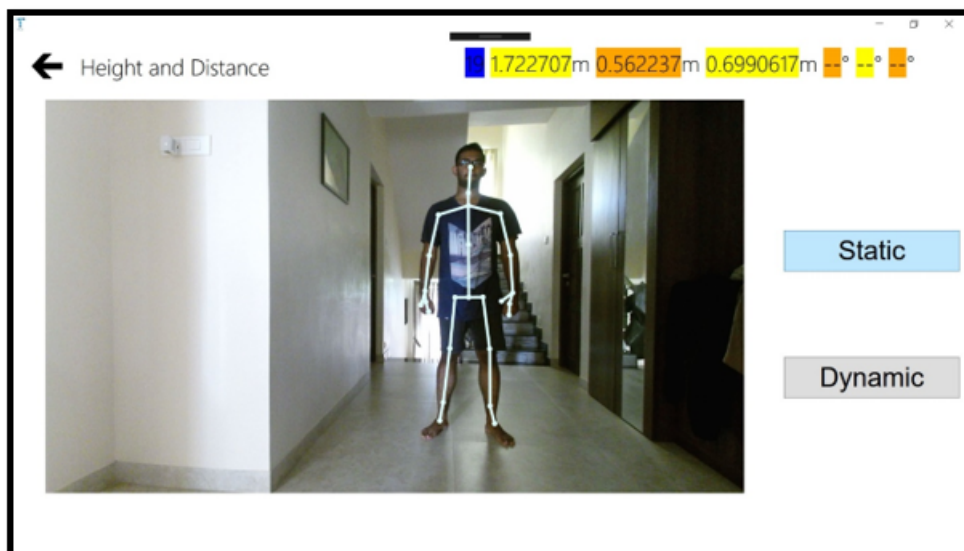

Figure 5.4: Trace

Figure 5.5: Skeletal View

This Page has two buttons on the right for capturing the static and kinematic/dynamic data respectively. Data displayed on the top right is basically for the operator to know that the process is working fine. The number in blue indicates the joints currently being tracked by the Kinect.

The next 3 are distance measurement between various joints. The final 3 are used for displaying angles, currently null since only the static button has been pressed.

The two figures below show how TRACE writes the static and dynamic data in CSV format. Each new line represents each frame of the walking sequence captured.
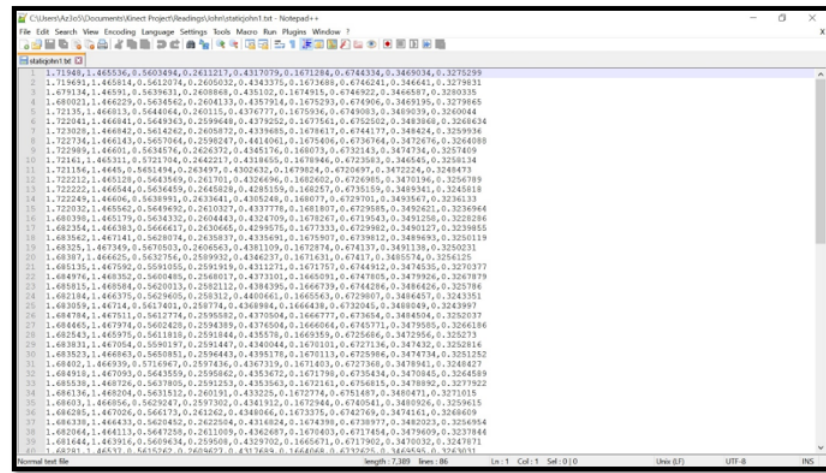

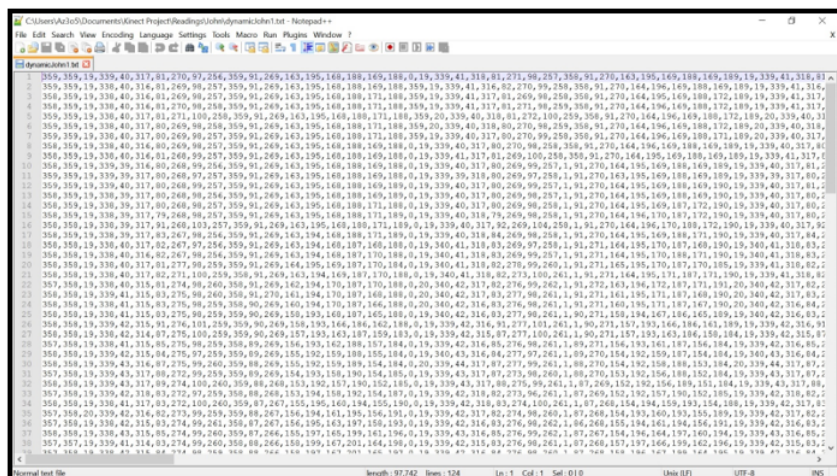Figure 5.6: Static data in CSV format

Figure 5.7: Dynamic data in CSV format

5.7 Gait Database

Since a standard gait dataset acquired with the Kinect sensor is not available, we have collected a set of gait samples. The gait database consisted of 8 participants (5 males and 3 females) who were asked to walk naturally in front of the Kinect, each participant was asked to walk 10 times giving us a total of 80 gait samples. Out of this, 8 gait samples of each participant were used in the training set giving us a total of 64 samples for training. Remaining 2 gait samples was used for testing, giving us 16 testing samples.

Apart from that, we also collected a single gait sample from 12 other different participants. These samples were then added to training set to degrade it. This was done to prevent the classifiers from overgeneralizing the data and checking their performance in the presence of outliners. Hence the final training set consisted of 76 gait samples.

5.8 Overhead Processing

All the static feature data that was obtained for the 3 second interval was averaged out by calculating the mean for each feature. Since we are interested in tracking the changes of the kinematic data, the same procedure cannot be used. Also Kinect having limited precision and frame rate is prone to noisy acquisitions, to smoothen this out we pass all the kinematic data through a moving average filter. Through trial and error we found 1/4 to be the best mask for this data. The figure below demonstrates the smoothening effect of the moving average filter.
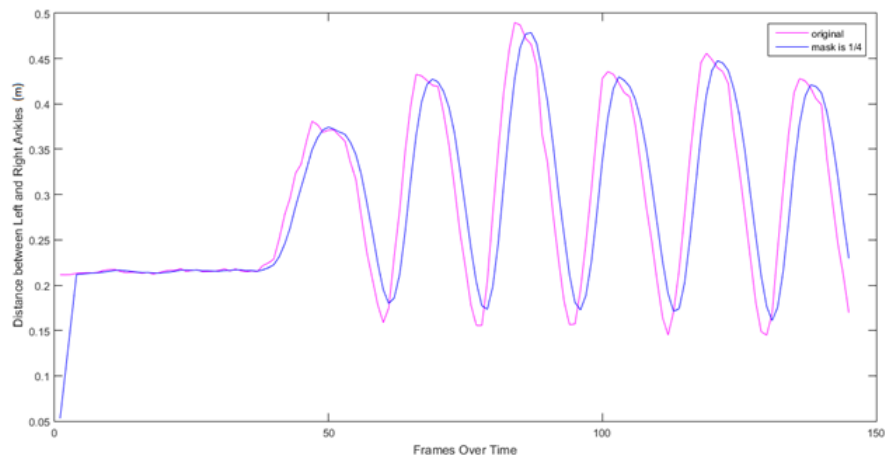

Figure 5.8: Effect of the Moving Average Filter Pink represents the original data Blue displays the filtered output Notice the smooth peaks and troughs in the filtered output

5.9   Gait Cycle Detection

Regular human walking is considered to be a cyclic motion, which repeats in a relatively stable frequency. Hence our first task is to extract one complete gait cycle, as it represents the complete gait signature. The figure below is a nice pictorial representation of a gait cycle.
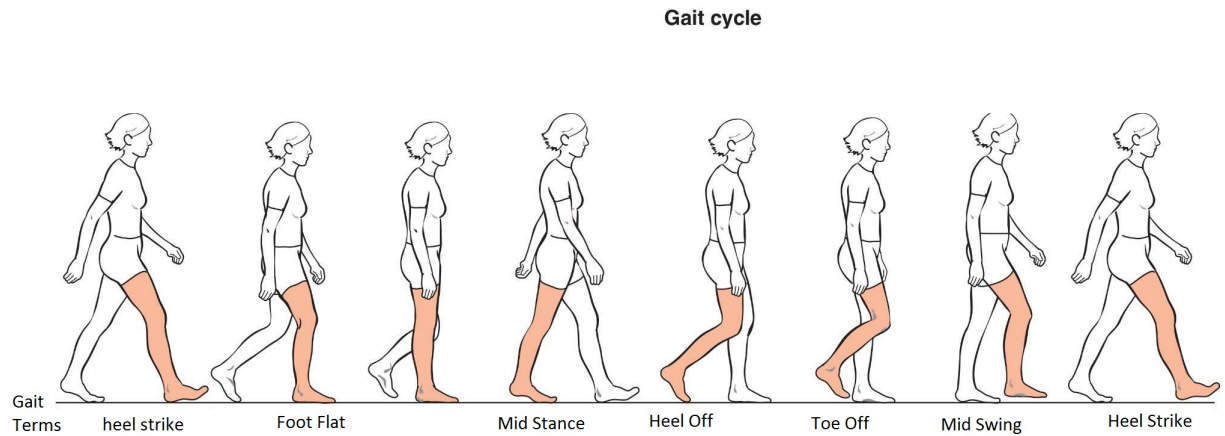

Figure 5.9: Phases Of the Gait cycle

In order to identify gait cycles, the distance between the Ankle Left and Ankle Right joints was tracked over time. A moving average filter was used to smoothen this distance. During walking, the distance between the two ankle joints will be at maximum when the right and the left leg are farthest apart and will be at minimum when the legs are in the rest (standing) position. Therefore, by detecting three subsequent minima, it is possible to find the occurrences of the two legs in the rest position, which corresponds to the beginning, middle, and ending points of a complete gait cycle, respectively.

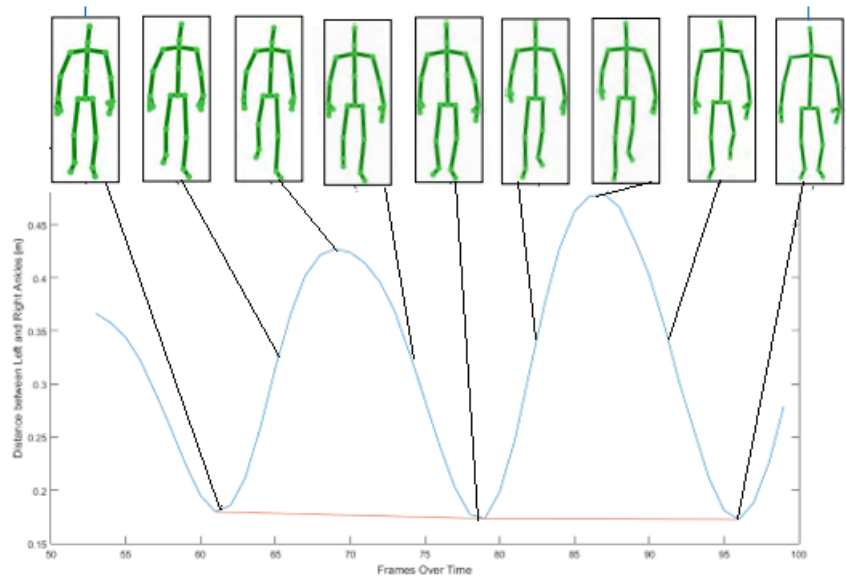The following figure explains this concept.

Figure 5.10: Detection of a complete gait cycle by tracking the distance between the left and right ankle joints. The Red line connects the 3 subsequent minima

The JRA data for this one gait cycle was extracted and considered. The rest of the kinematic features were averaged out for this period and the variances were calculated on the means of this cycle.

5.10 Observation and Results

The table below shows the 15 static and dynamic features used in our study for the purpose of gait recognition. We used a set of 5 classifiers: Linear SVM, Quadratic SVM, Fine KNN, Complex Trees and Neural Networks and we evaluate the performance of each classifier on the testing data in this section.

SR No.	Features
1	Height
2	Length of full arms
3	Length of fore arms
4	Length of upper arms
5	Length of full legs
6	Length of thighs
7	Length of lower legs
8	Distance between ankles
9	Distance between knees
10	Distance between elbows
11	Distance between hands
12	Variance of head along X
13	Variance of head along Y
14	Variance of left knee along Y
15	Variance of right knee along Y

Table 5.3: kinematic features involving distance between joints

Before applying any classifier we normalized our feature matrix to the range [-1, 1]. Scaling data is very important in order to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. We also conducted a 3-fold cross-validation for the first four classifiers in order to evaluate the effectiveness of the proposed method. In a 3-fold cross-validation, the whole dataset is randomly divided into 3 subsets, where each subset contains an equal number of samples from each category. The classifier is trained on 2 subsets, while the remaining one is used for testing. The average classification rate is calculated after repeating the above process for 3 times.

In the case of Linear SVM, 14 gait samples of the testing set were classified correctly. Giving us an accuracy of 87.5%. For the 2 samples which weren't, one was misclassified and the other was not recognised by the classifier.

Quadratic SVM gave us the best accuracy of 93.75% by categorizing 15 gait samples correctly. Surprising, the sample which was misclassified in Linear SVM was not recognised in this case.

Fine KNN yielded an accuracy of 81.25%, 3 samples from the testing set were misclassified. On the other hand Complex Trees gave us the poorest success rate of 68.75% were 2 samples were misclassified and 3 were not recognised by the classifier.

In the case of neural networks, we set aside 10 % data for validation and 5 % for testing from the training database. Validation samples are used to measure network generalization, and to halt training when generalization stops improving, whereas testing samples have no effect on training and so provide an independent measure of network performance during and after training. The figure below shows the architecture of the network we implemented. It consists of 15 neurons in the input layer, one hidden layer with 12 neurons and the output layer with 9 neurons. The output layer has one neuron extra as compared to the number of classes, this is for the data that does not get recognised in the network.
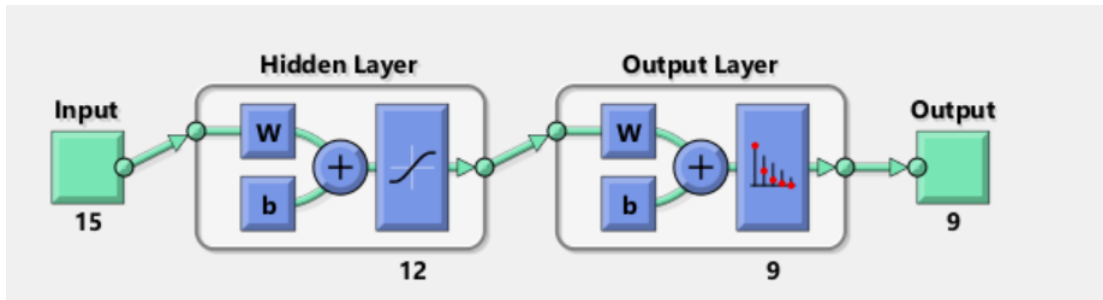

Figure 5.11: ANN Architecture

When the output threshold was set at 0.9, neural network correctly identified 13 of the 16 samples in the testing database, accuracy of 81.25 %. When this threshold was lowered to 0.8, the accuracy jumped to 93.75 %, where the network not being able to identify just one sample.

Chapter 6

Conclusion

This study presented a new Kinect-based gait recognition method that utilizes the 3D skeleton model in order to compute a robust representation of gait. It enables us to extract more complicated gait features from human walking sequences compared to conventional devices such as video cameras or wearable sensors. Although the set of 8 participants is statistically small, the maximum accuracy of 93.75% obtained is promising and shows that reliable discrimination of individuals in an indoor ambiance is possible by this method. The following table summarizes the success rates obtained with different classification models.

Classifier	Accuracy
Linear SVM	87.5 %
Quadratic SVM	93.75%
Fine kNN	81.24%
Complex Tree	68.75%
ANN	93.75%

Table 6.1: Different classifiers and their accuracy

6.1 Future Work

We are planning an extensive gait acquisition campaign to improve the validation of the classification results from a statistical viewpoint. The JRA (joint relative angle) data that we captured during a participants walk is not being used in this study. To provide a visual representation of the relevance of a particular JRA sequence in gait representation we constructed a heat map, where a high value corresponds to high relevance and low value corresponds to a low relevance. The heat map is symmetric on both side of the diagonal, since the JRA values between joints J1, J2 and J2, J1 are same.
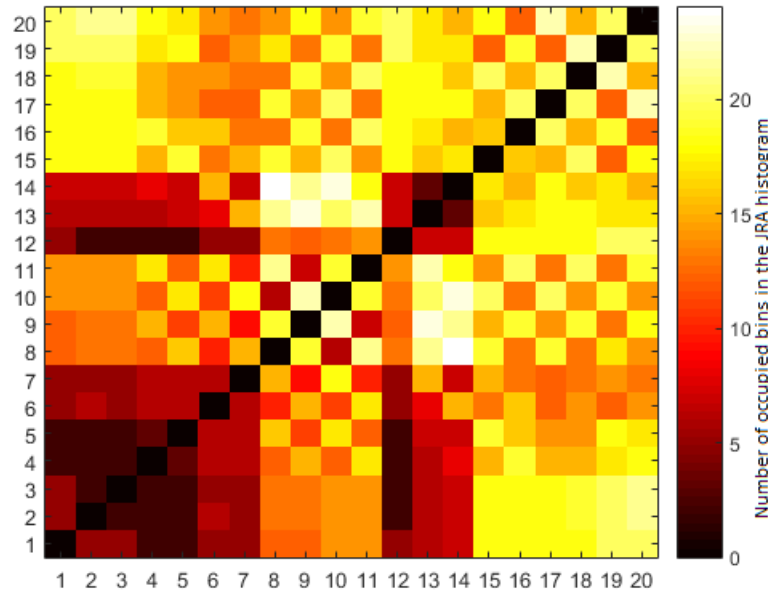

Figure 6.1: Heat map of the 20x20 average matrix obtained for the average number of bins occupied for different JRA

1 Head	6 Elbow left	11 Hand right	16 Knee right
2 Neck	7 Elbow right	12 Spine mid	17 Ankle left
3 Spine shoulder	8 Wrist left	13 Hip left	18 Ankle right
4 Shoulder left	9 Wrist right	14 Hip right	19 Foot left
5 Shoulde right	10 Hand left	15 Knee left	20 Foot right

Table 6.2: Body Joints

We need to find a classifier or kernel that will take the collection of the most relevant JRA sequences from both the train and test samples as parameters and computes a similarity measure. This kernel will have to be robust against variable walking speed. We could also

investigate the possibility of using more than one Kinect sensor to track gait. From the applicative point of view we will consider other scenarios that go beyond classification, e.g. security and health/well-being contexts.

Bibliography

[1] Kenneth J. W. Taylor, Peter N. Burns, Peter N. T. Wells. Clinical Applications of Doppler Ultrasound Second Edition, pages 1-3.

[2] http://www.genesis-ultrasound.com/history-of-ultrasound.html

[3] https://wiki.engr.illinois.edu/display/BIOE414/History+of+Color+Doppler+Ultrasound

[4] William R. Hendee and E. Russell Ritenour. Medical Imaging Physics, Fourth Edition, pages 344 - 346, 2002.

second page of bibliography