Gait Recognition

Mark Ruane Dawson

Final Report

Supervisor: Professor Guang-Zhong Yang

MEng Computing 4
Department of Computing
Imperial College of Science, Technology & Medicine
London, SW7 2BZ

June 2002

Abstract

Gait *- A particular way or manner of moving on foot.*

Gait recognition is the process of identifying an individual by the manor in which they walk. This is a marker less unobtrusive biometric, which offers the possibility to identify people at a distance, without any interaction or co-operation from the subject, this is the property which makes it so attractive as a method of identification.

This project aims to develop a system capable of semi-automatic gait recognition. A persons gait signature is created using a model based approach. Temporal and spatial metrics extracted from the modal, such as variation in angles of the limb or the amplitude of a persons walking pattern can all be used to create a *"gait signature"* of the individual which are transformed in eigenspace using Principle Component Analysis and can be used to identify the subject in subsequent video sequences.

Acknowledgement

I would like to thank Professor Yang, for the initial project proposal and accepting to supervise my project. He helped in a broad range of issues from giving me direction, helping to find solutions to problems, outlining requirements and always having the time to see me, even when I appeared unannounced asking questions.

I would also like to thank Benny Lo, one of Professor Yang's PhD students, who helped me solve a number of problems during the length of my project, regarding image processing and also for his expert knowledge of Visual C++.

Table of contents

1. Introduction 8
1.1 Biometrics 8
1.2 Why Gait? 8
1.3 Project motivation 10
1.4 Gait Recognition Scenario 10
1.5 Project scope 10
1.6 Report structure 11
2. Background 13
2.1 Overview 13
2.2 Model Free 13
2.3 Model based 14
2.4 Principle Component Analysis (PCA) 16
2.4.1 Effect of noise 18
2.5 Canonical Analysis (CA) 18
2.6 Image segmentation 20
2.6.1 Aim 20
2.6.2 Approaches 20
2.6.3 Morphological Filters 21
2.6.3.1 Connected Component Labelling 21
2.6.3.2 Erosion 22
2.6.3.3 Dilation 23
2.7 Self similarity Plots 23
2.8 Relational Modelling 24
2.9 Pattern Classification 25
2.9.1 K-Nearest Neighbour classifier 26
3. Conceptual Design & 27
3.1 Aim 27
3.2 Overall Program Architecture 27
3.2.1 Inputs 28
3.2.2 Processes: 28
3.2.3 Output: 28
3.3 Gait Recognition Design 29
3.3.1 Gait Signature Creation 29
3.3.2 Feature Extraction 29
3.4 Development Environment 29
3.5 Design Methodology 30
4. Utility Classes 31
4.1 Overview 31
4.2 CVector class 31
4.3 CMatrix class 31
4.3.1 Aim 31
4.3.2 Design 31
4.4 CHuman class 32
4.4.1 Aim 32
4.4.2 Design 32
4.4.3 Implementation 33
4.5 CHumanSequence 34
4.5.1 Aim 34

4.5.2 Design 34
5. Image segmentation 37
5.1 Aim..37
5.2 Assumptions & Limitations 37
5.3 Segmentation Software Design..37
5.4 Background Generation 38
5.5 Foreground Segmentation..38
5.6 Gait Data Set..40
6. Semi Automatic Model Fitting 43
6.1 Aim..43
6.2 Assumptions & Limitations 43
6.3 Human Model Design..43
6.3.1 Human Model Representation 43
6.3.2 Human Relational Model 44
6.4 Search Space Estimation..45
6.5 Body Dimension Estimation 45
6.6 Head Tracking..45
6.7 Torso..46
6.8 Head Body Location..46
6.9 Legs 47
6.9.1 Initial Pose Location 47
6.9.2 Leg Position Tracking 48
7. Gait Signature Extraction..49
7.1 Aim..49
7.2 Signature Design & Implementation..49
7.2.1 Joint Position Signature..49
7.2.2 Limb Angle / Angular Velocity Signature..50
7.2.3 Self Similarity Plots..50
7.2.3.1 Data Normalisation 51
7.2.3.2 Creating Self Similarity Plot 51
7.2.3.3 Creating the feature vector representation.. 51
8. Gait Signature Recognition..53
8.1 Aim..53
8.2 Design..53
8.3 Gait Signature Feature Spaces 53
8.3.1 Eigenspace Transformation..53
8.4 Training Data Set 54
8.5 Recognition through classification..54
8.5.1 Spatio Temporal Correlation..54
8.5.2 K-Nearest Neighbour..55
9. 3D Graphics Engine..57
9.1 Aim..57
9.2 Design..57
9.3 Projection Theory..58
9.3.1 Rotation matrices..58
9.4 Implementation..58
9.4.1 Display Process..58
9.4.2 Data Normalisation 59
9.4.3 Object Rotation..59
10. User Application 61

10.1 Aim 61
10.2 Main Interface 61
10.2.1 Gait Toolbar 61
10.2.2 Video display & manual capture area 62
10.2.3 Manual click guide 62
10.2.4 Data visualisation 62
10.3 Gait application operations 63
10.3.1 Loading a bitmap image 63
10.3.2 Gait PCA Visualisation 63
10.3.3 Performing recognition 63
10.3.4 Saving captured gait data 64
11. Evaluation 65
11.1 Segmentation Results 65
11.1.1 Foreground Extraction 65
11.1.2 Background Generation 67
11.3 Gait Signature Extraction 68
11.3.2 Limb angles 70
11.3.3 Limb angular velocities 70
11.4 Recognition Rate 71
12.1 Achievements 72
12.2 Limitations 72
12.3 Future work 73
Appendix A – References 74

Project code and documents can be obtained from the following link:
http://www.doc.ic.ac.uk/~mrd98/gait

1. Introduction

1.1 Biometrics

Biometrics are used in a wide array of applications, which makes a precise definition difficult to establish. The most general definition of a biometric is:

> *"A physiological or behavioural characteristic, which can be used to identify and verify the identity of an individual"*

There are numerous biometric measures which can be used to help derive an individuals identity. They can be classified into two distinct categories:

> ***Physiological*** – these are biometrics which are derived from a direct measurement of a part of a human body. The most prominent and successful of these types of measures to date are fingerprints, face recognition, iris-scans and hand scans.
>
> ***Behavioural*** – extract characteristics based on an action performed by an individual, they are an indirect measure of the characteristic of the human form. The main feature of a behavioural biometric is the use of time as a metric. Established measures include keystroke-scan and speech patterns.

Biometric identification should be an automated process. Manual feature extraction would be both undesirable and time consuming, due to the large amount of data that must be acquired and processed in order to produce a biometric signature. Inability to automatically extract the desired characteristics which would render the process infeasible on realistic size data sets, in a real-world application.

With a biometric a unique signature for an individual does not exist, each time the data from an individual is acquired it will generate a slightly different signature, there is simply no such thing as a 100% match. This does not mean that the systems are inherently insecure, as very high rates of recognition have been achieved. The recognition is done through a process of correlation and thresholding, but systems offering 100% recognition should be greeted with a pinch of salt.

1.2 Why Gait?

The definition of Gait is defined as:

"A particular way or manner of moving on foot"

Using gait as a biometric is a relatively new area of study, within the realms of computer vision. It has been receiving growing interest within the computer vision community and a number of gait metrics have been developed. Early psychological studies into gait by Murray

[2], suggested that gait was a unique personal characteristic, with cadence and was cyclic in nature. Johansson [3] carried out studies by attaching moving lights onto human subjects on all the major body parts and showed these moving patterns to human observers. The observers could recognise the biological patterns of gait from the moving light displays (MLD's), even when some of the markers were removed, once again indicating gait as a potential candidate as a prominent biometric.

We use the term gait recognition to signify the identification of an individual from a video sequence of the subject walking. This does not mean that gait is limited to walking, it can also be applied to running or any means of movement on foot. Gait as a biometric can be seen as advantageous over other forms of biometric identification techniques for the following reasons:

- ***Unobtrusive*** – the gait of a person walking, can be extracted without the user knowing they are being analysed and without any cooperation from the user in the information gathering stage unlike fingerprinting or retina scans.
- ***Distance recognition*** – the gait of an individual can be captured at a distance unlike other biometrics such as fingerprint recognition.
- ***Reduced detail*** – gait recognition does not require images that have been captured to be of a very high quality unlike other biometrics such as face recognition, which can be easily affected by low resolution images.
- ***Difficult to conceal*** – the gait of an individual is difficult to disguise, by trying to do so the individual will probably appear more suspicious. With other biometric techniques such as face recognition, the individuals face can easily be altered or hidden.

Being a biometric, an individual's biometric signature will be affected by certain factors such as:

- ***Stimulants*** – drugs and alcohol will affect the way in which a person walks.
- ***Physical changes*** – a person during pregnancy, after an accident/disease affecting the leg, or after severe weight gain / loss can all affect the movement characteristic of an individual.
- ***Psychological*** – a persons mood can also affect an individuals gait signature [1].
- ***Clothing*** – the same person wearing different clothing may cause an automatic signature extraction method to create a widely varying signature for an individual.

Although these disadvantages are inherent in a gait biometric signature, other biometric measures can easily be disguised and altered by individuals, in order to attempt to evade recognition.

The process of automatic gait feature extraction is made more difficult from the external factors such as lighting conditions, self occlusion of feature points when the subjects legs cross over and different types of clothing, which can all affect the data acquisition process.

Results from previous work on automatic gait recognition look promising [6,7,8,9,10,12], with result ranging up to 100% successful identification rates on small database samples. Subsequent work has to be carried out on larger databases to ascertain how effective an identification method this will be with a large population size, but initial reports are promising.

1.3 Project motivation

The ability to be able to identify an individual efficiently and accurately is an important task. Controlled environments such as banks, military installations and even airports need to be able to quickly detect threats and provide differing levels of access to different user groups. Recent events such as September 11th have brought biometrics a lot of attention as a method of identification.

Gait as a biometric has many advantages as stated above which make it an attractive proposition as a method of identification. Gaits main advantage, unobtrusive identification at a distance, makes it a very attractive biometric. The ability to identify a possible threat from a distance, gives the user a time frame in which to react before the suspect becomes a possible threat. Another motivation is that video footage of suspects are readily available, as surveillance cameras are relatively low cost and installed in most buildings or locations requiring a security presence, the video just needs to be checked against that of the suspect.

As well as the inherent advantages of gait, the increase in processor power, along with the fall in price of high speed memory and data storage devices have all contributed to the increased availability and applicability of computer vision and video processing techniques. Real time video processing, which is required for gait recognition is a feasible possibility on current home PC technology, making this technology a viable security application.

1.4 Gait Recognition Scenario

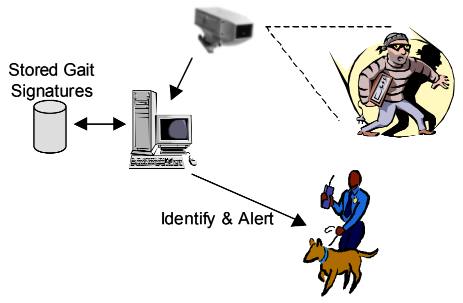


Figure 1.1 – gait recognition in action

Gait recognition can be used in a number of different scenarios. One example would be to analyse the video stream from surveillance cameras. If an individual walks by the camera who's gait has been previously recorded and they are a known threat, then the system will recognise them and the appropriate authorities can be automatically alerted and the person can be dealt with before they are allowed to become a threat. The threat has been successfully detected from a distance, creating a time buffer for authorities to take action.

Such systems have a large amount of potential application domains, such as airports, banks and general high security area.

1.5 Project scope

The main objective of this project is to:

1. Develop a program capable of performing recognition of individuals derived from a video sequence of a person walking. The program should be able to store the derived gait signature for comparison at a later stage.

2. Automatic extraction of relevant gait feature points should be available from a video sequence in order to automate the classification process.

The project can then be broken down into three main sections, which were completed in the following order:

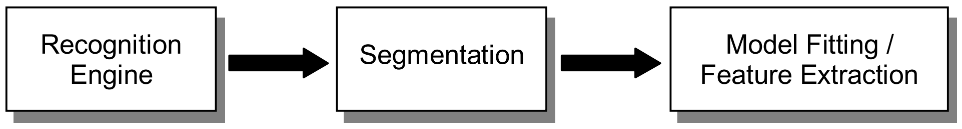


Figure 1.2 – project process

- ***Recognition Engine*** – develop the algorithms and functionality that can classify individuals based on extracted gait information.
- ***Segmentation*** – extract the foreground subjects from the video sequence, ready for extracting gait features.
- ***Feature Extraction*** – the segmented image map is used to extract the relevant gait features which will be used for classification.

1.6 Report structure

The rest of this report is structured as follows:

- ***Chapter two*** gives an insight into previous work which has been pursued into gait recognition, along with theory based upon the methods used throughout this project.
- ***Chapter three*** explains about the overall conceptual design of the project along with the different technologies and development environments used.
- ***Chapter four*** discusses some of the ubiquitous classes, which are implemented throughout the whole project. Having an understanding of these classes will help understanding of other parts of the system.
- ***Chapter five*** outlines the segmentation process used to extract the person from the video sequence, detailing design and implementation.
- ***Chapter six*** explains the methods used in the semi-automatic model fitting. A model based approach was taken to extract the feature points from the video sequence and all of the implementation is explained here.
- ***Chapter seven*** explains how the gait signatures are generated from the captured data, these include signatures based on angles, angular velocity and self similarity plots.
- ***Chapter eight*** gives details how the classification process takes place and how the gait signatures and projected into feature space.
- ***Chapter nine*** explains more about the 3D graphics engine which was produced to displaying the human skeletal models on screen.
- ***Chapter 10*** gives an overview of the application interface, and a tutorial on how to perform the basis functions that the program is capable of performing.
- ***Chapter 11*** is the evaluation of all the processes carried out in the project, such as segmentation, model fitting and recognition.
- ***Chapter 12*** gives the overall conclusion of the outcome of the project.

2. Background

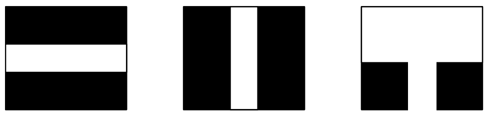
2.1 Overview

In this section I will investigate the current state of the art in gait recognition, providing an overview of the methods which are currently being investigated and will provide background information on all of the techniques which have been used throughout this project. Gait recognition techniques can be broken down into two main sections, model based and model free. These different approaches are described in more detail below.

2.2 Model Free

The advantages of a model free approach are that the methods derived are not linked to one object, it is a holistic approach, therefore a method detecting human gait could be used also for animal gait and vice versa with little modification. A number of model free approaches to gait recognition have been investigated. Some of them are detailed below:

1. Nixon *et al* [8] aimed to use an area based metric for measuring gait signatures. The aim is to combine holistic (concerned with the whole) and model-based approaches to recognition, by using statistical data which linked to the gait of a subject. Foster, Nixon & Bennett suggested an implementation where the human silhouette is captured from the video sequence and then a gait mask is placed over the silhouette. Each gait mask isolates a specified portion of the original image, the change in the area of the silhouette inside each of the gait masks can be measured and then used to produce series, which can be analysed for unique characteristics. Examples of gait masks, figure 2 below:

   
   Figure 2 – sample gait masks

   The technique has produced encouraging results with recognition rates of over 80% on a small database, even when noise has been added to the original video sequence and has shown that gait recognition is possible by only using the temporal components of the human silhouette.

2. Dr V. Huang [9] performed gait recognition using PCA and Canonical Analysis. He used the silhouette of the subject during motion to derive the gait parameters, this motion was then compressed using PCA. He then applied Canonical analysis to derive the signature from which the subject can be recognised. A recognition rate on a small database had a success rate of 100% suggesting that this technique is reliable and has the potential to be improved and extended.

3. J.E. Boyd & J.J. Little [10] used a technique of identifying individuals by studying the variations in the motion description of a subject as they walked. They took a short video sequence and derived the dense optical flow of the subject in both the x and y direction. Scalars of these values were then created based on moments of moving points in order to characterise the shape of the motion not the shape of individual points. They used least-squares linear prediction spectrum analysis to find the fundamental frequency and phase. These were then used as the basis for identification comparison. The results from this looked quite promising with recognition rates of 95% when using the four best recognition signature features, based on the nearest neighbour algorithm.
4. BenAbdelkader et al. [18] used a motion based approach, segmenting the person from the background, and then computing a self-similarity plot of the of the captured foreground images over a number of frames. The self-similarity plot is a measure of correlation between two different extracted foreground regions at time t_i and t_j. Principle Component Analysis was then performed to reduce the dimensionality of the extracted images followed by clustering analysis. Results were promising with recognition rates up to 78% being obtained from a near-fronto-parallel view.

2.3 Model based

Model based approaches to feature extraction, use priori knowledge of the object, which is being searched for in the image scene. Models used are typically stick representations either surrounded by ribbons or blobs, as shown in figure 1. When modelling the human body, there are various kinematical and physical constraints we can place on the model which are realistic i.e. maximum variation in angle of knee joint.

The advantages of a model based approach are that evidence gathering techniques can be used across the whole image sequence before making a choice on the model fitting. Models can handle occlusion and noise better and offer the ability to derive gait signatures directly from model parameters i.e. variation in the inclination of the thigh. They also help to reduce the dimensionality needed to represent the data.

The disadvantage of implementing a model based approach to is that the computational costs, due to the complex matching and searching that has to be performed are high. Although this is a limitation computing power is always increasing so this can be seen as less of a disadvantage, especially in non real-time applications, and efficiency improvements can be found for most algorithmic implementations, which help to reduce the computational costs.

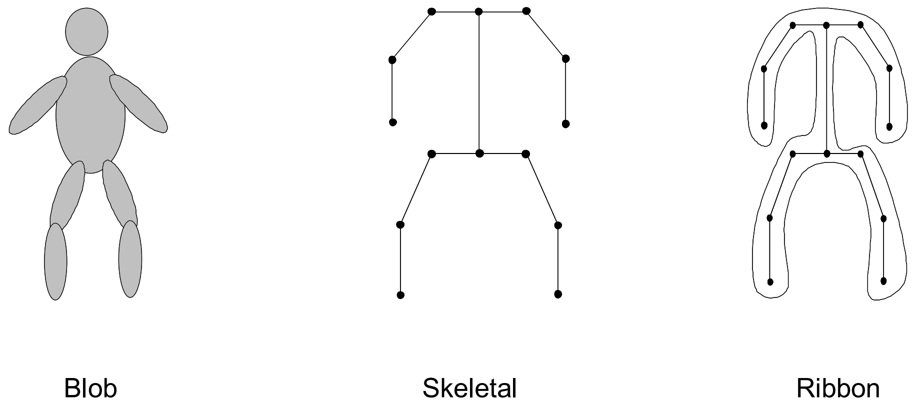


Figure 2.1 – different modelling techniques

When modelling the human body the method shown in figure 1 can also be used. Modelling the human body as a skeleton is a good representation due to the underlying skeleton of the physical body and the restrictions it creates. In this model the skeleton can be thought of as rigid segments between articulating joints. In a 2D environment, rotation not in the x,y direction can cause changes in the length of these bones and has to be taken into account during the modelling stage.

It is possible to extract a skeletal model from an image using Medial Axis Transformation (MAT) [Blum, 1967]. MAT is a process which can be used for reducing foreground regions to a skeletal representation, whilst preserving the connectivity of the original region. Each point on the skeletal representation contains both time and space information, allowing the original figure to be reconstructed. It was found stick figures could be extracted from video sequences using Medial axis transformations and these matched quite closely to the skeletal models of humans [4].

Wren et al [5] produced a real-time people finder program (PFinder), which models and tracks the human body using a set of blobs, the blobs correspond to the person's hand, head, feet, shirt and trousers. The foreground region is extracted from the background and then a model building process follows where blobs are placed over the foreground region. Tracking involves a loop of predicting the appearance of the person in the new image, determining the likelihood for each pixel to be part of the blob model or part of the background.

One of the most important aspects in gait recognition is capturing accurately the positions of the legs, these are the best source for deriving a gait signature and will contain most of the variation in the subjects gait pattern. The legs of a human are usually modelled based upon Singular Harmonic Motion (SHM) in gait applications [6,7,12]. The legs can be considered as a pair of connected pendulums, with a number of constraints imposed that accurately reflect the physical constraints of the human legs [6, Kuan(reference 8)].

The aim of a model based approach is to model the motion of a human, and then fit this model to the motion of a human being tracked see Cunado [7]. Previous projects involving gait recognition using model based approaches are detailed below:

1. Dr. D. Cunado [7] modelled the leg as a pendulum. The method of identification was defined by calculating the difference between SHM and the motion of the subjects' thighs. The gait signature was successfully extracted and could withstand differing amounts of noise and occlusion. This method achieved recognition rates of 100% on a database of ten subjects.
2. Nixon *et al* [6] considered just the legs to extract the gait signature, the legs were modelled as the motion of interlinked pendula. The Hough transform was used to extract the lines which represent the legs in a video sequence. The change in the inclination of these lines was recorded and Fourier transform analysis is used to reveal the frequency change in the inclination of the legs. Classification rates using phase-weighted magnitude spectra incorporating the k-nearest neighbour rule had a classification rate of 90%.

An example of the main modules of a model based approach using image segmentation is shown below:

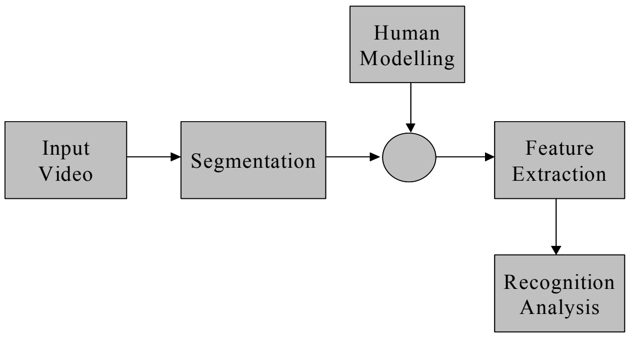


Figure 2.2 – overview of gait recognition process

The basic concept is to first segment the foreground information (i.e. the subject) from the background. The features used to derive the gait signature must be extracted as accurately from the segmented object as possible. These parameters can then be analysed and stored for recognition use at a later stage.

2.4 Principle Component Analysis (PCA)

Principle component analysis (PCA), also known as eigenanalysis, is a technique used to reduce the dimensionality of data and examine the relationship between a set of correlated variables. PCA has been used successfully before in both gait and face recognition techniques [9, 16, 18].

Dimensionality reduction is vital to the recognition purposes because the size of recognition matrices can be vast and very computationally expensive or infeasible. For example Huang et al[reference] create a feature vector by concatenating the columns of each image into one feature vector, obviously this has a large dimension, possibly greater than 1000, which would be infeasible to use for recognition purposes. PCA extracts the main variation in the feature vector and allows an accurate reconstruction of the data to be produced from only a few of the extracted feature values, hence reducing the amount of computation needed.

The aim of using PCA is to be able to represent most of the variation of the original variables using only a few “principle components”. This can be seen in the diagram below, after performing PCA, the eigenvectors (which are all orthogonal to each other) represent the new axis, in ascending order of variation in the original data set.

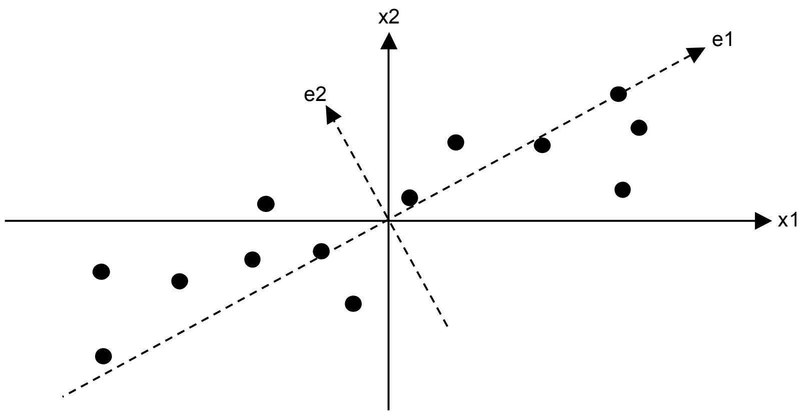


Figure 2.3 – example of PCA in action

In order to calculate the PC of a given set of feature vectors X = $x_1, x_2, \ldots x_n$, which are created by placing all of the original data for a given configuration in a single vector, first we find the correlation matrix C. This is a symmetric matrix that helps to reduce the computation when calculating the eigenvalues and eigenvectors, the formulation is shown in 2.1.

$$C = \frac{1}{N}\sum_{k=1}^{N}(x_k - x_m)(x_k - x_m)^T \tag{2.1}$$

The mean value of the vectors is given by:

$$x_m = \frac{1}{N}\sum_{k=1}^{N} x_k \tag{2.2}$$

The mean is subtracted from all of the vectors in order to produce a set of normalised difference vectors. The correlation matrix can then be represented by a set of special vectors, which satisfy the following equation:

$$Ce_k = \lambda_k e_k \tag{2.3}$$

These vectors are called eigenvectors, each eigenvector, e_k, has an associated eigenvalue λ_k. The largest eigenvalues of the correlation matrix, represent the largest inherent variation in the original data set and tell us most about the original data. The eigenvectors can be rearranged back into the form they were derived from at the end process if required (as in face recognition).

The most popular way to derive the eigenvalues and vectors is to use Singular Value Decomposition (SVD), although may other techniques are also possible. The technique for computing these values has been outlined in Numerical Recipes [17] and will not be discussed any further here. It should be noted that if all of the variables are uncorrelated to begin with then the PCA will not be of much use and return nearly as many non-zero eigenvalues as we had variables in the original data.

Given that X is the feature matrix, where the matrix dimensions have the following property, m >> n, calculating the square matrix $R=XX^T$, which is used for PCA becomes computationally infeasible as mentioned earlier, Huang[19] used the following technique based on SVD to reduced the amount of computation needed to computed the eigenvalues and eigenvectors of R. Given:

$$\lambda_i e_i = \mathrm{Re}_i \tag{2.4}$$

Normally the square matrix R which is used for calculating PCA is computed by:

$$R = XX^T \tag{2.5}$$

Instead an alternative square matrix $\tilde{R}$ can be computed by:

$$\tilde{R} = X^T X \tag{2.6}$$

$\tilde{R}$ is now a square matrix of n x n, which is a much smaller size than m x m. This significantly reduces the computation needed for calculating the eigenvector. The eigenvectors and values of $\tilde{R}$ are related to R in the following manor:

$$\begin{cases} \lambda_i = \tilde{\lambda}_i \\ e_i = \tilde{\lambda}^{-\frac{1}{2}} X \tilde{e}_i \end{cases} \tag{2.7}$$

Once the principle components have been calculated, the next step is to decide how many of the PCs should be kept, in order to maintain a correct and accurate representation of the original data. One method is to define a threshold value t, such that the total number of principle components kept should be greater than this value usually 80-90%. The total proportion of the variance of the original variables, accounted by p principle components is given by:

$$v = \frac{\sum_{k=1}^{p} \lambda_k}{\sum_{i=1}^{N} \lambda_i} \qquad (2.8)$$

It is normally the case that just the first three or four principle components will be kept, as the first PC is usually very large and the drop off of variance representation is quite steep. From the graph below (figure 4), we can see that with relatively few PCs we can represent 80-90% of the total variance.

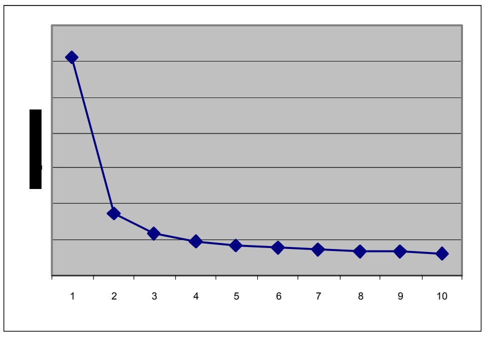


Figure 2.4 – eigenvalues of principle components

The final step is to project the feature vectors into the new eigenvector space, these projected points can then be used for classification. Given the feature vectors $x_{i,j}$, they can be projected into eigenspace by multiplying the feature vectors by the newly calculated eigenvectors:

$$y_{i,j} = [e_1, e_2, \ldots, e_{k-1}, e_k]^T x_{i,j} \qquad (2.9)$$

2.4.1 Effect of noise

As with all real numbers there will be errors introduced either by rounding (inherent in floating point operations) or as noise in the initial capture of the data set. This will have the effect of making eigenvalues who should have zero values to have very small non-zero values. This should not be a problem if the threshold method is applied.

2.5 Canonical Analysis (CA)

Canonical Analysis offers the ability to express the relationship between two or more *sets* of variables. If items can be classified into one of g groups, then the total variation can be seen as the combination of between-group variation and within-group variation. Ideally we want to be able to maximise the between-group ratio and minimise the within-group ratio in order to increase separation of the different classes. Canonical Analysis finds the linear variables (canonical variates), which help maximise this ratio.

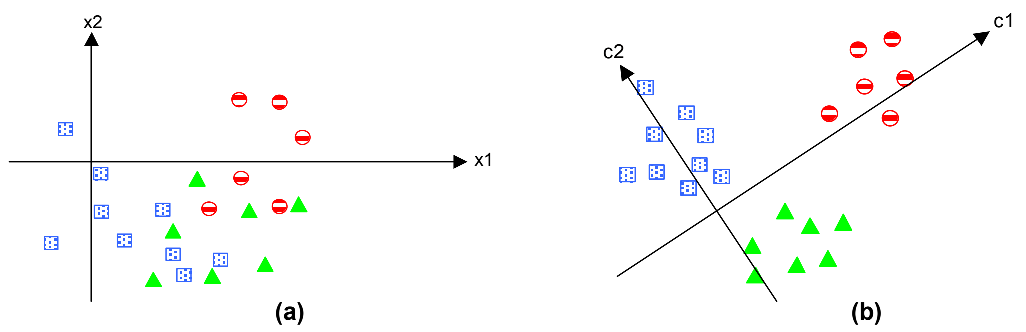


Figure 2.5 – example of canonical analysis

An example of the outcome of Canonical Analysis can be seen above in figure 2.5. The different groups (classes) of data are initially quite close and have quite large inner variance. Ideally CA aims to separate the classes (as in figure 2.5 b) and minimise the within class variance, hence improving the classification rates of the different groups.

Canonical Analysis is used in advanced classification techniques, and has been successfully applied to gait recognition [19]. Huang et al [19] used Canonical Space Transformation to separate different training classes of individuals walking, based on feature points projected into Eigenspace (created after applying PCA). Computing the within S_w and between S_b class variance of the projected eigen points, $y_{i,j}$ represents feature point j of class I (total of c different classes), is done by:

$$S_w = \frac{1}{N_T} \sum_{i=1}^{c} \sum_{y_{i,j} \varepsilon \Psi_i} (y_{i,j} - m_i)(y_{i,j} - m_i)^T \qquad (2.10)$$

$$S_b = \frac{1}{N_T} \sum_{i=1}^{c} N_i (m_i - m_y)(m_i - m_y)^T \qquad (2.11)$$

where

$$m_i = \frac{1}{N_i} \sum_{y_{i,j} \varepsilon \Psi_i} y_{i,j}$$

$$m_y = \frac{1}{N_T} \sum_{i=1}^{c} \sum_{j=1}^{Ni} y_{i,j}$$

Once S_b and S_w have been calculated the generalised eigenvector problems (2.12) needs to be solved in order for us to find the new set of canonical axis. This will give use a set of eigenvectors which represent the orthogonal axis in canonical space where classification can take place:

$$S_b w_i^* = \lambda_i S_w w_i^* \qquad (2.12)$$

This problem can be solved by initially preparing the S_b and S_w matrix in order to be able to calculate the eigenvalues and eigenvectors. Firstly we perform Singular Value Decomposition on S_w:

$$S_w = VSV^T$$

V is orthogonal (i.e. V = inv(V)), and S is a diagonal matrix, where the diagonal elements represent the singular values. The next step is to calculate a matrix U:

$$U = (VS^{-\frac{1}{2}})^T S_b (VS^{-\frac{1}{2}})^T$$

Next the matrix U is split using Singular Value Decomposition:

$$U = ABA^T$$

$$delta = VS^{-\frac{1}{2}} A$$

Delta is used to help diagonalise the between and within matrices, this is needed to be able to compute the eigenvectors and eigenvalues. Finally the matrix eigenproblem is computed, and this is the matrix which we derive the eigenvectors and eigenvalues for, which will represent the axis of the canonical space. There will be c-1 (c is the number of different classes, i.e. number of different people caught on video), non-zero eigenvectors which can be used for projecting the points into canonical space:

$$eigenproblem = delta * B * inv(delta)$$

Canonical Analysis is like PCA, the eigenvectors who have large eigenvalues represent the axis of most variation in the data set, we only need to use the larger eigenvectors in order to be able to represent most of the variation in the original data set.

2.6 Image segmentation

2.6.1 Aim

In order to be able to perform analysis on the gait of the individuals caught on video the subject needs to be extracted from the video sequence. Image segmentation is used to separate dynamic objects such as people, which are part of the foreground, from the background of the image sequence. It is a very important pre-processing step in many computer vision applications, accurate retrieval of the foreground objects are vital in order to minimise distortion or inaccuracies. These may propagate into other parts of the vision system, which could affect results at later stages in the processing.

2.6.2 Approaches

One of the most common and simplest methods for performing segmentation is to first carry out image subtraction. The known background image is subtracted from the current picture frame, comparing the intensities of relating pixels, then thresholding is performing. A pixel is considered part of the foreground when the current pixel value differs from its mean value by more than a pre defined threshold value ϕ:

$$f(x,y) = \begin{cases} background, & if \quad abs(I_{current}(x,y) - I_{known}(x,y)) \le \phi \\ foreground, & otherwise \end{cases} \tag{2.13}$$

Segmentation is a complex process and there are several problems inherent with extracting foreground regions, such as occlusion, shadows (cast by the foreground objects as they move) and noise. The process is complicated by the fact that the background may not be static, i.e. a changing television screen in the background does not want to be considered as a foreground object but is continually changing. Also a foreground objects intensity/colour at a given point may be very similar to the background and hence the foreground object may be considered part of the background. Several techniques have been developed that include both range and colour [11] in order to minimise the distortion of these effects. The effect of these distortion factors can be seen in figure 2.6 below where simple subtraction has taken place:

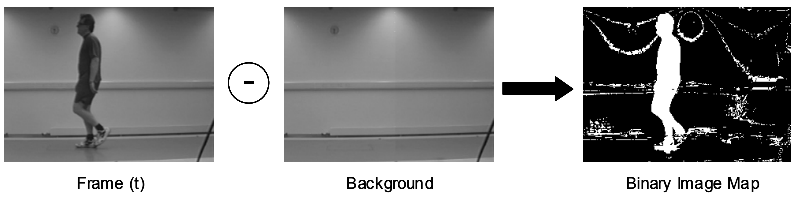


Figure 2.6 – background subtraction

In the above pictures, although the background is static, the flicker of the light, which is undetectable to the human eye causes large changes in intensities, along with the subjects shadow, resulting in poor segmentation, the binary image shows just the subtraction process.

Segmentation can be improved by building a model of the background pixel intensities. The model of the background can be built on a combination of statistical range and colour values for each pixel in the scene. If the background is continually changing gradually over a period of time, the model will have to be updated over time to reflect these changes. Various simplification assumptions can be made in controlled environments to enhance the performance of segmentation.

As well as creating a better model of the background in the image, it is also possible to apply image filters to the foreground image map, which help to reduce noise (pixels which have been misclassified as foreground pixels) and classify pixels into groups.

2.6.3 Morphological Filters

Such filters are called Morphological filters. They usually take a binary image and process the objects in the input image based on characteristics of its shape, which are derived from the overlayed structuring element, using basic set operations such as intersection and union. A good example of these filters in use is the W^4 [20](Who? When? Where? What?) project. This project tracks individuals and body feature points in numerous settings. They used morphological filters to remove background noise and classify blobs. The main filters used for image pre-processing are Erosion, Dilation and Connected Component Labelling, these are all subsequently discussed in more detail.

2.6.3.1 Connected Component Labelling

The purpose of connected component labelling is to group together pixels which have similar properties and are connected in some way. The image is scanned from top to bottom and left to right, pixels which should be grouped together are given the same label.

The basic pseudo algorithm for 8-neighbour connectivity is:

```
for( each foreground pixel ($p_i$) ){
        1. examine neighbouring pixels which have already been labelled
        (ie top left, top, top right and left)

        2. Based on the information above:
                i)If all 4 neighbouring pixels are 0 then give $p_i$ a new label.
                ii)If only 1 neighbour ($p_j$) is a foreground pixel, assign $p_i$ $p_j$'s label.
                iii)If more than 1 neighbour is a foreground pixel, assign one of the
                labels to $p_i$, make a note of the equivalences of the existing labels.
}
```

After completion of initial scan loop through all the pixels once again replacing equivalent labels with one single label.

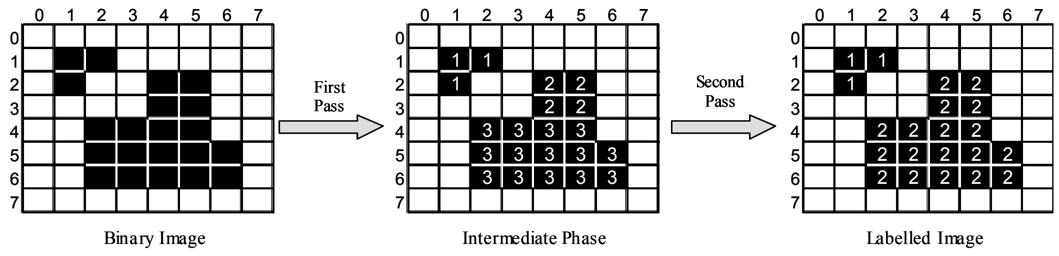


Figure 2.7 – example of connected component labelling

A two pass scan can be seen above. Initially a binary image is passed into the function, the image is then scanned and each foreground pixel is given a label, note for pixels (4,3), (4,4) and (4,5) they have the choice of being assigned label 2 or label 3, therefore this equivalence (ie label2 == label3) must be noted. In the final pass all equivalent labels are replaced by one unique label.

Once all regions have been labelled, it is then possible to remove regions which are smaller than a certain threshold from the binary image as it is most likely these do not belong to the main foreground blobs.

2.6.3.2 Erosion

The basic effect of the erosion operator is to erode away at boundaries of foreground pixels, thus in effect shrinking foreground regions. This is an excellent way of removing small noise spots in an image.

It works by placing a kernel, a pre defined search area shape, on top of each foreground pixel. All foreground pixels which fall inside the kernel are counted and if the number of pixels is less than a predefined threshold then the centre foreground pixel is eroded and now is considered to be part of the background.

The 3x3 kernel is the most common structure used in erosion, but other kernels of varying sizes, orientations and shapes can be used, depending on the nature of the image.

$$e(i,j) = \sum_{(m,n)\in\Omega_{i,j}} f(m,n)$$

$$\text{where } f(m,n) \text{ returns } 0,1$$

$$\Omega = \text{all pixels belonging to kernel} \tag{2.14}$$

$$f'(i,j) = \begin{cases} f(i,j), \text{if } e(i,j) \geq T \\ 0, \text{otherwise} \end{cases}$$

An example of erosion on a binary image using a square 3x3 kernel, and applying a threshold of five foreground pixels can be seen below:

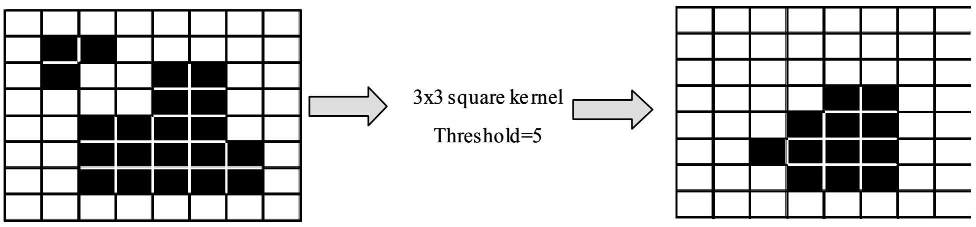


Figure 2.8 – example of erosion filter on binary image

2.6.3.3 Dilation

Dilation is the complementary operation to erosion. The effect of dilation is to gradually enlarge the foreground region by a factor each iteration of dilation which is performed. Dilation is useful for enlarging regions which may have been eroded by the erosion process.

For each background pixel in the binary image, overlay the kernel (as described in the erosion section), if any of the pixels inside the kernel are foreground pixels the make the centre pixel part of the foreground. It is also possible to add a threshold to the equation which allows areas to be dilated only if the number of foreground pixels is greater than a certain threshold.

An example of dilation is shown below, using a 3x3 square kernel and 8-neighbourhood connectivity.

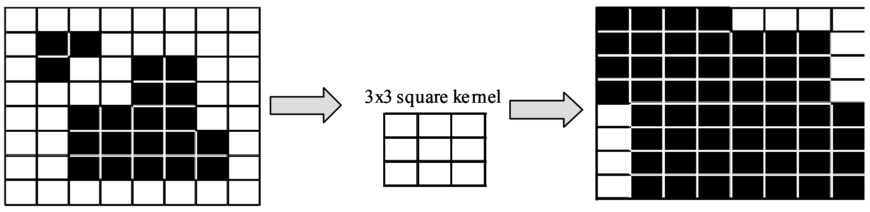


Figure 2.9 – example of dilation on binary image

2.7 Self similarity Plots

A self similarity plot is a measure of how similar a measured parameter is over a period of time, it measures the correlation of the variable over a number of recorded readings. For example, in relation to gait recognition the similarity plot may measure the similarity of the angle of incline of the thigh, or the similarity of extracted foreground regions over N frames.

The latter was investigated by BenAdbelkader et al [18], they successfully used a self similarity plot for gait recognition. The plot was constructed by comparing the similarity of the pixel intensities of segmented foreground region blobs. An example of one of the plots produced can be seen below:

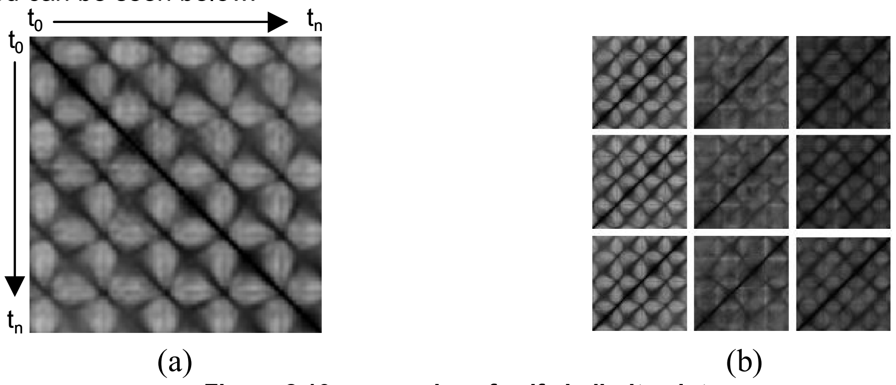


Figure 2.10 – examples of self similarity plots

The dark areas signify high correlation, and the lighter areas signify a drop in the correlation of the measured parameter, each pixels represents one period in time. The plots above are for people walking, it is evident that the frequency and phase of the person walking is displayed in the plot.

Figure 2.10 (b) shows the plots produced from three different subjects. Each different subject represents one column, and each row represents a different video sequence of the individual. It can be seen that the different people maintain a similar plot on different video sequences, and that the plots differ from person to person, giving evidence that they could be used as a gait classifier.

Calculating the plot based on image intensities can be done as follows:

$$S(t_1,t_2) = \sum_{(x,y)\in B_{t_1}} \left| O_{t_1}(x,y) - O_{t_2}(x,y) \right| \tag{2.15}$$

Where O_t are scaled image templates, representing the extracted object at time t and B_t represents the minimum bounding box surrounding the extracted object. In order to account for small errors we can introduce error tracking in a small radius around the target area:

$$\overline{S}(x,y) = \min_{|dx,dy|<r} \sum_{(x,y)\in B_{t_1}} \left| O_{t_1}(x+dx,y+dy) - O_{t_2}(x,y) \right| \tag{2.16}$$

Properties of self similarity plots are:

1. S(t,t) = 1, i.e. the main diagonal has high correlation
2. $S(t_1,t_2) = S(t_2,t_1)$, i.e. it is symmetrical along the main axis

Obviously self similarity plots are not limited to being produced in just this way, they can be computed from any parameter which can be measured over a period of time.

2.8 Relational Modelling

Modelling in computer vision is strongly reliant on the use of prior knowledge of the desired scene in order to be able to build a model for feature extraction. Using models in the feature extraction process helps to improve the accuracy of the final results, as well as helping to narrow down the search space during model fitting, due to the constraints placed upon the model.

Relation modelling is used to make assumptions about the structure of objects in a scene, based upon the relations of the different objects which make up the whole scene. The granularity of the model can be at any level of detail, depending on the nature of the image being analysed and the requirements of the system. Relational models help vision systems relate to real world scenarios, when perfect data extraction is not always possible.

Rules can be incorporated that help to reduce the objects extracted in the image, examples of constraints for a human relational model could be:

- If a circle (i.e. head) is not located above an ellipse (i.e. body) then remove it from the extracted feature list.
- If an ellipse is connected to a circle, buts the length of its major axis is smaller than the circle, then remove the ellipse from the extracted feature list.

Although relations may have the same name, such as “connected”, they may have different implied physical meanings. For example, head connected to body means that the head touches the body but does not overlap, whereas leg connected to body, would mean the leg does overlap the body, because the leg is connected to the hip joint.

One way to express a relational model is in the form of a semantic net. They offer a way of expressing the relational structures with a high level of generality. An example of a semantic net at various levels of detail, representing the relation between different parts of the human body can be seen below:

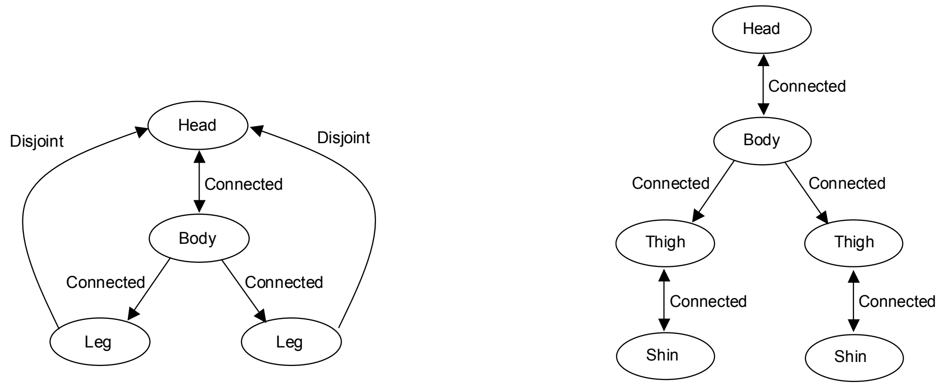


Figure 2.11 – examples of relational models

Semantic nets aim to be as general as possible, so that they can be applicable in many different situations. An important point to note is that semantic nets only store the relational structure of an object, they don’t store any size or location information about the individual components, as this would severely limit their usefulness to extract objects from different scenes.

One of the main advantages from using semantic nets is the ability to infer about a scene. Even if all the components of the object are not successfully matched in the image, but the majority are with the correct relation, then it is possible to infer that the object is indeed in the scene, but some parts have not been extracted, possibly due to occlusion or noise in the original image. We assume that there are functions available for extracting the geometric or iconic shapes represented in the nodes.

2.9 Pattern Classification

Once the gait feature has been extracted from the person, it will be projected into a feature space and it will then have to be classified. This means we have to determine which group in the feature space (i.e. which person) the unknown feature point should belong to.

A classifier defines boundaries in a feature space which are used to separate different sample classes from each other in the data. The most simple type of classifier is a linear classifier. This is a straight line which is defined in the feature space, points above the line are in one class, points below the line are placed into another class. This is a simple method and does not provide the best results due to small non-linear fluctuations around the boundary region which result in poor classification results. An improvement on this method is called the K-Nearest Neighbour (KNN) classification rule.

2.9.1 K-Nearest Neighbour classifier

A feature vector of unknown class can be classified as belonging to a class by using the k-nearest neighbour rule. A training set of points (i.e. feature vectors projected in eigenspace) T is used to determine the classification of feature vector X, by the following method:

1. Calculate the k-nearest points to the unclassified feature vector X in the feature vector set T. There are a number of distance measures which can be used (as described in section 2.12) to calculate the separation of two points in n-dimensional space.
2. Determine the class which has the most points in the k selected points, from set T

The class which has the most points is chosen as the class which point X now will belong to. This is called a voting classifier, and clearly it is able to handle non-linear classification spaces. An example of a linear classifier can be seen in figure 2.12 (a), it shows all the points below the line being classified as class 'a', and the points above the line as being in class 'b'. Figure 2.12 (b) shows the k-nearest neighbour approach to classification. The five closest training points to unknown point U, have been chose. Out of the five points four are classified as belonging to class 'a', therefore by majority vote the unknown point U will be classified as belonging to class 'a'.

This voting clearly has the advantage of being more resilient to noise, with spurious training projection points not dominating the classification process.

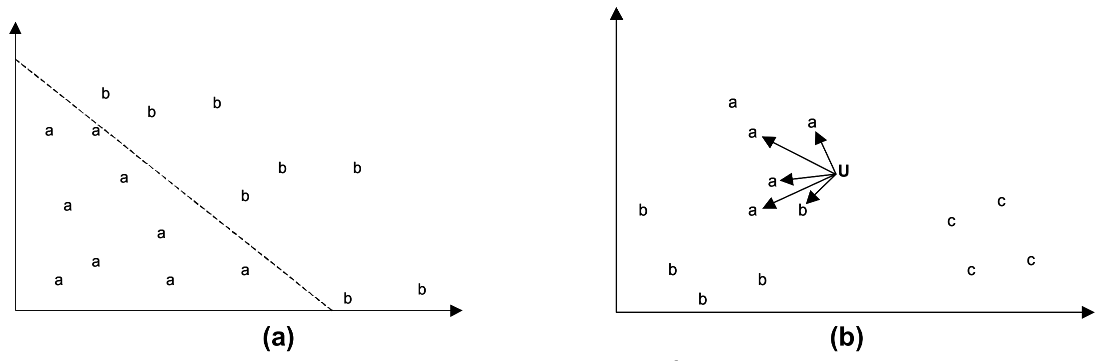


Figure 2.12 – Pattern classification

3. Conceptual Design & Development Environment

3.1 Aim

This chapter discusses the overall conceptual design of the gait recognition architecture, along with the reason for choosing a specific development environment and the design methodologies used during implementation.

3.2 Overall Program Architecture

The holistic view of the project can be broken down into three main subdivisions, inputs, process and outputs. Each one of these subdivisions comprises of a number of modules used for processing and displaying the information flow within the program.

The program architecture can be seen in figure 3.1 below:

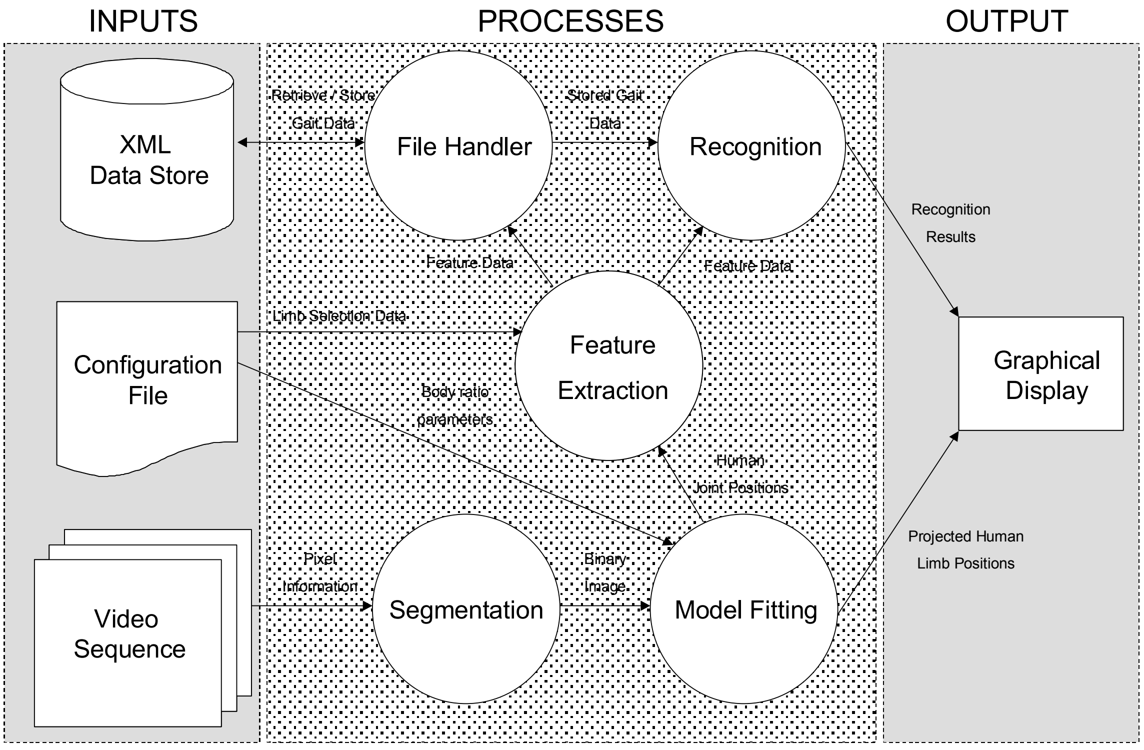


Figure 3.1 – Overall conceptual design

Each different module in the design has a disjoint and individual purpose. These are described in more detail in the following paragraphs.

3.2.1 Inputs

3.2.1.1 XML Data Store – the data store is used to record all of the captured data about each subject in XML format. Users gait data can either be stored in individual query files, or concatenated together to create a database file used for comparison in the recognition process.

3.2.1.2 Configuration File – the configuration file will store parameters such as body part ratios and limb selection data which will be used in the model fitting and feature extraction process. By separating out the parameters used in the program changes can be easily made to any of the assumptions made in the program.

3.2.1.3 Video Sequence – the video sequence is the medium used for viewing the gait of an individual. Video will be transformed into an intermediate format, bitmap, once being loaded into the program in order to allow process of the information within the program.

3.2.2 Processes:

3.2.2.1 Segmentation – this module will take the video sequence as an input, then perform processing in order to determine which pixels are part of the foreground and which are part of the background.

3.2.2.2 Model Fitting – the binary image produced from the segmentation process will be used as input for the model fitting process. This module will fit a model of the human form onto the segmented area of the image.

3.2.2.3 Feature Extraction – once the model has been fitted to the image, features, which can be used to create a gait signature, will be derived from the model parameters i.e. variation in thigh angle over n frames.

3.2.2.4 Recognition – the recognition engine will take data from either a newly captured subject via the feature extraction module, or a previously stored signature, and perform recognition based on a database of test subjects. Various different metric, will be incorporated in the recognition process to facilitate this procedure.

3.2.2.5 File Handler – the file handler is the interface between the XML gait data store. It will have the functionality of reading and writing to and from numerous files, and parsing the XML to load the information into abstract data types used internally within the program.

3.2.3 Output:

3.2.3.1 Graphical Display – the results from segmentation, model fitting and recognition will all be shown in the main application which is developed using MFC.

3.2.3.2 XML Data Store – as well as being able to retrieve data from the data store it is also possible to store new gait data into XML format through the user interface of the application. This offers the ability to record a persons gait and then store it for use in later recognition processes.

3.3 Gait Recognition Design

3.3.1 Gait Signature Creation

The main algorithm used for gait recognition is explained fully later on in this document. The overall method used in this project is to classify different peoples gait using Principle Component on the extracted feature points of each person. This was chosen because it has previously led to good results [18,19] and I wanted to explore the possibilities of using it with different gait feature vectors.

The feature vectors used for gait recognition will be of two types:

- ***Primary features*** – these are features that are directly extracted from the human model, such as the angles of the limbs at each frame in the video sequence, or the position of each joint over the video sequence.
- ***Secondary features*** – features which are derived from primary features. In this project, self similarity feature plots (mentioned earlier) are used to derive a gait signature based on the self similarity of the angle of the limbs over the N frames of video.

3.3.2 Feature Extraction

The features which are used to build the gait signatures can be extracted in two ways:

- ***Manually*** – the user captures the points by manually selecting the areas of interest on each frame of the video sequence.
- ***Semi Automatically*** – A process of image segmentation followed by model fitting is undertaken which will be used to automatically extracted the desired features from the video sequence.

All of these methods are described in more detail in the subsequent sections.

3.4 Development Environment

This project was developed using Microsoft Visual C++ 6.0 and was run and tested on a Pentium 1.3Ghz, with 256MB RAM. C++ was the chosen implementation language, because of the speed of execution of the code. It was possible to develop a program using Java, but this was not chosen as the language because of its slower execution speed. This is because Java is compiled into an intermediate form, Java Bytes rather than machine code, which have to be interpreted at program run time. Also using Visual C++ environment offers the possibility for the Gait tool to be integrated into other programs developed by the Visual Information Processing group at Imperial. Visual C++ also offers excellent debugging tools which aid quick development.

Microsoft Foundation Classes (MFC), which can be used as part of Visual C++, offer the possibility to produce professional applications in the Windows environment through the use of the Windows API.

The main disadvantage of using language is that it limits the application to the Windows environment, because Visual C++ applications based upon MFC will not run on the Linux operating system. Although this is a slight drawback it does not affect the project in any way.

3.5 Design Methodology

The following methodologies were used during implementation of the project:

3.5.1 Object Orientated Design – one of the reasons Visual C++ was chosen for this project, was because of its object orientated capabilities. Within this program, common functionality should be grouped into classes, to allow reuse of code and a minimisation of the amount of repeated code within the program.

3.5.2 Extensibility of data structures and functionality – the design of the overall architecture of the program and the individual classes should be as modular as possible in order to allow future extension of the project. A well structured design will allow new functionality to be added to the overall program design with the minimal of difficulty.

4. Utility Classes

4.1 Overview

The purpose of this section is to explain the functionality of a number of ubiquitous classes which have been implemented in the project. They are not linked specifically to any one main module of the program but are used throughout the entire code, either providing data handling and processing or acting as utility functions to format the data in the correct manor. An overview of all the classes is subsequently given.

4.2 CVector class

The CVector class provides functionality to store data in a vector format and perform vector operations. The vector can be of any size (memory permitting), the size is specified by the user in the constructor of the object. The class offers functionality such as:

- Vector addition/subtraction
- Dot product
- Multiplying / dividing all elements in the vector by a specified value

The vector stores variables which have a double data type, this means it is possible to use the vector for both real and integer calculations. The vector data type is used throughout the project, for storing data such as human joint positions and graphical point projection calculations. The vector is implemented using dynamic memory allocation, therefore it is important to clear up any memory after execution has completed because C++ has no garbage collection facilities.

4.3 CMatrix class

4.3.1 Aim

Once the data has been captured it needs to be stored inside the program in such a way that it can be easily manipulated and used in calculations. The CMatrix class aims to encapsulates all of the required functionality for matrix computations and is used through the code of the project.

4.3.2 Design

A class diagram showing the structure of the CMatrix object and its relationship with the CVector class are shown in figure 4.1 below:

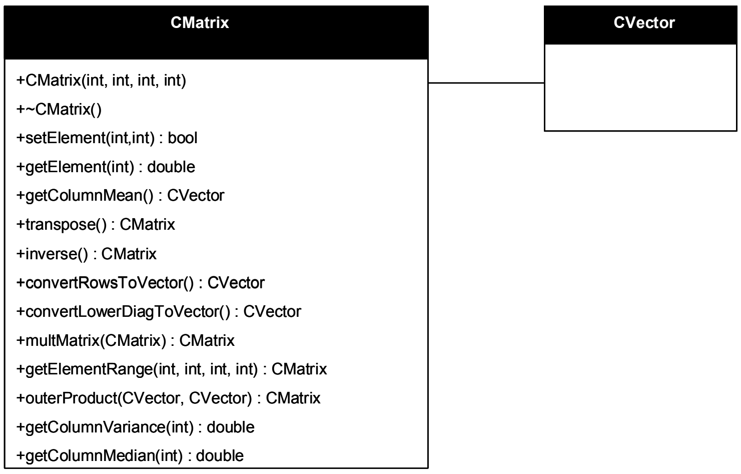

CMatrix
+CMatrix(int, int, int, int)
+~CMatrix()
+setElement(int,int) : bool
+getElement(int) : double
+getColumnMean() : CVector
+transpose() : CMatrix
+inverse() : CMatrix
+convertRowsToVector() : CVector
+convertLowerDiagToVector() : CVector
+multMatrix(CMatrix) : CMatrix
+getElementRange(int, int, int, int) : CMatrix
+outerProduct(CVector, CVector) : CMatrix
+getColumnVariance(int) : double
+getColumnMedian(int) : double
CVector

Figure 4.1 – CMatrix class structure

The CMatrix class has a lot of different functionality, some of the main tasks it performs are:

- Matrix multiplication
- Multiply and add two matrices
- Transpose the matrix
- Compute the inverse of the matrix
- Get the variance of each row/column in the matrix
- Calculate the median value of any row/column
- Append all rows/columns of the matrix to form a single vector
- Convert matrix to lower tri-diagonal form. This is used in eigenvalues computations.
- Extract sub regions of the matrix

All of these different calculations are needed at some point in the project and help to reduce the amount of code needed, due to the easy reuse of the CMatrix object.

The CMatrix class holds numbers stored in double data type. The constructor allows the number of rows and columns in the matrix to be specified at creation, this allows the data type to be dynamic in size which makes it versatile throughout the program.

Most of the algorithms used in the various matrix calculations can be found in the Numerical Recipes book collection [17].

4.4 CHuman class

4.4.1 Aim

The CHuman abstract data type is used to store a single instance of a human configuration, i.e. it can store a snapshot of the captured position of the joints of a human, in one frame.

4.4.2 Design

The human model used in this project is an eight point model. Each point maps onto one joint of the human body. The joints that will be stored in the data type are shown below in figure 4.2.

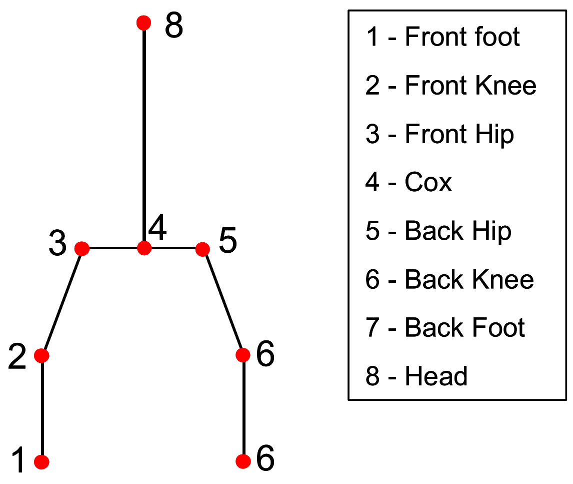


Figure 4.2 – skeletal model of the human frame

More explanation about the actual model and reasons for the choice of joints are detailed in section 6, which talks more about the modelling, normalisation and data acquisition process. The actual class implementation of CHuman can is shown in figure 4.3.

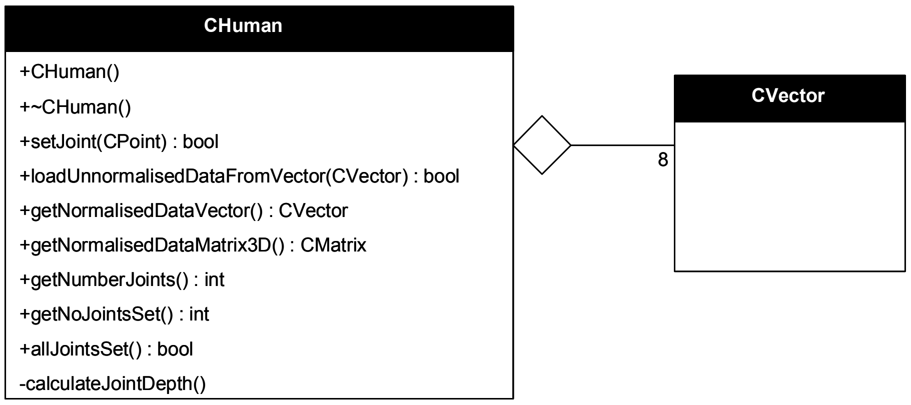


Figure 4.3 – CHuman class design

4.4.3 Implementation

4.4.3.1 Data Storage Structure

Each joint in the data type is stored as a three dimensional CVector data type. Each point is then stored in an array of CVector and macros are defined in the class to give a more intuitive array indexing mechanism i.e.

```
const int FRONT_FOOT = 1;
const int FRONT_KNEE = 2;
…
const int NECK = 8;
```

These macros are used in the code when accessing the array to aid development and produce more understandable code.

4.4.3.2 Data normalisation

Data capture can take place on any size image, therefore the absolute values of the extracted joint positions can be different in scale depending on what image size the data was captured on. Ideally when we are performing comparison we do not want the data to differ in any way as this can affect results. Data normalisation aims to take any ambiguities or factors away from the data and make all the data comparable without bias.

4.4.3.3 Functionality

There are a number of functions which process the stored joint location data in various ways, as shown in figure 4.3. A number of the main functionalities are explained below:

- ***setJoint(CPoint)*** – this sets the position of the currently selected joint
- ***allJointsSet()*** – indiciate is all the joints in the given instance of the data type have had their location data set.

  loadUnnormalisedDataFromVector(CVector) – an alternative way to set the position of the joints, rather than setting them individually using setJoint, is to pass the location data to the object using the function above. The vector passed to the function contains unnormalised data, which is then loaded into the data object. Unnormalised data simply means the actual absolute pixel values where the joint were selected on screen.
- ***calculateJointDepth()*** – this function calculates the z component of each captured joint. Because the joints are captured in 2D the 3D component has to be estimated, this is done in the following steps:
  1. When each joint is captured it is given an arbitrary depth i.e. 1000
  2. Once all joints have been captured the depth of each joint in relation to a reference joint i.e. cox, is calculated by:

     $$joint.z = cox.z \pm widthToHeightRatio * normalised\ height$$

  This gives a rough depth for each joint which can then be viewed using the 3D graphics engine.
- ***getNormalisedDataMatrix3D()*** – returns a data matrix comprising of the normalised 3D data points of the human joint positions. This data is mainly used for the graphical display of the model.

4.5 CHumanSequence

4.5.1 Aim

The CHumanSequence data type is used to store a sequence of body configurations over a number of frames. It is a container for the CHuman class, and provides functionality for computing differing statistics on the captured human poses over a number of video frames.

4.5.2 Design

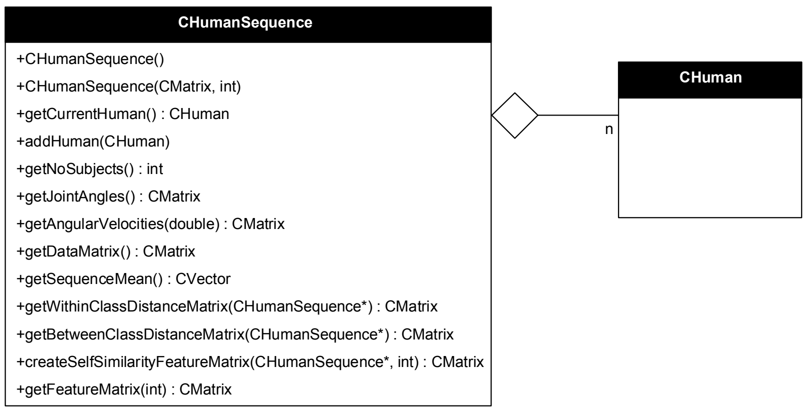

CHumanSequence
+CHumanSequence()
+CHumanSequence(CMatrix, int)
+getCurrentHuman() : CHuman
+addHuman(CHuman)
+getNoSubjects() : int
+getJointAngles() : CMatrix
+getAngularVelocities(double) : CMatrix
+getDataMatrix() : CMatrix
+getSequenceMean() : CVector
+getWithinClassDistanceMatrix(CHumanSequence*) : CMatrix
+getBetweenClassDistanceMatrix(CHumanSequence*) : CMatrix
+createSelfSimilarityFeatureMatrix(CHumanSequence*, int) : CMatrix
+getFeatureMatrix(int) : CMatrix
n
CHuman

Figure 4.4 – CHumanSequence class

4.5.2.1 Data Storage

The data is stored in the CHumanSequence data type as an array of CHuman objects. Each frame of the video sequence is captured and the joint positions stored in a CHuman object and then added into the CHumanSequence object. Grouping the different frames together in a class has a number of advantages, mainly that all the processing which needs to be performed on the sequences can be encapsulated inside the class, reducing the amount of supplicate code needed in the application.

4.5.2.2 Functionality

There are a number of prominent methods in the class, namely:

CHumanSequence(CMatrix, int) – allows the data structure to be initialised with pre captured data, the CMatrix stores a number of frames of captured human poses, each column of the matrix represents one pose.

getCurrentHuman() – returns a CHuman object, of the currently selected frames in the captured sequence.

addHuman(CHuman) – adds a new captured pose to the existing sequence of human poses.

getJointAngles() – returns a CMatrix object consisting of all of the angles of each joint for each frame in the sequence. Each column of the matrix represents one pose, and each row represents a different limb i.e. front thigh, back shin etc.

getAngularVeclocities(double) – returns a matrix representing the angular velocities of each of the limbs, at each frame over the captured video sequence.

getDataMatrix() – returns a CMatrix object consisting of all captured data so far.

getFeatureMatrix(int) – returns a CMatrix object which stores the feature vectors for this sequence. There are numerous feature matrices which can be returned, depending on the value integer parameter passed to the function:

- RECOGNITION_ANGLES – perform recognition based on the angles of the limbs.
- RECOGNITION_ANGV – perform recognition based upon the angular velocity of the limbs.
- RECOGNITION_SELFSIMILARITY – perform recognition based on self similarity plots.

getWithinClassDistanceMatrix(CHumanSequence*) – computes the within class variance for a group of human sequences. For more information look at background section 2.4. Same for the between class distance matrix function. These matrices are used in the Canonical Analysis stage of recognition.

5. Image segmentation

5.1 Aim

The main aim of image segmentation in this project was to extract a good quality binary foreground image, from a video frame. This was needed in order to perform model registration and fitting onto the current image, which is then used to extract the gait feature points from the walking sequence.

5.2 Assumptions & Limitations

The main assumption made in this design process, was that the background in the video sequence is static, the only moving object is the person. If this was not the case then due to changes in pixel intensities regions would be incorrectly classified as foreground regions, this simplification was made because segmentation is not the main part of the project, and a non static background is a non trivial problem.

As well as having a static background it is assumed that all external conditions, such as lighting remain quite constant as well, otherwise this would affect results.

The next main assumption was that the camera position was fixed throughout all of the data capture. This was assumed in order to be able to perform frame subtraction. If the camera was moving, then simple image subtraction would not work, because the temporal correspondence between equivalent pixels would be lost.

5.3 Segmentation Software Design

The main conceptual design of the segmentation engine is shown below:

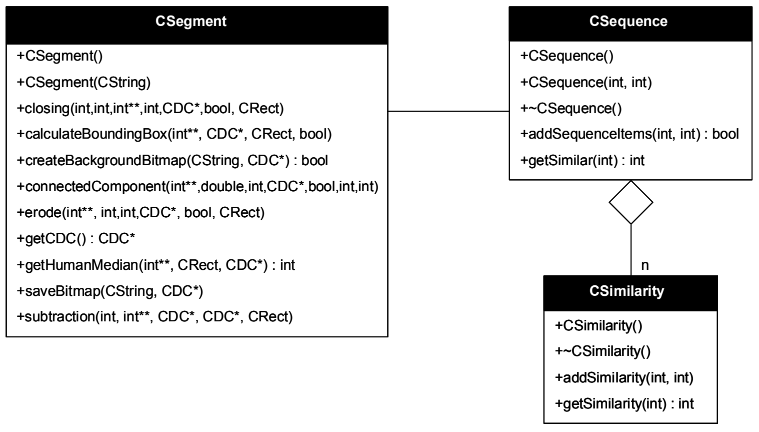

CSegment
+CSegment()
+CSegment(CString)
+closing(int,int,int**,int,CDC*,bool, CRect)
+calculateBoundingBox(int**, CDC*, CRect, bool)
+createBackgroundBitmap(CString, CDC*) : bool
+connectedComponent(int**,double,int,CDC*,bool,int,int)
+erode(int**, int,int,CDC*, bool, CRect)
+getCDC() : CDC*
+getHumanMedian(int**, CRect, CDC*) : int
+saveBitmap(CString, CDC*)
+subtraction(int, int**, CDC*, CDC*, CRect)
CSequence
+CSequence()
+CSequence(int, int)
+~CSequence()
+addSequenceItems(int, int) : bool
+getSimilar(int) : int
n
CSimilarity
+CSimilarity()
+~CSimilarity()
+addSimilarity(int, int)
+getSimilarity(int) : int

Figure 5.1 – segmentation design

The segmentation engine offers the ability to perform a number image pre-processing operations, such as:

- ***Erosion*** – helps reduce the small noise introduced during background subtraction
- ***Background generation*** – the ability to generate a background from a sequence of video images
- ***Median calculation*** - calculates the median x and y position of the person walking in the video sequence
- ***Connected Component Labelling*** – offers the ability to group pixel blobs together under a single label.

5.4 Background Generation

The gait data set used in this project [21], does not have any images in the video sequence which do not include the human subject. Therefore there was no unique background picture to perform image subtraction against. Because of this, the functionality was added to the CSegmentation class which automatically generates a static background image from a sequence of moving images, which have a static background.

This is done by generating a model of the image pixel intensities over a number of frames. The intensity of each pixel was stored for 50 frames of the video sequence i.e. $P(x,y)_{T_{1-N}} = [P(x,y)_1, P(x,y)_2,\ \ldots\ , P(x,y)_N]$. Once these have all been recorded a Quicksort is performed on all the pixel values to order them and then the median value is taken to be the value of the background pixel. An example output from the program is shown below:

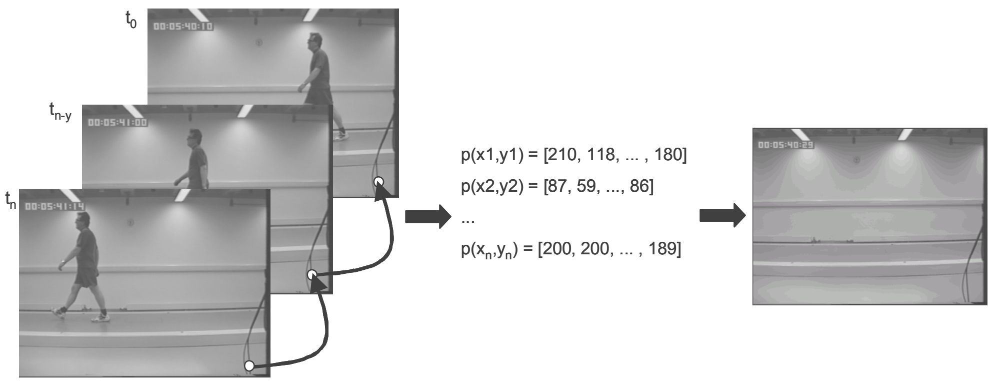

t_0
00:05:40:10
t_{n-y}
00:05:41:00
t_n
00:05:41:14
p(x1,y1) = [210, 118, ... , 180]
p(x2,y2) = [87, 59, ..., 86]
...
p(x_n,y_n) = [200, 200, ... , 189]
00:05:40:29

Figure 5.2 – background frame generation

A median value is taken rather than the mean value of the pixel intensities over n frames, because a mean value will be distorted by the large change in pixel intensities when the person moves past that pixel. The median value is unaffected, as long as large number of frames are recorded, because the median is unaffected by spurious values.

The assumption made in this process is that the person does not stand still over the frames which are analysed, if this is the case then the background generation would class the person as part of the background.

5.5 Foreground Segmentation

It is imperative to remove as much noise and distortion as possible from the segmented foreground image map, to reduce the chances of misclassification of body parts during model registration. My main segmentation algorithm design was based around a method used by [20], which had proved successful in extracting foreground regions from a video sequence.

The process is performed in a two pass approach over an image:

- 1st Pass – generates a static background from the available images, and performs global background subtraction, filtering and small blob removal to locate the smallest bounding region encapsulating the person.
- 2nd Pass – is similar to pass one, but the filtering takes places in a small local region around the person. The subtraction, erosion are performed at lower thresholds to allow a stronger binary image to be produced, which is possible now a smaller search region is being processed.

The individual steps in the algorithm can be seen in figure 5.3 below:

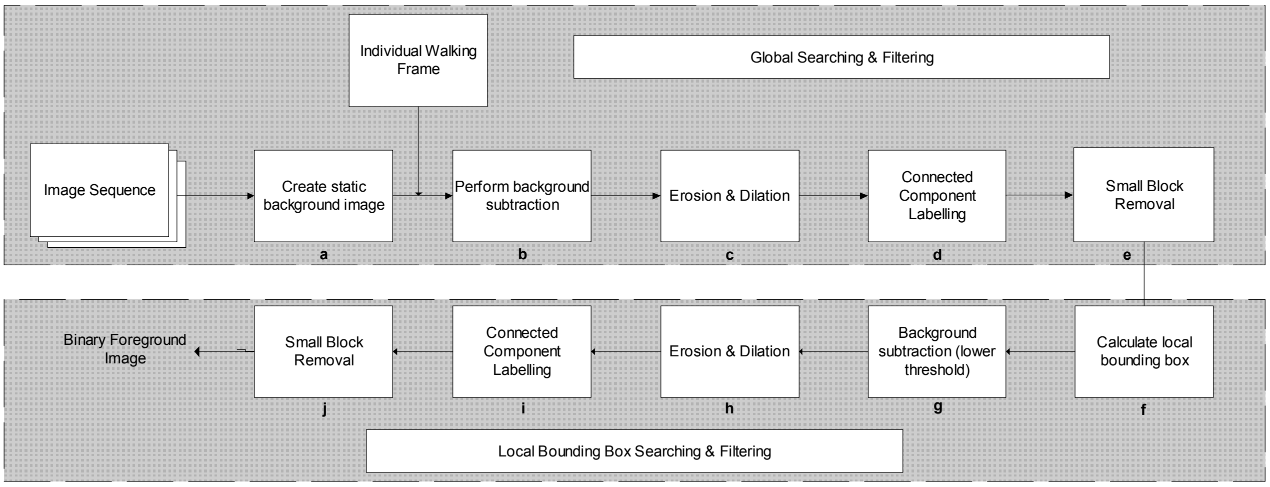

Individual Walking Frame
Global Searching & Filtering
Image Sequence
Create static background image
Perform background subtraction
Erosion & Dilation
Connected Component Labelling
Small Block Removal
a
b
c
d
e
Binary Foreground Image
Small Block Removal
Connected Component Labelling
Erosion & Dilation
Background subtraction (lower threshold)
Calculate local bounding box
j
i
h
g
f
Local Bounding Box Searching & Filtering

Figure 5.3

The stages are detailed below:

a) In order to perform background subtraction to extract the foreground regions of the image, a static background image much be available. The gait dataset used in this project did not provide any frames without moving foreground objects, so these background images were automatically generated by a function developed for the project.

b)Background subtraction between the current image and background image provides a foreground map, but this has a large amount of noise pepper noise over the picture and is not suitable to use for model fitting.

c) Erosion is performed on the subtracted image to remove small amount of noise which have been introduced to the binary image map. It is important to carry out dilation after erosion otherwise this will make small amounts of noise into larger regions

d)Component labelling gives each blob on the foreground image map a unique label, a list is produced with all of the labels in the binary image map and the corresponding number of pixels which have this label, the list is sorted in descending order of blob size.

Stages e and j remove labelled blobs from the foreground region based upon three main criteria, relative and absolute size and the distance of the blob from the centroid of the main blob. Both the physical size of the blob and its size compared to other blobs in the image are taken into consideration:

```
if(listPosition(label_x) > listThreshold || blobSize(label_x) < blobSizeThreshold ||
   distanceFromCentroid(label_x) > distanceThreshold ){
      remove all pixels with label_x
}

where:
  listPosition(x) – gives the position of the blob in the list of ordered blobs
  blobSize(x) – returns the number of pixels associated with label x
```

After a number of tests, it was found to be unrealistic to be able to define a cut off threshold for the position of a label in the ordered list. This was because sometimes even labels appearing near the bottom of the list, represented quite large areas of the human foreground region. Therefore just the blobSizeThreshold and distance was used as a criterion for inclusion of a blob in the foreground region. This blobSizeThreshold value itself is dependent on the quality of the images and the amount of noise present in the data capture. For the dataset used in this project consisting of frames of dimension 320x160 pixels, a threshold of 10 pixels was a satisfactory cut off point.

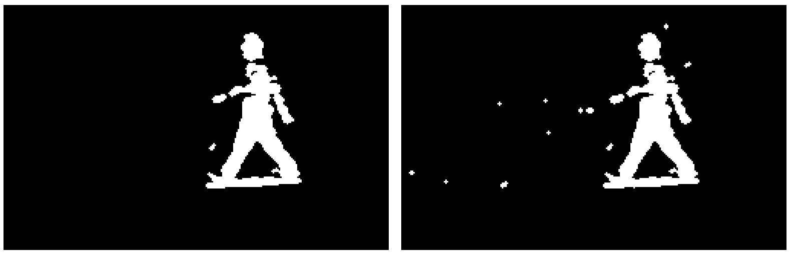


Figure 5.4 – blob size thresholding

Figure 5.4 shows how this method removes small blobs and helps derive a tighter minimum bounding box around the subject.

5.6 Gait Data Set

The data set (video sequences) used in this project came from a previous experiment by Boyd & Little [21] and has been used as the data set for many other previous gait experiements [19,18]. The gait laboratory setup for this data can be seen in the diagram below:

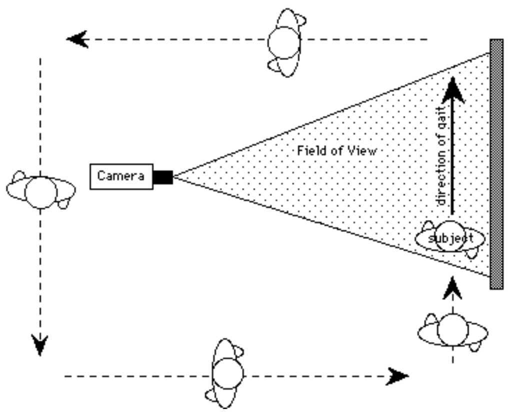


Figure 5.5 - Courtesy of Boyd & Little

A static camera was positioned, facing towards a static background. Each gait subject walked in front of the camera several times, during a 15 minute time period. There are six different subjects and each subject was recorded seven times, giving a total of forty-two video sequences. The video sequences were initially digitised in 24-bit colour, at a resolution of 640x480 pixels. They were then cropped to a resolution of 320x160 and converted to greyscale images.

It must be noted that this video was of very good quality and this helped in the segmentation process. The rest of the chapter is dedicated to the methods used to find the various body parts.

6. Semi Automatic Model Fitting

6.1 Aim

The aim of model fitting is to match the model of the human to the moving video sequence. When this is done, gait signatures can be automatically derived from the fitted model to be used in the recognition process. This chapter describes the methods used for tracking various parts of the body, such as the head, torso and legs. At the time of writing the head and torso was being track successfully, but the leg tracking was not very accurate. This is why the term semi automatic model fitting is used because if any of the body parts is mismatched in any frame there is always the option to select the joint positions manually, using the mouse.

6.2 Assumptions & Limitations

There were a number of simplifying assumptions which were made in order for the following methods to work successfully:

- Only one person in the video footage at any one time
- The camera lens direction is always perpendicular to the direction of the person walking, therefore we always have a side view of the individual. This means that we can assume that the length of the parts of the body remain the same throughout the video footage, otherwise if the person could walk at any angle, then the parts of the body would seem to change size depending on the perspective you were viewing from.
- Static camera – the camera should always be in the same position throughout the whole video sequence.

6.3 Human Model Design

With the model used for recognition, there are two main parts to it, which help make up the tracking, the physical representation of the model and the semantic representation.

6.3.1 Human Model Representation

As mention in section two, there are a number of ways in which a model of the human body can be represented. In this project a blob representation was chosen to model the human body. This was to allow the model parts to correlate the segments of the body against the binary image that was produced during segmentation. An example of the model used for model fitting is shown in figure 6.1

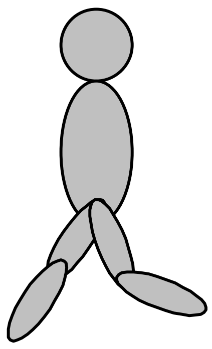


Figure 6.1 – six segment human model

There are a number of kinematical constraints we can imply from the structure of the human body which enable us to cut down the possible search space which must be searched at each stage in the model fitting:

1. ***Body part dimensions*** – the size of the all the body parts must remain constant throughout the data capture process. This is a fair assumption because of the simplifications which were made in the data capture process. Namely that all the subjects walk parallel to the camera. If they walked at various angles to the camera then because of projection from 3D to 3D space the length of their limbs would appear to change, but this is not the case in this project.
2. ***Constant velocity*** – we can assume that the subjects in the video, walk at a fairly constant velocity. From this we can estimate this velocity and use this for narrowing down the search area.
3. ***Joint dynamics*** – other constraints that we can make based on the human body, are that the lower legs are limited in their freedom by the knee joint. It is not possible for the lower section of the leg to bend past 180 degrees with the top half of the leg. This constraint help to reduce the possible search space as well as providing a more accurate model fitting result.

6.3.2 Human Relational Model

As mentioned in the background relational modelling is a useful tool for modelling real world structures, and also the constraints placed on the model help to cut down on the possible search spaces. A relational model is used in this modelling approach which uses the following semantic net:

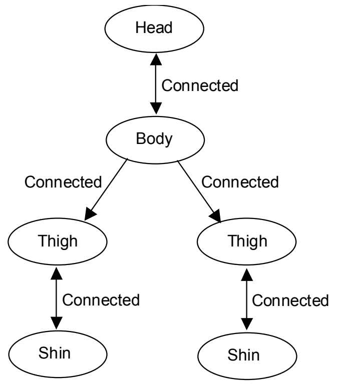


Figure 6.2 - Semantic net of human form

The relations between the various body parts have the following meaning:

- ***Head connected-to Torso***: The top of the body is touching the bottom of the circle representing the head, they do not overlap at all.

- ***Thigh connected-to Torso***: The thigh overlaps the torso slightly, but the top of the thigh is positioned in the middle of the torso.
- Shin connected-to Thigh: The shin segment slightly overlaps the bottom of the thigh, as both are jointed together at the knee joint.

6.4 Search Space Estimation

After segmentation has been performed on an image, the class returns a bounding box representing the search space inside which the person is located. In order to speed up the execution time, we need to reduce the size of the search space, because the larger the search space the more processing has to be carried out. In this project the bounding box around the binary silhouette of the person is split into four distinct areas, as in figure 6.3 below:

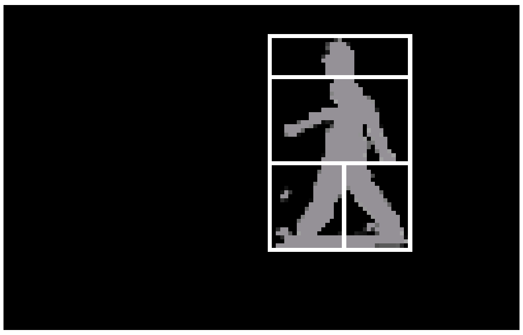


Figure 6.3 – search space reduction

The four main areas are:

1. Head – top 25% of the boundingbox
2. Torso – middle section of the bounding box.
3. Front leg – bottom 50% of box, to the left hand side of the box.
4. Back leg – as above, apart from on the right hand side of the box.

6.5 Body Dimension Estimation

Another point to note about the bounding box is that all of the dimensions of the model are derived as ratios of the height of the bounding box. In order to ensure that they are misjudged due to a bounding box being too large, possibly due to noise, the height of the bounding box is recorded for n frames, then the median height of the bounding box is taken to be the fixed height of the bounding box. The median height is used rather than the mean height, because the median height is not as affected by large values as the mean value is.

6.6 Head Tracking

From the segmentation process we should have an initial search space which we can be fairly confident encloses the moving subject. As shown above in figure 6.3 the search space of the head can be narrowed down to a sub-region of the minimum bounding box. To locate the position of the head we can calculate the amount of coverage of the head, overlapping with foreground pixels. The region with the best coverage for the circle will be allocated as the position of the head. The size of the head is estimated from the bounding box height.

In order to calculate the correlation between the circle, which represents the head and the foreground pixels, the following formula is used:

$$E(x_c, y_c, r) = \sum_{x,y \in Bhead} e(x_c, y_c, r, x, y)$$

$$where \qquad e(x_c, y_c, r, x, y) = \begin{cases} 1, & if \quad (x - x_c)^2 + (y - y_c)^2 <= r \\ o, & otherwise \end{cases} \qquad (6.1)$$

Noting that x,y are only foreground pixels inside the head bounding box. Each pixel inside the reduced head search space is selected as being x_c,y_c and the best match position is recorded as being the position of the head.

6.7 Torso

The correlation of the torso compared the search area of the torso uses the same type of formula as (6.1) apart from the fact that now we are considering an ellipse orientated around the y-axis:

$$E(x_c, y_c, a, b) = \sum_{x,y \in Btorso} e(x_c, y_c, a, b, x, y)$$

$$where \qquad e(x_c, y_c, a, b, x, y) = \begin{cases} 1, & if \quad \left(\frac{x - x_c}{a}\right)^2 + \left(\frac{y - y_c}{b}\right)^2 <= 1 \\ o, & otherwise \end{cases} \qquad (6.2)$$

6.8 Head Body Location

Using the idea that the body should be connected to the bottom of the head, as described in the constraints for the relation model, we do not let the head and body search for the best positions they can find independently, but instead they work together and find the best combined coverage with the body located directly below the head. The body is free to move along the x axis for a small range either side of the centre x point of the head to try to find the best position.

The general pseudo code for locating the position of the head and the body is shown below:

```
for each possible location of the head
        calculate the coverage of the head
        for each possible constraint body position
                calculate body coverage
                if body coverage + head coverage > best_combined_total
                        update best coverage
                        make not of the location of the head and the body
                endif
        endfor
endfor

draw the head and body at the position where the best combined
coverage was obtained
```

6.9 Legs

The legs are the most difficult part of the model to track with non-linear motion and the problems of self occlusion. This section aims to explain how the legs were modelled in the program.

There are three main constraints placed on the movement of the leg:

1. The value of β (the angle of incline of the shine) must always be less than or equal to α (the angle of incline of the thigh.
2. The angle of α can never be greater than ±50 degrees. This limits the thigh from being attracted to a position which is obviously not probable due to noise or incorrect tracking parameters. The value of 50 degrees was acertained after perfoming a number of test on the various subjects to find out what the maximum amplitude of the leg expansion was.
3. The angle of incline of the front thigh should be exactly opposite to the angle of incline of the back thigh, i.e. angleThigh1 = -angleThigh2. This forces the legs to move together and not wildly flap about.

Figure 6.4 below shows a snapshot of the leg with angles alpha and beta. Zero degrees is parallel with the y-axis.

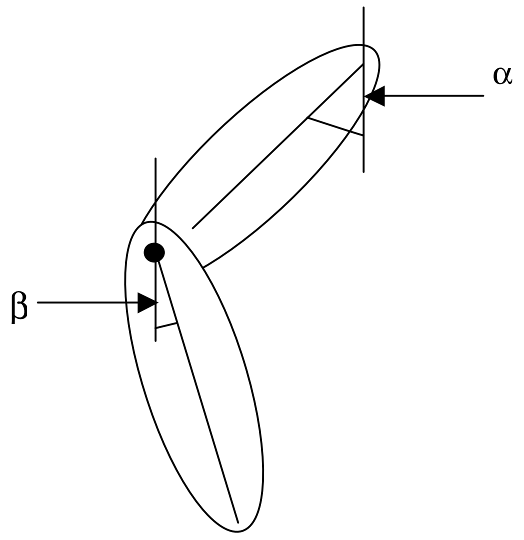


Figure 6.4 – snapshot of leg with angles

6.9.1 Initial Pose Location

Initially on the first frame of video, after the initialisation period where the dimensions of the bounding box are calculated, we need to find the best pose for the leg, but we need some sort of starting position. This is done using moments (6.3) to find the angle of incline of the longest edge of the bounding box around the leg in either the front or back box in the bounding rectangle. The formula for finding the angle is now shown (6.4):

$$\mu_{p,q} = \sum_{x}^{N}\sum_{y}^{M} x^p y^q f(x,y) \tag{6.3}$$

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right) \tag{6.4}$$

This angle is found for the first and second thigh and then the angle of the shins are initially given the same angle as the thigh, so the legs start out being fully stretched. We can then apply the more advanced searching to refine the position of the legs into the point of best fit.

6.9.2 Leg Position Tracking

In order to track the position of each part of the leg, we first consider how to track one leg and then generalie this for two legs.

The fist step is to work out the coverage of the foreground pixels by the model of the legs at time t and t-1. The legs can be though of as two sections (see figure 6.4) and so we have two degrees of freedom in the leg, angles alpha and beta. We can define a function which will calculte the coverage of one leg, for both sections of the leg when at angle alpha and beta (6.5):

$$E(\alpha, \beta) \tag{6.5}$$

We can then find the partial differential of the equation (6.6), by first just changing the alpha value, and then just changing the beta value. This will give us the rate of change of coverage of the leg due to changes in alpha and beta.

$$\frac{dE}{d\alpha} \qquad \frac{dE}{d\beta} \tag{6.6}$$

These values can be used to calculate the gradient of the function:

$$\nabla\left(\frac{dE}{d\alpha}, \frac{dE}{d\beta}\right) \tag{6.7}$$

From these gradients we can work out the value of the two angles alpha and beta at time t+1:

$$\alpha_{t+1} = \alpha_t + \frac{dE}{d\alpha}\delta \tag{6.8}$$

$$\beta_{t+1} = \beta_t + \frac{dE}{d\beta}\delta \tag{6.9}$$

We can apply these steps iteratively changing the values of alpha and beta using formula (6.9) until there is not more increase in the function $E(\alpha, \beta)$, at this point the leg has moved into one of the best positions and from that point on it will start to decrease in coverage.

We can now apply this for both legs, and the model legs should stay inside the foregound regions as the legs move.

The total coverage of the legs is then taken into account when trying to find the best position for the legs, combining the coverage values for the front and back, shin and thighs. when this has a maximum value, then we know it is the best place for the legs to match the foreground region.

7. Gait Signature Extraction

7.1 Aim

Once the feature points have been extracted from the video sequence, in the case of this project the feature points are the position of the joints (eight point model discussed in section 4.4.2), a gait signature has to be derived which will be able to uniquely identify, or at least provide a high rate of correct recognition for the gait subject. This chapter describes in detail the methods used in this project to produce a gait feature vector for the different gait subjects.

7.2 Signature Design & Implementation

The data which can be used for deriving a gait signature is the position of each joint in the human model, over n frames of video. A single captured instance of the configuration of the points can be thought of as a vector P, such that:

$$P = [p_1, p_2, \ \dots \ , p_n] \qquad \qquad where \ \ p_i = [x_i, y_i] \tag{7.1}$$

From the manual or automatic feature extraction a vector of human poses is now the data which can be used to generate the signature. The more prominent of these features use a mixture of temporal and spatial information:

- ***Absolute joint positions*** – the normalised physical position of each joint in each frame can be used as a basis for a gait signature.
- ***Limb angles, angular velocities*** – the angle of each limb at over a series of frames can be used. Also the angular velocity of selected limbs can be used to derive the signature.
- ***Self similarity plots*** – as described earlier in section 2.7, self similarity plots can be used to describe the correlation between a variable over a period of time with itself. With gait self similarity plots offer a good way to encode the phase and frequency of the subjects walking pattern.

The rest of this section is dedicated to explaining each of the processes above in more detail.

7.2.1 Joint Position Signature

One possible signature for the gait of an individual is to use the position of the captured joints, over all of the captured video sequence. The pattern of positions over time may be unique enough to act as a gait signature. The ChumanSequence data type which stores all of the joint positions over all of the frames, has the functionality to returns a feature matrix, where each column of the matrix is equaivalent to the positions of the joints in one frame. Hence the number of columns in the feature matrix is equivalent to the number of captured frames.

$$\begin{bmatrix} p_{1x}(t) & p_{1x}(t+1) & & p_{1x}(t+N) \\ p_{1y}(t) & p_{1y}(t+1) & & p_{1y}(t+N) \\ p_{2x}(t) & p_{2x}(t+1) & & p_{2x}(t+N) \\ p_{2y}(t) & p_{2y}(t+1) & \dots & p_{2y}(t+N) \\ \vdots & \vdots & & \vdots \\ p_{8x}(t) & p_{8x}(t+1) & & p_{8x}(t+N) \\ p_{8y}(t) & p_{8y}(t+1) & & p_{8y}(t+N) \end{bmatrix}$$

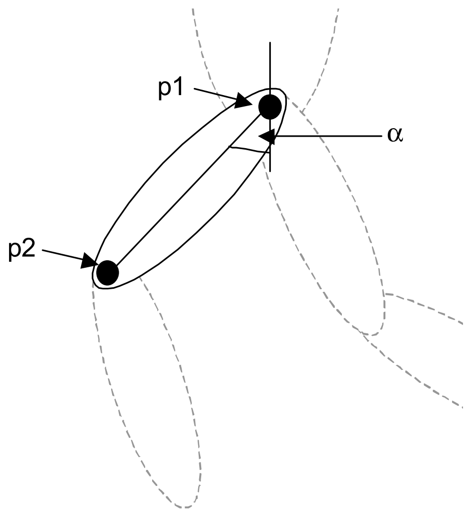
Figure 7.1 – layout of joint position feature matrix

Producing a matrix like this has the advantage that each individual person will be plotted in feature space as a cluster of points, allowing more advanced cluster recognition techniques to be performed on the data.

7.2.2 Limb Angle / Angular Velocity Signature

The CHumanSequence data type offers the functionality to extract a matrix from the sequence of frames consisting of either the angle of the limbs or the angular velocities over the number of captured frames. As with the joint position method shown above, the output is a matrix, where each column represents the angles of each limb in a single frame, and each element in the row is the angle for one of the limbs.

To calculate these values the end points of each limb should be known, as shown in figure 7.2.



$$\alpha = \tan^{-1}\left(\frac{p2_x - p1_x}{p2_y - p1_y}\right)$$

Figure 7.2 – Limb angle calculation

The reason for using the angle data is to provide a way of encoding the change in angle of each limb over a period of time. Producing a matrix of angular velocities of each joint is calculated the same way, the only difference is now we need to take into account the time difference between each frame to be able to calculate the rate in change of the angles.

7.2.3 Self Similarity Plots

Self similarity plots are a good way to extract the phase and fequency of the walking pattern of an individual. BenAdbendulker [18] used them successfully for recognising gait. He created a self similarity plot based on the pixel image of the foreground region. For this project a number of self similarity plots were created for each person based on the angles of various body parts over n frames. All in all there were five self similarity plots for each person, for the following parts of the body:

- Left Shin
- Left Thigh
- Right Shin
- Right Thigh

Initially a self similarity plot for the torso was also produced, but the small variation in the torso meant a large amount of error was present in the plot, making it not very powerful as a gait identifier.

7.2.3.1 Data Normalisation

Before generating the self similarity plots for each subject the data was normalised into a form which allowed non-bias comparison. In the case of the video sequences, all of the sequences were edited so that they started with each persons legs at maximum distance apart. This made sure that the plots which were produced, were not out of phase intially which may have affected the recognition results.

7.2.3.2 Creating Self Similarity Plot

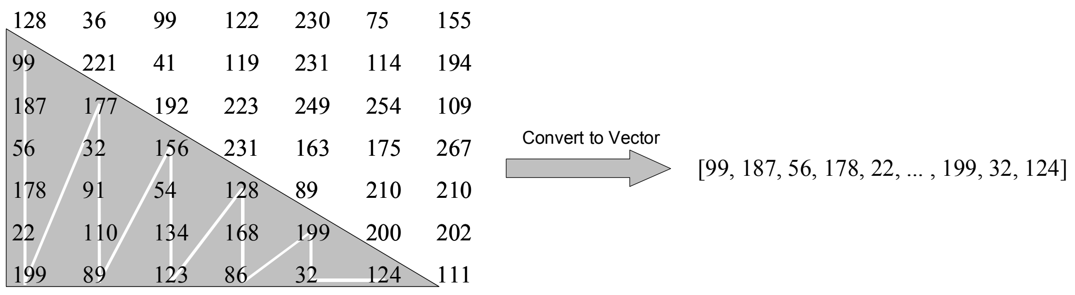
The self similarity plot for limb J_i was calculated using the following fomula:

$$S(J_i) = \min_{-d\alpha \le r \le d\alpha} \left| J_i(t_1) + r - J_i(t_2) \right| \tag{7.2}$$

The formula above is carried out on t_1 to t_n frames, which gives a measure of correlation for the desired feature, for instance J_i could be the angle of limb i, or possibly the angular velocity of the selected limbs. The value r is an error minimising term which is used to reduce the amount of error in the plot. The value of $d\alpha$ was set to 5 degrees.

7.2.3.3 Creating the feature vector representation

Once the self similarity matrix has been produced we can reduce the amount of information we need to use because of one of the features of self similarity plots, symmetry. Because the matrix is symmetrical then we can discard half of the matrix as this is redundant data, we can also leave out the values in the main diagonal because they are guaranteed to be always the same number because S(t,t) = 1, i.e. an item is always correlated with itself.



128	36	99	122	230	75	155
99	221	41	119	231	114	194
187	177	192	223	249	254	109
56	32	156	231	163	175	267
178	91	54	128	89	210	210
22	110	134	168	199	200	202
199	89	123	86	32	124	111

Figure 7.3 – converting matrix to vector

So instead of creating a vector of length n^2 items, it now has length:

$$length = \frac{n(n+1)}{2} - n \tag{7.3}$$

This will reduce the amount of computation needed for calculating the eigenvectors and eigenvalues of the matrix, whilst preserving the gait information about the person.

The final stage is to concatenate the self similarity feature vectors, produced above for each of the different body parts together to form one large vector which can be used as the gait signature of the individual. An example of the five self similarity plots which are generated for each query subject is shown in figure 7.4

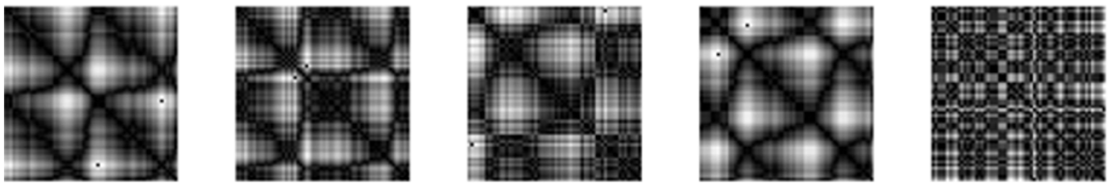


Figure 7.4 – Example of self similarity plot

8. Gait Signature Recognition

8.1 Aim

Once the gait signature has been extracted from the gait subjects, it needs to be further transformed in such a way that comparison and recognition between the different classes (i.e. people) is possible. Once this has been completed, classification methods have to be used on the data sets in order to be able to perform gait recognition. This project implements two main classification methods and a number of feature transformation operators.

8.2 Design

The whole recognition process is encapsulated mainly within one main class, CRecognition. This class provides the following functionality:

- Perform PCA on feature vectors
- Perform Canonical Analysis
- Project feature vectors to different feature space.
- K-Nearest Neighbour (KNN) classification
- Spatio Temporal Correlation based recognition
- Recognition Analysis

8.3 Gait Signature Feature Spaces

Comparing the extracted feature vectors in their primary extracted form would not lead to high rates of recognition, due to overlapping of different subject classes i.e. poor separation and also the data might have high dimensionality. In order to be able to achieve possible higher levels of recognition we need to be able to project the feature vectors into a different feature space which will provide the maximum amount of separation. As mentioned in the background there are two main ways of reducing the amount of data needed for comparison or separating classes of data, either in Eigenspace or Canonical space.

8.3.1 Eigenspace Transformation

In the program, once feature vectors have been extracted each vector is projected into Eigenspace in order to help aid the recognition rate. This is done by first calculating the eigenvectors from the covariance matrix of the extracted feature points. In this project all points in the model are appended together to form a vector then all instances are appended to form a feature matrix, see figure 7.1.

To project this feature matrix into Eigenspace need to multiply all of the eigenvectors with the feature matrix, see equation 2.9. By doing this, the PCA will preserve most of the variation in the data, but you need les information to be stored to do that, than with the original information.

8.4 Training Data Set

In order to be able to perform recognition there needs to be a set of data which the query can be compared against and classified. In this project the gait signatures of six different individual was recorded, each person had seven video sequences taken at different times (see section 5.6).

For the spatio temporal correlation method (see below 8.5.1) the training data is created by choosing randomly one sequence of each of the different people (a total of six subjects) and storing the details in the database XML file. Queries are then compared against this file.

With K-Nearest Neighbour classification (see section 8.5.2) the training database set is created by placing all of the captured gait signatures of all the video sequences apart from one sequence. The sequence which has been left out is then compared against this database. This is done for all video sequences and is know as the leave-one-out cross-validation method.

8.5 Recognition through classification

Once the feature vector representation of the gait signature has been extracted from the subject, and projected into a feature space which gives the best level of recognition, the next step is to perform the recognition by pattern classification. There are two different methods of classification used in the project depending on the method of gait feature extraction used. The first is spatio temporal correlation and the second is K-nearest neighbour with leave-one-out cross-validation to estimate the error rate.

8.5.1 Spatio Temporal Correlation

As mentioned previously in the chapter, there are two different types of signature produced for a person, a cluster or a point. For spatio temporal correlation we use cluster comparison to ascertain which training class the query subject should belong to.

The idea is that each corresponding temporal element of the class, in our case each point created from the same number frame in the video sequence, are compared and the distance calculated. So the point projection of person x, in frame i is compared against the projection point of person y in frame i. The total distance of all corresponding points is calculated:

$$d_i = \sum_{t=1}^{N_i} \left| c_i(t) - q(t) \right| \qquad (8.2)$$

As can be seen in figure 8.2 below, each element of the class cluster one is compared with the corresponding temporal instance of the other class, and the distance is calculated. The total distance between all corresponding class elements are summed and a measure of the distance of the two classes is calculated. In order to perform recognition, the query class is compared against all of the training classes and the separation distance based upon spatio temporal (as in figure 8.1) is calculated. The training class which has the smallest distance from the query cluster is chosen to be the class (i.e. person) which the query belongs to.

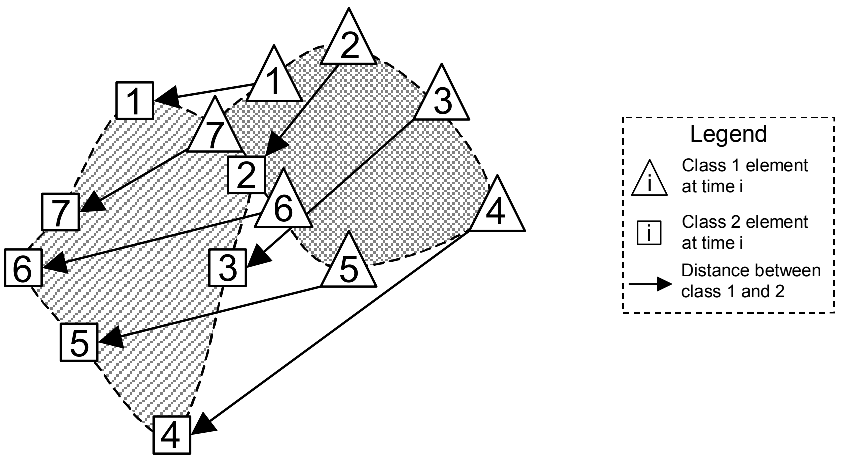

1
2
3
4
5
6
7
1
2
3
4
5
6
7
Legend
Class 1 element at time i
Class 2 element at time i
Distance between class 1 and 2

Figure 8.2 – Spatio Temporal Correlation

In the program, once all of the training classes have been compared against the query class, the cumulative distances are sorted in descending order, and the first class in the list is chosen as the best match.

This method of classification is used for recognition in the following different gait signatures:

- Joint Positions using Principle Component Analysis
- Limb Angles using Principle Component Analysis

8.5.2 K-Nearest Neighbour

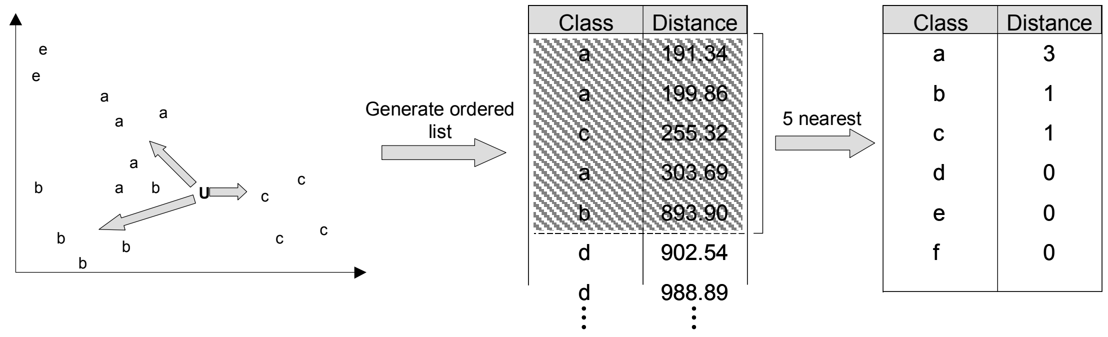
Another method of classifying classes is to use the K-Nearest Neighbour classification method. As described previously, this is used when you want to compare a single query feature point (compared to a cluster in spatio temporal correlation) to the training data set.

This method of classification is used in the program when performing recognition using the following feature extraction and recognition methods:

- Self Similarity Plot using Principle Component Analysis

Recognition of gait using the K-Nearest Neighbour is executed in the program by performing the following steps (see figure 8.3 below):

1. Compare the unknown query point against all known training points (all points have been previously projected into some feature space).
2. Generate an ordered list of class elements and their distance from the query point.
3. Extract the top k elements from the ordered classification list.
4. Count the number of occurrences of each class (i.e. person) in the k extracted points, the class with the most occurrences is chosen as the class which the unknown query point should belong to. This is known as the majority vote method, and it is quite resilient to noise in the training set.



Class	Distance
a	191.34
a	199.86
c	255.32
a	303.69
b	893.90
d	902.54
d	988.89
⋮	⋮

Class	Distance
a	3
b	1
c	1
d	0
e	0
f	0

Figure 8.3 – K-Nearest Neighbour classification

In the case that there is a tie for the class who has the majority vote, the summation of the distances of all the class elements is considered for the equal classes, the class which has the smallest total distance is named as the selected class.

9. 3D Graphics Engine

9.1 Aim

As part of the project, a 3D visualisation of the extracted human model was required. After looking at possible alternatives I decided that I did not have enough time to learn a completely new package and how to integrate it into the Visual C++ environment. I therefore implemented my own 3D graphics engine, which allows 3D models of the person to be projected on the application window.

9.2 Design

The graphics class provides needed to provide the following functionality:

- Display a set of 3D coordinates as a 2D projection on the viewport.
- Rotate the object around the X,Y,Z axis.
- Zoom the 3D object in and out.
- Display a title in the top of the projection window.
- Change the colour of vertices in the projection.

The overall conceptual software design of the graphics system is shown below:

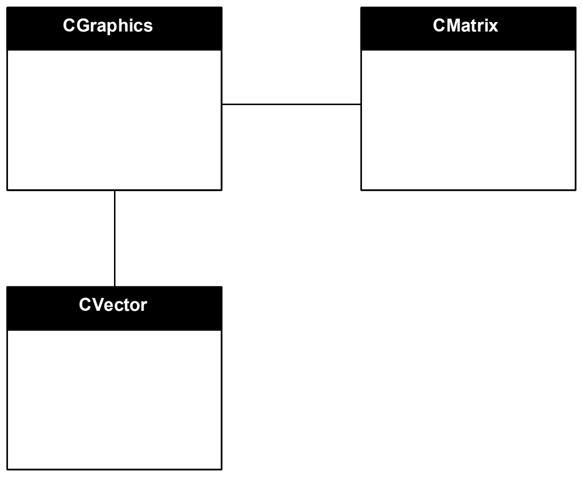


Figure 9.1 – Graphics engine design

The CGraphics class encapsulates all of the required visualisation functionality, and the CMatrix and CVector data types are used for storing the graphics data within the CGraphics object, as well as providing all the necessary functionality for matrix and vector calculations.

9.3 Projection Theory

A point in 3D space can be projected into 2D space in perspective projection, by the following relation:

$$x' = f\frac{x}{z} \quad (9.1)$$

$$y' = f\frac{y}{z} \quad (9.2)$$

Where x,y,z are the co-ordinates of the point in 3D space and x’ and y’ are the positions of the points after projection in the 2D space. The distance of the viewing plane from the camera origin is denoted by f. An example of this can be seen in figure X.X.

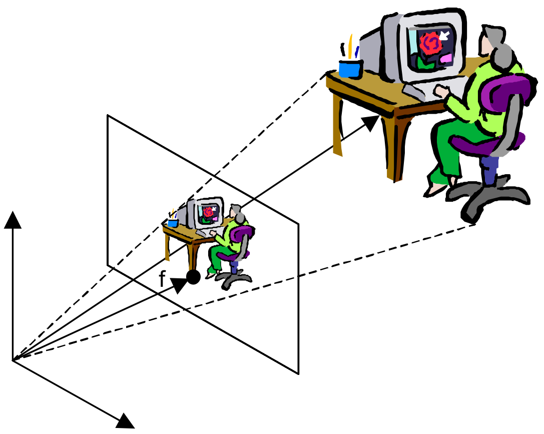


Figure 9.2 – projection from 3D to 2D

9.3.1 Rotation matrices

In order to rotate a point about the origin by a given angle θ around the X,Y or Z axis, the point must be multiplied by one of the following matrices:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (9.3)$$

9.4 Implementation

The viewport, which is displayed on the screen can be of any dimension and placed at any position on the screen. These parameters are passed to the constructor when the graphics object is created, and allow the CGraphics object to be very versatile.

9.4.1 Display Process

There are a number of steps performed in order to project the object inside the application viewport:

1. Create a CGraphics object, defining the size and position of the viewport on the application window.
2. Load the data into the graphics object, this is done by calling setGraphicsData(CMatrix). These co-ordinates are the 3D data points.

3. Convert all of the 3D points into 2D points using perspective projection, this is done internally within the CGraphics object.
4. Calculate the minimum bounding box that contains all of the 2D projected points.
5. Normalise all of the 2D projected points to fit within the viewport window defined by the user
6. Project the points on the display and join the points by line sections.

9.4.2 Data Normalisation

The normalisation of the points is necessary to ensure that all of the projected points are always displayed inside the area defined by the user on the application window. The points are normalised by finding the ratio of the height of the minimum bounding box of the 2D projected points and the height of the user defined graphics area. The height to width ratio of the projected 2D points is maintained in the process to make sure distortion of the projected points does not take place.

9.4.3 Object Rotation

There is an mouse event handler which is activated when the user right clicks on the graphics window. Whilst the right mouse button is held down, when the user moves the mouse left or right the 3D projected model will move clockwise or anticlockwise. This is done by rotating all the points using the rotation matrices shown above and then refreshing the screen. Examples of the object being rotated can be seen in figure 9.3.

The points are rotated around an arbitary point, the point chosen in this project was the cox of the human (at the bottom of the spine), this ensures that the human is rotated inside the viewport area. Because we are not rotating around the origin, but around an arbitrary point, we need to translate the points in relation to the origin and then rotate the points:

1. Subtract the rotation point from all of the data points, this has the effect of translating all the points in relation to the origin
2. Rotate the points using one of the rotation matrices defined above
3. Add the rotation point to the rotated points to translate them back to the correct position.

This process can be defined mathematically by (rotating around the Y-axis):

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_r \\ 0 & 1 & 0 & y_r \\ 0 & 0 & 1 & z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_r \\ 0 & 1 & 0 & -y_r \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (9.4)$$

Figure 9.3 below shows the output of the CGraphics class being rotated. Not that one of the joints is a different colour to the rest. The user can specify the joint colours of any joint in the image.

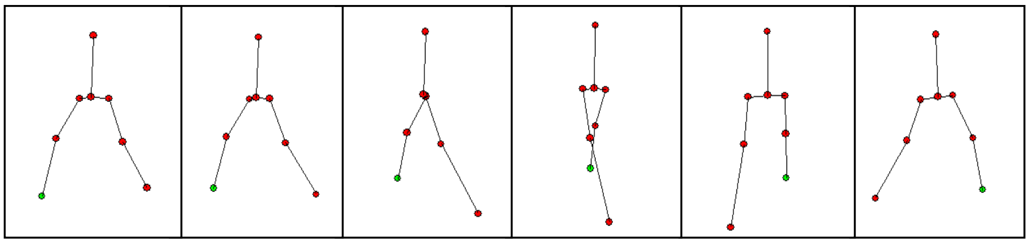


Figure 9.3 – human skeleton rotation

Screen shots of the graphics window in action in the final application can be seen in the user interface design, section 10.

10. User Application

10.1 Aim

As part of the project a graphical user interface is provided, that acts as an interface between all of the functionality of the system and the actions performed by the user. The interface should be clear, concise and intuitive to the user.

10.2 Main Interface

The main window in which all of the operations are performed is shown in figure 10.1 below.

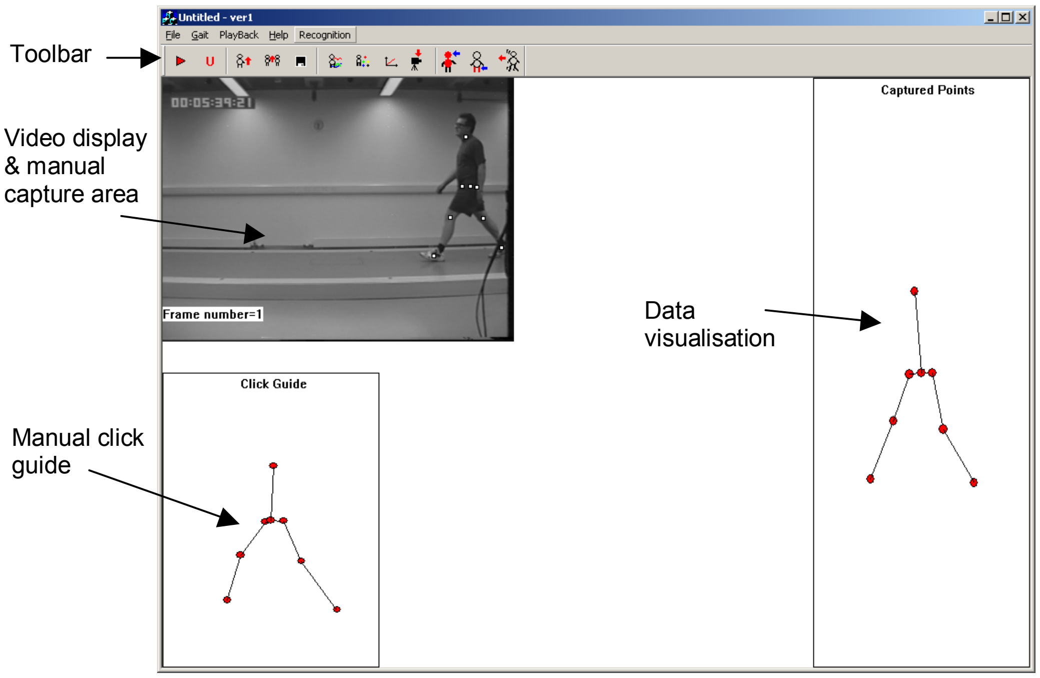


Figure 10.1 – main program interface

The window has a number of main components and sections that the user should be aware of when performing operations in the program, these are all explained in more detail in the following sections.

10.2.1 Gait Toolbar

The toolbar at the top of the window provides a fast way of executing the various operations available to the user. Figure 10.2 is a close up of the toolbar icons.

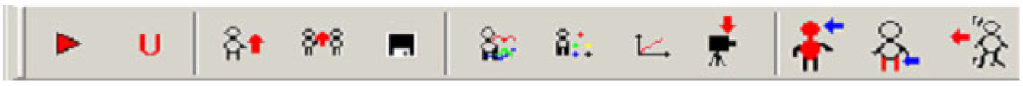


Figure 10.2 – gait toolbar

Each of the icons on the toolbar are explained in the order they appear on the toolbar from left to right:

1. ***Next Button*** – this will move onto the next frame in the video sequence.
2. ***Undo*** – will undo the action of manually selecting one of the points.
3. ***Load Query*** - Loads a query signature from a file for use in the application
4. ***Load Data Store*** – Loads a file which will be used to compare against the query.
5. ***Save captured gait signature*** – use this when a new gait signature has been captured. The gait data will be stored in an XML file.
6. ***Joint Trails*** – Displays all of the points which were selected over the video sequence, an example of this can be seen in section 10.4
7. ***Last Captured Joints*** – shows the position of the captured joints one frame before the current frame.
8. ***PCA Gait Visualisation*** – opens a dialog box where the user can experiment with the effects of principle components on the relative positions of the joints.
9. ***Load Image*** – this icon is used to load the first image in a sequence of images. The image that is selected must be of Bitmap file type.
10. ***Extract head & body*** – clicking on this button will start/stop the automatic tracking of the head and the body of the human.
11. ***Extract legs*** – clicking on this button will start/stop the automatic tracking of the legs.
12. ***Animate*** – instead of the user having to manually click on the next image icon, the animate button will automatically load the images into the application after a finite period of time.

10.2.2 Video display & manual capture area

This area of the application displays the current video sequence. When the user loads a new image to display, the screen is updated to display the new image in this region. Also once the image has loaded, the user can click on the image with the mouse and manually select where the joints are on the human body through the video sequence. This is also where the model is mapped on top of the video sequence and is displayed to the user.

10.2.3 Manual click guide

The manual click guide is used when the user is capturing data manually through the video view display area. The joint which should be clicked on by the mouse is displayed in a different colour (green) to the rest of the joints. This is useful when the user is entering data and forgets which was the last point they clicked on. The user can right click on the window and by moving the mouse, the skeletal model will rotate in the

10.2.4 Data visualisation

This area of the application displays the captured joint points of the human in 3D form, each time the user captures a new frame of information this area will automatically update to show the captured pose in 3D. As with the manual click guide area, the user can right click inside the visualisation window and by dragging the mouse, left or right, the model will rotate around the y axis.

10.3 Gait application operations

There are a number of operations that the user can carry out using the interface. These are detailed in the subsequent sections.

10.3.1 Loading a bitmap image

The user can load a bitmap to start the capture or playback process by selecting the load bitmap button. Once the user has selected the starting image they must replace the last numbers in the file name by %d, for example, the file path and name c:\gait\pics\man0001.jbmp is transformed to c:\gait\pics\man0%d%d%d.bmp. This will allow the program to iterate over many files in the sequence, because internally it replaces the %d letters with a counter value.

10.3.2 Gait PCA Visualisation

One way to visualise the effect of PCA on the captured human joint positions is by looking at the Gait PCA visualisation window. This can be done by selecting "PCA Visualise" from the gait menu drop down option, or clicking on the visualisation button on the main toolbar. This action will open the window shown in figure 10.3 below:

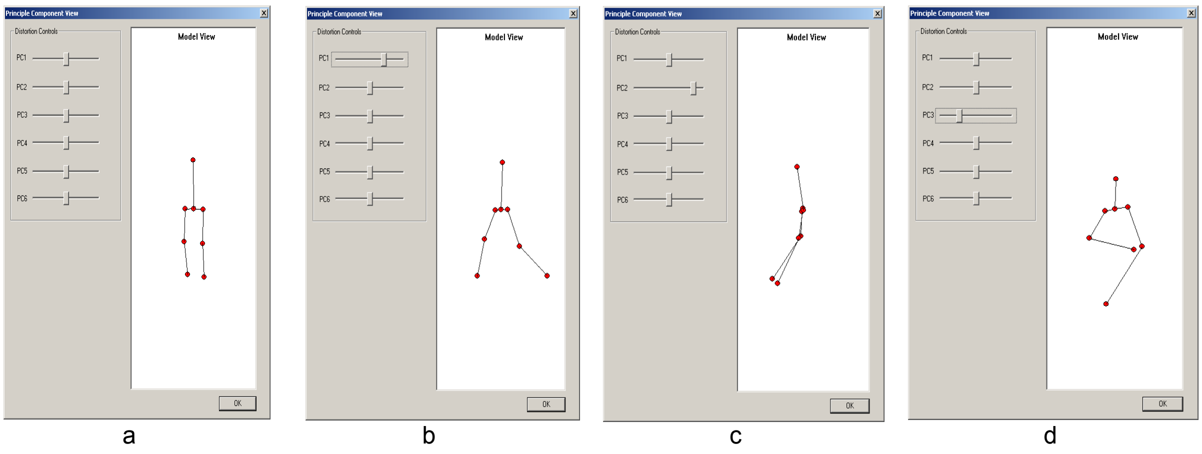


Figure 10.3 – effects of principle components on joint positions

Figure 10.3 shows four different instances of the window. The sliders on the left hand side each correspond to a single principle component, slider one controls the affect of the first principle component on the model positions, slider two controls the second principle component and so on, The further the slider is away from its stable position in the centre of the slider, the more effect the principle component has on the model.

10.3.3 Performing recognition

In order to compare the gait of an individual to the gait signatures stored in the database the user has to perform the following steps:

1. Load a query file into the application
2. Load a data store file which contains all of the training data, the query will be compared against all training classes in this file.
3. Click on the "execute recognition" button on the task bar.

The results of the recognition process will be displayed in the centre of the window at the bottom of the screen (see figure 10.4).

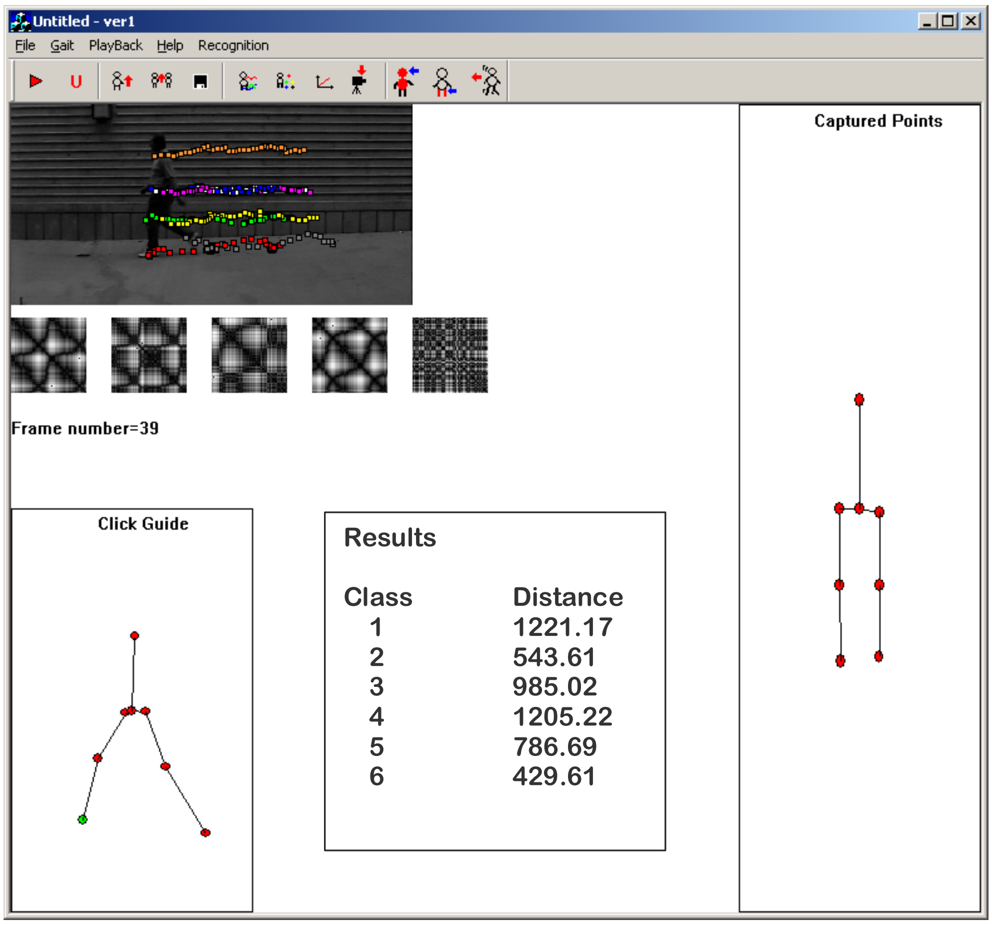
10.3.4 Saving captured gait data

When a new sequence has been captured in the application, it is useful to be able to save the data for use in comparisons later. To save the data, click on the save icon in the toolbar and a dialog prompt will be displayed asking where the file should be saved, there are then two options:

1. Save data in a new file – this is useful for creating query files
2. Save data in existing file – this will append the new gait data as a new element in the existing file. This technique can be used to create database files which store the details about many individuals.

Figure 10.4 below shows a number of the features of the program in practise. The user has loaded a bitmap sequence and has selected to turn on the joint trails options (these are the small dots trailing the person). The trails help us to visualise where the points were selected previously.

Also below the image, you can see four self similarity plots for the person. The self similarity plots are of the following parts of the body, front shin, front thigh, back thigh and back shin.



Results

Class	Distance
1	1221.17
2	543.61
3	985.02
4	1205.22
5	786.69
6	429.61

Figure 10.4 – Program display in action

The results pane is also visible in the application window, this shows the results of comparing the query subject against a number of previously captured gait signatures. As you can see, in this example there were six people in the training set, and the man in the image is closest to class 2. This means that he will be classified as being equal to person two.

11. Evaluation

This chapter goes through all of the gait recognition main sections, segmentation, model fitting and recognition and explains the outcome of each of the individual sections, then a holistic evaluation of the whole process is given.

11.1 Segmentation Results

The segmentation results can be broken down into two main parts, the foreground extraction, where the person moving across the background image was extracted and the background generation, which is needed for image subtraction.

11.1.1 Foreground Extraction

From the given image dataset, the segmentation process described in section x generally provided good segmentation results, with a number of different people and video sequences. Figure 11.1 below shows the binary image map which is produced from the original image, as you can see there are still small blobs of noise present, but these will hopefully not cause any adverse affects as they are relatively small.

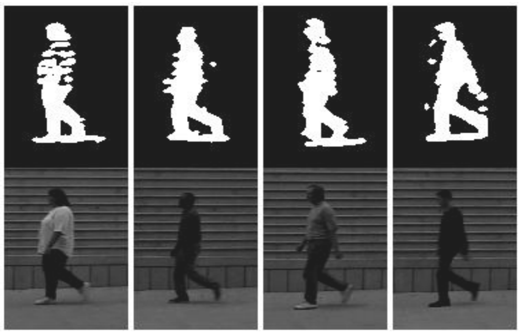


Figure 11.1

One of the main problems is the shadow cast by the person as they walk, due to the change in pixel intensity it causes. The shadow is incorrectly classified as a foreground region, which does affect the width of the minimum bounding box around the person. Because the images are in greyscale format, shadow removal is more difficult that if the images were in colour, further extension need to be introduced to this process to perform pre-processing on the image in order to remove the shadow.

Another problems was that some of the subjects had clothing that was quite close in intensity to the background. The effect of this was that part of person was classified as the background. An example of this can be seen in the first column of figure 11.1 and below in figure 11.2.

One way to reduce this effect is to lower the value of the threshold used in subtraction, but this then makes small noise more prominent, a trade off between the two effects has to be accepted. Figure 2.2 shows the result of the segmentation process with different levels of thresholding. All of the images are of the same person, the first row is using a thresholding of twenty, the second row uses a threshold value of ten and the last is a threshold value of five.

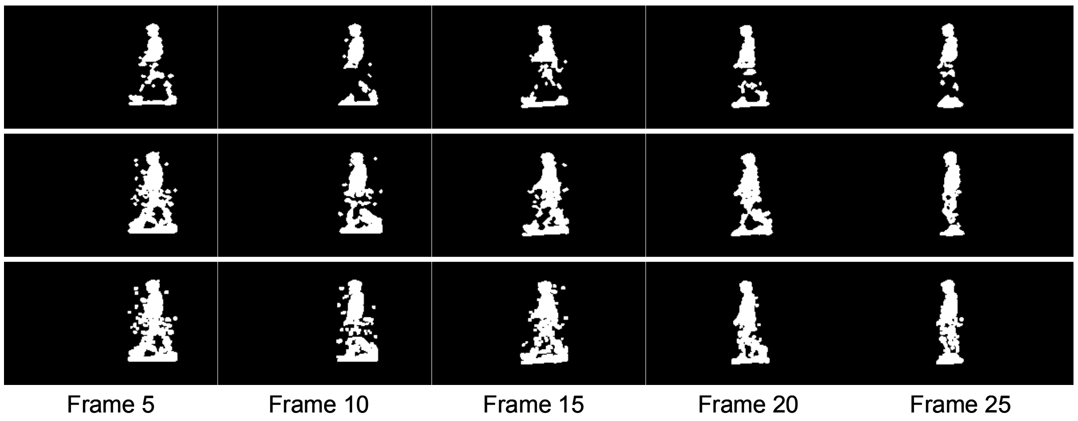


Figure 11.2 – segmentation at various threshold levels

As you can see the higher the value of the threshold used during subtraction the more the subject is classified as part of the background instead of the foreground. Lowering the thresholding value reduces that effect, but introduces more noise into the binary image.

It must also be noted that the quality of the video sequences was also very good, had there been more noise in the images, or differing lighting conditions then the results of segmentation would have been poorer and a different background model, which can take this into account.

Another example of noise was in the initial dataset I had started to use. The images had been taken and the lighting used a florescent light, the flickering of this light caused large amounts of noise to be produced using this subtraction technique and the background model I had used. This can be seen in figure 11.3.

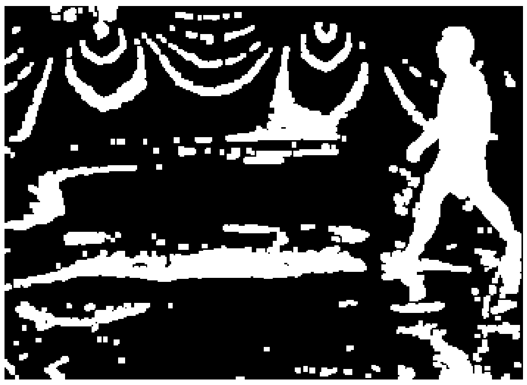


Figure 11.3 – noise in binary image

A better background model could be used for this data set with temporal and range distance included in the model to help reduce the effect of the lighting noise.

11.1.2 Background Generation

The background was generated from a number of frames of the chosen video sequence. Each pixels intensity was recorded over the period of N frames and then the median value was taken to be the background intensity.

This method had good results, with a good representation of the background being produced from each video sequence used in the project. The only complication was choosing the number of frames which would be used to make up the model. As can be seen in figure 11.4, a number of different length video sequences were used to generate the background. The shorter the length, the more the foreground person was included as part of the background. Also the speed of the person affected how many frames had to be used, the slower the person walked the more frames were necessary to remove the person from the background.

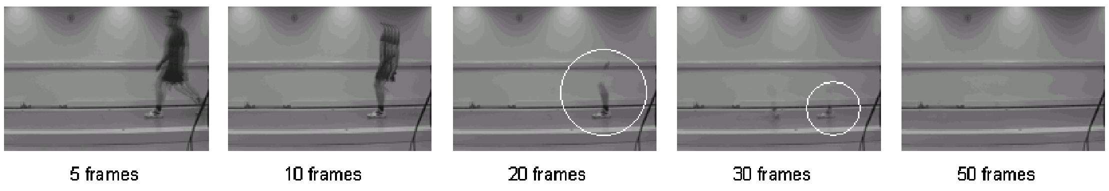


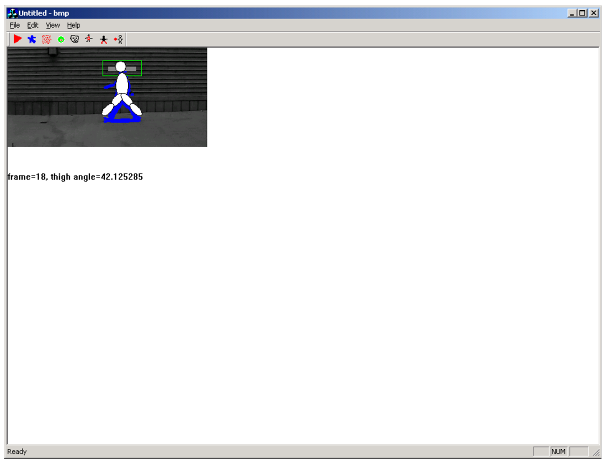
Figure 11.4 – background frame generation

In figure 11.4 we can see that below 30 frames some of the subject is still evident in the background (as circled in 5, 10, 20 and 30 frames). This is because not sufficient readings have been taken to differentiate the foreground and background using the median calculation.

After a number of tests on the data set, 50 frames was found to be an adequate value to be used for background generation.

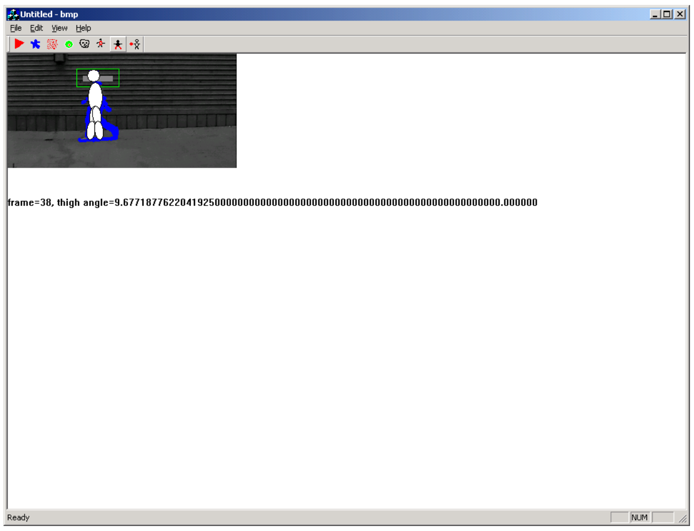
11.3 Model Fitting

The model fitting process was successful in tracking the head and the body but the legs were not so successful. Below are a few screen shots of the application working.



```
frame=18, thigh angle=42.125285
```

Below is a picture of the process not working so well, and missing the legs:



```
frame=38, thigh angle=9.67718776220419250000000000000000000000000000000000000000000000000000.000000
```

11.4 Gait Signature Extraction

The purpose of gait signature extraction is to create a signature which will be unique to a persons gait. A number of different types of signatures were proposed, and this section gives examples of what types of output were generated from the signature creation process.

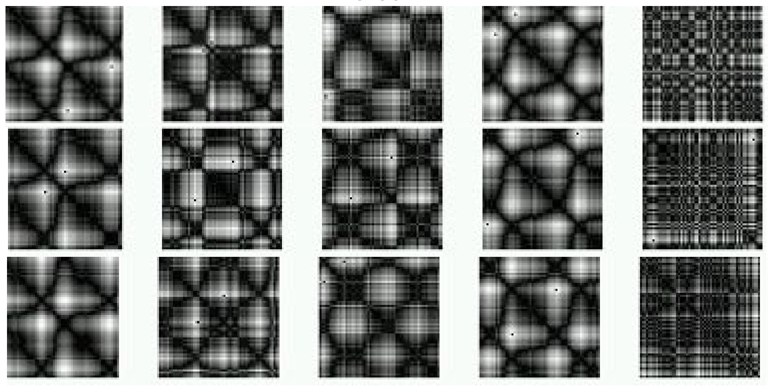
11.4.1 Self Similarity Plots

As explained previously self similarity plots give a measure of correlation of a given variable over time. The aim in this project was to evaluate the effectiveness of the self similarity plot as a gait signature, by creating the self similarity plot from the angles of various limbs on the subjects body, namely:

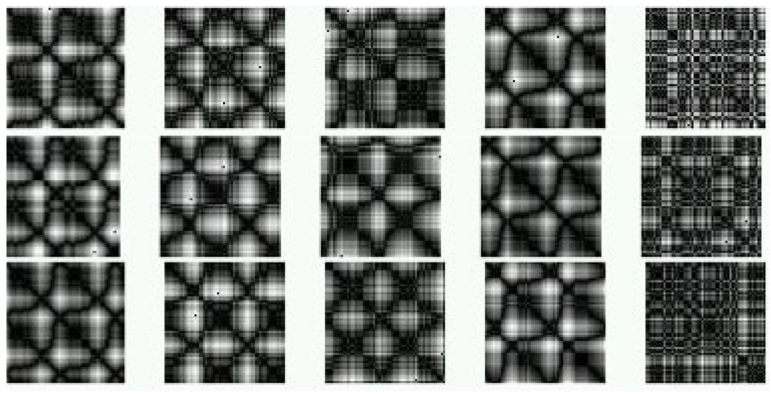
- Front shin
- Back shin
- Front thigh
- Back thigh
- Torso

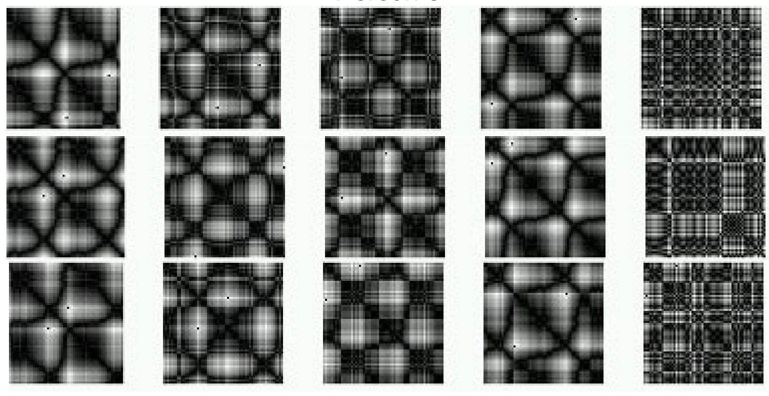
After initially creating the self similarity plots, it could be seen that the variation in the angles of the torso was very small and seemed to be quite random, as it change from sequence to sequence quite dramatically for the same person, possibly due to error in the data acquisition stage. The self similaity plots for all six subjects, for the first three sequences are shown below, column one is the front shin, column two is the front thigh, column three is the back thigh, column four is the back shin and column five is the torso. The angle of all these limbs were the variable used to create the self similarity plot.

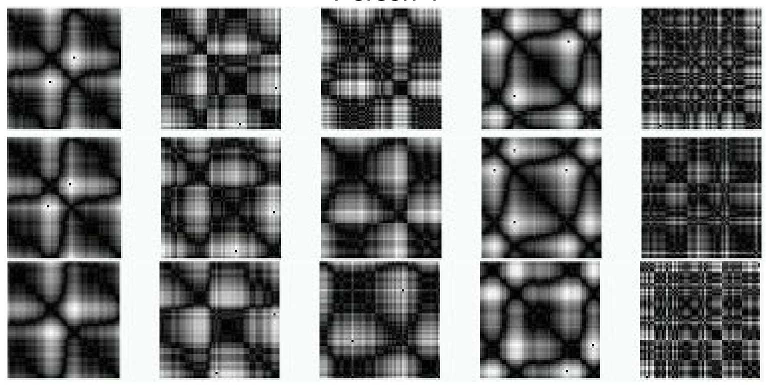
Person 1

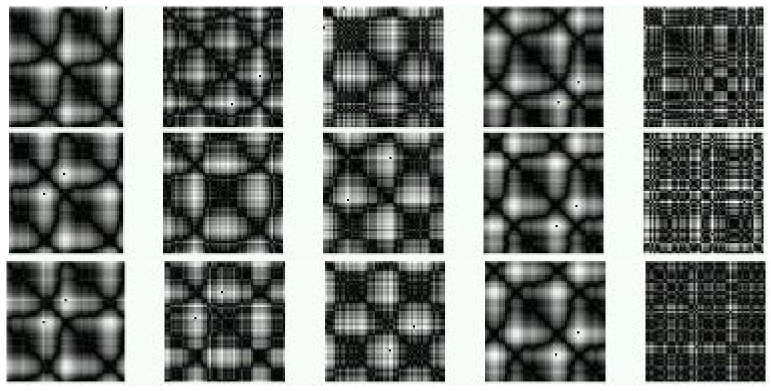


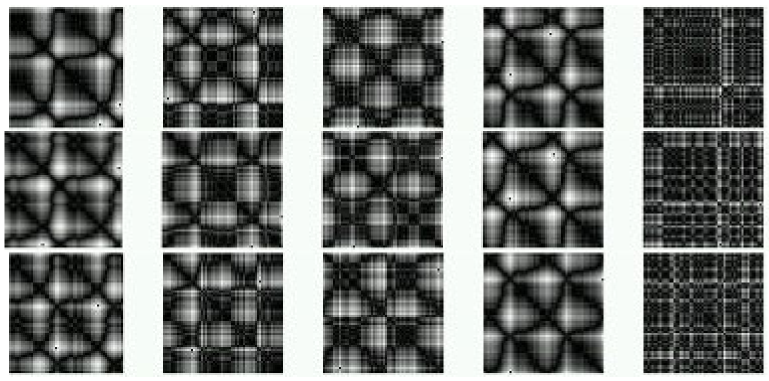
Person 2


Person 3


Person 4


Person 5


Person 6

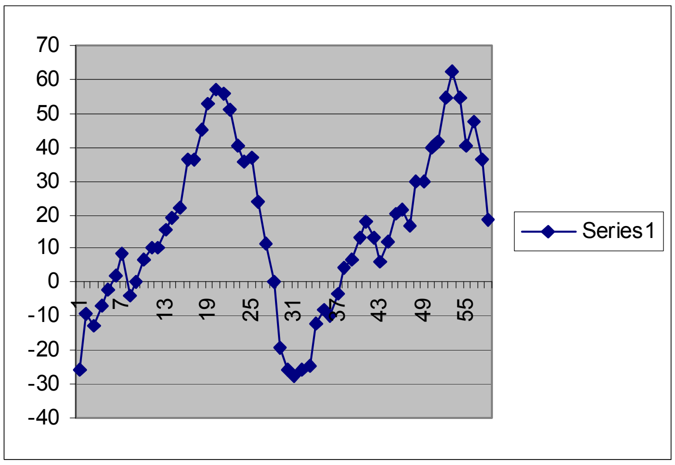


From looking at the plots above can see that the self similarity plot for the torso is very different for the same person at different sequences. It is also evident that the same person has very similar plots for different sequences. Different peoples plots differ, especially subject four, who has quite different plots than the rest of the subjects, which probably explains why he was classified correctly in all of the tests carried out.

From this we can see that self similarity plots do offer us the opportunity to create a gait signature, as stated by BenAbdelkader [18].

11.4.2 Limb angles

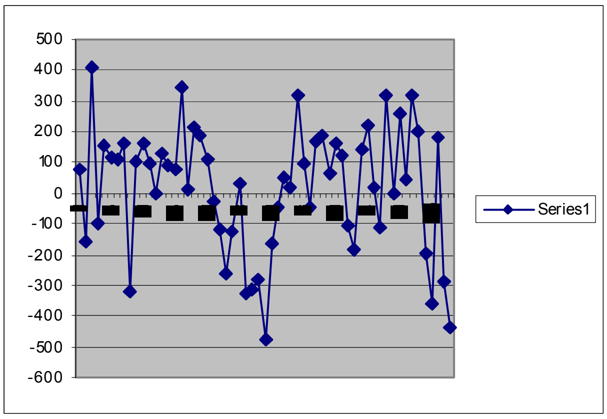
The next measure which was used to derive a gait signature was the angle of the different parts of the body over the whole image sequence. Below is a graph plotting the change in the angle of a persons shin (subject 1).



It is possible to make out the cyclic nature of the gait cycle, and comparing this to other graphs for different subjects shows a definite uniqueness.

11.4.3 Limb angular velocities

The angular velocities of the limbs proved to be a poor measure for a gait signature. of the limbs proved to be a very poor gait signature. This was probably due to errors introduced in the data acquisition process which greatly affected the recognition abilities. The figure below shows the plot of the angular velocity for a person over 60 frames. You can see that it is very jagged and would not be much use for a gait identifier.



11.5 Recognition Rate

One of the most important measures in the whole project, as it is the aim of the project is the recognition rates, for the various measures. Below are a number of tables showing the rates of recognition success for the different types of gait signatures using PCA.

The attempts column shows how many times a person was given as a query for the various signatures. The success column shows how many of the queries were classified correctly by the different methods.

Signature type	Attempts	Success	Percentage %
Joint Positions	36	20	55.6
Limb Angles	36	21	58.3
Limb Angular Velocity	36	13	36.1
Self Similarity Plot	40	22	55.0

Figure 11.1 – gait recognition results

You can see that using the angles of the limbs proved to be the best way to derive a gait signature, with angular velocities being the worse.

12. Conclusion

This chapter gives an overview of the whole project, its successes, limitations, and future improvements.

12.1 Achievements

The aim of this project was to firstly develop a system that was capable of recognising people based upon their gait. I tried differing measures to be used as building blocks for the "gait signatures", including:

- Physical position of joints on the body over a period of time.
- Angles of limbs over a period of time, i.e. over N frames.
- Angular velocity of limbs over N frames.
- Self Similarity plots created from limb angles.

Principle Component Analysis offered a good way to represent most of the variation in the data, whilst offering the advantage of reduced dimensionality. This is evident when using the Gait visualisation option of the program, where changing the magnitudes of the first three principle components makes the points move on the model in a walking like manor.

Recognition rates of up to 58% of correctly classified people out of 36 attempts were achieved by the program, further adding to the notion that gait can be used as a biometric. I am sure that with more advanced classification techniques, higher rates of recognition could have been achieved.

12.2 Limitations

Not all areas of the project were successful. I had hoped to be able to extract the feature points, such as the positions of all the joints automatically from the video sequence by use of model fitting. The segmentation process went well but tracking of the legs took a lot more time than expected and in the end, only a very crude model fitting procedure was implemented, due to time constraints. The head and the body were being tracked well, but the legs were not being tracked to a high enough standard for gait recognition.

Gait as a means of recognition has been prove to be accurate, but is susceptible to things such as clothes, change mood, running/walking.

There are also several limiting simplification assumptions made, which limit some features of the project in the real world:

- The subjects in the video are always walking perpendicular to the camera. This would not be the case in real life as people would be walking at all angles to the video camera.
- The classifiers used are rather simple, and better rates of recognition would be better with different more advanced classifiers. The use of canonical analysis to make bigger separation of the individual classes in feature space could be used to improve the recognition rate.

12.3 Future work

There are an obvious number of improvements, which can be made to the program, in the future, including:

1. ***Improved Pattern Classifiers*** – the pattern classifiers used in this project, K-Nearest neighbour and Spatio Temporal Classification are good classifiers but there are more advanced classifiers available that could potentially offer better rates of recognition. Non-linear classifiers offer the ability to segment groups in feature space in a non-linear manner, leading to improved successful pattern matching. Use of these classifiers could potentially offer the opportunity to increase the recognition rates.
2. ***Enhanced model fitting algorithms*** – the model fitting section of the project definitely needs future work in order to be able to improve the tracking methods used for modelling the leg movements. This is vital if autonomous identification is to take place which is really a prerequisite for a real world application of gait recogntion.
3. ***Code optimisation*** – throughout the development of this project, good design and providing functionality took precedence over optimisation. After running the program it can be seen that the segmentation and the recognition both require large amounts of calculations and both would benefit performance wise from optimising the code. This could be done by improving the algorithms used in the project, or maybe by creating smaller more functional data types which would not use up as much memory during execution.
4. ***Different camera angles –*** this program is limited to acquiring gait signatures from a single camera position, it would be advantageous to have the ability to recognise people from numerous angles, rather than being limited to just one angle.
5. Creating a larger gait data set for test to be tried against. These experiments were carried out on a limited number of subjects, and it would make a good extension to be able to ascertain a larger data set to get a better measure of how effective the gait mechanisms are for recognition.

Appendix A – References

Below is a list of all of the references that I have used in order to develop and understand my project:

1. ***Psychological influences on gait***
   Chris Kirtley
   http://guardian.curtin.edu.au/cga/teach-in/psych/

2. ***Gait as a total pattern of movement***
   M.P. Murray
   American journal of Physical medicine 46(1):290-333, 1967

3. ***Visual perception of biological motion and a model for its analysis***
   G. Johansson
   Perception & Psychophysics, 1973

4. ***Lower limb kinematics of human walking with medial axis transformations***
   A.G. Bharatkumar et al, 1994

5. ***PFinder: Real-time tracking of the human body***
   Christopher Wren, Ali Azarbayejani, trvor Darrell & Alex Pentland
   IEEE transactions on Pattern Analysis and machine intelligence, 97, vol 19, no 7

6. ***Using gait as a biometric, via phase-weighted magnitude spectra***
   D. Cunado, M.S. Nixon & J.N. Carter
   1st Int. Conf. audio and video based biometric person authentification, pp95 –102
   Springer-Verlag, 1997

7. ***Model based gait recognition – variation in hip inclination***
   Dr. David Cunado
   http://www.isis.ecs.soton.ac.uk/image/gait/david_cunado/index.php3

8. ***New area based gait recognition***
   J.P. Foster, M.S. Nixon & A. Prugel-Bennett
   Audio and Video based biometric person authentication
   Springer – Verlag, pg 312-7, June 2001

9. ***Gait Recognition by combined PCA and CA***
   Dr. Vincent Huang
   http://www.isis.ecs.soton.ac.uk/image/gait/vincent_huang/index.php3

10. ***Recognising people by their gait: the shape of motion***
    James J. Little & Jeffrey E. Boyd
    MIT Press *Journal Videre*, 1996

11. ***Background segmentation using range and colour***
    G. Gordon, T.Darrell, M.Harville & J.Woodfill
    http://www.ai.mit.edu/~trevor/papers/1998-071/

12. ***Model-based dynamic human shape extraction by evidence gathering***
    http://www.soton.ac.uk/~dkw198/project/project.htm

13. ***Active Shape Models for visual speech feature extraction***
    Juergen Luettin, Neil A. Thacker & Steve W.Beet
    University of Sheffield, Electronic Systems Group Report No. 95/44

14. ***XML specification***

http://www.w3.org/XML/

15. ***Active Appearance Models***
T. Cootes, G.J. Edwards & C.J. Taylor
In H.Burkhardt & B. Neumann, editors, 5th European Conference on Computer Vision, volume 2, pp484-498. Springer, 1998

16. ***Face recognition using infra red images and eigenfaces***
Ross Cutler, 1996
http://www.cs.umd.edu/~rgc/pub/ir_eigenface.pdf

17. ***Numerical recipes in C: The art of scientific computing (second edition)***
William H. Press, Saul A. Teukolsky, William T. Vetterling & Brian P. Flannery
Cambridge university press, 1988-1992
Available: http://www.ulib.org/webRoot/Books/Numerical_Recipes/bookcpdf.html

18. ***EigenGait: Motion-based Recognition of People using Image Self-Similarity***
Chiraz BenAbdelkader, Ros Cutler, Harsh Nanda & Larry Davis
http://research.microsoft.com/~rcutler/Gait/avbpa67.pdf

19. ***Canonical Space Representation For Recognizing Humans By Gait & Face***
Ping S. Huang, Chris J. Harris & Mark S. Nixon

20. ***W4: Who, When, Where, What: A real time system for detecting and tracking people***
I. Haritaoglu, D.Harwood & L.Davis

21. ***J.Boyd & J.Little – Gait data set***
http://pages.cpsc.ucalgary.ca/~boyd/gait/gait.html