

AN ABSTRACT OF THE THESIS OF

Dharin Haresh Maniar for the degree of Master of Science in Computer Science
presented on December 10, 2008.

Title: Classification of Motion Capture Sequences

Abstract approved:

Weng-Keen Wong

Motion capture data is a digital representation of the complex temporal structure of human motion. Motion capture is widely used for data-driven animation in sports, medicine and entertainment, because of its ability to capture complex and realistic motions. Due to its efficiency and cost, methods for reusing collections of motion capture data are becoming important in the field of computer animation. These motions can then be used for motion blending and morphing, which in turn requires identification and retrieval of the motion from the large collection of motions. Currently, motion data is manually labeled and segmented through a labor-intensive process. This thesis investigates algorithms for the classification of motion capture sequences. This classification task is challenging due to the data being high dimensional, continuous, and time-variant. The main contribution of this thesis is an empirical comparison of a variety of classification algorithms for motion capture sequences. We investigate three different aspects of these classification algorithms: 1) the use of discrete versus continuous models of the data, 2) generative versus discriminative models and 3) dimensionality reduction

through Principal Component Analysis, a linear technique, versus the Gaussian Process Latent Variable Model, a non-linear technique.

© Copyright by Dharin Haresh Maniar

December 10, 2008

All Rights Reserved

Classification of Motion Capture Sequences

by

Dharin Haresh Maniar

A THESIS

submitted to

Oregon State University

in partial fulfillment of

the requirements for the

degree of

Master of Science

Presented December 10, 2008

Commencement June, 2009

Master of Science thesis of Dharin Haresh Maniar

presented on December 10, 2008.

APPROVED:

Major Professor, Computer Science

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Dharin Haresh Maniar, Author

ACKNOWLEDGEMENTS

The author expresses sincere appreciation to Dr. Weng-Keen Wong for his guidance, financial support throughout the master's program and encouragement in the planning, implementation and analysis of this research. Dr. Ronald Metoyer in the computer graphics specialization provided great help with graphics and motion capture fundamentals. Dr. Raviv Raich helped me gain a better understanding of dimensionality reduction. Without their kind assistance and help, it would be impossible to finish this thesis. Special thanks should be given to all the faculty and staff in Electrical Engineering and Computer Science community, with whom I spent really wonderful two years in Corvallis. I would also like to thank my parents for their moral and financial support and assistance during this research.

TABLE OF CONTENTS

	<u>Page</u>
1. Introduction.....	1
2. Motion Capture data format	5
2.1 What is motion capture?.....	5
2.2 How motion is captured?.....	5
2.3 Motion capture data format.....	6
3. Related Work	12
4. Background.....	16
4.1 Dimensionality reduction	16
4.2 Discretization	25
4.3 Graphical models	27
4.3.1 Directed models.....	27
4.3.2 Undirected models.....	32
5. Data and preprocessing.....	34
5.1 Data description.....	34
5.2 Preprocessing.....	36
5.3 Generic pipeline.....	40
5.4 Dimensionality reduction.....	42
6. Discrete modeling	46
6.1 Generative models.....	46
6.2 Discriminative models.....	48

TABLE OF CONTENTS (Continued)

7.	Continuous modeling	53
	7.1 Generative models.....	53
	7.2 Discriminative models.....	56
8.	Discrete results and discussion	61
	8.1 PCA vs GPLVM.....	62
	8.2 Generative vs Discriminative.....	64
	8.3 Other observations.....	65
9.	Continuous results and discussion.....	67
	9.1 PCA vs GPLVM.....	68
	9.2 Generative vs Discriminative.....	69
	9.3 Other observations.....	70
10.	Conclusion.....	71
11.	Future Work.....	73
	Bibliography.....	74
	Appendix A.....	80

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
2.1 Hierarchical structure of human figure.....	7
2.2 Human skeleton corresponding to hierarchical structure given in figure 2.1.....	8
4.1 Directed model.....	28
4.2 Unrolling of hidden Markov model in time slice.....	29
4.3 Graphical representation of linear dynamics system.....	31
4.4 Unrolled conditional random fields.....	32
5.1 Mocap classification pipeline.....	42
6.1 Generative models for discrete approach.....	46
6.1 a First order generative model for discrete approach.....	46
6.1 b Second order generative model for discrete approach.....	46
6.2 Discriminative models for discrete approach.....	48
6.2 a First order discriminative model for discrete approach.....	48
6.2 b Second order discriminative model for discrete approach...	48
6.3 Maximal clique for first order discriminative structure shown in figure 6.2(a).....	49
6.4 Maximal clique for second discriminative structure shown in figure 6.2(b).....	51
7.1 Discriminative models for continuous approach.....	54
7.1 a The first order generative model for continuous approach.....	54
7.1 b The second order generative model for continuous approach.....	54
7.2 Shows the discriminative model for continuous modeling.....	57
7.2 a Shows the First order discriminative model for continuous approach.....	57
7.2 b Shows the model corresponding to the model in 7.2(a).....	57

LIST OF TABLES

<u>Table</u>	<u>Page</u>
2.1 Motion capture file format and references.....	9
4.1 Comparison of different Dimensionality reduction method according to their General Properties	25
4.2 Conditional probability table for the directed model in figure 4.1.	28
5.1 Data description.....	34
5.2 Experiment description.....	36
8.1 Classification accuracy for 2-fold cross validation in discrete approach.....	61
8.1 a Classification accuracy averaged over the two folds	61
8.1 b Fold1 classification accuracy.....	61
8.1 c Fold2 classification accuracy.....	62
8.2 Dimensions and number of states for fold1 and fold2.....	62
8.3 The results of Wilcoxon signed rank sum test conducted for PCA vs. GPLVM (discrete).....	63
8.4 Running time for (discrete) experiment number 5 for the 2 nd fold.	64
8.5 The results for Wilcoxon signed rank sum test conducted for CRF vs. DMP (discrete).....	64
8.6 The results for Wilcoxon signed rank sum test conducted for 1 st order vs. 2 nd order models (discrete).....	65
8.7 Confusion matrix for experiment 6 (discrete).....	66
8.7 a Confusion matrix for Experiment 6 (CRF-1 with PCA) fold1	66
8.7 b Confusion matrix for Experiment 6 (CRF-1 with PCA) fold2	66
9.1 Classification accuracy for 2-fold cross validation in continuous approach.....	68
9.1 a Classification accuracy averaged over the two folds	68
9.1 b Fold1 classification accuracy.....	68

LIST OF TABLES (Continued)

<u>Table</u>		<u>Page</u>
	9.1 c Fold2 classification accuracy.....	69
9.2	Wilcoxon signed rank test for PCA vs. GPLVM in continuous approach	69
9.3	Running time for experiment number 5 for the 2 nd fold.....	70
9.4	Results for Wilcoxon signed rank sum test for CRF vs Gaussian Markov.....	71

LIST APPENDIX OF FIGURES

<u>Figure</u>		<u>Page</u>
A.1	First order CRF model.....	81
A.2	Maximal clique corresponding to figure A.1.....	81
A.3	First order AR-CRF model.....	83
A.4	Shows the model equivalent to the model shown in figure A.3....	83
A.5	Undirected graph obtained after applying graph moralization and triangulation to the model in figure A.4.....	83
A.6	Maximal clique for time slice t corresponding to AR(1)-CRF	83

Classification of Motion Capture Sequences

1. Introduction

Motion capture (mocap) is the process of recording movement and translating movement into digital form. After the data has been captured, it can be used in military, sports, entertainment and medical applications. Mocap systems can simulate more realistic motions, than motions created by hand. It can also synthesize more complex motions. Mocap is frequently used in games and movies due to its detail and ability to express subtle expressions such as the movement of fingers and facial expressions.

There have been great achievements in the field of mocap. Some famous movie characters such as Gollum, the Mummy and King Kong were created using mocap. Two out of three nominees of the 2006 Academy Awards for best animated film were “Monster House” and “Happy Feet” which used mocap technology (Academy Awards, 2007). Gait Analysis is the major application of mocap in clinical medicine.

There are a few shortcomings to mocap. Mocap requires specific hardware and software. The cost of hardware is currently very high. Secondly, movement that does not follow the laws of physics generally cannot be captured. Due to these reasons, data is usually stored in short clips for easy hand labeling, sequencing and searches based on keywords describing the behavior. Extended sequences have many advantages over short clips. Longer shots may be more comfortable for the actors and capture natural transitions between the behaviors. However, in capturing the long sequences, the major problem is to manually segment and label the motion sequences for retrieval and other processing. This manual labeling is a tedious process. We categorize mocap problems broadly into four basic problems: classification, segmentation, motion warping and motion synthesis.

Classification refers to building the systems that can recognize the human motion and activities. Recognition of human body movements is recently receiving more attention in robotics research, since robots must recognize the motion of humans to interact with them. Human motion can be used as a model to train robots. In some computer games, a human-controlled sprite fights with a computer-controlled sprite. The common actions are kicks, punches, blocks, lean and swerve to the left or right, which are usually controlled by a keyboard or joystick. Now consider a game where animations are not computer generated but use actual mocap data. There must be a system that identifies the correct actions in the data for appropriate animations.

Segmentation refers to segmenting a long sequence into distinct behaviors. Usually, long sequences consist of several activities performed by an actor in a single shot e.g. A person walking then jumping and then walking again. The task is to segment the long sequence into “walk”, “jump” and “walk”. This task can be done manually but for very long sequences and large numbers of such examples, the job is tedious. Consequently, we need to automate the segmentation process. This problem is related to the field of change point detection (Balakumar, 2002; Fang, Runger, and Eugene, 2006).

Motion warping (Witkin and Zoran, 1995) is used to edit the original captured motions. Mocap can be used to create custom animations or to create libraries of reusable clip-motions. These custom animations must be tweaked or adjusted to eliminate artifacts. In order to reuse clip motion we must be able to easily alter the geometry, such as animating a character to move on an uneven terrain. We might also add a transition between motions. We want these motions to be as smooth as possible and should look as close to reality as possible.

Motion synthesis is similar to motion warping. The animator specifies the constraints on the keyframe. For example, suppose the animator specifies that he needs a throwing action at a particular frame but the starting frame belongs to a walk motion. A motion synthesis system should create a smooth transition from walk to throw in a particular time frame. Some applications include specifying the path, such as requiring to take the shape

of the numerical digit ‘8’. The database in this case contains walk motions performed in a straight line.

All these problems have the common underlying problem of modeling the motion data. If we are able to find a reliable model then we can find an answer to the above mentioned problems. The high dimensional, continuous, non-linear and time-variant nature of mocap data makes it very difficult to model the data. This area is fairly new in the field of machine learning. Although there has been past research using graphical models for motion data (Yamato, Ohya, and Ishii, 1992; Bobick and Wilson, 1995; Goddard, 1994; Startner and Pentland 1995; Oliver, and Pentland, 1997), none of the past work has approached the task of classifying entire motion sequences where each frame represents a full 96 dimensional vector of the joint angles of an articulated skeleton.

In this paper, we are primarily concerned with classifying entire pure mocap sequences. First, we investigate the use of discrete models for this classification task. Discretization of the data results in faster algorithms during classification but it comes at a cost of losing information. Secondly, we explore the use of continuous models for classifying mocap sequences. For both discrete and continuous models, we explore the use of two main dimensionality reduction algorithms- PCA, which is a linear method, and GPLVM, which is a non-linear method. In addition, another angle that I investigate is the effectiveness of discriminative versus generative approaches. Discriminative models capture the conditional probability $P(\mathbf{M}|\mathbf{Y})$, where \mathbf{M} is the motion label and \mathbf{Y} is the observed data i.e. the frames in the sequence. In contrast, the generative models model the joint probability $P(\mathbf{M}, \mathbf{Y})$. In a nutshell, discriminative models are best suited for distinguishing between motions while generative model are best suited for synthesis of motions. The rest of the paper is organized as follows:

Chapter 2 gives a brief introduction to motion capture and the BVH motion capture data format. In Chapter 3 we discuss the related work. Chapter 4, introduces the basic concepts of dimensionality reduction methods, graphical models and discretization methods. We also give a detailed survey of dimensionality reduction methods. Chapter 5

is devoted to the preprocessing of motion capture data, We discuss discrete models in chapter 6 followed by continuous modeling in chapter 7. The chapter 8 and 9 are the results and discussion for discrete and continuous approach respectively. We conclude in Chapter 10 and in chapter 11 we discuss future work.

2. Motion capture data

In this section we describe how the mocap data is acquired, what the data formats are and what applications are possible for motion capture data.

2.1 What is motion capture?

Motion capture is a technique that digitally records the movements. These movements are widely used in entertainment, military and medical applications. In mocap, movements of actors are sampled many times per second. Mocap has some advantages over traditional computer animation of 3D model. Mocap can be used to capture more realistic and complex human movements but animal motions are difficult to capture. Another disadvantage is that, if something goes wrong in the process of data capture, the whole motion needs to be re-captured. Also, the specific hardware required for motion capture is fairly expensive.

2.2 How motion is captured?

A performer wears markers near each joint to capture the positions and angles of these markers as the performer moves. The mocap computer software records the positions, angles, velocities, acceleration and impulses.

Optical systems utilize data captured from image sensors to triangulate the 3D positions of a subject between one or more cameras calibrated to provide overlapping projections. These systems produce data with 3 degrees of freedom for each marker, and rotational information must be inferred from the relative orientation of three or more markers. For instance shoulder, elbow and wrist markers provide the angle of the elbow.

Passive optical systems use markers coated with reflective material to reflect light back. An object with markers attached at known positions is used to calibrate the cameras and obtain their positions and the lens distortion of each camera is measured. Provided two calibrated cameras see a marker, a 3 dimensional fix can be obtained. Typically a system

will consist of around 6 to 24 cameras although systems of over three hundred cameras currently exist to try to reduce marker swap.

Active optical systems triangulate positions by illuminating one LED at a time very quickly. Here the markers are themselves powered to emit their own light. Each marker is given a unique identification number.

Intensive research in computer vision is leading to rapid development of markerless mocap systems in which performers do not have to wear any markers. Special computer algorithms are designed to allow the system to analyze optical input and identify the human body.

Non-optical systems such as inertial systems, mechanical motion and magnetic systems also exist. An inertial system is based on miniature inertial sensors. This is a low-cost and easy-to-use system based on biomechanical models and fusion algorithms. In mechanical systems, a performer attaches a skeleton like structure to their body. Typically, they are rigid structures of jointed, straight metal or plastic rods linked together with potentiometers that are articulated at the joints of the body. Magnetic systems calculate position and orientation by the relative magnetic flux of three orthogonal coils on both transmitter and receiver.

2.3 Motion capture data format

2.3.1 Terminology.

The following list (Meredith and Maddock, 2001) outlines some important keywords that are used to identify a human motion.

- **Skeleton:** The whole character for which motion represents.
- **Bone:** The smallest segment within the motion to which individual translation and rotations are applied during the animation.

- **Channel or Degree of freedom:** Each bone is subject to a position, orientation. The each parameter specifying these transformations is referred as a channel or degree of freedom.
- **Frame:** Each motion is comprised of a number of frames, where each frames depicts the position of each bone. Mocap can be performed at the rate of 240 frames per second but 30 to 60 frames per second is the norm.
- **Hierarchy:** Mocap data contains a hierarchy of joints. For instance, the left wrist (lwrist) joint is the parent of the left fingers (lfingers) joint. Hence any transformation to lfinger joint is applied in accordance with the lwrist joint. The segment joining lwrist and lfinger is a bone.

Figure 2.1 shows the sample hierarchy for Mocap data and the corresponding human skeleton can be seen in Figure 2.2. The keyword “*End site*” in the hierarchy corresponds to end-effectors. Specifically this encapsulates an offset that is used to infer the bone’s length for the joints, which does not have child.

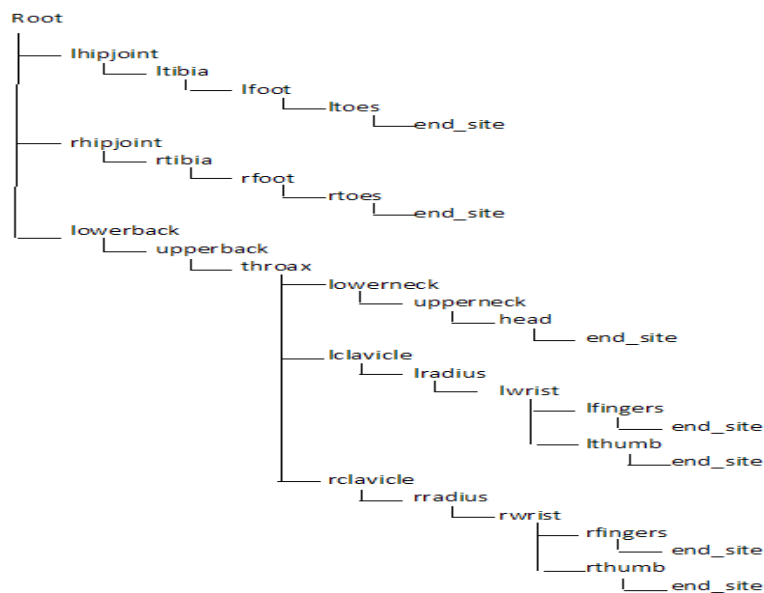


Figure 2.1: Hierarchical structure of human figure.

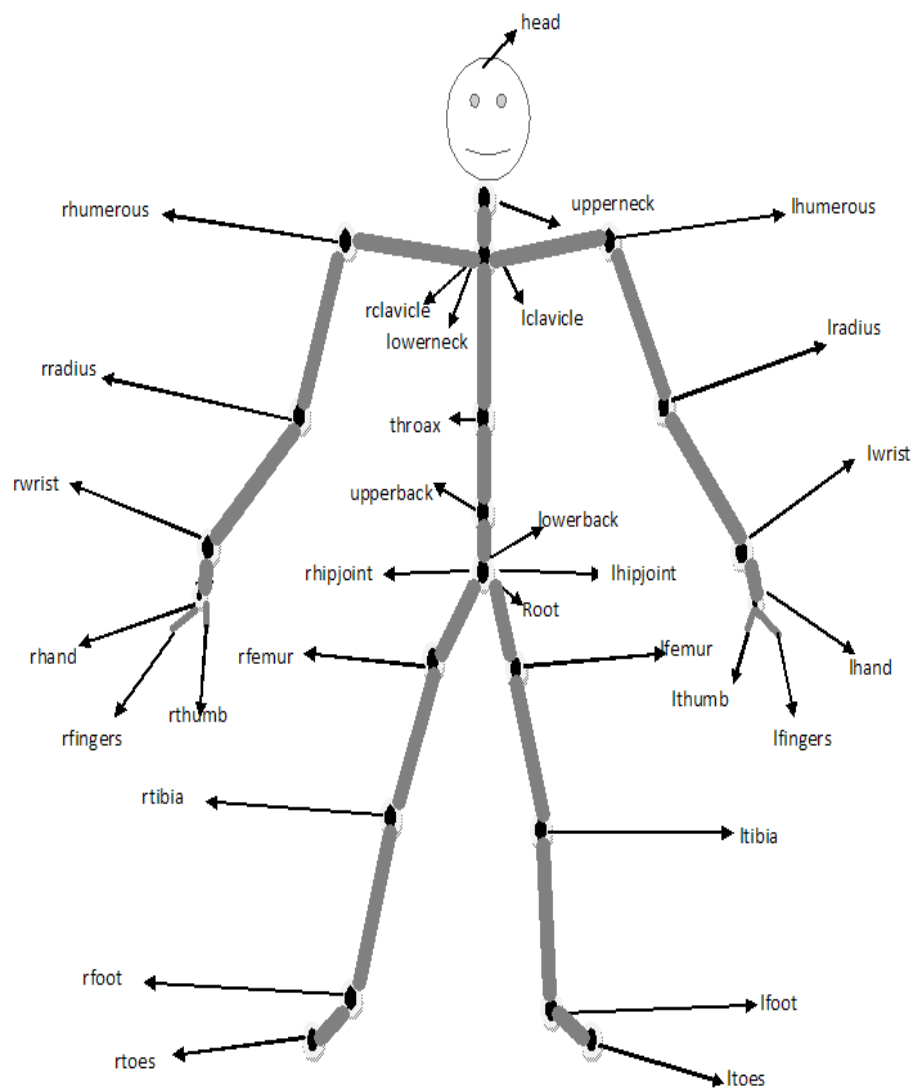


Figure 2.2: Human skeleton corresponding to the hierarchical structure given in figure 2.1.

2.3.2 Mocap file formats

The following table 2.1(taken from (Meredith and Maddock, 2001)) outlines the available mocap file formats. **ASF-AMC** and BioVision Hierarchy (**BVH**) are the most widely used formats. We will discuss the **BVH** format in detail in section 2.3.3.

File Extension	Associated Company / Description	File Format Reference
ASC	Ascension	NO LINK
ASF and AMC	Acclaim	http://www.darwin3d.com/gamedev/acclaim.zip
ASK and SDL	BioVision/Alias	NO LINK
BVA and BVH	BioVision	http://www.biovision.com/bvh.html
BRD	LambSoft Magnetic Format	http://www.dcs.shef.ac.uk/~mikem/fileformats/brd.html
C3D	Biomechanics, Animation and Gait Analysis	http://www.c3d.org/c3d_format.htm
CSM	3D Studio Max, Character Studio	http://www.dcs.shef.ac.uk/~mikem/fileformats/csm.html
DAT	Polhemous	NO LINK
GTR, HTR and TRC	Motion Analysis	http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/ {HTR.html, TRC.html}
MOT and SKL	Acclaim- Motion Analysis	(Under Development - http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/SKL-MOT.html)

Table 2.1: Motion capture file formats and references.

2.3.3 BVH file format

The BVH file format includes skeleton hierarchy information in addition to the motion data.

A BVH file has two parts. First the header section describes the skeleton hierarchy and initial pose of the skeleton. Second, the data section contains the motion data. The start of the header begins with the keyword '**HIERARCHY**'. The following lines start with the keyword '**ROOT**'. A BVH file may contain any number of hierarchies.

Each segment of the hierarchy contains some data relevant to just that segment then it recursively defines its children. The line following the keyword '**ROOT**' contains a left curly brace '{'. The first piece of information of a segment is the offset of that segment from its parent. The offset will generally be zero. The offset is specified by the keyword '**OFFSET**'. The line following the offset contains the channel header information. This has the '**CHANNELS**' keyword followed by a number indicating the number of channels and then a list of many labels indicating the type of parameter it is. The order of rotation is in the order Z,X,Y. On the line following the channels there could be one of two keywords: either '**Joint**' or '**End site**'. A joint definition is identical to the root definition except for the number of channels. The end site information ends the recursion and indicates that the current segment is the end effectors. The end site definition provides one more bit of information. It gives the length of the preceding segment just like the offset of a child defines the length and the direction of its parent's segment. The end of any joint, end site or root definition is denoted by a right curly brace '}'.

The data section begins with the keyword '**MOTION**' which is followed by a line indicating the number of frames by the '**Frames**' keyword. On the next line after the frame definition is the '**Frame time**' which indicates the sampling rate of the data.

The rest of the file contains the actual motion data. Each line is one sampled frame of motion data. In total, there are 31 joints, one for the root and the remainder for the joints. The root joint has 6 channels and the remaining 30 joints have 3 channels each. Therefore, we have a total 96 degrees of freedom. Each of the lines in the data section is a vector of 96 elements. The order is decided by the hierarchy. To calculate the position of a segment, a transformation matrix is created in which the translational information is

specified in the hierarchy section and the rotational information is found in the data section.

The rotation matrix R is given by, $R = ZXY$. Once the local transformation is created, it is concatenated it with the local transformation of its parent, then its grandparent and so on. We used the BVH format for our experiments. The data can be found at the CMUMocap Library (CMU Mocap Lirary).

3. Literature review

Human motion analysis can be broadly divided into three main categories: motion recognition, motion tracking, and analysis of body part movement. Since our main focus in this thesis is motion recognition, this chapter will discuss related work in motion recognition. Much of this chapter is based on earlier surveys (Agarwal and Cai, 1999; Wang, Weiming, and Tieniu, 2002).

Motion recognition

Human motion recognition is based on tracking the human body through the sequence of images or sequence of poses captured using a motion capture system. Motion recognition approaches can be broadly divided into two categories: template matching and state space. The major work to recognize human motion relies on state- space approaches (Yamoto, Ohya, and Ishii, 1992; Bobick and Wilson, 1995; Goddard, 1994; Startner and Pentland 1995; Oliver, and Pentland, 1997).

i) Template matching

Template matching (Polana and Nelson, 1994; Bobick and Davis, 1996; Galata, Johnson, and Hogg, 2001) is an approach, which extracts features from the unlabeled sequence and compares these features with pre-stored features obtained from existing motions in a database. Polana and Nelson (Polana and Nelson, 1994) used optical flow fields (Galata, Johnson, and Hogg, 2001) between frames to divide the motion into X and Y directions for template matching. Features derived from 2D meshes were used. Bobick and Davis (Bobick and Davis, 1996) interpret human motion in an image as Motion Energy Images (MEI) and Motion History Images (MHI). The motion images are obtained by subtracting two consecutive images and thresholding this difference into binary values. Over a period of time such images are accumulated. These images are called MEI, which are then enhanced to MHIs where each pixel value is proportional to the duration of motion. A nearest neighbor algorithm is then used for the motion recognition. Chomat and Crawley

(Chomat and Crowley, 1998) generated motion templates by using temporal-spatial filters computed by PCA. Recognition is performed by a naïve Bayes classifier. The advantage of template matching is its low computational cost and its simple implementation. The drawback is its sensitivity towards noise. For example, a motion of a person walking slowly will be considered distinct from a person walking quickly. Template matching also does not take into account the correlation between various joints such as while walking, the left leg moves in front in correspondence with the right hand. There has been a great deal of research on template matching by using similarity measures for multi-attribute pattern recognition. Various similarity measures have been defined based on PCA (Krzanowski, 1979; Shahabi and Yan, 2003).

ii) State space approach

State space models are widely used to predict, estimate and detect time variant signals. The state space approach defines each pose as a state. Each state can be defined using a probabilistic model of some form, such as a multivariate Gaussian which can incorporate the inter-joint correlation. These states are connected by certain transitional probabilities. Any motion can be seen as passing through the states with the given probabilities. During training of the model in the given motions from the database the major task is to calculate the probabilities. For a given unknown motion, the joint probability is calculated with each different motion. The maximum value is selected as a criterion for classification. The example of state space techniques are Hidden Markov Model (HMM) and Neural Networks (NN).

Bobick (Bobick and Wilson, 1995) and Campbell (Bobick and Davis, 1996) applied 2D or 3D tracking of joints for activity recognition. Both approaches transform the continuous state space into a corresponding discrete version using K-means. Finally, a motion is described as a sequence of such discrete symbols. Goddard (Goddard, 1994) and Yamato et al. (Yamato, Ohya, and Ishii, 1992) used HMMs to recognize human motion. Yamato (Yamato, Ohya, and Ishii, 1992) used 2D blobs as a feature to identify human motion. Learning was implemented by generating symbol pattern for each class.

Neural networks is another approach to answer the human motion recognition task. Guo et al. (Guo, Xu, and Tsuji, 1994) used a neural network to understand human motion behavior and the underlying pattern. Similarly, Rosenblum (Rosenblum, Yacoob, and Davis, 1994) used a neural network to recognize emotions of a subject.

Apart from HMMs and basic neural networks there have been other types of models used for recognition such as methods based on PCA (Chomat and Crowley, 1998; Rosenblum, Yacoob, and Davis, 1994), variants of HMMs that include Coupled Hidden Markov Models (Bregler, Oliver, and Pentland, 1997), Variable Length Markov Model (Galata, Johnson, and Hogg, 2001) and Time Delay Neural Networks (Lin, Nein, and Lin, 1999) is another variation of NN which tries to recognize motion. Problems with many of these more complex models include the absence of a closed form solution, an intrinsic non-linear model, and local optima. Also, ‘under fitting’ or ‘over fitting’ could be a problem depending on the amount of training data.

Other recognition methods:

The above taxonomy was discussed in (Agarwal and Cai, 1999 ; Wang, Weiming, and Tieniu) but there exist other analysis methods which do not fall into the above categories.

Recently, linear dynamic systems (LDS) have been shown to be successful at modeling motion. Fitzgibbon (Pavlovic, Rehg, Cham, and Murphy, 1999) proposed autoregressive (AR) models. These approaches model the temporal behavior but breaks down when underlying characteristics are non-linear. To overcome the above problem multiple linear systems were used. Pavlovic et al. (Pavlovic, Rehg and Maccormick, 2002) propose a switching linear dynamic system (Bregler, 1997; Murphy K. P., 1998). However, it is difficult to learn the model with transitions between the linear systems. Li et al. (Yan, Wang, and shum, 2002) propose two-level statistical models to learn non-linear dynamics. In their model, they have N textons, which are image template that can be transformed geometrically and photometrically. Each texton is defined by a LDS. These

LDSs are used to find the local linear dynamics and transition matrix to model global non-linear dynamics. Later on, the textons are used to describe the pattern in the process.

We now briefly summarize other work which do not use LDSs. Nakata (Nakata, 2007) uses inter-limb correlation analysis to segment and recognize the distinct human behavior. Chuanjun Li et al. (Li, Kulkarni, and Prabhakaran, 2007), used Singular Valued Decomposition (SVD) and Support Vector Machine (SVM) to segment and recognize the human motion data. The SVM classifiers neglect the temporal dependencies across the frames.

Altun et al. (Altun, Hoffman, and Smola, 2004) used Gaussian Process classification for segmenting and annotating the human motion sequences. The main objective of Altun et al. was to combine the advantages of Conditional Random Fields (CRF) with SVMs. They retained the probabilistic semantics of CRF which helps to incorporate prior knowledge within a probabilistic framework. Also, posterior probabilities can be used for predictions. The curse of dimensionality is overcome by the use of kernels.

Sminichisescu et al. (Sminichisescu, Kanaujia, Zhiguo, and Metaxas, 2005) proposes a complementary discriminative approach to human motion recognition based on CRFs and Maximum Entropy Markov Model (MEMM). They were able to significantly improve the accuracy over the HMMs.

Most of the previous research considers human motion as a sequence of images. Moreover, past work models temporal dynamics with a HMM. In this thesis we deal with all the joint angles on an articulated human skeleton and not with images.

4. Background

In this chapter we will discuss the background of various algorithms and method. First, we discuss dimensionality reduction followed by discretization methods and at the end we discuss the basic concept of graphical models.

4.1 Dimensionality reduction.

The high dimensional nature of mocap data makes the distances between two motions almost the same, making them difficult to distinguish. This is nothing but “Curse of dimensionality (Beyer, Goldstein, Ramkrishnan, and Shaft, 1999)”. A large number of dimensions makes motions difficult to model and increases the computational expense at the same time. As discussed earlier, mocap data has 96 dimensions to represent a single human motion frame. Many times, only a few dimensions are sufficient to represent the motion, such as when a person is clapping, the only joint angles that have more variations are the hands as compared to the legs. This property allows us to use a dimensionality reduction technique as a preprocessing step. Dimensionality reduction is a technique to reduce the number of dimensions under consideration by finding a lower dimensional representation of the original data while preserving most of the information. In this section we will explore the different dimensionality reduction techniques. Also, we will discuss the pros and cons of using these techniques on MOCAP data.

Dimensionality reduction can be divided into feature selection and feature extraction. In this thesis we will concentrate on feature extraction, which maps a higher dimensional representation into a lower dimensional space. We will compare these methods on the basis of the number of parameters required, time complexity, memory requirement and out-of-sample extension. By out-of-sample extension, we mean the ability to incorporate new high dimensional data points into an existing low dimensional space (Matten et al., 2007). In a parametric out-of-sample extension, dimensionality reduction techniques provide all necessary parameters in order to transform a new data point.

There are two major categories of feature extraction techniques.

- i. Linear techniques.
- ii. Non-linear techniques.

4.1.1 Linear techniques

Let \mathbf{Y} be a higher dimensional representation of data with dimensionality D and \mathbf{X} is its lower dimensional embedding with dimensionality d where $d < D$. Linear techniques try to find the linear mapping of data into a lower dimensional space so as to retain the maximum amount of information. In mathematical terms, linear techniques try to find \mathbf{A} , such that $\mathbf{X} = \mathbf{A}\mathbf{Y}$. The matrix \mathbf{A} has dimensions $(d \times D)$. Similarly we can reconstruct the original data from its lower dimensional representation $\mathbf{Y}' = \mathbf{A}^{-1}\mathbf{X}$. This reconstruction results in the following reconstruction error,

$$e = \sum_{i,j=1}^n \|y_i - y_j'\|^2 \quad (4.1)$$

where $\|X\|$ and n indicates Euclidean distance and the number of data points respectively.

Linear techniques try to find \mathbf{A} such that the reconstruction error ' e ' is minimized. The classical linear techniques include PCA, Multi-Dimensional Scaling (MDS), Independent Component Analysis (ICA), and Linear Discriminant Analysis (LDA).

4.1.1.1 Principal Component Analysis (PCA):

PCA (Hotelling, 1933; Smith, 2002) is the most commonly used dimensionality reduction technique. It performs a linear mapping to a lower dimensional subspace in such a way that as much of the variance of the data is retained as possible in the lower dimensional space.

First, the original data is centered, causing its empirical mean to be 0. This can be done by subtracting the mean of the data from each data point. Then the correlation matrix of the data is constructed and the eigenvectors on this matrix are computed. The eigenvectors can also be constructed by performing a Singular Value Decomposition of the centered data. The eigenvectors that correspond to the d largest eigenvalues (the principal components) form the columns of matrix A . Matrix A can now be used to reconstruct a large fraction of the variance of the original data. The original space with dimensionality D has been reduced with some loss in accuracy to the d dimensional space spanned by a few eigenvectors.

With PCA, we are left with the issue of selecting the number of dimensions d in lower dimensional space.

Suppose we have D eigenvalues, $(e_1, e_2, e_3, \dots, e_d)$. We select the first d eigenvectors corresponding to the first d eigenvalues. Then the percentage of information that we are retaining can be calculated as,

$$r = \frac{\sum_{i=1}^d e_i}{\sum_{i=1}^D e_i} \quad (4.2)$$

We select the minimum d such that, $r \geq 0.90$. This is a typical criterion for PCA analysis (Fukunga, 1990).

PCA, however has few disadvantages. First, PCA assumes that there exists a linear relationship between the observed data and the basis vector. Secondly, PCA neglects the lower $(D - d)$ dimensions resulting in some reconstruction error. Finally, PCA assumes that the important dynamics are summarized by the principal components that account for the majority of the variance.

Probabilistic PCA (Tipping and Bishop, 1997) was developed to provide probabilistic interpretation of PCA which can account for noise in the dimensionality reduction process.

To relax the linear assumption non-linear methods such as kernel PCA (Scholkopf et al., 2001) have been developed. This will be discussed in non-linear technique section.

Due to its simplicity PCA is widely used as a preprocessing step to reduce the dimensionality. PCA provides the matrix A , which can then be used to transform any new data point. Hence, PCA is a parametric out-of-sample extension technique. It has time complexity of $O(D^3)$ and memory requirement of $O(D^2)$.

4.1.1.2 Independent Component Analysis (ICA):

Independent Component Analysis (Comon, 1994) is a method for separating multivariate data into independent components. ICA assumes that data is generated from k independent components S_i , $i = 1, 2, 3, \dots, k$, and the observed data $Y = (Y_1, Y_2, Y_3, \dots, Y_n)$. are generated as a sum of independent components.

$$Y_i = a_{i,1} \cdot S_1 + a_{i,2} \cdot S_2 + \dots + a_{i,k} \cdot S_k \quad (4.3)$$

weighted by the weights $a_{i,k}$. Now, the observed data can be represented as $Y = AS$. ICA tries to find $W = A^{-1}$. Hence the original sources can be found as $S = WY$. One approach to estimating the independent component is minimization of mutual information. ICA is closely related to PCA; we chose to use PCA due to it being a more commonly used technique for motion capture analysis.

4.1.1.3 Linear Discriminant Analysis (LDA):

LDA (Fisher, 1936) is a supervised technique requiring class labels in order to maximize the between-class scatter and minimizes the within-class scatter. LDA considers maximizing the following objective:

$J(W) = \frac{w^T \cdot S_B \cdot w}{w^T \cdot S_w \cdot w}$, where S_B is the “between-class scatter matrix” and S_w is the “within-classes scatter matrix”. Due to its supervised nature, LDA was not used as a dimensionality reduction method in our work.

4.1.1.4 Multidimensional Scaling(MDS):

Multidimensional scaling (Cox, Cox, and Raton, 2003) maps data from a high dimensional space to a lower dimensional space by retaining the pair wise distances between the data points. The mapping is assessed in terms of a stress function. A stress function is a measure of error between pair-wise distances in the high dimensional space and the corresponding low dimensional embedding. MDS has many variants which simply modify the input distance matrix. Performing multidimensional scaling with the input expressed as squared Euclidean distance is equivalent to performing PCA. This is due to the relationship between the eigenvectors of the covariance matrix and the squared Euclidean distance matrix (Matten et al., 2007). It has the time complexity of $O(n^3)$ and the memory complexity of $O(n^2)$. Due to its high computational and memory requirement, we prefer PCA over MDS in this thesis.

4.1.2 Non- linear technique.

The assumption that there exists a linear relationship between the higher and lower dimensional space makes linear dimensionality reduction techniques difficult to use on natural datasets. Non-linear techniques assume that the data of interest lies on an embedded non-linear manifold within the higher dimensional space. Non-linear techniques try to unfold the embedding.

There exist two categories of non-linear techniques, those which actually provide a mapping and those that just provide visualization. Non-linear mappings can be subdivided into three subcategories: 1) techniques that attempt to preserve ^{global} properties, 2) techniques that attempt to preserve local properties and 3) techniques that perform global alignment of mixture local models. In this section we discuss non-linear techniques such as: Isomaps, Kernel PCA, Locally Linear Embedding (LLE), Laplacian Eigenmaps and the GPLVM.

4.1.2.1 Kernel PCA:

Kernel PCA(Scholkopf et al., 2001) is an extension of PCA that uses kernel methods which allow the linear operations of PCA to solve non-linear problems. Kernel PCA is a global technique that preserves global properties of the data. The kernel trick maps the original non-linear observations into a higher-dimensional space. The most frequently used kernel functions are linear kernel, polynomial kernel and Gaussian kernel. The major disadvantage of kernel PCA is the size of the kernel matrix. It is proportional to the square of number of instances 'n'.

A linear kernel makes kernel PCA equal to traditional PCA. Kernel PCA has a time complexity of $O(n^3)$ and a memory complexity of $O(n^2)$. Due to such a high time and space complexity PCA is preferred over kernel PCA.

4.1.3 Isomap:

MDS suffers from the fact that it depends on the Euclidean distance between two data points, thereby having a memory requirement of $O(n^2)$. Also, it might not be able to unfold the manifold and the straight line distance might not be the actual distance between two points on the manifolds.

Isomap (Tenenbaum, De Silva, and Langford, 2000) is a global technique that takes into account the distribution of the neighboring data points. Isomap preserves the pair wise geodesic distance between two points, where the geodesic distance is the distance between two points measured over a manifold.

In Isomap, the neighborhood graph G is first constructed. In this neighborhood graph every data point is connected to its k -nearest neighbor. The shortest path between any two points is a good approximation to the geodesic distance. The shortest path can be easily calculated using Dijkstra's shortest path algorithm. Once the neighborhood graph G , is constructed MDS is applied on it in order to find the lower dimensional representation. If k is equal to $n - 1$, Isomap reduces to MDS.

There are few disadvantages of Isomap. Isomap may construct erroneous connections, it cannot unfold non-convex manifolds, it may suffer from holes in the manifold and finally, it is not resilient to noise.

The main disadvantage that forces us to discard the Isomap as a candidate for dimensionality reduction method is the out-of-sample state extension. For Isomap a nonparametric out-of-sample extension is presented in (Bengio, Paiement, and Vincent, 2004) which tends to be computationally expensive. For classification of motion data, the out-of-sample property is very important.

4.1.2.4 Locally Linear Embedding (LLE):

In contrast to isomap, LLE is a local dimensionality reduction technique that attempts to preserve local properties. In LLE every data point is written as a linear combination of its k -nearest neighbors. LLE attempts to retain the weights learned in the higher dimensional space into the lower dimension. Roweis and Saul (Roweis and Saul, 2000) showed that the lower dimensional subspace can be found by an eigenvalue decomposition of the covariance matrix of the local manifold. The smallest d nonzero eigenvalues give the lower dimensional embedding. Similar to Isomap, LLE also has a nonparametric out-of-sample extension with high time complexity.

4.1.2.5 Laplacian eigenmaps:

Another local technique is the Laplacian Eigenmap. In a Laplacian Eigenmap, the distances between a data point and its k -nearest neighbors are minimized. This method uses weights such that the point which is closest to the data point is given more weight than the other points. The Laplacian Eigenmap algorithm first finds the neighborhood graph G . The weight of the edge is calculated by using the Gaussian kernel function. In the cost function, larger weights correspond to small distances between two data points. Similar to Isomap and LLE, Laplacian Eigenmaps also has the problem of not having a computationally efficient way to handle out-of-sample extension.

4.1.2.6 Gaussian Process Latent Variable Model (GPLVM):

The GPLVM is equivalent to non-linear probabilistic PCA. GPLVM can be seen as a Gaussian Process with a Radial Basis Function (RBF) kernel. Suppose we have high dimensional observation $\mathbf{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_n\}$, where $\mathbf{Y}_i \in R^D$. The task is to find its lower dimensional representation $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n\}$, where $\mathbf{X}_i \in R^d$ and $d < D$. In PPCA, the relationship can be written as,

$\mathbf{Y}_n = \mathbf{W}\mathbf{X}_n + \boldsymbol{\eta}_n$ where, $\mathbf{W} \in R^{D \times d}$ is the linear relationship between the observation space and the latent space $\boldsymbol{\eta}$ represents Gaussian noise. $p(\boldsymbol{\eta}_n) = \mathcal{N}(\boldsymbol{\eta}_n | 0, \boldsymbol{\beta}^{-1} \cdot \mathbf{I})$

In (Lawrence, 2005) Lawrence has shown that the marginal likelihood for each point can be found analytically through marginalization of latent variables as,

$$p(\mathbf{Y}_n | \mathbf{W}, \boldsymbol{\beta}) = \mathcal{N}(\mathbf{Y}_n | 0, \mathbf{W} \cdot \mathbf{W}^T + \boldsymbol{\beta}^{-1} \cdot \mathbf{I}) \quad (4.5)$$

This is a typical representation of latent models where we marginalized latent variables. In a Bayesian framework, parameters such as \mathbf{W} , are viewed as random variables. We can find the dual of PPCA by optimizing latent variables. i.e. we marginalize the parameters \mathbf{W} instead of latent variables now. The dual of equation (4.5) is given by,

$$p(\mathbf{Y}_{:,d} | \mathbf{X}, \boldsymbol{\beta}) = \mathcal{N}(\mathbf{Y}_{:,d} | 0, \mathbf{X} \cdot \mathbf{X}^T + \boldsymbol{\beta}^{-1} \cdot \mathbf{I}) \quad (4.6)$$

In order to optimize \mathbf{X} , we must maximize the following log likelihood term,

$$l(\mathbf{Y}) = -\frac{D \cdot N}{2} \ln(2\pi) - \frac{D}{2} \ln|\mathbf{K}| - \frac{1}{2} \text{tr}(\mathbf{K}^{-1} \cdot \mathbf{Y} \cdot \mathbf{Y}^T) \quad (4.7)$$

Where, $\mathbf{K} = \mathbf{X} \cdot \mathbf{X}^T + \boldsymbol{\beta}^{-1} \cdot \mathbf{I}$

the values for \mathbf{X} which maximize the likelihood are given by,

$$\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^T \quad (4.8)$$

Where, \mathbf{U} is an $n \times d$ matrix whose columns are the first d eigenvectors of $\mathbf{Y} \cdot \mathbf{Y}^T$, \mathbf{L} is a diagonal matrix whose diagonal entries give the eigenvalues and \mathbf{V} is an arbitrary rotation matrix. We refer the reader to (Lawrence, 2005) for more details.

We can view the PPCA equation (4.7) as a Gaussian Process with the covariance function equal to $\mathbf{K} = \mathbf{X} \cdot \mathbf{X}^T + \boldsymbol{\beta}^{-1} \mathbf{I}$. In fact, PPCA is equivalent to a special case of the GPLVM with a linear kernel. If we replace this kernel with a RBF kernel, we get non-linear probabilistic PCA which is the standard representation of the GPLVM. The RBF kernel takes the form,

$$k(\mathbf{X}_i, \mathbf{X}_j) = \theta_{rbf} \exp\left(-\lambda \cdot \frac{(\mathbf{X}_i - \mathbf{X}_j)^T (\mathbf{X}_i - \mathbf{X}_j)}{2}\right) + \theta_{bias} + \theta_{white} \cdot \delta_{ij} \quad (4.9)$$

θ_{rbf} , θ_{bias} , θ_{white} , λ are kernel parameters and δ_{ij} is Kronecker's delta.

4.1.3 Comparison according to MOCAP data.

In section 4.1.1 and 4.1.2 we discussed various linear and non-linear dimensionality reduction techniques. This list is not exhaustive. There exist many more dimensionality reduction methods. Also there are many variations available for the above discussed methods. In this section we will compare the performance of these dimensionality reduction methods on the mocap data. We assess the comparison according to the classification accuracy achieved for mocap data. Please refer to the table 4.1, which was taken from (Matten, Postma, and Van der Herik, 2007) for comparison of the general properties of these dimensionality reduction methods. These properties include the convexity of the optimization problem, number of parameters required, computation complexity, and time complexity.

All the methods optimize a convex cost function, allows a global optimum to be found. The second column gives the free parameters that need to be optimized. Here k is the number of nearest neighbors. $\mathbf{K}(\cdot, \cdot)$ is the kernel function chosen and σ is the weight. Columns 3 and 4 gives the computational and memory complexities. n is the number of

samples. D is the number of dimensions and p denotes the ratio of nonzero elements in sparse matrix to the total number of elements.

We selected PCA and GPLVM as dimensionality reduction methods for the classification of mocap sequences. PCA was selected due to its simplicity, computational efficiency and its wide use on natural datasets. GPLVM is a non-linear technique which is prevalent in motion synthesis (Yan, Wang, and shum, 2002). Recent work has shown that GPLVM performs well on the problems of synthesizing motion.

Technique	Convex	Parameters	Computational	Memory
PCA	Yes	None	$O(D^3)$	$O(D^2)$
LDA	Yes	k	$O(n.k)$	$O(n)$
MDS	Yes	None	$O(n^3)$	$O(n^2)$
Kernel PCA	Yes	$K(.,.)$	$O(n^3)$	$O(n^2)$
ISOMAP	Yes	k	$O(n^3)$	$O(n^2)$
LLE	Yes	k	$O(pn^2)$	$O(pn^2)$
Laplacian eigenmaps	Yes	k, σ	$O(pn^2)$	$O(pn^2)$

Table 4.1: Comparison of different Dimensionality reduction method according to their General Properties

4.2 Discretization.

One approach for Mocap Classification is to discretize the data. Suppose we have continuous data $X = \{X_1, X_2, X_3, \dots, X_T\}$ where $X_i \in R^d$. We need to discretize the space R^d into Q different states. This section describes the options that we considered.

1) KD-trees: A KD-tree is a multi-resolution space-partitioning data structure for organizing data points in a k-dimensional space. A KD-tree is built by first placing all

data points in one large node at the root of the tree. Then, this node is partitioned into two by choosing the point located at the median of a particular dimension. The two “child” nodes now continue this process recursively¹. This process terminates when either: i) the desired depth is reached, ii) the number of points in the KD-tree node falls below a certain threshold or iii) we have reached the desired number of nodes. Once the tree is built we represent each node with a hyper rectangle. All data points in the hyper rectangle are represented as the same state. The problems with a KD-tree is that all the points in a hyper rectangle may not be similar to each other and should not in fact be considered as the same state.

2) K-means: K-means is an algorithm to cluster n objects into k partitions where $k < n$. The K-means objective is to minimize the total intra-cluster variance or the squared error function.

$$V = \sum_{i=1}^k \sum_{(X_j \in s_i)} (X_j - \mu_i)^2 \quad (4.10)$$

Suppose there are k clusters $s_i, i = 1, 2, \dots, k$, and μ_i is the centroid or mean point of all the points $(X_j \in s_i)$.

The K-means algorithm first selects k initial centroids, either at random or using some heuristic. The algorithm then assigns all the points to the nearest centroid. It then calculates a new centroid for each set of clusters. In the second step, the algorithm reassigns the point according to these new centroids. These two steps are repeated until convergence, which is obtained when the points no longer change their assignments to the current clusters. Note that K-means converges to the local optima of (4.10). K-means overcomes the drawback of KD-trees by taking all dimensions into account rather than splitting one dimension at a time.

¹ There are many variations on the KD-tree splitting criterion, One may choose the mean instead of median and one may choose the dimension (i.e. axis randomly) instead of cycling through the axes in a round-robin fashion.

While using K-means for discretization, we assign each cluster as a state and represent the state with its centroid. Any new point then can easily be assigned by finding the closest centroid. The main problem in using K-means for discretization is knowing the value of k . i.e. the number of clusters.

3) X-means: X-means (Pelleg and Moore, 2000) is an extension to K-means that determines the value of k using the data. It searches the space of cluster locations and the number of clusters by optimizing a model selection criterion such as the Bayesian Information criteria (BIC), which introduces a high penalty term for a model having a large number of parameters in order to avoid overfitting. The algorithm first applies traditional K-means to the data. In the next step X-means tries to determine k . The algorithm does so by deciding which cluster to split and if splitting does in fact improve the model selection criteria. These two steps are repeated until no improvement in the model selection criteria is observed.

4.3 Graphical Models.

“Graphical models are a link between probability and graph theory” (Pearl, 1988). They provide a natural tool for dealing with uncertainty and complexity. Probability theory ensures that the system as a whole is consistent and provides a way to connect models to data. The graph theoretic side of graphical models provides a graphical modeling language that is easy for humans to use for general purpose algorithms. Examples of graphical models include HMM (Rabiner, 1990), Kalman Filters (Kalman, 1960), CRFs (Sutton and McCallum, 2006) and Bayesian Networks (Pearl, 1988). Probabilistic graphical models are graphs in which nodes represent random variables. The arc represents the dependencies between the variables. There are two major categories of graphical models: directed and undirected graphical models.

4.3.1) Directed Graphical models: In directed graphical models, one can very informally interpret an arrow from A to B as “A causes B”. For example, weather causes the condition of the ground. i.e. cloudy whether causes the ground to be wet.

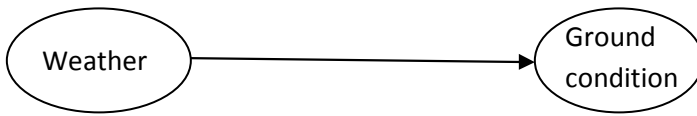


Figure 4.1. Directed model

The probability distribution is given as,

P(weather = sunny)	P(weather = cloudy)
0.5	0.5

Weather

P(GC/weather)	P(GC = wet)	P(GC = dry)
Weather = sunny	0.1	0.9
Weather = cloudy	0.9	0.1

Ground condition

Table 4.2. The conditional Probability table for the directed model in figure 4.1.

Each node encapsulates the conditional probability $p(X_i | parents(X_i))$ where $parents(X_i)$ are the parents of X_i in the graph. In example above, the conditional probability distributions are represented as tables. By the chain rule, the joint probability of the models is given by, $p(W, GC) = p(W) * p(GC | W)$. the examples of directed graphical models are HMM, Linear Dynamic System (LDS), Bayesian Networks and Kalman Filters (KF). We discuss HMMs and LDS briefly.

1) Hidden Markov Models: Suppose we have a system which may be in one of N possible states s_1, s_2, \dots, s_N . The system undergoes set of random transitions from one state to another. The probability that the system is in state 'A' and will go in state 'B' is called a transition probability. Suppose that the timestamp is denoted as $t = 1, 2, \dots, T$ and the actual state at time t is s_t . In the simplest form, the system state X_t depends only on the previous state X_{t-1} . This type of model is known as first order Markov chain since the current state only depends on the previous state. Also, we can say that X_t is independent of all other previous states except X_{t-1} . We can define the probability of the system being in state s_t at time t as,

$$p(X_t = s_t | X_{t-1} = s_{t-1}, X_{t-2} = s_{t-2}, \dots, X_{t-k} = s_{t-k}) = p(X_t = s_t | X_{t-1} = s_{t-1}) \quad (4.11)$$

We refer to probabilities of the form in Equation 4.11 as transitional probabilities. For compactness of notation, we represent the transition probability of moving from state s_j to state s_i as a_{ij} where $a_{ij} \geq 0$ and $\sum a_{ij} = 1$. So far we have seen a Markov model in which every state corresponds to an observable state. In HMM the observation is a probabilistic function of the state.

A HMM is characterized by the following properties:

1) N - the number of states in the model. Let the state take on N possible values. i.e. $\mathbf{S} = \{s_1, s_2, \dots, s_N\}$

2) M - the number of observable symbols per state. Let the symbols per state be $\mathbf{V} = \{v_1, v_2, \dots, v_M\}$.

3) The state transitional probabilities $\mathbf{A} = \{a_{ij}\}$,

4) The observation symbol probability distribution in state j ,

$$B_k = p [Y_t = v_k | X_t = s_i] .$$

5) The prior distribution, $\mathbf{C} = p(X_1 = s_j)$.

Let the model be $\lambda = \{S, V, A, B, C\}$. We can use the information given by λ to generate the sequences given the initial state, to find the likelihood of the sequence given the model λ . As a HMM is very useful for temporal sequences, we can illustrate the HMM by unrolling the time slices. Here node X s are the hidden states whereas Y s are the observation nodes.



Figure 4.2. Unrolling the hidden Markov model in time slice. X can take values from S . Y can take any values in V .

Given the observation $Y = Y_1 Y_2 \dots Y_T$ and a model λ , we can calculate the probability of observed sequence given the model $p(Y | \lambda)$ to find how likely the given sequence is. In this thesis we address the above which is of more interest in classification of mocap data. If we are given a sequence of observation Y . and we are required to find the likelihood of the sequence given the model λ , we can find the likelihood as,

$$p(Y|S, \lambda) = b_{x_1}(Y_1) \cdot b_{x_2}(Y_2) \dots b_{x_T}(Y_T) \text{ and } p(X | \lambda) = C_{x_1} \cdot a_{x_1, x_2} \dots a_{x_{T-1}, x_T}. \quad (4.12)$$

Since we are dealing with pure motion sequences, there are no hidden state transitions.

Discrete and Continuous HMM:

According to the nature of the observable state we can have discrete and continuous HMMs.

The HMM that has observable state as finite and discrete is called as a discrete HMM. Model with continuous and infinite state values are considered to be continuous HMMs. When hidden states are also continuous we call such a model a Linear Dynamics System (LDS). We will see a LDS briefly in the next section.

2) Linear Dynamic Systems: HMMs and LDSs (Minka, 1999) are based on the same assumption: a hidden state variable. Both have the same independence diagram, learning and inference algorithms. The difference is that a HMM uses discrete state hidden variables and a LDS uses continuous state hidden variables with linear Gaussian dynamics. In this section we briefly discuss the similarities and differences between LDSs and HMMs. For a detailed explanation refer to (Minka, 1999). In both cases we specify a joint probability distribution over hidden states and observations. The independence diagram is the same as for a HMM,

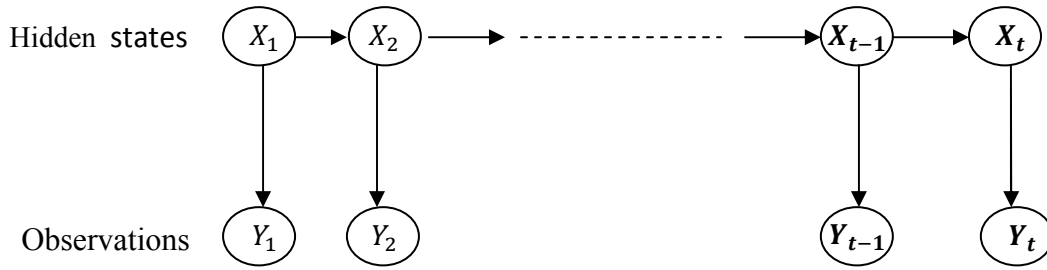


Figure 4.3. Graphical representation for linear dynamics system.

The joint distribution of the above dependence diagram is given as.

$$p(\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_T, \mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_T) = p(\mathbf{X}_1) \cdot p(\mathbf{Y}_1 | \mathbf{X}_1) \cdot \prod_{t=2}^T p(\mathbf{X}_t | \mathbf{X}_{t-1}) \cdot p(\mathbf{Y}_t | \mathbf{X}_t) \quad (4.13)$$

We will focus on using a LDS for classification problem. In HMM, as state variables are discrete, the integrals become sums. This leads to a HMM forward-backward propagation algorithm. To get a linear time algorithm, we must have a constant number of parameters; otherwise the integrals take time proportional to T . The only distribution with this property is an exponential family distribution. A Gaussian belongs to this family. Let us consider the state X_t to have a Gaussian distribution conditioned on X_{t-1} as follows:

$$p(\mathbf{X}_t | \mathbf{X}_{t-1}) \sim \mathcal{N}(\mathbf{A} \cdot \mathbf{X}_{t-1}, \mathbf{\Gamma}) \quad (4.14)$$

$$p(\mathbf{Y}_t | \mathbf{X}_t) \sim \mathcal{N}(\mathbf{C} \cdot \mathbf{X}_t, \mathbf{\Sigma}); \quad (4.15)$$

We can write a LDS as a set of linear equations,

$$\left. \begin{aligned} \mathbf{X}_t &= \mathbf{A} \cdot \mathbf{X}_{t-1} + \mathbf{W}_t; \\ \mathbf{W}_t &\sim \mathcal{N}(0, \mathbf{\Gamma}); \\ \mathbf{Y}_t &= \mathbf{C} \cdot \mathbf{X}_t + \mathbf{V}_t; \\ \mathbf{V}_t &\sim \mathcal{N}(0, \mathbf{\Sigma}); \end{aligned} \right\} \quad (4.16)$$

The parameters $(\mathbf{A}, \mathbf{C}, \boldsymbol{\Sigma}, \boldsymbol{\Gamma})$ depend on time t . We can use a Kalman filter (Kalman, 1960) and Kalman smoothing (Kalman, 1960) to find the forward backward propagation similar to forward-backward algorithm of HMM.

4.3.2) Undirected Graphical Models: Undirected graphs are also defined as $G = (V, E)$. Where V is a set of vertices and E represents a set of edges. We describe Conditional Random fields (CRF) in this section.

1) Conditional Random Fields: A CRF (Sutton and McCallum, 2006; Wallach, 2004) is a discriminative model, meaning it models the conditional likelihood $P(X | Y)$, which is used to discriminate between the labels X conditioned on observing an input sequence Y . In contrast, a HMM is a generative model that models the joint likelihood $P(X, Y)$ which generated the motion sequence. In classification tasks CRFs have an advantage over HMMs by focusing on the discriminative aspects of the data allow it to distinguish between class labels while HMMs focus on modeling the distribution that generated the data. A CRF corresponding to a HMM is illustrated below:

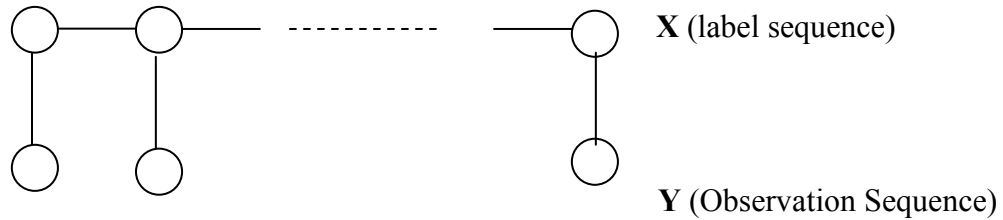


Figure 4.4. Unrolled conditional random fields

Now, let us combine the advantages of discriminative model and sequence modeling. This yields a linear chain CRFs. The joint distribution of HMM is given by,

$$p(\mathbf{X}, \mathbf{Y}) = \prod p(\mathbf{X}_t | \mathbf{X}_{t-1}) \cdot p(\mathbf{Y}_t | \mathbf{X}_t), \text{ where } p(\mathbf{X}_1 | \mathbf{X}_0) = p(\mathbf{X}_1) \quad (4.17)$$

We can write the equivalent CRF conditional distribution $P(X | Y)$ as,

$$p(\mathbf{X} | \mathbf{Y}) = \frac{p(\mathbf{X}, \mathbf{Y})}{\sum_{\mathbf{X}'} p(\mathbf{X}', \mathbf{Y})} = \frac{\exp \{ \sum_{k=1}^K w_k \cdot f_k(\mathbf{Y}, \mathbf{X}, \bar{\mathbf{X}}) \}}{\sum_{\mathbf{X}'} \exp \{ \sum_{k=1}^K w_k \cdot f_k(\mathbf{Y}, \mathbf{X}, \bar{\mathbf{X}}) \}} \quad (4.18)$$

Where K is the total number of features.

Parameter estimation: Parameter estimation is performed by penalized maximum likelihood. We optimize the following conditional log likelihood,

$$l(\mathbf{Y}) = \sum_{i=1}^N \log\{ p(\mathbf{X}_i | \mathbf{Y}_i) \} \quad (4.19)$$

As a measure of over fitting we add the penalty term on number of parameters. Adding a penalty term to regularize eqn. (3) yields,

$$l(\mathbf{Y}) = \sum_{i=1}^N \log\{ p(\mathbf{X}_i | \mathbf{Y}_i) \} - \sum_{k=1}^K \left(\frac{\lambda_k}{\sigma^2} \right) \quad (4.20)$$

In general, the function cannot be maximized in closed form; hence we use an iterative convex optimization method such as L-BFGS (Nocedal, 1980).

Inference: Inference for linear chain CRFs can be performed efficiently using dynamic programming. Refer to section 1.3.3 of (Sutton, McCallum, 2006) for more details.

5. Experimental data and preprocessing

In this section we describe the data used in our experiments and pre-processing steps that were required.

5.1 Data Description

Source of Data

The mocap that we used was from the CMU Graphics Lab Mocap database (CMU Mocap Library). This mocap data was created for research and commercial use and is publicly available; There are 2605 trials in 6 categories (Human Interaction, Interaction with environment, Locomotion, Physical activities and sports, Situations, Test motions) with 23 subcategories available. The mocap was performed at CMU including 144 subjects. For the purpose of modeling and classifying the motion data we use the BVH file format. We focus on motions with a single subject and pure motions, where by a pure motion; we mean that the motion consists of a single motion rather than a mixture of motions in a single trial.

Set of motions

After selecting the motions appropriate for our classification problem we were left with the motions as listed in table 5.1. We selected the number of training motions in order to have a reasonably balanced dataset and avoid the problem of severely imbalanced classes.

Serial Number	Motion	Total motions	Training motions	Testing Motions	Average no of frames per motion.
1	Basketball dribble	12	6	6	857
2	Bend	13	7	6	417
3	Boxing	9	5	4	3427
4	Cartwheel	8	4	4	453
5	Dance	29	14	15	931
6	Golf picking ball	10	5	5	471

Serial Number	Motion	Total motions	Training motions	Testing Motions	Average no of frames per motion.
7	Golf placing tee	10	5	5	444
8	Golf placing ball	10	5	5	428
9	Golf putt	10	5	5	416
10	Golf swing	20	10	10	404
11	Hop	19	9	10	435
12	Jump	32	16	16	514
13	March	10	5	5	446
14	Punch	8	4	4	146
15	Run	30	15	15	153
16	Run Adventure	34	17	17	331
17	Sit stand	11	5	6	2183
18	Skateboard	15	8	7	281
19	Swimming	18	9	9	2463
20	Walk	43	20	23	735
21	Walk action	34	15	19	1110
22	Walk baby	26	12	14	904
23	Walk martial	12	6	6	2264
24	Walk MJ	14	7	7	1098
25	Walk obstacle1	14	7	7	436
26	Walk obstacle2	28	14	14	368
27	Walk slope	8	5	3	487
28	Walk stylish	40	20	20	2766
29	Walk uneven ground	32	16	16	3683
30	Walk weird	40	20	20	1790
Total	30	565	278	287	1221

Table 5.1 Data description. This table describes the motions from the CMU mocap database that were suitable for our experiments.

Experiments:

We created a set of 8 experiments to assess the performance of various graphical models. We chose the data for these 8 experiments so that the classification tasks seemed sufficiently difficult and there were a challenging mix of motion types. A description of each experiment can be found in table 5.2

In all the experiments except for experiment 8, we made sure that data from the same subject was present in both the training and testing data. Between-subject variability is a challenging problem in motion capture analysis. In experiment 8, we allow data from different subjects to be randomly assigned to the training or testing data. The purpose of experiment 8 was to determine the robustness of the algorithms to differences between subjects.

Exp number	Experiment description(Motions included)
1	Dance, Hop, Jump, Punch, Run and Walk
2	Hop, Jump, Punch, Run, Sit-Stand and Swimming
3	Basketball dribble, Cartwheel, Dance, Hop, Jump, March, Punch, Run and Walk.
4	Jump, Run and Walk
5	Walk action, Walk like a baby, Martial art walk, MJ walk, Walking on obstacle, Walking on a slope and Walking on an uneven ground.
6	Run, Run Adventure, Walking on a obstacle, Stylish walk and Weird walk
7	Golf pickingball, Golf placingball, Golf placingtee, Golf putt, Golf swing, March and Sit-Stand
8	Bend, Boxing, Punch, Skateboard, Swimming and Walking on a uneven ground

Table 5.2 Experiment description.

5.2 Preprocessing

The BVH data is 96 dimensions with the first 6 dimensions corresponding to root joint translation and rotation. The remaining 90 dimensions are the X , Y , Z rotation angles for each of the 30 joints. In total, we have 6 dimensions (root joint) + 30 joints * 3 dimensions per joint (for the X , Y , Z rotation angles) = 96 dimensions. The rotations are

defined in Euler angle notation. In data preprocessing we first remove the root joint and then convert the Euler angle notation to Axis of angle notation.

1) Removal of root joint

The root joint contains information about the absolute body position and body orientation information. Excluding root joints makes all the motion independent of the specific body position and orientation. For our experiments absolute body positions is not necessary. In other experiments, such information may be useful.

2) Conversion to Axis of Angle notation

The different types of rotational angle representations include Euler angles, Axis of angle and Quaternions. We will not discuss the quaternion representation here.

i) Euler Angle: Leonhard Euler proved that any 3D rotation can be expressed as three basic rotation about the coordinate axes. Usually a right handed coordinate system is used as a reference. The order decides the sequence in which rotations will be applied to the respective axis. The order X, Y, Z means that the X rotation will be followed by Y which will then be followed by Z . In BVH format, the order is Z, X, Y . Let a_x, a_y, a_z be the amount of rotation around the x, y and z axes respectively. Let $R_i(X)$ be the function that rotates the object in the i^{th} dimension by an amount X , where the i^{th} dimension is either the $X, Y, \text{ or } Z$ dimensions. The angle values are specified in radians or degrees. The effective result for the rotation order of Z, X, Y would look like the following,

$$R_{zxy}(a_x, a_y, a_z) = R_z(a_z)R_x(a_x)R_y(a_y) \quad (5.1)$$

The rotation matrices are given as follows,

$$R_x(a_x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(a_x) & -\sin(a_x) \\ 0 & \sin(a_x) & \cos(a_x) \end{pmatrix} \quad (5.2)$$

$$R_y(a_y) = \begin{pmatrix} \cos(a_y) & 0 & -\sin(a_y) \\ 0 & 1 & 0 \\ \sin(a_y) & 0 & \cos(a_y) \end{pmatrix} \quad (5.3)$$

$$R_z(a_z) = \begin{pmatrix} \cos(a_z) & -\sin(a_z) & 0 \\ \sin(a_z) & \cos(a_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5.4)$$

Let $C_x = \cos(a_x)$; $C_y = \cos(a_y)$; $C_z = \cos(a_z)$; $S_x = \sin(a_x)$; $S_y = \sin(a_y)$; $S_z = \sin(a_z)$

$$R_{zxy}(a_x, a_y, a_z) = \begin{pmatrix} C_z \cdot C_y & -S_z \cdot C_x + C_z \cdot S_y \cdot S_x & S_z \cdot S_x + C_z \cdot S_y \cdot C_x \\ S_z \cdot C_y & C_z \cdot C_x + S_z \cdot S_y \cdot S_x & -C_z \cdot S_x + S_z \cdot S_y \cdot C_x \\ -S_y & C_y \cdot S_x & C_y \cdot C_x \end{pmatrix} \quad (5.5)$$

Euler angles suffer from a “gimbal lock” problem. A Gimbal lock occurs when two of the three pivoted supports that allow rotation (gimbal) needed to compensate for the rotations in the three dimensional space are driven to the same direction. Another explanation would be that a gimbal lock is a phenomenon of two rotational axes pointing in the same direction.

For example, suppose we rotate the object around the X axis by 90 degrees then Y by 90 degrees and finally 90 degrees in the Z direction. If we put these rotations in eqn. (5.5) we get the effective result as rotating 90 degrees in the Y direction: $R_{zxy}(90, 90, 90) = R_y(90)$. This means that we can achieve the same results by rotating 90 degrees in the Y direction.

To avoid the above problem we need to convert the Euler angle representation to some other format such as the Axis of Angle representation and the quaternion representation. We will discuss only the Axis of Angle representation here and the conversion procedure.

ii) Axis of Angle: Any 3D rotation can be specified by an axis and the angle of rotation around the axis. The axis is defined as the unit vector. The raw BVH data is found in Euler angle representation. After the removal of 6 root joint dimensions we are left with 90 dimensions corresponding to 30 joints. We have Z , X and Y rotations for each joint. The task is to convert the Euler angle to Axis of angle notation. Axis of angle is represented by a unit vector axis and the amount of rotation about the axis. In total, we have 4 dimensions. At the end of the preprocessing step we have $90 \times 4/3 = 120$ dimensions. Now, every joint angle is specified by unit vector and the amount of rotation. An example of how to convert from Euler angles to axis of angle is as follows:

Let,

$$C_1 = \cos (Y/2)$$

$$C_2 = \cos (Z/2)$$

$$C_3 = \cos (X/2)$$

$$S_1 = \sin (Y/2)$$

$$S_2 = \sin (Z/2)$$

$$S_3 = \sin (X/2)$$

Then,

The angle of rotation is given by,

$$\text{Angle} = 2 * \cos (C_1.C_2.C_3 - S_1.S_2.S_3);$$

$$X = S_1. S_2. C_3 + C_1. C_2. S_3;$$

$$Y = S_1. C_2. C_3 + C_1. S_2. S_3;$$

$$Z = C_1. S_2. C_3 - S_1. C_2. S_3;$$

To find the unit vector we divide the X , Y , Z by,

$$\text{magnitude} = \sqrt{X^2 + Y^2 + Z^2}$$

$$X = X / \text{magnitude};$$

$$Y = Y / \text{magnitude};$$

$$Z = Z / \text{magnitude};$$


After conversion we have 120 dimensions instead of 96 dimensions. As the preprocessing step is independent of the actual experiment we convert all the data to Axis of angle notation and use the preprocessed data in our experiments.


5.3 Generic pipeline


The high dimensional nature of the data makes the classification procedure computationally challenging. Also in human motion data, hence dimensionality reduction is necessary. We explore PCA (a linear technique) and GPLVM (a non-linear technique).


We explore both discrete and continuous models for representing mocap data. We use **Gaussian Markov process** (for continuous data) and **Markov model** (for discrete data) to incorporate temporal characteristics of the data. Furthermore, the order of the Markov model is yet another variation in the temporal models. We consider 1st order and 2nd order Markov process for the human motion classification process. Markov models are generative models and CRFs are the discriminative counterpart of Markov models.

These different combinations have been evaluated on all 8 experiments. Figure 6.1 shows a graphical representation the basic modules in the experiments. Here is the meaning of each graphical object.

Solid Arrow (): Shows the flow of data.

Dotted Arrow (): Parameters generated by the module at the tail of the arrow can be use by the module pointed to by the head.

Curved edge Rectangle (): Each curved edge rectangle describes a module in the whole pipeline. Input and output data is shown as entering and exiting arrows. Dotted arrows show the generation of parameters.

Filled diamond (): It describes that one of the possible path must be taken. However, if you have chosen the discrete path during training then you must choose the discrete path during testing.

Below, we provide an overview of the entire pipeline.

Preprocessing: Input to this module is raw data (96 dimensions) and output is data in Axis of angle representation (120 dimensions.)

Dimensionality Reduction: This step takes input as 120 dimensional data. We can choose the DR method to be PCA or GPLVM. The DR step generates the parameters for reduction that is applied to the test data.

Discretization(X means): Input to this module is the continuous data with dimensions d ($d \ll 120$). The parameter d is determined in the DR step. The output of this module is the discrete data. This module generates the k centroids, which can be used to discretize the test data. (here k is the number of states found by the X-means algorithm with BIC).

Training: Input to this module is continuous/ discrete data. Depending on the selection, one of the possible algorithms is chosen. The output is a trained classifier that is then used in the testing phase. Possible algorithms are:

- **For Continuous :**
 - i) Gaussian Markov process (order 1)
 - ii) Gaussian Markov process (order 2)
 - iii) Conditional Random field (order 1)
 - iv) Conditional Random field (order 2)

- **For Discrete :**
 - i) Gaussian Markov process (order 1)
 - ii) Gaussian Markov process (order 2)
 - iii) Conditional Random field (order 1)
 - iv) Conditional Random field (order 2)

Testing: The testing module receives the model created by the training phase and also the testing data. Testing data can be continuous or discrete depending upon the selection

done by the user during the training phase. The testing phase generates the confusion matrix for the corresponding experiment along with the and overall accuracy.

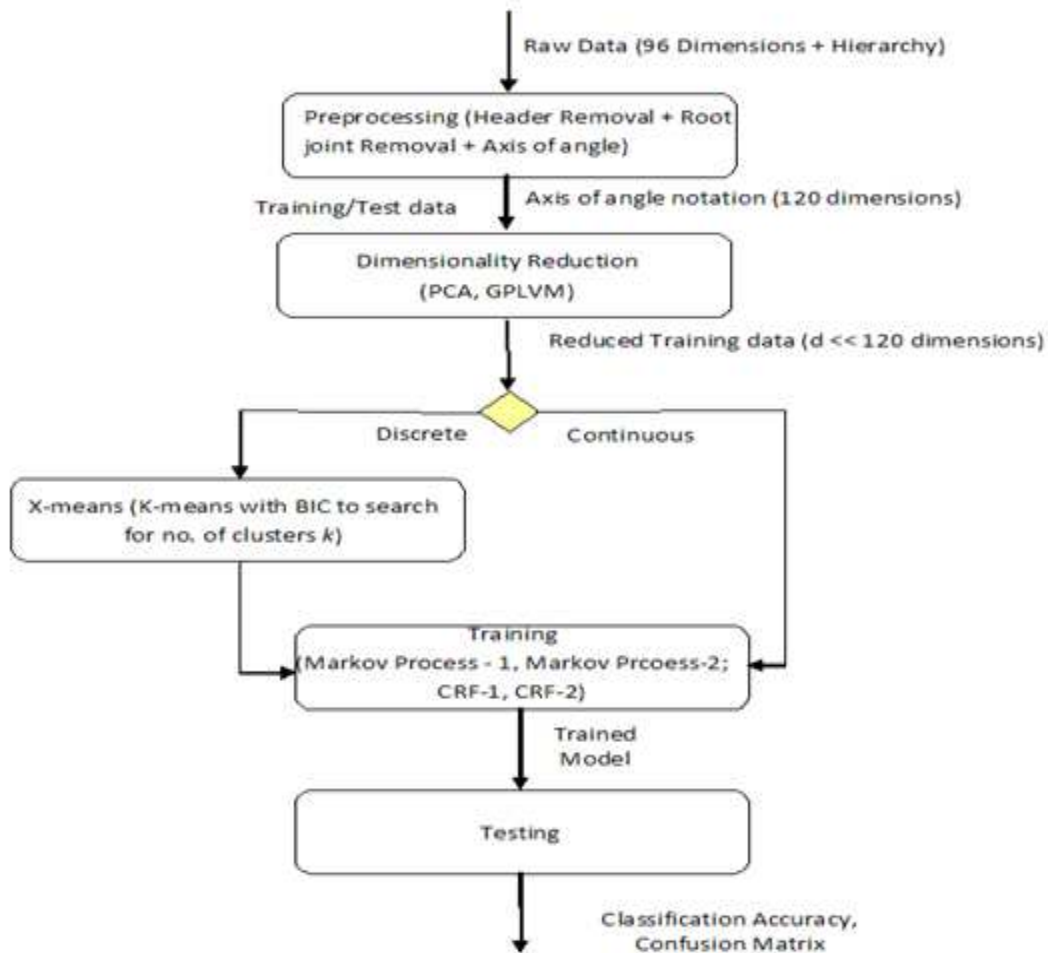


Fig 5.1 Mocap Classification pipeline. See the description in the text for the meaning of each graphical object.

As preprocessing has been already discussed we will now discuss the other modules in detail. We will start discussing dimensionality reduction in this chapter. Discrete and continuous methodologies will be discussed in later chapters.

5.4 Dimensionality reduction

This section discusses the details that we encountered during experiments.

5.4.1 Principal Component Analysis (PCA):

Input: 120 dimensional output of preprocessing step.

Step I) Merge: To ensure the consistency of state space through all the motions we merge all the training data together. Let \mathbf{Y} be the $(n \times 120)$ dimensional data $\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_T)$ where T is the total number of frames for merged data and \mathbf{Y}_i is a D -dimensional vector.

Step II) Centering of data: As a part of PCA we first subtract the empirical mean of data from \mathbf{Y} . Let μ be vector of size (1×120) as the empirical mean of \mathbf{Y} . Let $\hat{\mathbf{Y}} = \mathbf{Y} - \mu$.

Step III) Eigenvalues: Next we calculate the covariance matrix Σ for centered the data. The size of Σ is (120×120) . We then calculate the eigenvalue (\mathbf{E}_{val}) and eigenvector (\mathbf{E}_{vect}) for the covariance matrix Σ . Arrange these matrices in descending order of eigenvalues. Let us call these matrices \mathbf{E}'_{val} and \mathbf{E}'_{vect} for eigenvalue and eigenvector respectively.

$$\mathbf{E}'_{val} = \begin{pmatrix} e_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & e_{120} \end{pmatrix}; \quad \mathbf{E}'_{vect} = \begin{pmatrix} v_{1,1} & \cdots & v_{1,120} \\ \vdots & \ddots & \vdots \\ v_{120,1} & \cdots & v_{120,120} \end{pmatrix}$$

Where, $e_1 > e_2 > e_3 > \dots > e_{120}$

Step IV) Selecting d (the number of reduced dimensions): Most of past research (Barbic, Safanova, Pan, Hodgins, and Pollard, 2004) use the criteria which selects d such that 90% of information is retained. This criteria is given by equation (5.6)

$$r = \frac{\sum_{i=1}^d e_i}{\sum_{i=1}^{120} e_i} \quad (5.6)$$

We select the minimum d such that, $r \geq 0.7$ (i.e. 70% information). Once d is found, we select only the eigenvalues from $e_1 > \dots > e_d$ and their respective eigenvectors. We discard the rest of the eigenvalues and eigenvectors. Let the truncated eigenvector matrix be \mathbf{A} of size $(n \times d)$.

Step V) Reduce the data: Finally the reduced data is Y is found by,

$$\mathbf{X} = (\mathbf{A}^T \cdot \hat{\mathbf{Y}})^T \quad (5.7)$$

\mathbf{X} is the final data with $(n \times d)$ as dimension. The empirical mean $\boldsymbol{\mu}$, and the $d \times 120$ matrix of principal components \mathbf{A} . are the parameters stored for future use. These parameters are used to reduce the Test data.

Step VI) Reduce the test data: We use the parameters stored in (step V) to reduce the dimensionality of the test data. Let \mathbf{Y}' be the new test data set and \mathbf{X}' be the reduced test data. Then \mathbf{Y}' is discretized as follow,

$$\mathbf{Y}' = \mathbf{Y}' - \boldsymbol{\mu}; \quad (5.8)$$

$$\mathbf{X}' = (\mathbf{A}^T \cdot \mathbf{Y}')^T \quad (5.9)$$

5.4.2 Gaussian Process Latent Variable model (GPLVM):

We used the MATLAB implementation available from (Lawrence, 2005). Here we discuss the steps that we took to reduce the data using GPLVM.

Step I) Merge: We decided to use the same method that we used for PCA. GPLVM calculates a gigantic kernel matrix of size $(n \times n)$ (where, n is the total number of frames). It is infeasible to keep such a big amount of data in memory. This limitation forced us to down sample the data by a factor of 8 frames. Let \mathbf{Y} be the merged training data.

Step II) Centering of data: At this step we receive the sampled data from step I. Similar to PCA we find the empirical mean of the data and subtract it from the data.

Step III) Selecting d (reduced dimensions): It is infeasible to search over d in the GPLVM setting because it is too computationally expensive. Hence, we used the criterion used by the PCA experiments to determine d .

Step IV) Learning the GPLVM model. GPLVM involves maximizing the likelihood given in equation (4.7 and 4.9). We used Scaled Conjugate Gradient (SCG) for optimization. (Refer (Scaled Conjugate Gradient,1995)) for detailed explanation.

We learn the hyper parameters θ and find the latent space representation of \mathbf{Y} . We call \mathbf{X} the latent dimensional representation. The model along with θ , \mathbf{X} and \mathbf{Y} are saved for future use. Also, \mathbf{K}_Y^{-1} , a kernel matrix learned through optimization is saved so that it can be used in the future and need not be recomputed during the inference of a new point.

Step V) Reducing the training data. Recall that in (step I) we down sampled the training data by the factor of 8. We use equation (14, 15, 16 of (Urtasun and Darrell,2007)) to reduce all the frames of the training data which we missed during step I. As optimization requires optimization and saved parameters, we use θ , \mathbf{X} , \mathbf{Y} and \mathbf{K}_Y^{-1} learned in (step IV). The missing frames of step I can now be reduced to d dimensions.

Step VI) Reducing testing data. The same method is used as in Step V. Here the input data is testing data instead of training data.

6. Discrete modeling

Let us revise some notations that will be used throughout this chapter. Let, $\mathcal{X} = \{\mathbf{X}^{(i)}, m^{(i)}\}_{i=1}^N$ be the training data, where each $X^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \dots, x_T^{(i)}\}$ be the sequence of discretized training input. Let M be a vector of motion label such that, $\mathbf{M} = \{m^1, m^2, \dots, m^N\}$. Let $\mathbf{X}' = \{x'_1, x'_2, \dots, x'_T\}$ be a motion sequence in the test data where X'_i represents the i^{th} frame of test sequence. Let m be the number of motions in the experiment and k be the number of discrete states found during X-means.

The first step in discrete modeling is to discretize the continuous state space using X-means, which produces k centroids.

We need to convert \mathbf{X}' into the discrete states defined by X-means during training. To do so, we assign each x'_i to the nearest of the k centroids, where Euclidean distance is used as a measure of proximity. Note that the centroids are represented by a d dimensional vector. The rest of the chapter is divided into sections for generative and discriminative models.

6.1 Generative models.

Figure 6.1 shows the 1st and 2nd order generative models. The top layer of each model corresponds to the motion label assigned to the entire sequence. These models are [Markov models of pure motions, hence there are no transitions between each motion label.]

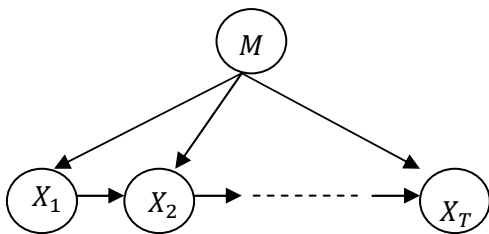


Fig 6.1 (a). First order generative model for discrete approach.

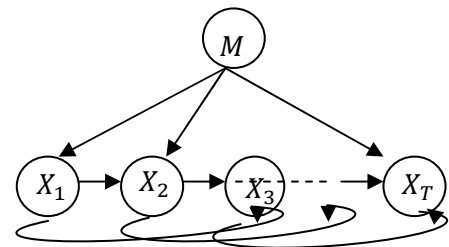


Fig 6.1 (b). Second order generative model for discrete approach.

6.1.1 First order model: Figure 6.1(a) illustrates a first order Markov model. This model requires prior probabilities $p(\mathbf{X}_1 | \mathbf{M})$ and $p(\mathbf{M})$ as well as transition probabilities $p(\mathbf{X}_t | \mathbf{X}_{t-1}, \mathbf{M})$. These probabilities are learned through Maximum Likelihood Estimation (MLE). We assume a uniform prior for $P(\mathbf{M})$. For the transition probabilities, MLE effectively “counts” the transitions between states, conditioned on the motion label. We use Laplacian smoothing to deal with the zero counts.

To classify a motion sequence, we need to compute

$$\begin{aligned} & \underset{m}{\operatorname{argmax}} P(M = m | \mathbf{X}') \\ &= \underset{m}{\operatorname{argmax}} \frac{P(\mathbf{X}' | M=m) P(M=m)}{P(\mathbf{X}')} = \underset{m}{\operatorname{argmax}} P(\mathbf{X}' | M = m) P(M = m) \\ &= \underset{m}{\operatorname{argmax}} P(\mathbf{X}' | M = m) \text{ (due to uniform priors)} \end{aligned}$$

As a result, we need to compute the likelihood of the test sequence with respect to each motion m . The new motion sequence is assigned the label of the motion with the highest likelihood. As before let \mathbf{X}' is a motion sequence in the test data, then the log-likelihood $l_m(x'_1, x'_2, \dots, x'_T)$ of the m^{th} motion is calculated as:

$$l_m(x'_1, x'_2, \dots, x'_T) = \log p(x'_1 | m) + \sum_{i=2}^T \log p(x'_i | x'_{i-1}, m) \quad (6.1)$$

6.1.2 Second order models: In the first-order models we assumed that the current observation only depends on the previous observation. We now modify the model such that the current observation depends on the past two observations. Such an assumption increases the power of the model but it also increases the number of parameters in the model. We could also increase the order of the model further but we limited our experiments to only 1st and 2nd order models. In fact, we tried the results with 3rd order models but there was no significant improvement in the results. Figure 6.1(b) illustrates the 2nd order generative models.

As in the first order case, we have prior probabilities $P(X_1 | M)$ and $P(M)$ to learn. The two changes with a 2nd order model is in an additional prior probability $P(X_2 | X_1, M)$ and in the transition matrix $p(X_t | X_{t-1}, X_{t-2}, M)$. In the first order transition matrix, we had mk^2 entries. In a second order model, we have mk^3 entries to learn. As before these parameters are learned using MLE with Laplacian smoothing.

The classification procedure is similar to that of a first order Markov model. The log likelihood of the m^{th} motion $l_m(x'_1, x'_2, \dots, x'_T)$ for second order generative models is computed as:

$$l_m(x'_1, x'_2, \dots, x'_T) = \log p(x'_1 | m) + \log p(x'_2 | x'_1, m) + \sum_{i=2}^T \log p(x'_i | x'_{i-1}, x'_{i-2}, m) \quad (6.2)$$

6.2 Discriminative models.

CRFs are the discriminative counter-part of Markov models. Undirected models are described by potential functions rather than a probability distribution. However, to make it probability distribution we divide it by normalization constant. Recall from section 4.3 that the CRFs take the form of $p(\mathbf{M} | \mathbf{X}) = \frac{1}{Z} \exp\{\sum_{k=1}^K w_k \cdot f_k(m, x_t, x_{t-1})\}$ where K is the total number of features. We now describe the 1st and 2nd order discrete CRFs and show what features were used.

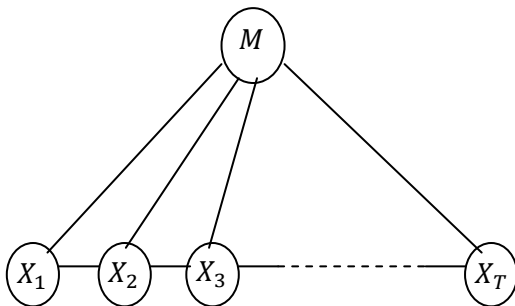


Fig 6.2 (a) First order discriminative model for discrete approach.

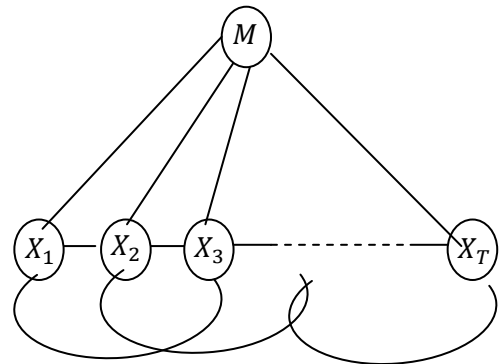


Fig 6.2 (b) Second order discriminative model for discrete approach.

6.2.1 First order models: The features in our first order model are as follows:

$$\left. \begin{aligned} f_1(m, x_t, x_{t-1}) &= I\{M = m\} \cdot I\{X_{t-1} = x_{t-1}\} ; \\ f_2(m, x_t, x_{t-1}) &= I\{M = m\} \cdot I\{X_t = x_t\} \cdot I\{X_{t-1} = x_{t-1}\} \end{aligned} \right\} \quad (6.3)$$

In the features above, x_t is the discrete state taken by the t^{th} frame and m is the label of the given motion. $I\{M = m\}$ is an indicator function taking the value 1 when $M = m$ otherwise it takes the value 0. These features are defined over the nodes in the maximal clique for each time slice. This maximal clique is shown in figure 6.3.

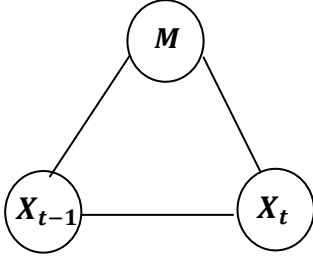


Figure 6.3 Maximal clique for first order discriminative structure shown in figure 6.2(a)

i) Training: In training, we learn the parameters w_k associated with the features. These parameters are learned through a convex optimization routine, which in our implementation is SCGD. The conditional probability is given by,

$$p(M|\mathbf{X}) = \frac{1}{Z} \exp\left\{ \sum_{t=1}^T \sum_{k=1}^K w_k \cdot f_k(m, x_t, x_{t-1}) \right\} \quad (6.4)$$

Let

$$Z = \sum_{m'} \exp\left\{ \sum_{t=1}^T \sum_{k=1}^K w_k \cdot f_k(m', x_t, x_{t-1}) \right\} \quad (6.5)$$

Then,

$$p(M|\mathbf{X}) = \frac{\exp\left\{ \sum_{t=1}^T \sum_{k=1}^K w_k \cdot f_k(m, x_t, x_{t-1}) \right\}}{\sum_{m'} \exp\left\{ \sum_{t=1}^T \sum_{k=1}^K w_k \cdot f_k(m', x_t, x_{t-1}) \right\}} \quad (6.6)$$

Then the log likelihood $l_n(X)$ for N motion datasets is given as,

$$l(\chi) = \sum_{i=1}^N \sum_{t=1}^T \sum_{j=1}^2 w_j \cdot f_j(m^{(i)}, x_t^{(i)}, x_{t-1}^{(i)}) - \sum_{i=1}^N \log Z \quad (6.7)$$

As there is no closed form solution we compute the gradient to learn the parameters. The gradient is given by (Sutton & McCallum , 2006),

$$\frac{dl}{dw_j} = \sum_{i=1}^N \sum_{t=1}^T f_j(m^{(i)}, x_t^{(i)}, x_{t-1}^{(i)}) - \sum_{i=1}^N \sum_{t=1}^T \sum_m f_j(m^{(i)}, x_t^{(i)}, x_{t-1}^{(i)}) \cdot p(m^{(i)} | x_t^{(i)}, x_{t-1}^{(i)}) \quad (6.8)$$

ii) Testing: To predict the class label, we compute

$$\begin{aligned} & \operatorname{argmax}_m P(M = m | \mathbf{X}') \\ &= \operatorname{argmax}_m \sum_{t=1}^T \sum_{j=1}^2 w_j \cdot f_j(m, x'_t, x'_{t-1}) - \log Z \end{aligned} \quad (6.9)$$

6.2. b Second order models:

The second order discriminative model we used is show in figure 6.2(b). Second order learning and testing is similar to that of first order modes but with additional parameters. Figure 6.4 shows the maximal clique for time slice t for a second order model. The features at each time slice are defined as,

$$\left. \begin{aligned} f_1(m, x_t, x_{t-1}, x_{t-2}) &= I\{M = m\} \cdot I\{X_{t-2} = x_{t-2}\}; \\ f_2(m, x_t, x_{t-1}, x_{t-2}) &= I\{M = m\} \cdot I\{X_{t-1} = x_{t-1}\} \cdot I\{X_{t-2} = x_{t-2}\} \\ f_3(m, x_t, x_{t-1}, x_{t-2}) &= I\{M = m\} \cdot I\{X_t = x_t\} \cdot I\{X_{t-1} = x_{t-1}\} \cdot I\{X_{t-2} = x_{t-2}\} \end{aligned} \right\} \quad (6.10)$$

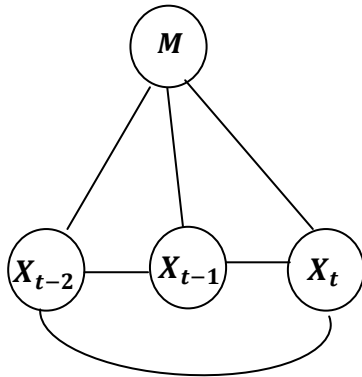


Figure 6.4. Maximal clique for second order discriminative structure shown in figure 6.2(b)

i) Training: Training is similar to first order models. The corresponding equations are given as follows:

$$p(M|\mathbf{X}) = \frac{1}{Z} \exp\left\{ \sum_{t=1}^T \sum_{k=1}^K w_k \cdot f_k(m, x_t, x_{t-1}, x_t) \right\} \quad (6.11)$$

Let,

$$Z = \sum_{m'} \exp\left\{ \sum_{t=1}^T \sum_{k=1}^K w_k \cdot f_k(m', x_t, x_{t-1}, x_{t-2}) \right\} \quad (6.12)$$

Then,

$$p(M|\mathbf{X}) = \frac{\exp\left\{ \sum_{t=1}^T \sum_{k=1}^K w_k \cdot f_k(m, x_t, x_{t-1}, x_t) \right\}}{\sum_{m'} \exp\left\{ \sum_{t=1}^T \sum_{k=1}^K w_k \cdot f_k(m', x_t, x_{t-1}, x_{t-2}) \right\}} \quad (6.13)$$

Then the log likelihood $l(\chi)$ for N motion datasets is given as,

$$l(\chi) = \sum_{i=1}^N \sum_{t=1}^T \sum_{j=1}^3 w_j \cdot f_j(m^{(i)}, x_t^{(i)}, x_{t-1}^{(i)}, x_{t-2}^{(i)}) - \sum_{i=1}^N \log Z \quad (6.14)$$

The gradient is given by eqn. (6.8) ,

ii) Testing: Predicting the class label is the same as in first order models. Log likelihood with m^{th} motion can be found as,

$$\begin{aligned} & \operatorname{argmax}_m P(M = m | \mathbf{X}') \\ &= \operatorname{argmax}_m \sum_{t=1}^T \sum_{j=1}^3 w_j \cdot f_j(m, x_t^i, x_{t-1}^i, x_{t-2}^i) - \log Z \end{aligned} \quad (6.15)$$

7. Continuous modeling

With continuous models, we can work with the continuous data as is, rather than discretizing it in a pre-processing step. In doing so, we avoid the problem of aggregating certain data points into a single discrete state when they should in fact be treated as distinct “states”. However, in order to make computation tractable, we still need to reduce the dimensionality of the data and we need to make some assumptions about the form of the data. We receive the input after it has gone through a dimensionality reduction algorithm. The number of dimensions is found by the PCA criteria described in section 5.4. In continuous modeling, we assume that each dimension is distributed as a Gaussian and that the mean of the Gaussian is a function of the value of all other dimension in the previous time step. Similar to discrete modeling discussed in previous chapter we discuss the 1st and 2nd order continuous models.

We now define the notation used throughout this chapter. Let $\mathcal{X} = \{ \mathbf{X}^{(i)}, m^{(i)} \}_{i=1}^N$ be the sequence of training data obtained after dimensionality reduction, where each $\mathbf{X}^{(i)} = \{ \mathbf{X}_1^{(i)}, \mathbf{X}_2^{(i)}, \dots, \mathbf{X}_T^{(i)} \}$ be the sequence of T frames. As before, we simplify notation by assuming each motion sequence consists of T frames when in fact sequences do vary in their length. Each $\mathbf{X}_t^{(i)}$ is a d -dimensional vector representing a frame. Note that this vector is in the reduced dimension space R^d , where the number of dimensions d is found in the dimensionality reduction step described in section 5.4. We denote the j^{th} dimension of the t^{th} time frame and i^{th} sequence as $x_t^{j,(i)}$. Since this notation is cumbersome, we often drop by the (i) superscript when we are not referring to a specific sequence eg. x_t^j .

7.1 Generative models.

7.1.a First order models: We model motion sequences as a first order Gaussian Markov Process (GMP). Figure 7.1(a) illustrates the graphical order model for the first order GMP.

The GMP can be written as a linear equation:

$$\mathbf{X}_t = \mathbf{A} \cdot \mathbf{X}_{t-1} + \mathbf{W}_t \quad (7.1)$$

$$\text{where, } \mathbf{W}_t \sim \mathcal{N}(0, \Gamma) \quad \text{where } \Gamma \text{ is the variance of Gaussian noise } \mathbf{W}_t. \quad (7.2)$$

$$\text{i. e. } p(\mathbf{X}_t | \mathbf{X}_{t-1}) \sim \mathcal{N}(\mathbf{A} \cdot \mathbf{X}_{t-1}, \Gamma) \quad (7.3)$$

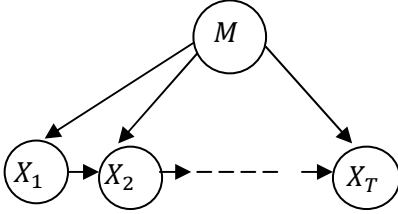


Fig 7.1(a) The first order generative model for continuous approach.

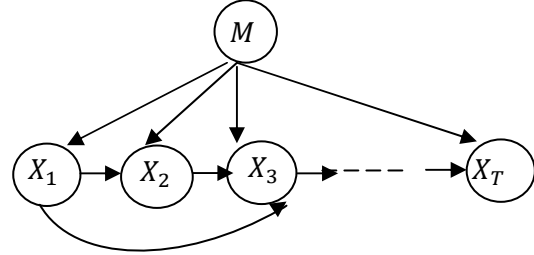


Fig 7.1(b) The second order generative model for continuous approach.

i) Training: For each dimension we find parameters for a multivariate Gaussian distribution. Let $\mathcal{N}(\mathbf{A} \cdot \mathbf{X}_{t-1}, \Gamma)$ be the Gaussian distribution for the t^{th} time slice. The overall parameters can be seen as follows:

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1d} \\ A_{21} & A_{22} & \dots & A_{2d} \\ \dots & \dots & \dots & \dots \\ A_{d1} & A_{d2} & \dots & A_{dd} \end{bmatrix} \quad (7.4)$$

$$\Gamma = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} & \dots & \Gamma_{1d} \\ \Gamma_{12} & \Gamma_{22} & \dots & \Gamma_{2d} \\ \dots & \dots & \dots & \dots \\ \Gamma_{d1} & \Gamma_{d2} & \dots & \Gamma_{dd} \end{bmatrix} \quad (7.5)$$

The likelihood is given by,

$$p(\mathbf{X}_t | \mathbf{X}_{t-1}) = p(\mathbf{X}_1) \cdot \prod_{t=2}^T \mathcal{N}(\mathbf{A} \cdot \mathbf{X}_{t-1}, \Gamma) \quad (7.6)$$

$$\mathcal{N}(\mathbf{X}_t | \mathbf{A} \cdot \mathbf{X}_{t-1}, \Gamma) = \frac{1}{(2\pi)^{d/2} \cdot |\Gamma|^{1/2}} \cdot \exp \left\{ - \frac{(\mathbf{X}_t - \mathbf{A} \cdot \mathbf{X}_{t-1})^T \cdot (\Gamma)^{-1} \cdot (\mathbf{X}_t - \mathbf{A} \cdot \mathbf{X}_{t-1})}{2} \right\} \quad (7.7)$$

We use negative log likelihood to optimize for \mathbf{A} and Γ using Scaled Conjugate Gradient (SCGD).

$$l(\mathcal{X}) = \log\{p(\mathbf{X}_1)\} + \sum_{t=2}^N \left\{ -\frac{(\mathbf{X}_t - \mathbf{A} \cdot \mathbf{X}_{t-1})^T \cdot \Gamma^{-1} \cdot (\mathbf{X}_t - \mathbf{A} \cdot \mathbf{X}_{t-1})}{2} \right\} - \log\left((2\pi)^{\frac{d}{2}} \cdot |\Gamma|^{1/2}\right) \quad (7.8)$$

ii) Testing: For prediction we calculate the log likelihood with each motion and label the motion with the highest log likelihood. The log likelihood is calculated using equation (7.8).

7.1 b Second order models: Figure 7.2(a) depicts a second order model. We can write this model as a linear equation:

$$\mathbf{X}_t = \mathbf{A} \cdot \mathbf{X}_{t-1} + \mathbf{B} \cdot \mathbf{X}_{t-2} + \mathbf{W}_t \quad (7.9)$$

Where, $\mathbf{W}_t \sim \mathcal{N}(\mathbf{0}, \Gamma)$

From this linear equation, we see that the transition function is a normal distribution i.e.

$$p(\mathbf{X}_t | \mathbf{X}_{t-1}, \mathbf{X}_{t-2}) = \mathcal{N}(\mathbf{A} \cdot \mathbf{X}_{t-1} + \mathbf{B} \cdot \mathbf{X}_{t-2}, \Gamma) \quad (7.10)$$

i) Training: For each dimension we find parameters for Multivariate Gaussian distribution. Let $\mathcal{N}(\mathbf{A} \cdot \mathbf{X}_{t-1} + \mathbf{B} \cdot \mathbf{X}_{t-2}, \Gamma)$ be the Gaussian distribution for t^{th} time slice.

In all we have d dimensions. The overall parameters can be seen as follow,

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1d} \\ A_{21} & A_{22} & \dots & A_{2d} \\ \dots & \dots & \dots & \dots \\ A_{d1} & A_{d2} & \dots & A_{dd} \end{bmatrix} \quad (7.11) \quad \mathbf{B} = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1d} \\ B_{21} & B_{22} & \dots & B_{2d} \\ \dots & \dots & \dots & \dots \\ B_{d1} & B_{d2} & \dots & B_{dd} \end{bmatrix} \quad (7.12)$$

$$\Gamma = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} & \dots & \Gamma_{1d} \\ \Gamma_{12} & \Gamma_{22} & \dots & \Gamma_{2d} \\ \dots & \dots & \dots & \dots \\ \Gamma_{d1} & \Gamma_{d2} & \dots & \Gamma_{dd} \end{bmatrix} \quad (7.13)$$

The likelihood is given by,

$$p(\mathbf{X}_t | \mathbf{X}_{t-1}) = p(\mathbf{X}_1) \cdot \prod_{t=2}^T \mathcal{N}(\mathbf{A} \cdot \mathbf{X}_{t-1} + \mathbf{B} \cdot \mathbf{X}_{t-2}, \Gamma) \quad (7.14)$$

$$\mathcal{N}(\mathbf{X}_t | \mathbf{A} \cdot \mathbf{X}_{t-1} + \mathbf{B} \cdot \mathbf{X}_{t-2}, \Gamma) = \frac{1}{(2\pi)^{d/2} \cdot |\Gamma|^{1/2}} \cdot \exp \left\{ - \frac{(\mathbf{X}_t - \mathbf{A} \cdot \mathbf{X}_{t-1} - \mathbf{B} \cdot \mathbf{X}_{t-2})^T \cdot \Gamma^{-1} \cdot (\mathbf{X}_t - \mathbf{A} \cdot \mathbf{X}_{t-1} - \mathbf{B} \cdot \mathbf{X}_{t-2})}{2} \right\} \quad (7.15)$$

We maximize the likelihood with respect to \mathbf{A} , \mathbf{B} and Γ using (SCG).

$$l(\mathcal{X}) = \log\{p(\mathbf{X}_1)\}$$

$$+ \sum_{t=2}^N \left\{ - \frac{(\mathbf{X}_t - \mathbf{A} \cdot \mathbf{X}_{t-1} - \mathbf{B} \cdot \mathbf{X}_{t-2})^T \cdot \Gamma^{-1} \cdot (\mathbf{X}_t - \mathbf{A} \cdot \mathbf{X}_{t-1} - \mathbf{B} \cdot \mathbf{X}_{t-2})}{2} \right\} - \log((2\pi)^{\frac{d}{2}} \cdot |\Gamma|^{1/2}) \quad (7.16)$$

Testing: Prediction is the same process as before. Equation (7.16) is used to find the likelihood.

7.2 Discriminative models.

7.2.1 First-order model

The Gaussian Markov Process in the previous section can be converted into a CRF by training the model parameters to maximize the conditional likelihood $p(M | X)$. An illustration of the CRF is shown in figure 7.2(a). In figure 7.2(a), the nodes $X_t^{(i)}$ are a multivariate vectors. We can convert this graphical model into an equivalent model by breaking the multivariate vector $X_t^{(i)}$ into individual components $x_t^1^{(i)}, x_t^2^{(i)}, \dots, x_t^d^{(i)}$. where $x_t^j^{(i)}$ corresponds to the value of the j^{th} dimension of the t^{th} frame of the sequence. This equivalent model is shown in 7.2(b)

We assume that for each dimension j ,

$$x_t^j \sim \mathcal{N} (w_1 \cdot x_{t-1}^1 + w_2 \cdot x_{t-1}^2 + \dots + w_d \cdot x_{t-1}^d, \sigma) \quad (7.17)$$

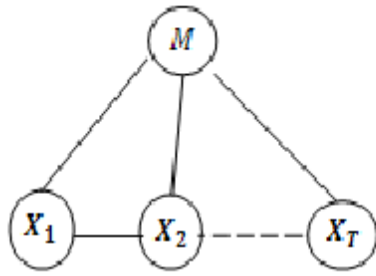


Figure 7.2(a) Shows the First order discriminative model for continuous approach. Here X_t is a d -dimensional vector.

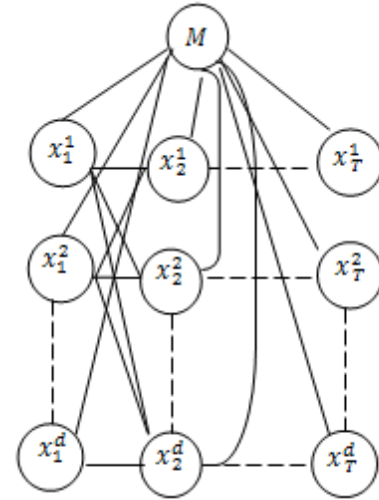


Fig 7.2(b) Shows the model corresponding to the model in 7.2(a). Note that the value of x_t^j depends on the value of all dimensions in the previous time slice.

This is the same autoregressive (AR) model used in the Gaussian Markov Process. However, to convert the GMP to an AR-CRF, we need to first define features over the maximal cliques in the graph. If we consider the maximal clique in the CRF (formed after moralization and triangulation), we get a fully connected graph with $(d+1)$ nodes. This large clique poses substantial computational challenges for learning and inference. In fact, if one re-arranges the terms of the likelihood of the data in terms of the sufficient statistics of the Gaussian distribution, the pair-wise interactions terms within a time slice and across adjacent time-slices can be considered as features. (Refer to Appendix A for the derivation of AR-CRF features) Thus, to simplify matters, we define the features over this maximal clique to be as follows:

Features for $t-1$ observation:

$$(x_{t-1}^j) \cdot (x_{t-1}^k); \text{ For } 1 \leq j \leq k \leq d.$$

We can see this feature as an upper-triangular matrix capturing the interactions between every pair of dimensions in the $(t-1)$ th frame giving $\frac{d(d-1)}{2} + d$ features .

Features for t observation:

$$(x_t^j) \cdot (x_t^k); \text{ For } 1 \leq j \leq k \leq d.$$

Similarly, this feature captures the interactions between every pair of dimensions in the t th frame giving a total of $\frac{d(d-1)}{2} + d$ features .

Features capturing the cross-interaction terms between the t th and $(t-1)$ th frame:

$$(x_{t-1}^j) \cdot (x_t^k); \text{ For } 1 \leq j \leq d; 1 \leq k \leq d.$$

There are a total of d^2 cross-interaction features.

$$\text{In total, at each time slice we have, } N_{p1} = \frac{d(d-1)}{2} + d + \frac{d(d-1)}{2} + d + d^2 = 2 \cdot d^2 +$$

d features

(7.18)

By defining the features in this way, we simplify the problem by only considering for each motion; all interactions within a time slice and between adjacent time slices. Training involves learning the weights on these features. At this point, we find the weights that maximize the conditional log likelihood through SCGD. For training and testing we use the equation (6.11 – 6.14).

7.2.2 Second order models: For second order models, each X_t^j now depends on all dimensions from the $(t-1)$ th time slice and the $(t-2)$ th time slice. The features are defined as follows:

Features for $(t-2)$ th observation:

$$(x_{t-2}^j) \cdot (x_{t-2}^k); \text{ For } 1 \leq j \leq k \leq d.$$

There are a total of $\frac{d(d-1)}{2} + d$ features.

Features for $(t-1)$ th observation:

$$(x_{t-1}^j) \cdot (x_{t-1}^k); \text{ For } 1 \leq j \leq k \leq d.$$

There are a total of $\frac{d(d-1)}{2} + d$ features.

Features for (t) th observation:

$$(x_t^j) \cdot (x_t^k); \text{ For } 1 \leq j \leq k \leq d.$$

Features based on the pair-wise cross-interaction terms between the t th and $(t-2)$ th frame:

$$(x_{t-2}^j) \cdot (x_t^k); \text{ For } 1 \leq j \leq d \text{ and } 1 \leq k \leq d.$$

There are a total of d^2 features.

Features based on the pair-wise cross-interaction terms between the t th and $(t-1)$ th frame:

$$(x_{t-1}^j) \cdot (x_t^k); \text{ For } 1 \leq j \leq d \text{ and } 1 \leq k \leq d.$$

There are a total of d^2 features.

Features based on the pair-wise cross-interaction terms between the $(t-1)$ th and $(t-1)$ th frame:

$$(x_{t-1}^j) \cdot (x_{t-1}^k); \text{ For } 1 \leq j \leq d \text{ and } 1 \leq k \leq d. \quad \text{Total } d^2 \text{ features}$$

There are a total of d^2 features.

In total, for each time slice we have,

$$N_{p2} = \frac{d(d-1)}{2} + d + \frac{d(d-1)}{2} + d + \frac{d(d-1)}{2} + d + 3 \cdot d^2 =$$

$$\left(\frac{9}{2}\right) \cdot d^2 + \left(\frac{3}{2}\right) \cdot d \text{ features}$$

(7.19)

As in the first-order models, we learn these weights by optimizing the conditional through SCGD.

8. Results and Discussion (Discrete modeling)

Due to a limited number of sequences available for training and testing, we used 2-fold cross validation to measure the classification accuracy. The Table 8.1 shows the classification accuracy obtained for discrete modeling. Table 8.1(a) reports the average accuracy for 2-fold cross validation. 8.1(b) gives the classification accuracy for fold 1 and fold 2 accuracy is reported in table 8.1(c).

Exp No.	Discrete							
	PCA				GPLVM			
	DMP-1	DMP-2	CRF-1	CRF-2	DMP-1	DMP-2	CRF-1	CRF-2
1	86.32	88.17	86.945	82.62	88.16	86.935	83.81	83.81
2	93.275	90.66	95.835	94.075	88.935	87.21	89.825	80.66
3	80.115	81.1675	79.59	75.92	45.965	53.585	47.55	42.815
4	90.355	88.45	91.39	85.51	85.62	85.565	88.395	87.63
5	92.85	93.565	96.425	93.565	82.85	84.28	79.995	82.855
6	68.03	68.02	72.09	69.785	71.51	70.345	62.2	62.2
7	97.53	100	100	96.31	79.145	81.435	81.55	77.77
8	84.34	87.445	85.36	88.51	79.37	81.495	76.265	76.265

Table 8.1(a) Classification accuracy averaged over the two folds. The best accuracy for each experiment is shown as the shaded value.

Exp No.	Discrete (fold1)							
	PCA				GPLVM			
	DMP-1	DMP-2	CRF-1	CRF-2	DMP-1	DMP-2	CRF-1	CRF-2
1	83.75	83.75	85	85	82.5	83.75	77.5	77.5
2	90	91.67	91.67	93.33	91.67	91.67	90	71.67
3	83.15	84.21	82.1	76.84	26.31	37.38	30.52	21.05
4	86.27	84.31	90.19	78.43	82.35	80.39	82.35	88.23
5	94.28	92.85	97.14	91.42	81.42	82.85	74.28	80
6	76.76	73.25	80.23	77.9	83.72	80.23	79.06	79.06
7	97.56	100	100	95.12	68.29	70.37	75.6	78.04
8	85.71	89.79	87.75	89.79	69.38	69.38	65.3	65.3

Table 8.1(b) Fold1 classification accuracy.

Exp No.	Discrete (fold 2)							
	PCA				GPLVM			
	DMP-1	DMP-2	CRF-1	CRF-2	DMP-1	DMP-2	CRF-1	CRF-2
1	88.89	92.59	88.89	80.24	93.82	90.12	90.12	90.12
2	96.55	89.65	100	94.82	86.2	82.75	89.65	89.65
3	77.08	78.125	77.08	75	65.62	69.79	64.58	64.58
4	94.44	92.59	92.59	92.59	88.89	90.74	94.44	87.03
5	91.42	94.28	95.71	95.71	84.28	85.71	85.71	85.71
6	59.3	62.79	63.95	61.67	59.3	60.46	45.34	45.34
7	97.5	100	100	97.5	90	92.5	87.5	77.5
8	82.97	85.1	82.97	87.23	89.36	93.61	87.23	87.23

Table 8.1(c) Fold2 classification accuracy.

In table 8.1 (a) , (b) , (c) , DMP – 1 means the first order discrete Markov process, DMP-2 stands for second order discrete Markov process. Similarly, CRF-1 and CRF-2 stands for first and second order conditional random fields respectively. The same sets of experiments are repeated when GPLVM was used as a dimensionality reduction method.

8.1 PCA vs. GPLVM

Number of states:

Table 8.2 shows the number of dimensions and number of states found by performing PCA, GPLVM and X-means respectively.

Exp. No	Dimensions and No of states					
	Fold 1			Fold2		
	Dimensions	states(PCA)	states(GPLVM)	Dimensions	states(PCA)	states(GPLVM)
1	9	9	9	10	15	20
2	11	24	14	10	10	12
3	11	15	3	11	11	4
4	8	17	5	7	25	13
5	9	11	4	9	10	8
6	9	8	5	9	5	2
7	8	18	5	9	16	7
8	10	16	2	10	6	3

Table 8.2 Dimensions and number of states for fold 1 and fold 2. Number of dimensions remains the same for PCA and GPLVM.

There is a noticeable difference between the number of states found when X-means is applied after PCA than when X-means is applied after GPLVM. GPVM produces fewer states than PCA. For the experiments 3 - 8 it is as low as 2 to 7. We can see the direct relation between the accuracy and the no. of states found during X-means. As the number of states increases, so does the classification accuracy. For experiment 3 (fold 1), we have accuracy as low as 37% due to only 3 states.

Wilcoxon signed rank sum test:

From table 8.1(a),(b) and (c) we can say that PCA performs better than GPLVM for discrete models. To establish statistical significance, we applied the Wilcoxon signed rank test to the 16 accuracy results for both folds 1 and 2. PCA was found to produce significantly better accuracy.

Test	Significance p
PCA-DMP-1 vs GPLVM-DMP-1	0.0209
PCA-DMP-2 vs GPLVM-DMP-2	0.0193
PCA-CRF-1 vs GPLVM-CRF-1	0.0034
PCA-CRF-2 vs GPLVM-CRF-2	0.0067

Table 8.3 shows the result of Wilcoxon signed rank sum test conducted for PCA vs. GPLVM. The first column describes the test that was conducted. The second column describes the significance level of the test.

Running time:

Table 8.4 describes the time taken to run the different algorithms on fold 2 of experiment 5. The experiment was run on a dual-core 1.6 GHz Intel processor with 2GB of memory. We can see a significant difference in the dimensionality reduction step. PCA takes around 6 minutes whereas GPLVM takes around 42 hours to reduce the data. We are using a relatively unoptimized version of GPLVM. GPLVM can be sped up through a number of methods such as the Informative Vector Machine (Lawerence, Seeger and Herbrich, 2003) or KD-trees (Bentley, 1975). Regardless of the optimizations to

GPLVM, PCA is still much faster in terms of running time. Furthermore, PCA gives better classification accuracy as shown in Table 8.19(a). For discrete models, these advantages indicate that PCA is a better choice for dimensionality reduction than GPLVM.

Exp. Detail	Experimental time for Experiment 5 (Fold2) - Discrete case									
	Dimensionality Reduction(sec)		Discretization (X-means) (sec)		Training and Testing (sec)		Total Time (Sec.)		Total Time (Hrs.)	
	PCA	GPLVM	PCA	GPLV M	PCA	GPLV M	PCA	GPLV M	PC A	GPLV M
DMP-1	173	156239	1228	1528	14	15	1415	157782	0.39	43.83
DMP-2	173	156239	1228	1528	14	15	1415	157782	0.39	43.83
CRF-1	173	156239	1228	1528	1254	1165	2655	158932	0.74	44.15
CRF-2	173	156239	1228	1528	1634	1345	3035	159112	0.84	44.20

Table 8.4 Running time for (discrete) experiment no. 5 for the 2nd fold.

8.2 Generative (DMP) vs. Discriminative (CRF).

From table 8.1 (a), we can see that when PCA was used as a dimensionality reduction method CRF-1 performs better than the Discrete Markov process. The exceptions are experiment 1 and 3 where DMP-2 outperforms CRFs. The improvement in accuracy achieved by discriminative models with discrete states was not statistically significant in most cases as shown in Table 8.5.

Test	Significance p
PCA-CRF-1 vs PCA-DMP-1	0.005
PCA-CRF-2 vs PCA-DMP-2	0.3222
GPLVM-CRF-1 vs GPLVM-DMP-1	0.3421
GPLVM-CRF-2 vs GPLVM-DMP-2	0.1118

Table 8.5 The result of the Wilcoxon signed rank sum test conducted for CRF vs. DMP discrete model.

8.3 Other Observations

Order of the models:

Table 8.6 shows the Wilcoxon signed rank sum test for first and second order models. The results show that there is no significant difference between both the models. Although there is no clear winner in terms of the order of the model, the number of parameters in the first and second order models might be a criterion for selecting between them. The first order model has, $m.K + m.K^2 = mK.(1 + K)$ parameters while the second order model has $m.K + m.K^2 + m.K^3 = mK.(1 + K + K^2)$ parameters. where K is the total no. of states and m is the total motions in the experiments. As we increase the order of the models number of parameters increases exponentially. For example Experiment 2 (PCA - fold 1) will have 3600 parameters for order 1 in comparison to 86544 parameters for the second order (here $N = 6$ and $K = 24$). This large number of parameters can be a concern if data is limited and overfitting becomes a serious issue. From the discussion in section 8.1, 8.2 we can say that combination of PCA, CRF and 1st order models attains the accuracy of (85% - 100%).

Test	Significance p
PCA-DMP-2 vs PCA-DMP-1	0.2891
PCA-CRF-2 vs PCA-CRF-1	0.0139
GPLVM-DMP-2 vs GPLVM-DMP-1	0.3077
GPLVM-CRF-2 vs GPLVM-CRF-1	0.4692

Table 8.6 The result of Wilcoxon signed rank sum test conducted for 1st order vs. 2nd order models (discrete).

Classification of similar motions:

Another interesting aspect of the results is the accuracy obtained for Experiments 5, 6, 7 and 8. The main objective of Experiments 5, 6 and 7 was to evaluate the performance of the algorithms when the motions appear to be similar but with slight variations. From a human perspective, these motions seem to be difficult to distinguish. Surprisingly,

Experiments 5 and 7 achieved the accuracy of 96.5% and 100% respectively. Experiment 6 posed some difficulties as it contained variations of run and walk eg .run, adventure run, walking on a obstacle 2, stylish walk and weird walk. Table 8.7 shows the confusion matrix for experiment 6 (CRF-1 with PCA). The accuracy was 78% (67 out of 86 motions classified correctly). For Fold 1, all 19 misclassifications were due to other motions being classified as Walk_Stylish. Similarly, for Fold 2, 55 out of 86 were classified correctly for an accuracy of 64%. Out of 31 misclassified motions, 27 were incorrectly classified as Walk_Obs2.

Randomly distributed training data:

In Experiment 8, we did not ensure that motions from a particular subject appeared in both training and test data. Instead the motions, regardless of subject, were randomly assigned between training and test data. Surprisingly we achieved maximum accuracy of 88.7%.

Actual \ Predicted	Run	Run_adventure	Walk_obs2	Walk_Stylish	Walk_weird
Run	5	0	0	10	0
Run_adventure	0	15	0	2	0
Walk_obs2	0	0	14	0	0
Walk_Stylish	0	0	0	20	0
Walk_weird	0	0	0	7	13

Table 8.7(a) Confusion matrix for Experiment 6 (CRF-1 with PCA) fold1

Actual \ Predicted	Run	Run_adventure	Walk_obs2	Walk_Stylish	Walk_weird
Run	9	1	5	0	0
Run_adventure	1	2	14	0	0
Walk_obs2	0	2	12	0	0
Walk_Stylish	0	0	4	16	0
Walk_weird	0	0	4	0	16

Table 8.7(b) Confusion matrix for Experiment 6 (CRF-1 with PCA) fold2

From the above discussion we conclude that the combination of PCA with 1st order CRF models gives the better accuracy for discrete approach. Our model was able to successfully discriminate between the similar motions and achieve the accuracy above 90% in most of the cases.

9. Results and Discussion (Continuous modeling)

As in the discrete modeling experiments, 2-fold cross validation was performed to calculate the classification accuracy due to limited training and testing data. Table 9.1 shows the classification accuracy obtained. Table 9.1(a) gives the average accuracy for 2-fold cross validation. 9.1(b) gives the classification accuracy for fold 1 and finally, fold 2 accuracy is reported in table 9.1(c).

Exp No.	Continuous Results (Average)							
	PCA				GPLVM			
	GMP-1	GMP-2	CRF-1	CRF-2	GMP-1	GMP-2	CRF-1	CRF-2
1	51.555	34.97	90.7	94.405	61.58	40.42	91.32	91.32
2	55.145	56.175	91.61	88.935	54.17	67.645	95.745	94.91
3	47.105	76.97	92.655	91.605	53.48	61.905	92.15	92.675
4	75.25	70.04	94.385	93.3	58.495	64.755	92.53	92.48
5	41.425	28.57	92.855	92.14	42.85	42.135	97.855	98.57
6	36.04	65.115	74.415	80.81	36.625	49.415	77.9	81.975
7	33.35	52.85	87.5	92.53	45.7	51.765	91.28	92.53
8	18.71	67.8	88.555	91.655	45.455	63.5	81.195	80.975

Table 9.1(a) Classification accuracy averaged over the two folds. The best accuracy for each experiment is shown as shaded value.

Exp No.	Continuous Results (Fold1)							
	PCA				GPLVM			
	GMP-1	GMP-2	CRF-1	CRF-2	GMP-1	GMP-2	CRF-1	CRF-2
1	52.5	41.25	93.75	93.75	76.25	47.5	93.75	93.75
2	51.67	41.67	86.67	91.67	58.34	76.67	96.67	95
3	45.26	78.94	90.52	88.42	48.42	50.26	92.63	93.68
4	91.25	88.23	98.03	92.15	72.54	64.7	98.03	96.07
5	32.85	34.28	100	100	42.85	62.85	97.14	97.14
6	29.06	62.79	84.88	83.72	22.09	47.67	86.04	88.37
7	31.7	70.7	100	97.56	43.9	58.53	97.56	97.56
8	20.4	63.26	87.75	91.83	63.26	65.3	83.67	93.87

Table 9.1(b) Fold1 classification accuracy for continuous models.

Exp No.	Continuous Results (Fold2)							
	PCA				GPLVM			
	GMP-1	GMP-2	CRF-1	CRF-2	GMP-1	GMP-2	CRF-1	CRF-2
1	50.61	28.69	87.65	95.06	46.91	33.34	88.89	88.89
2	58.62	70.68	96.55	86.2	50	58.62	94.82	94.82
3	48.95	75	94.79	94.79	58.54	73.55	91.67	91.67
4	59.25	51.85	90.74	94.45	44.45	64.81	87.03	88.89
5	50	22.86	85.71	84.28	42.85	21.42	98.57	100
6	43.02	67.44	63.95	77.9	51.16	51.16	69.76	75.58
7	35	35	75	87.5	47.5	45	85	87.5
8	17.02	72.34	89.36	91.48	27.65	61.7	78.72	68.08

Table 9.1(c) Fold2 classification accuracy.

In table 9.1 (a) , (b) , (c) , GMP – 1 stands for first order Gaussian Markov process, GauMP-2 stands for second order Gaussian Markov process.

9.1 PCA vs. GPLVM

Unlike discrete case, there were mixed results regarding PCA and GPLVM. Experiments 1, 4, 7 and 8 has the best accuracy with PCA as a dimensionality reduction method while Experiment 2, 3, 5 and 6 have the best accuracy with GPLVM.

Wilcoxon signed rank sum test:

Table 9.2 shows the results of Wilcoxon significance test involving PCA vs. GPLVM. There is no significant difference between both of these methods.

Test	Significance p
PCA-GMP-1 vs GPLVM-GMP-1	0.238
PCA-GMP-2 vs GPLVM-GMP-2	0.6171
PCA-CCRF-1 vs GPLVM-CCRF-1	0.9362
PCA-CCRF-2 vs GPLVM-CCRF-2	0.7642

Table 9.2 Wilcoxon signed rank test for PCA vs GPLVM in continuous approach

Running time:

Table 9.3 gives the running time for experiment 5 fold 2. The experiment was run on a Core 2 Duo 1.6 GHz processor with 2GB of memory.

Exp. detail	Experimental time for Experiment 5 (Fold2) - Continuous case									
	Dimensionality Reduction(sec)		Discretization (X-means) (sec)		Training and Testing (sec)		Total Time (Sec.)		Total Time (Hrs.)	
	PCA	GPLVM	PCA	GPLV M	PCA	GPLVM	PCA	GPLVM	PCA	GPLVM
GMP-1	173	156239	N/a	N/a	730	454	903	156693	0.25	43.52
GMP-2	173	156239	N/a	N/a	923	560	1096	156799	0.30	43.55
CRF-1	173	156239	N/a	N/a	8759	7863	8932	164102	2.48	45.58
CRF-2	173	156239	N/a	N/a	10823	8144	10996	164383	3.05	45.66

Table 9.3 Running time for Experiment no. 5 for the 2nd fold.

As before, PCA takes around half an hour to complete the experiment while GPLVM takes approximately two days to complete the experiment. The majority of time was spent in applying GPLVM to the data. Both PCA and GPLVM were able to achieve accuracy over 80% hence it is very difficult to select the one best method out of PCA and GPLVM. One may choose PCA over GPLVM due to its speed and relatively high accuracy.

9.2 Generative (DMP) vs. Discriminative (CRF).

From table 9.1 (c), we can conclude that discriminative models outperforms the generative models for continuous state process. CRFs were able to achieve the best accuracy of more than 91% except in experiment 6 with 81% accuracy. Also, the low accuracy of Gaussian Markov process models indicates the inability of generative continuous models to discriminate between the motions in our experiments whereas much better accuracy achieved with CRFs indicates the benefits of discriminative

models. The improvement in accuracy was found to be statistically significant in all cases as shown in Table 9.4. was found to be 0.0005% significant in all the cases.

Test	Significance p
PCA-CRF-1 vs PCA-GMP-1	0.0005
PCA-CRF-2 vs PCA-GMP-2	0.0005
GPLVM-CRF-1 vs GPLVM-GMP-1	0.0005
GPLVM-CRF-2 vs GPLVM-GMP-2	0.0005

Table 9.4 Results for Wilcoxon signed rank sum test for CRF vs Gaussian Markov process.

9.3 Other Observations

From our results, we conclude that second-order discriminative models perform the best on our experiments. CRFs were successfully able to classify similar motions like walk (Experiment 5, 6, 7). Also, when subjects were randomly divided across the training data, CRFs performed fairly well to give accuracies over 91%. In the continuous case, it is very difficult to pick one best dimensionality reduction method. One may choose PCA over GPLVM due to its simplicity and faster running time.

10. Conclusion

The main contributions of the thesis are twofold. First, we provide an empirical comparison of various sequential supervised learning algorithms for classifying entire sequences composed of 96-dimensional frames. Second, we develop an Auto-Regressive Conditional Random Field which was shown to be very effective at classifying high dimensional sequences.

Our first contribution was an empirical comparison of different graphical models for the problem of classification of mocap data. We compared various algorithms along the axes of 1) discrete versus continuous models, 2) generative versus discriminative models and 3) PCA (a linear dimensionality reduction technique) versus GPLVM (a non-linear dimensionality reduction technique). We also investigated models which varied in terms of the order of the Markov assumption. Different variants of these algorithms were evaluated on a total of 8 separate experiments. In each experiment, we created a data set consisting of a variety of motion sequences. The classification accuracy for 2-fold cross validation was used as a measure to evaluate the models.

From the results presented in chapter 9 and 10 we can conclude that discriminative approaches such as Conditional Random Fields (CRF) outperform generative approaches such as Gaussian Markov Processes. When we compared discrete models against continuous models, we found that in the discrete case, classification accuracy was highly dependent on the number of clusters found during discretization, with PCA surprisingly producing better results than GPLVM. For continuous models, there was no clear winner in terms of dimensionality reduction techniques, although the extremely long running time of GPLVM can be considered a detriment. However, discriminative models clearly outperformed generative models.

In our second contribution, we developed an Auto Regressive (AR-CRF) model for continuous state spaces. Earlier research on CRFs focused on discrete state spaces and most approaches using CRFs for motion capture operated on much fewer than 96 dimensions. Furthermore, we investigated the problem of classifying entire motion sequences rather than each frame. Our AR-CRF addressed these issues and outperformed its generative counterpart – a Gaussian Markov Process. We were pleasantly surprised by its ability to distinguish between motions that, to the human eye, appear to be slight variants of each other.

11. Future work

There are a number of different future directions that we would like to pursue. In our work with CRFs, we did not take advantage of adding arbitrary features defined over the input motion sequence, which is one of the powerful benefits of using CRFs. We believe that adding features such as periodicity, velocity and acceleration can improve classification accuracy. In addition, another assumption was the full connectivity of the graphical model structure in the continuous case. We can use techniques from graphical model structure learning to find the dependencies between the dimensions of two consecutive time slices and thus reduce the complexity of the model. In our experiments with GPLVM, we found that time and memory were considerable bottlenecks to using this algorithm. We would like to investigate the use of KD-trees to speed up the performance of GPLVM.

There is also a need to develop algorithms for motion segmentation. The models used for classification in this thesis could be extended to motion segmentation by using a sliding window over the motion sequence. Data prior to the sliding window is used to train a model while data falling within the sliding window is treated as test data to be classified. If the log-likelihood of the data in the sliding window exceeds a threshold, the data is considered to be different from the current model and hence a segmentation point can be declared. Currently, the expensive running time of our models prevents online segmentation and future work would consist of developing efficient online segmentation algorithms.

Finally, we would like to apply our algorithms for classifying high-dimensional, continuous, and time-variant data to domains other than motion capture. In the future, we would like to apply our models to other datasets with these properties, such as functional magnetic resonance imaging (fMRI) data and bird acoustics.

Bibliography:

Academy Awards (2007). Retrieved from Wikipedia:

<http://www.imdb.com/title/tt0366548/awards>

Agarwal, J. K., and Cai, Q. (1999). Human motion analysis : A review. *Computer Vision and Image understanding* , 73, 90-102.

Altun, Y., Hoffman, T., and Smola, A. J. (2004). Gaussian Process Classification for segmenting and annotating sequences. *Proc. Of 21st International conference on machine learning*. Banff, Canada

Balakumar, S. (2002). Multivariate Changepoint Problem. *Missouri Math. Sci.* , 14 (3), 186-195.

Becker S., Thrun S. and Obermayer K.(2003). *Advances in Neural Information Processing Systems*, vol. 15, Cambridge, MA, MIT Press.

Barbic, J., Safanova, A., Pan, J.-Y., Hodgins, J. K., and Pollard, N. S. (2004). Segmenting motion capture data into distinct behaviors. In *GI '04: Proceedings of Graphics Interface 2004* (pp. 185-194). London: School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.

Bengio, Y., Paiement, J.-f., and Vincent, P. (2004). Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. *Annual Advances in Neural Information Processing Systems* . , 177-184.

Bentley, J. L. (September, 1975). Multidimensional binary search trees used for associative Searching. *Commun. ACM*. 18(9), 509–517.

Beyer, K., Goldstein, J., Ramkrishnan, R., and Shaft, U. (1999). when is "Nearest Neighbor" Meaningful ? *International Conf. on Database Theory* , 217-235.

Bishop, C. M. (1999). Bayesian PCA. In *Proc. of conf. on Advanced neural information processing Systems II* , 382-388.

Bobick, A. F., and Davis, J. W. (1996). Real-time recognition of activity using temporal templates. In *Workshop on Applications of Computer Vision* , 39-42.

Bobick, A. F., and Wilson, A. D. (1995). A state-based technique for the summarization and recognition of gesture. *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision* , 382.

- Bregler, M., Oliver, N., and Pentland, A. (1997). Coupled hidden Markov models for complex action recognition. *Proc. of IEEE CS Conf. on Computer Vision and Pattern Recognition* , 994-999.
- Bregler, C. (1997). Learning and recognizing human dynamics in video sequences. *Proc. of IEEE CS Conf. on Computer Vision and Pattern Recognition* , 568-574.
- Chomat, O., and Crowley, J. L. (1998). Recognizing motion using local appearance. 271-279.
- Chomat, O., and Crowley, J. L. (1998). Recognizing motion using local appearance. *International Symposium on Intelligent Robotic Systems* .271-279
- CMU Mocap Library. (). Retrieved from CMU motion capture library: <http://mocap.cs.cmu.edu/>.
- Comon, P. (1994). Independent Component Analysis : A new concept. *Signal Processing* , 36 (3), 287-314.
- Cox, T., Cox, M., and Raton, B. (2003). Multidimensional Scaling. *Technometrics* , 45 (1), 182-182.
- Drake, A. W.(1962). Discrete state Markov processes. In *Fundamentals of Applied probability theory*. New York NY Mcgraw hill.
- Fang, L., Runger, G. C., and Eugene, Z. (2006). Supervised learning for change-point detection. *International Journal of Production Research* , 44 (16), 2853-2868.
- Fisher R., (1936). Fisher Linear discriminant analysis.
- Fukunga, K.(1990). *Statistical Pattern Recognition* (2 ed.).
- Galata, A., Johnson, N., and Hogg, D. (2001). Learning variable-length Markov models of behavior. In *Proceedings of the IEEE International Workshop on Modelling People*, 81, pp. 398-413.
- Goddard, N. H. (1994). Incremental model-based discrimination of articulated movement from motion features. In *Proc. of IEEE Computer Society Workshop on Motion of Non-Rigid and Articulated Objects* , 89-95.
- Gombay, E., and Horvath, L. (1997). An application of the likelihood method to change point detection. *Environmetrics* , 8, 459-467.

- Guo, Y., Xu, G., and Tsuji, S. (1994). Understanding human motion patterns. *Proc. of Intl. Conf. on Pattern Recognition* , 325-329.
- Hotelling, H. (1933). Analysis of complex of statistical variables into principal components. *Journal of educational psychology* , 24, 417-441.
- Jain, A. K., Duin, R. P., and Mao, J.(2000,January). *Statistical Pattern Recognition : A Review.IEEE transactions on pattern analysis and machineintelligence.22(1),4-37.*
- Kalman, R.E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic engineering*, 82(1), 35-45.
- Kojima, A., Tamura, T., and Fukunaga, K. (2000). Generating natural language description of human behaviors from video images. *Proc. of Intl. Conf. on Pattern Recognition*, (pp. 728-731).
- Krzanowski, W. J.(1979) Between-groups comparison of principal components. *Journal of American Statistics Association* , 74, 703-707.
- Lawrence, N. (2008). *Gaussian Process Latent variable modeling*. Retrieved from <http://www.cs.man.ac.uk/~neill/fgplvm/>
- Lawrence, N. (2008). *Gaussian Processes*. Retrieved from <http://www.gaussianprocess.org/>
- Lawrence N. D., Seeger M., and Herbrich R.(2003). Fast sparse Gaussian process methods: The informative vector machine. In Becker et al. , pages 625–632.
- Lawrence, N. (2005). Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. 6.
- Lawrence, N. (2005). Probabilistic Principal Component Analysis with Gaussian Process Latent Variable models. *Machine Learning Res.* , 6, 1783-1816.
- Li C., Kulkarni, P. R., and Prabhakaran, B. (2007). Segmentation and Recognition of Motion capture Data stream by classification. *Multimedia Tools Appl.* , 35 (1), 55-70.
- Li, Y., Wang, T., and Shum, H.-Y. (2002). Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on graphics* , 465-472.
- Lin, C.-T., Nein, H.-W., and Lin, W.-C. (1999). A space-time delay neural network for motion recognition and its application to lipreading. *International Journal of Neural Systems* , 9 (4), 311-334.

- Matten, V. d., Postma, E. O., and Van der Herik, H. J. (2007). Dimensionality reduction : A Comparative Review.
- Meredith, M., and Maddock, S. (2001.). motion Capture files Format Explained.
- Minka, T. P. (1999). *From Hidden Markov Models to Linear Dynamical Systems*.
- Murphy, K.(1998). *An Introduction to graphical models*.
- Murphy, K. P. (1998). Learning switching kalman-filter models.
- Nakata, T(2007). Temporal Segmentation and Recognition of body motion data based on Inter-limb correlation analysis. *Proc.of Intelligent robots and Systems*.
- Niyogi, S. A., and Adelson, E. H. (1994). Analyzing and recognizing walking figures in XYt. 469-474.
- Nocedal J. (1980). Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of computation*,35, 773-782.
- Pavlovic, V., Rehg, J. M., and Maccormick, J. (2000). Impact of dynamic model learning on classification of human motion. *In IEEE International Conference on Computer Vision and Pattern Recognition*.
- Pavlovic, V., Rehg, J. M., Cham, T. J., and Murphy, K. P. (1999). A dynamic Bayesian network approach to figure tracking using learned dynamic models. *IEEE International conference on computer vision*.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. San Mateo: Morgan Kaufmann.
- Pelleg, D., and Moore, A. (2000). X means: Extending K-means with efficient estimation of the number of Clusters. *Proceedings of Seventeenth International Conference on Machine learning* , 727-734.
- Polana, R., and Nelson, A. (1994). Low level recognition of human motion (or how to get your man without finding his body parts). *In Proc. of IEEE Computer Society Workshop on Motion of Non-Rigid and Articulated Objects* , 77-82.
- Rabiner, L. R. (1990). A tutorial on hidden Markov models and selected applications in speech recognition. In *Readings in speech recognition* (pp. 267-296). San Francisco, CA, USA: Morgan Kaufmann Publisher Inc. .

- Rasmussen, C. E., and Williams, C. K. (2005). *A Complete Reference to Gaussian Process*.
- Rosenblum, R., Yacoob, Y., and Davis, L. (1994). Human emotion recognition from motion using a radial basis function network architecture. 43-49.
- Roweis, S. T., and Saul, L. K. (2000). Nonlinear Dimensionality reduction by Locally Linear Embedding. *Science*, 290, 2323-2326.
- Scaled Conjugate Gradient*. (1995). Retrieved from <http://www.nada.kth.se/~orre/snns-manual/UserManual/node241.html>
- Scholkopf, B., Smola, A., and Muller K. (2001, September 25). Nonlinear Component Analysis as a kernel Eigenvalue Problem. *Journal of Neural computations*, 10(5), 1299-1319.
- Shahabi, C., and Yan, D. (2003). Real-time pattern isolation and recognition over immersive sensor data streams. *In Proceedings of the 9th International Conference on Multi-Media Modeling*.
- Sminchisescu, C., Kanaujia, A., Zhiguo, L., and Metaxas, D. (2005). Conditional models for contextual Human motion recognition. *Int'l Conf. on Computer Vision*, (pp. 1808-1815).
- Smith, L. I. (2002, February 26). A tutorial on Principal Component Analysis.
- Sutton, C. *An Introduction to conditional random fields*.
- Tenenbaum, J. B., De Silva, V., and Langford, J. C. A. (November, 2000). global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2319-2323.
- tipping, M. E., and Bishop, C. M. (1997). *Probabilistic principal component analysis*. Neural Computing Research group.
- Urtasun, R., and Darrell, T. (2007). Discriminative Gaussian Process Latent Variable Model.
- Vail, D., Veloso, M., Lafferty, J. (2007). Conditional Random Fields for Activity Recognition. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 1-8.
- Wallach, H. M. (February, 2004). *Conditional Random fields: An introduction*.

Wang, J. M., Fleet, D. J., and Hertzmann, A. (2008). Gaussian Process dynamical Models for Human motion. *IEEE Transactions on Pattern Analysis and machine intelligence* , 30 (2).

Wang, L., Weiming, H., and Tieniu, T (May, 2002). *Recent developments in Human motion analysis*. National laboratory of pattern recognition.

Witkin, A., and Zoran, P. C. (1995). *Motion Warping*.

Yamoto, J., Ohya, J., and Ishii, K. (June,1992). Recognizing human action in time-sequential images using hidden markov model. *CVPR* , 379-385.

Yan, L., Wang, T., and shum, H.-Y.(2002). Motion Texture: A two level statistical model for character motion synthesis. Microsoft research.

Appendix A

In this section, we show how we derive features for a one-dimensional first-order autoregressive Condition Random Field (AR-CRF). This derivation can be easily extended to AR-CRFs of higher dimensions. Figure A.1 shows the first-order AR-CRF models and its corresponding maximal clique is shown in Figure A.2

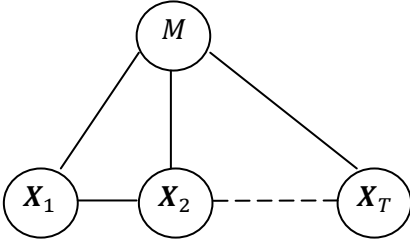


Figure A.1: First order CRF model. m is the motion label and X s are the observation in reduced space. Here $d = 1$

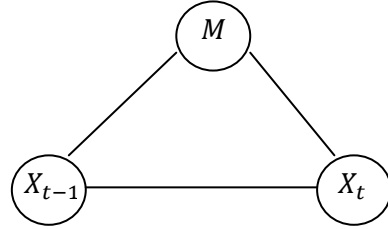


Figure A.2: Maximal clique corresponding to figure A.1.

The dependency between X_{t-1} and X_t can be seen as a Gaussian distribution.

$$X_t \sim \mathcal{N}(w \cdot X_{t-1}, \sigma)$$

The probability for directed model corresponding to fig A.2 is given as,

$$\prod_{t=2}^T p(X_t | X_{t-1}) \cdot p(M) \quad (A.1)$$

Since we consider uniform prior over $P(M)$ we can drop the second term. Equation (A.1) can be rewritten as,

$$\prod_{t=2}^T \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left\{-\frac{1}{2} \left(\frac{X_t - w \cdot X_{t-1}}{\sigma}\right)^2\right\} \quad (A.2)$$

The equation for undirected model (i.e. a CRF) in figure A.2 is given as,

$$\prod_{t=2}^T \exp\left\{\sum_{k=1}^K W_k \cdot f_k(m, x_t, x_{t-1})\right\} \quad (A.3)$$

Following Vail et al. (Vail et al. 2007), our goal is to convert, Equation A.2 into a form suitable for undirected models (ie. Equation A.3)

First we take the log:

$$\sum_{t=1}^T \left(\log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) + \left\{ \frac{-1}{2} \left(\frac{X_t - w \cdot X_{t-1}}{\sigma} \right)^2 \right\} \right)$$

Let us drop the summation and find the features for a time slice ‘ t ’

$$-\frac{1}{\sigma^2} \cdot (X_t)^2 + \frac{w}{\sigma^2} \cdot (X_t)(X_{t-1}) - \frac{w^2}{\sigma^2} \cdot (X_{t-1})^2 - \log(\sqrt{2\pi\sigma^2})$$

This can be written as,

$$W_1 \cdot (X_t)^2 + W_2 \cdot (X_t)(X_{t-1}) + W_3 \cdot (X_{t-1})^2$$

We can define,

$$\left. \begin{aligned} f_1(m, x_t, x_{t-1}) &= (X_t)^2 \\ f_2(m, x_t, x_{t-1}) &= (X_t)(X_{t-1}) \\ f_3(m, x_t, x_{t-1}) &= (X_{t-1})^2 \end{aligned} \right\} \quad (A.4)$$

Note that the features defined in equation (A.4) are sufficient statistics for the 1-dimensional Gaussian distribution.

We now briefly show the feature derivation for a two-dimensional first-order AR-CRF. Figure A.3 shows the first order AR-CRF model. X_t is a d-dimensional vector where $d=2$. Figure A.4 shows the model equivalent to the model in figure A.4.

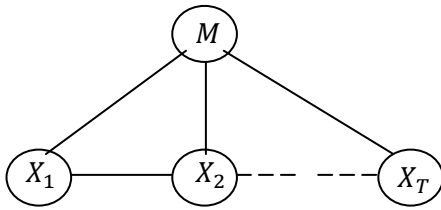


Figure A.3: First order AR-CRF model. M is the motion label and X_s are the observations in reduced space. Here $d = 2$ and X_t is the 2-dimensional vector.

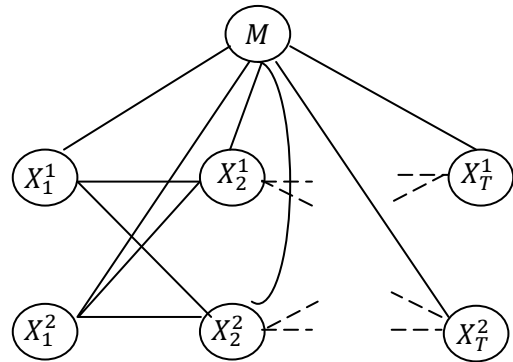


Figure A.4: Shows the model equivalent to the model shown in figure A.3. Note that the value of X_t^j depends on the value of all dimensions in the previous time-slice.

We assume that the value of X_t^j depends on the value of all dimension in the previous time slice. After applying graph moralization and graph triangulation to the figure A.4 we get the undirected graph as shown in figure A.5.

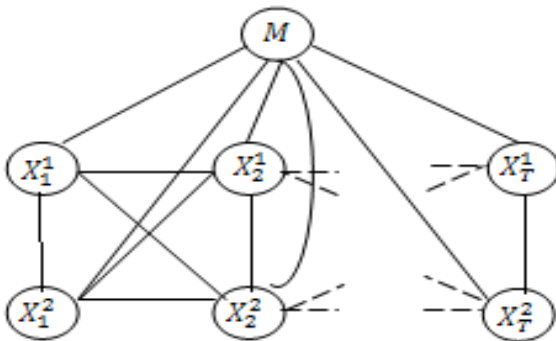


Figure A.5: Undirected graph obtained after applying graph moralization and triangulation to the model in figure A.4

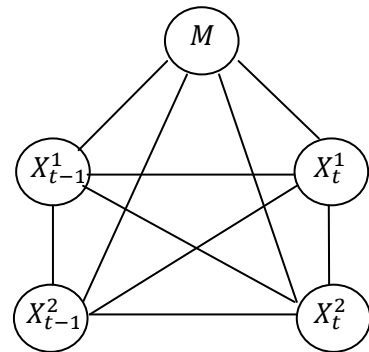


Figure A.6: Maximal clique for time slice t corresponding to AR(1)-CRF.

The maximal clique for the time slice t can be seen in figure A.6. In auto-regressive CRF model, the distribution for $P(X_t^j | X_{t-1}^j)$ is a Gaussian distribution:

$$X_t^j \sim \mathcal{N}(w_1 \cdot X_{t-1}^1 + w_2 \cdot X_{t-1}^2 + w_3 \cdot X_t^k, \sigma); \quad j \neq k$$

Following a similar procedure involving rearranging the terms in the Gaussian, we can obtain features for the two-dimensional AR-CRF that involve nodes within the t th time slice, nodes within the $(t-1)$ th time slice, and interactions between the nodes in the t th and $(t-1)$ th time slices. These features are described below:

The features for the ' t 'th time slice are:

$$(X_{t-1}^1). (X_{t-1}^1); (X_{t-1}^1). (X_{t-1}^2); (X_{t-1}^2). (X_{t-1}^2) ;$$

The features within the $(t-1)$ th time slice are:

$$(X_t^1). (X_t^1); (X_t^1). (X_t^2); (X_t^2). (X_t^2) ;$$

The features involving interactions between t th and $(t-1)$ th time slice are:

$$(X_{t-1}^1). (X_t^1); (X_{t-1}^1). (X_t^2); (X_{t-1}^2). (X_t^1); (X_{t-1}^2). (X_t^2)$$

We can generalize the procedure above to d -dimensional first-order AR-CRFs where the features can be written as:

Features for the $(t-1)$ th time-slice:

$$(X_{t-1}^j). (X_{t-1}^k); \text{ For } 1 \leq j \leq k \leq d .$$

Features for t th time-slice:

$$(X_t^j). (X_t^k); \text{ For } 1 \leq j \leq k \leq d .$$

Features capturing the cross-interaction terms between the t th and $(t-1)$ th time-slice:

$$(X_{t-1}^j). (X_t^k); \text{ For } 1 \leq j \leq d; 1 \leq k \leq d .$$