Gesture recognition in motion captured data using Hidden Markov Models

Sharon Lourduraj

993833107

M.Eng Report

University of Toronto

Mechanical and Industrial Engineering

Jan 4, 2011

Abstract

Hidden Markov Models (HMMs) lend themselves well in a variety of applications that are dependent on spatio-temporal variability, gesture recognition is one such application. This report details the experimental process of constructing, and optimizing a HMM framework for gesture recognition. A gesture recognition rate of 63% for an acted data set is achieved and a recognition rate of 60% for an unacted data set is achieved. In the experiments, motion capture data is used as a primary means for posture tracking, than the traditional image sequences.

Contents

1. Introduction

Technology as we know, from our hand-watches to parts for space stations, are built by automated machines on a daily basis. Today, it is an important part of our lives. And in the future it will be even more important as we start interacting with intelligent agents to accomplish our tasks. The key difference between today's robots and tomorrows robot is not the computing power or sensing capabilities it will be their seamless interaction with their users. A robot that manufactures parts for automobiles certainly does not need to be socially friendly with its human operaters. These robots do what they are told, without question; but that is their purpose, to build parts continuously. But we want to look into robots that might one day service users, help you make coffee, assist someone in a grocery store, help children cross traffic lights, be a friendly companion, act as a guide dog, or help elders in need around the house. These robots are not performing tedious tasks on a daily basis, but are to be interactive with humans, and socialize with them. For this to happen, robots need to sense and interpret human interaction, their gestures and body language.

A human expressing a gesture through body language, goes through a sequence of observable postures. These observations can be discretized to extract a smaller subset of postures through which a gesture can be conveyed. Think of it as watching a movie. If we drop the frame-rate significantly we can still visually interpret the movie. The order in which each observation occurs is of importance, shifting the observation's order in the sequence would convey a completely different gesture. Expression of a gesture also varies from person to person, it also varies depending on the context that someone is experiencing.

We introduce the problem of recognizing gestures from data obtained by a motion capture system. Many methods exists in recognizing gestures[1], template-matching, dictionary lookup, statistical matching, neural networks, and ad hoc methods. Some techniques are widely applicable to various problems in gesture recognition while others are designed specifically to resolve a particular problem. Gesture recognition is considered to have stochastic properties[1], as such a technique that uses the stochastic nature of the problem to its advantage would be of value.

A Hidden Markov Model (HMM) is defined as a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols[2]. Being robust to adapt to spatio-temporal variability[3], HMMs lend themselves well to the task of gesture recognition as gestures performed by humans

are stochastic in nature.

This report is an account of the experimental steps taken in using HMMs to recognize gestures in 3D motion captured data. For a full mathematical account of HMM, Rabiner and Juang [2] is a great source of information. The initial experiments focused on discovering the process involved in using HMMs to train and classify motion capture data. The later experiments focus on improving the performance of HMM to provide better results. Finally, we test the robustness of the HMM framework by using a different dataset than the one which was used to develop the framework. We also perform a custom experiment using Xbox Kinect to capture body postures performed by an actor, and classify them.

Outline

We introduce the overall gesture recognition process using HMMs in Chapter 2 with a summary of some of the works consulted. Chapter 3 outlines the main components of the process whose considerations are invaluable in making decisions to improve gesture recognition performance. Chapter 4 discusses the results of the experiments conducted, and analyzes the various pitfalls encountered in building a system to achieve a 63% recongition rate from data sets in literature, and a 75-80% recognition rate on a custom experiment performed using a Xbox Kinect.

2. Gesture Recognition with HMM

We discuss the details of gesture recognition process using HMM in this chapter. This will put the reader in prespective of the various parts that work together in using HMMs for gesture recognition. HMM as a machine learning algorithm is pretty straight forward in application. Due to the generality of the algorithm it can be applied to any problem, in any domain, that can be translated into entities that a HMM process can take advantage of.

2.1 Works consulted

Fong et al. [4] conduct a survey of socially-interactive robots providing us with sufficient history and knowledge into the applications such robots. The paper also discusses the various design considerations such as robot embodiment, emotion sensing, emotion as a control mechanism, speech, facial expressions, body language, personality traits, and the various forms of human-oriented preception. A key point is noted in the paper in regards to having meaningful interactions with humans. Social robots must preceive the world as humans do, that is sensing and interpreting the same phenomena that humans observe [4]. It is identified that perceptual abilities similar to humans are necessary in addition to the sensing capabilities required for conventional functions. People tracking, speech recognition, gesture recognition and facial preception are some of the human-oriented perceptual abilities listed.

The purpose of the experiment was to implement and evaluate the use of HMMs to train and classify gestures in motion captured data. HMMs can be applied to any problem where spatio-temporal variability is of importance. As such, the implementation of HMM is straight forward besides the overhead in implementing the algorithms necessary to train the HMM, and calculate outcomes. This means that HMMs can virtually be applied to any domain; applications include speech recognition[5], hand writing recognition[6], data mining[7], image classification[8], and of course gesture recognition[9].

In almost all of the works the implementation of the HMM algorithm remains the same. The difference between the various works is that they innovate in the extraction of feature vectors from an acted gesture. Huang and Jeng [10] propose a feature extraction by a hybrid technique combining the spatial and the temporal information of each frame to extract the feature images. Lee and Wong [3] introduce a technique

of template matching to generate observation symbols using a Johansson display that encodes the motion of a moving person. Lee and Kim [11] introduce a HMM-based threshold model that rejects input patterns which are recognized as non-gesture motions. Chen et al. [12] introduce a method to recognize continuous gestures with a stationary background. Marcel et al. [13] extract features by tracking the skin-color blobs corresponding to the hand into a body-face space centered on the face of the user[13]. Rigoll et al. [14] propose feature vectors based on global motion features, extracted from each difference image of the image sequence. Pellegrini and Iocchi [15] use stereo vision sensors to track posture, and then match against a 3D model of a human to extract feature vectors, such as hip angle, and knee angle.

Feature vectors improve the recognition accuracy of the HMM in a specific problem domain, works exist that show how HMM can be used in applications that use gesture recognition. Ott et al. [16] demonstrate the application of imitating a human's motion by a humanoid through a motion capture system. They use a variation based on HMMs to learn, recognize and generate motion. Starner and Pentland [17] demonstrate how HMMs can be used to recognize American Sign Language. Similarly, codebook creation plays an important role as we discovered while performing the experiments. Lee and Wong [3] and Park et al. [18] demonstrate the use of template database as a codebook. While, Yang and Xu [1] demonstrates a traditional approach of using k-means algorithm to cluster and quantize data into a codebook.

The initial experiments were performed on the data set from Kleinsmith et al. [19]. And a final experiment was conducted on a different data set from Kleinsmith et al. [20]. Though they employed neural networks as the machine learning framework to train and recognize gestures, their work was significant in understanding proper selection of feature vectors. They also detail the use of 3D motion capture data directly for affective posture recognition. Their works demonstrate how an emotion is conveyed by different cultures, and the various emotions that people portray when they are performing a task naturally.

We also looked into some literature regarding gesture classification using means other than HMMs. Caridakis et al. [21] take advantage of the clustering properties of Self-Organizing Maps for gesture recognition; as self-organizing feature maps lead themselves well to model spatio-temportal information extracted from images. Oz and Leu [22] use artificial netural networks for gesture recognition in American Sign Language, where data is captured via a sensory glove. The bibliography section of the report includes many more works that were consulted to gather knowledge about gesture recognition and HMMs in general.

2.2 Terms

For clarity we establish some terms and their relative meaning, strictly as we have used them in this report and in our experiments.

Feature Data A specific property of a feature on a person's body, such as the position of the wrists, angle

between the elbows, angle at the knees, or postion of the left-side of the head.

Posture A posture is made up of a group of feature data. Together, they establish a person's pose.

Gesture A gesture is made up of many postures. When one or more postures are executed in a sequence they convey a message from the person to an observer.

Emotion An emotion categorizes a group of gestures together. All gestures that indicate that a person is angry would fall under the angry emotion.

We call the data captured from the motion capture system a `gesture`, which is a specific instance of an `emotion`. One or more gestures might be similar to each other and they would both fall under the same emotion. For example, being `angry` is an emotion, but one can show anger through various gestures. The data from the motion capture system is essentially a sequence of frames with motion capture data (angles, or position of body points in space). A `posture` is a specific frame in a gesture. A gesture is made up of a sequence of postures, as shown in figure 2.1.

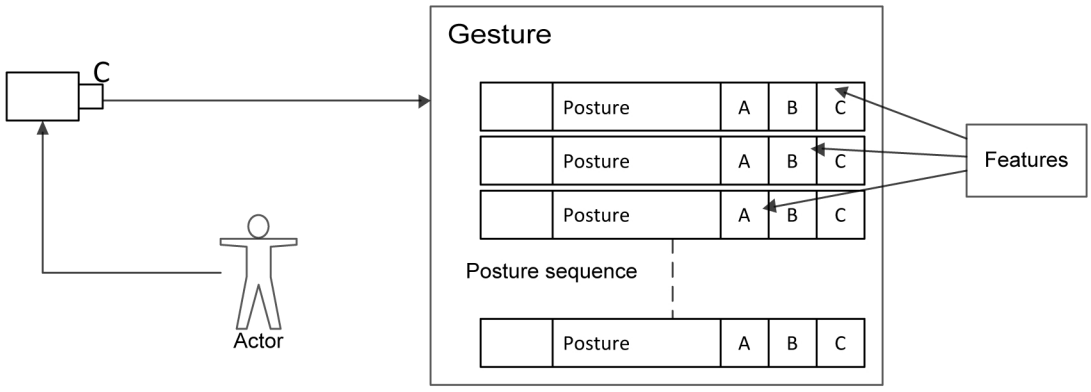


Figure 2.1: Gestures, postures and features from motion capture data.

2.3 Data set

As in any machine learning algorithm, a data set plays an important role in enabling the algorithm to train, classify and re-train. There are two basic ways throug which data can be captured for gesture recognition, data captured through a video camera, and data captured through a 3D marker tracking system. The camera captures a sequence of images that is used to extract features later in the process. The sophistication, or the details, of the data captured is reliant on the camera itself. A hi-definition camera would capture more details versus a low definition camera, likewise a camera that is able to collect color data can provide additional details of the scene.

A 3D marker tracking system is a sophisticated derivative of using a video camera to capture data. Rather than capture details of the scene, the system captures movements of an actor's skeletal features. This is done by having many video cameras positioned at significant locations. These are then used to track markers on an actor wearing a specialized suit. Markers are positioned appropriately at significant locations such as wrists, knees, head, waist, spine, hips and many more. An algroithm then interprets the data from the various cameras and transforms them into position coordinates as it corrosponds to the human sekeletal frame. In essense, a single frame of this transformed data is a posture as defined earlier. This data can be further processed into rotation angles, and euler angles corresponding to a datum, as necessary.

Morie et al. [23] provide a summary of their experiments in a creating markerless visual tracking system inspired by the brain; for face and object recognition. Another example of a markerless 3D posture tracking system is the use of Microsoft Xbox Kinect [24]. Regardless of how the data is captured, for gesture recognition a data set contains a collection of gesture movies. Dissecting further, a movie contains a sequence of frames, in our case a gesture movie contains a sequence of posture frames. And each posture frame contains data related to a feature that was tracked. The differences between the various tracking mechanism narrows down to the posture frame. A simple video capture system records raw data of the scene in each frame, while a 3D marker motion capture system captures joint locations or angles in each frame. A raw data would need to be processed further to extract the required features; in fact, any data can be processed further to refine features necessary to accomplish gesture tracking and recognition.

2.4 Feature Vector

The next step in the process is to extract feature vectors. Feature vectors isolate the necessary data from the redundant and non-essential data; data related to the scene from data related to the actor's limb positions. This is one of the more involved steps in the process. Any machine learning algorithm relies on feature vectors for training and classification, so does HMM. The feature extraction process can vary from being very simple to being extremely complex. Feature vectors are extracted from a posture or across multiple postures. The feature, location of the actor's head can be extracted from the posture, while the velocity of the actor's head needs to be extracted across two postures over time. A posture can have multiple features, combined they are called feature vectors. So for a gesture containing 500 frames (postures), 500 feature vectors can be derived. The dimension of a feature vector is dependent on the choice of the features. Location of the actor's wrist has three dimensions in space, while the actor's elbow angle has only one dimension (the angle itself). This process is repeated for all the gestures captured.

Features that are to be extracted depends on the data representation in the data set. A 3D marker based motion capture system provides the information of spatial coordinates in all three planes. Spatial coordinates are not the only features that can be extracted, many other features can be extracted that define complex

relationships between a sequence of postures. Applying a difference operator to two posture frames would give us velocity feature vectors. Having spatial coordinates of various features allows us to calculate other information easily, such as head tilt angle, direction of head, angle at the elbow, depth of each wrist. While, a 2D video camera does not provide that information directly. To extract such information from a 2D image one needs to apply image processing techniques. Typically, this would involve normalizing the image, then extracting the actor from the image, and then finally perform a template analysis to extract data from the image. One can even calculate the average movement of pixels in a 2D image to estimate motion of a specific feature in that gesture.

Feature vectors can be anything from the posture frames that is calculateable, quantizable and provides relevant information about the data that is going to be trained. Specific examples include, position coodinates, velocity, acceleration, angular velocity and angular acceleration of any number of features that are necessary to accomplish the training of a gesture and provide accurate classification. Suppose we have a gesture where the actor is hopping in the air, we should extract feature data corresponding to the position and velocity of knees, ankles and angle at the knees for a system to train and recognize a related gesture accurately. Ignoring a feature could result in missclassification.

The feature extraction process does impose a significant overhead in the training and recognition process, suppose if the camera were to provide us with 10 images per second, then we would have to isolate and calculate 10 frames in less than one second to keep up with incoming data. Depending on the resolution of the data, the number of features,and the complexity of features to be extracted, this overhead could either increase or decrease.

2.5 Codebook

Once the feature vectors are extracted from the data sets, a codebook needs to be created. This is an abstraction layer that separates the problem domain from the mathematics invovled in HMM. Templates need to be extracted from these feature vectors. Templates are just a mechanism to group similar postures together. If someone is raising their hand, we can take three snapshots while the person is raising their hand. When their hand is at the initial position, when the hand moves half way up, and then at the final position. Any intermediate snapshot is not necessary since, the three snapshots convey that the person is raising their hand. Thus, given a set of feature vectors we identify few templates from it. Note that templates from one gesture can be used in another gesture, hence the template dataset is essentially a database of unique postures. The existing postures in the template database can also be used to create other gestures, or variation of the particular gesture. This introduces the idea of a `codebook`, also a synonym for a template database. We label each posture in the codebook with a unique number from $1 \ldots n$, and one can refer to them with a page number. In HMM terms, the page number, or a reffering id, is known as an observation

`symbol`.

Consider a gesture and a codebook, we need to transform each posture in the gesture into a symbol in the codebook. Thus rather than having a $n \times m$ matrix of postures and features, we have a $1 \times m$ vector where each column corresponds to a symbol in the codebook. Note that there is a loss of information in this transformation. The original gesture file contains accurate information about the gesture, while the codebook transformed version contains an approximate, hopefully sufficient, version of the gesture due to quantization. The concept of codebooks is necessary to work with HMMs, as HMMs are built around the concept of states and observations. We discuss briefly about states and observations in the following section.

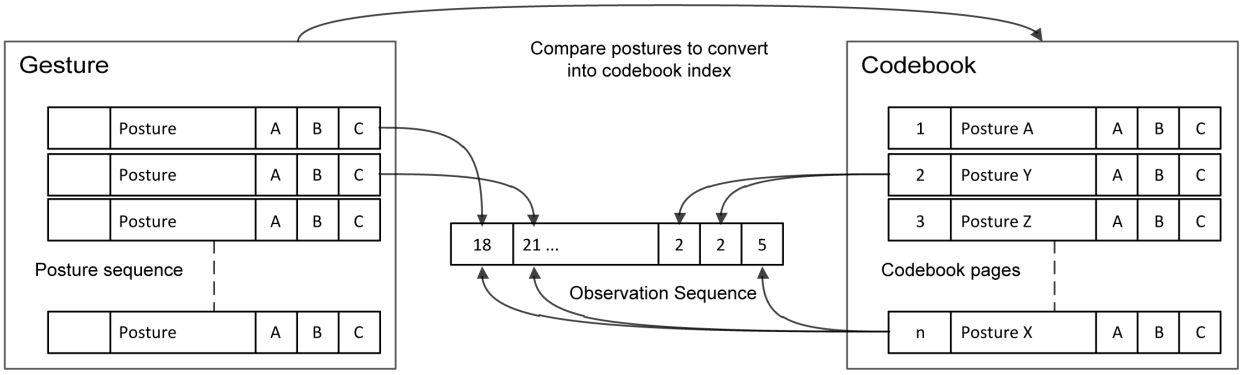


Figure 2.2: Observation sequence of a gesture determined from a codebook.

2.6 Observations sequences and states

With the codebook created, we are now able to transform a sequence of input frames with rich posture data into a sequence of symbols, that approximates the input sequence, from the codebook. This input sequence is known as the observation sequence. Indicating that the system observes the occurances of symbols in the codebook as the gesture is taking place. Figure {fig:fw2 shows how a gesture input (a sequence of posture frames) is converted into an observation sequence. Every input posture frame in the gesture is converted into a feature vector. This conversion allows us to compare features from the posture to all the stored feature from the templates in the codebook. The symbol for the closest matching template is obtained. Thus, we obtain the observation sequence by obtaining the corresponding symbols from the codebook and saving it in the same order. The numerical order of the sequence itself is irrelevant as they are only a reference to the location of the template in the codebook.

The order of the symbols in the sequence of observations matter. Changing the order of the symbols in the sequence would imply a very different gesture or emotion. As such, each observation sequence is said to occur at specific arbitrarly defined states in a gesture. A gesture can have many states depending on the

complexity of the gesture. Similar to Yang and Xu [1] we assume the gestures to have three observable states, a ready state, an intermediate state and a distinctive state. Between these states there are many hidden states, or transition states, that we cannot visibly distinguish as an observer. If we consider someone giving a wave, there are three visible poses we notice. The ready state where the hands are relaxed, the intermediate state where the hands are bent in front of the person, and the distinctive state where the person has their palm open and tilted about the wrist. However, one can convey a hand wave with two postures, a posture with the wrist tilted to the left and a posture with the wrist titled to the right. With this interpretation we cannot say that a gesture must have three distinct states; namely a ready state, an intermediate state or a distinctive state. A gesture is considered to have temporal properties [25] because gestures consists of a sequence of poses across time. This leads us to consider that states are hidden and non-observable. States maybe hidden but we can observe the output of those states, the postures that each state emit. And a HMM can determine some information about the states through the sequence of observations[2]. Hence, the application of Hidden Markov Models for gesture recognition. The number of states are optimized through experimentation of various gestures gestures. Different gestures have different states, and certain states improve the accuracy of using HMMs to classify similar gestures.

2.7 Training and Classification

A HMM is trained by providing a sequence of observations, symbols from the codebook. However, it is unaware of the posture that a symbol represents. Mathematically, a state has a probability distribution for each observation. The task of training calculates the maximum likelihood estimate of the parameters that best represent the probability distributions. The paramters obtained are the state transition probability matix and the emission probability matrix. The state transition probability matrix dictacts the likelyhood of the transition from one state to another, while the emission probability matrix calculates the likelyhood of an observation occuring given a particular state. The Baum-Welch algorithm calculates the local maximum likelihood estimates given an observation sequence.

	State 1	State 2	State 3
State 1	0.183	0.196	0.621
State 2	0.256	0.744	1.3e-05
State 3	0.999	2.2e-17	3.1e-05

Table 2.1: State transition matrix for a 3-state HMM.

Given an observation sequence to train a HMM for, the Baum-Welch algorithm[2] is a convergence algorithm that begins the likelihood estimation process with a randomized initial state transition probability, and emission probability matrix. An example of a transition probability matrix is shown in table 2.1. The algorithm iteratively improves the maximum likelihood of the occurance of the observation sequence, until the

probability matrices reach a local maxima. When we train a HMM for an input gesture, the postures in the gesture are converted to observation symbols. Which are then given as inputs to the training algorithm. The probability matrices encode the occurance of the observation symbols for a given gesture. Many gestures can be trained under the same HMM as well. When another observation sequence is provided to the algorithm for training under the same HMM, the already existing probability matrices are used as a starting point. The matrices are then optimized for the occurance of the new observation sequence as well.

For classification, motion from the actor is captured in a similar manner that is used to create a training data set, as discussed in previous sections. The captured motion is processed and converted to a sequence of posture frames, which is then translated into a sequence of codebook symbols by comparing feature vectors. This is known as the observation sequence, which serves as an input to a HMM for classification.

Suppose we have a HMM that is trained to recognize the hand wave gesture, and a observation sequence of someone waving their hand is provided. The HMM would return an estimate for the occurance of observations in the input gesture. The then question becomes, does the given sequence of observation make sense for it to be classified as a hand wave? If the answer is yes then the gesture is classified as a hand wave. Mathematically, a HMM returns a likelihood estimate of the input sequence occuring, based on the training provided. If the estimate is high then we consider the input sequence to contain the said gesture, otherwise it is classified as some other gesture. As in figure 2.3, by evaluating the same sequence of inputs in parallel with other trained HMMs, we can obtain their corresponding likelihood estimates for each HMM. We then select the best HMM for the input sequence to belong to by analyzing the likelihood estimates, and classify it under the corresponding gesture label.

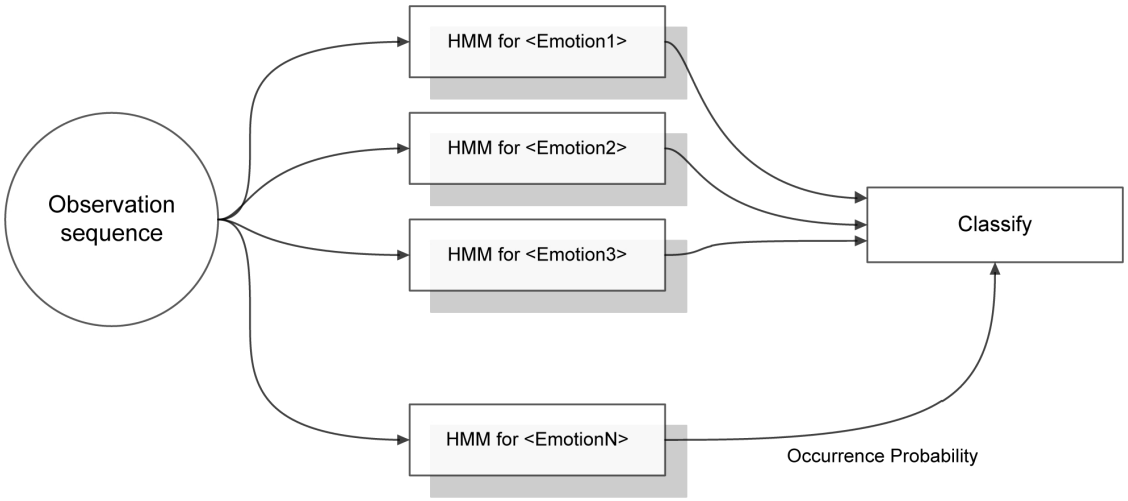


Figure 2.3: Classifying an observation sequence with multiple HMMs.

3. Framework Components

Three principle components were considered when making decisions on improving performance and accuracy of the gesture recognition framework, HMM topology, feature vectors, and codebook selection. Based on empirical evidence through the learning and discovery, that was involved in implementing and running these experiments, it can be said that the HMM topology and feature vectors play a key role in the performance and accuracy of using any HMM. This chapter discusses some of the points that were considered when performing the experiments.

3.1 HMM topology

A HMM Topology describes how the various states could be related to one another. Mathematically, this is described by the probability distribution from one state to another. Keeping in mind the problem of recognizing body gestures while performing an activity, if we consider the emotion of being `angry`, there are various transitions that a body goes through to exhibit an angry gesture.

Considering the simplest case of the angry gesture from a T-posture, first the arms bend in toward the hips, the legs spread apart and finally the hands rest on the hips. These sequence of states indicate that the person is an `angry` emotional state. This is a linear progression of states, a visualization is shown in figure 3.1. The numbers in the figure indicate a posture from the respective codebook page, and the sequence of numbers combined together convey a specific gesture. If the hands are rested on the hips before the legs are spread apart then the sequence of events most likely does not indicate an angry emotion. If one were to train a generic HMM to classify the angry gesture the probability distribution would force the HMM into a linear topology. The left-right linear topology is perfect for recognizing a gesture that is progressive in nature. There are various kinds of angry gestures. Consider an alternative where the hands bend in toward the hips, the legs spread apart, the left hand rests on the hips, while the right hand bent at the elbow points at someone or something. This gesture also represents a progressive left-right topology. Here we have two different gestures, but both indicate a form of `angry` emotion.

In the first architecture, a HMM for each distinct gesture is created (example: angry-1, angry-2,. . . , fear-1, fear-2,. . . ,fear-n,. . .), even though they might represent the same emotion category (example: angry,

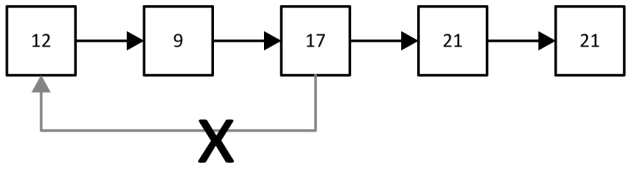


Figure 3.1: A simple left to right linear HMM topology.

fear). In this architecture the system is able to distinguish between various forms of gestures through which the same emotion can be identified. If someone is really angry they might point at someone and oscillate their hands toward them in short bursts. If someone is angry and disappointed, they might shrug the shoulders and throw their hands up in the air swiftly.

In the second architecture, we intend to create a HMM for each distinct emotion (angry, fear,...). In this architecture many similar gestures are grouped together into a single emotion, and some gestures share similar observation sequences, as shown in 3.2. HMM topology can be changed depending on the problem being tackled. Yang and Xu [1] and Ott et al. [16] discuss simple HMM network architectures as discussed here in their research implementation of gesture recognition in hand gestures, and human motion imitation respectively. While Park et al. [18] outline a unique multi-tier HMM topology, where one HMM is trained for every gesture, and then they are all combined together into a single HMM, for their experiment in gesture based robot motion control.

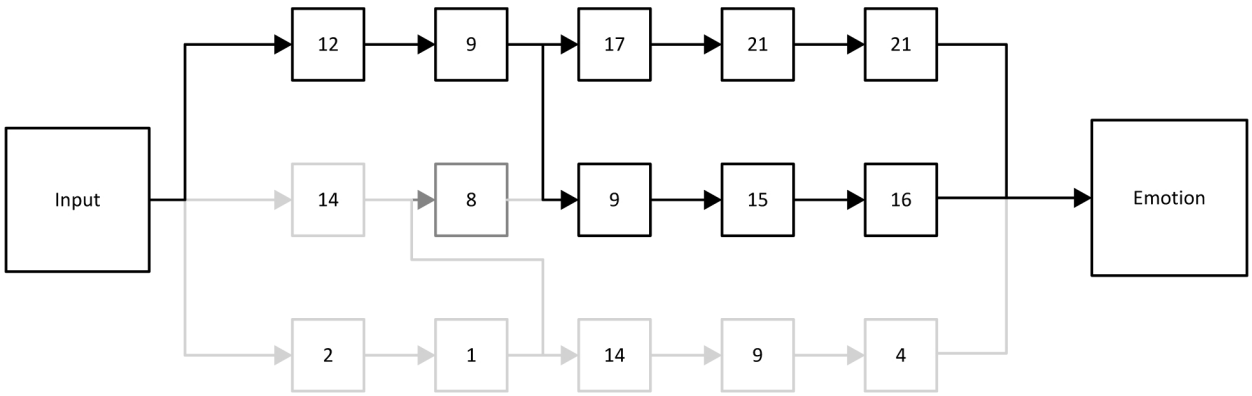
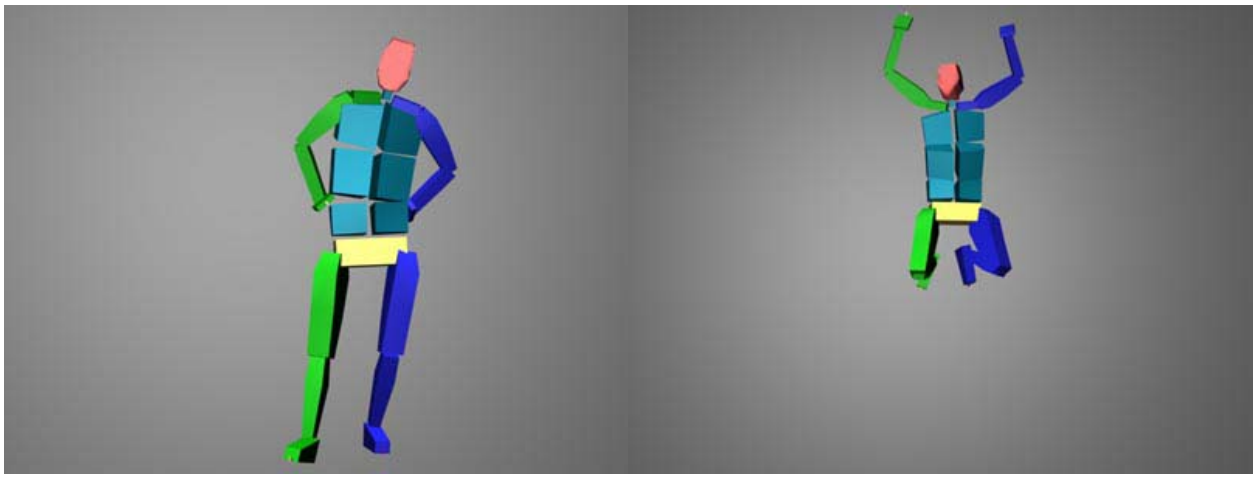


Figure 3.2: A multi-tier HMM topology for an emotion.

3.2 Data sets

The framework was built by conducting experiments with the UCLIC Affective Posture and Body Motion Database created by Nadia Bianchi-Berthouze and Andrea Kleinsmith[19]. This database contained gestures performed by various actors displaying the `angry`, `fear`, `sad` and `happy` emotions. The database also captures the cross-cultural variations of the gestures. Each gesture in the database contains approximately 400-1200 frames of data capture. Only a few frames were selected, at equally spaced time intervals to help in loading and reading the data set. The frames contain position coordinates of the points on the actor's body that were tracked using a motion capture system. In each frame 47 3D position coordinates were extracted; examples, include location of left wrist, right wrist, left knee, right knee.



(a) Angry (b) Happy

Figure 3.3: Posture samples from UCLIC Affective Posture and Body Motion Database[19]

Once the initial framework was developed, and the performance fine tuned, it was decided that a test should be done on the robustness of the framework. So another data set was obtained from UCLIC Affective Posture and Body Motion Database however this data set contained non-acted[20] motion capture data. In the non-acted data set the actors were not acting a specific emotion, rather they were asked to perform a task. Similar to the acted data set, a motion capture system captured the movements of the actor for later analysis. Whilst performing the task the actors expressed various emotions which were categorized into four groups[20]: *concentrating* (determined, focused, interested); *defeated* (defeated, give up, sad), *frustrated* (angry, frustrated), and *triumphant* (content, excited, motivated, happy, victory). However, the data set was not labeled into the above categories when we received them, we classified the data set manually with the following labels instead, *positive*, *negative* and *neutral* emotions. Unlike the acted data set this data set contained Euler joint angles rather than position coordinates. The data set was converted to position coordinates, extracting 23 3D position coordinates on the actor's body. A single data capture in the data

set contained approximately 15000-30000 frames, from these frames only a few hundred were extracted that contained a gesture being performed by the actor.

For the final experiment, Xbox Kinect was used for creating a dataset of four different gestures. The four gestures that were performed are `shurg`, `shake fist`, `surprised` and `thinking`. Only upper body join positions were captured in the dataset as these gestures primarily exhibited upper body motion. Approximately 10-12 gesture variations of the primary gestures were captured, for a total of 46 gestures. The gestures were performed across a time length of 1-2s, and the sensors captured 3-4 frames per second.

Features in the data set

Table 3.1 lists the features that were recorded by the motion capture system in the data sets. Features can be added or removed as necessary to get better results. For example, developing a system that only uses recognizes upper body postures, we can remove features related to the lower body without affecting the results.

3.3 Feature vector

Elbow angle, position of wrist, knee angle, angle at the hips, head tilt, are some of the data that one needs to assess to distinguish one gesture from another. Together, they form a feature vector. Having the right feature vector is essential in classifying and recognizing gestures. If one has a feature vector consisting of upper body joint positions alone, the accuracy of recognizing gestures that requires having information about the lower body point positions will be low. Likewise, the information in the feature vector is also important, a feature vector consisting of joint points and a feature vector consisting of joint angles might provide different recognition results.

Dimensioned data

Dimensioned data provide a means to an end in understanding how the HMM framework works in classifying and recognizing sequence of patterns. In the experiments conducted the dimensioned data consisted of having 3D positions of various points of the human body. When these points are plotted for a single frame it is easy to visualize, figure 3.4, how the HMM framework works to recognize and classify gestures. Plotting a sequence of frames relay information about the gesture being performed. However, there are greater disadvantages to using dimensioned data in a classification and recognition algorithm. This data might convey information visually to a human observer, but an algorithm sees no advantage in having dimensioned data. In fact, dimensioned data restricts the algorithm from performing equally well when the input data conveys the same gesture but is performed by a different actor. Different actors exhibit different dimensioned data for

CSM File (Acted data set)	
Left Front Head	Left Front Waist
Left Back Head	Left Back Waist
Right Back Head	Right Back Waist
Right Front Head	Right Front Waist
Top Chest	Top of Spine
Center Chest	Middle of Back
Left Outer Knee	Right Outer Knee
Left Inner Knee	Right Inner Knee
Left Outer Ankle	Right Outer Ankle
Left Heel	Right Heel
Left Outer Metatarsal	Right Outer Metatarsal
Left Inner Metatarsal	Right Inner Metatarsal
Left Toe	Right Toe
Left Shoulder	Right Shoulder
Left Outer Elbow	Right Outer Elbow
Left Inner Elbow	Right Inner Elbow
Left Wrist Stick End	Right Wrist Stick End
Left Wrist Stick Base	Right Wrist Stick Base
Left Wrist Inner near thumb	Right Wrist Inner near thumb
Left Wrist Outer opposite thumb	Right Wrist Outer opposite thumb
Left Hand	Right Hand
BVH File (Unacted data set)	
Left Hip	Right Hip
Left Knee	Right Knee
Left Ankle	Right Ankle
Left Collar	Right Collar
Left Shoulder	Right Shoulder
Left Elbow	Right Elbow
Left Wrist	Right Wrist
Head	
Neck	
Xbox Kinect joint captures	
Left Hip	Right Hip
Left Knee	Right Knee
Left Ankle	Right Ankle
Left Shoulder	Right Shoulder
Left Elbow	Right Elbow
Left Wrist	Right Wrist
Nexk	
Head	
Spine	

Table 3.1: Feature labels recorded in the motion capture data sets.

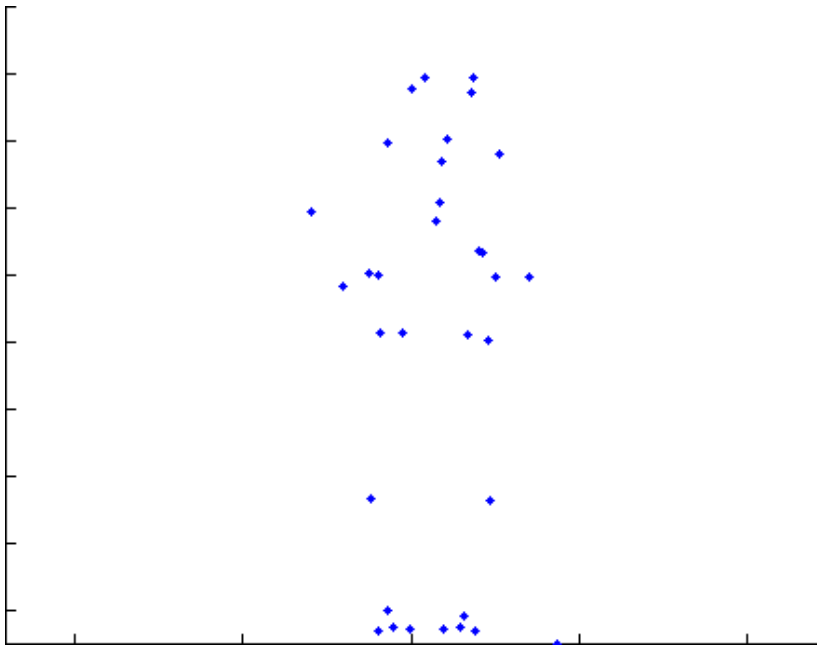


Figure 3.4: Visualization of a dimensioned feature vector (position).

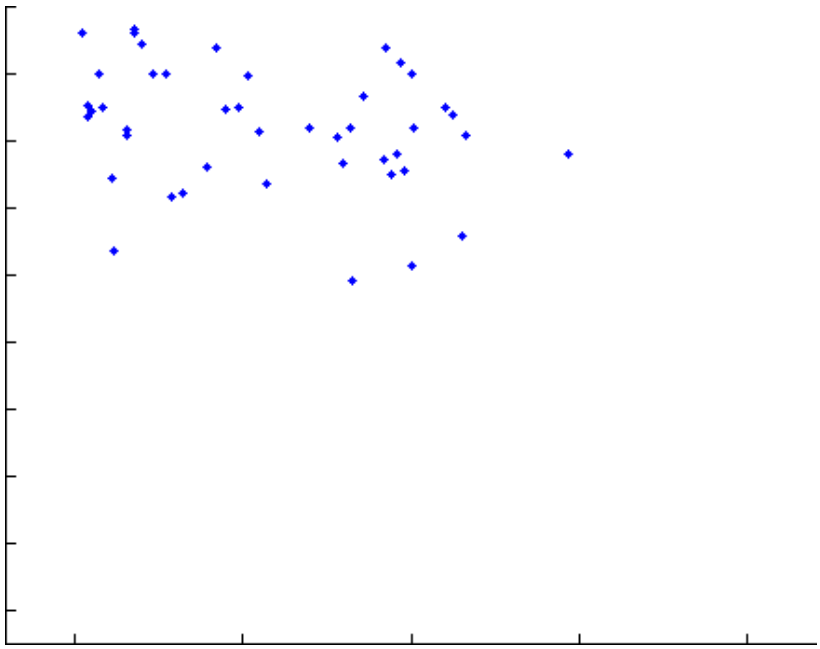


Figure 3.5: Visualization of a dimensionless feature vector (velocity).

the same gesture, the actors height, length of arms, length of legs are all different. Though a human observer is able to filter out these data when trying to recognize the gesture being performed, an AI algorithm is unable to filter or transform this data in such a way that its recognition and classification is unaffected. Thus, the need for dimensionless data.

Dimensionless data

Dimensionless data play an important role in machine learning, specifically in gesture recognition, it allows data to be generalized for any actor regardless of their physical features. When we compare angry, fear, happy or sad gestures from one actor to another, they will be quite similar when dimensionless data is used. Dimensionless data represent temporal data of a gesture, such as velocity and acceleration of points on a human body, or of joints; essentially data that represents a state in time. Such data is difficult for a human observer to visualize and categorize effectively. Figure 3.5 is the corresponding velocity feature vector of the posture presented in figure 3.4. Consider looking at the performance of an angry gesture. By observing the velocity of points on a human body one cannot classify the gesture being performed without the help of spatial data. However, this provides no challenge for a machine learning framework, since they will be trained to recognize gestures from this perspective. Point velocities are essentially velocity of data points on a human body at an instance in time. We realize that gestures are dependent on speed at which it is being performed. If someone ducks within a second, it can be said that they reacted to a dangerous situation. If they ducked slowly, they are most likely reacting to a neutral situation.

3.4 Codebook

Vector quantization plays an important role with any HMM framework, they are necessary to reduce the dimensionality of the feature vector. The generation of a codebook is a vector quantization process. The idea, is to take a feature vector and assess all possible ranges across which the feature can have values. Then, this range is quantized, divided into many parts, with some structure. For gesture recognition with motion capture data, two different codebooks techniques are considered in the experiments. Any feature vector considered for training, or classification, is converted into one of the codebook pages that best represent the feature vector.

The first codebook involves the selection of template postures from the training data. These postures were then saved in the template database, which is the codebook. An example of a codebook is shown in figure 3.6. This template matching technique is used by Lee and Wong [3] and Park et al. [18]. An advantage to creating such a codebook is that the templates tend to be unique and of value, as it is most likely to be hand-picked by a human. The disadvantage is that if there is a large data set from which the templates need

to be chosen from, it will be a tedious task.

The second codebook involves the use of the k-means clustering algorithm to group similar postures together, and these similar postures are put in the same cluster. The k-means algorithm is a common method used in the creation of codebook[1]. Thus, any gesture that has a posture where the actor is raising their hand would have the same codebook page for that posture. If the number of clusters is large, then variations of the same posture would be spread out, for a small cluster the variations would fall in the same cluster.

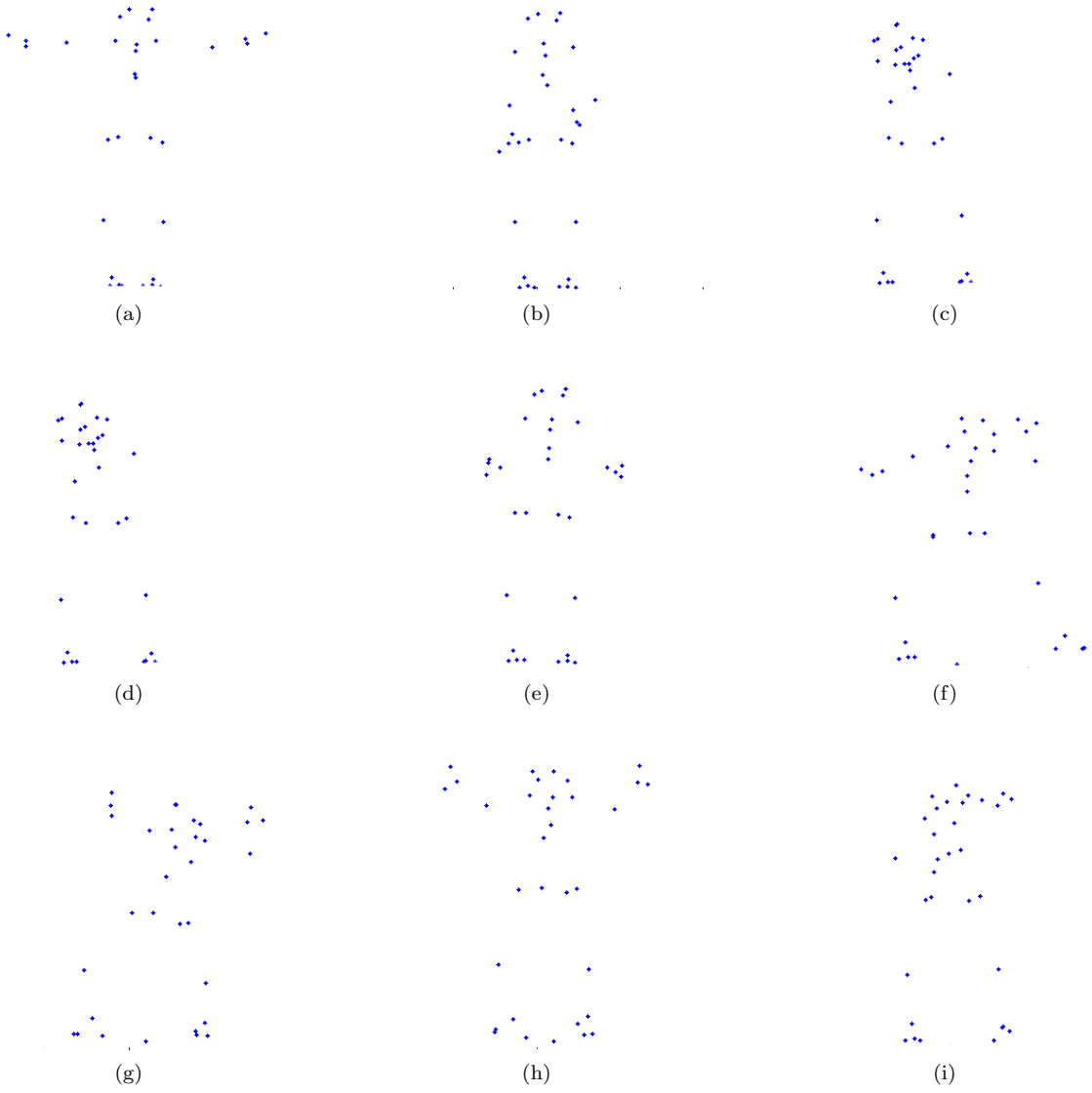


Figure 3.6: Dimensioned postures stored in a selected Codebook.

4. Experiment Results and Analysis

4.1 Initial discovery

The first step in the experiment was to understand how the HMM framework worked, a dimensioned feature vector was used for this purpose. A left-right linear HMM was used for each gesture, as shown in figure 3.1. The codebook vectors were obtained manually. Table 4.1 summarizes the results obtained from the various test runs. These experiments used the acted and labeled data set from Kleinsmith et al. [19].

The first type of test consisted of 70 gestures, 35 of which were used for training and 35 for training. We realized that the training sample was quite low, so a second round of tests were conducted which consisted of 90 gestures, 35 were used for testing and 55 for training. Again a third round with even more training data was conducted, with 120 gestures in total, 85 for training and 35 for testing. From the table we can see that the total classified gestures improves as we increase the number of training samples. The number of correctly classified gestures also improves, but at a very low rate. To understand the low classification rate it was necessary to take a closer look at the framework.

	Correctly Classified (%)	Total Classified	Testing data	Training data	Total Gestures
Manual #1	8 (22.8)	10	35	35	70
Manual #2	10 (28.5)	24	35	55	90
Manual #3	11 (31.1)	31	35	85	120
K-means	3 (8.5)	10	35	55	90

Table 4.1: Recognition

While training a HMM for a specific gesture, the framework produced the sequence of observations listed in table 4.2 for one of the fear gestures. Each number corresponds to a page in the codebook, where the contents of the page is a single frame of data from the motion capture system. Thus, the frame in page 1 appears after the frame in page 23, similarly the frame in page 10 appears after the frame in page 1, so on. The HMM training algorithm trains a probabilistic model for such a pattern of observation, such that if the sequence in the pattern is altered the HMM would return a low probability for that sequence to appear.

Now, we look at the the observation pattern for one of the fear gestures in the testing set that was misclassified as one of the angry gestures. When the pattern is compared against the fear gesture in the

Fear in Training	28	28	28	28	28	28	28	28	14	14	14	30	30	30	30
Fear in Testing	21	21	21	21	21	21	10	10	10	10	10	10	20	32	33
Angry in Training	28	28	28	21	21	21	21	21	21	21	28	28	28	28	28

Table 4.2: Sequence of observations for an incorrectly classified fear gesture.

training set, it is very different. However, when we compare the pattern to an angry gesture in the training set, half the pattern matches a sequence of observations in the angry pattern. Though only the repetition of the number 21 in the angry gesture matches the fear gesture, the HMM classifies it as angry with a very high probability.

Angry in Training	24	24	24	24	24	24	21	21	22	22	22	22	21	21	24
Angry in Testing	24	24	24	24	24	21	22	22	22	22	21	23	23	24	24

Table 4.3: Sequence of observations for a correctly classified angry gesture.

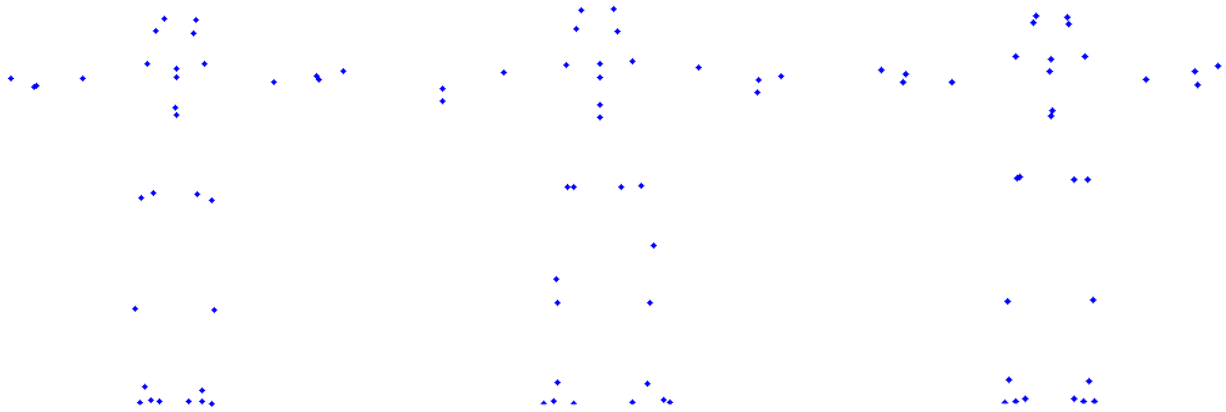
Table 4.3 displays the observation pattern for a correctly classified angry gesture. Note the similarity of the patterns between the training angry gesture and the testing angry gesture. The HMM classified this as an angry gesture with a very high probability. From these tables we conclude that both, the observation that follows the current observation, and the repetition of the same observation in a sequence plays a key role in determining the outcome of a HMM.

If one were to extract template frames from a training data set of 90 gestures, approximately 1-2 template frames per gesture, we would have approximately 225 templates. This also means that our codebook has 135 pages. A large codebook means a larger training data set is necessary to estimate a large observation probability matrix for our HMM. We quickly realize that many of the gestures share similar postures, a manual selection of templates ensured that the posture selection for codebook pages were unique in nature. Through observations and visual analysis we realized that in our experiments the problem lies in the creation of this codebook. Manual creation of the template frames though accurate was infeasible when the training data set became larger and larger. It was not as straight forward as selecting a few universal template for angry gestures, and use these templates to convert the training data set into a sequence of observation patterns as in tables 4.2 and 4.3. It was necessary to go through each training data set to identify unique gestures, and extract frames from them to properly classify similar gestures. The training data itself contained new patterns that needed to be extracted.

We introduced a clustering algorithm to replace the manual codebook creation process. The `k`-means clustering algorithm, creates `n` clusters and groups similar data together in the `i`th cluster of the `n` clusters. The algorithm clusters similar data by calculating the distances of the centroid of the given data, thus data that is closer to each other are grouped in the same cluster. Now, the codebook creation was simplified for a large data set. Table 4.1 lists the results of the test runs conducted after using the clustering algorithm for codebook creation. The results are not impressive at all, visual analysis of the codebook vectors and the

observation patterns created for each posture tells us why.

Having a codebook that is as large as 225 pages, even for small variations between two similar gestures the observation sequences are widely different, as in the fear gesture in table 4.2. Consider the images in figures 4.1, they represent figures from codebook pages 2, 11 and 18. Visually they are very similar to each other, however the k-means algorithm clusters them into different pages because of small variations that categorize these images as outliers, such as hip angle, position of wrist, head tilt, and spacing between the feet. These are very subtle from the figures, however the k-means algorithm picks up on these.



(a) Codebook Page 2 (b) Codebook Page 11 (c) Codebook Page 18

Figure 4.1: Similar codebook pages

With this in mind, we plotted the postures in the pages listed for the fear gesture, from table 4.2; pages 21 and 28 were similar, and pages 14 and 10 were similar. The misclassification in the experiment is caused by such occurrences, of similar postures being classified into different codebook pages. Similar postures are being categorized under different pages because of small variations in position of points on the human body in 3D-space. We suspect that the use of dimensioned feature vectors is causing this, so we proceed by using dimensionless feature vectors.

We also make one final observation about the current architecture. Every gesture has a HMM associated with it even though it might be similar to an existing gesture that has already been trained. This does not pose a problem if every single gesture in the data set were unique. In our data set, the gestures that belong to the `angry` category are not all unique, many of the gestures are similar with small variations. In a problem that involves humans, whose features and expressions are varied, it is necessary to group up these variations in gestures under a single gesture. Thus, we consider training all unique and non-unique gestures that belong to the `angry` category under one HMM. If this HMM results in a high probability of a gesture occurring during testing, then it will be classified as `angry`. This is a multi-tiered HMM topology as shown in figure 3.2.

4.2 Improving performance

The initial experiments conducted above has given us important insights into how gesture recognition heavily depends on organizing the codebook pages. The second part of the experiment modified the HMM architecture such that similar gestures were categorized and trained under a single HMM. Thus for a data set that contains four emotions (angry, fear, sad, and happy) four HMMs were created.

	Angry	Fear	Happy	Sad
Angry	13	2	0	1
Fear	11	7	6	3
Happy	2	1	11	6
Sad	3	1	1	19

Table 4.4: Confusion matrix for 3-state HMMs with velocity feature vectors.

Table 4.4, displays the confusion matrix for one of the test runs that classified 98% of the testing data, and 56% of the testing data were classified correctly with high probability. The test run had a data set of 192 gestures, 93 were used for training the HMMs, and 89 were used for testing. This test run used velocity feature vectors of the training and testing data, for codebook creation and template matching respectively. Note the classification rate is very high, this is caused by the change in the HMM architecture from using one HMM for every gesture to one HMM for every emotion, where similar gestures are grouped together. This change resulted in having a sufficient amount of training data for each HMM, approximately 23 gestures were used to train each of the angry, fear, sad, and happy gestures. The confusion matrix allows us to identify areas where improvements can be made, we note that most of the `fear` gesture is being misclassified as `angry`. The only property to control in a HMM model is the number of states associated with it. The results in table 4.4 are for HMM models with three states (for each emotion). Table 4.5 lists the confusion matrix for a table where the `angry`, `happy` and `sad` HMMs have three states and the `fear` HMM has four states. We note that the classification of the `fear` emotion has improved slightly, but the classification of the

	Angry	Fear	Happy	Sad
Angry	5	4	3	4
Fear	5	9	9	4
Happy	1	4	9	6
Sad	1	2	1	20

Table 4.5: Confusion matrix where fear HMM has 4-states.

`angry` emotion has dropped. Comparing table 4.4 and 4.5, the classification of `happy` emotion has dropped and now the majority of the `fear` motions are also being classified as `happy`. We proceed by adding more training data to see if this has any effect. A caveat that needs to be noted here is that, if there are gestures that are unique to a particular emotion that were not being classified properly but now this gesture is in the training data, this gesture will not be tested for in the testing data.

	Angry	Fear	Happy	Sad
Angry	4	2	0	0
Fear	0	5	2	1
Happy	1	0	4	2
Sad	1	2	0	6

Table 4.6: Confusion matrix for 3-state HMMs with additional training data.

Table 4.6 lists the confusion matrix for which there were 152 gestures for the training data, approximately 38 gestures for each emotion, and 30 gestures for the training data. All the data were classified and it achieved a 63% correct classification rate. With the caveat, that some unique gestures might not be in the testing data, in mind, we can say in general that higher training data improves the recognition rate of the HMMs. One scenario to think about is how are the classification rates when only two of the emotions are trained and tested for.

	Angry	Happy	States	Recognition rate
Angry	11	11	3	71%
Happy	2	24	4	
Angry	15	7	3	78%
Happy	3	23	3	
	Angry	Sad		
Angry	14	6	3	81%
Sad	1	20	3	
Angry	18	2	3	83%
Sad	4	17	2	
	Angry	Fear		
Angry	17	4	3	66%
Fear	10	12	3	
	Fear	Happy		
Fear	14	4	3	72%
Happy	8	20	3	
Fear	11	7	2	74%
Happy	4	24	4	
	Fear	Sad		
Fear	15	9	4	77%
Sad	0	21	3	
Fear	19	5	4	85%
Sad	0	21	4	
	Sad	Happy		
Sad	14	4	4	61%
Happy	2	13	4	

Table 4.7: Confusion matrix for comparison against each emotion.

Table 4.7 displays a confusion matrix, recognition rate and the number of states used for each emotion's HMM. We note that when comparing two emotions to each other an average of 15-20% misclassification rate exists. Table 4.7 also displays improvements on recognition rate when comparing some emotions, for example `fear` vs. `sad`. Training the HMMs with 4 states fear and 3 states for sad, a recognition rate of 77% is achieved, but result is improved upon by training both HMMs with 4 states. Optimization of HMMs

for states is an important step in improving the recognition rate. This property not only affects how well the respective gestures are classified (a sad gesture classfied as a sad emotion), but also how other gestures are classified against it (a fear gesture classified as a sad emotion). Improving a classification rate for one gesture might affect the classification rate of another gesture. Thus when one or more emotions are involved, as in tables 4.5 or 4.6, a balance of states between each individual HMM needs to be achieved.

4.3 Measured Performance

It is insightful to look into the overhead experienced by each component of the HMM process. There are two areas where the overhead is significant, when creating a codebook and when training a HMM. Other parts of the process such as extracting feature vectors from the inputs, or even classifying HMM recognition experience very little overhead. Table 4.8 displays the measured performance of the various components of the framework over time. In general, the training data size and the overhead seen are directly correlated. The preprocessing, feature vector extraction, codebook creation and HMM training are *one time* overheads, during the setup phase. When a single new data needs to be trained on top of the existing models, the overhead is very minimal. The performances were measured on a Intel i7, Quad Core CPU, with 8 GB memory.

Training Data Size	10	90	152
Feature Vector Extraction	1.5s	30.5s	44.2s
Codebook (k-means)	1s	25.0s	61.2s
HMM Training (total)	1s	74.9s	37.6s
HMM Recognition (total)	-	1.4s	1.77s

Table 4.8: Timed performance

The overhead of the feature vector extraction process is linear in nature, and is very dependent on the process itself. If we are given a data set with spatial positions then algorithmically it is easy to calculate or extract feature vectors from it. If we are given a data set with static images from a camera then the time taken to process the image, isolate human features, calculate depth and positions becomes significant. The overhead of the codebook creation using the k-means algorithm depends on the length of the feature vector, and the number of feature vectors. As the k-means is an optimization algorithm that iteratively moves data around till the best cluster of data is found, it also depends on the values of the feature vector.

Figure 4.2 shows a graphical comparison of the codebook creation time versus the number of features in the feature vector. The amount of time needed to create the codebook using the k-means algorithm increases linearly as the size of the feature vector grows. Figure 4.3 shows an alternative comparison of codebook creation time with the number of feature vectors to cluster. Similar to the other graph, the amount of time needed to create the codebook increases linearly as the number of feature vectors to cluster

is increased. We notice that figure 4.3 contains some outliers where some sets with a higher number of

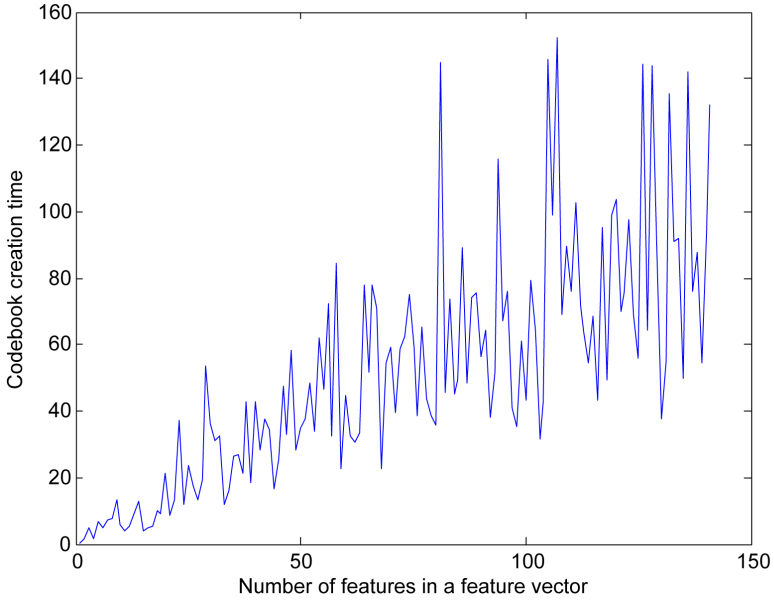


Figure 4.2: Graph of codebook creation time (in seconds) versus number of features in a feature vector.

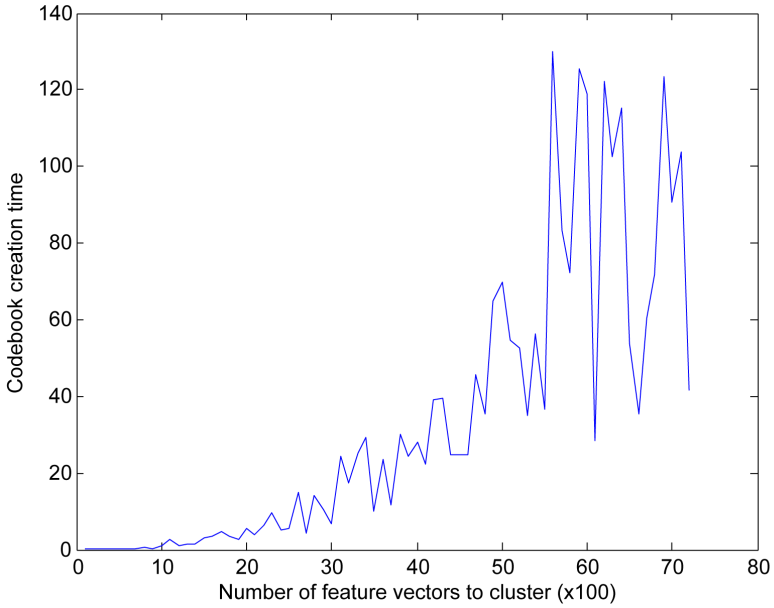


Figure 4.3: Graph of codebook creation time (in seconds) versus number of feature vectors to cluster.

feature vectors tend to have their codebooks created faster than others, this is because of the selection process of the k-means algorithm. If the algorithm makes an initial guess for cluster classification that leads to its convergence criteria faster, then it takes less time for creating the codebooks.

Table 4.8 also lists the overhead incurred for HMM training. This is the total training time for all four emotions. Individually, the training of each emotion depends on the observation sequence pattern and the rate of convergence of the HMM probability matrices. Figure 4.4 shows a graph of the training times of a

HMM as the number of gestures to be trained are increased. Similar to codebook generation, HMM training depends on the convergence of the training algorithm as new gestures are introduced for training. The higher the number of new gestures the longer it takes for training. There are instances when the training time is reduced, that is the convergence of the probability matrix is faster. This is because the gestures are similar to already trained gestures, thus the probability matrix sees less or no improvement at all. Outliers also exist, when the training does not converge to a local optimum we see a large overhead in waiting for the HMM to complete its training.

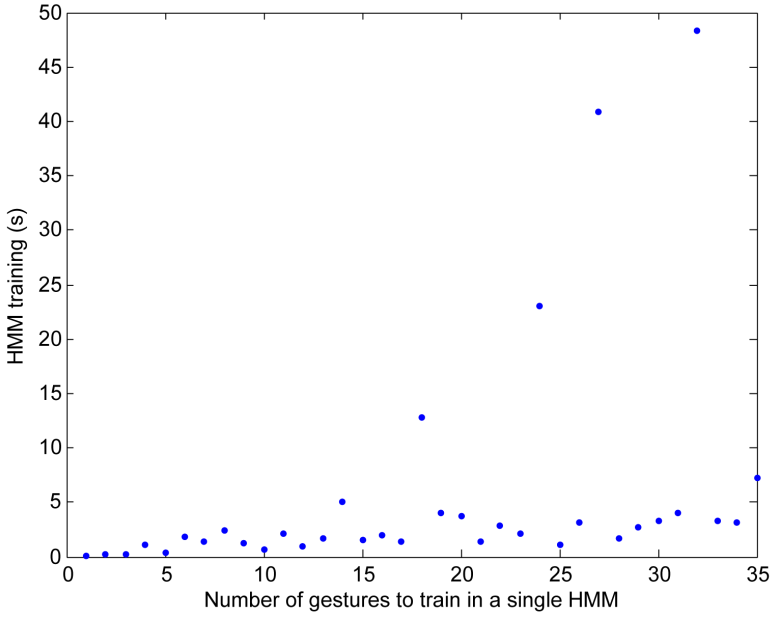


Figure 4.4: Graph of HMM training time (in seconds) versus number of gestures to train for.

4.4 Unacted data set

The experiments that have been discussed until now were performed on an acted data set, where the actor acting the gesture was asked to specifically demonstrate an emotion. A test was also run on an unacted data set from Kleinsmith et al. [20]. In this data set the actor was asked to perform a task, during this task various emotions were extracted and classified. The idea was to capture the natural essence of an emotion. The gestures were classified into three categories, `positive`, `negative`, and `neutral`. A `positive` emotion consisted of content, excited, motivated, happy or victory gestures, while a `negative` emotion consisted of defeated, give up, sad, angry, and frustrated gestures. All other gestures were considered `neutral`. Even for a human observer it was difficult to label the data set with one of the emotions, thus training a HMM to detect and recognize such gestures would be a test in robustness.

From the tests performed on the acted data set, we see that a dimensionless velocity feature vector with one HMM for each emotion gave us better performance. We apply the same preference towards training

	Positive	Negative	Neutral
Positive	6	1	1
Negative	2	3	1
Neutral	2	2	11

Table 4.9: Confusion matrix for the unacted data set.

and testing the unacted data set. Also, this data set had a smaller feature vector than the acted data set, as displayed in table 3.1. Table 4.9 displays the confusion matrix for the unacted data set. A total of 64 gestures were extracted and labeled manually from the data set, of which 33 was used for training the HMMs, and 31 for testing. It is important of to note the distribution of the emotions in the data set. Out of the labeled gestures, 19 were labeled as `positive`, 17 as `negative` and 28 as `neutral` gestures. The framework recognized 61% of the gestures correctly. The `positive` and `negative` HMMs used 2 states while 3 states were used for the `neutral` HMM.

	Positive	Negative	Neutral
Positive	3	1	0
Negative	0	2	3
Neutral	0	1	4

Table 4.10: Confusion matrix for the unacted data set.

A modified test was run to see how additional training data affects the classification results; 49 gestures were used for training, and 15 were used for testing. Table 4.10 displays the results of the test. We note that in both cases (tables 4.9 and 4.10) the `negative` emotion is performing poorly. The tests did not use a sufficient amount of gestures for training to conclude the reasons behind the misclassification concretely. 60% of the testing data was classified correctly. Thus we see no improvement in adding more training data in this case, but an observation to note was that only 2 additional `negative` gestures were added to the training data in the modified test, as more `negative` gestures were unavailable.

The purpose of these tests were to determine the robustness of the HMM framework for recognizing gestures under different circumstances (acted, or unacted). From the results above we can safely conclude that the HMMs are quite robust to adapt to situations as long as sufficient training is provided.

4.5 Xbox Kinect data set

A final experiment was performed with motion captured data from Xboc Kinect. This was a custom experiment to test the framework's performance with Xbox kinect data capture, also acted as a performance sanity check. Brian was to be outfitted with a Kinect camera to capture user interactions, as they are interacting with the social robot. When a person enters the Kinect's view, it starts calibrating joint locations on the user for tracking in real time. This provides us with joint spatial data.

The code that was used to capture data from the Kinect provided us with 3-4 frames per second. We believe that the code could be optimized further to provide us with a higher frame rate. However, 3-4 frames per second was more than sufficient to distinguish the four gestures that were performed for training, testing and classification. The four gestures that were performed are `shurg`, `shake fist`, `surprised` and `thinking`. Note that we have switched from classifying emotions to classifying gestures, this was due to the limited scope of the experiment. However, we follow a similar setup that ensures that the same steps for emotion classifications are applied. Even though we have four gestures, each gesture was performed 10-12 times, that is there were 10-12 varitions of the same gesture to give us a total of 46 gestures. We consider `shurg`, `shake fist`, `surprised` and `thinking` as emotion labels for the purpose of this experiment.

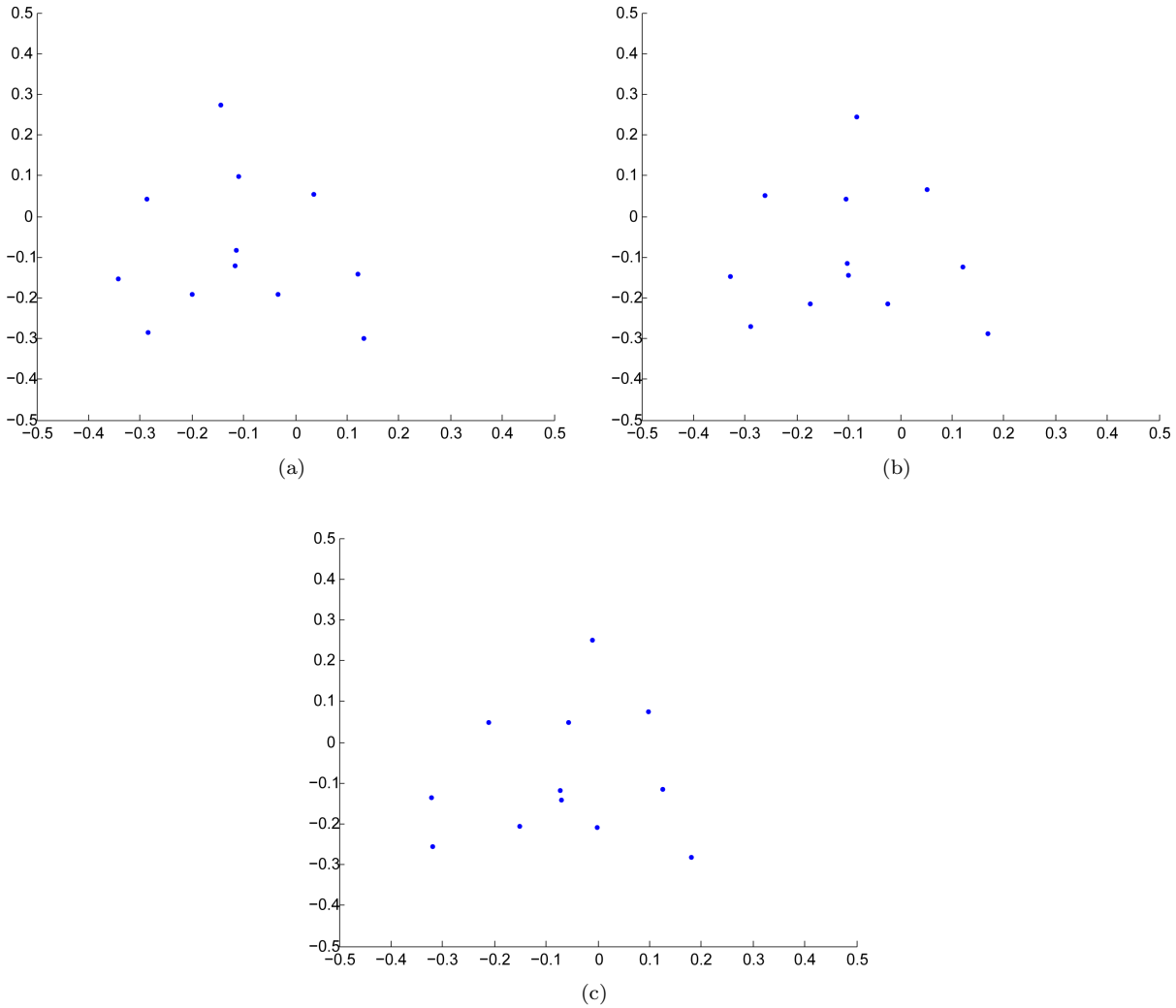


Figure 4.5: Sequence of posture progression in a `thinking` gesture

Figure 4.5 displays three of the frames captured in the thinking gesture. Note the subtle changes in the position of the head and the hands. Figure 4.6 displays two of the frames captured in a shrug gesture. Note the change in position of the hands. Something to realize is the power of HMMs to detect and appropriately

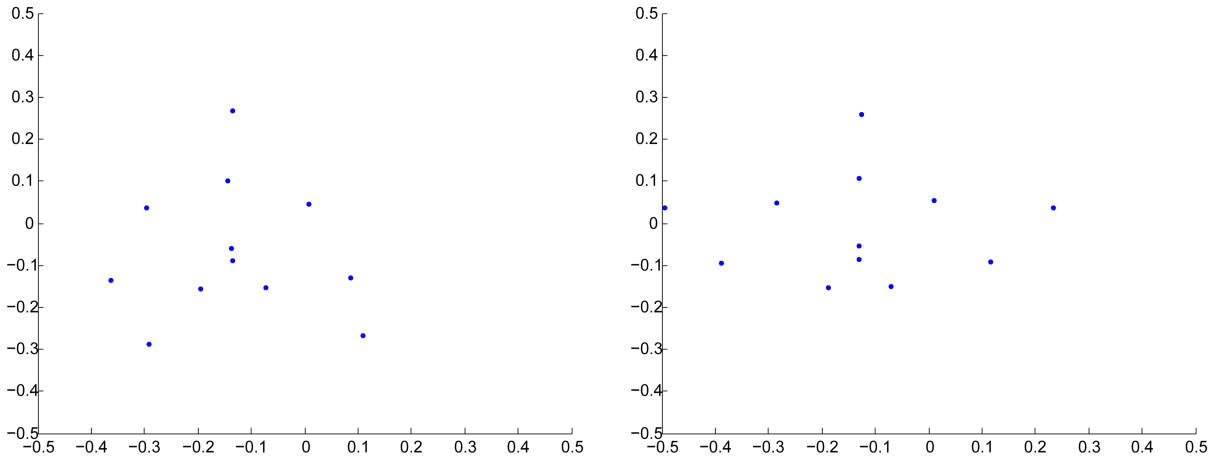


Figure 4.6: Sequence of posture progression in a `shrug` gesture

classify these subtle changes in postures among different gestures.

Once the data was captured, the same steps as the other experiments was followed to extract velocity feature vectors and the k-means algorithm was used to create a codebook. With the training observations created, the HMM was trained for each emotion label with the corresponding gestures. The initial test run consisted of 22 training data and 24 testing data. Table 4.11 contains the confusion matrix for the 3-state HMM that was trained and tested against. The classification rate for this test run was 59%, 13 out

	Shrug	Surprised	Thinking	Shakefist
Shrug	4	0	0	0
Surprised	0	2	6	0
Thinking	1	0	1	0
Shakefist	6	0	2	0

Table 4.11: Confusion matrix for 3-state HMMs with velocity feature vectors for Xbox Kinect data set.

of 22 gestures in the testing data set were classified correctly. From table 4.11 we notice majority of the `surprised` and `shake fist` gestures are being missclassified. The `shurg` gesture has a 100% classification rate, but the `shake fist` gesture is being missclassified mostly as the `shrug` gesture. To identify the cause of miss classification we review the contents of the shrug and shakefist gestures visually. Visually the gestures are quite different, as the shrug was dependent primarily on shoulder movements, while the shake fist was dependent on a moving hand in front of the actor. Next we analyze the gestures in the training data set, and found that only one shakefist was being used to train the HMM. As such the variations in this gesture was not being captured.

The second test run involved shifting more data from the test data set to the training set. The gestures were shuffled and randomly selected for testing and training. It was also ensured that sufficient amount of gestures were available to provide training for each emotion label. For training 31 gestures were used, while

15 gestures were used for testing. Similar to the first test run, a 3 state HMM for each emotion label was trained. Table 4.12 displays the confusion matrix of this run. Out of the 15 gestures, 12 were classified

	Shrug	Surprised	Thinking	Shakefist
Shrug	3	0	0	1
Surprised	0	3	1	0
Thinking	0	2	2	0
Shakefist	0	0	0	3

Table 4.12: Confusion matrix for 3-state HMMs with velocity feature vectors for Xbox Kinect data set.

correctly giving us a recognition rate of 80%. We notice that the shakefist and surprised gesture recognition has improved significantly from the first run. This leads us to the obvious conclusion that having a lack of necessary training data will result in overfitting of the observation sequence probabilites in a HMM.

Noise variations and retraining

We perform one last experiment with this data set to see how the recognition results are affect with noise. The data set was divided into two, one of which was modified with noisy values. This simulates a random noise disturbance in the motion capture system. Both sets of data were shuffled and re-divided into test and training data sets randomly. This time we retraind the existing HMMs that were trained for the previous test run. The retraining process is similar to the training process, except instead of the random initial probability matrices, we start with an existing probability matrix from the previous run. Similar to the last experiment, 31 gestures were used for training and 15 gestures were used for testing. Table 4.13 displays the confusion matrix for this test run. In all the test run achieved a correct classification rate of 73%. To

	Shrug	Surprised	Thinking	Shakefist
Shrug	4	0	0	0
Surprised	0	2	2	0
Thinking	0	1	3	0
Shakefist	1	0	0	2

Table 4.13: Confusion matrix for 3-state HMMs with noisy Xbox Kinect data set.

improve the missclassifications we would need to provide more training data, or work on optimizing the HMM parameters, such as the probability transition matrix, emission probability matrix, or the number of states allowed in a gesture. Overall, these results tells us of the success of the experiment in implementing it for use with Xbox Kinect. Assuming we provide sufficient training for each emotion or gesture that needs to be classified, HMMs are quite robust to variations, very easy for re-training and adaptation.

5. Future recommendations

In the experiments detailed in this report, many performance improvements have been made across the components described above. HMM as an algorithm itself can see some improvements in training. but keeping in focus the applications of HMM, there is plenty of room for growth in optimizing the components further before we debug the theory behind HMM for improvements. Any future research in gesture recognition using HMMs should focus on obtaining better feature vectors. The velocity feature vectors provided us with sufficient performance improvement in the experiments, we suspect that a hybrid feature vector with velocities, and angles, would give us even better performance. Employment of a better algorithm for codebook creation would also improve the performance of HMM framework recognition significantly, as the codebooks are the basis of the symbols in the observation sequences. We believe that we have explored the best options for HMM topology, and it should of less concern in further work.

6. Conclusion

In this report, we have demonstrated the application of Hidden Markov Models (HMMs) to recognize gestures in motion captured data. The existing works demonstrate how HMMs are very applicable in the field of gesture recognition because of their ability to train and recognize over spatio-temporal data. We start by discussing the process involved in using HMMs for gesture recognition. Then we discuss in detail some of the components related to the architecture of the over all recognition process.

We identify and discuss four key components in the framework: HMM topology, feature vector, data sets, and the codebook. We propose a linear HMM topology and a multi-tier linear HMM topology. We identified the data sets and the purpose of each data set in the experiment. We explore feature vector selection, and identify the importance of dimensionless feature vectors. And finally, we discuss codebook construction by the use of template matching, and alternatively by applying a k-means algorithm.

The initial discovery phase of the experiment, analyzed the observation sequences and determined the cause of poor performance. We determined the cause of misclassification of gestures in the recognition process, and the steps necessary to improve it. We also analyzed how the codebook played an important role in HMMs. It was also established that the manual selection of template database is not feasible, as the data size increased. Later, by using a dimensionless feature vector and a multi-tier architecture for classifying similar gestures into emotions, we achieved a gesture recognition rate of 63%. Finally, we test the robustness of the framework to be able to adapt and perform equally well for a different data set. In this final experiment the framework achieved a recognition rate of 60%.

The report has provided a detailed account of the components and the analysis of the framework (process) involved in using HMMs for recognizing gestures in motion capture data. Achieving a 63% recognition rate; by supplying sufficient, and related, training data and tuning HMM properties specifically for recognizing emotions, an even higher recognition rate could be achieved.

Bibliography

[1] Jie Yang and Yangsheng Xu. Hidden markov model for gesture recognition. Technical Report CMU-RI-TR-94-10, Robotics Institute, Pittsburgh, PA, May 1994.

[2] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4 –16, jan 1986. ISSN 0740-7467. doi: 10.1109/MASSP.1986.1165342.

[3] Jason Lee and Willy Wong. A stochastic model for the detection of coherent motion. *Biological Cybernetics*, 91:306–314, 2004. ISSN 0340-1200. URL `http://dx.doi.org/10.1007/s00422-004-0516-0`. 10.1007/s00422-004-0516-0.

[4] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(34):143 – 166, 2003. ISSN 0921-8890. doi: 10.1016/S0921-8890(02)00372-X. URL `http://www.sciencedirect.com/science/article/pii/S092188900200372X`. ¡ce:title¿Socially Interactive Robots¡/ce:title¿.

[5] J. Picone. Continuous speech recognition using hidden markov models. *ASSP Magazine, IEEE*, 7(3): 26 –41, jul 1990. ISSN 0740-7467. doi: 10.1109/53.54527.

[6] Jianying Hu, M.K. Brown, and W. Turin. Hmm based online handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(10):1039 –1045, oct 1996. ISSN 0162-8828. doi: 10.1109/34.541414.

[7] Zhang Youzhi. Research and application of hidden markov model in data mining. In *Geoscience and Remote Sensing (IITA-GRS), 2010 Second IITA International Conference on*, volume 1, pages 459 –462, aug. 2010. doi: 10.1109/IITA-GRS.2010.5602641.

[8] Xiang Ma, D. Schonfeld, and A. Khokhar. A general two-dimensional hidden markov model and its application in image classification. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 6, pages VI –41 –VI –44, 16 2007-oct. 19 2007. doi: 10.1109/ICIP.2007.4379516.

[9] S. Mitra and T. Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C:*

Applications and Reviews, IEEE Transactions on, 37(3):311 –324, may 2007. ISSN 1094-6977. doi: 10.1109/TSMCC.2007.893280.

[10] Chung-lin Huang and Sheng-Hung Jeng. A model-based hand gesture recognition system. *Mach. Vision Appl.*, 12:243–258, May 2001. ISSN 0932-8092. doi: 10.1007/s001380050144. URL `http://dl.acm.org/citation.cfm?id=375397.375409`.

[11] Hyeon-Kyu Lee and J.H. Kim. An hmm-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(10):961 –973, oct 1999. ISSN 0162-8828. doi: 10.1109/34.799904.

[12] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, 21(8):745 – 758, 2003. ISSN 0262-8856. doi: 10.1016/S0262-8856(03)00070-2. URL `http://www.sciencedirect.com/science/article/pii/S0262885603000702`.

[13] S. Marcel, O. Bernier, J.-E. Viallet, and D. Collobert. Hand gesture recognition using input-output hidden markov models. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 456 –461, 2000. doi: 10.1109/AFGR.2000.840674.

[14] Gerhard Rigoll, Andreas Kosmala, and Stefan Eickeler. High performance real-time gesture recognition using hidden markov models. In *In Proc. Gesture Workshop*, pages 69–80. Springer, 1998.

[15] Stefano Pellegrini and Luca Iocchi. Human posture tracking and classification through stereo vision. In *in Proc. of Intern. Conf. on Computer Vision Theory and Applicartions (VISAPP*, 2006.

[16] C. Ott, Dongheui Lee, and Y. Nakamura. Motion capture based human motion recognition and imitation by direct marker control. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 399 –405, dec. 2008. doi: 10.1109/ICHR.2008.4755984.

[17] Thad E. Starner and Alex Pentland. Visual recognition of american sign language using hidden markov models, 1995.

[18] Hye Sun Park, Eun Yi Kim, Sang Su Jang, Se Hyun Park, Min Ho Park, and Hang Joon Kim. *HMM-Based Gesture Recognition for Robot Control*. 2005. doi: 10.1007/11492429_73.

[19] Andrea Kleinsmith, P. Ravindra De Silva, and Nadia Bianchi-Berthouze. Cross-cultural differences in recognizing affect from body posture. *Interact. Comput.*, 18:1371–1389, December 2006. ISSN 0953-5438. doi: 10.1016/j.intcom.2006.04.003. URL `http://dl.acm.org/citation.cfm?id=1221613.1222203`.

[20] A Kleinsmith, N Bianchi-Berthouze, and A Steed. Automatic recognition of non-acted affective postures. *IEEE transactions on systems man and cybernetics Part B Cybernetics a publication of the IEEE Systems Man and Cybernetics Society*, 41(4):1–12, 2011. URL http://discovery.ucl.ac.uk/735772/.

[21] George Caridakis, Kostas Karpouzis, Athanasios Drosopoulos, and Stefanos Kollias. Somm: Self organizing markov map for gesture recognition. *Pattern Recognition Letters*, 31(1):52 – 59, 2010. ISSN 0167-8655. doi: 10.1016/j.patrec.2009.09.009. URL http://www.sciencedirect.com/science/article/pii/S0167865509002372.

[22] Cemil Oz and Ming C. Leu. American sign language word recognition with a sensory glove using artificial neural networks. *Engineering Applications of Artificial Intelligence*, 24(7):1204 – 1213, 2011. ISSN 0952-1976. doi: 10.1016/j.engappai.2011.06.015. URL http://www.sciencedirect.com/science/article/pii/S0952197611001230. ¡ce:title¿Infrastructures and Tools for Multiagent Systems¡/ce:title¿.

[23] Takashi Morie, Hiroyuki Miyamoto, and Akitoshi Hanazawa. Brain-inspired visual processing for robust gesture recognition. *International Congress Series*, 1301(0):31 – 34, 2007. ISSN 0531-5131. doi: 10.1016/j.ics.2006.12.010. URL http://www.sciencedirect.com/science/article/pii/S0531513106006479. ¡ce:title¿Brain-Inspired IT III. Invited and selected papers of the 3rd International Conference on Brain-Inspired Information Technology BrainIT 2006 held in Hibikino, Kitakyushu, Japan between 27 and 29 September 2006¡/ce:title¿.

[24] Abhishek Kar. Skeletal tracking using microsoft kinect the microsoft kinect sensor. *Methodology*, pages 1–11, 2010. URL http://home.iitk.ac.in/~akar/cs397/Skeletal%20Tracking%20Using%20Microsoft%20Kinect.pdf.

[25] Andrew Wilson, Aaron F. Bobick, and Justine Cassell. Recovering the temporal structure of natural gesture. In *In Proc. 2nd Int. Conf. on Automatic Face and Gesture Recognition*, pages 66–71, 1996.

[26] Naoufel Werghi and Yijun Xiao. Recognition of human body posture from a cloud of 3d data points using wavelet transform coefficients. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, FGR '02, pages 77–, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1602-5. URL http://dl.acm.org/citation.cfm?id=874061.875439.

[27] Ahmed Elgammal, Vinay Shet, Yaser Yacoob, and Larry S. Davis. Gesture recognition using a probabilistic framework for. In *Pose Matching, The Seventh International Conference on Control, Automation, Robotics and Vision, ICARCV 2002, Singapore in*, pages 2–5.

[28] Ying Wu, Thomas S. Huang, and N. Mathews. Vision-based gesture recognition: A review. In *Lecture Notes in Computer Science*, pages 103–115. Springer.

[29] Rómer Rosales and Stan Sclaroff. Learning and synthesizing human body motion and posture. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, FG '00, pages 506–, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0580-5. URL `http://dl.acm.org/citation.cfm?id=795661.796179`.

[30] Bernard BOULAY, Francois BREMOND, and Monique THONNAT. Applying 3d human model in a posture recognition system, November 2006.

[31] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1297 –1304, june 2011. doi: 10.1109/CVPR.2011.5995316.