University of Magdeburg

Faculty of Computer Science

Master Thesis

3-D Hand Gesture Recognition with Different Temporal Behaviors using HMM and Kinect

Author:

Suryakiran Maruvada

Feb 16, 2017

Advisors:

Dr. Claudia Krull

Department of Simulation and Graphics

Tim Dittmar

Maruvada, Suryakiran:
3-D Hand Gesture Recognition with Different Temporal Behaviors using HMM and Kinect
Master Thesis, University of Magdeburg, 2017.

Abstract

Vision based hand gesture recognition is getting increasingly popular due to its intuitive and effective interaction between man and machines. However, there are not sufficient means of support for deployment, research and execution for these tasks. In this thesis, we present 3-D hand gesture recognition system to recognize, especially when dealing with gestures of similar shape but different temporal pattern using Hidden Markov Models (HMMs). This is facilitated by recording gestures using Microsoft Kinect sensor. Recorded gestures include different shapes but with varying temporal patterns. We train the Hidden Markov Model (HMM) models based on feature vectors extracted from the recorded gestures. Training and classification process was implemented using machine learning framework Accord.Net. We analyze the feasibility of HMMs in detecting and recognizing different gestures in the classification process. We implemented five-fold cross validation strategy to determine accuracy rates of different gestures for different training parameters. We also analyzed the results of classification process for gestures drawn by specific users. Our results suggests that our algorithm can achieve an average best of 63.6% in recognizing gestures with different temporal patterns and an average best of 86.8% in recognizing gestures drawn by a specific user. This shows that within limits, HMMs can be applied to distinguishing user gestures with similar shape and different temporal patterns.

Keywords: 3-D gesture recognition, Hidden Markov Models, Kinect, Accord.Net

Acknowledgements

Firstly, I wish to express my sincere gratitude to my main advisor, Dr. Claudia Krull, for providing me this research opportunity. Thank you for your motivation, patience, and support. Without your mentoring and supervision, this thesis would have been impossible. I would like to thank my second advisor, Tim Dittmar, for all the motivational and resourceful discussions. Special thanks to my best friend Dushyant Priyadarshee, for his immense support and inspiration. I sincerely thank all the people who participated as volunteers in this thesis. Lastly, all the credit to my family for their constant motivation and support.

Declaration

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning. I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the law enforcement agency.

Magdeburg, 16.02.2017

Contents

List of Figures

List of Tables

List of Acronyms

APIs	Application Programming Interfaces
AR	Augmented Reality
CHMM	Continuous Hidden Markov Model
DHMM	Discrete Hidden Markov Model
DTW	Dynamic Time Warping
FD	Fourier Descriptors
GUI	Graphical User Interface
HMM	Hidden Markov Model
HMMs	Hidden Markov Models
HRI	Human Robot Interaction
LRB	Left-Right Banded
SDK	Software Development Kit
SQL	Structured Query Language

SVM	Support Vector Machines

1. Introduction

1.1 Motivation

Hand gesture recognition research in 3D space has been gaining importance widely all over the world. Apart from ordinary uses in day-to-day life, gesture recognition is slowly gaining entry into automotive, medical systems, communication systems, gaming, education, mobile devices, virtual reality, sign language etc. Previous works on gesture recognition can be divided into two parts glove based and vision based approaches. Glove based approach is where a user has to wear a glove attached with sensors and cables to detect specific movement of hand. The output of the sensors are then given as an input to a computer for further processing. Evidently, glove based approach is inconvenient and bulky to carry. However, vision based approach is gaining momentum because of the reason that an user doesn't require to carry bulky devices but instead the gestures can be performed in a much more natural environment. These movements by the user are then recorded using depth cameras.

As discussed previously, vision based approach needs cameras to record human movements. We can use any ordinary web camera to record video sequences and then apply gesture recognition techniques. However, an ordinary camera can only work on a 2D space. To work in 3D space we would need two ordinary web cameras and complex computer vision algorithms. With recent developments in camera technologies we have precise body motion sensing input devices like Microsoft Kinect and Intel Real Sense. Microsoft Kinect incorporates a RGB color camera, depth sensor and a microphone for voice recognition. Microsoft Kinect has been one of the most widely used camera for gesture recognition[ISTC14, SFS+14] because of the fact that it allows users to act freely as they would in real-life. Its low-cost and depth sensor capable of capturing videos in real time 3D under any ambient lighting made Microsoft Kinect an obvious choice for our experiments.

Focusing on vision based methods, captured video data using Microsoft Kinect are used for interaction and hand gestures are one of the most common methods of human interaction. Gestures can be anything as simple as a hand wave or drawing complex shapes. For better understanding, hand gestures are classified into different categories: 1) static or dynamic and 2) isolated or continuous

There has been lot of papers for recognizing simple static and dynamic gestures. For instance, a simple static gestures are like calculating angle between thumb and index finger and simple dynamic gestures are like drawing a circle, right swipe, left swipe etc. Again, gestures can be further classified into two parts, isolated and continuous gestures. For an instance, drawing a number 32 as a gesture would be considered as a continuous gesture because there exist a connecting vector from the number three to the number two which must be discarded, to be nominated as a meaningful gesture. Whereas, drawing a circle or a fast circle would be considered as an isolated gesture. Here in this paper we focus only on isolated gestures.

The above mentioned examples share a common governing principle. Machine learning and computer vision algorithms act as the driving force in gesture recognition techniques. Some of the most commonly used algorithms in gesture recognition are Hidden Markov Models (HMMs), Dynamic Time Warping (DTW), Support Vector Machines (SVM). HMMs using Kinect has been one of the most widely researched combination [PSF, XL12, ISTC14]. HMMs are statistical method which creates models based on provided parameters from a stochastic process emerging through time. A probabilistic likelihood of any other sequence being generated by the model can be calculated using a trained model. HMMs are discussed in detail under Section 2.2.

However, there has been limited or not much research to define different temporal aspects of different gestures using HMMs. In other words, is it possible for HMMs to recognize a gesture with different velocities evolving with time? A research question to answer would be to test how efficient is HMMs in recognizing specific gestures with different temporal behaviors and evaluate its success rate.

1.2 Goal of the Thesis

The goal of this thesis is to evaluate how feasible it is to use HMMs to perform gesture recognition after recording gestures with different temporal patterns using Microsoft Kinect skeletal tracking. HMMs will be used as training and classification algorithm for the recorded gestures. We will measure the success rate of HMMs for each gesture. Some research questions to answer would be as follows:

- ⇒ Are HMMs capable of recognizing gestures with different temporal patterns?
- ⇒ How will training parameters affect the recognition results?

1.3 Project Description

Contributions made in this thesis are focused on defining and recognizing gestures with different temporal patterns for a defined gesture set using HMMs and Kinect. HMMs has been the most widely used method for modeling gesture and speech recognitions. Precisely, gesture detection and recognition is computer vision and pattern recognition problem. We propose a method to define and recognize gesture patterns performed with different velocities in real time by training HMMs. Our main motivation is, can HMMs be able to differentiate between a set of gestures with varying velocities?

To achieve the goals as described in the Section 1.2 specific tasks were performed. Before recording the gestures, Microsoft Kinect was setup for skeletal tracking of 20 different joints of the user. Specifically, the right hand joint was tracked for right hand coordinates. Using the Kinect, 10 different types of gestures were performed by five different users (including two females) which then was subsequently used to build a gesture set. Local database was used to store the hand coordinates along with unique shapeID and userID of gestures. The database structure is discussed in detail under Section 4.2.

Subsequently, the recorded hand coordinates were converted into feature vectors for training HMMs as described in the Section 3.4. For training the HMMs, a machine learning framework Accord.Net was used. The framework allows user to use the built-in libraries and functions for building HMMs.

To explore the feasibility of HMMs, experiments were designed and conducted around the idea of recognizing gestures with different temporal aspects and how are they affected with the change in training parameters. For classification of gestures, two cross-validation experiments were carried out as discussed in Section 4.2. Firstly, by dividing the gesture set into 80% as training data and 20% as testing data. Average accuracy rate was then calculated for the whole testing data. Secondly, the whole gesture set was again divided into 90% and 10% and the average accuracy rate for the testing data was calculated. To learn how the training parameters affect the training and classification performance of HMMs, cross-validation experiments were conducted with different number of states and topologies as described in Section 4.2.

HMMs are known to perform well with certain gestures like drawing shapes, numbers or swiping [Hu14, YSBS01]. Velocity as a training parameter would change drastically, depending on how fast a gesture is performed. Is it possible to recognize two similar gestures but with different speeds? For example, drawing a circle in air and another circle with different velocity are considered to be two different gestures. But, is it possible for HMMs to recognize them as two different gestures? Keeping in mind that both these gestures have the same training parameters. So, how feasible are HMMs to recognize different temporal patterns of gestures with same training parameters would be a research question to explore.

1.4 Structure of the Thesis

Before representing the effort in this thesis, Chapter 2 gives an overview of previous and related works on hand gesture recognitions using HMMs. This elucidation includes discussion on hand gesture recognition problems, types of gestures, definition of HMMs, types of HMMs, limitations of using HMMs and skeletal tracking using Microsoft Kinect. Chapter 3 describes implementation of algorithm, training process overview of HMMs for recognizing gestures, technology setup for implementation, definition of gestures and gesture set, feature extraction of hand, training of HMMs using feature vectors and classification. Chapter 4 presents the experiment setup, database structure, algorithms, experiment results and their evaluation. Chapter 5 summarizes our thesis. It includes discussion about evaluation of goals, limitations in the thesis and benefits including future work extensions.

2. Related Work

This chapter gives an overview of concepts that go through the whole thesis. In this section we discuss about how a basic hand gesture recognition system works and state of the art research papers involving HMMs. Gestures have been the most preferred way of interaction for humans. We talk about how gestures are classified into different categories. Based on several research papers, HMMs are one of the most popular training and classification algorithms for hand gesture recognition where time is not a constraint. Here we discuss about how HMMs are defined and classified depending on probability distributions and its underlying markov processes. A brief overview of how HMMs are used in hand gesture recognition and their applications. Even though HMMs are the most preferred algorithms for speech and gesture recognition, it itself has lots of limitations. In the end, we discuss about how skeletal tracking works in Microsoft Kinect.

2.1 Hand Gesture Recognition System

Hand gestures are natural way of human communication. Basically, hand gesture recognition system can be modularized into 3 different sections as shown in Figure 2.1. Firstly, since the entry of Microsoft Kinect there has been number of proposed extraction methods for not only hand gestures but lot of other applications like foot gestures, face recognition, 3D tracking and so on. Segmentation and tracking a moving hand is a complex process. For example, [EAHM11] explains how they used 3D depth data and color information using stereo color images to segment and track moving hand. For robust segmentation of skin colored regions (Y, C_b, C_r) color space was used. A different kind of approach for extraction was used here in [KF11] where the authors have mapped a circular plate on the regions of users hand and the centroid of the circle using skin color for hand detection. Applying, previous hand detection method, hand tracking was done in fixed window search region. Boundary of the hand and centroid point of hand region was calculated. Iteratively, motion trajectory of the gesture path was obtained by collecting the hand centroid points.

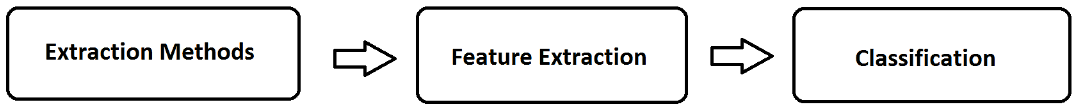


Figure 2.1: Gesture Recognition Model

Secondly, feature extraction is critical step in the system. Once the hand detection and tracking is done, we need features of the tracked hand for further computation. Good feature selection plays a significant role in performance of gesture recognition. In the same paper [EAHM11] chose to use basic features: location, orientation and velocity. They achieved an overall average accuracy rate of around 97% for their experiment. We may choose to describe the object using their features in frequency domain, instead of temporal or spatial domain. [Che03] used Fourier Descriptors (FD) and motion vectors as features for HMMs for their experiment.

Static gestures can be defined as orientation and shape of a hand in a spatial domain for a specific duration of time without any movement. These gestures don't change with time. For instance, joining forefinger and thumb for a "*ok*" pose. Whereas, if there is certain movement like waving a hand represents a dynamic gesture and is a time varying signal.

There has been work done on differentiating static and dynamic gestures automatically. For example, [RWA+07] presented a automatic recognition model using HMMs to differentiate static and dynamic gestures using position of hand and angle between the fingers. They used infrared tracking sensors on the fingers of the user to receive position and angle parameters. Static gesture like *pointing* do not vary with time. So, the angles between the fingers stay the same but the position might vary. For a dynamic gesture like *clapping* position and angles vary with time. These concluded gestures are then applied to compare execution times or interaction intuitiveness in real environment, a connected Augmented Reality (AR) application and gesture recognition system.

2.2 Definition of Hidden Markov Models

Hidden Markov Model (usually denoted by λ) is a probabilistic model which carry dual layer stochastic processes. First layer is the first-order Markov process, represented by states, transition probability matrices and observations. As we see, HMMs is built on the foundation of Markov process, so it is bound to follow Markov property, which states, the conditional probability of the future states only depends on the present state and not on the states preceding it. Second layer is the set of output probabilities for each state. For example, [SFS+14] explains, there is a robot which follows simple instructions by an user performing 3 gestures one after another. They are: 1) Wave; 2) Stop; and 3) Come .

Markov process analysis for the above mentioned example can predict the probability of the next gesture being Come given that the current gesture is Stop. Mathematically, this Markov analysis can be modeled as shown in Equation 2.1

$$P(X_n|X_{n-1}, X_{n-2}, ...X_1) = P(X_n|X_{n-1}) \tag{2.1}$$

Equation 2.1 shows that the probability of an observation X_n at time n depends only on X_{n-1} at time $n-1$.

Above example, presents us with a model following Markov properties. A HMM is a mathematical structure which obeys Markov properties but in contrast to the presented example, the user can only see the sequence of observations and the states are hidden in this case. This complements the term "hidden" in the Markov process. Graphically, HMMs can be modeled as shown in Figure 2.2.

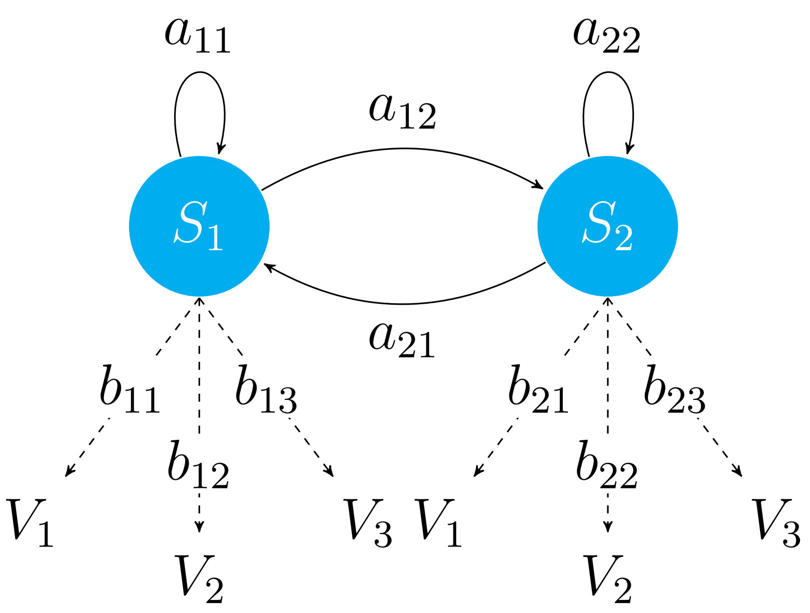


Figure 2.2: A Simple HMM Model with Two States

According to [Ern89], mathematically, a HMM can be expressed as shown in Equation 2.2

$$\lambda = (S, V, A, B, \Pi) \tag{2.2}$$

and can be described as follows:

- a sequence of observations $O = \{O_1, O_2, ..., O_T\}$, where time $t = 1, 2....., T$;
- S set of N hidden states $\{S_1, S_2, ..., S_N\}$;
- V set of M discrete observation symbols $\{V_1, V_2, ..., V_M\}$;
- transition matrix $A = \{a_{ij}\}$ where a_{ij} is the transition probability from state S_i at time t to state S_j at time $t + 1$;
- matrix $B = \{b_{jk}\}$ is the observation symbol probability, where b_{jk} is probability of generating symbol V_k from state S_j;
- Π is the initial probability vector of the states $\Pi = \pi_j, j = 1, 2, ..., N$, where $\pi_j = P(s_j)$ at $t = 0$.

For easy understanding, a compact mathematical notation $\lambda = (A, B, \Pi)$ can also be used which includes probabilistic parameters.

2.3 HMMs for Hand Gesture Recognition

Lots of research and development has been done on gesture recognition using human skeletal movement and depth cameras like Microsoft Kinect and Intel RealSense. Here are some reference articles to the state of the art algorithms and models for gesture recognition using HMMs.

Modeling a HMM has three basic problems: evaluation, decoding and training. Training a HMM requires feature vectors as an input for model parameter estimation. Baum-Welch algorithm is most widely used for solving training problem. [EAHM11] used combined features like location, orientation and velocity of the hand to train HMMs. The authors used HMMs to recognize alphabet characters $(A - Z)$ and numbers $(0 - 9)$ in real time. In another approach [FLO14] used the skeleton data $(X - Y - Z)$ of the hand and elbow from Kinect and calculated the angle between them in both $(X - Y)$ and $(Y - Z)$ plane. These 2 angles were used as the feature vector inputs for HMMs training. HMMs were used to detect and recognize 6 different gestures like follow me, wait and so on for a robotic task.

[FLO14] suggested Human Robot Interaction (HRI) for office tasks using HMMs and Kinect. They proposed HMMs based on skeletal tracking using X-Y-Z coordinates. Here they managed to convert skeleton data into symbol sequence and subsequently processing it through gesture recognition engine till result of gesture recognition. They used to 6 different types of gestures like Follow me, Move, Standby, Wait, Transport and Bye-Bye.

[EAHM08] proposed automatic system that recognizes isolated and continuous gestures for Arabic Numbers (0-9) in real time using different Hidden Markov models topologies. They used hand motion trajectory to determine the angle between each subsequent trajectory and then dividing the resultant angle into a code word. 18 different Code words are defined for different values of resultant angle which are used as a sequence for training HMM parameters.

2.4 Discrete and Continuous HMMs

According to [Ern89], HMMs based on the probability density continuity of state observations can be divided into 2 types: Continuous Hidden Markov Model (CHMM) and Discrete Hidden Markov Model (DHMM). In the DHMM, the raw input data is quantized into codewords using vector quantization for each observation. On the other hand, a CHMM models the observations using continuous probability density functions without vector quantization. As described here [Tan95], the observations of DHMM are discrete symbols that correspond to a vector quantized codebook. Figure 2.3 shows a HMM with two output symbols; likelihood of producing ”0” symbol is more likely than a ”1” for the state S_1 and vice-versa for the state S_2.

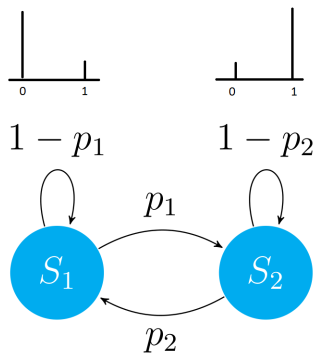


Figure 2.3: Discrete HMM

In CHMM, the probability of observations are represented as continuous data. Figure 2.4 shows a 2-state HMM where each state has a 1-dimensional probability density function; state S_1 is still more likely to produce symbol ”0” than a ”1”, but CHMM doesn't require any quantization for the observations. Here in this thesis we deal with CHMM since our density distribution is continuous.

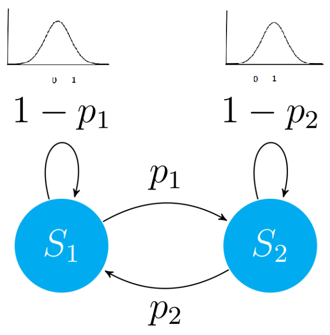


Figure 2.4: Continuous HMM

2.5 Limitations of Hidden Markov Models

HMMs work exceptionally well in temporal classification problems, provided we have a large training dataset. Training a large dataset is both expensive and time consuming. Main limitation of Continuous HMMs algorithm (detailed discussion in Section 2.4) is that the accuracy and speed of the classifier can vary greatly with the change in parameters like number of states, Gaussian distribution etc. Some of the most common implementation issues in HMMs are suitably explained here in [Cha16]. They explained that while using multiple observation sequences, the left-right model of HMM, a state transition occurs in a sequential manner. This limits re-estimation of model parameters when we have only single observation sequence because until a transition is made to next state, model only allows a limited number of observations for any state. That is why we need multiple sequences to generate sufficient data for model parameter estimation.

Choosing initial estimates of HMM parameters is a critical issue. Right choice of initial parameters estimation will lead to global maximum from local maximum of the likelihood function. In yet another revolutionary paper on HMMs, the author [Rab89] points out that in most of the cases a random estimate of π and A are suitable enough to generate parameter re-estimates. For B good estimation is important for continuous distributions.

Topology selection for a HMM model is again an implementation issue. Selection of different available topologies like ergodic, forward or some other will lead to different levels of accuracies and speeds. Choosing a different model size, number of states, classes or discrete and continuous observation symbols generate varying results. So, there is no direct or straight-forward way to achieve satisfactory results. The selection of parameters depends highly on the application being modeled.

2.6 Skeleton Tracking in Kinect

Gesture recognition using skeletal data from Kinect is a challenging task. For instance, a wave gesture detection using skeleton tracking can be easily done, by writing a small piece of code. However, skeleton tracking using Kinect is not fully reliable because sometimes it can lose track of joints when occluded by environmental issues like chairs or overlapping of joints. Defining a proper start and end of a gesture using skeletal tracking is challenging. [EAHM11] implemented automatic segmentation and tracking algorithm using skin color detection for defining start and end of gestures. Using six different joints returned by Kinect sensor [CATA13] concatenated all the 3D coordinates for six joints to generate feature vectors for a gesture sequence which would act as an input for DTW. Noisy frames at the starting and ending of gestures are inevitable. We have considered to train our HMM models because these noisy frames are just the natural way of human-computer interaction.

In the past, lot of techniques employing Kinect has been proposed for hand gesture recognition. Detection and segmentation of hand is a crucial part where segregation

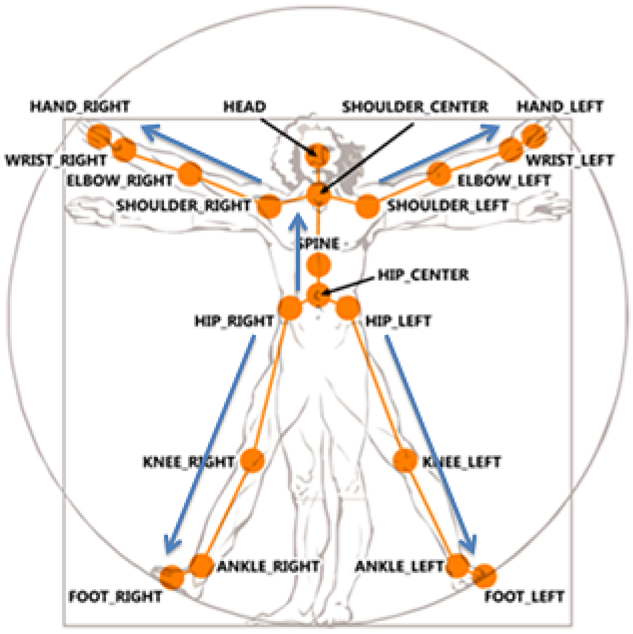


Figure 2.5: A Model Showing 20 Different Joints Tracked Using Kinect v1

is carried out to eliminate image background except for the hand to be tracked and recognized subsequently.

Using a depth sensor and a infrared sensor, Kinect has ability to track different body-joints as described here in Microsoft research [SFC+13]. Kinect sensor returns the raw depth data using depth sensor and then process each pixel to check for human skeleton.

Here in this paper [KSS] the authors have described how skeletal tracking works and what are the challenges faced in real-time pose recognition of human skeleton. In this thesis, we have used skeleton tracking using Kinect for Windows Software Development Kit (SDK). Windows SDK for Kinect provides us with a bunch of Application Programming Interfaces (APIs) which allow us to track specific joints in a human skeleton. Figure 2.5 shows 20 different joints that can be tracked using Kinect v1.

3. Implementation

This chapter represents our approach and tasks to achieve our defined goals. We discuss about our flow of execution, technology used, definition of gestures, extraction of features, training and classification process of gestures.

3.1 Training and Process Overview

Figure 3.1 represents the flow of execution for the recognition process. Video frames are captured using Kinect camera. These captured frames are then processed one by one for extraction of significant features of the hand like orientation of the hand and length of the vector. The extracted features act as an input for HMM training. The training algorithm (Baum-Welch) creates a HMM model for each gesture and calculates the probabilities for each HMM models using Forward algorithm. When a test observation is given for recognition it generates the maximum likelihood probability that matches the trained HMM model and provides us with an output.

3.2 Technology

Visual based gesture recognition requires a camera to capture and record video sequences. In this thesis, we used Microsoft Kinect to accomplish skeletal tracking of the users. For training HMM parameters with Baum-Welch algorithm, Accord.Net framework is used. Accord.Net framework is a .NET machine learning framework completely written in C#. The framework provides easy classification tools like HMM, SVM etc. Using the framework a Hidden Markov classifier was instantiated with initial parameters like number of states set to 2, number of gesture classes, forward topology and distribution density of feature vectors α, β and $\vec{v}$.

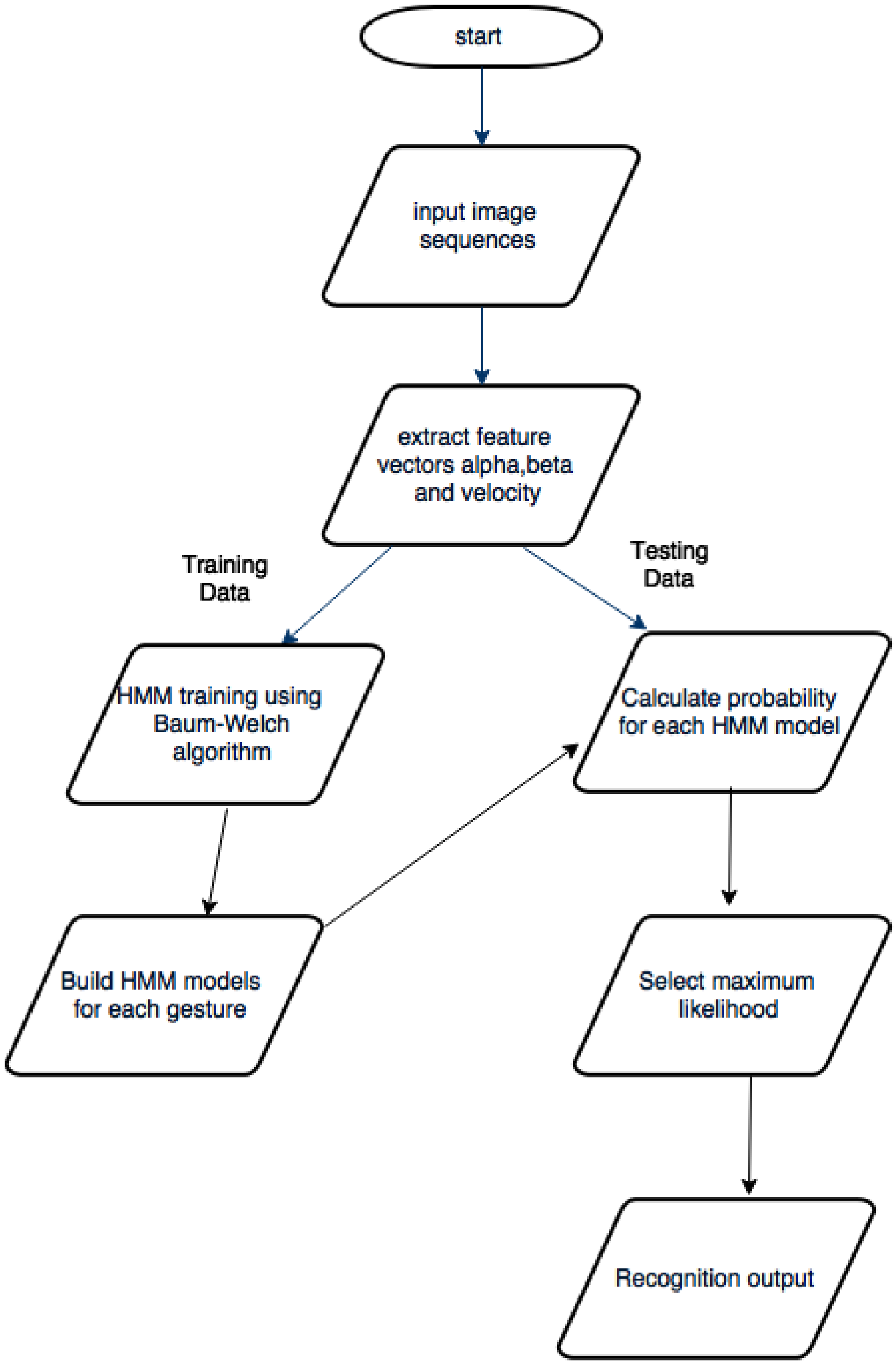

start
input image sequences
extract feature vectors alpha,beta and velocity
Training Data
Testing Data
HMM training using Baum-Welch algorithm
Calculate probability for each HMM model
Build HMM models for each gesture
Select maximum likelihood
Recognition output

Figure 3.1: Flow Chart for Recognition Model

3.3 Gesture Definition

Giving commands to your infotainment system in your car through only computer vision techniques without sounds or any other media is comparable to making gestures to your friend. In order to do a feasibility study involving HMMs, our system requires body language which varies with time. There are many kinds of gestures which can be represented by the hand motion as discussed here in Section 2.1. Our main goal is mainly to recognize the dynamic hand gesture with different temporal patterns from continuous hand motion in real-time, and implement on interaction between human and machine. In this thesis, we describe 10 types of gestures using right hand as shown in the Table 3.1. The first column in the Table 3.1 is the unique ShapeID assigned to each of the 10 gestures and is used for training & classification of HMMs. The gesture set was chosen so as to keep a mixture of both normal and gestures with different velocities. The gestures shown in Table 3.1 are named accordingly for easy understanding.

Shape ID	Gesture
0	Vertical Line
1	Horizontal Line
2	Circle
3	Square
4	Triangle
5	Circle_Fast
6	Circle_Slow_Fast
7	Circle_Fast_Slow
8	Triangle_Fast
9	Diagonal

Table 3.1: Defined Gesture Set

Figure 3.2 shows different gesture patterns we recorded for our database. The green circle represents the starting point of a gesture, single arrow represents normal speed and double arrows denote faster speed.

For instance, the gesture *Triangle_Fast* is represented as a triangle with double arrows on its sides, showing the gesture is performed faster than normal speed.

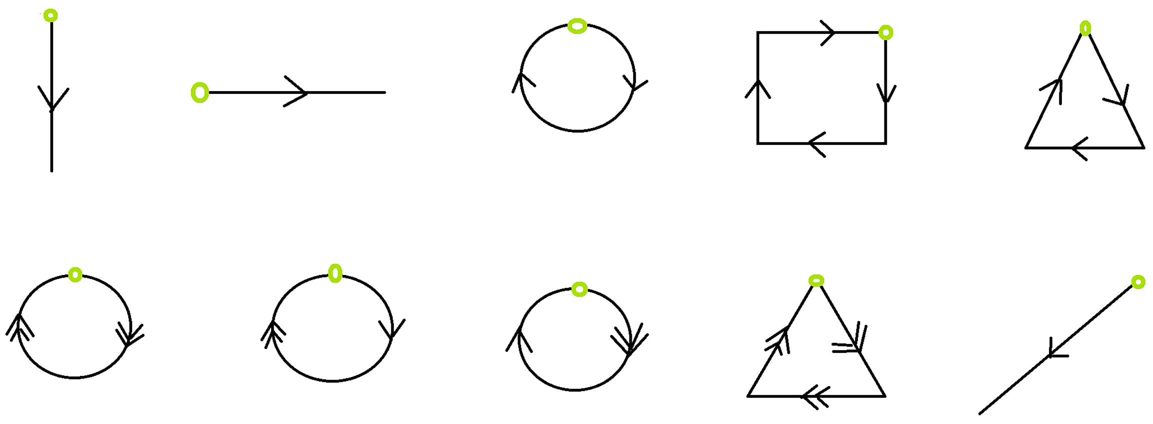


Figure 3.2: 10 Different Gestures with Varying Temporal Patterns

3.4 Feature Extraction

Good feature selection plays an important role in hand gesture recognition performance. These features are derived in cartesian coordinate space directly from the gesture frames. Skeleton tracking using Kinect SDK, provides us with x, y, z coordinates i.e. the position of the hand in 3D space. Initially, x, y, z coordinates were selected to be the features of the hand for HMM training. But the training performance was very poor to be taken into consideration. As shown in the Figure 3.3 the coordinate system of Kinect, x, y and z values are measured in the meters. For example, hand position $(0.3, -0.65, 1)$ would suggest the hand is $1m$ away in z-direction from Kinect sensor, $0.3m$ and $-0.65m$ in x-direction and y-direction respectively.

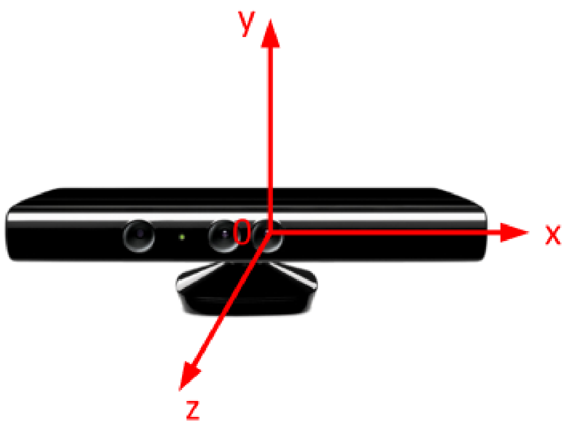


Figure 3.3: Kinect Coordinate System

Due to the reason, that the HMM training with Cartesian coordinates performed poorly, we opted to new feature vectors like orientation of hand in $X - Y$ and $Z - Y$ plane.

Previous research [EAHM08, LLKD] have shown that orientation as a feature has been the best for accuracy results. So, the orientation is calculated between 2 consecutive points in the gesture path. Firstly, the change in x, y and z was calculated as shown below.

Change in x,y,z coordinates:

$$\begin{gathered}\delta_x = x_{t+1} - x_t \\ \delta_y = y_{t+1} - y_t \\ \delta_z = z_{t+1} - z_t \\ t = 1, 2, ...T - 1\end{gathered} \tag{3.1}$$

Feature vector α is calculated as the angle between x and z axes in radians. α was calculated in radians in order to avoid problem of division by zero and the ambiguity if the angle in degrees is returned as 0° or 360°. $\arctan 2(x, z)$ will return a valid answer as long as x is not zero.

$$\alpha = \arctan 2(\delta_x, \delta_z) \tag{3.2}$$

Angle β is the angle (in radians) between y-axis and the length of the vector $\vec{u}$ in xz-direction.

$$\vec{u} = \sqrt{\delta_x^2 + \delta_z^2} \tag{3.3}$$

$$\beta = \arctan 2(\delta_y, \vec{u}) \tag{3.4}$$

α and β were used to train HMMs but the results were considered to be too noisy. Reason for this noisy training probabilities is because of different temporal behaviors with same patterns. For example, drawing a circle in the air will have almost similar α, β angles as drawing another circle with different speed. Since we are dealing with gestures with different speeds, we incorporated another feature vector $\vec{v}$ into our training. Velocity as a feature was calculated as shown below:

$$\vec{v} = \sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2} \tag{3.5}$$

In this thesis, feature vectors α, β and $\vec{v}$ were finally considered as training parameters for HMMs.

3.5 Training

Training HMMs for a large set of database can be very complex. To train HMM parameters, each gesture is manifested with a corresponding model. Feasibility of HMMs might vary with different set of model parameters, so here we have tried using different model parameters accordingly to achieve maximum likelihood for given training data. There is no analytic solution to this problem so far. However, re-estimation of model parameters can be achieved using Baum-Welch algorithm. This algorithm iterates over the model parameters to achieve the local maximum for the given training data.

Design of topology in HMMs includes a set of states $S_1, ..., S_N$ and how they are connected to each other. For example, a forward topology represents that the state transitions only occur in forward direction; in other words, it is also known as Left-Right Banded (LRB) topology. Research[FLO14, EAHM08] have suggested LRB is the best suited topology for hand gesture recognition. As shown in the Figure 3.4, 4 connected states could make a transition to the state itself or its subsequent state only.

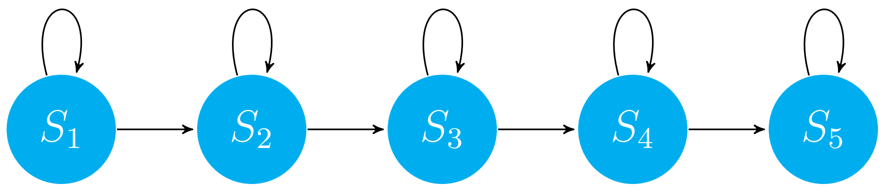


Figure 3.4: Forward Topology Representation

In order to test out the feasibility of HMMs, a fully-connected topology(ergodic) and forward topology was used. A fully-connected or ergodic topology is a set of connected states which allows non-constrained transition to any state starting from a specific state.

Figure 3.5 shows 4 different connected states allowing transition to any other state or to the state itself.

To better understand the behavior of HMMs, training was conducted with different parameters for each experiment. Parameters considered for the experiments are number of states and topology of the model. Using the framework Accord.Net, two different classifiers were used to train the HMMs. First classifier was used to train only to classify different shapes; it was trained with four different states and two topologies and the outputs were plotted as graphs in Section 4.4. Likewise, a second classifier was trained using same methods, but this classifier was used classify gestures drawn by a specific user. In other words, recognizing how many times a gesture drawn by a specific user is recognized correctly. The experiment details and results are discussed in the Section 4.5

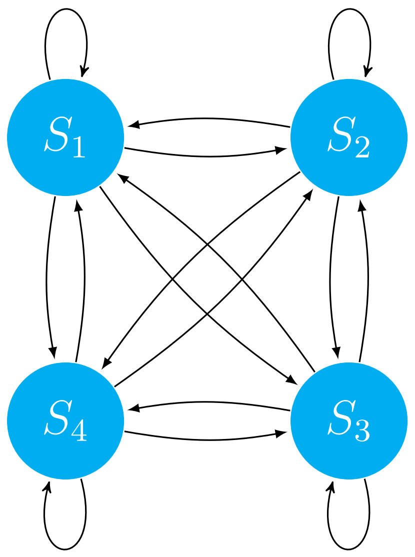


Figure 3.5: Ergodic Topology Representation

3.6 Classification Process

Gesture recognition using video sequences can be considered as a pattern recognition problem. While training the HMM is an elaborate process, recognition process is much simpler. For classification, forward algorithm is used. Given an observation, all the HMMs generate probabilities but the highest probability generating model best describes the observation sequence. The HMM model with the highest probability is chosen as the likely generator of the observation, resulting in a label. Accord.Net provides necessary functions for probability calculation and classification of gestures.

4. Experiments and Evaluation

To understand the behavior of HMMs, two different experiments were performed. This section describes about gesture recording using Graphical User Interface (GUI), database structure and interpretation of the test results. For all the following experiments, cross-validation strategy was used. These test results include interpretation of how effective are HMMs in recognizing gestures and in recognizing specific gesture drawn by a specific user. We discuss about average accuracy rates and confusion matrices.

4.1 Graphical User Interface

The experiments are conducted on a Windows 10 machine powered by i7 2.6 GHz processor. The whole code is written in C# using Windows Forms and Microsoft Visual Studio 2015. For better interaction, a GUI was designed to assist users. Initially, for recording gestures, we used 3-second delay rule to mark start of gestures. To start recording gestures, the user has to keep the right hand static for 3 seconds and then the GUI prompts the user *recording started*. Once the recording has been started, the user starts drawing a gesture until the end, where the user has to keep the right hand static for a brief moment, till the GUI prompts *recording stopped*. Algorithm was designed in such a way that, when the recording is underway it checks for standard deviation in right hand coordinates for the last 60 frames. This standard deviation is calculated for each and every frame until it hits a certain threshold, thereby, marking the end of gesture.

Recording 10 different types of gestures was a challenge since the users have to follow the 3-second rule to initiate recording. It was a cumbersome task. Later, different choices of action were made available to user through the GUI interface, namely:

1. Recording gestures using buttons and subsequently storing them into the database.

The GUI was developed in C# for better interaction with the Kinect. Kinect starts tracking the skeleton the moment it detects a user and is displayed on the screen. When the GUI starts, it displays options for the user as shown in the Figure 4.1 below:

- Start: Start recording gesture.
- Stop: Stop recording gesture.
- UserID: User can select specific user name from the drop down list.
- GestureID: Option to select which gesture is being recorded.
- Create Markov Classifier: A button for training the hidden markov classifier.

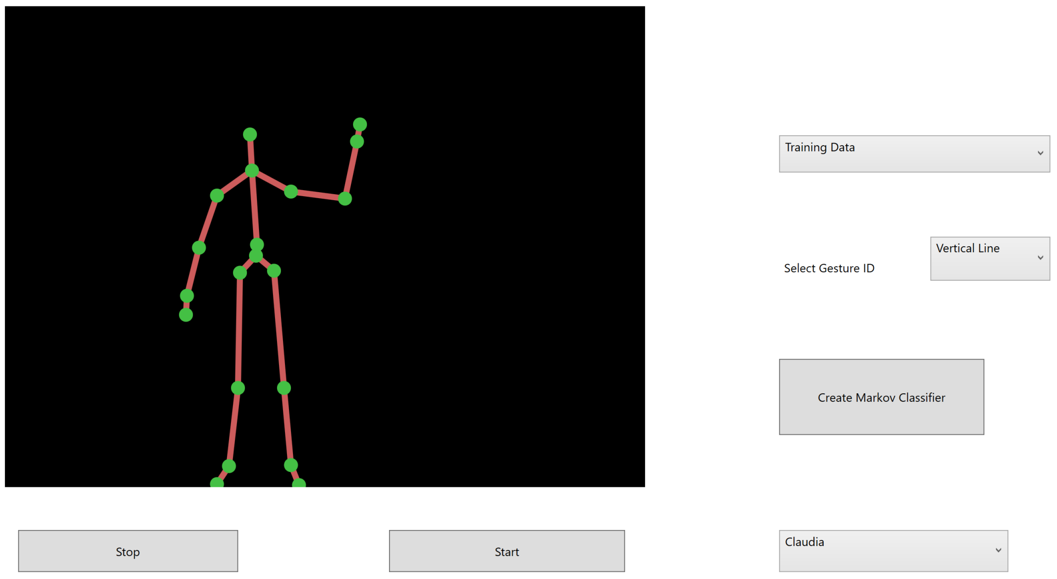
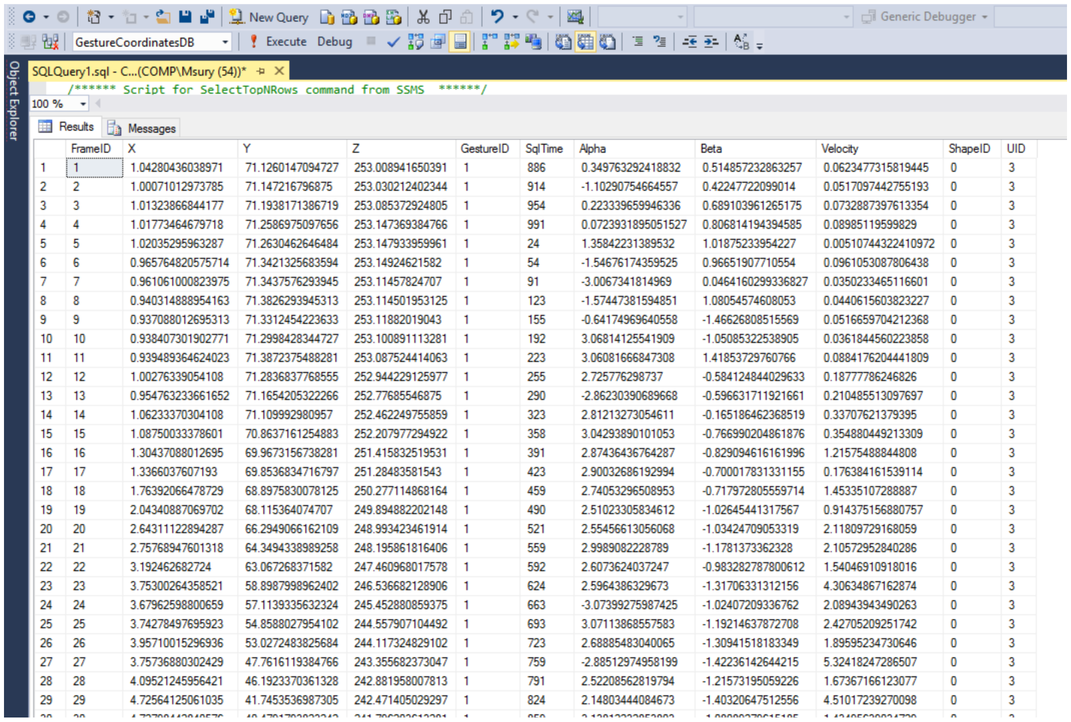


Figure 4.1: Graphical User Interface

4.2 Database

For the purpose of training and testing of data, a local Structured Query Language (SQL) server 2016 database was setup. The resolution of recorded sequences was 640 x 480 at 30 frames per second. We have recorded 10 different shapes with varying temporal patterns. 5 different volunteers including 2 females have recorded 20 sequences each for a single gesture. 100 sequences each for a gesture was recorded with varying number of frames in each sequence. This is to keep randomness in the experiment. All the performed gestures were recorded with clock-wise orientation. In total, we recorded 63,636 frames for 1000 sequences in our database. The volunteers performed all the



```
/****** Script for SelectTopNRows command from SSMS  ******/
```

	FrameID	X	Y	Z	GestureID	SqlTime	Alpha	Beta	Velocity	ShapeID	UID
1	1	1.04280436038971	71.1260147094727	253.008941650391	1	886	0.349763292418832	0.514857232863257	0.0623477315819445	0	3
2	2	1.00071012973785	71.147216796875	253.030212402344	1	914	-1.10290754664557	0.42247722099014	0.0517097442755193	0	3
3	3	1.01323866844177	71.1938171386719	253.085372924805	1	954	0.223339659946336	0.689103961265175	0.0732887397613354	0	3
4	4	1.01773464679718	71.2586975097656	253.147369384766	1	991	0.0723931895051527	0.806814194394585	0.08985119599829	0	3
5	5	1.02035295963287	71.2630462646484	253.147933959961	1	24	1.35842231389532	1.01875233954227	0.00510744322410972	0	3
6	6	0.965764820575714	71.3421325683594	253.14924621582	1	54	-1.54676174359525	0.96651907710554	0.0961053087806438	0	3
7	7	0.961061000823975	71.3437576293945	253.11457824707	1	91	-3.0067341814969	0.0464160299336827	0.0350233465116601	0	3
8	8	0.940314888954163	71.3826293945313	253.114501953125	1	123	-1.57447381594851	1.08054574608053	0.0440615603823227	0	3
9	9	0.937088012695313	71.3312454223633	253.11882019043	1	155	-0.64174969640558	-1.46626808515569	0.0516659704212368	0	3
10	10	0.938407301902771	71.2998428344727	253.100891113281	1	192	3.06814125541909	-1.05085322538905	0.0361844560223858	0	3
11	11	0.939489364624023	71.3872375488281	253.087524414063	1	223	3.06081666847308	1.41853729760766	0.0884176204441809	0	3
12	12	1.00276339054108	71.2836837768555	252.944229125977	1	255	2.725776298737	-0.584124844029633	0.18777786246826	0	3
13	13	0.954763233661652	71.1654205322266	252.77685546875	1	290	-2.86230390689668	-0.596631711921661	0.210485513097697	0	3
14	14	1.06233370304108	71.109992980957	252.462249755859	1	323	2.81213273054611	-0.165186462368519	0.33707621379395	0	3
15	15	1.08750033378601	70.8637161254883	252.207977294922	1	358	3.04293890101053	-0.766990204861876	0.354880449213309	0	3
16	16	1.30437088012695	69.9673156738281	251.415832519531	1	391	2.87436436764287	-0.829094616161996	1.21575488844808	0	3
17	17	1.3366037607193	69.8536834716797	251.28483581543	1	423	2.90032686192994	-0.700017831331155	0.176384161539114	0	3
18	18	1.76392066478729	68.8975830078125	250.277114868164	1	459	2.74053296508953	-0.717972805559714	1.45335107288887	0	3
19	19	2.04340887069702	68.115364074707	249.894882202148	1	490	2.51023305834612	-1.02645441317567	0.914375156880757	0	3
20	20	2.64311122894287	66.2949066162109	248.993423461914	1	521	2.55456613056068	-1.03424709053319	2.11809729168059	0	3
21	21	2.75768947601318	64.3494338989258	248.195861816406	1	559	2.9989082228789	-1.1781373362328	2.10572952840286	0	3
22	22	3.192462682724	63.067268371582	247.460968017578	1	592	2.6073624037247	-0.983282787800612	1.54046910918016	0	3
23	23	3.75300264358521	58.8987998962402	246.536682128906	1	624	2.5964386329673	-1.31706331312156	4.30634867162874	0	3
24	24	3.67962598800659	57.1139335632324	245.452880859375	1	663	-3.07399275987425	-1.02407209336762	2.08943943490263	0	3
25	25	3.74278497695923	54.8588027954102	244.557907104492	1	693	3.07113868557583	-1.19214637872708	2.42705209251742	0	3
26	26	3.95710015296936	53.0272483825684	244.117324829102	1	723	2.68885483040065	-1.30941518183349	1.89595234730646	0	3
27	27	3.75736880302429	47.7616119384766	243.355682373047	1	759	-2.88512974958199	-1.42236142644215	5.32418247286507	0	3
28	28	4.09521245956421	46.1923370361328	242.881958007813	1	791	2.52208562819794	-1.21573195059226	1.67367166123077	0	3
29	29	4.72564125061035	41.7453536987305	242.47140502929 7	1	824	2.14803444084673	-1.40320647512556	4.51017239270098	0	3

Figure 4.2: Database Structure

gestures randomly without any training on gestures. All the recorded gestures were performed approx. 2.5-3 meters from the Kinect.

Our database is represented as shown in the Figure 4.2. We have 11 different attributes in the database, namely:

1. FrameID is the count of frames.
2. x, y and z are the coordinates of the right hand.
3. GestureID is numbered corresponding to the number of sequence.
4. SqlTime is used to calculate time difference between 2 frames and is measured in milliseconds.
5. *alpha*, *beta* and *velocity* are calculated using x, y and z as described here in Section 3.4.
6. ShapeID is unique labels for gestures.
7. UID is unique labels for users.

FrameID as discussed above has a count of 63,636 frames, GestureID runs from $1-1000$, ShapeID is labeled as $0-9$ for 10 different gestures and UID is labeled as $1-5$ for five different users.

4.3 Cross Validation

Cross validation, a model validation technique to assess results of a statistical analysis. It is mostly used to determine how well or accurately a predictive model performs in an experiment. Our five-fold cross validation experiment was split into 2 processes based on data set division: 1) 80% and 20% and 2) 90% and 10%. As we know, a better trained HMM model would apparently perform better than a less trained one. To figure out how does an extra 10% of training, impact the performance results of a HMM model dealing with gestures of different temporal patterns, was the reason why we split cross validation into 2 different processes.

First process involves, 80% training data and the rest 20% as testing data. This 80% of the data was used for training with different training parameters. The data was trained by varying the number of states and topologies. For instance, data was trained using 2-states with forward topology and the output results were recorded. Likewise, training was performed using 3,5 and 8 states with forward topology. Similarly, the process was repeated with ergodic topology.

In the second process, data was divided into 90% training data and 10% testing data. The whole process described above was iterated again.

Here in this thesis, for easy understanding, the processes were named as $Forward8020$, $Ergodic8020$, $Forward9010$ and $Ergodic9010$. $Forward8020$ here means the data is trained with forward topology but with 80% training data.

4.4 Gesture Recognition Experiment

This experiment was designed to test out the average accuracy rates of gestures. To train the HMMs, five-fold cross validation strategy was used. To perform the experiment, a hidden markov learning classifier was instantiated which accepts different parameters as discussed in Section 3.5. Since our experiment is a supervised learning process, unique ShapeID for each gesture was used as our output class labels. For the first iteration, training parameters were set as number of states-2, $Forward8020$ topology and 10 output classes(10 gestures). After the training is done, the remaining 20% testing data was given for classification. Our five-fold cross validation results in five different output percentages. The percentage on the graph shown in Figure 4.3 are the average of all five results for each process i.e. $Forward8020$. For subsequent iterations, training was conducted with states three,five and eight. This method was re-iterated for all the processes $Ergodic8020$, $Forward9010$ and $Ergodic9010$.

Consequently, as we know the HMMs get better with training, the classification results on the chart Figure 4.3 was expected to be better with 90% training data as opposed to 80% training data. As expected, our system performed the best with $Forward9010$ and eight-states. Although the previous research shows, 63.6% is not the best HMMs can achieve but it shows that HMMs are able to train and recognize gestures based on temporal patterns.

We know the learning algorithm creates a HMM for each gesture. In order to study the confusion of recognition rates between gestures with different temporal models, performed with different training parameters, we plotted a confusion matrix as shown in the Figure 4.4.

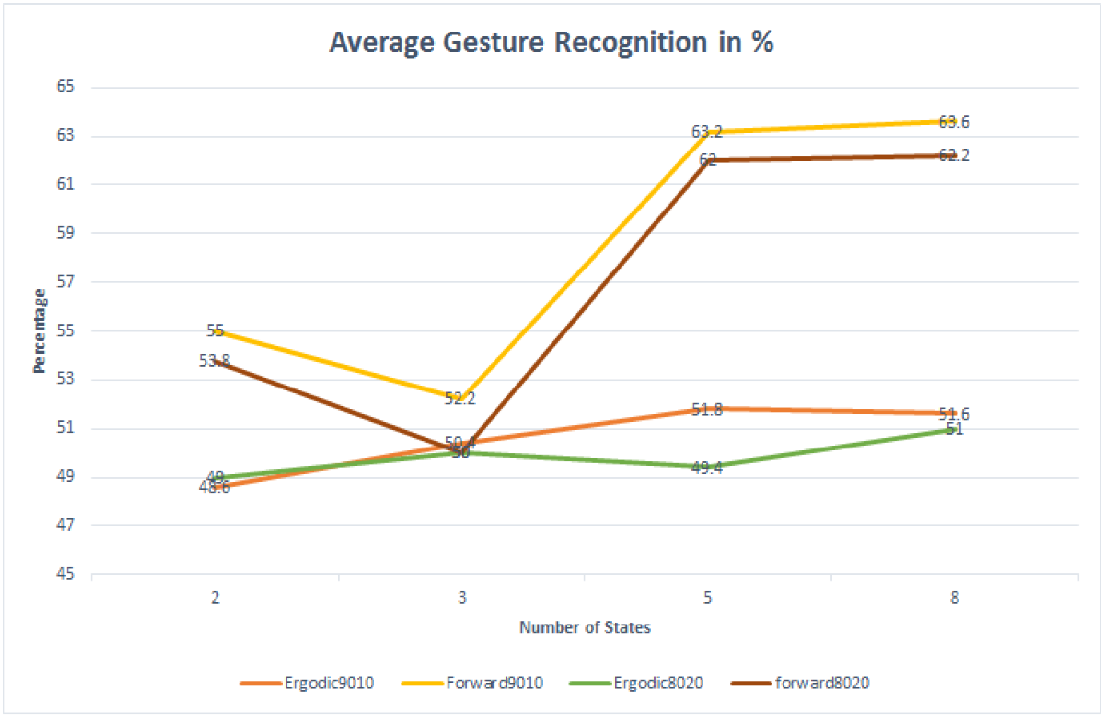
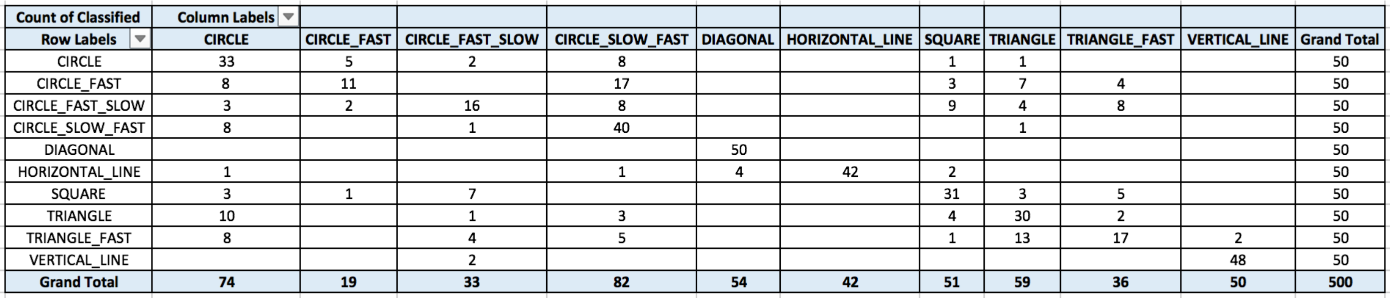


Figure 4.3: Average Gesture Accuracy Rate

The rows and columns in the matrix represent executed gestures and classified gestures respectively. The last column is the number of observations for each gesture. Conceivably, similar gesture models like *CIRCLE* and *CIRCLE_SLOW_FAST* are mostly confused with other gesture models. As we can see in the first row, gesture *CIRCLE* was executed 50 times but it was correctly classified as a *CIRCLE* only 33 times and was falsely classified as *CIRCLE_SLOW_FAST* 8 times. It is evident that the classifier was confused because our *CIRCLE* gesture has three more similar models but with different temporal patterns.

We can interpret that *CIRCLE_FAST* and *CIRCLE_FAST_SLOW* are the worst performing models with accurate classification of only 11 and 16 times respectively. Moreover, our best performing gestures are *DIAGONAL* closely followed by *VERTICAL_LINE*.



Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	33	5	2	8			1	1			50
CIRCLE_FAST	8	11		17			3	7	4		50
CIRCLE_FAST_SLOW	3	2	16	8			9	4	8		50
CIRCLE_SLOW_FAST	8		1	40				1			50
DIAGONAL					50						50
HORIZONTAL_LINE	1			1	4	42	2				50
SQUARE	3	1	7				31	3	5		50
TRIANGLE	10		1	3			4	30	2		50
TRIANGLE_FAST	8		4	5			1	13	17	2	50
VERTICAL_LINE			2							48	50
Grand Total	**74**	**19**	**33**	**82**	**54**	**42**	**51**	**59**	**36**	**50**	**500**

Figure 4.4: Forward9010 with Eight States

4.5 User-Gesture Recognition Experiment

Our database contains data for 10 different gestures drawn by five different users. To train a classifier for this experiment, we require a 1-dimensional vector consisting of class labels. To classify a gesture is drawn by a specific user, we need to train our classifier with both ShapeID and UserID as our output class labels. To achieve this, our output labels were calculated as shown in Equation 4.1:

$$(UserID) * 10 + ShapeID \tag{4.1}$$

Using the Equation 4.1, the range of our output labels would vary from 10-60. For the first iteration, parameters were set as number of states-2, $Forward8020$ topology and 60 output classes. After training, for accurate classification, the output labels has to be compared with the trained labels. As shown in the Table 4.1, the output label is 27 since it is the amalgamation of two different labels.

Output	Probability
27	0.99999999962688

Table 4.1: Raw Classification Output

To obtain the real output values for gesture and the user, we need to break it down. In order to do so, we used division and modulo operators. Whenever we divide the output 27 by 10 and consider only the integer part we get a 2, which represents UserID and modulo of 27 i.e. 27%10 gives an output of 7, which represents ShapeID. After this operation, we obtain result as shown in Table 4.2.

UserID	ShapeID	Probability
2	7	0.99999999962688

Table 4.2: Refined Classification Output

Subsequently, training and classification process was performed using *Ergodic8020*, *Forward9010* and *Ergodic9010* with different states. Figure 4.5 shows the average user-gesture accuracy rates for various states and topologies. Clearly, we can see *Ergodic9010* with three-states has the best average recognition percentage at 86.8%. On the contrary to the Section 4.4, ergodic topology has performed significantly better than forward topology for this experiment. One of the main reasons for this might be the additional output label attribute UserID for training the HMM.

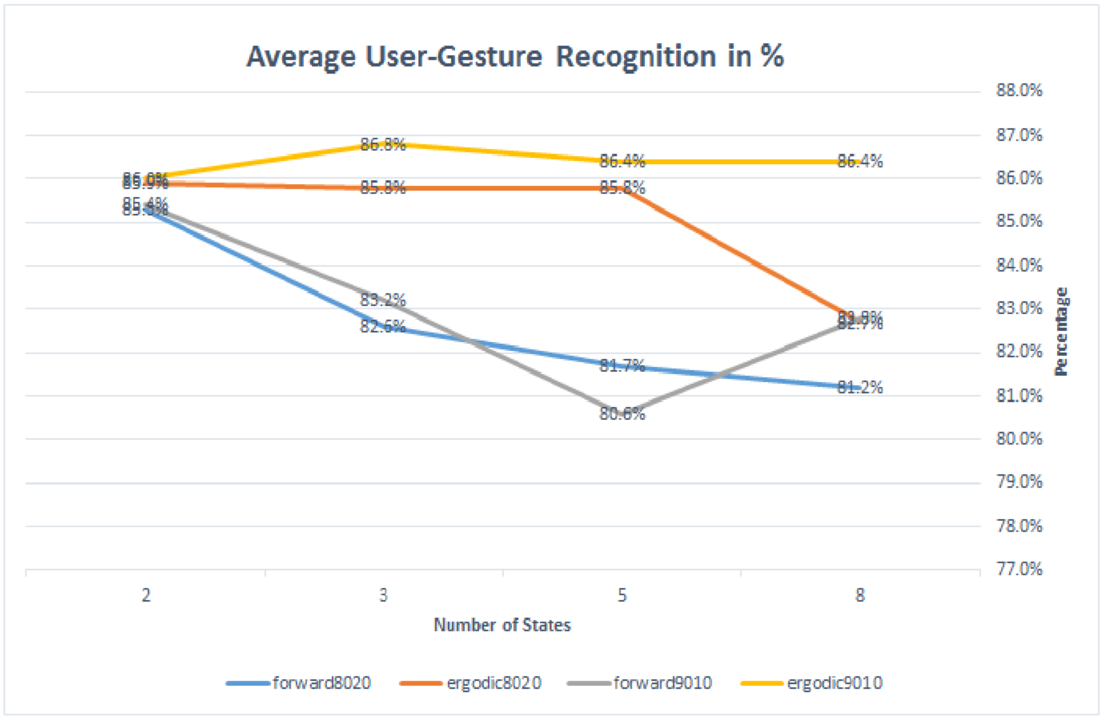


Figure 4.5: Average User-Gesture Accuracy Rate

Summarizing, to recognize gestures with varying temporal behaviors, we performed gesture recognition experiment with forward topology and eight-states and achieved an average success rate of 63.6%. For the user-gesture experiment, an average best of 86.8% was achieved with ergodic topology and three states.

5. Conclusion and Future Work

This chapter concludes our thesis with summary of the thesis on how 3-D hand gesture recognition using HMMs works, what are some of the limitations in our thesis which are yet to be solved, assessment of our predefined goals and finally benefits of our new concept and some potential extensions to our thesis which may help achieve better results.

5.1 Summary of the Thesis

Our thesis has been designed and implemented to recognize 3-D hand gestures with different temporal patterns by using HMMs and Kinect. Initially, our experimentation was simple, performance was observed to be poor even for a moderately sophisticated tasks, and resulted in insufficient and inaccurate models. Our experiments show how training parameters of HMMs contribute to its success.

HMMs are statistical based models, best recognition does not guarantee best models. Oftentimes, HMMs might be the best in training and recognition tasks, however, these HMMs fail badly on different testing data consisting of simple gestures but can recognize irrational patterns in real time. In order to maintain the evenness in the models, our system requires large data set which is however very expensive to collect and label. Primarily, motivated by HMMs efficiency in hand gesture recognition, the approach of this thesis has explored behavior of HMMs dealing with gestures with different temporal patterns. Through two different experiments with 3-dimensional hand gestures, this thesis analyzed the behavior of HMM training with various parameters and expressed their results with graphs and figures.

In the course of this thesis, we used skeletal tracking feature of Kinect to record isolated gestures. We studied how HMMs behave with different parameters. First, training is a very expensive procedure, selection of good feature vector is critical for both training and classification performance of HMMs. Five-fold cross validation strategy was used to perform training in two different processes with varying parameters. For gesture recognition experiment, training was conducted to learn and build models of gestures using only 10 classes(10 gestures). For testing, forward algorithm was used to classify test data by comparing it with the trained models. Test results were considered to be satisfactory. *Forward9010* with 8 states was considered the best with an average accuracy of 63.6%. Confusion matrix was analyzed to study how well the HMMs have performed for each gesture model.

User-gesture experiment was conducted to learn the performance of HMMs to recognize if a gesture is drawn by a specific user. Training for user-gesture recognition was conducted with 60 classes with varying parameters. The test results showed promising improvement in classification. A best score of 86.8% was achieved in *Ergodic9010* with 3 states. This demonstrates that with wise selection of feature vectors, right topology and good training parameters HMMs can achieve improved accuracy in gesture recognition even with different temporal behaviors.

5.2 Evaluation of Goals

First goal in our thesis, to study the feasibility of HMMs for recognizing gestures with different temporal behaviors. HMMs were trained using feature vectors with different parameters such as number of states and topologies. Accuracy results for gesture experiment and user-gesture experiment suggest that HMM for gestures with varying temporal patterns and user-gesture recognition are satisfactory and good respectively. However, the feasibility study on HMMs concludes that it can handle gestures dealing with varying temporal behaviors but with limited success.

Secondly, we analyzed how training parameters of HMMs resulted in different accuracy rates for both the experiments. Results established in the experiments prove that the performance of HMMs dealing with gestures of varying temporal patterns can still achieve satisfactory results if the training parameters are chosen carefully.

5.3 Limitations in the Thesis

HMMs are temporal based models. These models have been successfully used in speech and gesture recognitions. In our thesis, behavior and outputs of HMMs was studied for models with different temporal patterns. Even though we achieved a satisfactory result, there are some factor which can be considered as limitations in our algorithm. Defining a proper start and end of gesture is a demanding task. To save both time and effort we introduced a GUI to assist users for starting and stopping of recordings. This is done manually and requires an additional user to do so. While recording, illumination and complex backgrounds were not considered which might effect have an effect on skeletal tracking.

Skeletal tracking feature in Microsoft Kinect is a very useful tool to capture and store hand coordinates. Sometimes, skeletal tracking might lead to noisy or incomplete data.

For example, human hand motion is not as fast as frame capture rate of Kinect so it leads to some noisy frames in the database at the start and end of a gesture where the user hand is still and this is the case for any gesture. Here, we didn't consider removing them. Other limitation in skeletal tracking is, there might be some occasions where the tracked joint may overlap with other joints or is not tracked because of occlusions in the recording environment leading to incomplete data.

HMMs require huge training datasets to perform better. It limits proper experiment procedures because collecting and training large amounts of data is expensive and time consuming. Our database size might be still too small to properly evaluate HMMs behaviors.

5.4 Benefits and Future Work

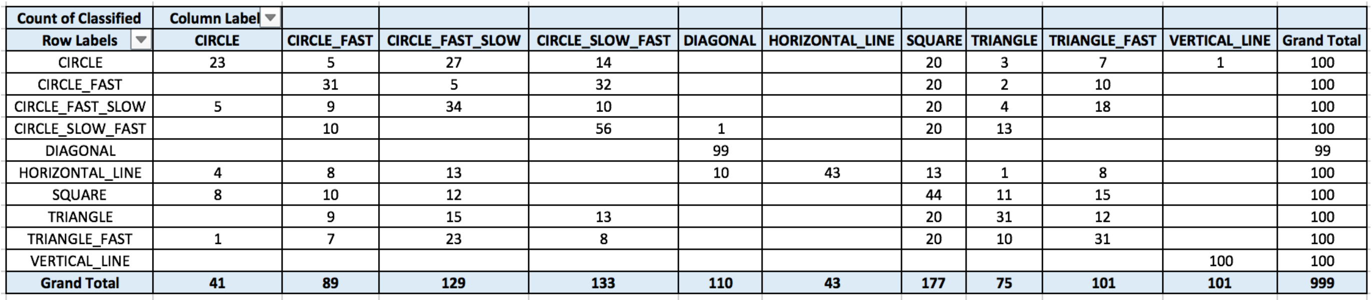
With any research, obvious enhancements and extensions are apparent. First, in the domain of 3-D hand gesture recognition, more experiments should be carried out. Particularly, more robust and better features needs to be extracted, such as orientation and relative coordinates of the hand instead of absolute coordinates. Accuracy results will improve with better features. Perhaps, better selection of shapes with temporal behaviors might lead to better convergence. Our HMM framework can be extended to two-stage HMMs. First-stage to recognize gestures and second-stage for task recognition as explained here [NdtLK12].

Recording gestures without noisy or incomplete data was still a limitation in our project. Maybe, better data collection with minimal noisy or incomplete frames will yield better accuracy results. Focusing on data, our experimental setup can be extended to more complex and bigger size, to allow more practical applications of the recognition system.

There are many use-cases for gesture enabled technology. We propose some use-cases which could be a potential application using gesture technology. New concept of classifying isolated gestures based on their temporal behaviors can be used in high security applications. Biometric authentication is one such application where this concept can be used extensively. Security features using static hand gestures have already been implemented as explained here [FZF13]. For authentication, the user can record his own custom gesture involving temporal behavior. It could act as a two-step verification along with password authentication.

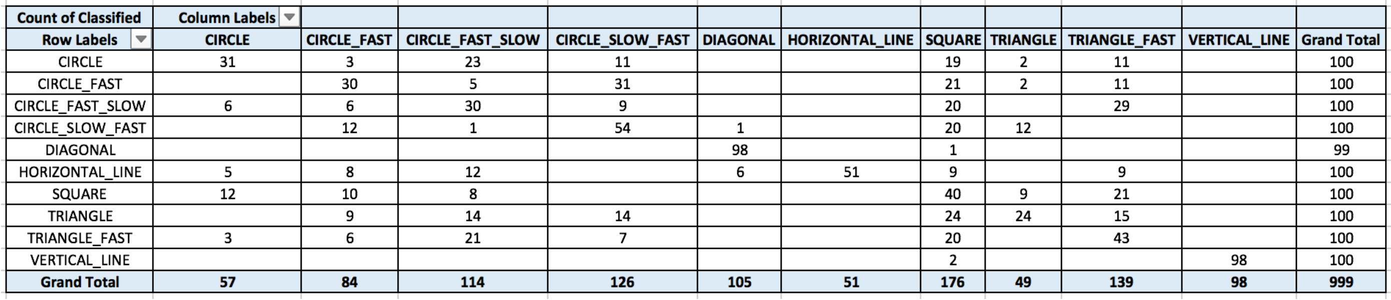
Gesture enabled technology is fast becoming popular with automotive innovations. Gesture based interfaces are successfully implemented in infotainment systems, car lock security and so on. In the world of connected cars, this concept could promote better road safety and driver safety by providing undivided interaction facilities and security to expensive assets.

A. Appendix



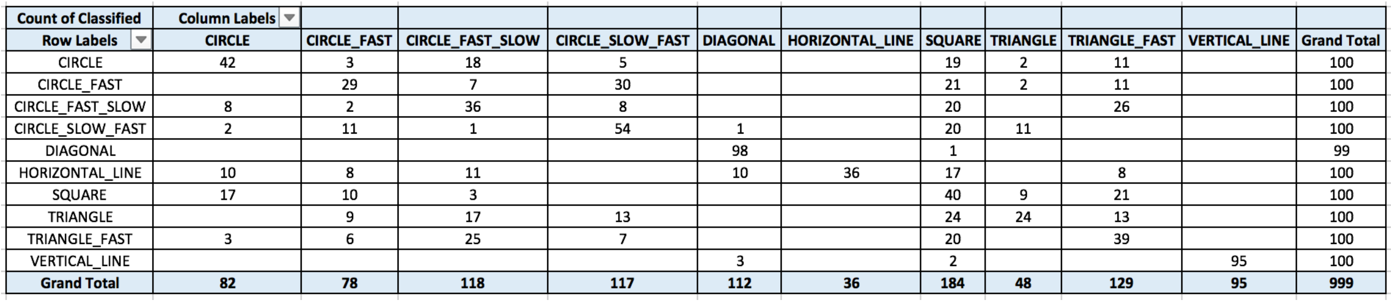
Count of Classified	Column Label										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	23	5	27	14			20	3	7	1	100
CIRCLE_FAST		31	5	32			20	2	10		100
CIRCLE_FAST_SLOW	5	9	34	10			20	4	18		100
CIRCLE_SLOW_FAST		10		56	1		20	13			100
DIAGONAL					99						99
HORIZONTAL_LINE	4	8	13		10	43	13	1	8		100
SQUARE	8	10	12				44	11	15		100
TRIANGLE		9	15	13			20	31	12		100
TRIANGLE_FAST	1	7	23	8			20	10	31		100
VERTICAL_LINE										100	100
Grand Total	**41**	**89**	**129**	**133**	**110**	**43**	**177**	**75**	**101**	**101**	**999**

Figure A.1: Ergodic8020 with Two States



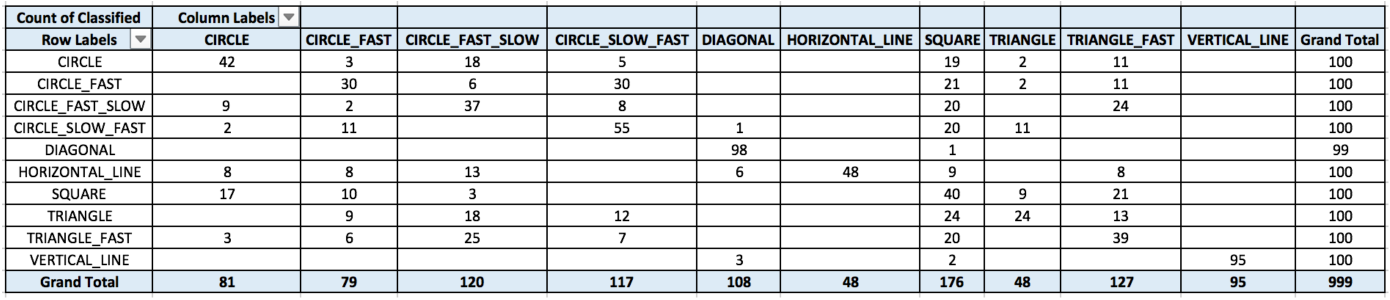
Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	31	3	23	11			19	2	11		100
CIRCLE_FAST		30	5	31			21	2	11		100
CIRCLE_FAST_SLOW	6	6	30	9			20		29		100
CIRCLE_SLOW_FAST		12	1	54	1		20	12			100
DIAGONAL					98		1				99
HORIZONTAL_LINE	5	8	12		6	51	9		9		100
SQUARE	12	10	8				40	9	21		100
TRIANGLE		9	14	14			24	24	15		100
TRIANGLE_FAST	3	6	21	7			20		43		100
VERTICAL_LINE							2			98	100
Grand Total	**57**	**84**	**114**	**126**	**105**	**51**	**176**	**49**	**139**	**98**	**999**

Figure A.2: Ergodic8020 with Three States



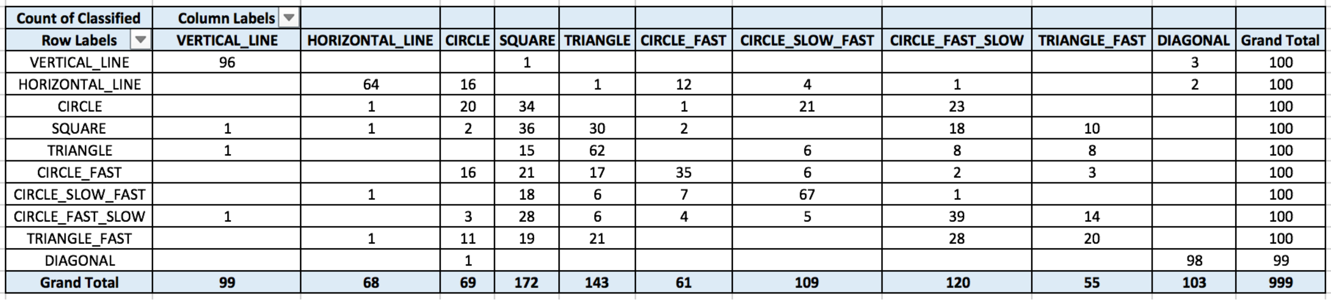
Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	42	3	18	5			19	2	11		100
CIRCLE_FAST		29	7	30			21	2	11		100
CIRCLE_FAST_SLOW	8	2	36	8			20		26		100
CIRCLE_SLOW_FAST	2	11	1	54	1		20	11			100
DIAGONAL					98		1				99
HORIZONTAL_LINE	10	8	11		10	36	17		8		100
SQUARE	17	10	3				40	9	21		100
TRIANGLE		9	17	13			24	24	13		100
TRIANGLE_FAST	3	6	25	7			20		39		100
VERTICAL_LINE					3		2			95	100
Grand Total	**82**	**78**	**118**	**117**	**112**	**36**	**184**	**48**	**129**	**95**	**999**

Figure A.3: Ergodic8020 with Five States



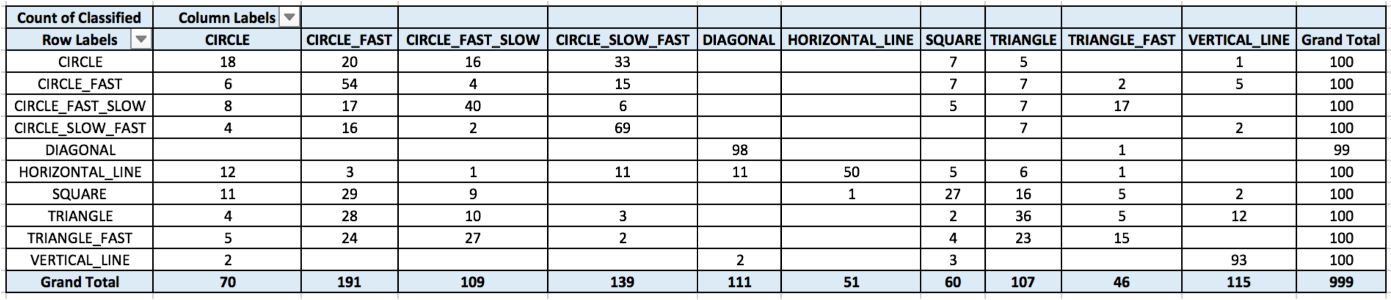
Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	42	3	18	5			19	2	11		100
CIRCLE_FAST		30	6	30			21	2	11		100
CIRCLE_FAST_SLOW	9	2	37	8			20		24		100
CIRCLE_SLOW_FAST	2	11		55	1		20	11			100
DIAGONAL					98		1				99
HORIZONTAL_LINE	8	8	13		6	48	9		8		100
SQUARE	17	10	3				40	9	21		100
TRIANGLE		9	18	12			24	24	13		100
TRIANGLE_FAST	3	6	25	7			20		39		100
VERTICAL_LINE					3		2			95	100
Grand Total	**81**	**79**	**120**	**117**	**108**	**48**	**176**	**48**	**127**	**95**	**999**

Figure A.4: Ergodic8020 with Eight States



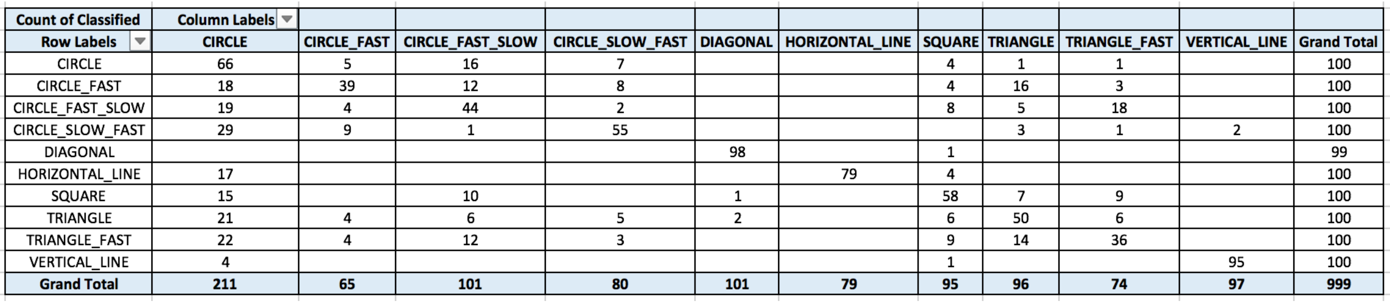
Count of Classified	Column Labels										
Row Labels	VERTICAL_LINE	HORIZONTAL_LINE	CIRCLE	SQUARE	TRIANGLE	CIRCLE_FAST	CIRCLE_SLOW_FAST	CIRCLE_FAST_SLOW	TRIANGLE_FAST	DIAGONAL	Grand Total
VERTICAL_LINE	96			1						3	100
HORIZONTAL_LINE		64	16		1	12	4	1		2	100
CIRCLE		1	20	34		1	21	23			100
SQUARE	1	1	2	36	30	2		18	10		100
TRIANGLE	1			15	62		6	8	8		100
CIRCLE_FAST			16	21	17	35	6	2	3		100
CIRCLE_SLOW_FAST		1		18	6	7	67	1			100
CIRCLE_FAST_SLOW	1		3	28	6	4	5	39	14		100
TRIANGLE_FAST		1	11	19	21			28	20		100
DIAGONAL			1							98	99
Grand Total	**99**	**68**	**69**	**172**	**143**	**61**	**109**	**120**	**55**	**103**	**999**

Figure A.5: Forward8020 with Two States



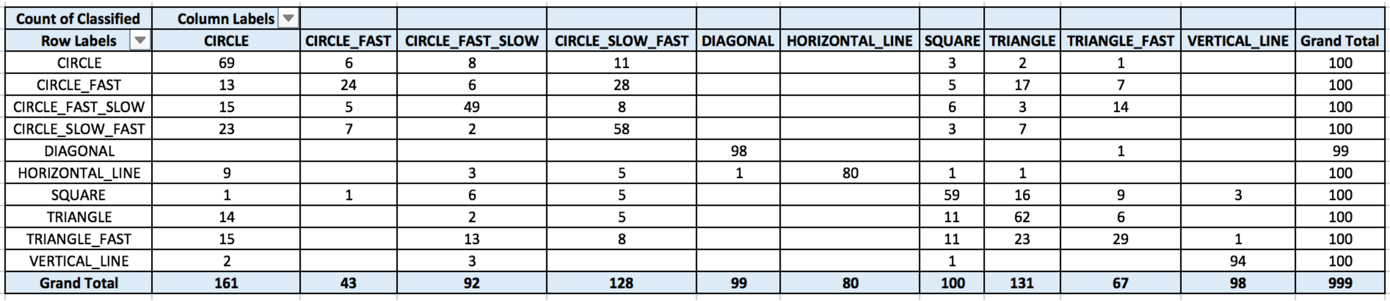
Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	18	20	16	33			7	5		1	100
CIRCLE_FAST	6	54	4	15			7	7	2	5	100
CIRCLE_FAST_SLOW	8	17	40	6			5	7	17		100
CIRCLE_SLOW_FAST	4	16	2	69				7		2	100
DIAGONAL					98				1		99
HORIZONTAL_LINE	12	3	1	11	11	50	5	6	1		100
SQUARE	11	29	9			1	27	16	5	2	100
TRIANGLE	4	28	10	3			2	36	5	12	100
TRIANGLE_FAST	5	24	27	2			4	23	15		100
VERTICAL_LINE	2				2		3			93	100
Grand Total	**70**	**191**	**109**	**139**	**111**	**51**	**60**	**107**	**46**	**115**	**999**

Figure A.6: Forward8020 with Three States



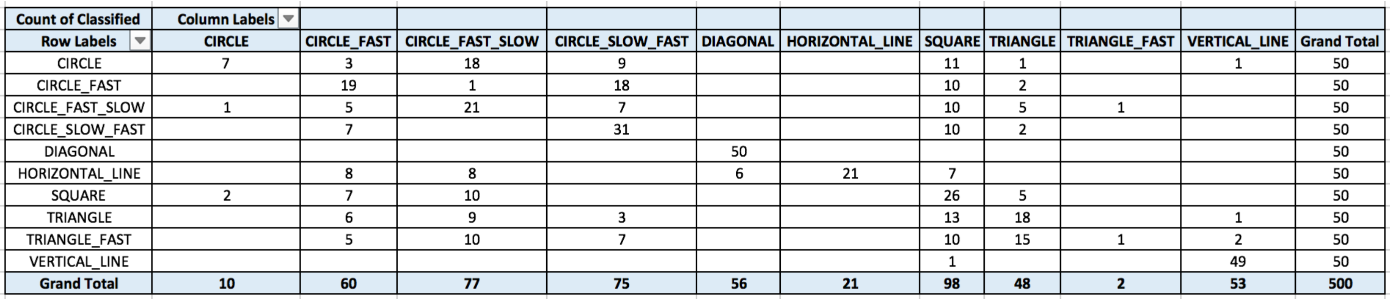
Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	66	5	16	7			4	1	1		100
CIRCLE_FAST	18	39	12	8			4	16	3		100
CIRCLE_FAST_SLOW	19	4	44	2			8	5	18		100
CIRCLE_SLOW_FAST	29	9	1	55				3	1	2	100
DIAGONAL					98		1				99
HORIZONTAL_LINE	17					79	4				100
SQUARE	15		10		1		58	7	9		100
TRIANGLE	21	4	6	5	2		6	50	6		100
TRIANGLE_FAST	22	4	12	3			9	14	36		100
VERTICAL_LINE	4						1			95	100
Grand Total	**211**	**65**	**101**	**80**	**101**	**79**	**95**	**96**	**74**	**97**	**999**

Figure A.7: Forward8020 with Five States



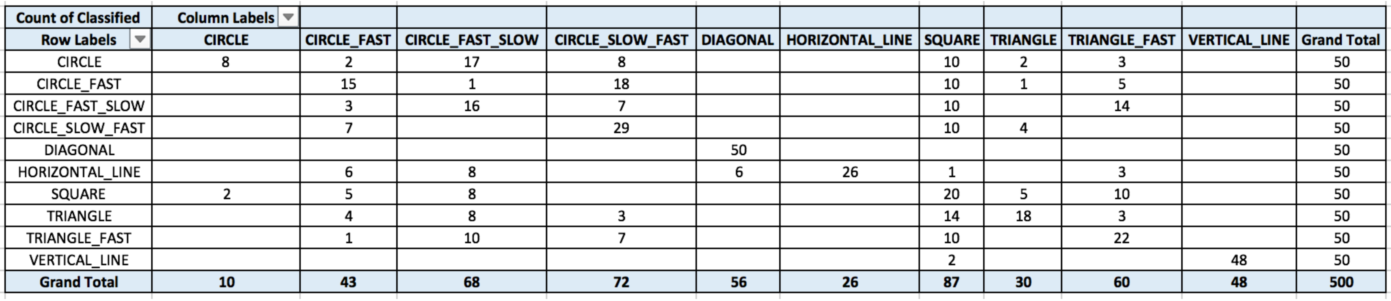
Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	69	6	8	11			3	2	1		100
CIRCLE_FAST	13	24	6	28			5	17	7		100
CIRCLE_FAST_SLOW	15	5	49	8			6	3	14		100
CIRCLE_SLOW_FAST	23	7	2	58			3	7			100
DIAGONAL					98				1		99
HORIZONTAL_LINE	9		3	5	1	80	1	1			100
SQUARE	1	1	6	5			59	16	9	3	100
TRIANGLE	14		2	5			11	62	6		100
TRIANGLE_FAST	15		13	8			11	23	29	1	100
VERTICAL_LINE	2		3				1			94	100
Grand Total	**161**	**43**	**92**	**128**	**99**	**80**	**100**	**131**	**67**	**98**	**999**

Figure A.8: Forward8020 with Eight States



Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	7	3	18	9			11	1		1	50
CIRCLE_FAST		19	1	18			10	2			50
CIRCLE_FAST_SLOW	1	5	21	7			10	5	1		50
CIRCLE_SLOW_FAST		7		31			10	2			50
DIAGONAL					50						50
HORIZONTAL_LINE		8	8		6	21	7				50
SQUARE	2	7	10				26	5			50
TRIANGLE		6	9	3			13	18		1	50
TRIANGLE_FAST		5	10	7			10	15	1	2	50
VERTICAL_LINE							1			49	50
Grand Total	**10**	**60**	**77**	**75**	**56**	**21**	**98**	**48**	**2**	**53**	**500**

Figure A.9: Ergodic9010 with Two States



Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	8	2	17	8			10	2	3		50
CIRCLE_FAST		15	1	18			10	1	5		50
CIRCLE_FAST_SLOW		3	16	7			10		14		50
CIRCLE_SLOW_FAST		7		29			10	4			50
DIAGONAL					50						50
HORIZONTAL_LINE		6	8		6	26	1		3		50
SQUARE	2	5	8				20	5	10		50
TRIANGLE		4	8	3			14	18	3		50
TRIANGLE_FAST		1	10	7			10		22		50
VERTICAL_LINE							2			48	50
Grand Total	**10**	**43**	**68**	**72**	**56**	**26**	**87**	**30**	**60**	**48**	**500**

Figure A.10: Ergodic9010 with Three States

Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	17	2	13	3			10	2	3		50
CIRCLE_FAST		15	1	18			10	1	5		50
CIRCLE_FAST_SLOW	1	1	19	6			10	1	12		50
CIRCLE_SLOW_FAST	2	6		29			10	3			50
DIAGONAL					50						50
HORIZONTAL_LINE	3	6	7		6	24	1		3		50
SQUARE	8	7	2				20	3	10		50
TRIANGLE		2	9	2			14	18	5		50
TRIANGLE_FAST		1	12	7			10	1	19		50
VERTICAL_LINE							2			48	50
Grand Total	**31**	**40**	**63**	**65**	**56**	**24**	**87**	**29**	**57**	**48**	**500**

Figure A.11: Ergodic9010 with Five States

Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	16	2	15	3			10	2	2		50
CIRCLE_FAST		15	1	18			10	1	5		50
CIRCLE_FAST_SLOW	1	1	21	6			10		11		50
CIRCLE_SLOW_FAST	2	6		29			10	3			50
DIAGONAL					50						50
HORIZONTAL_LINE	1	6	9		6	16	9		3		50
SQUARE	8	5	2				20	5	10		50
TRIANGLE		4	9	2			14	18	3		50
TRIANGLE_FAST		1	12	7			10		20		50
VERTICAL_LINE							2			48	50
Grand Total	**28**	**40**	**69**	**65**	**56**	**16**	**95**	**29**	**54**	**48**	**500**

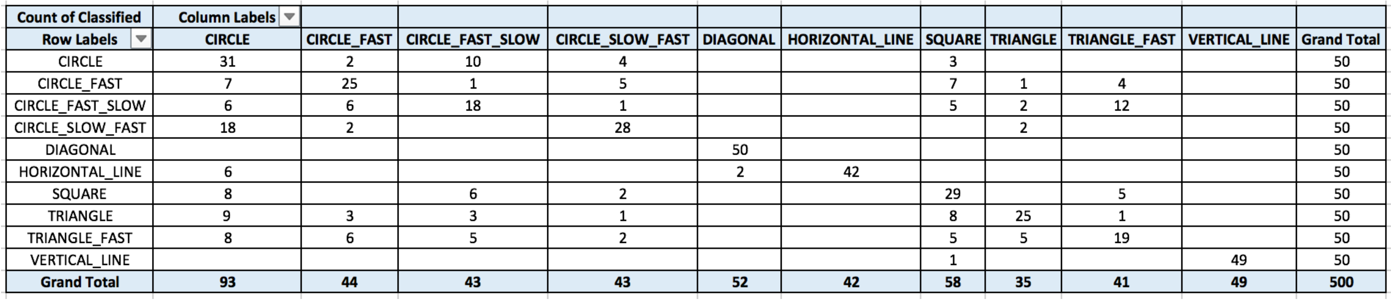
Figure A.12: Ergodic9010 with Eight States

Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	10	3	12	10		1	13	1			50
CIRCLE_FAST	5	16	1	6			9	11	2		50
CIRCLE_FAST_SLOW		3	20	4			13	5	5		50
CIRCLE_SLOW_FAST		2		36			8	4			50
DIAGONAL					50						50
HORIZONTAL_LINE	9	9	1		2	28		1			50
SQUARE	2	1	8				21	10	6	2	50
TRIANGLE			6				7	34	2	1	50
TRIANGLE_FAST	7		13				10	10	10		50
VERTICAL_LINE										50	50
Grand Total	**33**	**34**	**61**	**56**	**52**	**29**	**81**	**76**	**25**	**53**	**500**

Figure A.13: Forward9010 with Two States

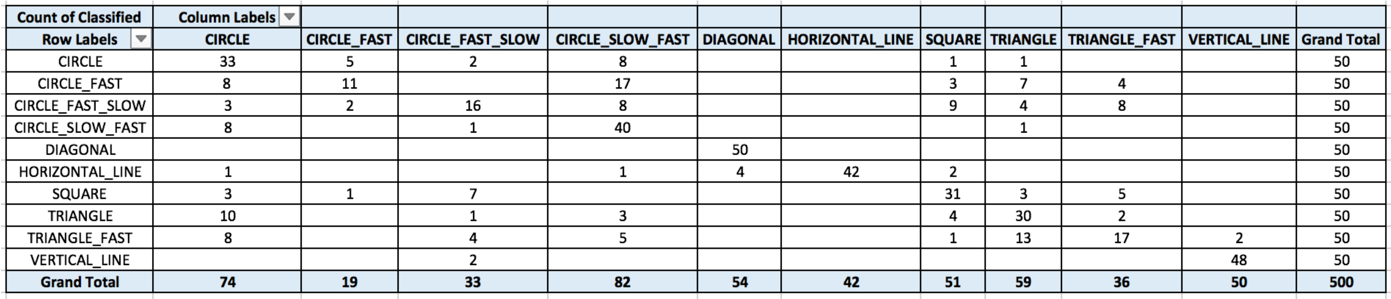
Count of Classified	Column Labels										
Row Labels	CIRCLE	CIRCLE_FAST	CIRCLE_FAST_SLOW	CIRCLE_SLOW_FAST	DIAGONAL	HORIZONTAL_LINE	SQUARE	TRIANGLE	TRIANGLE_FAST	VERTICAL_LINE	Grand Total
CIRCLE	18	6	5	17			4				50
CIRCLE_FAST	4	19	1	9			2	2	4	9	50
CIRCLE_FAST_SLOW	6	9	15	3			1	4	6	6	50
CIRCLE_SLOW_FAST	3	4		38						5	50
DIAGONAL					50						50
HORIZONTAL_LINE	4	2		8	8	26		1	1		50
SQUARE	6	9	3	2		1	16	6	5	2	50
TRIANGLE	2	8	5				2	22	2	9	50
TRIANGLE_FAST	3	12	8	1		2	1	11	11	1	50
VERTICAL_LINE	2						1		1	46	50
Grand Total	**48**	**69**	**37**	**78**	**58**	**29**	**27**	**46**	**30**	**78**	**500**

Figure A.14: Forward9010 with Three States



Count of Classified	Column Labels										
Row Labels	**CIRCLE**	**CIRCLE_FAST**	**CIRCLE_FAST_SLOW**	**CIRCLE_SLOW_FAST**	**DIAGONAL**	**HORIZONTAL_LINE**	**SQUARE**	**TRIANGLE**	**TRIANGLE_FAST**	**VERTICAL_LINE**	**Grand Total**
CIRCLE	31	2	10	4			3				50
CIRCLE_FAST	7	25	1	5			7	1	4		50
CIRCLE_FAST_SLOW	6	6	18	1			5	2	12		50
CIRCLE_SLOW_FAST	18	2		28				2			50
DIAGONAL					50						50
HORIZONTAL_LINE	6				2	42					50
SQUARE	8		6	2			29		5		50
TRIANGLE	9	3	3	1			8	25	1		50
TRIANGLE_FAST	8	6	5	2			5	5	19		50
VERTICAL_LINE							1			49	50
Grand Total	**93**	**44**	**43**	**43**	**52**	**42**	**58**	**35**	**41**	**49**	**500**

Figure A.15: Forward9010 with Five States



Count of Classified	Column Labels										
Row Labels	**CIRCLE**	**CIRCLE_FAST**	**CIRCLE_FAST_SLOW**	**CIRCLE_SLOW_FAST**	**DIAGONAL**	**HORIZONTAL_LINE**	**SQUARE**	**TRIANGLE**	**TRIANGLE_FAST**	**VERTICAL_LINE**	**Grand Total**
CIRCLE	33	5	2	8			1	1			50
CIRCLE_FAST	8	11		17			3	7	4		50
CIRCLE_FAST_SLOW	3	2	16	8			9	4	8		50
CIRCLE_SLOW_FAST	8		1	40				1			50
DIAGONAL					50						50
HORIZONTAL_LINE	1			1	4	42	2				50
SQUARE	3	1	7				31	3	5		50
TRIANGLE	10		1	3			4	30	2		50
TRIANGLE_FAST	8		4	5			1	13	17	2	50
VERTICAL_LINE			2							48	50
Grand Total	**74**	**19**	**33**	**82**	**54**	**42**	**51**	**59**	**36**	**50**	**500**

Figure A.16: Forward9010 with Eight States

Bibliography

[CATA13] Sait Celebi, Ali S Aydin, Talha T Temiz, and Tarik Arici. Gesture Recognition Using Skeleton Data with Weighted Dynamic Time Warping. *8th International Conference on Computer Vision Theory and Applications (VISAPP 2013)*, page Paper #79, 2013. (cited on Page 10)

[Cha16] Chandralika Chakraborty. Issues and Limitations of HMM in Speech Processing : A Survey. 141(7):13–17, 2016. (cited on Page 10)

[Che03] F Chen. Hand gesture recognition using a real-time tracking method and hidden Markov models. *Image and Vision Computing*, 21(8):745–758, 2003. (cited on Page 6)

[EAHM08] M Elmezain, A Al-Hamadi, and B Michaelis. Real-time capable system for hand gesture recognition using hidden Markov models in stereo color image sequences. *16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2008, WSCG'2008*, (January 2017):65–72, 2008. (cited on Page 8, 17, and 18)

[EAHM11] M Elmezain, a Al-Hamadi, and B Michaelis. Hand gesture recognition based on combined features extraction. *World Academy of Science, Engineering and Technology*, 60(12):395–400, 2011. (cited on Page 5, 6, 8, and 10)

[Ern89] Jason Ernst. Hidden Markov Models. 1989. (cited on Page 7 and 9)

[FLO14] Tatsuya Fujii, Jae Hoon Lee, and Shingo Okamoto. Gesture Recognition System for Human-Robot Interaction and Its Application to Robotic Service Task. I:1–6, 2014. (cited on Page 8 and 18)

[FZF13] Simon Fong, Yan Zhuang, and Iztok Fister. A biometric authentication model using hand gesture images. *Biomedical engineering online*, 12(1):111, 2013. (cited on Page 31)

[Hu14] Xiaodong Xu; Yi Zhang; Shuo Zhang; Huosheng Hu. 3D hand gesture tracking and recognition for controlling an intelligent wheelchair. *International journal of Computer Applications in Technology*, 49(2):104–112, 2014. (cited on Page 3)

[ISTC14] Rodrigo Ibañez, Álvaro Soria, Alfredo Teyseyre, and Marcelo Campo. Advances in Engineering Software Easy gesture recognition for Kinect. 76:171–180, 2014. (cited on Page 1 and 2)

[KF11] Chang-Yi Kao and Chin-Shyurng Fahn. A Human-Machine Interaction Technique: Hand Gesture Recognition Based on Hidden Markov Models with Trajectory of Hand Motion. *Procedia Engineering*, 15:3739–3743, 2011. (cited on Page 5)

[KSS] The Kinect, Windows Sdk, and Windows Sdk. Human Skeleton Tracking. (cited on Page 11)

[LLKD] Nianjun Liu, Brian C Lovell, Peter J Kootsookos, and Richard I A Davis. Model Structure Selection & Training Algorithms for a HMM Gesture Recognition System Hidden Markov Model Structure and Training Methods. *Chart*, pages 1–6. (cited on Page 17)

[NdtLK12] Nhan Nguyen-duc thanh, Sungyoung Lee, and Donghan Kim. Two-stage Hidden Markov Model in Gesture Recognition for Human Robot Interaction. 9, 2012. (cited on Page 31)

[PSF] Carlos Palma, Augusto Salazar, and H C Feb. HMM and DTW for Evaluation of therapeutical gestures using kinect. (52). (cited on Page 2)

[Rab89] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, 1989. (cited on Page 10)

[RWA$^+$07] S Reifinger, F Wallhoff, M Ablassmeier, T Poitschke, and G Rigoll. Static and dynamic hand-gesture recognition for augmented reality applications. *Human-Computer Interaction, Pt 3, Proceedings;4552: 728-737 2007*, 4552:728–737, 2007. (cited on Page 6)

[SFC$^+$13] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. *Studies in Computational Intelligence*, 411:119–135, 2013. (cited on Page 11)

[SFS$^+$14] Thesis Submitted, Partial Fulfillment, Computer Engineering Supervised, Technology Rochester, N Y Dec, Approved By, Raymond Ptucha, Primary Advisor, Computer Engineering, Andreas Savakis, Secondary Advisor, Computer Engineering, and Nathan D Cahill. Gesture Recognition Using Hidden Markov Models Augmented with Active Difference Signatures by Himanshu Kumar. 2014. (cited on Page 1 and 6)

[Tan95] Donald O Tanguay. Hidden Markov Models for Gesture Recognition by. 1995. (cited on Page 9)

[XL12] W Xu and E J Lee. Continuous gesture recognition system using improved HMM algorithm based on 2D and 3D space. *International Journal of Multimedia and Ubiquitous Engineering*, 7(2):335–340, 2012. (cited on Page 2)

[YSBS01] Ho Sub Yoon, Jung Soh, Younglae J. Bae, and Hyun Seung Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 34(7):1491–1501, 2001. (cited on Page 3)