Using Gesture Recognition to Control PowerPoint Using the Microsoft Kinect

by

Stephen M. Chang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author ....
Department of Electrical Engineering and Computer Science
May 24, 2013

Certified by ....
Randall Davis
Professor of Computer Science
Thesis Supervisor

Accepted by ....
Dennis M. Freeman
Chairman, Masters of Engineering Thesis Committee

Using Gesture Recognition to Control PowerPoint Using the Microsoft Kinect

by

Stephen M. Chang

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2013, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis describes the design and implementation of a speech and gesture recognition system used to control a PowerPoint presentation using the Microsoft Kinect. This system focuses on the identification of natural gestures that occur during a PowerPoint presentation, making the user experience as fluid as possible. The system uses an HMM to classify the performed gestures in conjunction with an SVM to perform real-time segmentation of gestures. The fusion of these two models allows the system to classify gestures in real time as they are being performed instead of waiting until completion. The incorporation of speech commands gives the user an additional level of precision and control over the system. This system can navigate through a PowerPoint presentation and has a limited control over slide animations.

Thesis Supervisor: Randall Davis
Title: Professor of Computer Science

Acknowledgments

First, I would like to acknowledge Professor Davis for his guidance and input throughout the year which made this project possible.

Additional thanks to all my friends in the MIT community who have offered constant support and encouragement through their friendship. They made my experience at MIT over the past five years not only bearable, but one of the most enjoyable experiences in my life despite the academic rigors.

Finally, I would like to thank my family for always being there for me. Most importantly, thanks to my parents for providing me with all the support and opportunities throughout my life that have led me to this point. None of this would have been possible without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In today's world, technology pervades nearly every aspect of the average person's daily life. People interact with computers and other technology as frequently as they do with other people and they should have the ability to communicate with computers as naturally as they do with other humans. Speech is perhaps the most comfortable form of communication between humans. It is quick, efficient, and allows people to express themselves with great degrees of freedom, limited only by their own vocabulary. Since the dawn of computers half a century ago, people have dreamed of being able to have conversations with robots and other artificial intelligences as easily as they do with other humans. Unfortunately, keyboard and mouse have been the primary means of interfacing with computers even to this day. While effective in many situations, they are limiting and not a particularly natural means of interaction.

Gesture recognition is a current area of research that is trying to address this problem. Everyone is familiar with gestural interaction with other humans. It occurs naturally during speech as a way people for people to express themselves. Gestures are a form of body language that are essential to effectively communicate ideas in addition to spoken language. People already gesture when communicating with other humans, so why not use this mode of communication for natural interaction with computers.

1.2 Goal

This thesis aims to build a speech and gesture recognition system that uses natural gestures to control a PowerPoint presentation. When people give PowerPoint presentations, they usually have a clicker object that can control the slides remotely. However, holding onto the clicker during the presentation occupies the hand. When a person's hand is already occupied, the range of motions and gestures that can be performed is limited. This limitation is not necessarily a physical limitation; the presenter may simply be (potentially unconsciously) unwilling to perform certain gestures while their hand is occupied. The primary goal of this system is to free the user from these restraints and automatically react to the naturally spoken words and gestures throughout the presentation.

In many gesture recognition systems, the vocabulary of recognizable gestures are contrived and unnatural. While they are usually not arbitrary as they have some logical connection between the gesture and its response, they are not gestures that a user would perform naturally on their own. This system focuses on recognizing natural gestures and phrases that users would be likely to perform or say during a PowerPoint presentation even without the system. By focusing the system's attention on natural phrases and gestures, the user should not have to think about performing artificial and awkward gestures to control the PowerPoint, which could potentially be even more distracting to the user than holding onto a small clicker. The gesture recognition system should not hinder the user, allowing the human-computer interaction to be as seamless and intuitive as possible.

1.2.1 What is a gesture?

In this thesis, a gesture is defined as a meaningful sequence of hand and arm poses over time. A pose is simply a configuration of the arm and hand joint positions at a single point in time. In this work, the body is divided into three sections: left arm, right arm, and torso. For the types of gestures in this system, the two arms

Gesture	Functionality	Possible Speech
forward	next slide	“moving on”
backward	previous slide	“going back”
forward scroll	skip ahead x slides	“skipping the next x slides”
backward scroll	skip back x slides	“going back x slides”
pointing	trigger animation	none

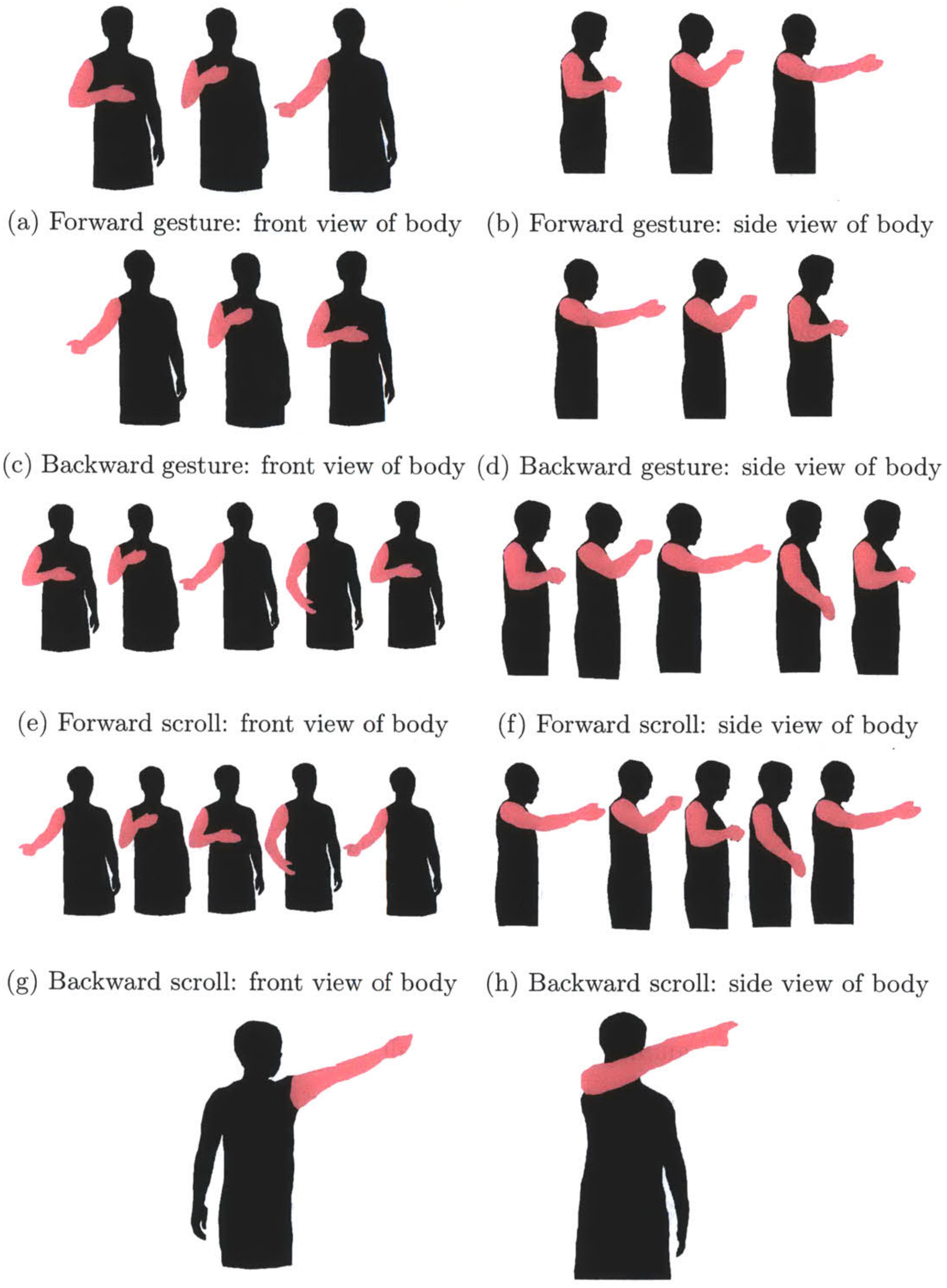
Table 1.1: Possible gesture and speech combinations

are the most significant parts of the body that are tracked. The torso is used for identifying the general location of the body with respect to the Kinect. The two arms are analyzed separately allowing the gestures to be performed with either arm independently of the other.

1.2.2 Target Gestures

This work focuses on two functions that are a key part of any presentation: navigation through slides and triggering of onscreen animations. Desired navigation functionalities are “next slide”, “previous slide”, and jumping forward or back any number of slides or to a specific slide. These are general navigation commands that can be applied to any PowerPoint presentation. The onscreen animation triggering, on the other hand, are less consistent between each slide. The exact effect that the triggering command has on a slide is dependent on the design of the slide. Slides may have any number of different animations or none at all. Common animations are on build slides where there is a list of bullet points that are initially hidden and sequentially revealed. The animation triggering function can be used to reveal each of these bullet points. In general, the triggering command will cause the next (if any) animation to occur.

Based on these desired functionalities there are five basic gestures that can be performed with either hand: forward, backward, forward scroll, backward scroll, and pointing. These gestures are diagrammed in Figure 1-1. Table 1.1 shows which gestures control which functionalities as well as possible corresponding phrases.



(a) Forward gesture: front view of body (b) Forward gesture: side view of body

(c) Backward gesture: front view of body (d) Backward gesture: side view of body

(e) Forward scroll: front view of body (f) Forward scroll: side view of body

(g) Backward scroll: front view of body (h) Backward scroll: side view of body

(i) Pointing to side (j) Pointing across body

Figure 1-1: This figure shows some basic body poses of the five different gestures with the right hand from the front and side points of view. 1-1i and 1-1j illustrate the two different ways a user can perform a pointing gesture to the left: either by extending the left hand out to the side or crossing the right hand across the body. Each of these gestures can be performed with the left hand in the same manner.

Figures 1-1a and 1-1b show the how the forward gesture is performed: an outward-directed extension of the arm that starts at the mid-torso level close to the body and ends with an outstretched arm and an upward facing palm in front of the body. The backward gesture, Figures 1-1c and 1-1d, is the opposite, performed as an inward-directed movement with the arm starting with an outstretched arm palm-up and ending close to the body, again at mid-torso level. The forward and backward scroll gestures can be visualized as continuous and smooth repetitions of the forward and backward gestures. Figures 1-1e through 1-1h each show a single cycle of the forward and backward scroll gestures. The pointing gesture can be performed in one of two ways. Given a pointing direction, to the left of the body for example, the user can use either hand to perform the pointing gesture: either with the left hand extended out on the left side of the body (Figure 1-1i), or with the right hand crossing in front of the body resulting in a leftward point (Figure 1-1j).

1.2.3 Communicative vs. Manipulative Gestures

In this work, we consider two types of gestures: communicative and manipulative. Communicative gestures convey a complete idea with the gesture. A system can only begin to react appropriately to a communicative gesture after it has been performed to completion and has been disambiguated from other gestures. An example of a communicative gesture on a touch screen is a "flick" to navigate between pages of an application like a web browser or photo viewer. The system will not know to interpret the gesture as a "flick" instead of a drag until the entire gesture has been performed. Manipulative gestures by contrast give the user direct control of the system which reacts in real-time to the gesture as it is being performed. An example of a manipulative gesture is a two finger pinch-to-resize an image on a touch screen. As soon as two fingers touch the image, the image starts to resize itself depending on the size of the pinch.

In this system, the forward, backward, and pointing gestures are communicative gestures and the scroll gestures are manipulative gestures. The system cannot accurately interpret a forward, backward, or pointing gesture until it has been performed to completion. On the other hand, the system should be able to identify when the user is performing a scroll gesture and react accordingly. In this case, the system should navigate forwards or backwards an additional slide each time another cycle segment has been performed.

Chapter 2

Background

This gesture recognition and PowerPoint controlling system uses a number of different technologies and techniques to achieve its goal. The major technologies used are the Microsoft Kinect sensor for body input data, Support Vector Machines (SVMs) and Hidden Markov Models (HMMs) for gesture recognition, and Carnegie Mellon University's Sphinx4 Java library for speech recognition. These are described in this section.

2.1 Microsoft Kinect

The Microsoft Kinect (Figure 2-1) is a motion sensing device with a variety of inputs that performs real-time skeleton tracking. The Kinect was originally released in November 2010 as an accessory to the Xbox gaming system for playing video games without using a physical controller. In 2011, Microsoft released the Kinect SDK to the public. The Kinect has a standard RGB camera, a 3D infrared depth sensor, and

Figure 2-1: Kinect sensor

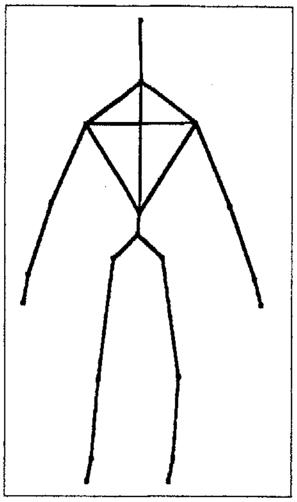


Figure 2-2: Sample output skeleton from the Kinect, drawn in Processing

a multi-array microphone allowing a variety of different inputs signals to be used. Body tracking had previously been its own area of research. With the release of the Kinect SDK the general public was able to start using body tracking in their own applications and research with little overhead and cost.

2.1.1 How It Works

The Kinect performs 3D depth sensing by emitting a structured point cloud pattern of infrared (IR) light and calculating the depth from the images taken with its IR sensor. Because this IR point cloud originates at a single point in the Kinect sensor, as the distance from the Kinect increases, the point cloud pattern disperses proportionally to the distance the light has traveled. By measuring the offset between the expected location of the IR grid pattern at a calibrated distance and its actual location, the Kinect can calculate the depth at each point in the projected point cloud. Using this depth image, the Kinect can identify foreground objects and determine people and their poses by comparing the detected body to millions of stored examples of body poses. The Kinect then uses a randomized decision forest technique to map the body depth image to body parts from which the skeleton representation can be built [13] (Figure 2-2).

2.2 Recognition Tools

2.2.1 Support Vector Machines - SVMs

A Support Vector Machine (SVM) is model that can classify single n-dimensional points based on a training set of points and a classification kernel. SVM classification attempts to find classification boundaries between the points that maximizes the distance between the closest set of data points with opposite signs while simultaneously minimizing the number of misclassified data points. The kernel function $\kappa(x_i, x_j)$, which can be linear, polynomial, Gaussian, etc., determines the type of classification boundary used. This classification problem can be formulated as the following primal-form optimization problem [2] [4] given the training vector $\mathbf{x_i} \in R^n, i = 1, ..., l$ and corresponding indicator vector $y = R^l$ where $y_i \in \{1, -1\}$:

$$\begin{aligned} \underset{\mathbf{w},b,\boldsymbol{\xi}}{\arg\min} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, ..., l, \end{aligned}$$

where $\phi(\mathbf{x})$ is a fixed feature-space transformation, b is a bias parameter, ξ_n are the slack variables that penalize misclassifications, and C is the regularization parameter for the ξ_n. This particular formulation of this optimization problem is known as C-Support Vector Classification (C-SVC). However the quadratic programming problem posed by this primal form is computationally complex. An alternative dual-formulation of this problem, which is less computationally complex, is presented as follows:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\arg\min} \quad & \tfrac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{e}\boldsymbol{\alpha} \\ \text{subject to} \quad & \mathbf{y}^T\boldsymbol{\alpha} = 0, \\ & 0 \leq \alpha_i \leq C, i = 1, ..., l, \end{aligned}$$

where $\mathbf{e} = [1, ..., 1]^T$, Q is an l-by-l positive semidefinite matrix, $Q_{ij} \equiv y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$, and $\kappa(x_i, x_j) \equiv \phi(\mathbf{x_i})^T\phi(\mathbf{x_j})$ is the kernel function. The dual form reformulates this problem in terms of kernels allowing this classification problem to be easily applied to higher dimensional feature spaces. This dual form is solved to give the optimal

$$\mathbf{w} = \sum_{i=1}^{l} y_i \alpha_i \phi(\mathbf{x_i})$$

which yields the decision formula

$$\operatorname{sgn}(\mathbf{w}^T \phi(\mathbf{x}) + b) = \operatorname{sgn}(\sum_{i=1}^{l} y_i \alpha_i (\kappa(\mathbf{x}_i, \mathbf{x}) + b).$$

This C-SVC classifier is used to classify the skeleton poses. This project uses LIBSVM [3], an open source SVM library written in C++, to solve this problem. The LIBSVM implementation uses a $\kappa(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$ where $\gamma = \frac{1}{\text{num features}}$. In this formulation, the SVM classification problem is fundamentally a two-class problem. However, for this pose detection problem where each pose is a data point, the n joints are the feature space, and the k different poses are the classification results, there are more than two classes. A system that wants to identify specific poses and not simply determine if there is a pose must go beyond this two-class formulation. To solve this problem, LIBSVM uses a one vs. many approach that generates k SVMs for each pose then uses a generalized Bradley-Terry [6] model that does pariwise comparisons to determine the SVM that produced the most likely result.

As gestures are sequences of skeleton positions over time, an SVM is not able to perform gesture recognition on its own. While the SVM cannot actually identify entire gestures, it is able to recognize when a skeleton is in a particular pose. These poses are very useful when considering that gestures are nothing more than sequences of skeleton poses.

2.2.2 Hidden Markov Models - HMMs

Hidden Markov Models (HMMs) are much better suited towards learning gestures than SVMs. SVMs can classify single points, while HMMs can be used to classify sequences of points. Markov Models are simply a collection of states that have transition probabilities between each state. The Hidden Markov Model is a variation on the standard Markov models where the states are "hidden" and are reflections of the known observable states. HMMs are great for learning gestures, which are simply sequences of body positions. Given an input sequence of observation features, the HMM can identify how likely it is that the sequence belongs to any of the classes of sequences that the HMM was trained on. HMMs are already frequently used in speech recognition systems [7] [11] and natural language modeling [9]. These HMMs can also be applied to gesture recognition systems.

HMMs are characterized by two variables,

1. N, the number of hidden states, denoted as $S = \{S_1, ..., S_N\}$

2. M, the number of observation symbols in each state

three probability distributions,

1. $A = \{a_{ij}\}$, the state transition probability distribution where

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_j] \quad 1 \leq i, j \leq N$$

2. $B = \{b_j(k)\}$, the observation symbol probability distribution in state j where

$$\begin{aligned} b_j(k) = [v_k \text{ at } t | q_t = S_j] \quad & 1 \leq j \leq N \\ & 1 \leq k \leq M \end{aligned}$$

3. $\pi = \{\pi_i\}$, the initial state distribution where

$$\pi_i = P[q_1 = S_i] \quad 1 \leq i \leq N$$

and a vocabulary of observation symbols. Together, these components provide a complete specification of an HMM [12]. N,M and the observation symbols describe the general structure of the HMM, while A,B, and π characterize the behavior of the HMM. Given an HMM, the three probability distributions, A,B, and π, are the parameters that must be adjusted to maximize the likelihood that a given set of observation sequences is described by the HMM. For convenience, the full set of model parameters will be represented as $\boldsymbol{\theta} = (A, B, \pi)$. This thesis uses the Kevin Murphy's HMM toolbox for MATLAB[1] which implements the expectation-maximization (EM) algorithm to train the HMMs and the forward algorithm for calculating the probability of an observation sequence given a model.

EM Algorithm

The expectation-maximization (EM) algorithm [5] [10]is an iterative algorithm that calculates the $\boldsymbol{\theta}$ to maximize the likelihood function $p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ where $\mathbf{X}$ are the observation variables and $\mathbf{Z}$ are the latent variables. The overall EM algorithm is described as follows [1]:

1. Initialize $\boldsymbol{\theta}^{\text{old}}$
2. E step - evaluate $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.
3. M step - evaluate $\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$.
4. If not convergence criteria not met, set $\boldsymbol{\theta}^{\text{old}} = \boldsymbol{\theta}^{\text{new}}$ and repeat from Step 2.

Because these parameters $\boldsymbol{\theta}$ fully characterize an HMM, these maximum likelihood estimations will produce an HMM that maximizes the probability of the occurrence of the training sequences.

Forward Algorithm

The forward algorithm calculates the probability $\alpha_t(i) = P(O_1 O_2 ... O_t, q_t = S_i|\theta)$ of an observation sequence, $O_1, ..., O_t$ for each model θ with the following inductive procedure:

1. Initialization: $\alpha_1(i) = \pi_i b_i(O_1), 1 \leq i \leq N$.

[1] http://www.cs.ubc.ca/ murphyk/Software/HMM/hmm.html

2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \quad 1 \leq t \leq T-1,$$
$$1 \leq j \leq N$$

3. Termination: $P(O|\theta) = \sum_{i=1}^{N} \alpha_T(i)$.

This procedure only takes $O(N^2T)$ calculations, which is a significant improvement compared to $O(2TN^T)$ calculations using a naive enumerative method of all the possible state sequences. Having the probability that a given observation sequence will occur in each of the HMMs gives a viable means of classification. After running the forward algorithm on a sequence for each model, the most likely classification is simply the model that returns the highest probability.

2.2.3 Speech Recognition with Sphinx-4

The speech recognition in this system uses Sphinx-4, a Java speech recognizer developed by the Sphinx group at Carnegie Mellon University in collaboration with Sun Microsystems Laboratories, Mitsubishi Electric Research Labs (MERL), and Hewlett Packard (HP) [14]. The Sphnix-4 framework (Figure 2-3) has three primary modules: FrontEnd, Linguist, and Decoder. The FrontEnd takes the raw input and extracts the features. The Linguist has three components: an AcousticModel (HMMs that encode the structure of the basic sound units), a Dictionary that provides a mapping between these sound units and the appropriate phonemes (the pronunciation information of each sound in a language), and a LanguageModel that provides the scope of language to be used in the application. The LanguageModel can be provided in a variety of different formats, such as word grammars or n-grams. The Linguist provides a SearchGraph which is any topological search space that allows the Decoder to produce valid sequences of sounds and words from the set of input features. The exact form of the topological search space depends on the implementation of the

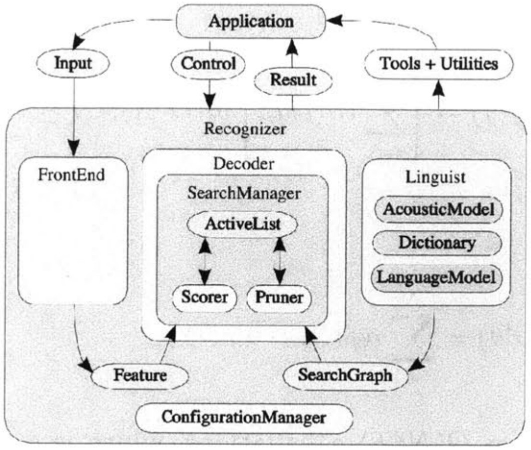


Figure 2-3: "Sphinx-4 Decorder Framework. The main blocks are the FrontEnd, the Decoder, and the Linguist. Supporting blocks include the ConfigurationManager and the Tools blocks. The communication between the blocks, as well as communication with an application, is depicted."[2]

SearchGraph. The Decoder does the final scoring of the possible valid words and outputs the result to the application.

[2] http://cmusphinx.sourceforge.net/sphinx4/doc/Sphinx4Whitepaper.pdf

Chapter 3

How the system works

3.1 Gesture Recognition Approach

This system achieves gesture recognition by using both SVMs and HMMs. While it appears that the HMM is perfectly suited for gesture recognition, an HMM alone cannot solve this because we need to segment the continuous sequence of body movements into distinct sequences for the HMM to classify.

One solution could be to take the approach used in speech recognition and look for motion silence (lack of body motion). Gestures are sequences of body positions, and a body cannot be gesturing if it is motionless. This approach is a good starting point, however there are a number of limitations. A motion silence based system would only work if the user froze after the completion of each gesture. This is not well suited our focus of giving a PowerPoint presentation. The system's goal is to identify the relevant gestures that are performed naturally and react accordingly. The assumption of natural gestures is that people perform gestures continuously hence motion silence is not always applicable: gestures are typically not immediately preceded and followed by freezing in place. While there are often abrupt velocity changes before or after a gesture, they are not particularly reliable measures of gesture beginnings and endings. In initial tests using motion silence to segment the gestures, the thresholds for determining motion silence were found to be unreliable. If the thresholds were too large, consecutive gestures would get grouped together. If the thresholds are too small, the system would prematurely decide a gesture has ended, sending only partial sequences to the HMMs. This erratic behavior is undesirable and made it difficult for the system to reliably detect the starts and ends of gestures indicating that motion silence may not be the most appropriate method for dividing these gesture sequences.

The approach that this system uses is to explicitly identify the beginnings and endings (and other key points) of the gestures with an SVM. By using an SVM to identify these poses instead of relying on motion silence, constraints are lifted off the user. For instance, users can transition directly from one gesture to another gesture without interruption. There is also the advantage that using an SVM will make the system react more quickly to the gestures. With motion silence, the system has to wait for some specified amount of time to pass after a gesture the skeleton has stopped moving before it can determine that the gesture is definitely over. However, with the SVM approach, once the "end" of the pose is identified, the system can send the skeleton sequence to the HMM without delay.

3.1.1 Skeleton Preprocessing

The skeleton data provided by the Kinect are the joint positions in a coordinate system where the Kinect is at the origin. Before these features are sent to the SVM, they are converted to a body-centered coordinate system and scaled. The body origin is defined as the 3D centroid of the left, right, and center hip positions. The conversion from the Kinect's world coordinates to the body coordinates makes future classification of the skeletons robust to the body's relative position to the Kinect sensor. These body-centered coordinates are then scaled to the range $[l, u]$ in order to avoid bias for features that have larger numerical values, (i.e., for joints that are farther from the origin).

$$\frac{x - x_{min}}{x_{max} - x_{min}} = \frac{x' - l}{u - l}$$

$$x' = \frac{(x - x_{min})(u - l)}{(x_{max} - x_{min})} + l$$

AnkleLeft	FootLeft	Head	KneeLeft	**ShoulderRight**
AnkleRight	FootRight	HipCenter	KneeRight	Spine
ElbowLeft	**HandLeft**	HipLeft	ShoulderCenter	**WristLeft**
ElbowRight	**HandRight**	HipRight	**ShoulderLeft**	**WristRigh**t

Table 3.1: List of 20 joints provided by the Kinect. Bolded joints are the 8 arm and hand joints used in the gesture recognition.

where x is the original feature, x' is the scaled feature, and $[x_{min}, x_{max}]$ are the values that indicate the minimum and maximum values of the feature in the entire training set. The next step of data preprocessing is the selection of the relevant joints that the models are trained on. Out of the 20 joint positions provided by the Kinect, only 8 are used for this gesture recognition (Table 3.1). These are the arm joints: hand, wrist, elbow, and shoulder for both left and right arms.

For the SVM, those scaled, body-centered arm joints are the final form of the feature set. The features for the HMMs are further augmented. While each SVM instance is a single skeleton feature vector, HMMs require a sequence of feature vectors. With this sequence of skeleton poses, it becomes useful to include joint velocities as additional features. The addition of joint velocities incorporates not only the overall speed of the gesture, but also directionality at each point. These velocities can be potentially useful in the discrimination of gestures that occupy the same physical space but pass through those spaces differently. These velocities were calculated with a sliding window $w = \min(3, \text{ceil}(\text{gestureLength}/3))$. The ceil(gestureLength/3)) is used to assure that there will be a window of an appropriate size even in the case of a skeleton sequence that is very short (<10 frames). This window size was determined empirically by looking at the average gesture segment lengths shown in Table 3.2.

It should be noted that the SVM feature set could also be augmented to include joint velocities as is done for the HMM features. While the SVM itself only receives a single skeleton at a time, the overall system receives a constant stream of skeletons. The system could keep a rolling window of past skeletons from which velocities for

Gesture	Average Frames	Std Dev
Forward	28.8907	4.0585
Backward	37.7568	20.3833
Fwd Scroll (start-end)	10.5285	4.7573
Bwk Scroll (start-end)	10.0545	6.8098
Fwd Scroll (start-start)	14.7841	1.7342
Bwk Scroll (start-start)	14.8336	2.0040
Fwd Scroll (end-end)	14.5779	2.3535
Bwk Scroll (end-end)	14.7146	2.3715

Table 3.2: Average length of gesture segments

every skeleton could be computed as each new skeleton is received. This augmentation to the SVM feature set was not done in this implementation, but could improve the SVM's ability to identify the start and end poses of each gesture.

3.2 System Architecture

3.2.1 Data flow

This system is composed of a number of different interconnected programs and modules. There are 3 main components:

1. KinectDaemon (C#) - serializes joint data, streams it over network

2. PowerPoint Controller (Java) - fuses speech, pose, and gesture recognition to control PowerPoint

3. MATLAB - data manipulation, HMM, and SVM learning and classification

Figure 3-1 diagrams the overall data flow through the system. The Skeleton Visualizer is not one of the three main components mentioned above because it is not essential to the system. It is an optional application written in Processing[1] that shows a visualization of the skeleton. This gives the user some feedback about what kind of data the system is receiving. The Kinect is also unlisted above. While it performs

[1] Processing is an open source programming language that builds off of Java. Processing is used to programmatically create drawings and animations using Java libraries, but with a simplified syntax. www.processing.org

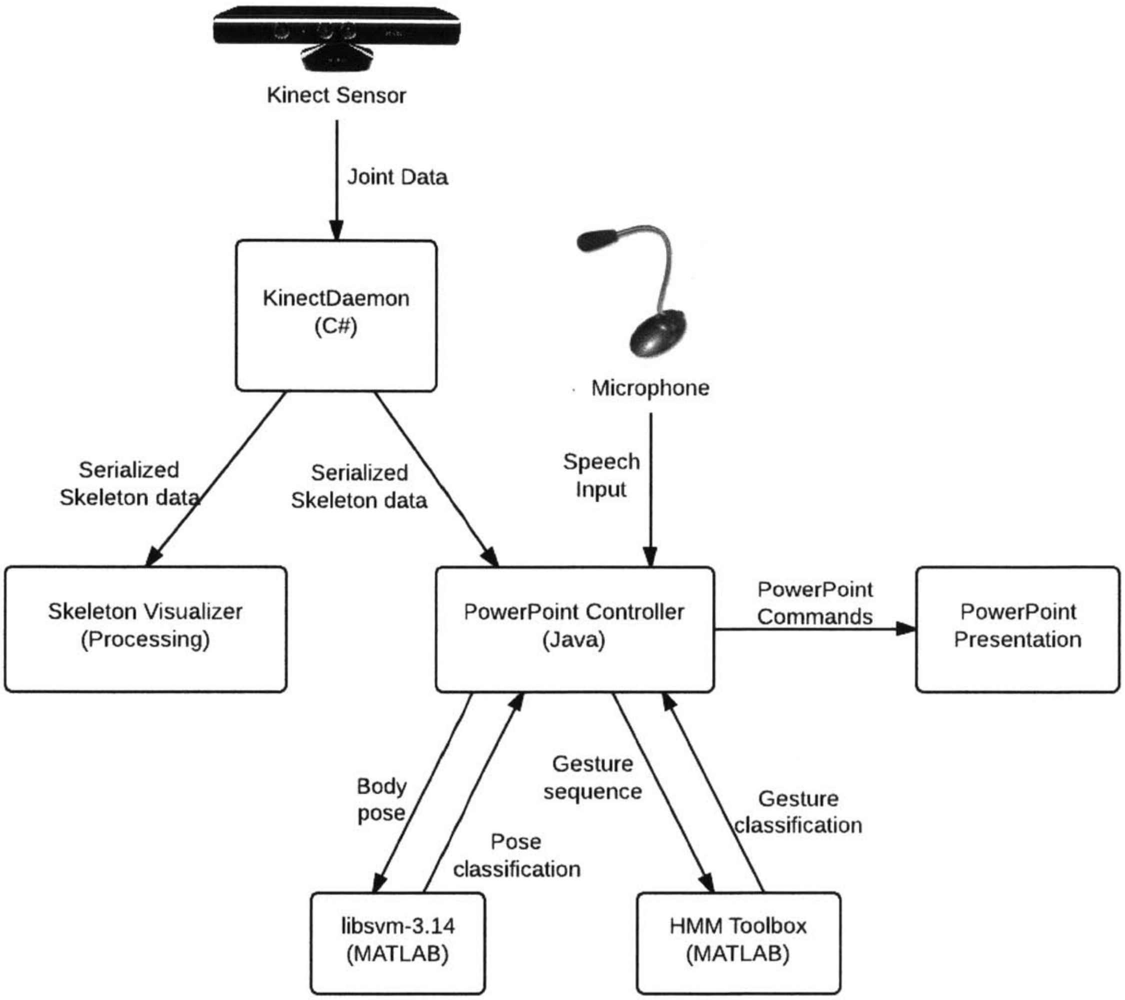


Figure 3-1: A diagram of the overall system architecture showing the data flow

much of the raw data processing, it is ignored in this section and is treated as a black box sensor that simply outputs skeletal joint positions. Section 2.1 discusses the Kinect data processing in more detail.

KinectDaemon

The joint data starts in the KinectDaemon project. The KinectDaemon project is an open-source project that sends out Kinect joint data over a simple TCP server [8]. The KinectDaemon is a C# program that accesses Microsoft's official Kinect SDKv1.0. The data can be sent either as serialized C# objects to other C# programs or as JSON strings encoding the joint data. Sending the data as JSON strings makes it

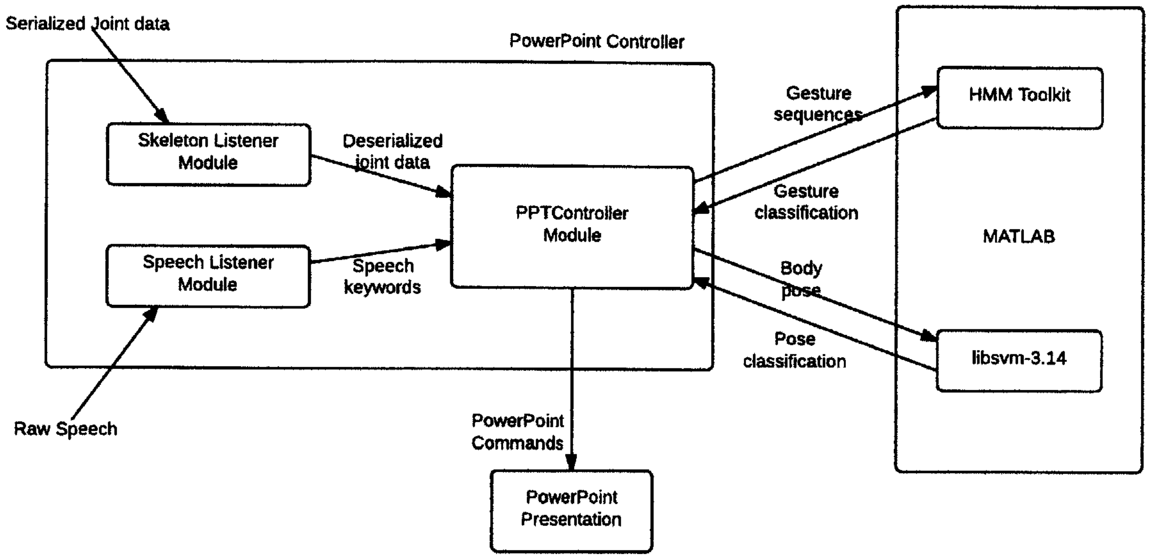


Figure 3-2: A diagram of the PowerPoint Controller architecture and data flow

convenient for any program to use the basic skeleton information without needing other SDKs to access the Kinect. Another benefit of using this server system to stream the data is that multiple applications can use the Kinect data simultaneously. This is not strictly necessary, but does allow the user to run the Skeleton Visualizer application concurrently as the gesture recognition is being performed.

PowerPoint Controller module

The PowerPoint Controller is the primary destination for the KinectDaemon's joint data. The PowerPoint Controller is the high level controller that fuses the different modalities of inputs and pose/gesture classifications. Figure 3-2 diagrams the overall architecture of the PowerPoint Controller project. The SkeletonListener and SpeechListener modules run concurrently on their own threads and notify the PowerPoint Controller module when new skeleton or speech data is acquired. Each new skeleton is sent to the SVM to classify the body pose. The body poses are then used to determine when to start and stop the recording the skeleton sequences to be sent to the HMM for gesture recognition. Section 3.3 describes this exact process in depth.

The PowerPoint Controller module is also responsible for combining the speech information with the classified gestures to determine the user's desired outcome and send the appropriate commands to a PowerPoint presentation. This fusion of speech and gestures is described more fully in Section 3.4. The PowerPoint Controller module exerts control over a presentation by running in the background and injecting keystrokes. While not the most sophisticated solution, the module would easily be able to incorporate a Java-compatible API for programmatically manipulating a PowerPoint presentation.

Trainer module

The Trainer is a parallel class to the PowerPoint Controller class that is used to collect data and train HMM models if so desired. The system is not fully automated and is primarily meant to collect data which are later used to train the models separately in MATLAB. While the Trainer does not produce finalized models, it can collect and format gesture sets, then save them as a MATLAB dataset. The Trainer module also has speech integration that allows a single user to collect fully formed gesture sets without aid. For each gesture class, the user can indicate the beginning and ends of each gesture example with the following phrases: "begin gesture" to start recording and "cut gesture" to stop recording an example. After all the examples for a single gesture class has been collected, the user can start collecting gestures for the next class by saying "next gesture". The integration of speech in the training module allows the user to efficiently exert control over the system and indicate exactly when the gestures should start and end without disrupting the actual gesture execution.

3.3 Gesture Abstraction

One of the problems that arises between the communicative (forward/backward/pointing) and manipulative gestures (scrolling) is the reaction time. The HMM is only able to classify a gesture once it has been performed to completion. That is the original reasoning for using the SVM in the first place. The SVM determines where the start and end of the gesture is and sends it to the HMM to be further classified. This works well for the short communicative gestures where the meaning becomes unambiguous only after it has been performed to completion. This is not the case for manipulative gestures which are meant to elicit a response from the system as it is being performed (i.e., before the gesture is finished). There needs to be some way to identify that this gesture is being performed at the same time it is being performed.

Real-time detection of these manipulative gestures is achieved by abstracting the forward and backward scrolling gestures as a quick and repetitive succession of their respective forward and backward communicative gestures. Take the forward scrolling gesture for instance. It is a repetitive cyclic motion composed of the forward gesture and the return of the hand to the starting position. Each cycle passes through two poses, *close* and *far*, which are the poses that are used to split each cycle into these two motions. These repetitive cyclic motions can be visualized with the plots of the gesture's joint positions over time as in Figure 3-3. The forward scroll plot in Figure 3-3b shows a sequence of 'mounds' that have the same general shape as the forward gesture (3-3a). The plateaus in the y and z axes in the forward gesture are a result of the arm freezing momentarily in its end position, or in the *far* pose.

The forward gesture starts with the arm in a *close* position, follows a concave-down curved path outward until it reaches a *far* position. The return motion is the reverse; the arm starts in the *far* position, follows a concave-up curved path toward the body until it reaches a *close* position. The backward scroll gesture is the same, but in reverse. The cycle generally follows the same path as the forward scroll, but rotates in the opposite direction and the backward gesture starts in the *far* position and ends in the *close* position.

This poses an interesting problem. The start pose of one gesture is the exactly the same as the end pose of another gesture. The original goal of the poses was to identify the beginning and endings of the gestures to send to the HMM for classification. However, if the system uniquely knew the start and end pose of each gesture before sending it to the HMM, the HMM would be little more than a confirmation of the gesture that the SVM has already classified. What the SVM actually provides are key poses that may *potentially* be the start or end of a gesture.

There are 3 meaningful poses that the SVM aims to classify:

1. *close*: hand is held in front of the body at the mid-torso level (between waist and chest) within a forearm's distance from the body, about one foot

2. *far*: hand is outstretched at mid-torso level in front of the body

3. *pointing*: hand pointing behind the body above shoulder level (either outstretched to the side or across the body with the opposite arm)

The pointing gesture is classified solely with the SVM. This system's goal is to pick out meaningful gestures from the presenter's natural gestural movements. The pointing gesture where an arm is outstretched to the side and angled behind the speaker's body. Pointing is a very deliberate gesture because the arm positions of a pointing gesture lies so far outside the body's natural range of gestural movement. As a consequence, it is unlikely that the presenter will unintentionally perform a pointing gesture. This assumption does not hold true for the other four gestures that are performed at the mid-torso level directly in front of or just to the side of the body. The mid-torso level in front of the body is the zone where a speakers hands most often reside while giving a presentation. Consequently it is likely that the speaker will match a *close* or *far* pose without performing the notable gestures, thus requiring the HMM to determine if these poses are indeed part of a recognized gesture.

3.4 Controlling a PowerPoint Presentation

3.4.1 Identifying Scroll Gestures

With the gesture abstraction described above, there are essentially only 3 identifiable gestures: forward, backward, and pointing. Despite there only being 3 identifiable gestures, the system still needs to differentiate between the 5 original gestures. This is achieved by keeping track of the recently classified gestures within a time frame. The system keeps track of the average length of the singular forward and backward gestures. In its default state, the system is idling and waiting for a gesture. During this idling state, once a forward or backward gesture is identified, the system begins to wait for another gesture.

If the specified time threshold elapses without identifying another gesture, the gesture is classified as a singular forward or backward gesture. On the other hand, if another gesture is identified within the allotted time, the system classifies the gesture sequences as a scroll gesture. The sequence of recently identified gestures are recorded. In this way, the system keeps a tally of the direction of each identified scroll cycles. As each new scroll cycle is identified, the running tally is updated. If the identified cycle direction matches the majority of the recent cycle directions, it is counted towards the overall scroll gesture. If the direction contradicts overall direction of the scroll so far, it does not send a command to the PowerPoint presentation but is still counted towards the tally in case the overall direction of the scroll reverses. This procedure provides a method for real-time error correction for each of the scroll segments as well as count the number of scroll cycles. This allows the system to reliably identify the direction of a scroll gesture without having to wait for the entire gesture to be performed to completion.

3.4.2 Speech Integration

PowerPoint can be controlled with either gestures alone or a fusion of speech commands and gestures. Each of the 5 programmed gestures, forward, backward, forward scroll, backward scroll, and pointing, have default actions associated with them. The addition of speech commands can have one of two effects on the resulting action: either augment or nullify the command.

Gestures are nullified by the speech if the direction of the spoken command contradicts the direction determined by the gesture recognition. The only commands that are

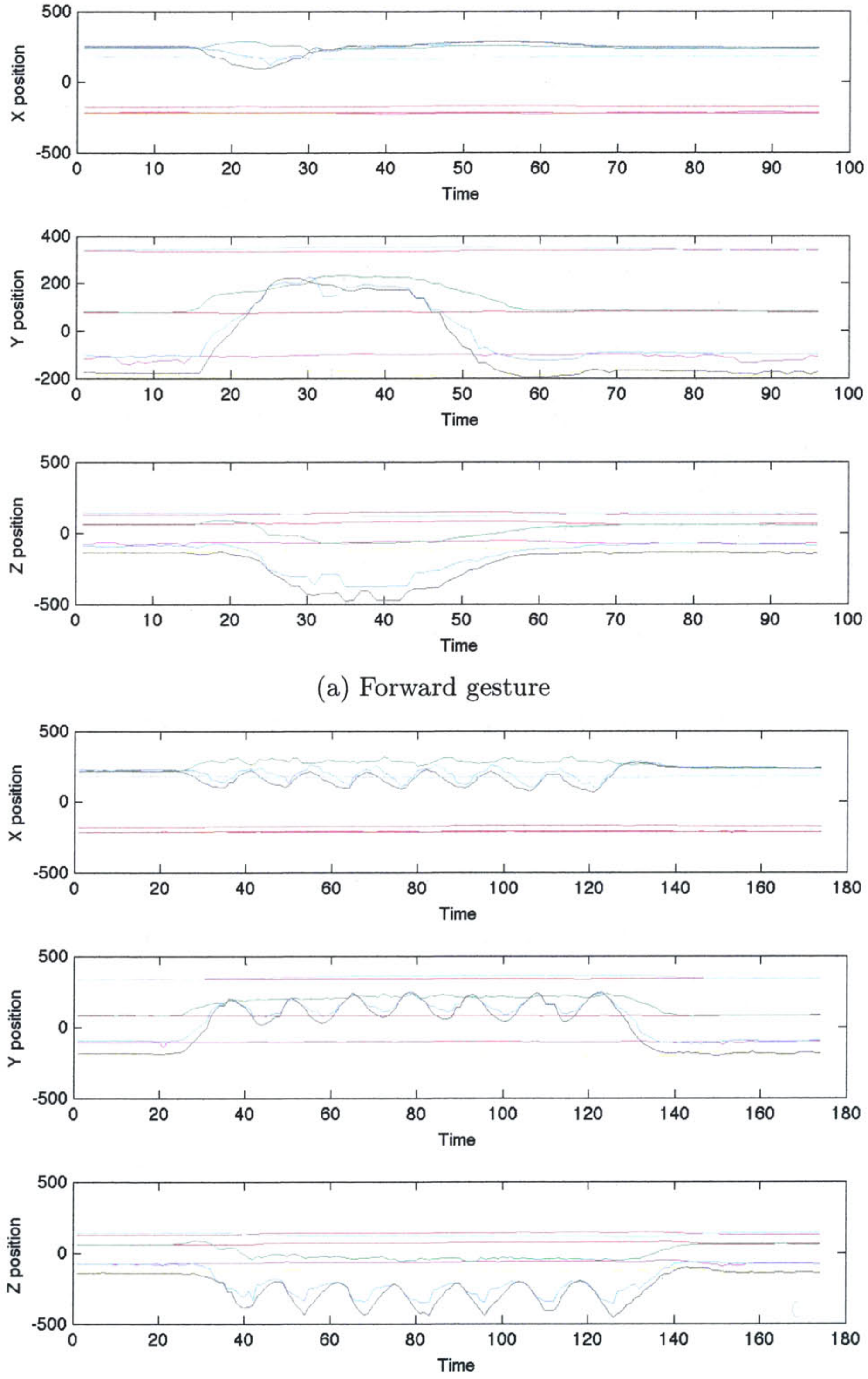
Gesture	**Action**
forward	forward one slide
backward	backward one slide
forward scroll	forward n slides, determined by number of scrolls
backward scroll	backward n slides, determined by number of scrolls
pointing	trigger animation, not next slide

Table 3.3: Default gesture commands

Speech	**Gesture**	**Action**
skipping ahead x slides	forward scroll	forward x slides
skipping back x slides	backward scroll	backward x slides
skipping to slide x	forward scroll	skip directly to slide x if x >current slide
skipping to slide x	backward scroll	skip directly to slide x if x <current slide
skipping to first slide	forward scroll	skip to first slide
skipping to last slide	backward scroll	skip to last slide
skipping to beginning	forward scroll	skip to first slide
skipping to end	backward scroll	skip to last slide

Table 3.4: Speech augmentations of gesture commands

augmented by speech are the scrolling gestures. The default scroll action is to count the number of scrolling repetitions and navigate the appropriate number of slides forward or backward. The speech commands can override this default behavior and specify either the exact number of slides to navigate in either direction or the exact destination slide. All other speech and gesture combinations will have no effect on the resulting action.



(a) Forward gesture

(b) Forward scroll

Figure 3-3: Graphical representation of gestures, plotting joint positions over time with respect to the body center

Chapter 4

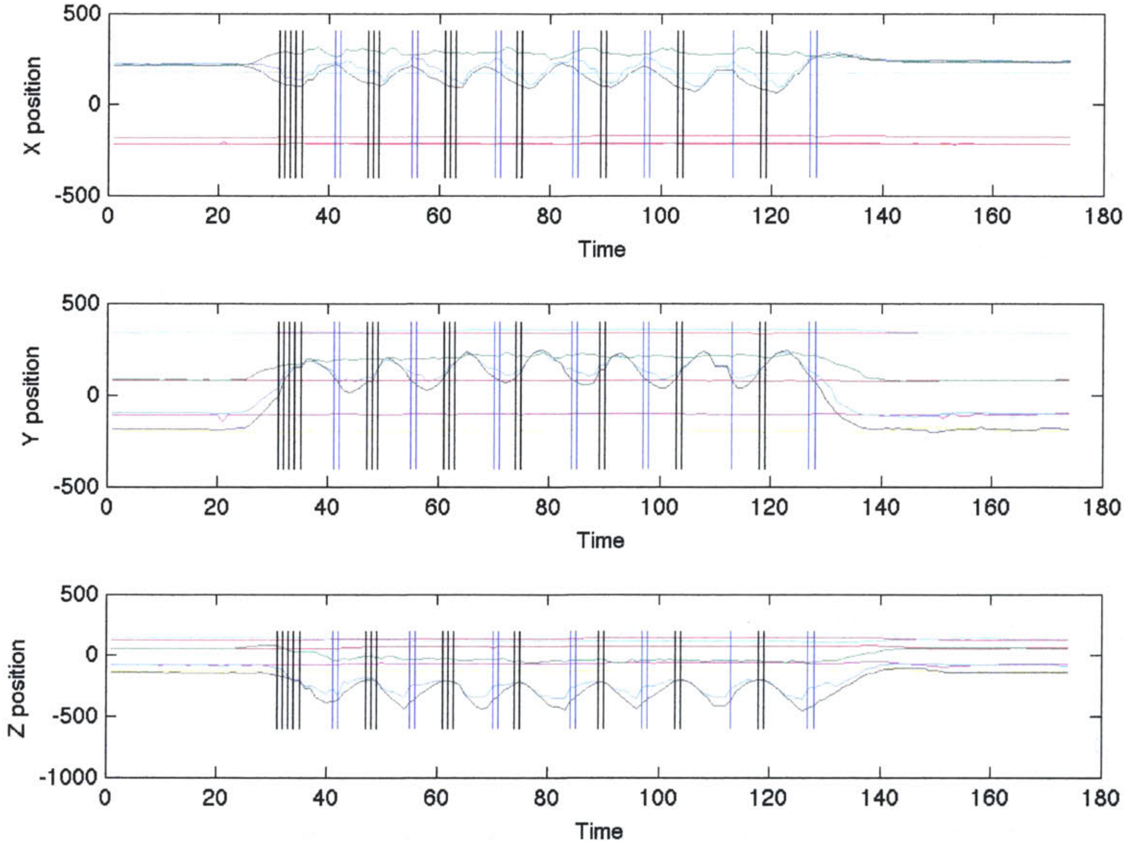
Results

4.1 Training the Models

4.1.1 SVM

The SVM training labels were determined manually by stepping through the skeleton sequences for each gesture training example and noting the frames at which the different poses occur. Any skeleton pose that was not one of the three *close*, *far*, or *pointing*, poses was labeled as a non-pose example. After the labels have been initially identified with the skeletons, they can be verified by overlaying them on plots of the gestures as in Figure 4-1. The *close* and *far* poses are indicated as vertical lines on the plots. *Close* poses are indicated in black and *far* poses are indicated in blue.

Viewing these SVM training labels in this way shows that the labels do make intuitive sense. In Figure 4-1a, the *close* poses are located at the local minima in the x direction, local maxima in the z direction, and halfway between the minima and maxima in the y direction as the hand joints are increasing. This corresponds to the hand when it is closest to the body center (x direction), farthest from the Kinect sensor (z direction), and halfway between the cycle peaks as the hand is rising (y) direction. In each cycle, the poses can occur a number of times depending on the speed of the gesture. If the cycle is acted out more slowly, the hand spends more time



(a) Forward scroll

Figure 4-1: Graphical representation of gestures, plotting joint positions over time with respect to the body center

in the zone associated with the different poses.

4.1.2 HMM

The HMM training procedure is much simpler. The training examples are collected with the Trainer module as described in Section 3.2.1. Once the different training examples have been collected, they are manually checked for quality using the gesture plots. There are a number of issues that may be wrong with the training gestures. Two gestures may not have been spliced properly and would be combined as a 'single gesture' in the eyes of the HMM. There may also be undesired joint motions within a single gesture. This could be caused by either unintentional bodily motion, or malfunction in the Kinect's skeleton output. Once these gestures are checked for quality, corrected and have had features extracted, they are simply run through the HMM training algorithms with the appropriate parameters.

Parameter Determination

To find the optimal HMMs, the number of hidden states Q and Gaussian mixtures M in each state were determined empirically by enumerating through cross-validation of the different combinations of Q and M. Given a set of training gestures, they were divided into training validation sets and each trained on the given combination of input parameters. When varying both $Q = \{1, 2, 3, 4, 5\}$ and $M = \{1, 2, 3, 4\}$, the validation sets yielded results that were completely independent of M, the number of mixtures in each hidden state. Based on this independence, $M = 1$ was chosen as the final value for the HMMs for simplicity. Even though larger values of M would produce the same results, they would add unnecessary computation time to the HMM training phase.

The search for the optimal number of states Q was performed over a wider range of values to make trends more visible. The values used in this exhaustive cross-validation search were $M = 1$ and $Q = [2 : 28]$. The results are are shown in Figure 4-2. This graph shows that there are diminishing returns for models with more than 2 states, and a significant drop in validation accuracy for models with more than 4 states. As the number of states continues to grow, the effects of overfitting begin to appear. At $Q = 22$, the validation accuracies have reached 100% again, indicating that the models have grown overly complex and have overfit the training data.

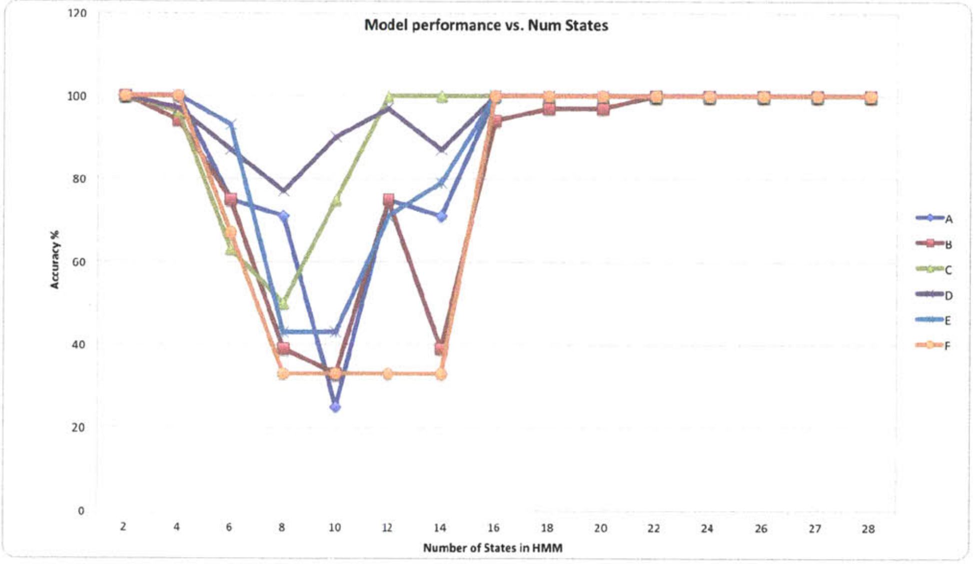


Figure 4-2: Cross-validation of test sets varying number of states in HMM. Diminishing returns for more than 2 states and over fitting for more than 10 states.

4.2 Classification Accuracies

4.2.1 SVM

Pose Classification Accuracies

The SVM was tested by running full test gestures through the system, taking the SVM labels, plotting the labels along the gesture plots and manually counting the number of times the labels occur in the correct locations. Figure 4-3 shows an example of this SVM label plot. The black lines show *close* poses and the blue lines show *far* poses. The plot in Figure 4-3 also shows that each pose is often labeled multiple times in quick succession as the hand passes through a range of positions that match the SVM's model. A label is counted as "correct" if the SVM identifies a viable pose accurately at least once each time the hand passes through the appropriate location. For instance, in this backward scrolling gesture, the first two "*far* poses" are missed completely while the rest are considered accurately classified. Compared to the manual SVM labels in Figure 4-1, the labels are generally in the right locations, but have significantly more classified poses. This result is more desirable than having too few or wrongly identified poses. These results are still useful, especially with the

Gesturing Hand	All poses	Close Poses	Far Poses	Pointing Poses
Right	84.7325%	97.9923%	73.5907%	100%
Left	83.0785%	99.0708%	67.9056%	100%
Both	83.8780%	98.5523%	70.0635%	100%

Table 4.1: Overall SVM classification accuracies for both left and right hands

non-pose skeletons identified in between the *close* and *far* poses.

Table 4.1 shows the general results of the SVM classifications with both the left and right hands. The overall classification rate of correct poses labels is 83.8780%, identifying *pointing* poses with 100% accuracy, *close* poses with 98.5523% accuracy, and *far* poses with 70.0635% accuracy. While the SVM accuracy is quite high for identifying *pointing* and *close* poses, the *far* poses have room for improvement. These results are explored further by looking at the different ways in which the gestures can be performed.

There are three different general locations where the forward, backward, and scroll gestures can be performed: hands near the waist gesturing in front of the body (low), hands in front of the chest gesturing in front of the body (mid), and hands near the waist gesturing to the side of the body (side). By dividing the gestures up in this manner, the hands pass through slightly different regions of space yielding differing results. Table 4.2 shows that there is no one region that performs the best for every pose. The low region has the highest accuracy (100%) for identifying *close* poses, but also the lowest accuracy for *far* poses (52%). The mid region has the highest overall accuracy with classification rates >90%.

Gesture Segmentation Accuracies

The results so far have shown the accuracies of each pose in isolation. However the application of the SVM is to be able to identify splice points in addition to segmenting the scroll gestures. This is evaluated by determining how many of the individual 'cycles' in each scroll gesture are identified at least once. Remember that these cycles may be defined by both enclosing *close* and *far* poses, or either *close* or *far* pose. As

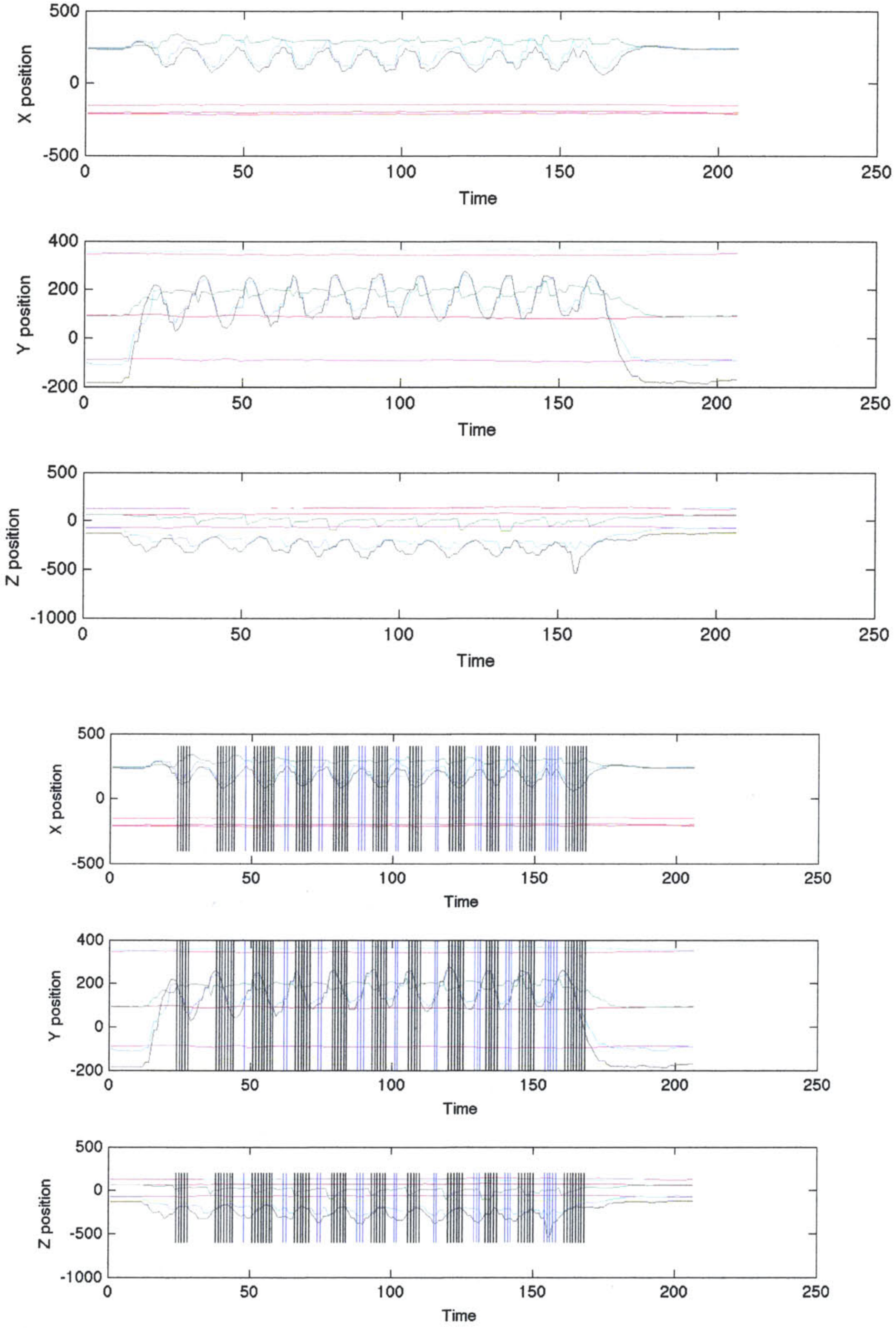


Figure 4-3: Plot of joints over time in a backward scrolling gesture, with and without plotted SVM labels. *Close* poses are indicated with black lines, and *far* poses indicated with blue lines.

Region	Hand	All Poses	Close Poses	Far Poses
Low	Right	76.9802%	100%	53.9604%
Low	Left	66.4179%	100%	32.8358%
Low	Both	71.3058%	100%	52.6117%
Mid	Right	95.6204%	99.7567%	91.7274%
Mid	Left	92.0872%	95.6422%	92.8899%
Mid	Both	93.8017%	97.6387%	92.3259%
Side	Right	78.4226%	96.7262%	72.0238%
Side	Left	85.7692%	98.9744%	73.3333%
Side	Both	82.3691%	97.9339%	72.7273%

Table 4.2: SVM classification accuracies for different regions

a repetitive cycle, a single pose for each cycle is sufficient enough to divide the overall scroll gesture into appropriately sized segments to send to the HMMs. Table 4.3 shows these cycle classification results.

Despite the shortcomings in being able to identify every *close* and *far* pose in every gesture, the SVM is able to identify 99.92% of the cycles missing only 2 out of 2455 cycles. Even when the gestures are performed in the low region and the correct pose classification rate is as low as 71%, the cycle identification rate is at 100%.

4.2.2 HMM

Forward vs. Backward

With much of the gesture abstraction done with the SVMs, the HMMs only have the single job of differentiating the direction of the forward vs. backward gestures. Note that the *pointing* gesture is excluded from the HMM discussion in this section because it is classified solely by the SVM with 100% accuracy. In the differentiation of the forward vs. backward gestures it is effective to join the forward gestures with the forward scroll segments and the backward gestures with the backward scroll segments as the abstraction suggests. This abstraction increases the accuracy of the discrimination by a few percentage points as well as simplifies the higher level logic of the system.

Region	Hand	Direction	Both poses	Close only	Far only	Miss	Identified
Low	Right	Forward	75.5319%	24.4681%	0%	0%	100%
Low	Right	Backward	35.1852%	64.8148%	0%	0%	100%
Low	Left	Forward	51.7073%	48.2927%	0%	0%	100%
Low	Left	Backward	18.1818%	81.8182%	0%	0%	100%
Mid	Right	Forward	99.4737%	0%	8.8670%	0%	100%
Mid	Right	Backward	84.6154%	15.3846%	0	0%	100%
Mid	Left	Forward	91.1330%	0%	8.8670%	0%	100%
Mid	Left	Backward	86.2661%	13.3047%	0.4292%	0%	100%
Side	Right	Forward	88.2716%	0.6173%	19.1358%	0%	100%
Side	Right	Backward	33.9080%	70.1149%	5.1724%	0.5747%	99.4845%
Side	Left	Forward	91.3265%	7.6531%	1.0204%	0%	100%
Side	Left	Backward	53.6082%	45.3608%	0.5155%	0.5155%	99.4845%
All	Both	Forward	82.5175%	14.0734%	4.5455%	0%	100%
All	Both	Backward	52.1739%	48.2075%	0.1526%	0.1526%	99.8474%
All	Both	Both	66.3136%	32.3014%	4.2926%	0.0815%	99.9186%

Table 4.3: Forward and Backward scroll segmentation accuracies. Shows the percentage of individual scroll cycles that are identified by *close*, *far*, both bounding poses, and the cycles that are missed completely.

	Right Hand	Left Hand
Unabstracted gestures	69.61%	75.69%
Abstracted gestures	69.81%	78.23%
Percentage change	+0.29%	+3.36%

Table 4.4: HMM classification accuracies using abstracted and unabstracted gesture models.

There are two HMMs for each gesture, one model for each hand. These models classify the correct hand performing the gesture with 100% accuracy. Without abstracting the gestures into the two gesture classes and leaving the four gestures separate, the HMM classification accuracies are 69.61% and 75.69% for right and left hands respectively. With the abstraction and combining the gestures in the HMM, the classifications are 69.81% and 79.23% for right and left hands. This does not significantly affect the classification rate of the right hand models, but increases the left hand classification rate by 3.36%. These results are shown in Table 4.4.

Model (Right)	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
Combined	70%	61%	78%	65%	85%	54%
Start-end	75%	66%	84%	70%	89%	61%
End-end	69%	63%	82%	72%	90%	56%
Start-start	75%	64%	77%	64%	82%	59%

Model (Left)	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
Combined	80%	75%	67%	66%	82%	81%
Start-end	82%	77%	75%	69%	82%	78%
End-end	80%	75%	77%	78%	95%	89%
Start-start	81%	77%	63%	63%	78%	81%

Table 4.5: Comparison of HMMs that use different combinations of scroll segments.

Scroll Segments

With the gesture abstraction and the unreliability of the SVM giving the HMMs gestures that all start and end at the same locations, the HMM uses all segment combinations from the training gestures in the HMMs. This means that for any given training scrolling gesture with n cycles there are $3n$ segments used to train the HMM. If every cycle i in the gesture has a "start" s_i and an "end" pose e_i, the HMM uses the cycle segments defined by (s_i, e_i), (s_i, s_{i+1}), and (e_i, e_{i+1}). The models where these segment combinations are separated into their own HMMs have slightly higher classification accuracies than the HMM that combines all the combinations. Table 4.5 shows the classification results of these different models on a number of different test sets.

Chapter 5

Conclusion

5.1 Evaluation of Results

With final classification accuracies ranging between 70-80% for the forward-backward gestures distinctions and 100% for the pointing gestures, the system works fairly reliably. In addition, the real-time counting of cycles in the scroll gestures provides increased levels of accuracy for scroll classification. By determining the overall scroll direction with a running tally of every cycle, the system can prevent any misclassified cycles in real time. This prevents undesired behavior in the controlled PowerPoint presentation and increases the overall effectiveness of the PowerPoint controller. Even in real time, the misclassified scroll cycles are just ignored. While this requires the user to keep scrolling to compensate for potentially missed, it is less disruptive for no action to occur than for the wrong actions to be performed. The system is designed so in the cases where a scroll gesture needs to be precise, the user can use speech commands to gain fine control over the system.

This gesture recognition system that combines SVMs and HMMs allows for comfortable gesture recognition and control of a PowerPoint presentation. The gestures that can be identified by gesture recognition systems are often contrived and unnatural. While the range of identifiable gestures are relatively limited, they are performed naturally in the context of giving a PowerPoint presentation.

5.2 Limitations and Areas for Improvements

Speech Integration One of the limitations in this system is in the speech integration. The speech commands must be spoken to completion before the gestures are performed. This creates an unnatural disconnect between the speech and the gestures. Another possible solution would be to prevent the system from reacting to a gesture until the speech has been fully processed. However, there are a number of reasons that would make this approach undesirable. The primary reason is the trade-off between the importance of speech versus the speed of the system's reaction to performed gestures. For this particular system where speech is not required and gestures are often performed independently, it seemed more important to have a reactive system than one that waits for speech to finish. This leads into the second problem, which shows why the system cannot begin to react to the performed gestures as it finishes processing speech. Some of the relevant commands, such as "move ahead 3 slides", are dependent on the current presentation. Beginning to navigate slides before that particular speech command is understood could result in confusing effects. One case is if there are more than 3 scroll cycles, the slides would progress past the destination slide then suddenly skip back to it. Even though the end result is the same, this behavior would be confusing to an audience that is trying to follow the slides.

Gesture Segmentation The procedure for segmenting the gestures also has room for improvement. With the wide range of poses that are classified by the SVM, the start and ends of each gesture or gesture segment are still ambiguous, as shown in Figure 4-3. The current solution is to identify the gesture "start" as the first start pose and the gesture "end" as the last end pose. This is a reasonable solution for the singular forward and backward gestures, as it ensures that the entire gesture is captured. This is slightly different in the case of the scroll gestures. The fact that the *close* and *far* poses are interchangeable as the "starts" and "ends" of each scroll segment means that there is quite a bit of variability between the segments sent to the HMM for classification during testing. One approach could be to train a more accurate SVM, which could narrow down the range of identified poses and solve this problem. However, this more restrictive SVM could miss more poses completely which would have the opposite effect on the system. Another approach could be to develop a more sophisticated procedure for determining the gesture "start" and "end" from the pool of possible poses.

Generalizing the System More general improvements could be to further generalize the system with more gestures and actions. With the expansion of different gestures and controllable actions, it would be useful incorporate an API that offers programmatic control over PowerPoint presentations. There is also room for training the system on different users. As every person will perform the gestures differently, it would be useful to collect more data from other people to generalize the models. User tests would also be useful to identify further potential deficiencies in the overall recognition system.

Appendix A

Plots of all gestures

This appendix shows the different plots of the five different gestures (forward, backward, forward scroll, backward scroll, and pointing) each performed in two different ways. The first four gestures are performed in front of the body and out to the side of the body. The pointing gestures are performed with the arm extending out to the side as well as across the body. The resting state for the arms is when they are hanging straight down at the side of the body.

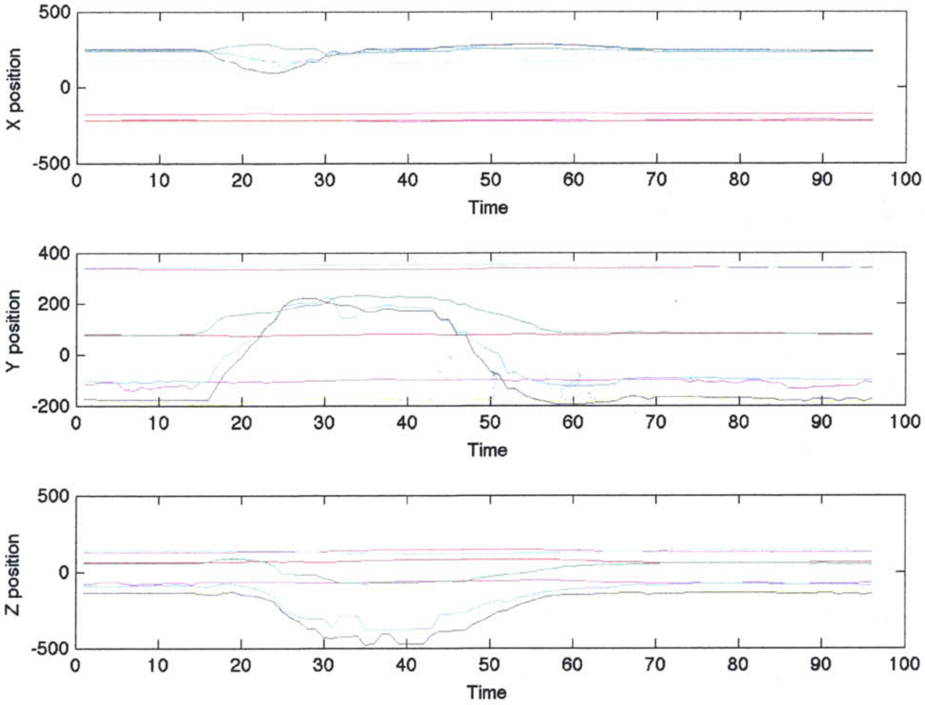


Figure A-1: Forward gesture: right hand in front of body

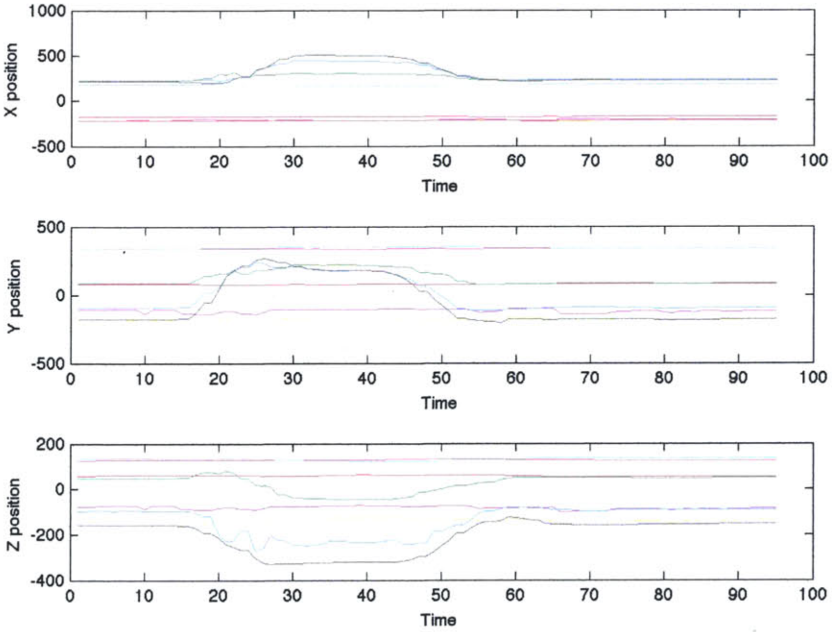


Figure A-2: Forward gesture: right hand to side of body

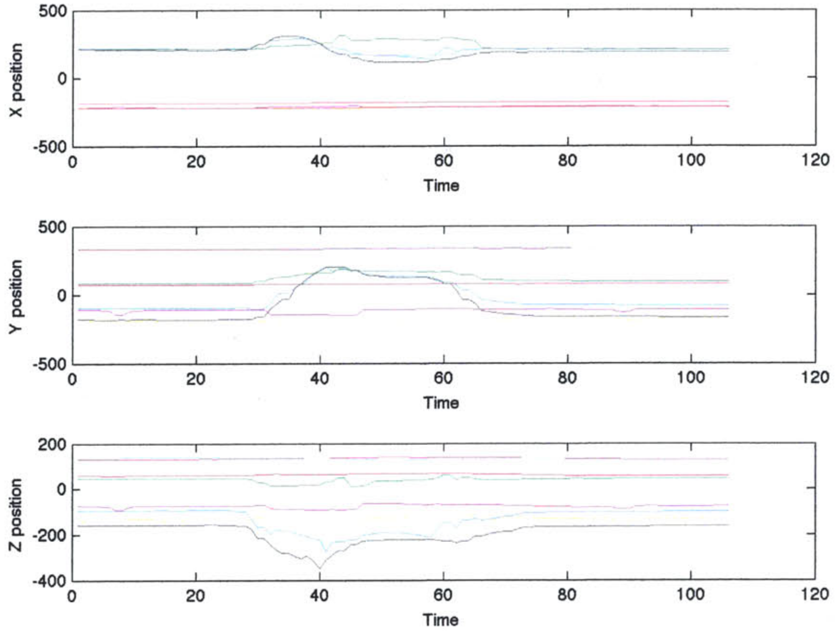


Figure A-3: Backward gesture: right hand in front of body

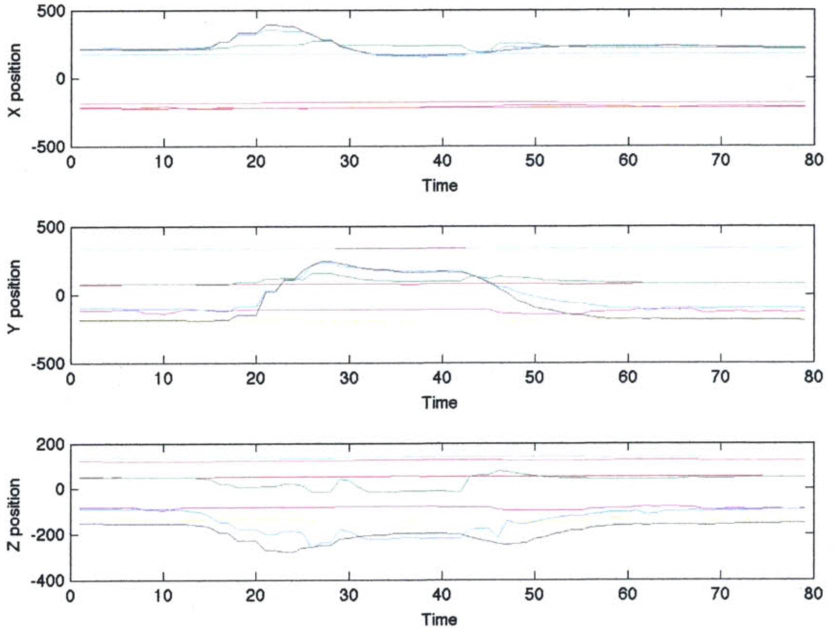


Figure A-4: Backward gesture: right hand to side of body

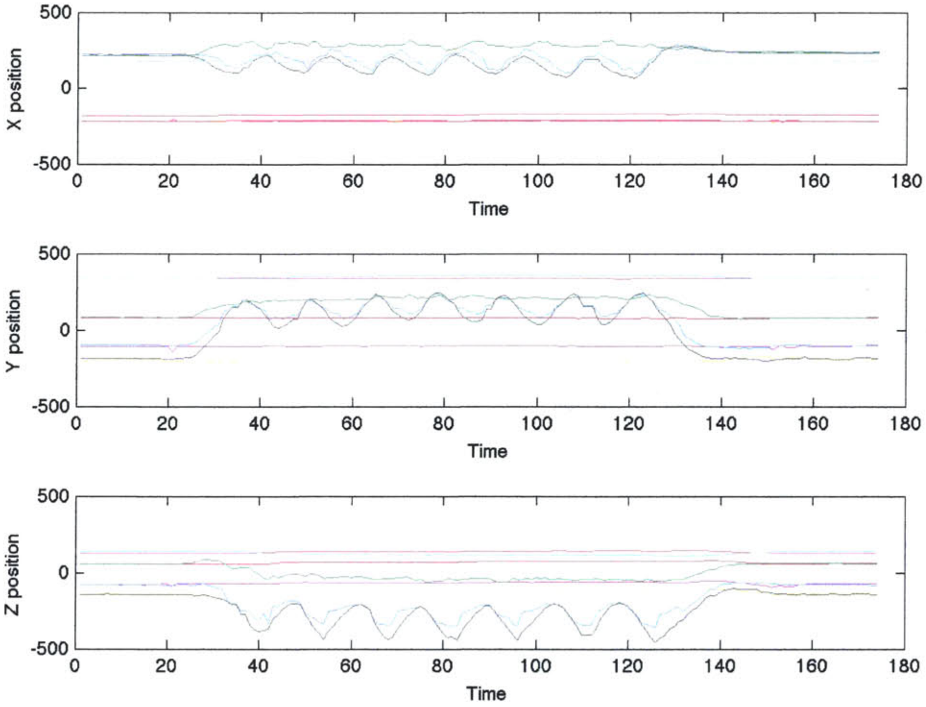


Figure A-5: Forward scroll: right hand in front of body

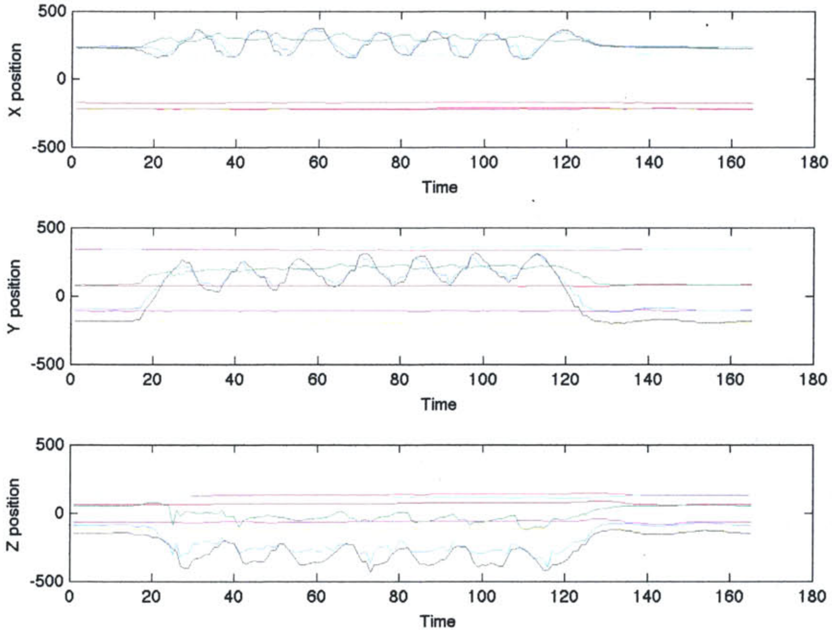


Figure A-6: Forward scroll: right hand to side of body

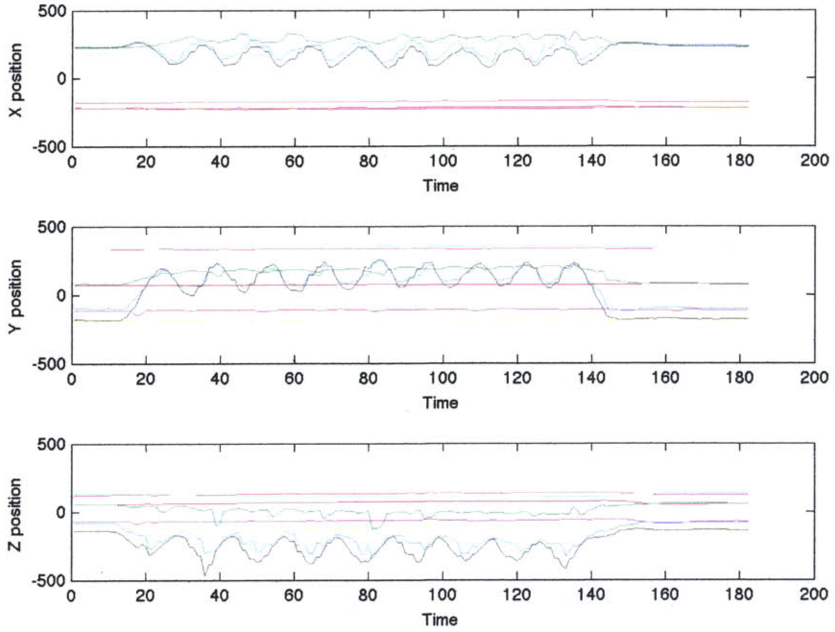


Figure A-7: Backward scroll: right hand in front of body

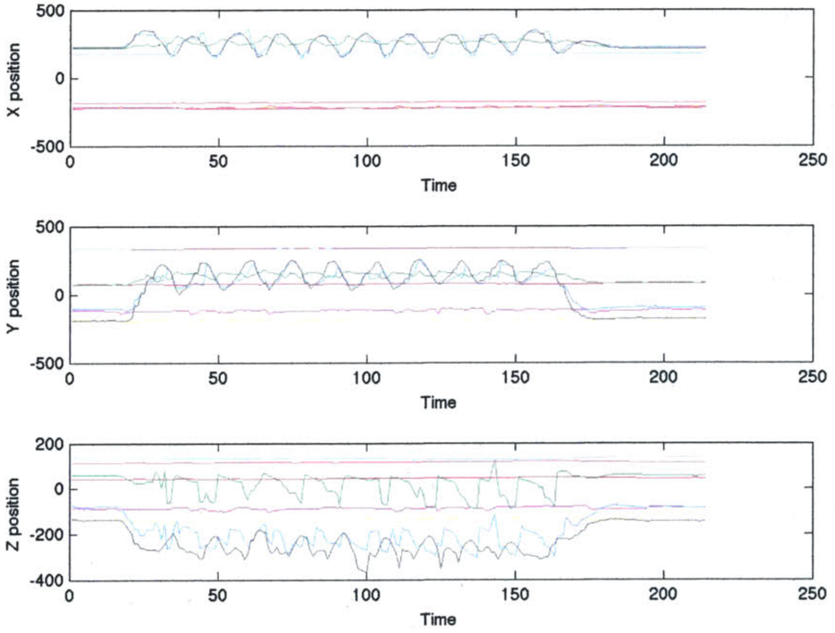


Figure A-8: Backward scroll: right hand to side of body

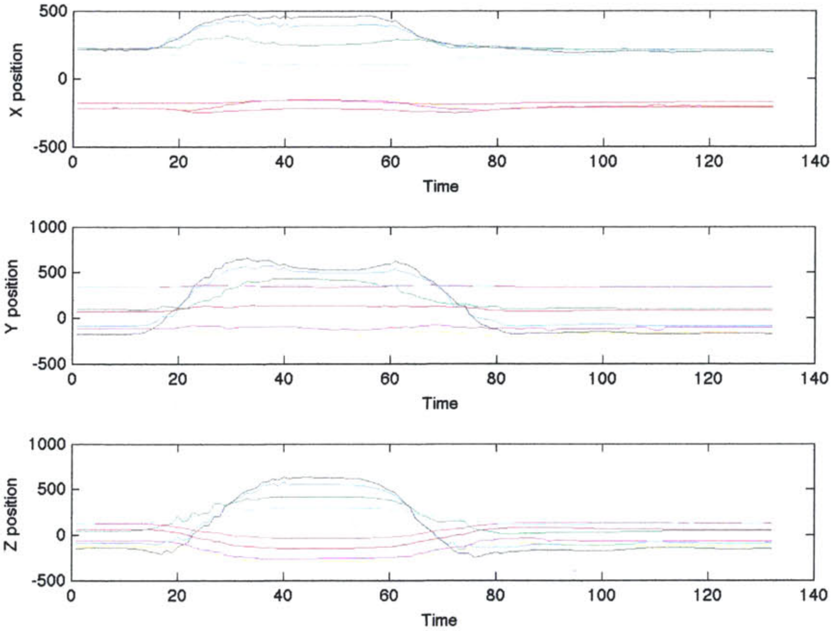


Figure A-9: Pointing to right with right hand

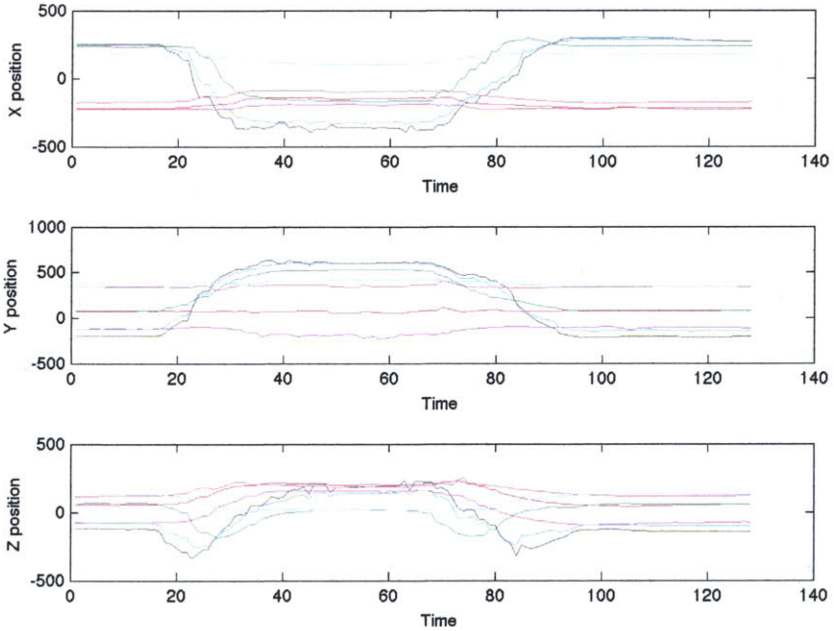


Figure A-10: Pointing to left with right hand crossing in front of the body

Appendix B

JSGF Speech Grammar

```
<ones>= one | two | three | four | five | six | seven | eight | nine;
<teens>= ten | eleven | twelve | thirteen | fourteen | fifteen | sixteen | seventeen |
eighteen | nineteen;
<tens>= twenty | thirty | forty | fifty | sixty | seventy | eighty | ninety;

<singleAction>= going | moving;
<skipAction>= (skipping | skip);
<direction>= forward | ahead | on | back | backwards;
<number>= <ones>| <teens>| <tens>| (<tens><ones>);
<location>= beginning | start | (first slide) | end | (last slide);

<singleCommand>= <singleAction>| <direction>;
<multipleCommands>= (<singleAction>| <skipAction>) <direction><number>slides;
<automaticSkip>= skipping | <direction>;
<skipCommand>= <skipAction>to ((slide <number>) | location);

public <command>= <singleCommand>| <multipleCommand>| <automaticSkip>|
<skipCommand>;
```

Bibliography

[1] C.M. Bishop. *Pattern Recognition and Machine Learning.* 2007, 2007.

[2] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings Fifth Annual Workshop on Computational Learning Theory (COLT)*, 1992.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[4] C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 1995.

[5] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1977.

[6] T. Huang, R.C. Weng, and Chih-Jen Lin. Generalized bradley-terry models and multi-class probability estimates. *Journal of Machine Learning Research 7*, 2006.

[7] F. Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, 1997.

[8] Jerdak, 2011. http://www.seethroughskin.com/blog/?p=1159.

[9] C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

[10] G.J. McLachlan and T. Krishnan. *The EM Algorithm and its Extensions.* Wiley, 1997.

[11] J.R Rabiner and B.H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[12] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceddings of the IEEE*, 1989.

[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. CVPR, IEEE, 2011.

[14] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A flexible open source framework for speech recognition. Technical report, Carnegie Mellon University, 2004.