University of Alberta

Gesture Recognition using Hidden Markov Models, Dynamic Time Warping, and Geometric Template Matching

by

Garett Hunter

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Abstract

Gesture recognition is useful in many applications, including human-computer interaction, automated sign language recognition, medical applications, and many more. The main focus of this thesis is to improve the isolated gesture recognition accuracy of Hidden Markov Models (HMMs) and to provide a comparison to Dynamic Time Warping and Geometric Template Matching. These techniques are compared with single-path gestures, such as the gestures created by one hand, and different coding techniques for multi-path gestures, such as gestures created with both arms. Subsequence duration structures of user-defined gestures are important for accurate recognition, and modelling these structures has been shown to increase the recognition accuracies of HMMs. It is hypothesized that Vector Quantization is responsible for the superior performance of HMMs under specific circumstances. Other contributions of this thesis include an analysis of user-defined full-body 3D gestures, several modification to increase the accuracy of HMM models, and a multi-path template matcher.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A gesture is a combination of movements or poses of body parts that conveys a meaning. The meaning can be symbolic, such as a wave to say hello, or it can be used to express direct control of an object. What can be considered a gesture might be surprising. For example, speech involves parts of the body responsible for vocalizing words and can be considered a gesture, but this type of gesture is not considered in this thesis. The type of gestures that are considered are easy to observe and involve parts of the upper body. An example of such a gesture is a person drawing a shape such as a check mark with the hand.

Gesture Recognition is the act of recognizing the class of a gesture. The gesture may be left unclassified if the recognition algorithm cannot decide on any class. For example, one can easily recognize if a person is performing a thumbs up gesture, a police traffic control stop gesture, or a hello gesture by waving. Furthermore, the types of gestures that are dealt with in this thesis are motion gestures that form paths through space. Pose gesture, like a thumbs up gesture, are not considered.

A common method for designing gesture interfaces is to design a gesture based on a reasonable guess, or heuristic. This heuristic design process is not guaranteed to produce intuitive gestures that are natural, easy to use, and easy to remember. Another solution is to allow users to create their own gestures, and use the most common gestures produced by participants for each task. These gestures have the advantage of being natural, easy to learn, and easy to use. This can create an intuitive user interfaces for novices, but there is no guarantee that interface is optimal for expert use.

Gestures have the advantage of being useful for spatial manipulations. For example, in surgery, gestures have been tested for minimal invasive laparoscopic surgery where the surgeon uses head gestures to manipulate spatial parameters of the laparascope [45]. The laparascope is a camera that allows to the surgeon to directly view the area underneath the skin where the surgeon is operating. This allows direct control of the camera while the surgeon can simultaneously use both hands for manipulating the hand instruments. Gestures are also common in video games, with dedicated platforms such as the Nintendo

Wii, Playstation Move, and Microsoft Kinnect, along with other similar devices. Other areas such as robotics, sign language recognition, and emergency response and control have also been using gesture recognition to improve performance [28, 45, 55, 7, 26, 66, 61].

The algorithms used in gesture recognition use template matching or parametric methods. Template matching compares unknown gestures to a template with a known class. Parametric classifiers involve learning a set of parameters from training data where an unknown gesture is given a score based on the learned parameters and classified based on this score. Each method has strengths and weaknesses that are discussed in the later chapters.

In order for a gesture to be stored in a computer, a suitable representation must be developed. The specific representation used in this thesis are positions that make up the path of each gesture. These positions can be considered raw data, and other representations are extracted from this data. For example, once the positions are known, acceleration and velocity can be calculated. These different representations are the features of the gestures, and each feature has specific properties such as rotation invariance, or scale invariance.

This thesis makes many new contributions to gesture recognition.

- First, a study is performed for User-Defined gestures that analyzes the how much participants agree on gestures for certain tasks for full-body gestures. A new agreement score is added that has many benefits over the current agreement score.

- Second, an analysis of the advantages and disadvantages of two feature coding conditions are provided and their affects on recognition accuracies of various methods.

- Third, a set of polar and relative distance features are contributed that allow for scale, rotation, and translation invariance for any single path gestures with known start and end times.

- Fourth, many of the geometric template matching methods failed to develop a uniformly scale invariant scoring technique for Euclidean distance, and in this work we provide a solution to this problem by minimizing the Mean Square Error between two gestures.

- Fifth, a synchronized multi-path gesture recognizer MD-Protractor was developed.

- Sixth, a GUI for Hidden Markov Model initialization was developed to provide good starting points before learning.

- Seventh, a Vector Quantization emission probability smoothing technique was developed to allow higher recognition accuracies for low amounts of training data.

- Eighth, a Hidden Markov Model with duration modelling was developed to improve recognition accuracies.

- Finally, a comparison between Dynamic Time Warping, MD-Protractor, HMMs, and HMMs with duration modelling is provided for recognition accuracy as a function of training size under the two feature coding conditions.

Chapter 2

User-Defined Natural 3D Gestures

User defined gestures involve the idea of designing gesture interfaces by allowing the user to create gestures instead of the designer creating them. One of the arguments for allowing users to define gestures is the possibility of using natural and intuitive interactions. The goal of intuitive interaction assumes an easier form of interaction, and an improved performance output for users of the system. However, not all users agree on the what is the most suitable gesture for a given task. A general consensus was reached for a small subset of tasks, despite the large degrees of freedom available to participants to define gestures. For any subset of tasks that have an agreement, we should extract a set of user-defined gestures for them.

2.1 Introduction

Gestural interfaces are often difficult to use because the user does not know how to execute the gestures without explicit instruction. Although many designers have created gesture sets that aim to be intuitive, many of them suffer from a large gulf of execution, where a user's mental model does not match that of the designers. In this chapter, we explore natural gestures from the perspective of the end user [48]. We asked participants to create gestures for 41 common 3D user interface tasks. The results demonstrate that, while there is less agreement when using full-body gestures in 3D space than is found on 2D surfaces, users did tend to reach consensus on a subset of common 3D gestures.

In recent years, gestural interaction has become very popular due to advances in sensing technologies, new application areas, and advancements in commodity hardware. Interfaces relying on gestural interaction can often be difficult to use for novices, as there are no intrinsic elements that guide the user to the desired action. This is especially true for free space 3D gestures, where the active interaction space is often not clear (e.g., there is no screen to touch). Several recent application areas have shown an immense potential for 3D gestural interfaces. Some examples of gesture-based 3D user interfaces (3DUIs) are surgical systems, where the users must not touch the screen to preserve sterility, controller-free video games or media interfaces, where the user does not have access to a physical input device [72], and public displays where interaction time is short and casual. While these applications are very diverse, they often share common tasks, such as navigating a virtual environment, modifying objects, and performing certain high-level commands.

In order for users to interact with 3D gesture systems, gestures must either be explicitly taught to the user using demonstration videos or on-screen guides, or must be simple enough for users to guess. In many cases, teaching the user is impractical due to time constraints or the negative impact on the user's experience. In these situations, a set of natural 3D gestures would provide efficient and enjoyable interactions.

One approach to defining intuitive gestures is for the designer to specify the required movement [44, 19]. However, designers often disagree on the correct choice of gesture. Previous work has shown that there can be substantial differences in the gestures users create when asked to define them [74]. Further, many of the standard or natural gestures used for 2D interaction do not have direct extensions into 3D space. The ability to use the full body and to perform movements that are not constrained to a particular device allows a much richer set of potential body movements. Additionally, many actions common in 3D interaction tasks (navigation, object manipulation, etc) are not typically found in 2D interfaces.

This study analyzes how users define free-space 3D gestures for common 3DUI tasks. Users were asked to define gestures for a set of 41 actions covering a variety of abstract commands, navigation and manipulation tasks.

2.2 Related Work

Prior work in gestural 3D interaction has often focussed on using a defined set of gestures [19, 46]. These gestures were explicitly taught to users of the system and allowed basic interactions. While this approach can be used in some scenarios, complex interaction requires a large set of gestures, which will either require substantial training or must be designed to be guessable by the user. Alternatively, users can define their own gestures for each action [46], but this can be overly cumbersome for novice users.

Designing guessable gestures has been examined in the context of 2D surface computing [74] as well as mobile interaction [58]. These studies use paradigms where users are asked to design and describe gestures for a specific interaction. In Wobbrock et al.'s study, participants were asked to design gestures for use on an interactive surface. They found that users could generally agree on a subset of the gestures, but more abstract commands had greater variance among the gestures. Ruiz et al. obtained results similar to those of Wobbrock et al., but reported a slightly higher overall level of agreement [58]. Both of these studies found that actions with inverse behaviours often had mirrored gestures. Norton et al. analyzed natural 3D interaction for controlling a video game [49], but their results are limited to navigational tasks and game-specific functions (translation, combat, climb pole, etc). While these studies provide some insight on natural user interaction, it is not clear that the results generalize to 3D user interfaces in general.

2.3 Experiment

2.3.1 Participants

Twenty-two subjects from the university community participated in the study (16 female, M=24, SD = 5.2 years). By self-report, 19 participants were right handed, one was left handed, and two were ambidextrous. Participants were paid $10.

2.3.2 Actions

Forty-one actions were derived from 5 categories of typical 3DUI interactions. While this is not a complete set of actions, it does provide substantial coverage for a variety of tasks common to most 3DUIs. Navigation and object manipulation, for example, are of central importance to 3D modelling, video games, and many other tasks.

WIMP (Windows, Icons, Menus and Pointers) actions represent abstract commands usually found in desktop interfaces, but can be extended to many 3D interaction tasks. Media actions are used in the playback of audio, video and other multimedia content. Navigation actions involve moving the users viewpoint in 3D space, and object manipulations involve selecting and transforming objects in 3D space. Finally, multi-object manipulations dealt with manipulations that occur on two or more objects.

The complete set of actions are shown in Table 2.1 and consist of WIMP actions (Next, Previous, Zoom In, Zoom Out, Menu Access, Cancel, Confirm), Media actions (Play, Pause, Stop), Navigation actions (Move Forward, Backwards, Left, Right, Up, Down, Turn Left, Turn Right, Accelerate, Decelerate), Object Manipulations (Rotations around X, Y, Z axis in Positive and Negative direction: RXP, RXN, RYP, RYN, RZP, RZN; Scaling Larger Uniformly: SLU, Scaling Smaller Uniformly: SSU, Scaling Larger along X-axis: SLX, Scaling Smaller along X-axis: SSX, Translations Up, Down, Left, Right, Forward, Backwards), Object Selection and De-Selection, and Multi-Object Manipulation (Group, Ungroup).

2.3.3 Procedure

Before the start of the experiment, participants were told that they would be designing a set of gestures to interact with a computer. They were told that each gesture should be defined by a body movement or pose. They were told they could use any parts of their body to define the gesture, but that they were not able to re-use gestures. A speak-aloud

Category	Action
WIMP	Next Previous Zoom In Zoom Out Menu Access Cancel Confirm
Media	Play Pause Stop
Navigation	Move Forward Move Backwards Move Left Move Right Move Up Move Down Turn Left Turn Right Accelerate Decelerate
Object Manipulation	Rotate X-axis Positively (RXP) Rotate X-axis Negatively (RXN) Rotate Y-axis Positively (RYP) Rotate Y-axis Negatively (RYN) Rotate Z-axis Positively (RZP) Rotate Z-axis Negatively (RZN) Scale Larger, Uniformly (SLU) Scale Smaller, Uniformly (SSU) Scale Larger, X-axis (SLX) Scale Smaller, X-axis (SSX) Translate Up Translate Down Translate Left Translate Right Translate Forward Translate Backwards Select De-Select
Multi-Object Manip	Group Ungroup

Table 2.1: Table of actions

protocol was used, where by participants verbally indicated the start and end of a gesture and then described the rationale for the choice of gesture. Furthermore, participants were wore a motion-capture jacket to track their upper body movements, which might have primed them to use more of their upper-body.

Participants stood approximately two meters from a projection screen, on which they viewed the actions. Figure 2.1 shows a single frame of video stimuli of a rotating cube rotat-

Rotate Y-axis Negatively

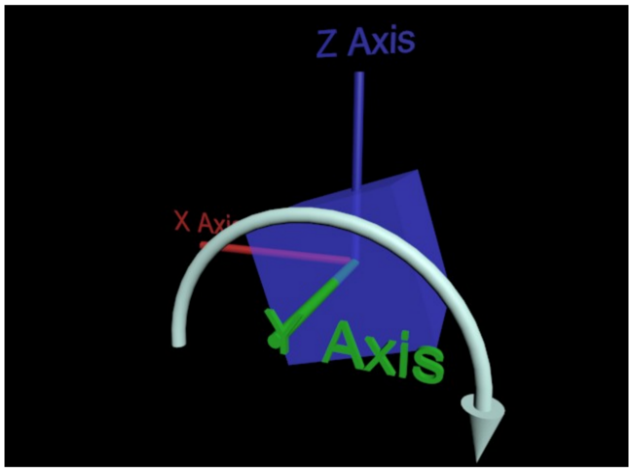


Figure 2.1: Sample stimulus of an action shown to participants.

ing around the y-axis. Each of the actions was displayed as the name of the command along with a target scene on which the command operated. In some instances, the target scene was animated to clarify the desired operation (e.g., direction of movement). Participants were permitted to view all 41 actions before defining any action-gesture pairing, and they were able to define the pairings in any order they chose. As in other studies, the participants did not receive any feedback. Their movements were recorded by video for later analysis.

After completing each action-gesture pairing, participants responded to the following statements on a 7- point Likert scale "The gesture I picked is a good match for its intended purpose"; "The gesture I picked is easy to perform". These questions are the same questions used in[74] and the responses range from 1 as "strongly disagree" to 7 as "strongly agree". For example, 1 = "strongly disagree", 2 = "disagree", 3 = "somewhat disagree", 4 = "neither agree or disagree", 5 = "somewhat agree", 6 = "agree", and 7 = "strongly agree".

2.4 Results

2.4.1 Manual Gesture Classification

In addition to the 19 point upper body tracking suit, one camera was used to capture the rest of the information of the full body. To this extent gestures were classified manually by by the experimenter and reported as equal. This allowed us to partition the gestures for each action into sets of identical gestures. Also it allows us to count the gestures so that we can compute the agreement and certainty scores of each action.

2.4.2 Agreement Scores

Equation 2.1 defines the agreement score A_a (used by Wobbrock et al. [74] and Ruiz et al. [58]), which is used to quantify the consensus for each action.

$$A_a = \sum_{P_i \subseteq P_a} \left(\frac{|P_i|}{|P_a|} \right)^2 , \tag{2.1}$$

P_a refers to the set of all gestures created for action a, and P_i refers to each subset of identical gestures identified in P_a.

Figure 2.2 shows the agreement scores for each action. With 22 participants and 41 actions, 902 gestures were recorded. Two gestures were considered identical if they had identical trajectories and poses. Two trajectories were considered identical if their direction of motion over time was the same. Two poses were considered identical if their shape was the same. The overall agreement found was 0.103, which is much lower than Wobbrock et al., who found an overall agreement of 0.32 with one handed gestures, and 0.28 with two handed gestures.

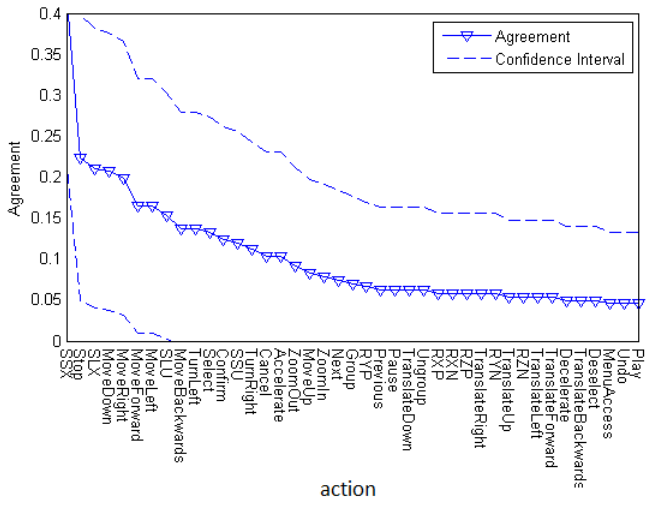


Figure 2.2: Agreement Scores for each action over 22 participants. Confidence intervals were calculated based on the binomial distribution and normal approximation method with a significance level of $\alpha = .05$.

The actions "Scale Smaller X-axis", "Scale Larger X-axis", "Move Down", "Move Right", "Move Left", "Move Forward", and "Stop" were the only ones with a confidence interval that did not include zero. Furthermore, the correlation between participants' Goodness rating and Agreements was found to be significant, r = 0.468, $\rho = 0.002$, suggesting that Goodness ratings are a good indicator of the consensus between users on gestures chosen for an action.

2.4.3 Certainty

Wobbrick et al [74] used agreement scores in their analysis to capture how much people agreed upon the same gestures. We use a different notation to clearly show each concept and to introduce another measurement of agreement using entropy [63]. Let G be an abstract matrix with size $N \times P$ where N is the number of actions and P is the number of participants in the experiment. Each element g_{ap} of G is a gesture where a is the index to the action, and p is the index to the person. More importantly is the set representation of identical gestures across participants for a given row of G. Lets denote $I(g_a)$ to be a set of identical gestures of the a'th row g_a of G. In addition for any $i \in I(g_a)$ we create a function $n(i)$ that gives us the number of people who designed the same gesture i for action a. Thus we have $\sum_{i \in I(g_a)} n(i) = P$ which implies $\sum_{i \in I(g_a)} \frac{n(i)}{P} = 1$. We can think of $\frac{n(i)}{P}$ as the probability that gesture i for action a is designed by a group of people.

Finally we can define the agreement score of Wobbrick et al [74] as $A_a = \frac{1}{P} \sum_{i \in I(g_a)} n(i) \frac{n(i)}{P}$ and $A = \frac{1}{N} \sum_{a=1}^{N} A_a$. Wobbrick et al [73] states that when all participants design unique gestures the value of $A_a = \frac{1}{P}$ since every participant "trivially agrees" with themselves. However, agreement occurs between two or more people instead of only one. If all participants design unique gestures the agreement score should be zero.

$$H_a = - \sum_{i \in I(g_a)} \frac{n(i)}{P} \log_b \frac{n(i)}{P} \tag{2.2}$$

$$C_a = 1 - \frac{H_a}{\log_b(P)} = \frac{1}{P} \sum_{i \in I(g_a)} n(i) \log_p n(i) \tag{2.3}$$

H_a represents the uncertainty and C_a represents the certainty that particular gestures should be used. Similarly, as with Agreement A_a, a represents the action. Notice that by dividing H_a by it's largest possible value $\log_b(P)$ we do not need to concern ourselves with the bits b. Borrowing from Wobbrock's example [73] of two hypothetical distributions of gesture counts with the first being 15/20 and 5/20 and the second being 15/20, 3/20, and 2/20. Then we will calculate $H_a^1 = -\left(\frac{15}{20} \log_b\left(\frac{15}{20}\right) + \frac{5}{20} \log_b\left(\frac{5}{20}\right)\right)$ for the first distribution, and $H_a^2 = -\left(\frac{15}{20} \log_b\left(\frac{15}{20}\right) + \frac{3}{20} \log_b\left(\frac{3}{20}\right) + \frac{2}{20} \log_b\left(\frac{2}{20}\right)\right)$. for the second distribution. However, both values of H_a depend on the bit b and it is better to focus on $H_a / \log_b(P)$. The first value gives $H_a^1 / \log_b(20) = 0.1877$ and the second value $H_a^2 / \log_b(20)$ gives 0.2439. When we compute C_a for both distributions we get 0.8123 and 0.7561 respectively, and we are more certain of the outcome in the first case than in the second.

The difference between these two scores lies in the multiplication term of $\frac{n(i)}{P}$ and $\log_P n(i)$. Note that $1 \leq n(i) \leq P$ which means $\frac{1}{P} \leq \frac{n(i)}{P} \leq 1$ and $0 \leq \log_P n(i) \leq 1$. The second score provides a better normalized factor especially when the number of participants are low and also says there is no agreement when no one else designs the same gesture.

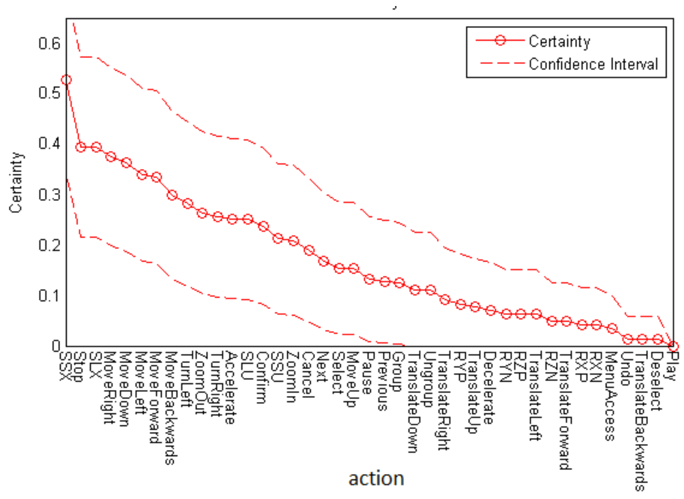


Figure 2.3: Certainty Scores

With 29 participants, and 41 actions, 1189 gestures were recorded. For each action a the set $I(g_a)$ was created based on identical gestures from g_a. Gestures were considered identical if they had identical trajectories and poses.

Confidence intervals on the agreement and certainty scores are calculated based on the binomial distribution and the normal approximation method. Each participants experiment is independent of any other and on a given action a participant will either agree with another or not. A standard significance level of 0.05 is used. A gesture should only be extracted for a given action if the lower bound on the confidence interval is above zero.

However both scores have different behaviours. Interestingly Certainty presents an overall higher degree of consensus compared to Agreement at 0.171 and 0.097 respectively, and is more strongly correlated with the Goodness rating at r = 0.547, p < 0.001 than Agreement is at r=0.468, p = 0.002. Firstly, this suggests for both scores, participants subjective Goodness ratings are a good indicator of how much consensus will exist on that action. Secondly Certainty is better at allowing more gestures to be extracted since it has smaller confidence intervals. Certainty keeps the same properties required by Agreement.

Certainty allows one to claim a larger subset of tasks that have a consensus shows a larger subset of tasks that do have a consensus than the agreement of Figure 2.2. This allows gesture interface designers to extract more user-defined gestures in their interface.

2.4.4 Use of Body Parts for Gestures

We analyzed the extent to which participants used different body parts for the gestures corresponding to each action. The proportion of body parts used for each gesture is shown in Figure 2.4. It is not surprising that a high proportion of participants used hand gestures.

This pattern is, however, not consistent across all actions.

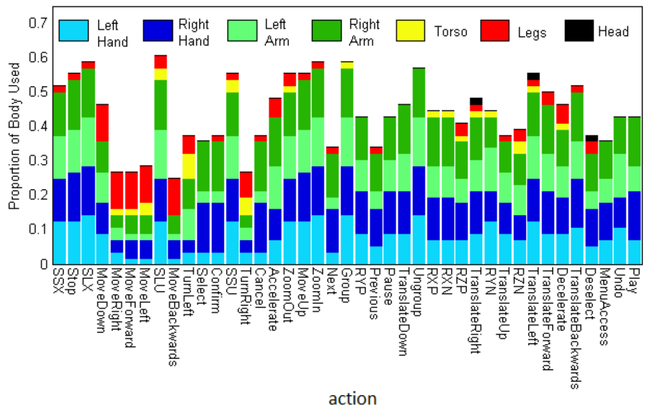


Figure 2.4: Proportion of each body part used for each action.

For navigation gestures, participants used mostly the torso and legs. For the actions "Move Forward", "Move Backwards", "Move Left", and "Move Right", the gestures included the legs with a higher proportion. With these gestures, participants tended to walk towards the location that corresponded to the direction of movement. The torso was used primarily in the turning actions, which participants would respond to by twisting the torso to the left or right. Gestures for navigation should thus involve tracking the orientation and position of the entire body over time. Additionally, for all other actions in our list it appears that hand gestures are the most natural form of interaction. Thus, there is strong support for hand gesture interaction for free space gestural interfaces.

2.4.5 Gesture Types

Each gesture was analyzed along two dimensions, pose and motion, that defined 5 gesture types. A Pure Motion gesture is defined by motion alone with pose having no meaning. For example, if a user rotates the arm in a circular pattern but the hand pose is not used systematically, then it is considered a pure motion gesture A Pure Pose gesture is defined by pose alone and motion has no meaning, as, for example, when the hand is extended with the palm flat as in a traffic stop signal. In a Pose-Motion gesture, a pose is held constant and the motion has meaning. A Dynamic Pose Motion gesture includes multiple poses and the motion has meaning. Finally, a Dynamic Pose gesture consists of multiple poses, but the motion has no meaning. For example, the user can point the thumb up and then down.

A pose can refer to the entire body or parts of the body, i.e., if all that mattered for a gesture was one body part, then the rest of the body was ignored. For example, when a participant was waving the right hand, pose and movement of the other body parts were ignored. It was clear from the participants' descriptions what elements of a gesture were considered important. The proportion of gesture types for each of the actions is shown in Figure 2.5.

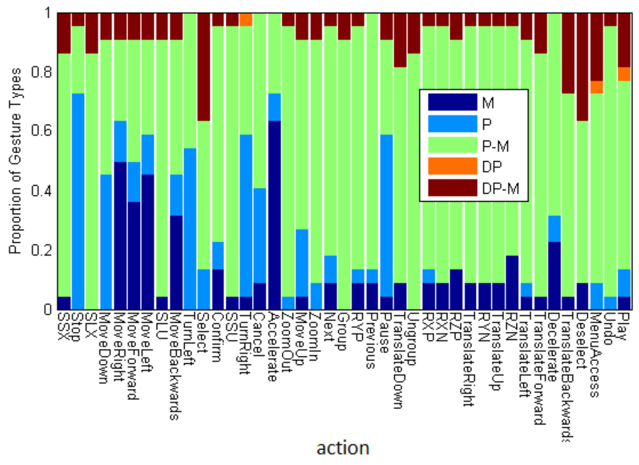


Figure 2.5: Proportion of Gesture Types for each action. P-M = Pose-Motion, P = Pure Pose, M = Pure Motion, DP = Dynamic Pose, DP-M = Dynamic Pose-Motion

Motion was a common characteristic for most gestures which can be seen in both Figure 2.6 and 2.5. The most common type of gesture was a Pose-Motion gesture. What is most interesting to notice is that participants used the same motion type for the same action. For example, each rotation action involved many Pose-Motion gestures. Gestures for rotation had very low agreement, not because the gestures were completely different, but rather because the poses were slightly different. Surprisingly, motion was very similar across participants. The translation gestures also showed low agreement, with similar motions, but different poses. The scaling actions did not fit this pattern. This may be due to the fact that rotation and translation gestures are used in our daily lives, but scaling is used rarely. This may, however, change with the increased popularity of direct-touch interfaces such as smart phones and tablets.

The only actions with higher than average proportions of Pure Pose gestures were “Move Down”, “Stop”, “Turn Left”, “Turn Right”, and “Pause”. For “Stop”, participants often performed a “traffic stop” gesture, and “Pause” typically involved the same gesture used by “Stop”, or at least a variation of it. For “Move Down”, most participants bent their legs and lowered their bodies for a short duration. For “Turn Left” and “Turn Right”, participants often oriented their bodies to face left or right so that while they were facing in the corresponding direction the view also kept turning in the corresponding direction. Actions with the highest proportions of Pure Motion included navigation, where participants walked in the direction of movement for “Move Forward”, “Move Backwards”, “Move Left”, and “Move Right”. For “Acceleration”, participants often ran forward briefly to indicate speeding up their motion.

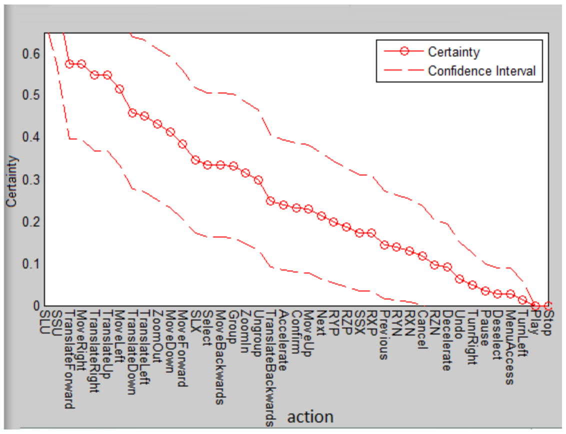


Figure 2.6: Motion Certainty.

2.5 Discussion

Our results show a much lower level of agreement between full body 3D gestures than was found in previous studies on 2D gestures for interactive surfaces and mobile devices. This is likely due in part to the larger degree of freedom in generating full body 3D gestures than is the case in 2D interaction. A similar effect can be seen in Wobbrock et al.'s work [74] that showed lower agreement when allowing two-handed gestures rather than one-handed ones. This raises important and unique challenges for the design of effective 3DUI gestures. In order to obtain a more robust, agreeable set of gestures, it may be beneficial to restrict the degrees of freedom available to the end user, i.e., to force the user to specify gestures using only the motion of the hands or the legs to reduce the possible set of gestures they are able to create.

However, permitting a large number of degrees of freedom does have benefits. For instance, 3D modelling work often involves simultaneous view changes along with object transformations. By allowing input from the arms as well as the legs, users would be able to perform these actions simultaneously, and naturally. In our study, many subjects performed navigation using the legs to walk forward or back, while many manipulations were done using the hands. Combined, they form a natural and efficient way to perform simple 3D modelling tasks.

Scaling and Zooming actions elicited similar gestures, involving a common centre in front of the body with either increasing the distance or decreasing the distance between the hands. These types of gestures appeared to be 3D extensions to the typical pinch-to-zoom gestures found in touch interfaces. Common navigation gestures involved walking, standing in different locations, twisting the torso, running faster, jumping, and bending down. These gestures show that users tended to embody the viewpoint and use typical body based locomotion to achieve movement.

Gestures tended to be more similar in their motion than in their pose. For example, most participants' translation gestures involved moving the position of the hand in the intended direction, but participants did not agree on the hand pose. Whether both hands were used was also quite variable between participants for the same action. This fact may be leveraged by gesture recognizers, as more weight can be given to motion parameters, and less to pose.

As with previous studies, abstract actions showed less agreement. For instance, the "Play" gesture showed almost no agreement between participants. Some participants attempted to draw a typical triangular play icon, while others danced or mimicked putting headphones on. Similarly, for "Undo", some participants would try to draw a counter-clockwise circle, others performed movements to the left or backwards. In contrast, more concrete actions such as Movement and Selection showed ore agreement, as they allow for more direct analogues to the physical world.

Symmetric gestures were also quite common among similar types of actions. For instance, with the Movement gestures participants had the same underlying gesture for all Movement gestures with only the direction of the gesture varying between them. A similar result occurred with the Rotation gestures, but there was high variability in how the underlying rotation action was represented, leading to lower agreement than was found for the Movement gestures.

2.6 Conclusions

We have presented a study of user defined, full-body 3D gestures to determine whether or not a set of natural gestures could be derived and used for a gestural 3D user interface. We find that there is less agreement using free-space 3D gestures than has previously been found with user-defined gestures on interactive surfaces or mobile devices. However, users did still tend to use similar gestures for an important subset of gestures, namely, navigation and some manipulation actions. Future work may examine the effects of restricting degrees of freedom on the types of gestures users create, and their subjective opinions regarding those gestures. This may allow for a larger set of gestures to be agreed upon, and make gestural 3D user interfaces more usable.

2.7 New Contributions

We make new contributions to user-defined studies with the extension into three dimensions, full-body free-form gestures, and a new agreement score using entropy. Many of the gestures elicited have an origin from everyday use. Navigation tasks, where the user moves in a virtual environment, elicited gestures that involved full body movement such as walking, or changing position of the entire body. Scaling gestures were very similar to the gestures used on the iPad. The new agreement score called certainty is better correlated with the Goodness ratings, it allows zero valued agreement scores to be possible without extra transformations, it can be interpreted using uncertainty, and allows more user-defined gestures to be extracted. One can interpret the normalized uncertainty $H_a / \log_b(P)$ of action a and of the P participants for how uncertain the participants are about which gesture should be extracted. By subtracting the normalized uncertainty from one, one can interpret this value as how certain that the P participants are that some gesture should be extracted for action a.

Chapter 3

Gesture Recognition and Previous Work

Gesture recognition has a wide range of applications that can be very useful for intuitive spatial manipulations. Gestures can also be used for more abstract commands such as indicating what task the user wants to perform. Applications of gesture interfaces have been implemented in robotics [28], medicine [45], Emergency Response and Control [55], Sign Language Recognition [7], and Entertainment [26, 66, 61].

3.1 Robotics and Medicine

Gestural interfaces have begun to be used in robotic assistants for health care. In a multimodal environment hand gestures can be used to give commmands to a robotic assistant [57]. Kawarazaki et al. [28] designed a gestural interface for a robot to instruct in picking up objects and bringing them to a person. Assistive aids, such as robots, can be very useful for people who require assistance in their daily living. For example, Zhu et al. [78] recommended that pet like robots should be commanded using multi-modal interaction combining gestural and speech input. In the domain of military and search-and-rescue missions, Shah et al. [62] argued that issuing commands to robots needs to be as natural as issuing commands to humans.

Nishikawa et al. [45] used face gestures to aid in laparoscopic surgery, by connecting face movements to laparascope movements. As they only had one camera with restricted frontal parallel viewing of the face, they developed a state transition gestures to switch between panning and zooming functions. Panning and zooming functions were controlled through tilting the head, but in different system states. They argued that direct parameter control of the laparascope through gestures is more precise than voice commands given to a robot or a human.

Wachus et al. [69] used hand gestures for image navigation of MRI image databases during surgery. By allowing control through free hand gestures, surgeons decrease the risk of infection through better sterilization. By detecting how the surgeon moves their hand in and out of a particular region in camera space they decided whether the next or previous MRI image should be displayed.

Graetzel et al. [18] used hand gestures as a method to control a mouse in the operating room. The 2D position of the hands mapped to the 2D controls of the mouse, and a rectangular prism was chosen as the interaction space for gestures, with a motionless hand indicating the start of a gesture for easy gesture spotting.

Kuno et al. [34] used hand gestures to remotely guide a wheelchair, and used face movements to control the direction of the wheelchair while riding. Additionally, non-registered gestures with intentional meaning were inferred based on their repetitions.

3.2 Entertainment

Electronic gaming is now seeing the rise of gesture recognition incorporated into their interface. The Microsoft Kinect, Playstation Move, Nintendo Wii, and other devices use gestures as a way to interact with games. A benefit to this type of interaction is the idea of incorporating exercise by performing gestures repeatedly during game-play for a prolonged period of time. People who are entertained by video games find little difficulty playing for long periods of time. Despite the fact that video games are excellent sources of entertainment, they can be very useful to society by promoting exercise, rehabilitation, or some other form of simulation implicitly [26, 66, 61].

Gestures in video games need to react very fast with little time delay, and therefore need efficient recognition algorithms. Touch-screen interaction such as surface computing, and mobile interactions use gestures for many of their tasks such as scrolling, selecting, and zooming. Examples of touch-screen devices are the iPad, the iPhone, and the Microsoft Surface, which have been successfully integrated into modern society.

3.3 Sign Language Recognition

Sign Language Recognition incorporated into video games can be a very useful teaching tool. Brashear et al. [7] developed a video game to aid in the Sign Language learning process for deaf children.

3.4 Emergency Response and Control

Emergency response and control operate in a collaborative environment that involves dialogue and parameter manipulations. Rauschert et al. [55] discuss the advantages incorporated gestures into emergency response situations by using speech and gestures to handle certain tasks more efficiently. They state that “speech provides an effective and direct way of expressing actions, pronouns and abstract relations, it fails when spatial relations or locations have to be specified.... and therefore gestures can provide an effective second modality that is more suitable for expressing spatial relations..." [55, p. 119].

3.5 Previous Work on Gesture Recognition

The techniques for gesture recognition vary from geometric template matching, warping the time dimension, boundary tests for classification, statistical models, and many others that are not mentioned here. These techniques suffer from different problems of the gesture recognition domain. We review recent developments for geometric template matchers and their rotation, scale, and translation invariant benefits, as well as Dynamic Time Warping, Support Vector Machines, and Hidden Markov Models.

3.5.1 $1 Recognizer

Wobbrock et al. [75] developed a geometric template matcher for gesture recognition in two dimensions. Uni-stroke gestures are gesture with one stroke in either two or three dimensions. They wanted to develop a method that is simple to implement and is accurate. The raw data gathered are the positions of each gesture in time, with start and end points being defined by the user. First, the distance from start to end is calculated and is used to divide the gesture into segments of equal lengths. This division re-samples the gesture and ensures that every gesture has the same number of data points for comparison. Next they define the "indicative angle" which is the angle between the first point and the center of the gesture, and they rotate the gesture so that its indicative angle becomes zero. This is a heuristic approach to guess the best angular alignment between gestures. To account for translation differences, each gesture is translated to the origin based on their centres. Finally, rotation invariance is accomplished by searching over a range of rotation angles that allows for a gesture to be optimally angularly aligned to a template. To compare gestures, the sum of the Euclidean distances between corresponding re-sampled points are calculated and then divided by the number of points to get an initial dissimilarity measurement between gestures. Limitations of the $1 recognizer include the inability to model features other than position, the inability to handle so called "1-dimensional" (e.g. a straight line) gestures that when scaled to a two dimensional region such as a square, become distorted, the inability to distinguish gestures with different aspect ratios, and the slow search for the best angular alignment.

3.5.2 Protractor

Li [39] also developed a geometric template matcher for gesture recognition in two dimensions. Similar to the $1 recognizer, the start and end points must be given. Li first re-sampled and translated every gesture as Wobbrock et al. done [75]. The difference in their methods results from their similarity measurements, where Li used the angle of the cosine distance and Wobbrock et al. used the Euclidean distance between gestures. The angle of the cosine distance between gestures is uniformly scale invariant, and therefore Li's approach does not suffer from "1-dimensional" gesture scale distortions. Interestingly, the angle of the cosine distance can be manipulated in such a way that a closed form solution to rotation invariance can be found. This results in an improvement in speed over the \$1 Recognizer for achieving rotation invariance.

3.5.3 \$3 Recognizer

Kratz et al. [32] developed a geometric template matcher for gesture recognition with uni-stroke gestures in three dimensions by extending Wobbrock et al.'s [75] approach. They claim that their method is for acceleration data, yet they pre-process the accelerations into positions before they begin recognition. As Wobbrock et al. [75] and Li [39], they re-sample and translate each gesture to the origin. Their next step is to rotate each gesture, so that its angle with the axis between the centre and first point of the gesture with a common axis they define becomes zero. This gives them a head start for their search to find the best rotation angles. After their initial rotation each gesture is scaled to a cube and then the search for the best Euler rotation angles begins. Like Wobbrock et al. [75], at each of the rotation angles the gesture is compared to the template using the average Mean Square Error. Problems with this method include two types of scaling issues, slow rotation search, and an inability to model features other than position. The first type of scaling issue is the inability to deal with aspect ratio variance, when spherically shaped and elliptically shaped gestures are transformed into the same cube without considering their shape difference in each dimension. The second type of scaling issue also stems from the scaling of each gesture to a cube, which causes one-dimensional gestures and even two-dimensional gestures to become distorted based on random noise in the other dimensions.

3.5.4 3D Protractor

Kratz et al. [33] developed a geometric template matching technique for gesture recognition of uni-stroke gesture in three dimensions. As in the other approaches [75, 32, 39], each gesture is re-sampled, translated to the origin, and scaled to a cube. Their contribution was to use Horn's [21] closed form solution for finding the best rotation axis and angle between two sets of multiple points in three dimensions. They used Quaternions to find the minimum sum of squares error between the input gesture and template gesture that is based upon finding the optimal rotation axis and rotation angle to rotate the input gesture. After the best rotation axis and angle is applied to the input gesture it is compared to the template gesture through the average Euclidean distance between them. One of the advantages to this approach is an efficiency in finding the best rotation axis and angle. Another advantage is the ability to choose a threshold of how much a gesture can be rotated around its optimal rotation axis, and this allows for partial rotation invariance if the user chooses so. However this method also suffers from the same scaling issues as the previous geometric template matching approaches [75, 32, 39]. Like Protractor by Li [39], Kratz et al. [32] also developed a closed form solution to rotation invariance, but instead of using the angle of the cosine distance they used the average Euclidean distance between gestures causing their solution to not have the advantages of Li's [39] approach to uniform scale invariance.

3.5.5 Point Cloud Recognizers

Iterative Closest Point (ICP) is an algorithm to find the best rotation and translation alignment between two sets of points the model shape and the data shape. The idea is to calculate rotation and translation parameters to transform the data shape to the model shape without knowing the time ordering of points. Hence the name Point Clouds. Initialization is important since ICP is only guaranteed to find a local minimum. Vatavu et al. developed the $PointCloud : \$PRecognizer$ [68] for recognizing multi-stroke gestures as a cloud of unordered points. The assumption here is the bag of uni-strokes make an overall shape that can be compared to other clouds of points by finding the best matching cloud. Vatavu's method used the Hungarian algorithm [68], which is of cubic order, but it does outperform all other methods. The other issue is that it does not follow the $-Family paradigm of implementation simplicity, and fast geometric template matching. Vatavu solves this by using heuristic matching, such as greedy solutions, to obtain similarity measurements between clouds. Their experiment show promising results for their heuristic methods.

3.5.6 Multi-Stroke Recognition and Synchronized Multi-Stroke Recognition

As discussed above Protractor, 3D-Protractor, $1-Recognizer, and $3-Recognizer each handle single uni-stroke gestures, but a solution to multiple uni-stroke gestures has not been discussed yet. An example of a synchronized multi-stroke gesture is a ballet dance where different parts of the body move in synchrony, and an example of a multi-stroke gesture is writing the symbol "x" using two strokes with one occurring after the other. Multi-strokes are multiple uni-stroke gestures that occur sequentially in time, with all the strokes together forming an overall shape. The stroke order is not important, causing an exploding number of possible stroke orders. Different algorithms [68, 1, 2] have different ways to cope with this problem. Synchronized Multi-stroke Recognition refers to the idea of multiple strokes occurring at the same time interval that can be tracked through different limbs or fingers.

3.5.7 Dynamic Time Warping

Dynamic Time Warping (DTW) is a technique for aligning two gestures of different lengths. It is not restricted to gestures, but is also applicable to other problems such as Speech Recognition since DTW is a general feature alignment technique. The idea is to optimally align the two gestures from an exponential number of possible alignments. The alignments must make sense and this is accomplished by restricting each alignment in certain ways such as ensuring that the starts and ends match up with each gesture. The reason DTW warps time is that it allows time to be warped for each alignment so that it appears that each gesture has the same number of points. DTW is not restricted to positional features, but can be applied to other features such as velocity, acceleration, and others. DTW does not inherently handle rotation, scale, and translation invariance like the previous approaches [75, 32, 39] do, but instead the features themselves must be invariant in one or more of these ways. Further details are discussed in Section 5.1

3.5.8 Hidden Markov Models

Hidden Markov Models (HMM) for gesture recognition are statistical classifiers. The idea of HMMs is to use different probability distributions to model the feature patterns for different temporal parts of each gesture, i.e. every gesture can be divided into different segments that have a specific type of distribution of features for each segment. HMMs model each class of gestures by defining specific segments of gestures, each responsible for the characteristics of that portion and only the gestures that follow these characteristics at each segment are calculated to be most similar. Otherwise the gestures that do not follow these characteristics will be least similar. The usual situation is that every class of gestures has its own HMM, and an unknown gesture is compared to each HMM of each class, and the HMM with the highest similarity score is the class the unknown gesture should belong to.

HMMs require large training data, have a large number of unknown parameters, and have difficulties with the durations of each segment. The requirement for large training data is a result of HMMs being statistical classifiers and the need to approximate the underlying probability distributions accurately. They do not properly handle durations of each segment, allowing for a large number of false positives to occur. More precisely, the length of time usually spent in each segment is actually not incorporated into the general formulation of a Hidden Markov Model. Further details are discussed in Chapter 6

3.5.9 Gesture Spotting

Gesture spotting is the process of finding the start and end times of a gesture from a large stream of data. A gesture is considered spotted when these times are found. HMMs, DTW, and geometric template matchers all suffer from the same problem of needing a threshold value for comparison to assume that a valid start and end location have been found. The threshold is usually guessed, or searched based on the recognition accuracy. DTW so far has the most efficient approach to dealing with gesture spotting, with HMMs [31, 38] coming second, and the geometric template matchers coming last. One can speculate that geometric template matchers are slower for spotting gestures due to the requirement that rotations, scaling, and translations must be applied to each candidate start and end location making them much slower but likely polynomial in complexity. Perhaps an efficient geometric template matcher can be found or created that is just as fast as DTW or HMM under the special case of ignoring rotation invariance.

3.5.10 Applicability of Motion Gestures

Many gestures are a combination of pose and motion. Motion gestures can be used to signal state changes in a system. For example, a left right motion could signal a state change from zooming to panning in laparoscopic surgery. If the pose of the hand is infeasible to be calculated then only motion gestures can be used to signal state changes. Sign Language communication can use motion to signal a words. Robot human interaction may involve commands issued through motion gestures. Almost any application that involves recurring motion patterns with single or multiple paths can benefit from the knowledge in this thesis. For example, automatic segmentation of motion gestures in evaluations of laparoscopic surgery skill is an applicable area. Motion is very large part of the universe and there may be many problems that can be formulated in the same way that motion gesture recognition problems are formulated. Therefore motion gestures should be taken seriously.

Chapter 4

Features and Vector Quantization

Features are attributes of raw data that describe characteristics over time. There are two types of features, local and global. Local features describe the characteristics at each point in time and global features describe characteristics over an interval of time. Velocity is an example of a local feature, whereas the average velocity is an example of a global feature. Properties of features include translation invariance, scale invariance, and rotation invariance Without local features, it is difficult to capture the changes associated with gestures.

4.1 Local Features

Let the matrices **a** and **b** be the same size and represent two different gestures. The rows represent time, and the columns represent features. These matrices will be used throughout this chapter. For example, positional features for one gesture could be represented with the first column of the matrix as the x-dimension, the second column as the y-dimension, and the third column as the z-dimension. Then each row would be a three dimensional point at a given time with $\mathbf{a}_1$ being the first point, $\mathbf{a}_2$ the second point, and so on. More generally, each column represents a different feature.

4.1.1 Position, Velocity, Acceleration

Let $\mathbf{p}_t$ represent the position of the gesture path at time t. Then velocity and acceleration can be approximated as

$$\mathbf{v}_{t+1} = \mathbf{p}_{t+1} - \mathbf{p}_t, \quad \mathbf{a}_{t+1} = \mathbf{v}_{t+1} - \mathbf{v}_t, \tag{4.1}$$

where $\mathbf{v}_{t+1}$ defines velocity and $\mathbf{a}_{t+1}$ defines acceleration. The sampling rate is always the same and division by the sampling interval can be ignored. An alternative computation is provided in Section 4.3 on Smoothing.

4.1.2 Translation, Rotation, and Scale Invariance for Positional Features

Positional features for template matching can be rotation, scale, and translation invariant. Depending on the application, the features of gesture classes can be any combination of rotation, scale, or translation invariant.

Translation Invariance

Translation invariance can be achieved by finding the mean position of **a** or **b** and subtracting it from every row of **a** or **b**. The result is a gesture centered at the origin.

Rotation Invariance

Li [39] developed a closed-form rotation invariant solution of the cosine angle between two high dimensional vector representations of gestures. Li's method also produces scale invariance. The same rotation angle is achieved using a mean square error loss function

$$\mathbf{L}(\mathbf{a}, \mathbf{b}, \theta) = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{a_i} - \mathbf{R}(\theta)\mathbf{b_i})^T (\mathbf{a_i} - \mathbf{R}(\theta)\mathbf{b_i}), \tag{4.2}$$

where $\mathbf{a_i}$, and $\mathbf{b_i}$ are the points of each gesture. $\mathbf{R}$ is the rotation matrix parametrized by θ. Solving Equation 4.2 using $\frac{dL}{d\theta} = 0$ gives

$$\theta = a \tan \left(\frac{\sum_{i=1}^{N} a_{iy} b_{ix} - a_{ix} b_{iy}}{\sum_{i=1}^{N} a_{ix} b_{ix} + a_{iy} b_{iy}} \right). \tag{4.3}$$

Therefore, the cosine distance yields the same optimal angular alignment as the mean squared error loss function of Euclidean distance. But Euclidean distance is not scale invariant. These results are limited to 2 dimensions.

Kratz et al. [33] used Horn's [21] method for rotation invariance in three dimensions. This technique minimized a squared loss function and allowed for efficient computations of finding the optimal rotation axis and angle using Quaternions. Unfortunately, Kratz et al. [33] have many errors in their matrix formulas, but Horn's paper [21] has the correct formulas. The use of Horn's method for rotation invariance allowed for brute force angle-search to be abandoned along with other search methods, such as the Golden Ratio search [32, 75].

Scale Invariance

A gesture in three dimensions can use all three dimensions or just one dimension. A **1-dimensional gesture** is a gesture that can be rotated in such a way that we can remove the other dimensions without losing meaningful information. If we then project it back onto all three dimensions, then we can rotate it to its original orientation and gain back a high similarity with other gestures of its class. A 2-dimensional gesture is one that can be rotated in a such a way that we can remove one of its dimensions without losing meaningful information. If we then project it back onto all three dimensions, then we can rotate it to its original orientation and gain back a high similarity with other gestures of its class. Finally, a 3-dimensional gestures cannot have any of its dimensions removed without losing meaningful information.

Interestingly, a 1-dimensional gesture should not be scaled to a cube, since it adds noise in the other dimensions. Similarly a 2-dimensional gesture should not be scaled to a cube, but less noise is added compared to the 1-dimensional case. The only advantage of scaling gestures to a cube or a square is to achieve aspect ratio invariance. One solution is to scale only to the dimension of template gesture, rather than to the highest dimension of the gesture space. This is exactly Anthony et al.'s [1] suggestion. Otherwise, the average Euclidean distance will be distorted by noise. Alternatively, one can use the angle between the high dimensional vectorized representations to achieve scale invariance, as Li [39] does.

Assume that the points $\mathbf{b_i}$ are optimally angular aligned with a template gesture $\mathbf{a}$. Then one can try to scale each point using a scaling matrix $\mathbf{S}$ to scale each point of $\mathbf{b}$ and compute the Mean Square Error (MSE) as

$$\mathbf{L}(\mathbf{a}, \mathbf{b}, \mathbf{S}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{a_i} - \mathbf{S}\mathbf{b_i})^T(\mathbf{a_i} - \mathbf{S}\mathbf{b_i}). \tag{4.4}$$

The MSE can be used to solve for the individual scaling parameters of $\mathbf{S}$ by taking the partial derivative of $\mathbf{L}$ and setting it to zero as $\partial\mathbf{L}/\partial\mathbf{S} = \mathbf{0}$.

$$s_x = \frac{\sum_{i=1}^{N} a_{ix}b_{ix}}{\sum_{i=1}^{N} b_{ix}^2},\ s_y = \frac{\sum_{i=1}^{N} a_{iy}b_{iy}}{\sum_{i=1}^{N} b_{iy}^2},\ s_z = \frac{\sum_{i=1}^{N} a_{iz}b_{iz}}{\sum_{i=1}^{N} b_{iz}^2} \tag{4.5}$$

To avoid aspect ratio invariance and to achieve uniform scale invariance that preserves the gestures geometric structure, the scaling parameters in each dimension can be set to be the same value as $s = s_x = s_y = s_z$. This allows us to find the parameter s that minimizes the MSE, by taking the partial derivative and setting it to zero as $\partial\mathbf{L}/\partial\mathbf{s} = 0$. The optimal scaling parameter is found to be

$$s = \frac{\sum_{i=1}^{N} \mathbf{a}_i^T\mathbf{b}_i}{\sum_{i=1}^{N} \mathbf{b}_i^T\mathbf{b}_i}. \tag{4.6}$$

This method can be used for uniform scale invariance of any finite dimension.

Aspect ratio invariance is a specific type of scale invariance that allows different scaling coefficients along each axis. Uniform scale invariance restricts the scaling coefficients to the same across all axes. The type of scale invariance depends on the definition for class membership. Aspect ratio invariance cannot distinguish rectangles from squares, and equilateral triangles from isosceles triangles. However, aspect ratio invariance can be useful when class membership is more tolerant to scale variations from training templates.

Figure 4.1 shows a comparison for a 1-dimensional gesture of a user moving the right hand towards the right of the body. It depicts two samples of this gesture and shows different scaling methods and how it affects the dis-similarity score of Euclidean distance. The first approach scales each gesture to a cube with side lengths of 100 and is shown in Figure 4.1d. The second approach uses equation 4.6 to scale the first sample to the second sample, but this approach must be normalized for each template to avoid unreasonably high and low dis-similarity scores for bigger and smaller templates. The second sample is then scaled using the same scaling transformation applied to the first sample when it was scaled to a cube resulting in Figure 4.1e. The Euclidean distance dis-similarity score was 12.90 for Figure 4.1e and 28.06 for Figure 4.1d. The second approach gives a better dis-similarity score since it ensures proper structure of the second sample relative to the first sample.

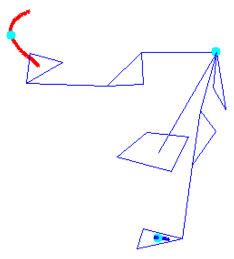
4.1.3 Curvature and Torsion

Curvature is the measure of how much a curve turns in space. Its best to imagine curvature in two dimensions. The sharper a curve turns, the higher its curvature at the turn, and the less it turns the lower the curvature. A straight line has zero curvature since it does not turn in the two dimensions of space.

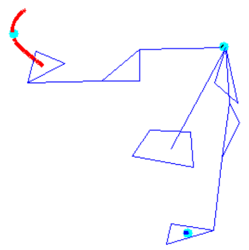
Figure 4.2 shows curvature and torsion for an “x” gesture drawn using the right hand in three-dimensional space. Curvature represents how much the tangent vector turns along the path. Torsion represents how much the principal normal vector-which has the same direction as acceleration- twists along the path. The velocity vector of Figure 4.2a, which is the non-normalized version of the tangent vector, turns the most at the points where the curvature is the highest. The acceleration vector, which is the non-normalized version of the principal normal vector, twists at the points where torsion is the highest and the lowest. The twisting of the principal normal vector twists the most before and after the two sharp turns, whereas the tangent vector turns the most at the two sharp turns.

The logarithm of curvature as shown in Figure 4.2c magnifies curvatures from 0 to 1 by transforming this range to $-\infty$ and 0. This magnification allows the feature representation to be more sensitive to smaller deviations from straight paths. A disadvantage is that perfectly straight paths are only represented by $-\infty$; however an advantage is that high curvatures from noise and stationary movements are in a more reasonable range of values. The choice between log curvature and curvature becomes important for methods that descretize features.

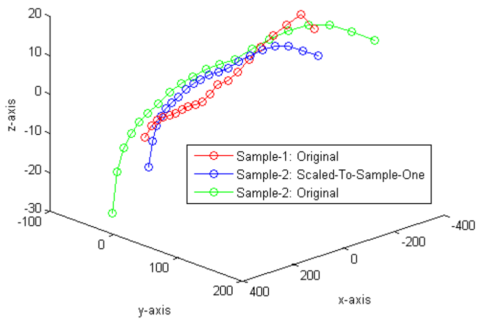
Unfortunately torsion is difficult to represent in practice due to noise. The data must be almost free of noise since noise is amplified at each higher level derivative. Fortunately



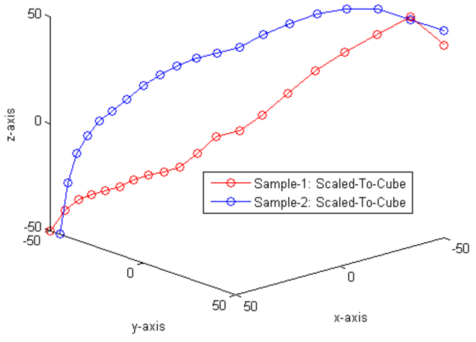
(a)



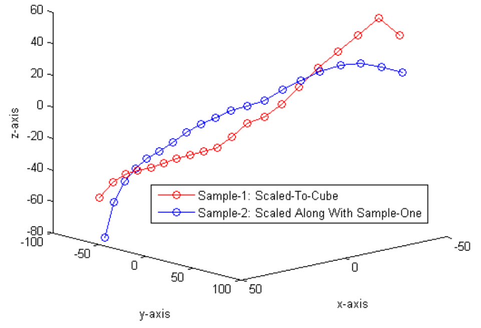
(b)



(c)



(d)



(e)

Figure 4.1: One-Dimensional Scaling Example. Each gesture is re-sampled using 20 points. The dis-similarity score between scaled samples using euclidean distance for 4.1d was 28.06 and 12.90 for 4.1e. The second approach gives a better dis-similarity score for 1-dimensional gestures. (a) Sample One: Push Right using Right Hand. (b) Sample Two: Push Right using Right Hand (c) Scaling using equation 4.6 with Sample Two as the template **a**. (d) Scaled gestures of Sample One and Two by scaling everything to a cube of 100 units in side length. (e) Another scaling was performed to the second sample from 4.1c by scaling it to a cube; However, the second sample is scaled using the same parameters applied to each dimension of the first sample when it was scaled to a cube. This process preserves the relative shape of Sample Two with respect to Sample One.

curvature is computed with low enough derivatives to be useful.

The curvature and torsion presented here is based on Thomas et al. [67]. We define a curve $\mathbf{r}(s) = x(s)\mathbf{i} + y(s)\mathbf{j} + z(s)\mathbf{k}$, which defines a set of points as a function of arc-length s. Curvature defines how much a curve turns in a plane, or how much the tangent vector $\mathbf{T}$ turns relative to a very small distance as

$$\kappa = \left\| \frac{d\mathbf{T}}{ds} \right\| = \left\| \frac{d}{ds}\frac{d\mathbf{r}}{ds} \right\| = \left\| \frac{d^2\mathbf{r}}{ds^2} \right\| . \tag{4.7}$$

Taking the norm of this value allows one to measure the magnitude and makes κ coordinate system independent. Curvature is not scale independent. The vector $\frac{d\mathbf{T}}{ds}$ is orthogonal to

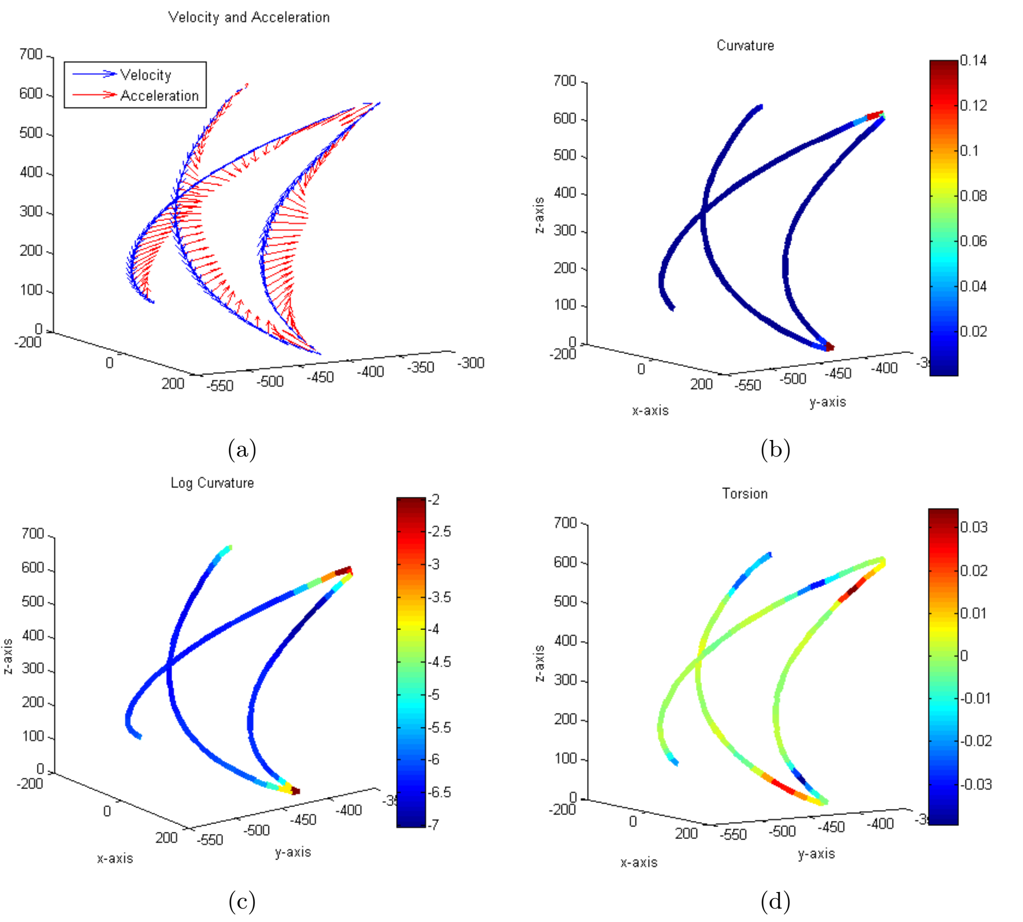


Figure 4.2: Curvature, Log-Curvature, and Torsion of an "x" gesture using the right hand. A smoothing parameter of 2 was used, a window size of 45 was used, and an arc-length re-sampling length of 20 units was used.

$\mathbf{T}$. The unit vector of $\frac{d\mathbf{T}}{ds}$ is called the Principal Normal Vector denoted as $\mathbf{N} = \frac{d\mathbf{T}/ds}{||d\mathbf{T}/ds||}$. Together they define the **Osculating plane** where curvature is defined. Curvature defines how much the curve turns within the Osculating plane.

Torsion, on the other hand, defines the amount the Osculating plane turns or twists. It can be thought as the rotation around the tangent vector $\mathbf{T}$. If $\mathbf{B} = \mathbf{T} \times \mathbf{N}$ is the vector perpendicular to the osculating plane, then $\frac{d\mathbf{B}}{ds}$ is the rate at which the Osculating plane rotates around the tangent vector $\mathbf{T}$. Thus $\frac{d\mathbf{B}}{ds}$ should be parallel to $\mathbf{N}$ meaning $\frac{d\mathbf{B}}{ds} = -\tau\mathbf{N}$. The rate of rotation of the Osculating plane around $\mathbf{T}$ is a scalar multiple of the Principal Normal Vector. The negative quantity is a matter of direction. From this we find

$$\tau = -\frac{d\mathbf{B}}{ds} \cdot \mathbf{N}. \tag{4.8}$$

Therefore, we can quantify the amount of twisting by a scalar instead of a vector. Unfortunately the computation of torsion is distorted by noise since the vector $\frac{d\mathbf{B}}{ds}$ is the third derivative of $\mathbf{r}$. Fortunately, curvature only uses the second derivative of $\mathbf{r}$ and in practice is a good representation of κ.

Curvature and Gestures

Humans have difficulties executing straight-lines. This results in curved portions of gestures that are intended to be straight. For gestures performed away from the body, the hands are far away from their pivots, constraining the user to curve the trajectory. Figure 4.3 shows some gestures with curved paths at different angles. Both "Check Mark" and "X" gestures have straight lines for front views, but higher curvatures for side views.

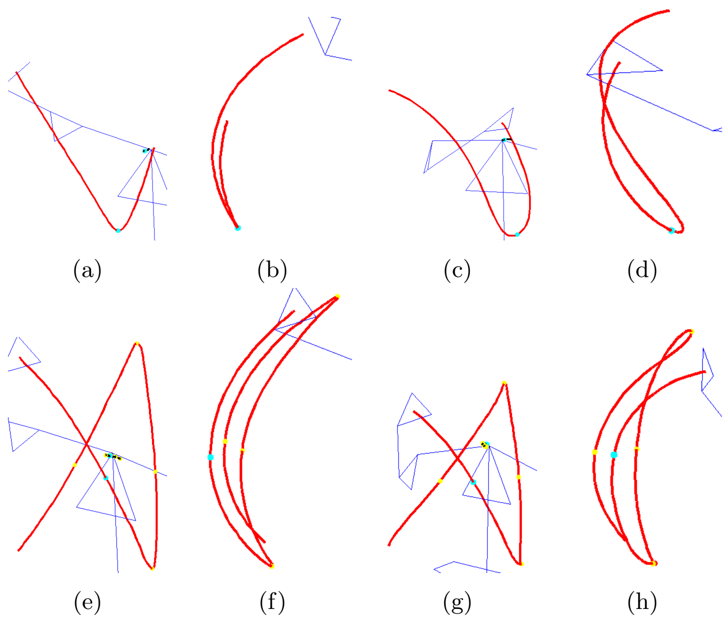


Figure 4.3: Front and side views of "Check Mark" and "X" gestures. (a) Front view of a check mark. (b) Side view of (a). (c) Front view of another check mark. (d) Side view of (c). (e) Front view of an "x" gesture. (f) Side view of (e). (g) Front view of another "x" gesture. (h) Side view of (g).

It is likely that users attempt to perform straight lines, but it appears that performers naturally curve idealized straight lines. The curved "straight" lines of naturally performed gestures have consequences for the number of states in a Hidden Markov model. If the advantage of Hidden Markov models is to use a minimum number of states without loss of accuracy, then one needs to increase the number of states from the idealized gesture to the actual recorded gesture. To be specific, the idealized gesture is an "X" or a "Check Mark" that can be perfectly fit to a plane since these are two dimensional gestures. The actual gesture encodes information about how the path varies within the dimension that would be ideally discarded if it were an idealized gesture. In practice, we should not project a gesture in 3D space to 2D space since this would allow the projected gesture to look identical despite variations in the discarded dimension. Therefore the variations in this dimension are important. The idealized gesture, such as a "Check Mark", would have less states than the actual gesture. An idealized "Check Mark" gesture parametrized by arc length would have two distinct states, however, due to the naturally curved portions it would have at least zero or more states than its idealized form. If we consider direction features, then those directions would not be constant across the portions that would be ideally straight. Therefore, to encode the gesture without loss of accuracy, the number of states should be increased to accommodate for this natural variation.

4.1.4 Polar and Relative Distance Features

The use of relative distance changes over time of positional features and angles between consecutive points of gestures results in features that are scale, rotation, and translation invariant. These features may, however, have undesirable properties.

Two-Dimensional Resampled Polar and Relative Distance Features

Let $\mathbf{a}$ and $\mathbf{b}$ represent two different gestures of the same class. The gestures are uniformly scale invariant, which ideally means $\mathbf{b} = \alpha\mathbf{a}$ after they are both translated to the origin. Both gestures have already resampled. Assume these gestures are single paths in two dimensions. The centre of $\mathbf{a}$ is $\mathbf{c_a}$ and the centre of $\mathbf{b}$ is $\mathbf{c_b}$. Lets assume that both gestures are already translated to the origin and $\mathbf{c_a} = \mathbf{b_a} = \mathbf{0}$.

The two features described in this section are angular and distance difference features from the two re-sampled gestures $\mathbf{a}$ and $\mathbf{b}$. The first feature is scale invariant and defines the change in length from one point $\mathbf{a_i}$ to the next $\mathbf{a_{i+1}}$. The length of each point with respect to its centre is defined as $l_i^a = ||\mathbf{a_i}||$. The change in length from one point to the next is defined as $d_i^a = \frac{l_{i+1}^a - l_i^a}{l_{i+1}^a}$. This feature is scale invariant because the denominator l_{i+1} cancels out the scaling lengths of the numerator. Since $\mathbf{b} = \alpha\mathbf{a}$ then, $l_i^b = ||\mathbf{b_i}|| = \alpha||\mathbf{a_i}|| = \alpha l_i^a$ and $d_i^b = \frac{l_{i+1}^b - l_i^b}{l_{i+1}^b} = \frac{\alpha l_{i+1}^a - \alpha l_i^a}{\alpha l_i^a} = d_i^a$, which proves the scale invariant property. Notice that d_i^a can be negatively or positively valued. Negative values mean that the next point $\mathbf{a_{i+1}}$ has a smaller length than the previous point, and positive values mean that the next point $\mathbf{a_{i+1}}$ has a larger length than the previous point. The second feature defines the angular change from one point $\mathbf{a_i}$ to the next $\mathbf{a_{i+1}}$. The angular change in direction is defined as $\theta_i^a = F\left(\frac{\mathbf{a_{i+1}} \cdot \mathbf{a_i}'}{\mathbf{a_i}' \cdot \mathbf{a_i}'}\right) \times \arccos\left(\frac{\mathbf{a_{i+1}} \cdot \mathbf{a_i}}{||\mathbf{a_{i+1}}||||\mathbf{a_i}||}\right)$. The function $F(x) = 1$ when $x \geq 0$ and $F(x) = -1$ when $x < 0$. F allows θ_i^a to retain a proper sequence of directions, since arccos only returns the magnitude the angle. For $\mathbf{a_i} = (a_{i,x}, a_{i,y})$, $\mathbf{a_i}' = (a_{i,y}, -a_{i,x})$ represents an orthogonal vector to $\mathbf{a_i}$. Since $\mathbf{b} = \alpha\mathbf{a}$ then $\theta_i^b = \theta_i^a$ due to the same cancelling property with the denominator inside the arccos function. Therefore rotation and scale invariance are achieved.

Unfortunately the features θ_i^a and d_i^a have a serious problem. The problem becomes

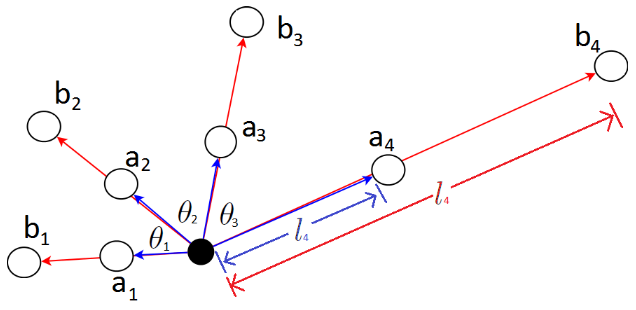


Figure 4.4: Two Dimensional Feature Representation.

apparent if the length l_i^a becomes zero. If $l_{i+1}^a = 0$ then d_i^a will become infinite and the value $\frac{\mathbf{a_{i+1}} \cdot \mathbf{a_i}}{||\mathbf{a_{i+1}}||||\mathbf{a_i}||}$ will also become infinite. Due to noise from sensors, it is unlikely that l_i^a is zero in practice, but one must worry about the length becoming very close to zero, and when this occurs, we need to handle it carefully when computing similarity scores. Otherwise we must design gestures where their paths do not cross through their centres.

The advantage to these approaches is the ability to define rotation and scale invariant features without the need to find the best scaling and rotation parameters. The scale invariant property allows us to avoid normalizing positional features through normalizing bounding boxes to squares. An additional advantage is that they can be extended to three dimensions by adding another rotation feature. The change in length feature remains the same, but we need another feature to describe how the next point $\mathbf{a_i}$ changes with respect to the plane defined by $\mathbf{a_{i+1}}$ and $\mathbf{a_i}$ with respect to the origin.

An extension to three dimensions involves a second angle that handles twisting motions. The twist motion involves a third vector $\mathbf{a_{i+2}}$ that twists out of the plane defined by the previous two vectors $\mathbf{a_{i+1}}$ and $\mathbf{a_i}$. In addition to this twist motion the angle of turning relative to the plane defined by $\mathbf{a_{i+1}}$ and $\mathbf{a_i}$ must be calculated. Initially, one must define an orthonormal basis $\mathbf{v} = \frac{\mathbf{a}_i \times \mathbf{a}_{i+1}}{||\mathbf{a}_i \times \mathbf{a}_{i+1}||}$, $\mathbf{w} = \frac{\mathbf{a}_{i+1}}{||\mathbf{a}_{i+1}||}$, and $\mathbf{u} = \mathbf{v} \times \mathbf{w}$. Then we can project $\mathbf{a_{i+2}}$ into this new coordinate system as $\mathbf{a_{i+2}} = \alpha_1 \mathbf{w} + \alpha_2 \mathbf{u} + \alpha_3 \mathbf{v}$. These resulting values are $\alpha_1 = \mathbf{a}_{i+2} \cdot \mathbf{w}$, $\alpha_2 = \mathbf{a}_{i+2} \cdot \mathbf{u}$, and $\alpha_3 = \mathbf{a}_{i+2} \cdot \mathbf{v}$. The new angles are defined as $\theta = -F(\alpha_2) \cos\left(\frac{\alpha_1}{\sqrt{\alpha_1^2+\alpha_2^2}}\right)$, and $\phi = \sin\left(\frac{\alpha_3}{\sqrt{\alpha_1^2+\alpha_2^2+\alpha_3^2}}\right)$

Two Dimensional Polar and Relative Distance Non-Resampled Features

The previous section discussed features that are rotation and scale invariant. They were also implicitly translation invariant. However they need to be resampled and translated given known start and end times before they can be compared. This section discusses two features that are translation, and rotation invariant in two dimensions, without the need to know the start and end positions.

Figure 4.5 shows two extracted vectors from a stream of raw positional points. Angles and relative distances are calculated from the two main vectors $\mathbf{v_i}$ and $\mathbf{v_{i+j}}$. These can be thought as velocity vectors, but these velocity vectors can represent different sampling rates by adjusting the parameter j which controls how far away the starting position $\mathbf{a_{i+j}}$ of $\mathbf{v_{i+j}}$ is from the starting position of $\mathbf{a_i}$ of $\mathbf{v_i}$. In general, both vectors have the form $\mathbf{v_i} = \mathbf{a_{i+k}} - \mathbf{a_i}$ where k represents the furthest index of the next raw position should be from $\mathbf{a_i}$. The index k determines how far to sample for the end positions of $\mathbf{v_i}$ and $\mathbf{v_{i+j}}$, whereas j determines the how far apart the starting positions of $\mathbf{v_i}$ and $\mathbf{v_{i+j}}$ should be. Figure 4.5 shows an example of $\mathbf{v_i}$ and $\mathbf{v_{i+j}}$ when $j = 2$ and $k = 3$.

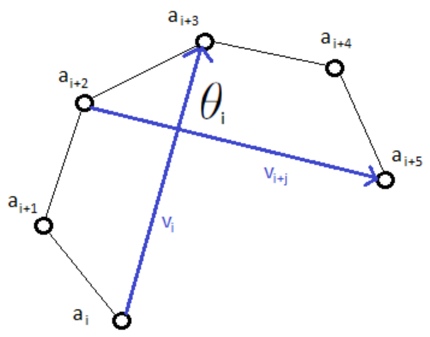


Figure 4.5: Two Dimensional Feature Representation.

The angle is calculated as $\theta_i = F\left(\frac{\mathbf{v_{i+j}}\cdot\mathbf{v_i'}}{\mathbf{v_i'}\cdot\mathbf{v_i'}}\right) \times \arccos\left(\frac{\mathbf{v_{i+j}}\cdot\mathbf{v_i}}{||\mathbf{v_{i+j}}||||\mathbf{v_i}||}\right)$ and the relative distances as $d_i = \frac{||\mathbf{v_{i+j}}||-||\mathbf{v_i}||}{||\mathbf{v_i}||}$ with $\mathbf{v_i'} = (v_{iy}, -v_{ix})$ if $\mathbf{v_i} = (v_{ix}, v_{iy})$. These features are rotation invariant in two dimensions, however they still suffer from the issue of the denominator becoming zero. Fortunately the gestures do not have to be designed such that their paths do not cross their centres since these do not focus on their lengths relative to the centre. The denominators become zero if either $\mathbf{v_i}$ and $\mathbf{v_{i+j}}$ have zero lengths, but this only occurs when there is no motion. If instead we parametrize by arc-length distance or displacement, then we can ensure a minimum sampling distance between consecutive points which allows the denominators to never be zero.

The advantage to these feature over the previous section of features is that there is no assumption of where the start and end positions are. The previous section needs the start and end points in order to calculate similarity scores. This approach does not assume start and end locations and therefore is more suited to gesture spotting from a continuous stream of data.

Similar to the previous extension, an orthonormal basis needs to be defined as $\mathbf{v} = \frac{\mathbf{v}_{i+j}\times\mathbf{v}_i}{||\mathbf{v}_{i+j}\times\mathbf{v}_i||}$, $\mathbf{w} = \frac{\mathbf{v}_{i+j}}{||\mathbf{v}_{i+j}||}$, and $\mathbf{u} = \mathbf{v} \times \mathbf{w}$. Then the third vector can be projected as $\mathbf{v}_{i+2j} = \alpha_1\mathbf{w} + \alpha_2\mathbf{u} + \alpha_3\mathbf{v}$, and exactly as in the previous section the angles θ and ϕ can be computed.

4.1.5 Other Features

Inter-path features describe how two or more paths relate to one another. A hand-rolling gesture, where the left and right hand move in a circular motion, can be described by how a vector connecting the left and right hand changes over the course of that gesture. Angles and angular velocities of this vector can be extracted. The direction of this vector can be used so that the distance between the hands is ignored. Raw distances between the left and right hand can be extracted and distance velocities can be extracted from those raw distances. Other features are contours, centre of mass, palm-direction similarity, and normalized velocities by standard deviation.

4.1.6 Summary of Local Rotation, Scale, and Translation Invariance

Table 4.1 provides a summary of the different rotation, scale, and translation invariant features. Not all features are invariant for all types of invariances.

Feature(s)	**Scale**	**Rotation**	**Translation**
Curvature & Torsion	No	Yes	Yes
Velocity	No	No	Yes
Acceleration	No	No	Yes
Position	No	No	No
Polar& Relative Distances	Yes	Yes	Yes
Direction	No	No	Yes

Table 4.1: Rotation, Scale, and Translation Invariant Summary.

4.2 Arc-Length Distance versus Arc-Length Displacement Parametrization

The arc-length distance $L(s, e)$ is the distance of a path measured over a time interval $[s, e]$.

$$L(s, e) = \sum_{t=s}^{e-1} ||\mathbf{a_{t+1}} - \mathbf{a_t}|| \tag{4.9}$$

We can use the idea of arc-length distance to re-sample a path using a pre-defined sample distance q. The idea is to convert the set of points $(\mathbf{a_1}, \mathbf{a_2}, \dots)$ to the set of points $(\mathbf{p_1}, \mathbf{p_2}, \dots)$ where the distance between each consecutive point is $q = ||\mathbf{p_{i+1}} - \mathbf{p_i}||$.

To do so, the first point $\mathbf{p_1}$ is set to be the same as $\mathbf{a_1}$. Additionally we set some parameters $d = 0$, and $s = 1$. The parameter d keeps track of the distance between the previous point $\mathbf{p}$ and $\mathbf{a_s}$, and s is the index to the point $\mathbf{a_s}$ that was used as the anchor point for resampling of the latest resampled point $\mathbf{p}$. Algorithm 4.2 allows one to resample the next point $\mathbf{p}$ and update d and s correctly. However it should only be applied when we know that the condition $L(s, s + j) - d < q$ can be satisfied.

Algorithm 4.1 Incremental Arc Length Distance Re-Sampling

```
$j \leftarrow 1$
while $L(s, s+j) - d < q$ do
    $j \leftarrow j + 1$
end while
$\alpha = \frac{||\mathbf{a_{s+j}} - \mathbf{a_{s+j-1}}|| - L(s,s+j) + q + d}{||\mathbf{a_{s+j}} - \mathbf{a_{s+j-1}}||}$
$\mathbf{p} \leftarrow \mathbf{a_{s+j-1}} + \alpha \left(\mathbf{a_{s+j}} - \mathbf{a_{s+j-1}}\right)$
$d \leftarrow ||\mathbf{a_{s+j}} - \mathbf{a_{s+j-1}}|| - L(s, s+j) + q + d$
$s \leftarrow s + j - 1$
```

Arc-Length displacement sampling is very similar to arc-length distance sampling. The idea is to use the distance between the current point s and any following point $s+j$ with $j > 0$, as the conditional distance that must be satisfied to produce the next point. Algorithm 4.1 would be replaced by Algorithm 4.2 with the first point being $\mathbf{p'} \leftarrow \mathbf{a_1}$ and $s \leftarrow 1$:

Algorithm 4.2 Incremental Arc Length Displacement Re-Sampling

```
$\mathbf{p} \leftarrow \mathbf{p'}$
$j \leftarrow 1$
while $||\mathbf{p} - \mathbf{a_{s+j}}|| < q$ do
    $j \leftarrow j + 1$
end while
$\mathbf{c} \leftarrow \mathbf{a_{s+j}} - \mathbf{a_{s+j-1}}$
$\mathbf{a} \leftarrow \mathbf{a_{s+j-1}} - \mathbf{p}$
$\alpha = \frac{-\mathbf{a}\cdot\mathbf{c} + \sqrt{(\mathbf{a}\cdot\mathbf{c})^2 - (\mathbf{c}\cdot\mathbf{c})(\mathbf{a}\cdot\mathbf{a} - q^2)}}{\mathbf{c}\cdot\mathbf{c}}$
$\mathbf{p'} \leftarrow \mathbf{a_{s+j-1}} + \alpha \left(\mathbf{a_{s+j}} - \mathbf{a_{s+j-1}}\right)$
$s \leftarrow s + j - 1$
```

The benefit of arc-length displacement sampling is that motion detection is not necessary since arc-length distance sampling can self-loop to For example, if a person is keeping a path stationary it is not guaranteed to that each consecutive point is in the same position in space. Sensor noise and small involuntary movement variations can cause differences in the recorded locations. These small location differences cause arc-length distances to accumulate and eventually become large enough to be considered as a new point to be re-sampled as an arc-length point. Therefore, even though the person tried to keep their body part at one location, the re-sampling method re-sampled several adjacent arc-length points to be approximately in that one location. Arc-length displacement sampling guarantees that the next point is at least a given distance away from the previous, but of course non-adjacent points in time can come back to the same location.

The above approach requires more computation to ensure that each point is displaced by the same distance q. If we are willing to accept points that are approximately the same displaced distance then we can modify the Algorithm 4.2 to Algorithm 4.3.

Algorithm 4.3 Incremental Arc Length Displacement Re-Sampling Approximate

```
$\mathbf{p} \leftarrow \mathbf{p}'$
$j \leftarrow 1$
while $||\mathbf{p} - \mathbf{a_{s+j}}|| < q$ do
    $j \leftarrow j + 1$
end while
$\mathbf{p}' \leftarrow \mathbf{a_{s+j}}$
$s \leftarrow s + j$
```

4.3 Smoothing

The existence of noise in all sensing devices is almost certain. Fortunately, the amount of noise varies with the quality of the sensing device and software. Smoothing is one method of filtering out unwanted noise from the raw data points. Convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau \tag{4.10}$$

can be used to smooth functions in a special way. Convolution has the special property of derivatives of $(f * g)(t)$ being equal to the convolution of one function $f(t)$ and the derivative of the other $g(t)$. Formally stated as $\frac{d}{dt}(f * g)(t) = \left(f * \frac{dg}{dt}\right)(t)$. This property has the advantage of computing the derivative of the Gaussian smoothing function first and then convolving instead of convolving using the Gaussian first and then computing the derivative.

Velocity and Acceleration can now be computed using $\left(\mathbf{r}_i * \frac{dG}{dt}\right)$ and $\left(\mathbf{r_i} * \frac{d^2G}{dt^2}\right)$ respectively where $G(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t}{\sigma}\right)^2}$ and $\mathbf{r_i}$ is the i'th dimension of the raw position data. An additional parameter is introduced from the Gaussian function and that is the smoothing parameter σ. The larger the smoothing parameter the smoother the result for velocity and acceleration. A second additional parameter is also included when we use the discrete version of convolution where a window size of the Gaussian function must be determined beforehand. This window size needs to be larger enough to perform proper filtering. In a real time program the window size must be set to an appropriate value to ensure fast response times.

4.4 Global Features

Global features capture information about the entire gesture. Global features include, for example sums, averages, variances, and overall angles of different features. Bounding volumes are volumes of space that enclose a gesture. They can be represented using bounding rectangular prisms, bounding spheres, or bounding convex hulls. One purpose of bounding volumes is to provide a way to normalize the feature space for scale invariant gestures. Unfortunately, if a gesture is a 2D gesture or a 1D gesture then scaling needs to be handled carefully since gestures can be distorted when they are scaled to a cube. For 3D gestures, bounding boxes can be used to scale gestures to a cube. However, there are better ways to normalize gestures without the need of bounding boxes.

Averages and variances can be calculated for many features. The average velocity, average distance, average acceleration, average curvature, average torsion, and average position are examples of averages of some features. Similarly, variances can be calculated for each of those features. Averages and variances can be used for feature normalization and post-processing verification. Post-processing verification is verifying that the features of a gesture are within some pre-defined or trained threshold. For example, the average magnitude of velocity must be close to the average magnitude of velocity from the training data. Another example is the average curvature for a circle gesture must not be too large or too small, otherwise it would not be an acceptable circle. If the variance is too large for this circle gesture, then the curvature might be changing too much locally. Variances could also be used to normalize features to have unit variances so that comparisons between the same features of different gestures can be compared.

Sums can also be used for scale invariance. The total arc length, which is the sum of all the distances between consecutive points of the gesture, can be used to re-sample a gesture so that those re-sampled points can be used to determine the proper scaling parameter. Perhaps the total arc-length should not be significantly different from the average arc-lengths of the training data. Overall angles are global angles describing certain angles of the shape of the gesture. One such overall angle is the “indicative angle” [75], which was used to heuristically rotate each gesture to the best starting place to search for optimal angular alignment.

4.5 Vector Quantization

Vector Quantization (VQ) is the process of representing a continuous space by a finite set of vectors (see Gersho et al. [16] and Linde et al. [40]). Vector Quantization can be used to convert a vector $\mathbf{a}_t$ into an observation symbol o_t represented by an integer.

The goal of VQ is to partition a data set of n vectors into a finite set of clusters that best model the data. The best model is defined as the one that minimizes the average distortion value for a pre-defined finite M number of clusters (Linde et al. [40]). Each cluster is a partition of the data set, with the centroid representing all vectors inside that cluster. The average distortion is defined as the average distance of all vectors to their respective centroids

$$D_m = D\left(\{A_m, P(A_m)\}\right) = \frac{1}{n}\sum_{j=1}^{n} \min_{\mathbf{y}\in A_m} d\left(\mathbf{x}_j, \mathbf{y}\right).$$

The codebook vectors $A_m = \{\mathbf{y}_i | i = 1, \ldots, M\}$ are a set of vector centroids that represent each cluster. The vector $\mathbf{x}_j$ is the j'th vector of the entire set of vectors, and $d(\cdot)$ is the distance measurement of each vector to its cluster centroid. $P(A_m)$ is the best partition found given of the vector centroids A_m, and it is defined as $P(A_m) = \{S_i | i = 1, \ldots, M\}$ where S_i is a subset of vectors of the entire data set of n vectors. A vector $\mathbf{x}_j$ is a member of cluster S_i if its codebook vector $\mathbf{y}_i$ has the smallest distance $d(\mathbf{x}_j, \mathbf{y}_i)$ to it, i.e. $\arg\min_i \{d(\mathbf{x}_j, \mathbf{y}_i)\}$.

Algorithm 4.4 LBG Vector Quantization Algorithm [40]

```
INPUT: $A_0$ and $\epsilon$
$D_m \leftarrow -\infty$
$m \leftarrow 0$
$P(A_m) \leftarrow \{S_i | i = 1, \ldots, M\}$
$D_m \leftarrow \frac{1}{n} \sum_{j=1}^{n} \min_{\mathbf{y} \in A_m} d(\mathbf{x}_j, \mathbf{y})$
while $(D_{m-1} - D_m)/D_m > \epsilon$ do
    $m \leftarrow m + 1$
    $A_m \leftarrow \left\{\mathbf{y}_i = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x} | i = 1, \ldots, M\right\}$
    $P(A_m) \leftarrow \{S_i | i = 1, \ldots, M\}$
    $D_m \leftarrow \frac{1}{n} \sum_{j=1}^{n} \min_{\mathbf{y} \in A_m} d(\mathbf{x}_j, \mathbf{y})$
end while
OUTPUT $A_m$
```

The distortion measure $d(\cdot)$ is usually Euclidean distance. Another possible similarity is the angle of the cosine distance. This distance measure ignores magnitudes and is affected by directions unlike Euclidean distance which is effected by both magnitude and direction changes.

The size of the initial codebook can be set using a splitting technique [40]. The idea is to start with a codebook of size 1, and then split this using a perturbation vector γ. The next codebook is $A_m = \{y_i + \gamma | i = 1, \ldots, M\} \cup \{y_i - \gamma | i = 1, \ldots, M\}$, giving rise to twice the number of symbols for every split. Let this splitting function be A_{m-1} as $SPLIT(A_{m-1}, \gamma)$. Once we have two codebook vectors we can use the LBG Algorithm 4.6 to get the best codebook for two symbols. We can split the result using Algorithm 4.5 and apply LBG again to get another codebook of four symbols. This process can be repeated until we have 2^R symbols where R is a predefined parameter.

Algorithm 4.5 Splitting Algorithm [40]

```
INPUT: $R$, $\gamma$ and $\epsilon$
$A \leftarrow \left\{\mathbf{y}_1 = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j\right\}$
for $r \leftarrow 1 \rightarrow R$ do
    $A \leftarrow SPLIT(A, \gamma)$
    $A \leftarrow LBG(A, \epsilon)$
end for
OUTPUT $A$
```

The perturbation vector γ must be chosen carefully since too large or too small splits cause some clusters S_i to be empty.

4.5.1 Cosine Angle VQ

Velocity has magnitude and direction. The angle of the cosine distance between two velocity vectors returns their similarity in terms of their directions. Single path velocity gestures described by codebook symbols through this distance measure are similar if their directions over time are similar. For multi-path gestures, differences in velocity magnitudes of the individual paths affect the similarity measure when the paths are combined into a single higher dimensional path. If movement along one path is moving much faster than the others, then that path will sets the standard for what is considered similar. For example, a two-handed gesture tracked using velocity has six components with three components belonging to each hand. If the right hand moves quicker than the left, then the three components of velocity for the right hand will be on average greater than the three components of velocity for the left hand. Combining the two paths into six dimensions causes the speed differences in both hands to be directed more towards the hand that is moving faster. A two-handed gesture with the right hand moving faster than the left hand will be in a different 6 dimensional direction than a two-handed gesture with the left hand moving faster than the right hand. This difference in speed of both hands causes the angle of the cosine distance to be different, and ultimately be quantized vector to be a different symbol. If one used the equal coding-condition, then a similar advantage to the MD-Protractor is gained where distinguishing different numbered of path gestures becomes easy. The reason is the same as for MD-Protractor, except this is realized at the level of a single local features instead of all local features.

4.5.2 Vector Quantization in Practice

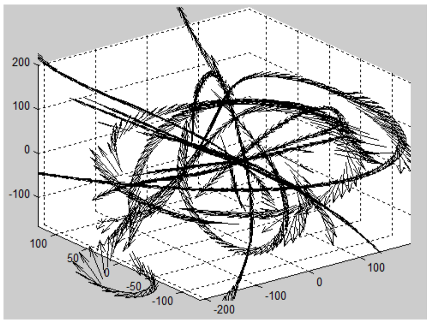
Figure 4.6 shows an example of Vector Quantization using direction features of single path gestures. Figure 4.6a represents of one gesture from each class using directional features. The directions are shown at 50 times their normal length. In Figure 4.6b, all the directions of each gesture of Figure 4.6a are translated to zero and displayed in a spherical representation, with Figure 4.6c being a scatter plot version of Figure 4.6b. Figure 4.6d includes all samples, not just one sample of every class. It is the first split after finding the best partition. There are now two observation symbols, or, equivalently the codebook size is two. The second split is shown in Figure 4.6e with four observation symbols, and this continues until there are 16 observation symbols in the fourth split of Figure 4.6g. The long thick lines protruding out of each sphere are the centroids and their corresponding numbers are the symbols. Each coloured area on the sphere is a cluster corresponding to a single centroid and symbol. Vector classification is performed by finding the closest centroid and labelling it with the corresponding symbol number. Finally, Figure 4.6h show the distortion curves for every iteration of the LBG Algorithm 4.6. Each curve is taken from a single split of the splitting Algorithm 4.5. In this example the y-coordinate represents the distortion level, and the x-coordinate is the iteration. There are many iterations since the stopping parameter ϵ was set to 0.001. A stopping value of 0.01 is what is conventionally used.

Figure 4.7 is an example of vector quantized gestures using the codebook from Figure 4.6. The colour of every feature corresponds to the codebook vector from the fourth split. A "Stepping Right" gesture has mainly red symbols corresponding to codebook symbol 15. A "Check Mark" gesture has one main codebook symbol of 12 for the downward motion and two main codebook symbols of 13 and 14 for the upward motion.

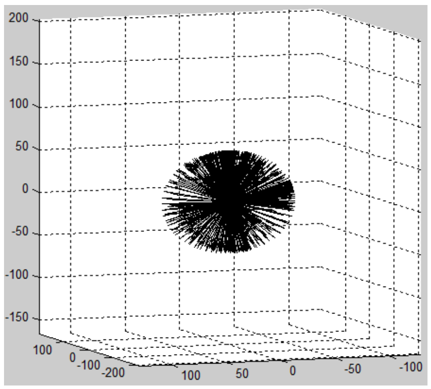
4.5.3 Unsupervised VQ Parameter Selection

As shown above, gestures can be Vector Quantized into a sequence of integer symbols. The choice of parameter R for the codebook size 2^R is unclear. It appears that it was arbitrarily selected. Notice that the larger the codebook size the more codebook symbols there are. For gestures of the same class, this means that there is a higher chance that their vector quantized sequences have different symbols due to variations in their features. At the same time, gestures from different classes are also very likely to have different symbol sequences with respect to another class. The advantage to using a large number of codebook symbols ensures better between class separation, but at the same time has the disadvantage of increasing the chance of within-class separation. This results in poor similarities between gestures of the same class and can ruin recognition accuracy. Alternatively, the smaller the codebook size the fewer symbols there are. This means that gestures of the same class have a lower chance that their symbol sequences have different symbols. In other words there is a higher chance that their symbol sequences are the same. At the same time, gestures from different classes are unlikely to have different symbol sequences, or, in other words they have a higher chance to have the same symbol sequences. The advantage of using a smaller codebook size is the within-class similarity between symbol sequences is high, but the disadvantage is the between-class similarity is also high. One must find the codebook size that gives the best balance between within-class similarity and between-class separation. Unfortunately, the higher the within-class similarity, the lower the between-class separation.

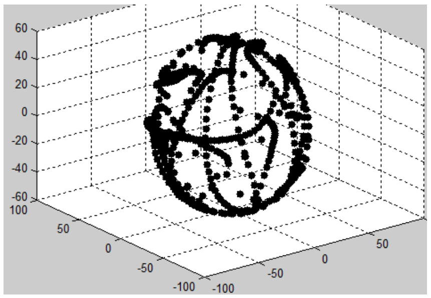
Glomb et al. [17] described a technique for finding the best codebook size for classification. The first step is to define a measure to compute the within-class similarity and between-class separation. This is accomplished using Levenshtein distance to measure the difference between two symbol sequences. A histogram of distance measurements can be computed to the within-class and of the between-class differences. In order to achieve high within-class similarity, the histogram of within-class differences should have its average close



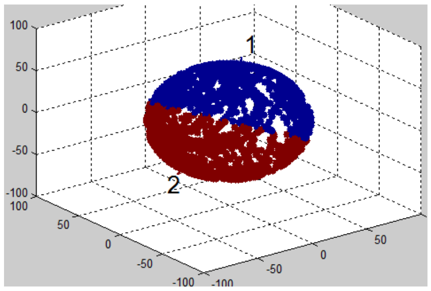
(a) A representation of all classes using one gesture per class.



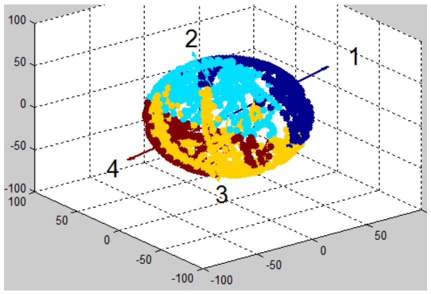
(b) Directions of 4.6a translated to zero.



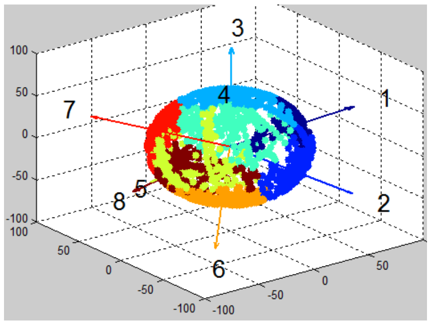
(c) Scatter plot of 4.6b.



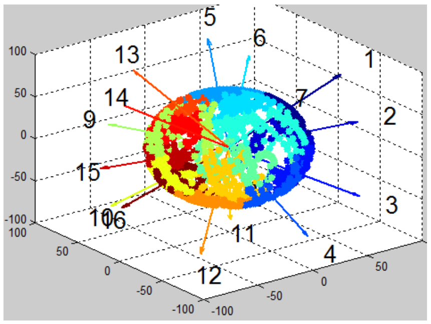
(d) First split of vector quantization using many more gesture samples.



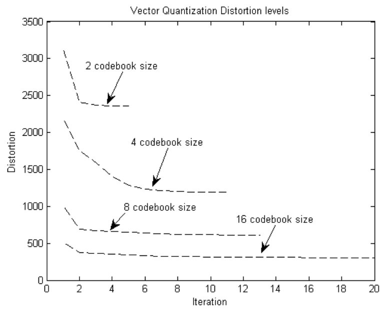
(e) Second split.



(f) Third split.

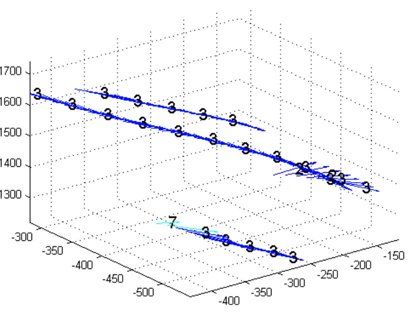


(g) Fourth split.

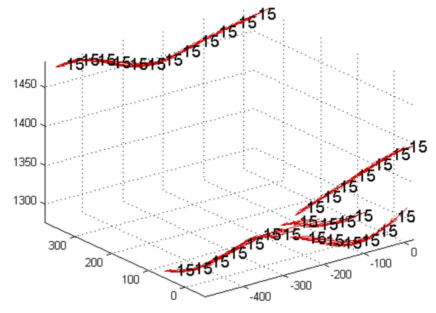


(h) Distortion curves for each split.

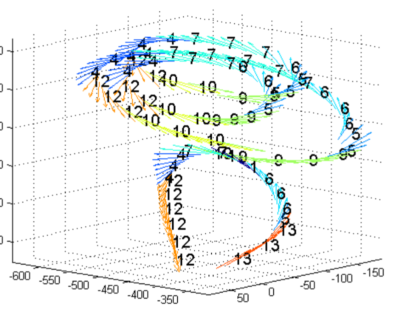
Figure 4.6: Vector Quantization using the LBG algorithm. The arrows of (d)-(g) are the quantized vectors, or centroids of each partition. The numbers of each feature is the codebook index, which is also the symbol number.



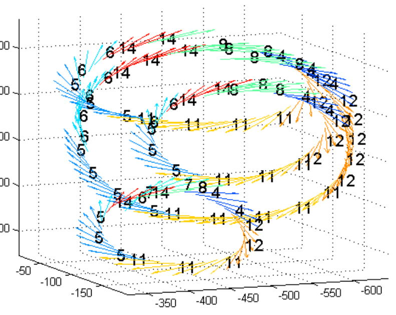
(a) “Stepping Left” gesture.



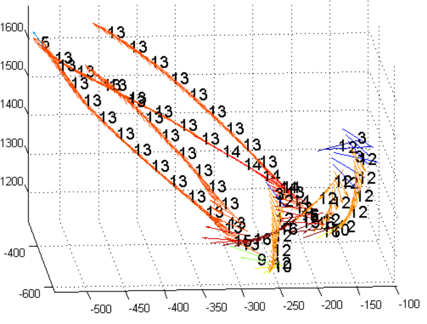
(b) “Stepping Right” gesture.



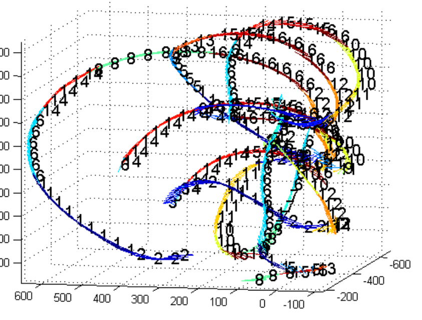
(c) “Arm Roll Backwards” of left hand.



(d) “Arm Roll Backwards” of right hand.



(e) “Check Mark” gesture.



(f) “Grouping Gesture” single path.

Figure 4.7: Trajectories of single path gestures showing their directions and their codebook representation. They are colour coded based on the closest quantized vector they belong to.

to zero. To achieve high between-class separation, the histogram of between-class difference should have its average as far away from zero as possible. The histograms are normalized to account for differences in the number of training data. Glomb et al. [17] used this property to find the best codebook size without the need to perform full recognition tests under different codebook sizes.

4.6 New Contributions

We developed polar features, relative distance features, and a uniform scale invariance method for positional features. The Polar features achieve rotation invariance. The relative distance features achieve scale invariance of re-sampled gestures for geometric template matching. Both polar and relative distance features must be used together to achieve both scale and rotation invariance simultaneously. For positional features, uniform scale invariance can be achieved through minimizing the MSE between two gestures. This is different from scaling gestures to cubes independent of template gestures where aspect ratio invariance can be problematic and lower dimensional gestures become distorted. Minimizing the MSE using a uniform scale parameter between two gestures avoids aspect ratio invariance and does not distort gestures based on their dimension.

Chapter 5

Template Matching

Template Matching performs gesture recognition through a similarity measure, such as Euclidean distance, between a test gesture and a reference gesture. Dynamic Time Warping is one example of template matching that warps time to find the best possible match subject to sensible constraints. Some template matching techniques do not warp time, but have lower time complexity and achieve similar recognition performance.

5.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a technique that warps time to align two gestures of different lengths. DTW is restricted in the way it warps time so that the alignment makes sense.

We now define DTW. Let the matrices **a** and **b** represents two different gestures as discussed in Section 4.1. DTW begins by computing distances between every local feature of gesture **a** with every local feature of gesture **b** and places them into a cost matrix **C**. If gesture **b** has N time points and gesture **a** has M time points then the cost matrix **C** is of size $M \times N$. The cost element $c_{ij} = c(i, j)$ computes the distance between the local features $\mathbf{a}_i$ and $\mathbf{b}_j$ of both gestures. This cost function $c(i, j)$ could be euclidean distance, cosine angle, or some other distance measure.

A warping path is a path that traverses this cost matrix. The goal is to find the optimal warping path having the minimum total path cost. One can imagine creating warping paths by walking from location (M, N) and trying to reach location $(0, 0)$ under certain constraints. At any location (i, j), one can only change position to one of the eight adjacent locations. This ensures that changes in location should be approximately adjusted by a constant distance in each direction. Immediately we can eliminate all the changes in location that lead back towards the start location $(0, 0)$ since they all lead to non-optimal warping paths. One is left with locations $(i - 1, j)$, $(i, j - 1)$, or $(i - 1, j - 1)$ to move to since these all are moving towards the goal. Once one reaches a wall defined by $(i, 1)$ or $(1, j)$ then the only option is to go straight towards the goal.

The minimum total path cost can be computed recursively through the following formula

$$\mathbf{D}(i,j) = c(i,j) + \min \begin{cases} \mathbf{D}(i-1,j-1), \\ \mathbf{D}(i-1,j), \\ \mathbf{D}(i,j-1). \end{cases} \tag{5.1}$$

Notice that this formula works backwards to the direction one walked through the optimal warping path. The matrix $\mathbf{D}$ is called the accumulating cost matrix since it computes the minimum total cost of potential warping paths. Then $\mathbf{D}(M,N)$ holds the total cost of an optimal warping path since there may be many optimal warping paths. One can find an optimal warping path by walking through $\mathbf{D}$ from location (M,N) and moving to location $(0,0)$. The choice to move to one of locations $(i-1,j)$, $(i,j-1)$, or $(i-1,j-1)$ from (i,j) is done by choosing location with the smallest accumulated cost in $\mathbf{D}$. If there is a tie, one can randomly break such ties. An optimal warping path must be computed before classification can be performed. A normalized accumulated cost is found by computing the length of an optimal warping path and dividing the accumulated cost $\mathbf{D}(M,N)$ by this length. Classification performed by finding the template with the smallest normalized accumulated cost with an unknown gesture and classifying it to the class of that template.

5.1.1 Speeding Up DTW

If a gesture is from the same class then an warping path should have a slope similar to the slope along the diagonal of the accumulated cost matrix. This is essentially the assumption of geometric template matching methods which work well. Therefore we can limit our search to margins within this slope. Both Sakoe et al. [59] and Itakura [22] limit the search space of a warping path as described here. Sakoe et al. [59] limits the search space by a parallel margins within a distance from the diagonal, whereas Itakura uses a parallelogram covering the main diagonal to limit the search alignment. The approach by SakoeChiba [59] can be slower depending on the margin distance, but will allow for a larger range to adjust the end point alignments which can be much more useful. Salvador et al. [60] discusses a multi-scale approach to DTW in linear time and space. Of course these are all approximations, but based on reasonable assumptions.

5.1.2 Step-Size Constraints

Sakoe et al. [59] discuss local step size constraints to avoid the problem of allowing a single point be aligned to multiple points. A single point is allowed to be aligned to multiple points in the current recursion formula for the accumulated cost matrix. The alternative is to restrict the local slope between points in the warping path. For example instead of allowing transitions to points (i-1,j), (i-1,j-1), and (i,j-1) from (i,j), you can restrict it to only transition to points (i-2,j-1),(i-1,j-1), and (i-1,j-2). There are other possible local slope constraints [59], but each local slope constraint may violate the boundary condition that the first point of a warping path is (1,1) and the last point is (M,N). A gesture with a relatively large duration compared to the template will have to violate the local defined slope constraint if it is to not violate the boundary condition. Therefore gestures of relatively too large differences in length will not have the ability to compute a valid optimal warping path.

5.2 MD-Protractor

MD-Protractor is a recognition algorithm used to classify gesture sets that are not rotation invariant and involve multiple synchronized paths. The paths are synchronized in time and are different from the gesture sets described by Anthony et al. [2, 1], and others [76, 68] where multi-stroke gestures are done in any order and are not synchronized. An example set of gestures that fits this description are full-body gestures where multiple paths map to different parts of the body. The gestures must be rotation variant.

MD-Protractor extends the re-sampling algorithm by various geometric template matching techniques [75, 39, 32, 33, 2, 1] to the higher dimensions. In order to ensure that the same time is re-sampled across all paths, all three dimensional paths are treated together as one large higher dimensional path. Re-sampling in 2D and 3D depends on arc length parametrization which has an intuitive meaning in these dimensions but not in higher dimensions. Yet we can perform a higher level arc length parametrization in higher dimensions. Translation is similarly done in M-dimensions by calculating the M-dimensional center and subtracting it from all points.

MD-Protractor uses the angle of the cosine distance as its dis-similarity score. Vectorizing a matrix is performed by taking all the columns and making one large one-dimensional column vector by appending one vector after the other. As Li [39] vectorized positional data, MD-Protractor also vectorizes the positions so that the gesture can be compared to other gestures using cosine angle. Each individual path is vectorized by taking the x-dimension, y-dimension, and z-dimension into the same column by first putting the x-dimension, followed by the y-dimension, and followed by the z-dimension. Then, at a higher level these vectorized paths are placed into another vector by one vectorized path being placed after the other. For a three path gesture that tracks the left hand, right hand, and center of body will have three individual paths that are each individually vectorized and then each path are grouped into a single long vector.

The cosine angle has a specific type of scaling property in higher dimensions. It is not completely scale invariant. Let vectors $\mathbf{u} = \alpha\mathbf{a}$ and $\mathbf{v} = \beta\mathbf{b}$ be scale invariant with respect to vectors $\mathbf{a}$ and $\mathbf{b}$. If we scale a subset of values in either vector, then the scale invariance property no longer applies completely. Assume $\mathbf{a}$ and $\mathbf{b}$ have the following form

$$\mathbf{a} = \begin{bmatrix} \mathbf{l} \\ \mathbf{r} \\ \mathbf{c} \end{bmatrix}, \ \mathbf{u} = \begin{bmatrix} \mathbf{u}_l \\ \alpha\mathbf{r} \\ \mathbf{u}_c \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} \mathbf{b_l} \\ \mathbf{b_r} \\ \mathbf{b_c} \end{bmatrix}$$

where $\mathbf{l}$, $\mathbf{r}$, and $\mathbf{c}$ are the left hand, right hand, and center of body paths in vectorized form of $\mathbf{a}$. The vector $\mathbf{a}$ is considered the higher dimensional vectorized path. Similarly the vectors $\mathbf{b_l}$, $\mathbf{b_r}$, and $\mathbf{b_c}$ are also the left hand, right hand, and center of body paths in vectorized form of $\mathbf{b}$. Note that the gesture $\mathbf{u}$ has its right path as a scaled version of the right path of $\mathbf{a}$. The argument for the cosine angle is

$$\frac{\mathbf{u}^T\mathbf{b}}{||\mathbf{u}||||\mathbf{b}||} = \frac{\mathbf{u}_l^T\mathbf{b}_l}{||\mathbf{u}||||\mathbf{b}||} + \frac{\alpha\mathbf{r}^T\mathbf{b}_r}{||\mathbf{u}||||\mathbf{b}||} + \frac{\mathbf{u}_c^T\mathbf{b}_c}{||\mathbf{u}||||\mathbf{b}||}.$$

Let $\mathbf{b}_l \approx \mathbf{0}$, $\mathbf{b}_c \approx \mathbf{0}$, $\mathbf{l} \approx \mathbf{0}$, $\mathbf{u}_l \approx \mathbf{0}$, $\mathbf{u}_c \approx \mathbf{0}$, and $\mathbf{c} \approx \mathbf{0}$; This corresponds to the case where the left and center paths are almost stationary. Then we get $||\mathbf{u}|| \approx \alpha||\mathbf{r}||$, $||\mathbf{b}|| \approx ||\mathbf{b}_l||$, $\mathbf{l}^T\mathbf{b}_l \approx \mathbf{0}$, and $\mathbf{c}^T\mathbf{b}_c \approx \mathbf{0}$. Together these result in $\frac{\mathbf{u}^T\mathbf{b}}{||\mathbf{u}||||\mathbf{b}||} \approx \frac{\mathbf{r}^T\mathbf{b}_r}{||\mathbf{r}||||\mathbf{b}_r||}$, $\frac{\mathbf{a}^T\mathbf{b}}{||\mathbf{a}||||\mathbf{b}||} \approx \frac{\mathbf{r}^T\mathbf{b}_r}{||\mathbf{r}||||\mathbf{b}_r||}$, and $\frac{\mathbf{u}^T\mathbf{b}}{||\mathbf{u}||||\mathbf{b}||} \approx \frac{\mathbf{r}^T\mathbf{b}_r}{||\mathbf{r}||||\mathbf{b}_r||}$ which states that the gestures are almost fully scale invariant. Scaling issues occur when the right path of the gesture is scaled smaller or the amount of noise in the stationary paths increases dramatically. Suppose we decrease α below 1, then the errors in the approximations of $||\mathbf{u}|| \approx \alpha||\mathbf{r}||$, $\mathbf{l}^T\mathbf{b}_l \approx \mathbf{0}$, and $\mathbf{c}^T\mathbf{b}_c \approx \mathbf{0}$ increase. This means that they are less similar to zero. If we keep decreasing α to the point where the errors are larger than the errors in the approximation of $\frac{\mathbf{u}^T\mathbf{b}}{||\mathbf{u}||||\mathbf{b}||} \approx \frac{\mathbf{r}^T\mathbf{b}_r}{||\mathbf{r}||||\mathbf{b}_r||}$ then they can begin to dramatically affect the computed angle. Alternatively, the noise in the magnitude of the supposedly stationary signals can increase to a point that it has the same effect as decreasing the scale factor α.

5.3 Coding Conditions

A solution to this scaling issue is to completely ignore the non-meaningful paths of the gesture and keep the meaningful paths. A meaningful path of a gesture is the part of the body that is required to provide the motion for that gesture, and the non-meaningful path of a gesture is the parts of the body that are not required to provide any motion for that gesture. For example, a “Check Mark” gesture drawn using the right hand does not need the data from the other parts of the body to have meaning, such as the left hand. In this case, the right hand is the only meaningful part of the body, and all other parts of the body are non-meaningful.

This creates two coding conditions that need to be distinguished, and they are the **Equal Coding Condition** and the **Minimal Coding Condition**. The **Equal Coding Condition** involves using all trajectories, whereas the **Minimal Coding Conidition** involves using only the meaningful or minimal number of trajectories. The Equal Coding Condition causes lower path gestures to be projected into higher path gestures, such as projecting a

“Check Mark” gesture into a multi-path gesture. For single path gestures such as a “Check Mark” gesture drawn using the right hand, the other trajectories such as the left hand now become meaningful since they are included in the coding for the equal coding condition. This means that the test gestures must have the same behaviour across all other paths, even the meaningless paths such as the left hand for a “Check Mark” gesture drawn using the right hand. If there is no motion in the non-meaningful trajectories then there must also be no motion for those same trajectories in the gesture of the same class. The Minimal Coding Condition does not constrain the non-meaningful parts of the body to template behaviour and preserves true scale invariance for single path gestures. It has other problems that are discussed in detail in Section 6.7.

5.3.1 MD-Protractor and Coding Conditions

There is a recognition advantage to using the equal coding condition and it is due to the high dimensional space that these gestures are projected into by including all trajectories. In this high dimensional space, it becomes very easy to distinguish single left path gestures from single right path gestures, and single right path gestures from single center body gestures, single left path gestures from single center body gestures, and single path gestures from two path gestures. This is due to the stationary paths projecting single path gestures into a different directions than two handed gestures. This also applies to single path gestures where the opposite hand is stationary, thus being able to easily distinguish left handed gestures from right handed gestures by simply keeping the other hand stationary. However, there are disadvantages when the noise of the supposedly stationary signal becomes too high in magnitude or the scale of the meaningful part of the gesture becomes too small. If the noise of the stationary path increases while simultaneously decreasing the scale of the meaningful path, then the disadvantage becomes even more apparent. In other words, the user must keep all other parts of their body still while the meaningful part of their gesture is executing.

5.3.2 Simultaneous Preparation Under Both Conditions and MD-Protractors

In a fast pace environment where preparation time is crucial to success, MD protractor is at a disadvantage when we try to take advantage of its discrimination ability under the equal coding condition. Preparation time is increased if the user must keep their non-meaningful body parts stationary during execution of meaningful parts. Simultaneous preparation and execution is a very difficult skill because simultaneous meaningful movements are non-intuitive and can cause the meaningful part to be incorrectly performed. Simultaneous preparation is not a natural form of interaction; however there exists environments where this type of preparation cannot be ignored. A competitive video game is such an environment where the number of offensive moves must be performed quickly to result in a more a more advantageous position, especially in human to human competition. In this case the person who has developed the skill to simultaneously prepare and execute is at the advantage. Games are designed to test a person's skill which can be the ability to master simultaneous movements, or other non-intuitive forms of skill, such as badminton, and are an area where simultaneous preparation is required.

Let us consider a specific example of developing a skill for simultaneous preparation and execution. Lets use a check mark gesture for both hands where each check mark is a mirrored version of the other. Lets also define how they map to specific tasks in a video game. The left hand check mark when completed signals an offensive attack, and the right hand check mark triggers a restoration effect. In certain situations of a game, the user is forced to alternate between attack and restoration. Depending on the game designer, they may desire to limit attack and restoration tasks to be mutually exclusive, whereas others might allow both to occur simultaneously. Whether they are mutually exclusive tasks or not, the user should be allowed to move the other hand in a garbage motion while executing a meaningful gesture with the other. Let us focus on the mutually exclusive case. If a user keeps their left hand still while performing the right hand check mark, they have two choices. The first choice is to perform the left handed check mark in the location where their left hand is, however after performing the check mark their left hand is now located further up and to the left. If they keep their left hand stationary again until they can perform the next left hand check mark, then at some point they must bring their hand back to an arbitrary starting location that allows them to perform a meaningful left handed check mark. The second choice is to always bring their hand back to an arbitrary starting location so they can always perform the check mark. The second choice is better and is ultimately the only strategy if they need to perform this gesture many times with their left hand. The same situation occurs for their right hand. Thus a user must alternate between these gestures quickly, but also perform them correctly. The fastest way of alternating between these gestures is to simultaneously prepare and execute gestures with the opposite hand. Otherwise, the user who does not perform this method will not be able to perform as many attacks and restorations as an expert opponent would. However during these simultaneous preparations and alternations it can become difficult to distinguish two handed gestures from one handed gestures under the equal coding condition.

An advantage to the minimal coding condition is the possibility of simultaneous gesture recognition of single path gestures of both hands. Perhaps a user wants to simultaneous perform a left handed check mark and a right handed check mark. If we use the equal coding condition, then the motion form the non-meaningful hand is added noise to the similarity measure, and the user is then forced to keep the non-meaningful parts of their body still.

Under the minimal coding condition, the user is free to move all parts of their body freely and is better for simultaneous recognition of multiple gestures from different parts of the body.

5.4 New Contributions

We make developed the multi-dimensional geometric template matcher MD-Protractor, analyzed the scaling invariant behaviour of MD-Protractor, and introduced and analyzed two types of coding conditions. MD-Protractor is a synchronized multi-path geometric template matcher. The grouping of all paths into an M-dimensional path allows us to re-sample each individual path at the same time point. Furthermore, by using the angle of the cosine distance we can achieve almost full uniform scale invariance. MD-Protractor under the Equal Coding Condition is not fully uniformly scale invariant since the noise from the non-meaningful path will change the angles value. Additionally, the smaller the scale of the meaningful path the larger the change in angle since the noise affect from the non-meaningful paths are always relative to the scale of the meaningful path. To overcome this effect, users must perform large gestures under the Equal Coding Condition. The Equal Coding Condition and Minimal Coding Condition are introduced and are important for two reasons. First, the user can perform simultaneous preparations under the Minimal Coding Condition with little effort from the developer, whereas the Equal Coding Condition is restricted by the data gathered. Second, recognition is easier under the Equal Coding Condition since it can easily distinguish between specific groups of paths only when the user keeps their non-meaningful body parts stationary.

Chapter 6

Hidden Markov Models

Given a sequence of symbols, such as *aaaabbbbcc*, is there an approach that statistically models such observations? One method is to use a counting approach over the entire sequence of symbols, which we call the observation sequence, and keep track of the frequency or probability of the occurrence of each symbol. The result is a probability of 4/10 for "a", 4/10 for "b", and 2/10 for "c". Alternatively, one can assume a separate probability distribution for each symbol by segmenting the observation symbol into three parts. The first part would be the first four symbols, the second would be the next four symbols, and the third would be the last two symbols. The first distribution would give 4/4 for "a", 0/4 for "b", and 0/4 for "c", the second distribution would give 0/4 for "a", 4/4 for "b", and 0/4 for "c", and the third distribution would give 0/2 for "a", 0/2 for "b", and 2/2 for "c". This alternative approach models local segments better than using one single probability distribution. Each local segment can be referred to as a state, but this raises the issue of how the different observation probability distributions or states-to-observation probability distributions transfer between each other. A Markov Chain can be used to model transitions between different observation distributions so that probabilities can be used throughout the method.

A Hidden Markov Model (HMM) $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ uses the idea of hidden states that individually correspond to a unique observation probability distribution as described above. Transitions to the next state only depend on the current state and are represented by transition probabilities $\mathbf{A}$. $\mathbf{A}$ is an $N \times N$ matrix representing all the transition probabilities from one state to another where N is the number of states. The element a_{ij} of $\mathbf{A}$ is the probability of a transition from state i to state j. Each observation probability distribution is represented by the matrix $\mathbf{B}$ of size $N \times M$ where M is the number of observation symbols and $b_i(o_t)$ is the probability that we see the observation o_t given state i. Note that $b_i(o_t)$ is another way to denote the i'th row and o_t'th column of $\mathbf{B}$. o_t represents an observation symbol and is usually given a positive integer to distinguish between different symbols. In the example above M would equal 3 and N would equal 3, yet the first version only had one observation probability distribution and N would equal 1. The HMM notation used here follows [53].

Figure 6.1 is an example of an HMM with 3 states and 5 observation symbols. The diamond shaped objects represent the observation symbols. The circles represents the states, and the square is the null starting state. The weights connecting the null state to the regular states are the initial state distribution probabilities **pi**, and the weights connecting each state to each other state are the state transition probabilities **A**. Every state has a probability distribution with associated for all observation symbols.

HMMs are used to generate a sequence of observations $O_{1,T} = o_1, \ldots, o_T$ that represent the feature patterns of a given application. T is the length of the sequence and o_t is the observation symbol at time t. An HMM can use continuous observation probabilities or discrete observation probabilities to represent an observation sequence. Continuous observation densities such as multivariate Gaussian densities can be used to estimate the point-probability of o_t, and Vector Quantized observations of o_t can be used to give a finite representation of the continuous space where discrete observation probabilities can be used.

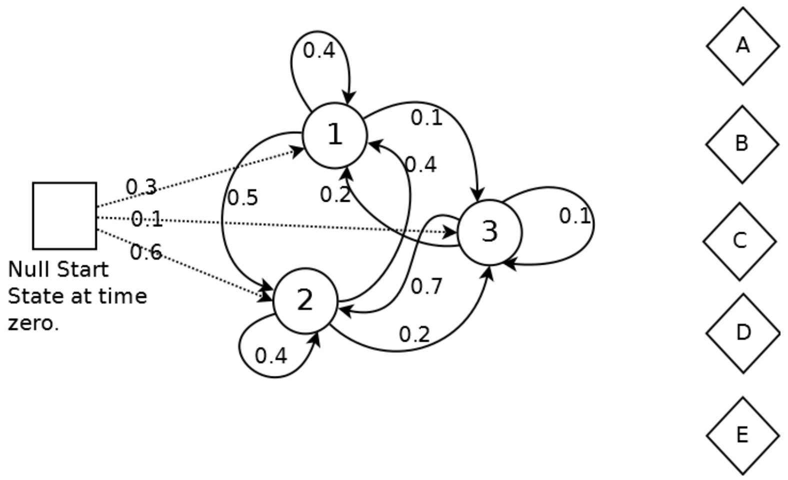


Figure 6.1: A Visualization of a discrete Hidden Markov Model.

A random number is used at every time point t where $1 \leq t \leq T$ to generate an observation. The HMM begins in the null state and transitions into any state i at $t = 1$, as represented by the initial probability for each state by a vector π, with the probability of transitioning into state i being π_i. At $t = 0$, using a random number, and π, an initial state can be generated. The probability distribution of the i-th row of B can be used to generate the first observation symbol o_1. Through successive state and observation generations an observation sequence $O_{1,T} = o_1 \ldots o_T$ and a state sequence $S_{1,T} = s_1 \ldots s_T$ can be generated. In practice, the observation symbols are not artificially generated like this, but are given, and the artificially generated observation sequence should be very similar to the given observation sequences. The higher this similarity the better the HMM models the given observation data.

6.1 Problems

Three problems must be solved for Hidden Markov Models. The first problem asks "Given an observation sequence, how likely is it that this HMM produced such a sequence?" The second problem asks "What is the best matching state sequence given an observation sequence?" Finally, the third problem asks "How should the parameters λ of the HMM be adjusted to maximize the likelihood of the given data". Essentially, the first handles **likelihoods**, the seconds deals with **decoding**, and the third problem deals with how to **train** the HMM.

Until now, only the parameter description $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ has been described, and nothing about how an HMM can be used in practice. An HMM must be able to provide a score with respect to an observation sequence. Otherwise distinguishing between different observation sequences would be very difficult without some value for comparison. This value this often refers to is the likelihood. Many authors refer to different types of likelihoods where the first likelihood considers all possible state sequences (first problem), and the second likelihood only considers the state sequence that gives the best value (related to the second problem). These likelihoods can be used to compare different observations sequences so that multi-class recognition becomes practical. There are further question to answer, namely "How many states should be used, and how many observation symbols should be used?". The first type of likelihood, the best matching path, and the best individual parameter values are the three important problems that must be solved to make Hidden Markov Models useful.

6.1.1 Problem 1: Evaluation

What is the likelihood of an observation sequence? It is the probability of how well it matches the exponential number of possible state transitions. Formally, the likelihood $P(O_{1,T}|\lambda)$ is the probability of an observation sequence $O_{1,T}$ given the the Hidden Markov Model parameters λ. The problem here is computing how well an observation sequence matches all possible state sequences, and since the number of possible state sequences is exponential in the number of observations of the observation sequence $O_{1,T}$ then the direct computation of this problem becomes intractable. Equation 6.1 shows the difficulty of directly computing the likelihood. To correctly compute the likelihood $P(O_{1,T}|\lambda)$, all possible state sequence $S_{1,T}$ must be assumed and jointly matched with the observation sequence $O_{1,T}$.

$$P(O_{1,T}|\lambda) = \sum_{s_1} \sum_{s_2} \cdots \sum_{s_T} P(O_{1,T}, s_1, s_2, \ldots, s_T|\lambda) \tag{6.1}$$

Figure 6.2 illustrates the exponential nature of equation 6.1 with an exploding number of state sequences. Fortunately, dynamic programming allows us to compute the likelihood by taking advantage of the problems recursive nature, as shown in Figure 6.3. Note that for every parent, the children are exactly the same children of the parents at the same point

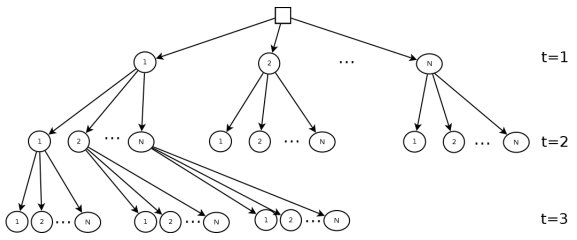


Figure 6.2: Exponential number of states as time increase.

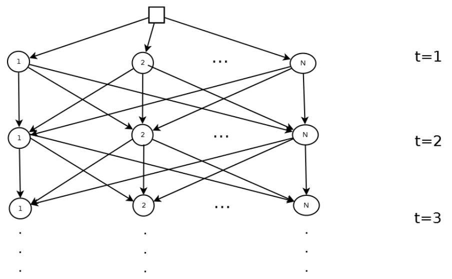


Figure 6.3: Recursive structural representation of exponential paths. Given this representation we can apply dynamic programming to evaluate $P(O|\lambda)$ efficiently.

in time, and this allows the children to be fused into the same children at the next point in time. Rabiner [53] shown that we can evaluate $P(O_{1,T}|\lambda)$ efficiently using the recursive representation of Figure 6.3 leading to Equation 6.2.

$$P(O_{1,T}|\lambda) = \sum_{i=1}^{N} \alpha_T(i), \quad \alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right) b_j(o_{t+1}), \quad \alpha_1(i) = \pi_i b_i(o_1) \qquad (6.2)$$

The time complexity is longer exponential, but rather cubic. It is cubic since the variables $\alpha_{t+1}\,(j)$ of Equation 6.2 must be computed for every point in time, for every state, and then for every state transition a_{ij} to a state.

6.1.2 Problem 2: Decoding

Decoding is the process of finding the best matching state sequence given an observation sequence. Formally, the best matching state sequence $S^*_{1,T}$ is the state sequence that has the largest joint probability $P(O_{1,T}, S_{1,T}|\lambda)$ with the given observation sequence $O_{1,T}$. The direct approach is to generate all possible state sequences and take the best one, but this has the same problem of a growing number of exponential state sequences as finding the likelihood. The joint probability of $P(O_{1,T}, S_{1,T}|\lambda)$ can be computed as $\pi_{s_1} b_{s_1}(o_1) \Pi_{t=2}^{T} a_{s_{t-1},s_t} b_{s_t}(o_t)$. Equation 6.3 formally describes this problem.

$$S^*_{1,T} = \arg\max_{S_{1,T}} \{P(O_{1,T}, S_{1,T}|\lambda)\} \qquad (6.3)$$

To solve this problem efficiently, the recursive structure of all state sequences must be exploited. By taking advantage of this structure, Rabiner [53] has shown that Equations of 6.4 and 6.5 can be used to efficiently compute the best state sequence. The variable $\delta_t(j)$ keeps track of the joint probability of $P(O_{1,t}, S_{1,t}|\lambda)$ of the current best partial state sequence $S_{1,t}$ for state j at time t. The variable $\psi_t(j)$ keeps track of the best previous state at time $t-1$. To recover the best matching state sequence the variable ψ can be used to follow the best path backwards in time shown in Equation 6.6.

$$\delta_t(j) = \max_{1 \le i \le N}\{\delta_{t-1}(i)a_{ij}\}b_j(o_t), \quad \delta_1(i) = \pi_i b_i(o_1) \tag{6.4}$$

$$\psi_t(j) = \arg\max_{1 \le i \le N}\{\delta_{t-1}(i)a_{ij}\}, \; s_T^* = \arg\max_{1 \le i \le N}\{\delta_T(i)\}, \quad \delta_T^* = \delta_T(s_T^*) \tag{6.5}$$

$$s_t^* = \psi_{t+1}(s_{t+1}^*), \quad t = T-1, T-2, \ldots, 1 \tag{6.6}$$

6.1.3 Problem 3: Training

The purpose of training is to find the optimal parameters of λ that give the highest likelihood. An efficient method for finding the global maximum λ^* has not been discovered yet; however there are methods to find the local maximum for the likelihood $P(O_{1,T}|\lambda)$. The method by Baum et al. [3] will be discussed. Liporace [41] discusses some variants of this problem that deals with probability density functions, but we deal only with discrete probability distributions. Formally, this problem is stated as

$$\lambda^* = \arg\max_{\lambda}\{P(O_{1,T}|\lambda)\}. \tag{6.7}$$

The maximization method by Baum et al. [4, 3] is not guaranteed to find optimal parameters λ^*, rather it improves until a pre-defined threshold is reached. The idea is as follows. We define a function $Q(\lambda, \lambda')$ that has several properties. First, if we maximize Q with respect to λ' then it implies that we maximize $P(O_{1,T}|\lambda')$ locally:

$$Q(\lambda, \lambda') \ge Q(\lambda, \lambda) \Rightarrow P(O_{1,T}|\lambda') \ge P(O_{1,T}|\lambda) \tag{6.8}$$

Second, when one takes the partial derivative of Q with respect to λ', the result is a closed form representation of the parameter λ' in terms of λ, allowing for incremental updating.

$$Q(\lambda, \lambda') = \sum_{S_{1,T}} P(O_{1,T}, S_{1,T}|\lambda) \log P(O_{1,T}, S_{1,T}|\lambda') \tag{6.9}$$

The partial derivative has the following form:

$$\frac{\partial Q}{\partial \lambda'} = \sum_{S_{1,T}} P(O_{1,T}, S_{1,T}|\lambda) \frac{\partial P(O_{1,T}, S_{1,T}|\lambda')/\partial \lambda'}{P(O_{1,T}, S_{1,T}|\lambda')} \tag{6.10}$$

Luckily the partial derivative of $P(O, S|\lambda')$ with respect to λ' has most of its terms cancelled out by its divisor $P(O, S|\lambda')$. This allows us to extract the desired parameter of $\lambda' = \{A', B', \pi'\}$. When solving for a specific parameter, the constraints of each parameter must be held, and Lagrange Multipliers can be used to constrain the solution.

Before continuing, a property of the joint probability $P(O_{1,T}, S_{1,T}|\lambda')$ of an observation sequence and a state sequence should be noted. It can be computed as $P(O_{1,T}, S_{1,T}|\lambda') = \pi'_{s_1} b'_{s_1}(k) \Pi_{t=2}^{T} a'_{s_{t-1},s_t} b'_{s_t}(o_t)$ which then has two useful forms for the first two parameters of λ'.

$$P(O_{1,T}, S_{1,T}|\lambda') = a_{ij}'^{\sum_{t \in \{\tau | s_{\tau-1}=i \wedge s_\tau = j\}} 1} \left[\pi'_{s_1} \Pi_{t=1}^{T} b'_{s_t}(k)\right] \left[\Pi_{t \in \{\tau | s_{\tau-1} \neq i \vee s_\tau \neq j\}} a'_{s_{t-1},s_t}\right] \tag{6.11}$$

$$= b'_i(k)^{\sum_{t \in \{\tau | s_\tau = 1 \wedge o_\tau = k\}} 1} \left[\pi'_{s_1} \Pi_{t=2}^{T} a'_{s_{t-1},s_t}\right] \left[\Pi_{t \in \{\tau | s_\tau \neq 1 \vee o_\tau \neq k\}} b'_{s_t}(o_t)\right] \tag{6.12}$$

Taking the derivative of $P(O_{1,T}, S_{1,T}|\lambda')$ with respect to each parameter of λ' and dividing by $P(O_{1,T}, S_{1,T}|\lambda')$ yields the following equations.

$$\frac{\partial P(O_{1,T}, S_{1,T}|\lambda') / \partial a'_{ij}}{P(O_{1,T}, S_{1,T}|\lambda')} = \frac{\sum_{t \in \{\tau | s_{\tau-1}=i \wedge s_\tau = j\}} 1}{a'_{ij}} \tag{6.13}$$

$$\frac{\partial P(O_{1,T}, S_{1,T}|\lambda') / \partial b'_i(k)}{P(O_{1,T}, S_{1,T}|\lambda')} = \frac{\sum_{t \in \{\tau | s_\tau = 1 \wedge o_\tau = k\}} 1}{b'_i(k)} \tag{6.14}$$

$$\frac{\partial P(O_{1,T}, S_{1,T}|\lambda') / \partial b'_i(k)}{P(O_{1,T}, S_{1,T}|\lambda')} = \frac{1}{\pi'_i} \tag{6.15}$$

Every row of **A** and **B** must sum to 1, and every component of π must sum to 1. Mathematically this is represented as:

$$\sum_{j=1}^{N} a'_{ij} = 1, \quad \sum_{k=1}^{M} b_i(k)' = 1, \quad \sum_{i=1}^{N} \pi'_i = 1 \tag{6.16}$$

Updating any parameter of λ' must be constrained using theses equations. Lagrangian multipliers can be used to find the constrained update equations by taking the derivative of each parameter of λ' and setting the following to zero. First, the state transition equations are derived.

$$\frac{\partial Q}{\partial a'_{ij}} - \gamma \frac{\partial}{\partial a'_{ij}} \left[\sum_{j=1}^{N} a'_{ij} - 1\right] = 0 \tag{6.17}$$

Solving for a'_{ij} yields

$$a'_{ij} = \frac{\sum_{S_{1,T}} P(O_{1,T}, S_{1,T}|\lambda) \sum_{t \in \{\tau | s_{\tau-1}=i \wedge s_\tau = j\}} 1}{\gamma} \tag{6.18}$$

By substituting this equation into the constraint $\sum_{j=1}^{N} a'_{ij} = 1$, the value of γ can be found to be

$$\gamma = \sum_{j=1}^{N} \sum_{S_{1,T}} P(O_{1,T}, S_{1,T}|\lambda) \sum_{t\in\{\tau|s_{\tau-1}=i\wedge s_\tau=j\}} 1, \tag{6.19}$$

and replacing γ back into a'_{ij} yields the following update equation

$$\begin{aligned} a'_{ij} &= \frac{\sum_{S_{1,T}} P(O_{1,T}, S_{1,T}|\lambda) \sum_{t\in\{\tau|s_{\tau-1}=i\wedge s_\tau=j\}} 1}{\sum_{j=1}^{N} \sum_{S_{1,T}} P(O_{1,T}, S_{1,T}|\lambda) \sum_{t\in\{\tau|s_{\tau-1}=i\wedge s_\tau=j\}} 1} \\ &= \frac{\sum_{t=2}^{T} P(O_{1,T}, s_{t-1}=i, s_t=j|\lambda)}{\sum_{t=2}^{T} P(O_{1,T}, s_{t-1}=i|\lambda)} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \end{aligned}$$

where $\xi_t(i,j) = P(s_t = i, s_{t+1} = j|O_{1,T}, \lambda)$. Using similar algebraic manipulations, one gets the following update equations for the other parameters as

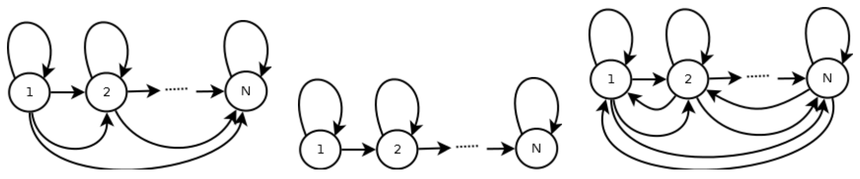
$$\pi'_i = \gamma_t(i),\ b'_i(k) = \frac{\sum_{t=1,o_t=k}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{6.20}$$

where $\gamma_t(i) = P(s_t = i|O_{1,T}, \lambda)$.

6.2 Model Structure and Initialization

Model structures are a way to create a general taxonomy of different types of state to state transition structures. They are distinguishable by the kinds of state transitions they allow, or alternatively which state transition probabilities must be zero. There are three conventional model structures commonly described by authors [13, 36]. **Left-Right** models are defined in terms of the states they can reach. If one orders the state based on state number, then lower state numbers are considered previous states. Left-Right models cannot reach previous states from the current state. Mathematically, they have zero transitions probabilities $a_{ij} = 0$ for all $j < i$. Self-transitions are allowed, but lower state transitions are not. **Left-Right Banded** models are left-right models with a more restrictive transition structure. They cannot reach previous states, and they can either transition to the next state, or perform a self transition. **Ergodic** models have the ability to reach any state from any other state. Figure 6.4 depicts the three models.

Most gestures can be modelled using a left-right-banded transition structure. The Left-Right-Banded transition structure has the ability to model gestures with state sequences where each state is always followed by the next without skipping a state or going back to a previous state. A "Check Mark" gesture for example, may have its first state as defined by a diagonal line, and its second by another diagonal line that is perpendicular to its first



(a) Left Right. (b) Left Right Banded. (c) Ergodic.

Figure 6.4: Hidden Markov Model State Transition Structures.

line. The second diagonal line is always preceded by the first diagonal line. It is natural to structure certain gestures using a left-right banded transition structure, but there are cases where this structure can be disadvantageous. If the gesture has a natural transition cycle then a left right banded transition structure needs more states and possibly multiple models compared to an ergodic structure that allows cycles. Alternatively, one can model a larger cyclic gesture by segmenting the gesture into multiple iterations of a left-right-banded model, but then one needs extra rules to keep track of this.

6.2.1 Model Initialization

Proper model initialization is the ideal to achieve global maximization of the likelihood during training. Unfortunately, current training algorithms only reach local maxima. One possibility is to initialize left-right-banded models as follows.

$$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.5 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}, \ \pi = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix} \tag{6.21}$$

The initial state distribution π is correct, since the initial state is always the first state. The state transition matrix $\mathbf{A}$ and the state-emission matrix $\mathbf{B}$ are initialized heuristically under the assumption that the state transition structure is left-right-banded and the state-emission matrix under uniform distributions allows proper moulding during training. If the model is ergodic, then even the state transition matrix can be initialized uniformly as $\mathbf{A} = \frac{1}{N}\mathbf{1}_{N\times N}$, where $\mathbf{1}_{N\times N}$ is an N by N matrix of all ones.

An alternative to partial and uniform initializations is to use different forms of random initialization. With random model initialization, one might get lucky and find the global maximum, but at best one can find the largest local maximum by testing different starting points. A model can be randomly initialized with heuristics, or without heuristics. Heuristics require setting the parameters of the HMM λ based on some domain-specific knowledge, such as a “Check Mark” gesture being “appropriately” modelled by a left-right-banded structure of two states, and adding some type of random noise to each parameter. Without the use of heuristics each parameter of λ could be set according to a uniform distribution from 0 to 1.

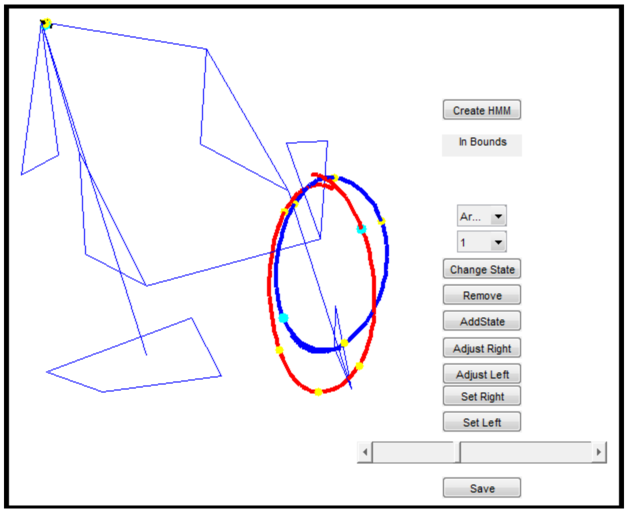


Figure 6.5: GUI Initializer. The thick red and blue paths are traces the left and right hands. The light blue line segments together form a skeleton of the user. In this particular frame, the right hand is above the left hand. Both hands are represented by triangles displayed from three markers per hand.

A Graphical User Interface for Model Initialization

A Graphical User Interface (GUI) was developed to specify the initial parameter values of each gesture. The number of states, and the start and end points of each state can be set using the GUI shown in Figure 6.5. The yellow and blue markers show the starts and ends points of each state. The slider allows the user to scrub through the gesture, and the other buttons allow the user to delete, create, and move the markers. It also allows the user to save the current segmentations, and change the current selected marker (blue) to another marker unselected marker in yellow. The selected marker can be manipulated to change its location. The red curve represents the path of the right hand, the blue curve represents the path of the left hand, and the black curve (barely visible) represents the path of the upper body.

One problem with this type of initialization is that it is time consuming to manipulate the markers. A balancing act between time and accurate settings needs to be considered. One can only initialize a certain fraction of the data before it becomes too time consuming. It might even be useful to initialize a single gesture for every class, rather than for all gestures.

The user can then develop a specific model structure, such as the left-right or ergodic structure based on the manual segmentation of the GUI. If the user wants to develop an ergodic structure, s/he simply needs to specify in the construction algorithm how to incorporate the segmentations into their cyclic structures. Additionally, the left-right model structure is also easily coded. For either model, the emission probabilities can be directly calculated based on the segmented portions of the gesture. For the state transitions, the number of times the state was visited is directly calculated, and the number of times it self transitions or leaves its state can also be directly calculated. Thus, the only difference in model initialization for different transitions structures is the pre-coded way the many segmentations are handled.

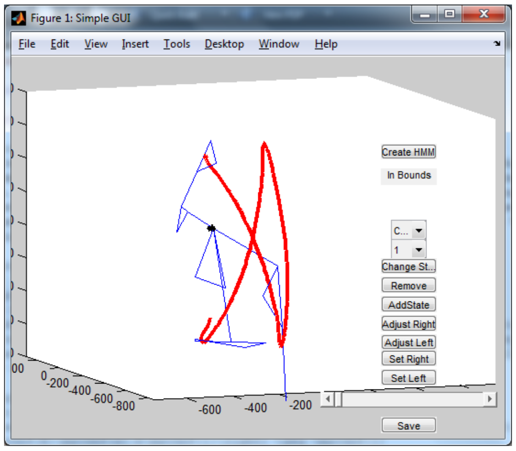
The hypothesis that this type of model initialization is better than uniform model initialization can only be realized if the likelihood in the learning phase increases more relatively to the uniform model initialization. Alternatively, recognition accuracy can be used for comparison between these types of model initialization. Unfortunately a test has not developed to determine if two initialization methods are significantly different.

The Figures 6.6 and 6.7 show an example of how to create state segments for an "x" gesture. One of the reasons for creating HMM initial parameters in this way is to get accurate state representations compared since the emission parameters can be directly calculated as shown in (e) and (f) of Figure 6.7. The button "Create HMM" allows the user to check if they should adjust the parameters of the HMM if it appears wrong.

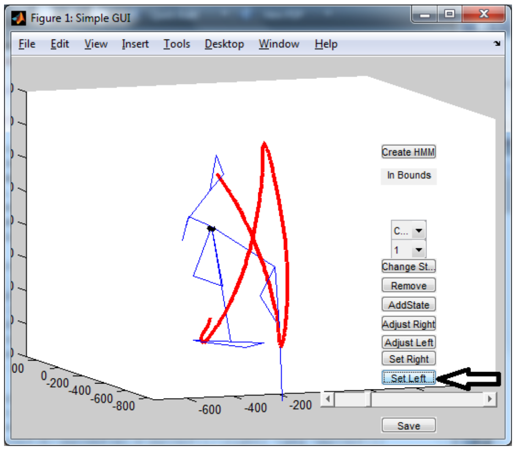
DTW Clustering for HMM Initialization

DTW clustering for HMM initialization is the idea of using the cost of the optimal warping path cost as a distortion measure for clustering groups of similar gestures within each class. Then for each group within a class a single HMM is responsible for modelling its gestures. This idea is applicable to the case when the features of the gestures are allowed to vary greatly in one class, which is likely to cause the observation sequences to be different for two gestures of that class. It is better to split these groups of different observation sequences among two different HMMs so that the emission probabilities can be as large as possible.

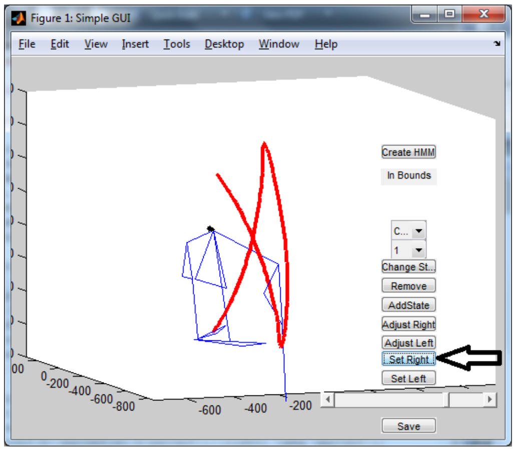
For example, if the developer uses velocity features and Euclidean distance as the distortion measure for Vector Quantization of velocity, then different gesture speeds affect how the observation sequence looks. A quick solution is to use directions, which are invariant to speed changes, or an entirely different set of features. However, if speed matters and must be kept, then one can check for speed violations using thresholds trained from the velocity data, or one can use DTW Clustering to allow the groups within each class to implicitly define the allowable variations in speed from the training sequences themselves. This method requires a large amount of training data to cover all the possible variations in the features. The average likelihood of each HMM group can be used to interpolate the variation in the features that are not covered, but this still does not guarantee good likelihood values since non of the emission probabilities would have a lot of training on those feature variations. Similar approaches have been tested by Keskin et al. [29], and Jang et al. [23].



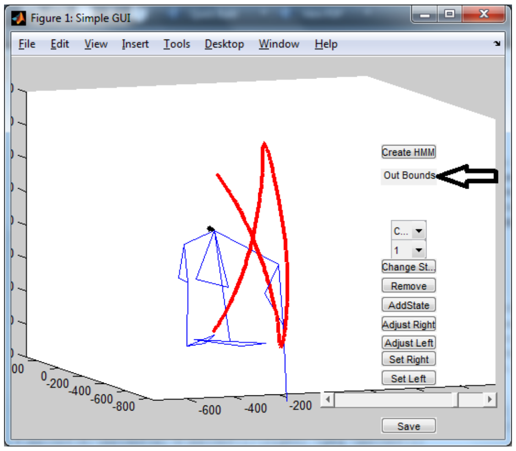
(a)



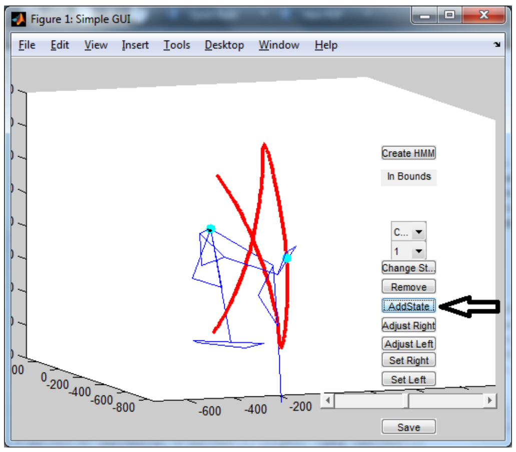
(b)



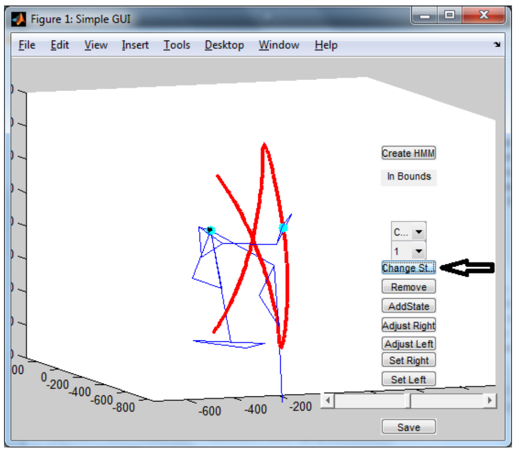
(c)



(d)



(e)

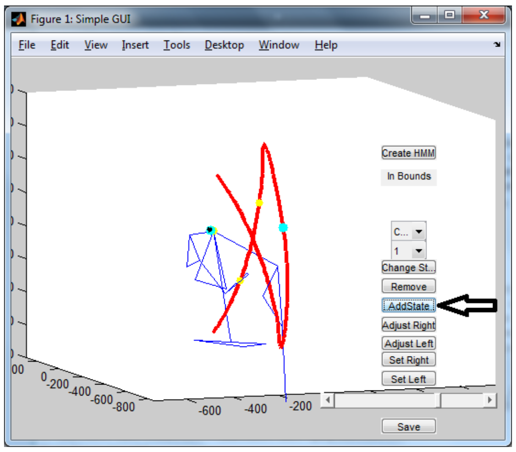


(f)

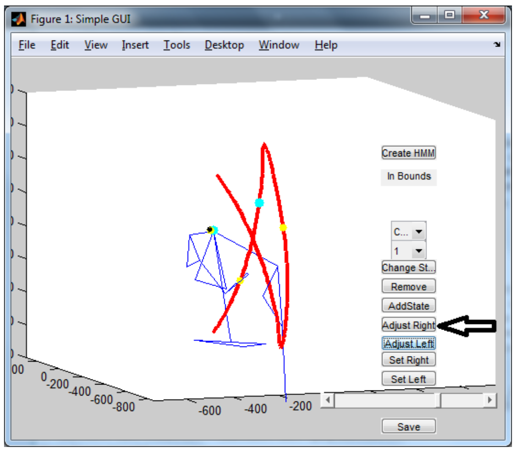
Figure 6.6: Hidden Markov Models initialization example. (a) Pre-segmented. (b) Setting left starting point. (c) Setting right starting point. (d) Out of bounds frame. (e) Adding a marker to divide states. (f) Changing the selected marker location.

One HMM per gesture within each class

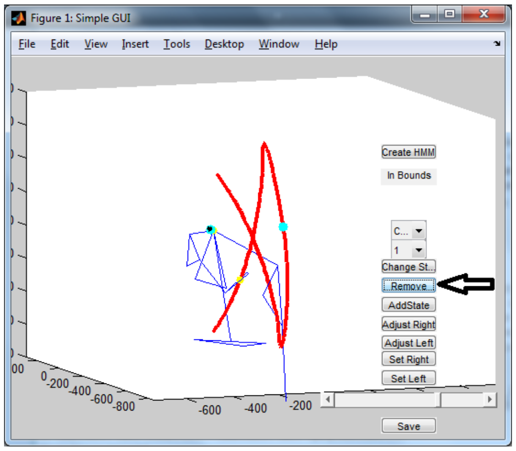
One Hidden Markov Model for every gesture of each class is not useful for accurate recognition. Hidden Markov Models suffer from low training data, and to reduce a single observation



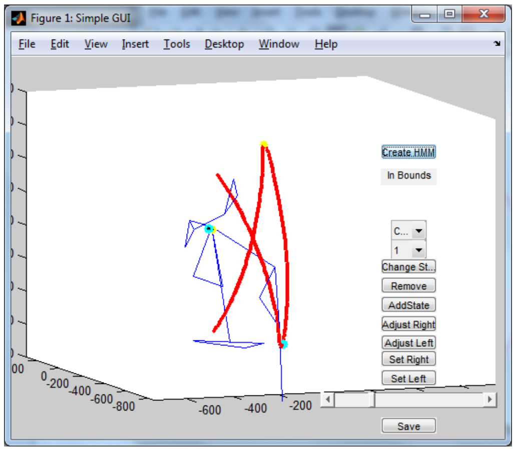
(a)



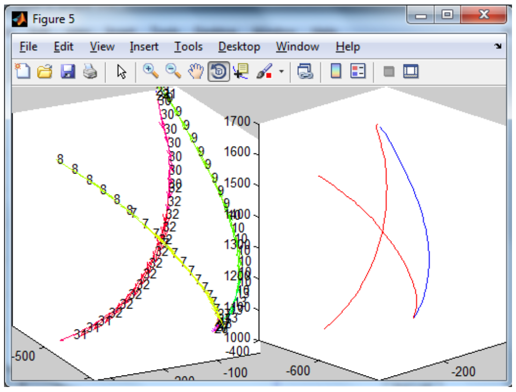
(b)



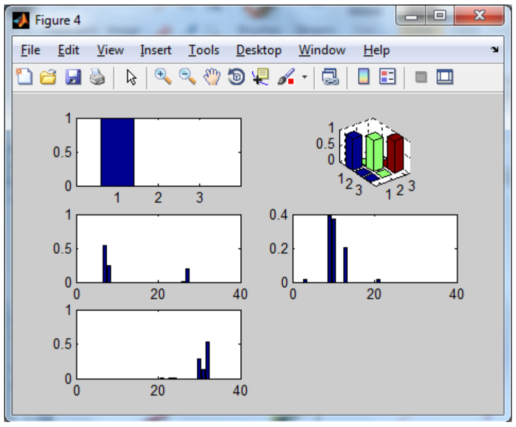
(c)



(d)



(e)



(f)

Figure 6.7: Hidden Markov Models initialization example continued. (a) Adding two more state markers. (b) Adjusting currently selected state. (c) Removing currently selected state from (b). (d) Creating a three state HMM. (e) Observation symbol sequence and alternating red-blue state segments. (f) HMM parameters. Top left: π, Top Right: $\mathbf{A}$, and the others: $\mathbf{b}_1(\cdot)$, $\mathbf{b}_2(\cdot)$, and $\mathbf{b}_3(\cdot)$. Both (e) and (f) were produced from the button "Create HMM".

sequence to each model reduces the chances of accurate emission probability modelling. It is difficult to estimate the correct underlying distribution of a function with a small sample size, and estimation improves with sample size. Template matching methods such as DTW or MD Protractor do not suffer from this problem.

Automatic Model Initialization

Automatic Model Initialization is the idea of automatically specifying the number of states, and observation sequence segmentations. Certain parameters must be specified beforehand to determine when a state should be exited, and when a new state should be created. Figure 4.7 shows how gestures have almost steady sub-sequences of symbols when they transformed through vector quantization, and this property of steady sub-sequences of symbols would be a good place to segment the start and end points of a state. This method must work for all the gestures in the class and not just one since it is not guaranteed that all gestures have the same sub-sequences of symbols. The number of states produced for each gesture might not agree, and this algorithm must account for such differences. An automatic model initialization using heuristics was developed by Glomb et al. [17] using fragment merging.

6.3 HMM Problems

Hidden Markov Models suffer from a variety of deficiencies. They suffer from a large parameter space, from problems with small training data, and from zero valued emission probabilities through learning. Solutions include heuristic parameter settings [17], increasing the size of training data, and adding small increments to each emission probability.

6.3.1 HMM Parameter Settings

Many of the papers with high recognition accuracies for Hidden Markov Models are due to searching for the parameter values that give that best recognition accuracy. Fortunately, the idea of learning parameters from training data is beginning to be supported through heuristic metrics that maximize class separation and class similarity at the codebook level by Glomb et al. [17]. An alternative is to use accuracies that are derived not from test data, but from training data. Splitting the training data into set of training and test data can give an internal recognition experiment that allows for parameter searches independent of the actual test data. However this does not avoid the problem of how far and how precise to search, which adds more model parameters.

6.3.2 Low Training Data

Low training data is problematic for the estimation of probability distributions. A low number of gesture samples runs the risk of larger errors in the estimations of the emissions probabilities. Non-statistical template matching techniques for gesture recognition do not suffer from low training data for this very reason. This is not to say that these template matching techniques do not suffer from low training data, but rather they suffer less since the overall structure and the feature sequences are not integrated into a state representation form during the training process. Katz [27] describes a method to estimate probabilities of sparse data. The idea uses smoothing techniques from biology where they estimate the population frequencies of species given sparse data. Similarly, Rabiner [53] also points out the issue of low training data and the idea of using fewer states per HMM to cope with this problem.

6.3.3 Non-zero Probability Symbols for Every State

Modelling state observation distributions using good initial model parameters and Baum-Welch learning does not incorporate noise as a valid possibility for future gestures. The Baum-Welch algorithm does not constrain each $b_i(o_k)$ parameter from becoming zero valued, but only the constrains that the sum over all observation symbols is one as $\sum_{k=1}^{M} b_i(o_k) = 1$. Noise in the feature space is unavoidable and must be handled properly in the evaluation of the likelihood. If any one of the emission probabilities $b_i(o_k)$ of the HMM is zero then this results in a zero overall likelihood when at least one observation symbol happens to have a zero emission probability over all states.

A simple solution to this problem is to ensure that no state-observation probability can become zero. This can be done by adding a very small increment to each zero valued state-observation parameter. If any state-observation probability is zero, then an observation with a zero probability observation symbol for an HMM will either make the likelihood zero, or make the duration structure of the best path sequence become improper. Therefore it is best to add a small increment to each parameter $b_i(o_k)$ that is zero valued after the Baum Welch method is finished.

6.4 HMM Properties

Hidden Markov Models have several desirable properties. First, certain model structures can cause the evaluation of the likelihood to go from cubic to quadratic in terms of time complexity. Second, the evaluation of the likelihood is a monotonically decreasing function. Finally, the actual advantage DTW has over HMMs may not be significantly different when their time complexities are compared to their accuracies.

6.4.1 Left-Right HMM Efficiency

Left-Right Hidden Markov Models can be computed in less time by skipping over computing impossible state sequences. Left-Right HMMs have the property of not allowing a state transition to a previous state, i.e. $a_{ij} = 0$ for all $i > j$. Therefore decoding and evaluation of likelihoods can be performed quicker. The variable $\alpha_{t+1}(j)$ of equation 6.2 can be computed as $\alpha_{t+1}(j) = \left(\sum_{i=1}^{j} \alpha_t(i)\, a_{ij}\right) b_j(o_{t+1})$ and the variable $\delta_t(j)$ of equation 6.4 can be computed as $\delta_t(j) = \max_{1\le i\le j}\{\delta_{t-1}(i)\, a_{ij}\}\, b_j(o_t)$. Left-Right-Banded structures can be computed even faster by taking advantage of the fact that only a_{ii} and $a_{i(i+1)}$ are non-zero. This gives the following new equations of $\alpha_{t+1}(j) = \left(\alpha_t(j-1)\, a_{(j-1)j} + \alpha_t(j)\, a_{jj}\right) b_j(o_{t+1})$ and $\delta_t(j) = \max\left\{\delta_{t-1}(j-1)\, a_{(j-1)j}, \delta_{t-1}(j)\, a_{jj}\right\} b_j(o_t)$, but this is only true if $j > 1$. If $j = 1$, then $\alpha_{t+1}(1) = \alpha_t(1)\, b_1(o_{t+1})$, and $\delta_t(1) = \delta_{t-1}(1)\, b_1(o_t)$. This allows the variables $\alpha_{t+1}(j)$ and $\delta_t(j)$ to be computed in a constant amount of time.

6.4.2 Decreasing Likelihood with Increasing Observations.

The time evolution of HMM likelihoods does not increase, i.e. the partially computed likelihood of an isolated pattern can either stay at the same value or decrease in value. Figure 6.8 is an example of a set of training gestures evaluated using two different HMMs, and it shows how the likelihood either stays the same or decreases with time. Formally this is stated as $P(O_{1,T}|\lambda) \ge P(O_{1,T+1}|\lambda)$.

Proof.

$$\begin{aligned} P(O_{1,T+1}, S_{1,T+1}|\lambda) &= \pi(s_1) b_{s_1}(o_1) \prod_{t=2}^{T+1} a_{s_{t-1}s_t} b_{s_t}(o_t) \\ &= \left(\pi(s_1) b_{s_1}(o_1) \prod_{t=2}^{T} a_{s_{t-1}s_t} b_{s_t}(o_t)\right) a_{s_T s_{T+1}} b_{s_{T+1}}(o_{T+1}) \\ &= P(O_{1,T}, S_{1,T}|\lambda) a_{s_T s_{T+1}} b_{s_{T+1}}(o_{T+1}), \end{aligned}$$

$$\begin{aligned} P(O_{1,T}|\lambda) &= \sum_{S_{1,T}} P(O_{1,T}, S_{1,T}|\lambda) = \sum_{S_{1,T}} P(O_{1,T}, S_{1,T}|\lambda) \sum_{s_{T+1}} a_{s_T, s_{T+1}} \\ &= \sum_{S_{1,T}} \sum_{s_{T+1}} P(O_{1,T}, S_{1,T}|\lambda) a_{s_T, s_{T+1}} = \sum_{S_{1,T+1}} P(O_{1,T}, S_{1,T}|\lambda) a_{s_T, s_{T+1}} \\ &\ge \sum_{S_{1,T+1}} P(O_{1,T}, S_{1,T}|\lambda) a_{s_T, s_{T+1}} b_{s_{T+1}}(o_{T+1}) \\ &\ge \sum_{S_{1,T+1}} P(O_{1,T+1}, S_{1,T+1}|\lambda) \\ &\ge P(O_{1,T+1}|\lambda) \end{aligned}$$

□

This proof shows that the observation symbol $b_{s_{T+1}}(o_{T+1})$ acts as a penalty to quantify how different the observation symbol is at time $T + 1$. The likelihood $P(O_{1,T+1}|\lambda)$ decreases if any emission probability $b_{s_T+1}(o_{T+1})$ at time $T + 1$ is less than 1. The likelihood becomes

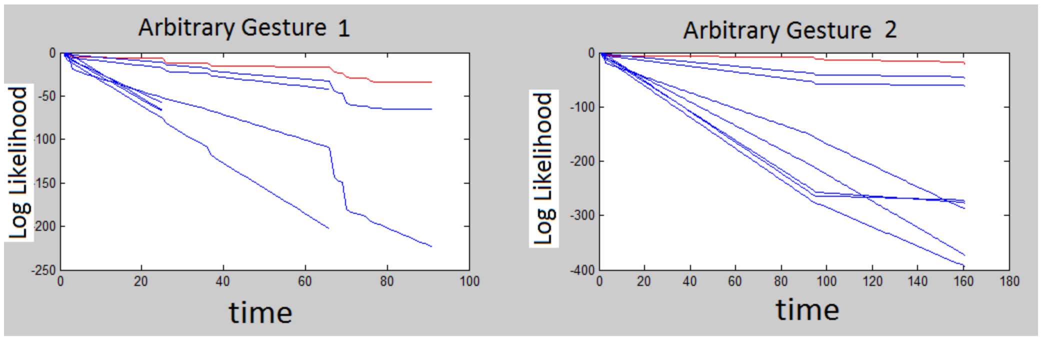


Figure 6.8: The red line depicts the log likelihood of a gesture for the correct model, and the blue represent the log likelihood for the incorrect models. The log likelihood is computed over over incomplete portions of the gesture. This is evidence for the proof of $P(O_{1,T}|\lambda) \geq P(O_{1,T+1}|\lambda)$.

less likely if the observation symbol o_{T+1} is unlikely for any state at time $T+1$. The actual state sequence itself makes no difference to the likelihood since an expansion of the state structure from time T to $T+1$ sums to one as $\sum_{s_{T+1}} a_{s_T,s_{T+1}} = 1$.

Isolated gesture recognition could benefit in terms of efficiency by pruning based on this property. The idea is to prune certain likelihoods by comparing partially evaluated likelihoods and only computing the likelihoods of HMMs with the maximum current likelihood. This requires switching between models whenever the best candidate becomes lower than the next best candidate.

6.4.3 HMM versus Variations of DTW with Complexity and Accuracy

DTW has a quadratic time complexity for evaluation, whereas HMMs have a cubic time complexity. For gesture recognition, many templates were used that caused the time complexity for evaluation of DTW to become cubic. On the other hand, HMMs become quadratic in their time complexity for evaluation when Left-Right-Banded structures were used for gesture recognition. In the experiments that follow, DTW outperforms HMMs in accuracy, but not in computational efficiency. If we can reduce DTW to a single best template, then it may lose it's accuracy advantage, but it becomes quadratic in evaluation. HMMs can perform relatively well given that they are plagued by low training data and their current need for heuristic parameter setting. DTW, on the other hand, is simple compared to HMMs.

6.4.4 HMM with Curvature Instead of Direction Features

Curvature is a rotation invariant features and directions are not. Curvature as it is usually computed, is not scale invariant, however direction features can be. Curvature is also coordinate system independent whereas directions are not. Gestures that have constant curvatures such as circle gestures should require fewer states using curvature as a features, but will require more state using directions. This is because curvature is constant when the change in direction is constant. However, when we use Vector Quantization on curvature we must be careful not to allow curvature to become too large. The reason is that the LBG algorithm will potentially migrate codebook vectors to every single point that has a large value so that it can minimize its' average distortion. We need to place an upper bound on curvature and provide enough VQ vectors to accommodate the range of values between 0 and the upper bound. However, this bound must be reasonable since you may want to distinguish between larger levels of curvature. Directions on the other hand are easier to quantize since the upper bound to each component is 1 and it's lower bound is 0. As you increase the number of codebook vectors you also increase the number of states needed to represent circular-like gestures with more fluid motions. On the other hand the number of states used for curvature remains unchanged. A disadvantage of curvature comes from the fact that it is coordinate system independent and only considers magnitude as a positive value. For example, let us consider a gesture that performs a smooth ninety degree turn to the left, then straight line for a few frames, and then a smooth ninety degree turn to the right in two dimensions. Now image another gestures that differs only by the last turn and instead turns to the left. Curvature will not capture these differences in two dimensions, yet directions will. Depending on the gesture set these differences must be carefully analayzed to choose what features are best for HMM gesture recognition.

6.5 Emission Probability Smoothing in VQ Features Space

Codebook symbols generated from VQ are represented by integers from 1 to M where M is the number of symbols. Each state of an HMM generates a distribution of these symbols. Symbols close in index from 1 to M are not guaranteed to be close in feature space. Therefore a simple smoothing over the symbols directly for a state ignores the proper similarity between codebook symbols. Each probability must be smoothed using distances from the actual feature space itself. Codebook vectors corresponding to each symbol should allow for a proper similarity measure between symbols based on some distance measure $d(\cdot)$. The idea is to use this distance measure as a proper way to smooth the distribution symbols for each state. Gaussian smoothing can be performed using the following formula

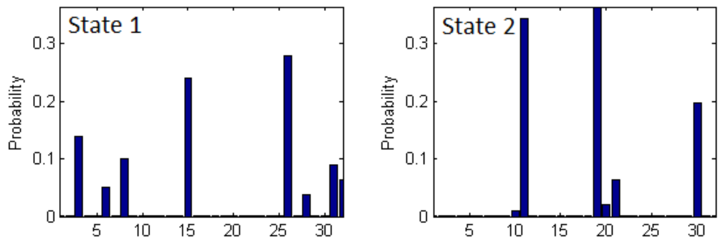
$$b'_i(o_k) = \sum_{j=1}^{M} e^{\frac{-d(\mathbf{A_k},\mathbf{A_j})^2}{2\sigma^2}} b_i(o_j), \quad 1 \le i \le N, \quad 1 \le k \le M \tag{6.22}$$

where N is the number of states. $\mathbf{A_k}$, and $\mathbf{A_j}$ are the codebook vectors corresponding to symbols k, and j respectively. $d(\mathbf{A_k}, \mathbf{A_j})$ is the distance measurement between $\mathbf{A_k}$, and $\mathbf{A_j}$. Euclidean distance is one possible measure represented for $d(\mathbf{A_k}, \mathbf{A_j}) = ||A_k - A_j||_2$, or the cosine angle distance for $d(\mathbf{A_k}, \mathbf{A_j}) = \arccos\left(\frac{\mathbf{A_k}\cdot\mathbf{A_j}}{||\mathbf{A_k}||||\mathbf{A_j}||}\right)$.

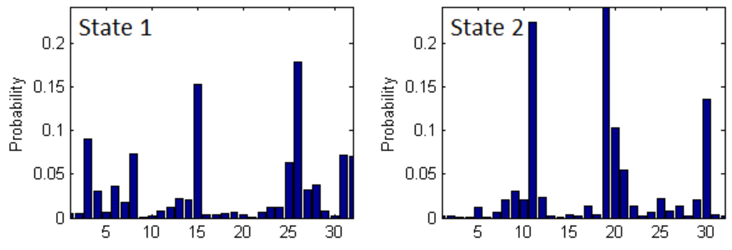
This method does not ensure the constraint of $\sum_{k=1}^{M} b_i'(o_k) = 1$ where each probability distribution sums to one for each state. Each new smoothed state distribution must be normalized by

$$b_i''(o_k) = \frac{b_i'(o_k)}{\sum_{j=1}^{M} b_i'(o_j)}. \tag{6.23}$$

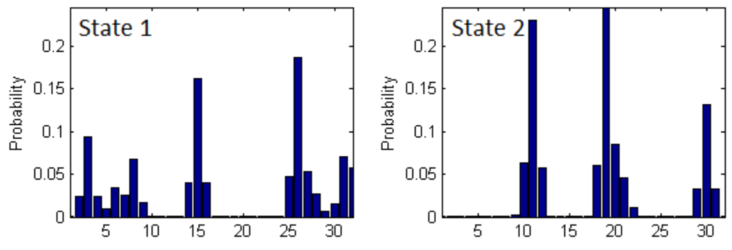
Figure 6.9 shows two distributions that were smoothed using the method presented here and compared it with direct smoothing of the distribution using a Gaussian pdf. Figure 6.9a are the original probability distributions of two states for the class of check mark gestures under the minimal coding condition with directional features. Figure 6.9b was smoothed using the feature space with a Euclidean distance measure and a smoothing parameter σ of 0.6. Notice the new probability values are not smoothed in terms of how close they are by symbol index, but rather by some other definition of closeness which in this case were the distance measurements from the trained codebook of symbols. Figure 6.9c was smoothed by convolving it with Gaussian pdf and a smoothing parameter σ of 0.6.



(a) VQ Emission Normal



(b) VQ Codebook Smoothing with $\sigma = 0.6$.



(c) VQ Gaussian Smoothing with $\sigma = 0.6$.

Figure 6.9: Observation Smoothing versus Gaussian Smoothing using a check mark gesture under the equal coding condition.

Figure 6.10 is an example of a recognition experiment using an HMM with this method of VQ smoothing under different smoothing values of σ. This is a single path class recognition experiment. The green curve uses a smoothing value of σ of 0.07 and the black curve uses a value of 0.05. The green curve will have the observation symbols of the Hidden Markov Models relatively more smoother. The recognition experiment tests the effect of training size on recognition accuracy. The more training data, the better the recognition; however, the green curve initially grows faster. With low training data and a higher smoothing rate the Hidden Markov Model is more robust against new gestures, and with lower smoothing it takes more training data to reach similar performance.

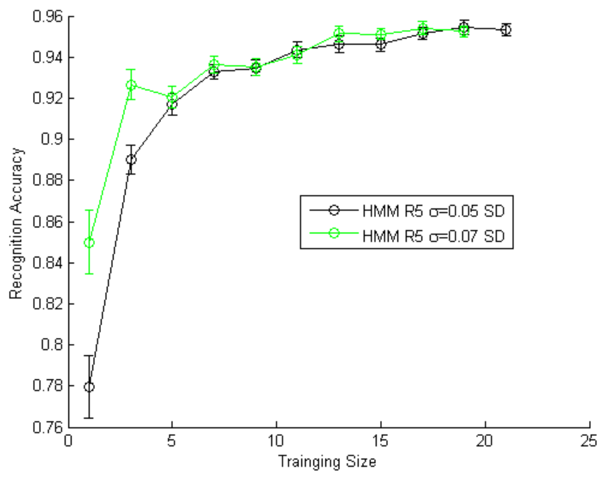


Figure 6.10: Single Path Recognition Experiment under two different smoothing values. Training size is 50% and Test size is 50%.

6.6 Duration Modelling

Duration modelling is the idea of constraining the amount of time spent in any state. Constraining this time can be split into two types, **normalized duration** and **non-normalized duration** modelling. Non-normalized duration modelling restricts the amount of time spent in a state based on the number of time steps, whereas normalized duration modelling restricts the proportion of time spent in a state relative to the observation sequence is. Non-normalized duration modelling usually uses a discrete or continuous variable τ as a measure of how long the current state has been active. There is usually a density function $p_s(\tau)$ or $a_{s,s}(\tau)$ associated with τ that represents how likely it is that state s lasted for τ time steps.

One of the limitation of HMMs is they do not model state durations accurately. This is particularly troublesome for classification because a test gesture may in fact be modelled accurately by a subset of states of an HMM from different classes. For example, an "x" gesture may be modelled using three states, one for the first downward diagonal motion, a second for the upward motion, and a third for the second downward diagonal motion. We could also define another class using the first and third states of this gesture. If we then use the HMM responsible for the "x" gesture to evaluate the likelihood of this new gesture, then it is still possible to gain a high likelihood since most of the best path sequences will quickly skip over state 2 incurring penalties from its emission probabilities. If those penalties are not enough to decrease the likelihood then this new gesture might be considered part of the other class. Likelihood and Viterbi path calculations of a normal HMM do not penalize fast state transitions that do not capture the structure of its training data. For example, a check mark gesture can be segmented into two well-defined states. The first state models the first downward diagonal motion, and the second state models its upward diagonal motion after the downward motion is completed. check mark gesture should stay in the state one third of the total time and two thirds of the time in the second state. A trained HMM on this gesture does not capture this structure, and for this reason gestures that are modelled accurately by the first or second state may end up having a higher likelihood on the check mark HMM even though they do not belong to its class.

Hidden Semi-Markov Models (HSMM) [77, 53] are an adaptation developed to model these structural constraints. The idea is to constrain the duration of a state by allowing it to exist for a pre-defined length of time. The idea is to define a duration variable based on the explicit number of time steps. The general model allows for conditional durations from previous states meaning if we know a state lasted ten time steps then there is a duration density that constrains the next state to last on average 20 steps. Therefore HSMMs can model state durations well. A Hidden Semi-Markov Model uses duration probability density $p_s(\tau)$ as its method of controlling state duration intervals. τ is defined to be greater than or equal to 1, and less than or equal to D, where D is the maximum duration [10, 15, 12, 29, 62, 53, 77]. Yu [77] provides an excellent survey of Hidden Semi-Markov Models that covers theory, special types of Hidden Semi-Markov Models and their usual quartic time complexity. The duration density $p_s(\tau)$ can be estimated using various distribution functions, such as the geometric distribution. Chien et al. [10] describes a way to use Bayesian learning techniques for learning empirical, geometric, Gaussian, Poisson, and gamma duration densities.

HSMMs model state durations on a local time scale [30]. Therefore if a gesture takes longer to execute, then the HSMM will not be able to model significantly large differences in the number of observations. HSMM actually suffer from a lack of global gestural constraint [50], but they are highly suited to gestures that take up approximately the same length of observation sequences.

6.6.1 Variable Transition HMMs

The idea behind Variable Transition HMMs is to allow for arbitrary duration distributions $p_s(\tau)$ or $a_{s,s}(\tau)$ as discussed by Johnson [24]. The Nonstationary HMM [65] and the Inhomogeneous HMM [54] are examples of the Variable Transition HMM. However, both Johnson and Yu [24, 77] describe the Variable Transition HMM as variants of the Explicit Duration HMM and thereby can be classified under the general Hidden Semi-Markov Model described by Yu.

6.6.2 Post-Processor Hidden Markov Model

The Post-Processor approach assumes that we know the start and end point of the observation sequence and therefore know its length. One method to spot gestures is to first identify possible gestures without considering duration, and then apply a duration verification to each candidate gesture. Rabiner [53, 51, 52] described a fast post penalization technique to verify candidate sequences for HMMs. Another type of duration modelling is to bound each state duration with an upper and lower limit [20]. The idea is to ignore states for time values outside the upper and lower bounds. Kim et al. [30] suggested normalizing the length of a known observation sequence and then applying normalized duration bounds to each state allowing for sequences of any length. Power [50] uses a length normalization like Kim et al. [30] and likelihood penalization like Rabiner [53, 51, 52], except that Power applies the penalization to the observation likelihood and not to the Viterbi path likelihood.

6.6.3 Other Duration Models

Duration modelling is a common realization for a variety of state transition models. Context dependent duration modelling [25, 35, 70], expanded state duration modelling [9], dynamic Bayesian network duration modelling [8, 47], finite state machines [5, 6], and others [37, 64, 14, 43, 42, 71]. These are examples of state transition models that developed duration constraints to improve recognition accuracy.

6.7 The Hidden Markov Model Entropy Punisher

Normal HMMs work well with gestures that do not share similar state observation segments across different classes. For example, the first state of a “Check Mark” gesture may be very similar to another state of a different gesture. The HMMs from each class to allow higher likelihoods due to their similar state emission probabilities. This is because HMMs are not able to model state durations directly. For the states with different emission probabilities between both HMMs the likelihood evaluation algorithm only needs to last one time step for these states. This allows for gesture from a different class for a given HMM to bypass any state that does not model it accurately. If a gesture is modelled well by a subset of states of another model, then in order to penalize deviations from the duration structure of its gesture class the model should restrict the evaluation of each state to a specific duration, instead of one step.

Hidden Semi-Markov Models (HSMM) perform a local state duration comparison with the observation sequence. However, as they increase the number of finite local duration values, they increase the memory requirements and complexity [77]. The magnitude of velocity for hand movement has a high range of values, and high sampling rates are crucial for accurate tracking of movements. As the sampling rate increases, so does the number of observations per gesture sequence, and larger maximum state duration parameter becomes necessary.

6.7.1 Proposed Method

We propose the idea of exponentially penalizing state sequences that do not hold a proper global duration structure. This method works using global state duration parameters with the restriction that only left-right banded structures are considered. This approach has the advantage that no extra parameter tuning is required. There exists related work by Power [50], Kim et al. [30], and Gu et al. [20].

The idea is to estimate the duration of each state as a proportion relative to all other states. For example, a 2-state HMM may have the first state occupy 30% of time of the observation sequence, and the second state occupy 70% of time of the observation sequence. Suppose we have a set of observation sequences as $O^1_{1,6} = aabbbb$, $O^2_{1,12} = aaaabbbbbbbb$, and $O^3_{1,4} = abbb$. Given these sequences and the assumption that they are training data for the same class, we can assume a two state model where the first state generates a's and the second state generates b's. If we find a Viterbi path and count the number of state durations relative to their total length we get 2/6, 4/12, and 1/4 as fractional representations of durations for the first state, and 4/6, 8/12, and 3/4 as fractional durations for the second state. We can now estimate the average durations of each state as $11/(12 * 3) \approx 0.306$ and $25/(12 * 3) \approx 0.694$. If a test observation sequence $O^{test}_{1,6} = aaaaab$ is calculated using the HMM trained on $O^1_{1,6}$, $O^2_{1,12}$, and $O^3_{1,4}$ then it has a high likelihood, but violates the structure of the HMM. If we use the Viterbi algorithm and calculate the durations spent in each state we end up with 5/6 for the first state and 1/6 for the second state. This is very different from 0.306 and 0.694, and should be penalized.

To penalize the differences in state durations amongst test observation sequences and the model, we can penalize the log likelihood function in the following way

$$\mathbf{L}(O|\lambda, \mathbf{d}) = \log(P(O|\lambda)) \times (1 + Pen(D(O, \lambda), \mathbf{d})), \tag{6.24}$$

where $P(O|\lambda)$ is the likelihood over the observation sequence, as discussed by Rabiner [53], $D(O, \lambda)$ are the **test state durations** calculated from the best path returned by λ over the observation sequence O, $\mathbf{d}$ are the **trained state durations** calculated from the observation sequences used to train λ, and Pen is a function that takes two state durations and calculates their distance. In the previous example $D(O_{1,6}^{test}, \lambda) = (5/6, 1/6)$ and $\mathbf{d} \approx (0.306, 0.694)$. The **Penalized Likelihood** $\mathbf{L}(O|\lambda, d)$ will become more distant from $\log(P(O|\lambda))$ as the test state durations becomes more different from the trained state durations. $Pen(\mathbf{p}, \mathbf{q})$ should be 0 if $\mathbf{p} = \mathbf{q}$, and increase as $\mathbf{p}$ becomes different from $\mathbf{q}$.

Training the trained state durations $\mathbf{d}$ is done by first training the model λ over the observation sequence with a good initial guess and an assumption of a left-right banded model. Then we use λ to calculate the best path from the Viterbi algorithm over all training observation sequences. From these best path results we can calculate each fractional state duration of every state and average these results to obtain $\mathbf{d}$ which is a vector of length N where N is the number of states for that model.

Let us formalize this by considering $\{O^1, \ldots, O^M\}$ as the training set of observation sequences for a given class. Using Baum-Welch [3] for training, we obtain a model λ. Next we obtain best path results from the Viterbi algorithm for each observation sequence. For each path, we count the percentage of time spent in state 1, state 2, and so on until state N. This gives the training durations $\{\mathbf{d_1}, \ldots, \mathbf{d_M}\}$ such that $\mathbf{d_i} = [d_{i1}, \ldots, d_{iN}]$ for any i between 1 and M inclusive, and $\sum_{j=1}^{N} d_{ij} = 1$. To obtain the trained state duration $\mathbf{d}$, we simply average every state, which is represented as $\mathbf{d} = \frac{1}{M}\sum_{i=1}^{M} \mathbf{d_i}$. The sum over all components of the trained state durations $\mathbf{d}$ is one.

Proof.

$$\sum_{j=1}^{N} \mathbf{d}_j = \sum_{j=1}^{N} \frac{1}{M} \sum_{i=1}^{M} d_{ij} = \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{N} d_{ij} = M/M = 1. \tag{6.25}$$

□

Given that the test and trained state durations are in percentage form, and that the sum over all components of $\mathbf{d}$ is 1, a reasonable distance metric between these two probability like vectors is the symmetric form of relative entropy. The Pen function can be the symmetric form of the Kullback-Leibler divergence function.

$$Pen(\mathbf{p}, \mathbf{q}) = \sum_{i} p_i \log(\frac{p_i}{q_i}) + q_i \log(\frac{q_i}{p_i}) \tag{6.26}$$

Assuming that the training data for each class holds a proper structure, each state should have a non-zero fraction duration such that $\mathbf{d} > \mathbf{0}$. Then the Pen function gives a valid distance value for any test state duration. In our case, $\mathbf{q} = \mathbf{d}$, which implies that $q_i > 0$

for every i value. If $p_i = 0$ then $p_i \log(\frac{p_i}{q_i}) + q_i \log(\frac{q_i}{p_i}) = \infty$ since $q_i \log(\frac{q_i}{p_i}) = \infty$ and $p_i \log(\frac{p_i}{q_i}) = 0$. Zero durations are infinitely penalized since zero durations are not proper duration values. If $p_i = q_i$ then $p_i \log(\frac{p_i}{q_i}) + q_i \log(\frac{q_i}{p_i}) = 0$ since $\log(1) = 0$. When the durations of a state are exactly the same then there is no penalty.

Classification is not performed by finding the hidden markov model with the largest likelihood, but instead by finding the Hidden Markov Model with the largest penalized likelihood:

$$c^* = \arg\max_{\lambda_c}\{\mathbf{L}(O|\lambda_c, \mathbf{d_c})\} \tag{6.27}$$

There is some overhead to classification using this approach. One must compute the Viterbi state sequences, and calculate the durations of each state in that sequence. This adds an additional quadratic time calculations using left-right HMMs which does not increase the complexity of the evaluation problem, but this is no where near the quartic time complexity needed by evaluations of HSMMs.

The main difference to HSMM duration parameters is these are represented in fractional form, rather than integer form, and they work on a global time scale rather than a local time scale. Gu et al. [20] approached state duration modelling in the same way as the HSMM approach by keeping track of durations through an integer parameter. Kim et al. [30] extended this approach by ignoring this integer parameter and instead compared it to the upper and lower bounds described by Gu et al. Unfortunately, there is a huge disadvantage to their approach. It works well only if there is more than one training sample per class, to allow for more Viterbi path variation in duration to ensure proper bounds. Proper bounds for upper and lower parameters of each state should ensure that the observation sequence length has no impact on the likelihood evaluation. More specifically, let $O_{1,4} = aaab$ be the only observation sequence for a class. If the first state produces only a symbols, and the second state produces only b symbols, then the only accurate Viterbi path result is $S = 1112$. Calculating the upper and lower limits for this model for every state gives the following matrix:

$$\begin{bmatrix} 0 & 1 \\ 0.75 & 1 \end{bmatrix}$$

The first row are the lower bounds and the second row are the upper bounds for each state. Each column represents a state. In this case the first state has a lower bound of 0 and an upper bound of 0.75. The second state has a lower bound of 1 and an upper bound of 1. These define the normalized time intervals that an unknown gestures can be evaluated in for each state. That is to say the first state during likelihood evaluation can only be considered for the first 75 percent of the observation sequence. Using these upper and lower bounds and a test observation sequences $O^{test}_{1,8} = aaaaaabb$, then the allowable state time intervals will are

$$\begin{bmatrix} 0 & 8 \\ 6 & 8 \end{bmatrix}.$$

This is an error. All state sequences are impossible when using a left-right banded state structure. Any state sequence in state one can only reach state 2 at time step 7, but time step 7 is invalid and causes state 2 to be unreachable. One solution is to detect when the lower bound of the next state is greater than the upper bound of the previous and either set their limits at equal values such as:

$$\begin{bmatrix} 0 & 1 \\ 0.75 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0.75 \\ 0.75 & 1 \end{bmatrix}$$

Then the allowable bounds will be transformed to

$$\begin{bmatrix} 0 & 6 \\ 6 & 8 \end{bmatrix},$$

which is allowable, but very restrictive since the path always passes through a small state transition window. A solution to this issue is to add a small increment to the upper bound of the previous state and a small decrement to the lower bound of the next state. Then one needs to ask by how much should this increment and decrement be to give the best performance. This causes an introduction of more parameters that need to be adjusted.

Penalizing state durations is not a new idea. Power [50] introduced this idea in 1996 with a connected-digit recognition task. The method proposed here was developed independently from Power out of a need to penalize the state durations of improper observation sequences. Since these methods were developed independently, they do have differences, but the core idea remains the same. The main difference lies in the penalty formula, and the memory structure of state durations. This method uses a single duration parameter per state, whereas Power's uses a smoothed histogram per state. Statistically, Power's method needs more training data since it is estimating a distribution of each state duration, and ours only estimates its mean.

The form of $\mathbf{L}(O|\lambda, \mathbf{d})$ allows developers to use existing HMM software, by computing $\log(P(O|\lambda)$ first, and then simply multiplying a penalizing term. This is also true for Power's equation. However the main difference is the way the penalty affects the likelihood $P(O|\lambda)$. If we take out the logarithm, we get

$$e^{\mathbf{L}(O|\lambda,\mathbf{d})} = P(O|\lambda) \cdot P(O|\lambda)^{Pen(D(O,\lambda),\mathbf{d}))}.$$

The difference between Power's penalization formula

$$P(O|\lambda) \rightarrow P(O|\lambda) \cdot P_W(d_W) \cdot \Pi_{i=1}^{N} P_i(d_i)$$

and the proposed formula is the exponential decreasing nature of the proposed penalty formula. Instead of using a histogram of penalties, we use a template of state durations **d** to compare with.

6.7.2 Results

Dynamic Time Warping, MD-Protractor, Hidden Markov Models, and penalized Hidden Markov Models were compared under two different recognition experiments. The first was a single-path recognition study, and the second was a multi-path recognition study. Directional, positional, and velocity features were used. Both experiments were tested on 50 percent training data and 50 percent testing data. Each experiment used randomized training and testing data for each trail. In both experiments, it is evident that low training data results in very poor performance for HMMs compared to Dynamic Time Warping and MD-Protractor. Both Figure 6.12 and 6.11 show the results for single-path and multi-path gestures, and the results have the recognition accuracy as the vertical axis, and the training size as the horizontal axis. The recognition accuracy is displayed as a function of training size.

Single-Path Results

The original gesture classes consisted of multi-path gestures. Each multi-path gesture was broken into single paths, and each single path gesture was grouped into its own class. This experiment shows the improvement achieved by post-processing duration penalties over non-duration modelling with Hidden Markov Models.

Figure 6.11 are the results of a recognition experiment using single path gestures. Dynamic Time Warping used directional features with Euclidean distance as its distance metric. MD Protractor used translation invariant positional features by translating each to the origin and comparing them using the angle of the cosine between each high dimensional vector representation of each gesture. Hidden Markov Models used directional features with Euclidean distance as the distortion measure for developing the codebook symbols. In addition to the normal HMM, there is also a state duration (SD) penalty model which used the same metrics and parameters as the normal HMM.

The HMMs were tested using different codebook sizes of $2^5 = 32$ and $2^8 = 256$. The State Duration Hidden Markov Models outperforms the normal Hidden Markov Models. Lower codebook sizes tend to handle lower training data better, but they do not have enough distinguishing power when there is sufficient training data.

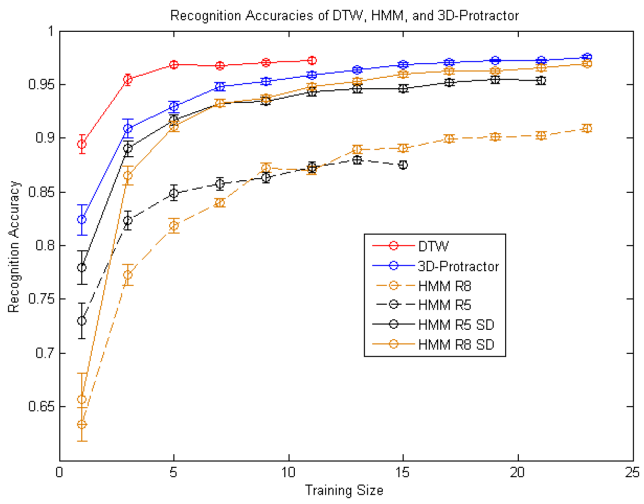


Figure 6.11: Recognition accuracy as a function of time for single path gestures. The standard error in the mean are displayed as error bars and the mean itself is displayed as small circles. Recognition accuracies are in fractional form in the range of 0 to 1, and the training sizes ranges from 1 to 23.

Multi-path gestures

The gestures set of this work consisted of three different paths. The first path is the left hand, the second is the right hand, and the third comes from the centre of the body. The single path gestures all have separate gestures such as move the right hand vertically, swipe the left hand to the left, and moved the entire body to the left. The two-path gestures consisted of two handed gestures such as circular motions of both hands rolling over each other, or the grouping gesture of multiple objects to a common location. In the equal coding condition all gestures are projected into three path gestures, whereas in the minimal coding condition all gestures only coded their path information.

Multi-Path Results

Figure 6.12 illustrates two coding conditions. The first used an **Equal Coding** condition, which involved encoding all paths into a single vector quantization codebook where, each class had the left hand, right hand, and center of motion coded into the codebook. The Minimal Coding condition involved using only the meaningful motion of each gesture to be coded into different codebooks, since single path gestures used fewer features than two path gestures. Single path gestures included left hand, right hand, and center of body gestures. Two path gestures included only two hand gestures.

For Hidden Markov Models, velocity features were used along with cosine angle distance as the distortion measure for vector quantization. In the minimal coding condition there was four codebooks used for each left handed, right handed, center tracked, and two handed gestures. The reason the left handed, right handed, and center tracked gestures used separate codebooks was due to their difference in tracked body location. The single path codebooks used three component velocity features, and the two-handed codebook used six component velocity features.

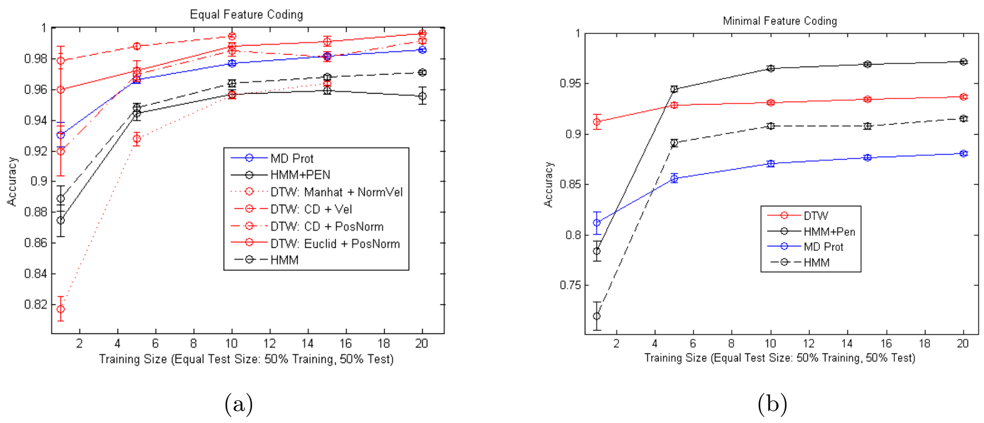


Figure 6.12: Minimal and Equal coding results. MD Prot stand for MD-Protractor, DTW stands for Dynamic Time Warping, and HMM stands for Hidden Markov Model. HMM+Pen was and HMM with the proposed state duration penalty. The standard error in the mean are displayed as error bars and the mean itself is displayed as small circles. Recognition accuracies are in fractional form in the range of 0 to 1, and the training sizes ranges from 1 to 23. (a) Equal coding of gestures. DTW:Manhat+NormVel was the manhatten distance used for the cost matrix, and the normalized velocity of the features. DTW:CD+Vel was the cosine angle distance for the cost matrix, and the velocity features. DTW:CD+PosNorm was the cosine angle distance and the normalized position coordinates used for MD-Protractor. DTW:Euclid+PosNorm was Euclidean distance for the cost matrix, and the same features described for DTW:CD+PosNorm. (b) Minimal coding of gestures. The features used for DTW was the directions of velocity.

The combination of normalized Euclidean distance and directions for DTW was found to work best for the minimal coding conditions. The different combinations of features and distance measures from the equal coding condition were tested in the minimal coding condition and the best combination of them and among others was displayed in the minimal coding results. Oddly, in the equal coding condition, the state duration HMM is slightly outperformed by the normal HMM, and in the minimal coding condition the state duration HMM outperformed the normal HMM. In the equal coding condition, DTW outperforms all other methods with MD Protractor in second place and HMM in third. Under the minimal coding condition, the proposed HMM outperforms all other methods including the normal HMM and DTW.

Figure 6.13 and 6.14 illustrate why template matching techniques such as DTW and MD-Protractor fail under the minimal coding condition, but succeed on the equal coding condition. Both figures show two templates that involve a two handed and a one handed gesture. The assumption here is that the gesture library only consists of these two classes.

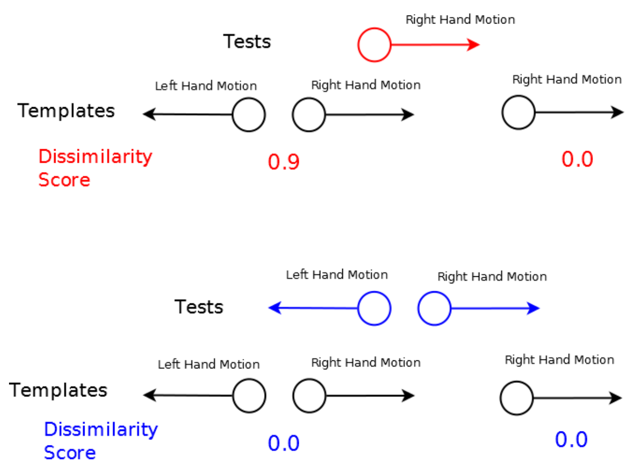


Figure 6.13: Minimal Coding Condition. A two-handed gesture on the left in black, and a one-handed gesture on the right in black for templates. In the minimal coding condition we ignore the other hand and center of body motion for the one handed gesture displayed in red. The two handed test input gesture is displayed in blue and ignores the centre of body motion. The dissimilarity scores are inversely related. The minimal coding condition causes ambiguity for two-handed gestures that are similar to one-handed gestures.

They are then compared across both conditions, which results in four different tests. In the minimal-coding condition of Figure 6.13, the red test gesture excludes motion of the left hand. When the two handed template is compared to the one handed motion, it actually takes what the left hand is doing since it must compare both hands. If the left hand is stationary, as shown in 6.14, then the dissimilarity score is high, since the left hand of the test gesture disagrees with the left hand of the two handed template. Still, we are in the minimal coding condition, and when we next compare it to the one handed template, there is a perfect match. We then focus on the second test, the two-handed blue gesture, under the minimal coding condition and realize there is confusion since the test comparison of the one handed template also returns a perfect match. If we apply the same testing for the equal coding condition there is no longer ambiguity since the inclusion of the left hand in every comparison allows the dissimilarity score to directly show which class the test gesture belongs to.

Under the minimal coding condition, Hidden Markov Models do not suffer from this issue as the template matching algorithms do. Of course, the parameter settings for Vector Quantization must be highlighted since they likely result in the HMMs ability to overcome this ambiguity faced in the minimal coding condition. Under the minimal coding condition Hidden Markov Models may only be better than Dynamic Time Warping due to Vector Quantization. For each multi-path gesture subset of classes, there exists one codebook per group. Two-handed gestures, left-handed gestures, right-handed gestures, and center-of-body gestures each have their own codebook. Within-class separation and between-class separation may be playing a role when comparing models across these different, codebooks

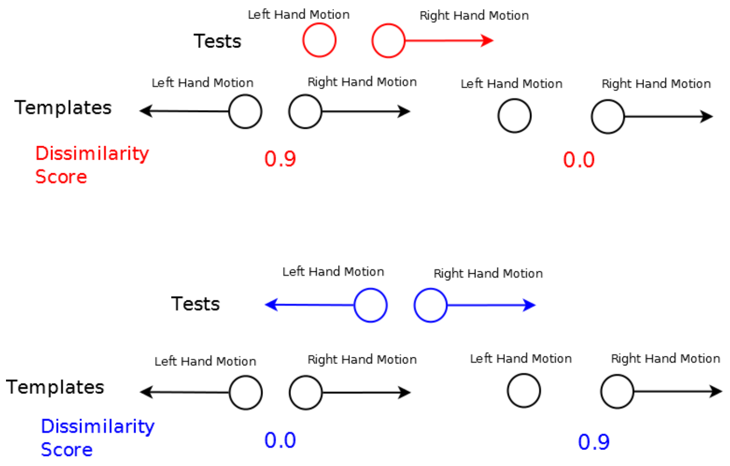


Figure 6.14: Equal Coding Condition. A two-handed gesture on the left in black, and a one handed gesture on the right in black for templates. In the equal coding condition we keep track of all other paths, and in this example we assume that center of body gestures are excluded. The equal coding condition in these types of cases have no ambiguity for test gestures that the minimal coding condition does.

where the level of separation happens to give more priority to the two handed gestures that are being confused with one handed gestures.

We predict that if we take the codebooks developed for the Hidden Markov Models, and apply them to Dynamic Time Warping, where the observation sequences are vector quantized, then we get a similar automatic priority advantage. We can gain speed advantages by storing a matrix representing distances between observation vectors, and simply invoke cost measures using an indexing approach instead of computing all possible distances directly on the features.

Rather than attributing natural priority advantages to vector quantization, it might be the natural normalization of hidden markov model likelihoods being between 0 and 1. HMMs always report a value between 0 and 1.

Unfortunately, the vector quantization codebook size was equal across all codebooks as an initial heuristic which could be just luck that it gave such a natural priority based decision. However, others have been discussing ways to select the number of parameter settings for mono-codebooks [17]. Furthermore if the codebook size is unequal, then performance should drop or increase depending on how this effects class separation between codebooks. In other words, if we choose the codebooks sizes in a way to promote better within-class similarity for two-handed classes, and relatively worse within-class similarity for one-handed classes, then performance should increase as long as the between-class separation across all codebooks has enough discrimination power.

6.7.3 Limitations and Future Work

The proposed method is limited to Left-Right-Banded HMMs. HSMMs are not limited to this and can handle durations of ergodic state transitions. Gesture spotting by state duration penalization can be done to Dui et al. and Kratz et al. framework. We can modify their method to give greater spotting accuracy. However, it would be more impressive is to derive an incremental form of Kratz's extension of Dui's.

It might be more useful to develop one codebook per class, or one codebook per group of similar gestures per class. Glomb et al. [17] suggested that within-class similarity should be high, and between-class separation should also be high to promote better discrimination between classes. Glomb et al. warned that a low number of observation symbols produces poor between-class separation, but a high number of observation produces a lower within class similarity. The main point is that any gesture that is not part of this class will be put on the outside observation symbols of the class, and should be pushed to the lower probabilities of each state.

6.7.4 Summary of Methods

Table 6.1 summarizes and compares the differences between recognition methods. The differences they are compared against are developer cost, time complexity, and recognition accuracies under different experimental conditions.

Method	**Developer Cost**	**Time Complexity**	**Recognition Accuracy (Single-Path)**	**Recognition Accuracy, Equal Coding Condition (Multi-Path)**	**Recognition Accuracy, Minimal Coding Condition (Multi-Path)**
Hidden Markov Model	High	Cubic	Good	Great	Good
HMM Entropy Punisher	High	Cubic	Great	Great	Excellent
Dynamic Time Warping	Moderate	Cubic	Excellent	Excellent	Great
MD-Protractor	Low	Quadratic	Great	Excellent	Borderline

Table 6.1: Summary of recognition methods.

Real Time Recognition

In real time recognition there are two possible circumstances to consider. First one can analyze real time isolated gesture recognition with the user indicating the start and end times of a gesture. Second, one can use gesture spotting where the user does not indicate the start and end times of the gestures, and one thus must search for such candidate points. The first is computationally much easier and is very likely to allow for real time performance. Gesture spotting is a much more difficult problem since one must search for the start and end points without explicit knowledge about there locations.

When we are given the start and end points the only problem left is to simply classify the gesture. Hidden Markov Models can be scaled down to quadratic time evaluations even with the proposed methods of state duration penalties. Additionally, since the likelihood is a monotonically decreasing function pruning may be possible using thresholds speeding the process up significantly. Dynamic Time Warping is a quadratic based algorithm for each template and ends up being cubic for multiple templates. The likelihood of an HMM is computed much quicker than DTW accumulated cost only when the number of states is less than the average template length. Therefore, HMMs have a better chance to achieve real time performance than DTW in its current form used in this thesis. One solution it to limit the number of templates that DTW can match against so that it becomes a constant upper bound allowing it to become quadratic in nature. However, limiting the number of templates will reduce its quality of accuracy so there is a trade off between recognition accuracy and the amount of templates used. HMMs in its evaluation of likelihood will always have the same performance in terms of time despite the amount of training data since it is a parametric method allowing most of the it's computation time devoted to learning. Geometric template matching techniques without rotation invariance gestures sets would be very fast in terms of evaluations of a single template. When comparing it to multiple templates it becomes quadratic. One can also set such an upper bound for the template to improve speed, but the recognition accuracy is likely to fall. All methods can benefit from parallel algorithms. Therefore it is likely that all methods can become real time in the isolated case.

Often what developers do to handle gesture spotting an explicit signal is limit the search window by some other means. For example, one could simply state that all gestures must be performed under 4 seconds and no less than 2 seconds. Another approach is to perform motion detection using a pre-defined threshold of speed. When someone moves their hand fast enough the system detects this and begins to define a time window to search for a gesture. Another approach is to define a volume of region of interaction where a user spatially signals the that they are performing a gesture only when their body or hands are in that space. Some approaches use stationary detection in combination with motion detection to signal the start and end points. Someone might pause for 3 seconds prior to the start of the gesture and pause for 3 seconds at the end of the gesture. Another approach uses specific motion vectors to signal start and end times. All of these approaches make gesture spotting much more similar to isolated recognition and limits the amount of computation needed by the recognition algorithms. This allows would allow for real-time evaluations to be possible, but it constrains the speeds, durations, spatial locations, and naturalness of gesture execution.

If we are to allow for users to naturally perform gestures at their own speeds, durations, and spatial locations then the problem of gesture spotting becomes much more difficult. This changes the time complexities of DTW, HMM, and Geometric Template Matching compared to isolated gesture recognition. Dynamic Time Warping becomes very attractive since it does not increase its time complexity. The accumulated cost matrix [56] can be computed incrementally from the previous point and assume no start or end locations. We simply keep track of the accumulated cost matrix and check at each new frame for an optimal warping path and use its length to calculate the normalized accumulated cost which is a quadratic time operation. The optimal warping path can be found by searching backwards through this matrix and ending when we reach the first row of the template. Of course we would need to multiple accumulated cost matrices for each template. From the perspective of a single new frame the computation is linear to compute the next column of this matrix and becomes quadratic for each additional template. The warping path per template results in a cubic time algorithm. Whenever the normalized accumulated cost passes a pre-defined threshold then a gestures has been spotted. Hidden Markov Model likelihoods can be computed incrementally using an incremental approach for each new feature vector [11]. Like DTW it uses thresholding to determine if a gesture exists. It does not incorporate state duration modelling. They use the Viterbi algorithm to find the starting location which can be quadratic in time complexity, and from this we can compute the state durations of the potential gesture. Both DTW and HMM have incremental computations for their gesture spotting using thresholds, however geoemtric template matching has yet to come up with such an incremental algorithm. One of the problems with creating an incremental algorithm is how does one create incremental algorithms for quaternions and uniform scaling parameter calculations. This is especially difficult if we do not know the start and end locations since every potential gesture must be re-sampled to a fixed length and therefore the re-sampled gestures of one time duration will not be the same points as the re-sampled gesture of a different duration. Therefore, uniform scale invariance and rotation invariance might be best handled through features representations other than rotated and normalized positions.

6.8 New Contributions

We developed a GUI to initialize HMMs, a VQ emission probability smoothing technique, an analysis of the time evolution of the HMM likelihoods, a fix to zero-valued emission probabilities across all states, and a post-processor duration modelling method for HMMs. First, a GUI can be used to initialize HMMs by allowing the user to specify the start and end times of each state for every gesture. It is speculated that this method of initialization can provide a good starting point for training, however testing needs to be done to confirm this. Second, a VQ emission probability smoothing technique can help increase the recognition accuracy for the problem of low training data. By smoothing at the codebook vector level, similar observation symbols can be smoothing based on the VQ distortion measure, and increase recognition accuracies for low training data. The time evolution of HMM likelihoods acts as a monotonically decreasing function and has the potential for early pruning. Early pruning of gestures can be possible since we know the likelihood will never increase. Baum Welch learning can give zero-valued emission probabilities which can be problematic for sensors with noise which essentially reduces any log-likelihood to negative infinity if one observation symbol happens to have be zero valued in terms of its emission probability across all states. One fix we developed was to add a small increment to all emission probabilities after learning so that noise affects the likelihood less and that the gesture still has a chance to be recognized. The last is the post-processor duration model for HMMs that improve gesture recognition accuracy over normal HMMs. It also improves accuracy rates under the Minimal Coding Condition when compared against DTW and MD-Protractor.

Chapter 7

Discussion and Contributions

This thesis focuses on the recognition of motion gestures that form paths through space. One might argue that motion gestures are not important for indicating state changes to a system and that voice commands should be the only method. For example, one might use an "x" gesture to suggest that a window be closed, but the voice command "close" could also be issued. This assumes that the user can speak or the user has full functionality of all body parts. Unfortunately, this is not always the case. Additionally, environmental constraints may not allow speech as a communication tool, for example, in an lab or office environment with multiple people working in the same room. This are just two reasons that motion gestures should be investigated to help improve interactive performance. The following sections summarizes all our contributions to the development of efficient gesture recognition systems.

7.1 User-Defined Gestures

Wobbrock et al. [74] developed a method to allow users to define their own gestures for 2D gesture interfaces. One of the goals for user-defined gestures is to give new users the ability to use an interface quickly using their own intuition of which gestures map to which tasks. By showing a consensus among users for a subset of tasks, it becomes possible to design gesture interfaces with such properties. Gesture sets developed by expert designers often do not agree on what should be the best gestures for a UI, and it might be best to allow users to create gestures for tasks that show agreement. Our contribution is an extension of Wobbrock et al.'s work to three dimensions and an extension to full body free form gestures that can be composed of any combination of motion and pose. We make the contribution of a study to see the types of gestures users developed for a set of tasks that we defined, and if there was any agreement of gestures amongst participants.

Most elicited gestures can be traced to an origin in every day use. Scaling tasks elicited gestures similar to touch-screen gestures on the iPad. Some tasks elicited gestures that require a shape formed through free-space hand drawing or body poses. For example, to gesture “pause” participants were drawing two line segments, or using their forearms to show two line segments, to match the pause symbol on most media devices, namely two lines. The results for the movement tasks suggest that 3D navigations should be done through tracking the movement of the body as a whole, rather than sub-parts of the body.

7.2 Coding

Coding is concerned with the feature representation of gestures. The way the gestures are coded affects the recognition accuracy of different recognition methods. For example, polar and relative distance calculations of features for re-sampled gestures have all the advantage of scale, rotation, and translation invariance. Uniform scale invariance can be achieved through minimizing an MSE between two gestures and eliminate the problems associated with scaling to a cube.

7.2.1 Equal versus Minimal Coding Condition

We contribute an analysis of the disadvantages and advantages between the equal and minimal coding conditions. One of the benefits of touch-screens is the ability to distinguish the number of paths present. With full free-body gestures, however, it is difficult to know the number of paths that are represented in a gesture. For two-handed and one-handed gesture sets, it can be trivial to show which paths should be used for gesture recognition using hand poses that signal which hand to track. However, when multiple parts of the body are tracked, such as the hips, shoulders, elbows, knees, feet, fingers, or toes, then signalling which body parts are meaningful becomes difficult since poses cannot be used. Alternatively, the user could use voice to signal which body parts are relevant in a gesture, but this has some problems. If there is a large number of gestures, the system must wait for the user to vocalize which parts of the body are important. The other problem occurs when the user cannot speak, and therefore cannot inform the interface using speech. One might suggest keeping the non-meaningful parts of their body stationary to indicate that they should not be included, but this raises the problem of natural interaction and decrease output in the number of gestures that the user can perform per minute. The reason why the users number of gestures per second will be less is due to the inability to prepare for the next gesture since they must keep the non-meaningful parts stationary.

The equal coding condition allows recognition algorithms to distinguish which body parts have meaning for each gesture through non-movement of the non-meaningful body parts. The recognition algorithms can implicitly separate gestures by their body parts that add meaning. As stated previously, the problem that arises is the inability to prepare for other gestures that use the non-meaningful body parts. The minimal coding condition provides a more difficult coding method, which does not allow for easy distinguishing between which body parts have meaning per gestures. On the other hand, the minimal coding condition allows preparation of the non-meaningful body parts since those parts are not included during the recognition process for specific classes.

7.2.2 Polar and Relative Distance Features

Only the re-sampled version of single path gestures are scale, rotation, and translation invariant when coded using polar and relative distance features. The initial re-sampling and translation to the origin allows these positional features to become translation invariant. The two polar angles from point to point allows the gestures to be rotation invariant. The relative distance feature then allows to account for scale invariance without the need to minimize an MSE loss function. Also there is no need for quaternions to find the optimal rotation angle and axis. One can compute the average Euclidean distance between the unknown gesture and template to get a dissimilarity score. These advantages are of course true when the start and end points are known, but it becomes difficult to achieve scale invariance when this is not the case. These polar and relative distance features are our contribution to gesture recognition.

7.2.3 Uniform Scale Invariance

The approaches applied to scale invariance that use the average Euclidean distance for current geometric template matchers often scale gestures to a cube. This causes problems for aspect variant gestures and for one- and two-dimensional gestures in three dimensions of space. Finding the optimal uniform scale parameter that minimize the MSE allows all these problems to be solved. Gestures are scaled to template gestures and each template gesture occupies a specific volume. Unfortunately, the average Euclidean distance is not scale invariant and the comparison between different classes must be normalized by scaling each template to a cube, but this must be done in a specific order to preserve the unknown gestures structure. Instead of independently scaling each gesture to a cube, the unknown gesture is first scaled based on the optimal uniform scale parameter. Then the template scaling values are computed and are applied to both template gesture and the uniformly scaled unknown gesture is also scaled using those same scaling values. The unknown gesture has been scaled twice so far, once by the uniform scaling matrix, and a second by the template cube scaling values. Scaling to the parameters of the template cube allows for the unknown gesture to hold its relative structure that it had when it was uniformly scaled. The Uniform Scale Invariance through minimizing an MSE approach is another contribution we make to the geometric template matching area.

7.3 MD-Protractor

We contribute MD-Protractor to the gesture recognition area. MD-Protractor is an M-dimensional gesture recognizer that re-samples an M-dimensional path to ensure groups of multiple 3- or 2-dimensional paths are synchronized in time when they are re-sampled based on an M-dimensional distance. Scale invariance is achieved through using the angle of the cosine distance between gestures. It is a very fast gesture recognizer that extends the uni-stroke recognizers into multiple synchronized paths. It's accuracy is great under the equal coding condition.

7.4 Hidden Markov Models

Hidden Markov Models have low recognition accuracies under low training data, they are unable to model state durations The HMM Entropy Punisher is one method to model state durations, and a VQ emission probability smoothing technique is one method to handle low training data. Vector Quantization may be the key to why HMMs naturally prioritize likelihood scores to classes with more paths, and this may be the reason why HMMs outperform gestures in the minimal coding condition versus the equal coding condition. Smoothing emission probabilities in HMMs may be the key to curing low training data.

7.4.1 Zero Valued Emission Probabilities

Baum Welch learning can result in zero-valued emission probabilities. This can be problematic for sensors with noise resulting in negative infinity log-likelihoods when any observation symbols happens to have a zero-valued emission probability across every state. One fix we developed was to add a small increment to all emission probabilities after learning so that noise affects the likelihood less and that the gesture still has a chance to be recognized.

7.4.2 Decreasing Likelihood with Increasing Observations

The HMM likelihood is a monotonically decreasing function. One can use this knowledge to stop evaluation part way through and quickly eliminate a gesture class from consideration. This assumes that thresholding is a valid method of pruning.

7.4.3 A GUI for Model Initialization

We developed a GUI for model initialization of HMMs. The idea is to allow users to mark the start and end points of states of each gesture. Each gesture can be used to create an HMM using a pre-defined VQ codebook, and display the segments of each state, and a visualization of the parameters of this HMM. This allows users to see how their manual segmentations affects how the initial HMM model will appear, instead of initializing HMMs randomly.

7.4.4 VQ Emission Probability Smoothing

We contributed a VQ emission probability smoothing approach for gesture recognition using HMMs. HMMs usually suffer from low training data. One solution to help with low training data is to smooth the emission probabilities of HMM based on the VQ codebook vectors. It is better to smooth based on the VQ codebook vectors rather than the integer ordering of the observation symbols. With a low number of codebook vectors, smoothing is useful with low training data. With a large amount of training data, it might cause lower performance. A full parameter study would need to performed to answer these questions, but at the moment the initial results appear promising. The analysis should consider the effect of the smoothing parameters, number of codebook vectors, and the training size on the recognition accuracy. This type of study will be a large search across three variables and comparing their combined effects on recognition accuracy. We suspect that, as the codebook size increases and the training size increase, the smoothing parameter needs to decrease in order to maintain the best recognition accuracy.

7.4.5 Duration Modelling

HMMs do not constrain the duration of the state other than to one time step. For example, an "x' gesture as the one in Figure 4.2, may be modelled using three states with one state for the first diagonal motion, another for upward motion, and a third state for the last diagonal motion. Three new classes can be constructed consisting of one state of each of the three states of this "x" gesture. Three more classes can be created by taking different orders of 2 states from the choice of 3 three states. Altogether there are nine classes each with a different HMM. Each HMM of two states will give high likelihoods for those of two and one states, and similarly the HMM of three states will give high likelihoods for all classes. Once duration modelling is included then the likelihood values will be affected such that gestures with improper state structures will have lower likelihoods and gestures with proper state structures will have higher likelihoods.

The HMM Entropy Punisher is a contribution we make to gesture recognition using HMMs. It uses proportions of the total time to model state transition durations from Viterbi path sequences. These sequences allowed for comparisons against trained data to penalize the likelihood of improper state sequence structures. The effect of penalization results in better discrimination amongst gestures with similar states. An additional advantage is the ability to discriminate gestures with the same HMM, but with have different state durations. For example, a new class can be created by having the third state of the "x" gesture twice as long in duration. Even though two different classes have the same HMM their duration state sequence structures are the key to their discrimination. This means the same HMM can be used for two different classes, but the joint combination of HMM and the trained duration vector will result in two different penalized likelihoods. The HMM Entropy Punisher can model classes that differ only in state durations, whereas Hidden Semi-Markov Models (HSMMs) must use separate models to achieve the same effect.

7.5 Comparison between HMM, DTW, and Geometric Template Matching

Our final contribution is a comparison between HMMs, DTW, and Geometric Template Matching methods. HMMs allow for one time step penalization in its state sequence. However, the DTW algorithm forces a test pattern to be penalized for at least the number of time steps that the reference pattern has and therefore DTW is an implicit duration punisher. DTW has a natural structure verification of test patterns when comparing it to its reference pattern. Similarly, other template matching techniques also have the natural structure verification power against test patterns. Geometric template matchers have the natural advantage of preserving gesture structure. The positional features kept the entire structure of the path, and template matching requires that all the features are compared point by point which ensures that the duration of a segment is penalized properly.

For single path gestures the HMM Entropy Punisher showed significant improvements over the normal HMM, yet DTW and MD-Protractor outperformed both. This is evidence that HMMs need a duration constraint when the application domain has an implicit duration structure.

For synchronized multi-path gestures the HMM Entropy Punisher outperformed both DTW and Geometric Template Matchers under the minimal coding condition, but the opposite is true for the Equal Coding Condition. This is most likely due to the way vector quantization affects the HMM likelihood. The assumption is that the within-class similarity and between-class separation allowed a natural likelihood priority to two handed gestures, when interacting with the different codebooks using in the minimal coding condition. This can be tested using DTW on the vector quantization gesture sequences. If DTW gives a natural advantage, then it is apparent that vector quantization allowed such priority advantages. In the Equal Coding Condition, HMMs did not benefit from duration modelling. This is only evidence that the specific gestures used in this study, when coded under the Equal Coding Condition, no longer allow differences in duration structure. This does not mean the equal coding condition eliminates the need for duration modelling in HMMs, but rather in this specific case the gestures no longer have ambiguous durations state structures.

In the Equal Coding Condition, MD-Protractor is great at distinguishing between meaningful parts of the body per gesture, but the non-meaningful parts of the body must be stationary. Furthermore, in the Equal Coding Condition it suffers from true scale invariance. The minimal coding condition for one class preserves true scale invariance for single path gestures, but has problems distinguishing between different meaningful parts of body gestures.

Bibliography

[1] L. Anthony and J. O. Wobbrock. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010*, GI '10, pages 245–252. Canadian Information Processing Society, 2010.

[2] L. Anthony and J. O. Wobbrock. $n-protractor: a fast and accurate multistroke recognizer. In *Proceedings of Graphics Interface 2012*, GI '12, pages 117–120. Canadian Information Processing Society, 2012.

[3] L. E. Baum and J. A. Eagon. An Ineqaulity with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model of Ecology. 1966.

[4] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occuring in the Statistical Analysis of Probabilistic Functions of Markov Chains. 1970.

[5] M. K. Bhuyan, D. Ghosh, and P.K. Bora. Finite state representation of hand gesture using key video object plane. In *TENCON 2004. 2004 IEEE Region 10 Conference*, volume A, pages 579–582 Vol. 1, 2004.

[6] M. K. Bhuyan, D. Ghosh, and P.K. Bora. Threshold finite state machine for vision based gesture recognition. In *INDICON, 2005 Annual IEEE*, pages 379–382, 2005.

[7] H. Brashear, V. Henderson, K. H. Park, H. Hamilton, S. Lee, and T. Starner. American sign language recognition in game development for deaf children. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and Accessibility*, Assets '06, pages 79–86. ACM, 2006.

[8] Feng C. and Wei W. Activity recognition through multi-scale dynamic bayesian network. In *Virtual Systems and Multimedia (VSMM), 2010 16th International Conference on*, pages 34–41, 2010.

[9] C. Chesta, P. Laface, and F. Ravera. Connected digit recognition using short and long duration models. In *Acoustics, Speech, and Signal Processing*, volume 2, pages 557–560, 1999.

[10] J. T. Chien and C.-H. Huang. Bayesian learning of speech duration models. *IEEE Transactions on Speech and Audio Processing*, 11(6):558–567, 2003.

[11] J.W. Deng and H.T. Tsui. An hmm-based approach for gesture segmentation and recognition. In *15th International Conference on Pattern Recognition, 2000. Proceedings.*, volume 3, pages 679–682 vol.3, 2000.

[12] T.V. Duong, H.H. Bui, D.Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *Computer Vision and Pattern Recognition, 2005.*, volume 1, pages 838–845, 2005.

[13] M. Elmezain, A. Al-Hamadi, J. Appenrodt, and B. Michaelis. A hidden markov model-based continuous gesture recognition system for hand motion trajectory. In *19th International Conference on Pattern Recognition, 2008.*, pages 1–4, 2008.

[14] M. Fan, D. Gravem, D. M. Cooper, and D. J. Patterson. Augmenting gesture recognition with erlang-cox models to identify neurological disorders in premature babies. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 411–420, 2012.

[15] J.T. Foote, M.M. Hochberg, P.M. Athanas, A.T. Smith, M.E. Wazlowski, and H.F. Silverman. Distributed hidden markov model training on loosely-coupled multiprocessor networks. In *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1992.*, volume 4, pages 569–572, 1992.

[16] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* 1992.

[17] P. Glomb, M. Romaszewski, A. Sochan, and S. Opozda. Unsupervised parameter selection for gesture recognition with vector quantization and hidden markov models. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part IV*, INTERACT'11, pages 170–177, 2011.

[18] C. Grätzel, T. Fong, S. Grange, and C. Baur. A non-contact mouse for surgeon-computer interaction. *Technololgy Health Care*, 12(3):245–257, August 2004.

[19] T. Grossman, R. Balakrishnan, G. Fitzmaurice, A. Khan, and B. Buxton. Interaction Techniques for 3D Modeling on Large Displays. *Proceedings of I3D'01*, pages 17–23, 2001.

[20] H. Y. Gu, C. Y. Tseng, and L. S. Lee. Isolated-utterance speech recognition using hidden markov models with bounded state durations. *IEEE Transactions on Signal Processing*, 39(8):1743–1752, 1991.

[21] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, 1987.

[22] Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. pages 154–158. 1990.

[23] M. Jang, M.-S. Han, J. Kim, and H. Yang. Dynamic time warping-based k-means clustering for accelerometer-based handwriting recognition. In *Developing Concepts in Applied Intelligence*, volume 363, pages 21–26. 2011.

[24] M.T. Johnson. Capacity and complexity of hmm duration modeling techniques. *Signal Processing Letters, IEEE*, 12(5):407–410, 2005.

[25] Oh W. K. and Chong U. Performance of connected digit recognizers with context-dependent word duration modeling. In *Circuits and Systems, 1996., IEEE Asia Pacific Conference on*, pages 243–246, 1996.

[26] H. Kang, C. H. Lee, and K. Jung. Recognition-based gesture spotting in video games. *Pattern Recognition Letters*, 25(15):1701 – 1714, 2004.

[27] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions onAcoustics Speech and Signal Processing.*, 35(3):400–401, 1987.

[28] N.i Kawarazaki, I. Hoya, K. Nishihara, and T. Yoshidome. 7 cooperative welfare robot system using hand gesture instructions. In *Advances in Rehabilitation Robotics*, volume 306, pages 143–153. 2004.

[29] C. Keskin, A.T. Cemgil, and L. Akarun. Explicit duration models for isolated hand gesture recognition. In *2011 IEEE 19th Conference on Signal Processing and Communications Applications (SIU).*, pages 1169–1172, 2011.

[30] W. G. Kim, J. Y. Choi, and D. H. Youn. Hmm with global path constraint in viterbi decoding for isolated word recognition. In *1994 IEEE International Conference on Acoustics, Speech, and Signal Processing.*, volume i, pages 605–608, 1994.

[31] L. Kratz, D. Morris, and T. S. Saponas. Making gestural input from arm-worn inertial sensors more practical. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1747–1750, 2012.

[32] S. Kratz and M. Rohs. A $3 gesture recognizer: simple gesture recognition for devices equipped with 3d acceleration sensors. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 341–344, 2010.

[33] S. Kratz and M. Rohs. Protractor3d: a closed-form solution to rotation-invariant 3d gestures. In *Proceedings of the 16th international conference on Intelligent user interfaces*, pages 371–374, 2011.

[34] Y. Kuno, T. Murashina, N. Shimada, and Y. Shirai. Intelligent wheelchair remotely controlled by interactive gestures. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 672–675 vol.4, 2000.

[35] O.W. Kwon and C.K. Un. Context-dependent word duration modelling for korean connected digit recognition. *Electronics Letters*, 31(19):1630–, 1995.

[36] Nianjun L., B.C. Lovell, P.J. Kootsookos, and R.I.A. Davis. Model structure selection training algorithms for an hmm gesture recognition system. In *Ninth International Workshop on Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004.*, pages 100–105, 2004.

[37] C. H. Lee and L. Rabiner. A frame-synchronous network search algorithm for connected word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(11):1649–1658, 1989.

[38] H.-K. Lee and J.H. Kim. An hmm-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(10):961–973, 1999.

[39] Y. Li. Protractor: a fast and accurate gesture recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2169–2172, 2010.

[40] Y. Linde, A. Buzo, and R. M. Gray. An Algorithm for Vector Quantizer Design. 1980.

[41] L.A. Liporace. Maximum Likelihood Estimation for Multivariate Observations of Markov Sources. 1982.

[42] P. Morguet and M. Lang. An integral stochastic approach to image sequence segmentation and classification. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 5, pages 2705–2708 vol.5, 1998.

[43] P. Morguet and M. Lang. Comparison of approaches to continuous hand gesture recognition for a visual dialog system. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 6, pages 3549–3552 vol.6, 1999.

[44] M.l Nielson, T. B. Storring, M.and Moeslund, and E. Granum. A procedure for developing intuitive and ergonomic gesture interfaces for HCI. *Gesture-Based Communication in Human-Computer Interaction, Lecture Notes in Computer Science, Volume 2915/2004*, pages 105–106, 2011.

[45] A. Nishikawa, T. Hosoi, K. Koara, D. Negoro, A. Hikita, S. Asano, H. Kakutani, F. Miyazaki, M. Sekimoto, M. Yasui, Y. Miyake, S. Takiguchi, and M. Monden. Face mouse: A novel human-machine interface for controlling the position of a laparoscope. *Robotics and Automation, IEEE Transactions on*, 19(5):825–841, 2003.

[46] H. Nishino, K.i Utsumiya, and K. Korida. 3D Object Modeling using Spatial and Pictographic Gestures. *Proceedings of VRST '98*, pages 51–58, 1998.

[47] U. Nodelman, C. R. Shelton, and D. Koller. Expectation maximization and complex duration distributions for continuous time bayesian networks. *CoRR*, abs/1207.1402, 2012.

[48] D. A. Norman and S. W. Draper. *User Centered System Design; New Perspectives on Human-Computer Interaction.* 1986.

[49] J. Norton, C. A. Wingrave, and J. J. LaViola Jr. Exploring Strategies and Guidelines for Developing Full Body Video Game Interfaces. *Proceedings of FDG'10.*, pages 155–162, 2010.

[50] K. Power. Durational modelling for improved connected digit recognition. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, pages 885–888 vol.2, 1996.

[51] L. Rabiner, J.G. Wilpon, and F. Soong. High performance connected digit recognition, using hidden markov models. In *1988 International Conference on Acoustics, Speech, and Signal Processing.*, volume 1, pages 119–122,, 1988.

[52] L. R. Rabiner, J. G. Wilpon, and B. H. Juang. A model-based connected-digit recognition system using either hidden markov models or templates. *Comput. Speech Lang.*, 1(2):167–197, 1986.

[53] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 –286, 1989.

[54] P. Ramesh and J.G. Wilpon. Modeling state durations in hidden markov models for automatic speech recognition. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 381–384 vol.1, 1992.

[55] I.r Rauschert, P. Agrawal, R. Sharma, S. Fuhrmann, I. Brewer, and A. MacEachren. Designing a human-centered, multimodal gis interface to support emergency management. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, GIS '02, pages 119–124, 2002.

[56] M. Reyes, G. Dominguez, and S. Escalera. Featureweighting in dynamic timewarping for gesture recognition in depth data. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1182–1188, 2011.

[57] O. Rogalla, M. Ehrenmann, R. Zollner, R. Becher, and R. Dillmann. Using gesture and speech control for commanding a robot assistant. In *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, pages 454–459, 2002.

[58] J. Ruiz and E. Li, Y.and Lank. User-Defined Motion Gestures for Mobile Interaction. *Proceedings of CHI'11.*, pages 197–206, 2011.

[59] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43 – 49, feb 1978.

[60] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5):561–580, 2007.

[61] T. Schlömer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, TEI '08, pages 11–14, 2008.

[62] D. Shah, J. Schneider, and M. Campbell. A sketch interface for robust and natural robot control. *Proceedings of the IEEE*, 100(3):604–622, 2012.

[63] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.

[64] Y. Shi, Y. Huang, D. Minnen, A. Bobick, and I. Essa. Propagation networks for recognition of partially ordered sequential action. In *.Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004*, volume 2, pages II–862–II–869 Vol.2, 2004.

[65] B. Sin and J. H. Kim. Nonstationary hidden markov model. *Signal Processing.*, 46(1):31–46, 1995.

[66] B. Starner, T.and Leibe, B. Singletary, and J. Pair. Mind-warping: towards creating a compelling collaborative augmented reality game. In *Proceedings of the 5th international conference on Intelligent user interfaces*, IUI '00, pages 256–259, 2000.

[67] G. B. Thomas, M. D. Weir, J. Hass, and F. R. Giordano. *Thomas Calculus Eleventh Edition.* 2005.

[68] R.D. Vatavu, L. Anthony, and J. O. Wobbrock. Gestures as point clouds: a $p recognizer for user interface prototypes. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, ICMI '12, pages 273–280, 2012.

[69] J.P. Wachus, H.I Stern, Edan Y., M. Gillam, Handler J., and M. Feied, C.and Smitth. A gesture-based tool for sterile browsing of radiology images. volume 15, pages 321–323, 2008.

[70] D. Willett. Context-dependent duration modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05).*, volume 1, pages 421–424, 2005.

[71] A.D. Wilson, A.F. Bobick, and J. Cassell. Recovering the temporal structure of natural gesture. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 66–71, 1996.

[72] B. G. Witmer and M. J. Singer. Measuring Presence in Virtual Environments: A Presence Questionnaire. *Presence, Vol 7, No 3, 225-240*, 1998.

[73] J. O. Wobbrock, H. H. Aung, B. Rothrock, and B. A. Myers. Maximizing the Guessability of Symbolic Input. *Proceedings of CHI'05.*, pages 1869–1872, 2005.

[74] J. O. Wobbrock, M. R. Morris, and A. D. Wilson. User-Defined Gestures for Surface Computing. *Proceedings of CHI'09.*, pages 1083–1092, 2009.

[75] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 159–168, 2007.

[76] J. Yin and Z. Sun. An online multi-stroke sketch recognition method integrated with stroke segmentation. In *Proceedings of the First international conference on Affective Computing and Intelligent Interaction*, ACII'05, pages 803–810, 2005.

[77] Shun-Zheng Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215 – 243, 2010.

[78] C. Zhu, Wei S., and Weihua S. Wearable sensors based human intention recognition in smart assisted living systems. In *Information and Automation, 2008. ICIA 2008. International Conference on*, pages 954–959, 2008.