Towards Objective Surgical Skill Evaluation with Hidden Markov Model-based Motion Recognition

by

Todd Edward Murphy

An essay submitted to The Johns Hopkins University in conformity with the requirements for the degree of Master of Science.

Baltimore, Maryland

August, 2004

Abstract

Modern surgical trainees are often given written examinations to test their knowledge and decision-making skills. However, there exists no widely accepted method for objective evaluation of technical skill. The need for such a method is particularly evident in the young field of robot-assisted minimally invasive surgery, where specific training methods have not been fully established and little is known about surgeons' skill acquisition. Our approach to objective evaluation is based on the assumption that technical skill will reveal itself in the motions used to complete a surgical task. We collect detailed motion data from Intuitive Surgical's da Vinci® robotic surgical system during the performance of such tasks and automatically segment and recognize these motions. With a list of the motions used to complete a task, we may evaluate skill by comparing the number, distribution, and sequences of motions used by novices and experts.

Our methodology is comprised of four major steps. First, a motion vocabulary must be defined. Second, segmenting the data into individual motions is done using the Cartesian velocities of surgeon's input motions. Third, individual motions are automatically recognized using hidden Markov models; recognition rates have been

improved by the application of linear discriminant analysis and a normalization procedure. Using these techniques, recognition rates as high as 85% have been achieved. Lastly, the motions are used to evaluate skill. Skill assessment using motion transcriptions is shown to agree with the skill implied by experience and other objective measures.

Reader: _____

Gregory Hager, Ph.D.
Department of Computer Science
Johns Hopkins University

Advisor: _____

Allison Okamura, Ph.D.
Department of Mechanical Engineering
Johns Hopkins University

Acknowledgements

This work would not have been possible without the direct and indirect contributions of many people. My advisor Allison Okamura was a constant source of optimism and inspiration. Her bright curiosity and warm leadership foster a vibrant, positive atmosphere for research that made my experience at Johns Hopkins outstanding. I am also thankful for Dr. David Yuh's sincere enthusiasm for this research and always-positive response to our progress.

Mike Shumski and Kunal Tandon both assisted in the development of the automatic segmentation algorithms. Mike also deserves a thank-you for the countless hours spent performing tedious manual segmentation of surgical tasks. I received helpful, expert technical advice from Drs. William Byrne and Zak Shafran of the Johns Hopkins Center for Language and Speech Processing (CLSP) on the application of HMMs. Collecting data from the da Vinci® housed in the Johns Hopkins\US Surgical Minimally Invasive Surgical Training Center (MISTC) wouldn't have been possible without the accommodation and friendly help of Sue Eller and Randy Brown.

I owe a huge thanks to Bob Webster, Chad Schneider, Jake Abbott, Lawton Verner, Masaya Kitagawa, and Panadda Marayong for making the Haptics Lab such

a great place. I thankful for their unfailing willingness to help me with research and classwork; I am even more thankful for the lasting friendships we formed.

This work was supported by Whitaker Foundation grant #RG-02-911 and the Johns Hopkins Division of Cardiac Surgery. I am grateful to these organizations for the opportunity afforded by their funding.

Contents

List of Figures

List of Tables

To the guys of Clement 3: Corey, Ryan, and Jim.

Your lives inspire me, your friendship sustains me.

Chapter 1

Introduction

1.1 Motivation

Surgical training programs require the highest possible standards to ensure proper acquisition of skill and the best standard of care for all patients. Surgical skill can be broken down into technical skill—the ability to carry out the manual tasks such as dissection and suturing—and decision-making skills. Decision-making skills are generally agreed to be the more important of the two; these skills are often taught in a classroom setting and are thought to be accurately tested with written examinations. Technical skill, on the other hand, is much more difficult to judge. In most modern training programs the technical skill of surgical residents is largely evaluated using unstructured, subjective criteria applied by senior surgeons.

Although the process is clearly successful, as many talented surgeons are trained and evaluated in this way, clinicians desire an objective method for evaluation of technical skill. Among other reasons, recent rule changes regarding working conditions

for medical residents have reduced the amount of time available for training and, in the eyes of some professionals, have created an even greater need to ensure residents are being taught efficiently and learning well. An effective, accurate method would enable several key initiatives. These include:

- a method by which to evaluate training methods themselves.
- insight into how surgical skills are obtained. Analyzing the effect of different training methods on skill level may reveal the underlying factors at work.
- the possibility of formal certification for specific procedures or techniques. Such a certification could provide confidence for patients, play a role in the approval of techniques by governing bodies, and possibly provide a degree of protection for surgeons for insurance and litigation purposes.
- a better understanding of the mechanisms that contribute to favorable outcomes, assuming that a measure of skill is correlated and validated with objective outcomes.
- a method for evaluation of new tools and techniques for surgery. For example, if the average objective skill measure for a group of surgeons is significantly different when using one tool or technique versus another, this might indicate the effectiveness of that tool or technique.

With these goals in mind, this research was conducted to develop a method for objective evaluation of technical skill in surgery. To demonstrate the method, we have

sought to enable identification of learning curves for surgeons new to a commercially-available robotic system for minimally invasive surgery (MIS). Minimally invasive (such as laparoscopic or thoracoscopic) procedures are performed through three or more small incisions 5–15 mm in size. A small camera is inserted through one incision and long-shafted tools are inserted through the others. While traditional MIS techniques have many benefits for patients, they create a number difficulties for the surgeon.

Intuitive Surgical's da Vinci® system [28] (pictured in Figure 1.1) is designed to overcome many of these difficulties. The system has robotic arms that hold the endoscope and specially-designed surgical tools. The system enables the surgeon to view and control the tools inside the patient by manipulating joystick-like devices at a console several feet away. The motions used closely replicate the same motions a surgeon would utilize if operating directly on the patient during open surgery; with the aid of the robot these operations can be performed through three small incisions roughly 15 mm in size.

Robotic assistance in MIS is still a relatively young technology. As such, many surgeons who might use the da Vinci® or a similar system clinically are already experienced in more traditional MIS techniques. When introduced to the da Vinci® system, the training for these surgeons is typically limited to the function of the system. Notably, it includes very little training regarding surgical techniques specific to such a system. Skill on the da Vinci® system, then, is largely learned through experience. While learning to use the system does seem very intuitive, history does give us some pause. In the first decade of laparoscopic surgery, an underestimation of the difficulty

Figure 1.1: The da Vinci® Surgical System (Intuitive Surgical, Inc.). A surgeon seated at the console controls surgical tools held by robotic arms at the patient's side. (Image used with permission of Intuitive Surgical, Inc.)

of these techniques led to inadequate training and a higher incidence of mistakes. Thus, the application of objective surgical skill evaluation is particularly necessary and appropriate in this new field.

1.2 Key Topics

Our approach to the skill-evaluation problem is to exploit the robotic nature of the da Vinci® to gather accurate motion data from a surgeon during performance of a surgical task. We desire to automatically recognize a set of high-level motions used during the task. With this list we may evaluate skill by making comparisons between novice and expert users, such as the total number of motions used and the

distribution of motions used by surgeons from each group. This skill evaluation must be correlated with a measurable functional outcome of the task.

In the pursuit to achieve automatic motion recognition there are perhaps two choices which have the greatest significance. The first is the recognition technique. For this purpose we have selected hidden Markov models, discussed fully in Section 2.1. The second lies in the definition of the motion "vocabulary," those motions the system will be capable of recognizing. Unlike speech recognition, with which this task shares many similarities, there exists no predefined vocabulary. Definition of the motion vocabulary could be done at several different levels. A lower-level vocabulary could include such motions as "moving forward" and "pushing away," while a higher-level one may include motions like "tying suture" or "dissecting vessel."

Numerous factors guide the choice of vocabulary. We desire a vocabulary that has an appropriate level of generality, so that the vocabulary remains relatively small yet comprehensively includes all possible motions. Alongside generality is a desire for portability, meaning that the vocabulary can be used across multiple domains. While artificial intelligence techniques could possibly be used to automatically define a vocabulary, we also desire for the motions in the vocabulary to be meaningful and intuitive to both ourselves and the surgeons who may potentially use such a system–a characteristic not guaranteed by algorithmic techniques. If a vocabulary meets all of these criteria, it will be evaluated further by its effect on our ability to automatically recognize the motions that comprise it.

1.3 Previous Work in Surgical Skill Assessment

There exists a great demand for objective skill assessment in surgery, as noted by many authors. Some of the most prominent and vocal advocates have been Darzi and colleagues from Imperial College London. A number of articles from these authors [9, 10, 11, 33, 39, 48] provide a good overview of the motivations for objective assessment and the variety of systems developed for this purpose. These papers also serve to guide the reader towards other writings from medical professionals on this topic.

There exist several different approaches to the skill evaluation problem: 1) structured human grading, 2) low-level data analysis, and 3) methods for higher-level surgeon/procedure modeling.

The Objective Structured Assessment of Technical Skills (OSATS) system designed by Martin, *et al.* [38] falls in the category of structured human grading. OSATS tests are conducted using a series of stations where trainees perform surgical tasks and are rated by an expert observer using both a task-specific checklist and a global rating scale. Other researchers have used this system as a reference for assessing the performance of automated evaluation systems. Although OSATS has many benefits over typical, unstructured subjective assessment, the grading at each station is done by a single human observer, introducing the possibility of bias.

Computerized and virtual reality training systems for surgery have gained increasing acceptance and sophistication in recent years. These tools lend themselves well to collecting data and providing objective scoring of some kind. The MIST-VR laparoscopic trainer is one of the earliest and most widely-used of these systems [24, 25].

The software in this system and a survey of other work in the field reveals systems that perform low-level analysis of the positions, forces, and times recorded during training on simulator systems [8, 42, 54].

Similar analyses are at the core of a system developed by Darzi and colleagues, the Imperial College Surgical Assessment Device (ICSAD). ICSAD uses electromagnetic markers placed on a trainee's hands to track the movements during performance of a surgical training task. The system software uses the motion data to provide information about the number and speed of hand movements, the distance travelled by the hands, and the time taken for the task. The technical details regarding the software for data analysis are found in [13]. In contrast to our work, which seeks to actually identify the motions used during task performance, the ICSAD system simply counts the number of motions, using hand velocity as the segmentation criteria. ICSAD has been validated and used extensively in numerous studies, e.g. [13, 14, 36].

Verner, *et al.* [51] collected data from the da Vinci® system during performance of a training task by several surgeons. Their analysis also examined tool tip path length, velocities, and time required to complete the task. Recently, the ICSAD analysis has also been applied to data collected from the da Vinci® [29].

Although not related directly to skill evaluation, Cao and MacKenzie [6, 37] published the results of their analysis of laparoscopic/endoscopic surgeries. In these studies they identify a small set of motions that are used to complete all such tasks. Rosen, *et al.* also performed a "task decomposition" of laparoscopic surgeries [46, 47], and independently defined a set of motions very similar to that of Cao and MacKenzie. These lists closely mimic the motion vocabulary which must be defined for our

own system.

The work of Rosen and colleagues can be categorized into the third type of skill assessment systems, that of higher level modelling. In fact, their work has many interesting parallels to our own. In both [46] and [47], an instrumented laparoscopic tool was used to collect force/torque data during performance of two surgical tasks by both expert and novice surgeons. In [46], the force/torque signatures for a set of five basic surgical motions were identified and the force/torque data was grouped into clusters. The first step of the analysis showed significant differences in the levels of forces used by experts and novices. In the second phase of analysis, a Markov model (not a *hidden* Markov model) was developed for each step required to complete the surgical tasks. The observations produced by these known states were discrete magnitudes of the cluster centers taken from the force/torque data. A manual video analysis was used to identify transitions between the states during surgery, and thus define a model for each surgeon in the study. A statistical distance between models was used to identify surgeons as experts or novices. The second study [47] followed a very similar approach as the latter half of the first, this time using pseudo-hidden Markov models for comparisons. In this study the authors were also able to identify differences in skill between residents at different stages in their training. The study identified these results as "learning curves," although the study did not track the performance of individual surgeons over time, so acceptance of these results as learning curves requires some assumptions. Our work differs fundamentally from all of the studies done by Rosen, *et al.* in that we train many models—one for each gesture or motion—and seek to assess skill through a direct analysis of all the motions used in

the completion of a task.

1.3.1 Application to Virtual Reality

It is also appropriate to consider the applications for skill evaluation in virtual reality. While the ultimate goal of our research is to accomplish skill evaluation for tasks performed in the real world, there is an ever-growing body of virtual reality surgical training systems. These tools, such as laparoscopic surgical simulators, open the door to a host of techniques not available with traditional training methods. For example, it is possible to create training scenarios that contain important complications that are rarely encountered in practice. Other benefits include the low cost of repetition, the opportunity to fail without consequences, and (potentially) increased realism in comparison to traditional training methods such as synthetic or animal models. An additional feature of these computerized systems is the wealth of data that may be collected during a training session. Presumably this data can be used to develop meaningful and objective metrics for skill, but in many applications the best way to do so remains unclear. A sampling of previous work in the medical field reveals systems that perform low-level analysis of the positions, forces, and times recorded during training on simulators and teleoperation systems [8, 47, 51, 54]. The results presented in Chapter 2 were used to validate our overall approach. They also validate the use of motion recognition as part of the evaluation criteria in virtual reality systems. Therefore, although our ultimate objectives are not specifically aimed at virtual reality, whatever successes we achieve here will likely have direct application

to such systems.

1.4 Thesis Organization

This essay is organized as follows. Chapter 2 presents the theoretical foundation for hidden Markov models (HMMs) and the results of applying these models to motion recognition for a dynamic task performed in a virtual environment. Chapters 3 and 4 address the motion recognition problem as two separate sub-problems of segmentation and recognition. Segmentation is the process of identifying the boundaries between motions. Recognition is the process of identifying which motion occurred between these boundaries. As elaborated in Section 2.1.1, continuous HMM recognition systems perform these two tasks simultaneously. However, this approach did not produce high recognition rates in our application, and following our preliminary work we adopted an isolated architecture. So, while Chapter 3 discusses the methods for automatic segmentation, Chapter 4 naturally covers the isolated recognition process and presents our results for recognition of surgical motions. Chapter 5 presents the variety of ways that automatically recognized motions can be used for objective skill evaluation using data from our preliminary work and the tasks described in Chapter 3. Chapter 6 contains the conclusions of this research and discusses areas for future work.

1.5 Thesis Contribution

This thesis describes the following contributions:

- methods for automatic segmentation of robot-assisted surgical motions.
- guidelines for selecting a motion vocabulary for robot-assisted surgical tasks and methods for automatic recognition of these motions using hidden Markov models.
- guidelines for the use of motions in objective evaluation of technical surgical skill.

Chapter 2

Background & Preliminary Work

The goal of this research is to develop methods for objective evaluation of technical surgical skill. Our approach to this greater problem is to use automatic motion recognition as a means for developing technical performance indices. This chapter presents the theory of hidden Markov models (HMMs) and how they are applied to motion recognition. It also describes the preliminary work to validate an HMM-based motion recognition approach to skill evaluation. In this work, users perform a dynamic task in a virtual environment using a three dimensional haptic device for motion input. Motions used during performance of the task are identified and models are trained for each motion. In turn, these models are then used to automatically recognize motions executed during additional repetitions of the task.

2.1 Hidden Markov Models

It is likely that automatic motion recognition could be achieved with numerous techniques; this essay describes the efforts to accomplish this problem through the use of hidden Markov models. A hidden Markov model operates under the premise that a system may be described as being in one of a set of distinct states. The observable output of the system is a probabilistic function of the state the system is in. As time progresses, the system will change state. The state changes are also controlled by a probabilistic function.

Another approach to modeling surgical motions would be to identify a particular "signature" from position or force information, much like that used by Rosen, *et al.* [46, 47]. Such deterministic models must explicitly encode the effects of noise, disturbances, etc. HMMs, on the other hand, are stochastic models that seek to predict the output of the system based on past observations.

HMMs have been applied extensively to recognition problems in several domains. The widest application has been in speech recognition (see [45] for an overview). They have also been used for recognizing driving behavior [35], handwriting [30], human motion [4], sign language [50] and other gesture recognition [44, 55]. Previous work in our laboratory used HMMs for gesture recognition in a cooperative (admittance control) human-robot surgical system [31] to provide appropriate assistance in the form of virtual fixtures [34]. The manipulation environment was far more constrained than in the application presented here, and only the forces applied by the user were included in the observations.

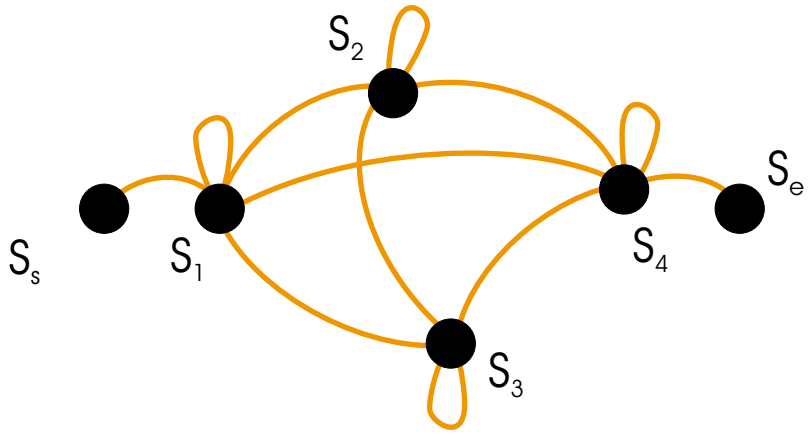

Figure 2.1: A 4-state HMM as implemented by the HTK. The model has starting (S_s) and ending (S_e) states that produce no observations but facilitate transitioning to other models.

HMMs have several attractive qualities for our application:

- they have been used extensively in speech recognition applications and the basic theory as well as many extensions have been thoroughly defined.
- model training is supervised—that is, the models are explicitly defined by us, the researchers. This allows us to ensure that each model has some real physical significance, a quality which we desire.
- HMMs capture the time history that is an essential component of physical systems
- the availability of toolkits and the body of previous work facilitates implementation of HMM theory

The structure of an HMM is illustrated in Figure 2.1. While each HMM consists of a network of states, an entire system (or task) may be understood as consisting of a higher-level network of HMMs. An HMM Θ is described by three components:

the state transition probability distribution matrix A, which defines the probability of transitioning from one state to another; the observation symbol probability distribution matrix B, which defines the observations a state is likely to produce; and the initial conditions of the system π. With this notation, a model Θ can be succinctly defined by writing $\Theta = (A,\ B,\ \pi)$. Note that the observations produced by a state are typically the values of several variables in the system; together these values form what is known as the observation vector.

Rabiner provides an excellent overview of the basic theory behind HMMs and references to many of the original works in [45]. There he explains the three basic questions that emerge when using HMMs to model real-world systems:

- Problem 1: Given the observation sequence $O = o_1, o_2, \ldots, o_t$ and a model $\Theta = (A,\ B,\ \pi)$, how do we efficiently compute $P(O|\Theta)$, the probability of the observation sequence given the model?

- Problem 2: Given the observation sequence $O = o_1, o_2, \ldots, o_t$ and a model $\Theta = (A, B, \pi)$, how do we choose a corresponding state sequence $Q = q_1, q_2, \ldots, q_t$ that best explains the observations?

- Problem 3: How do we adjust the model parameters $\Theta = (A,\ B,\ \pi)$ to maximize $P(O|\Theta)$?

2.1.1 Application of HMMs to Motion Recognition

The questions identified by Rabiner have direct applications in motion recognition. Problem 1 is essentially the recognition problem. Given the model Θ for some motion

and the series of observations O made from an unknown motion, solving Problem 1 reveals the probability these observations were produced by the model Θ. The so-called Forward-Backward procedure introduced by Baum and Egon [1, 2] is used to solve this problem.

HMM recognition systems can be classified into two groups, either continuous or isolated (note: HMM systems may also be classified in terms of continuous or discrete, relating to assumptions about continuous or discrete random variables—this is a different meaning of continuous). In continuous systems, the data input to the system may contain multiple motions. Isolated systems require segmenting the motions in advance. Depending on the system type, Problem 2 may have analogues in our application on two levels. In both cases we are interested in knowing the best state sequence for the purposes of recognizing each separate motion, but in the continuous case we are also concerned with knowing the most likely sequence of motions used during a task—that is, the most likely sequence of models. Fortunately, the Viterbi algorithm [52, 21], which is often used to solve this problem for state alignment, extends well to a network of models.

Finally, the answer to Problem 3 tells us how we can define appropriate models for each of our motions—models that we will need for the solutions to Problems 1 and 2. The approach to solving this problem is to use a series of observations known to be from a particular motion along with an iterative procedure such as the Baum-Welch method (a special case of the Estimation Maximization algorithm [16]) to estimate the model parameters.

Analogues to Speech Recognition

As noted earlier, HMMs have been applied extensively in speech recognition. As a result, most of the literature regarding HMMs and many of the tools for implementing them are specific to speech recognition. For the uninitiated, the domain-specific terminology can be obfuscating, so this section exists to show the connection between our approach and their typical use.

Most modern speech recognition systems model speech at the phoneme level, where one HMM is trained for each phoneme. A phoneme is the smallest phonetic unit in a language; English has 42 phonemes. In spoken language, phonemes are often used in similar groupings, so in some cases additional models are trained that contain up to three phonemes, called "triphones." The observations used to train and recognize these models are a parametric representation of the speech signal. The most common choice is mel frequency cepstral coefficients [15].

One benefit of phoneme-level modelling is that additional words may be added to the system vocabulary—the list of words it can recognize—simply by defining the phoneme sequences which are used to pronounce the word. Notably, adding this additional capability does not require training additional models. To improve recognition for complete sentences, many systems also rely on a language's grammar. By studying large volumes of text, it is possible to define the probability that one word will follow another, and this information may be used to weight candidate words.

Motion recognition with HMMs follows a similar structure. Basic motions or "gestemes" could form the basis for all higher-level motions. However, it is the task

of the system designer to define these basic motions, as well as the higher-level motions that would form a complete vocabulary. Until these are determined, it is impossible to identify or create a grammar defining how motions are used in context with one another. It is not known what observations would be best for motion model training and recognition, but much as frequency seems an appropriate choice for speech applications, positions, velocities, and forces are a natural choice for motions.

2.2 A System for Testing HMM-based Motion Recognition

Although hidden Markov models have been used for motion recognition applications before and reported in the literature, we desired to validate our approach using HMMs to recognize motions used in a dynamic, unconstrained task analogous to surgery. For this purpose we developed a system involving a three dimensional haptic device and a virtual environment. The device, pictured in Figure 2.2 (3GM, Immersion Corporation), is fitted with a modified laparoscopic tool (Auto Suture Endo Shears).

Interfacing with the 3GM haptic device is accomplished through an Immersion Impulse PCI card. A Hall-effect sensor on the scissor-like handle of the laparoscopic tool is used to determine the position of the gripper handle, and this data is obtained through a custom A/D card using the computer's parallel port. A representation of the laparoscopic tool and an end-effector are drawn in the virtual environment (shown

Figure 2.2: The 3GM haptic device with laparoscopic tool handle attached

in Figure 2.3), where the user interacts with other objects. The virtual environment was created with Microsoft Visual C++ and runs on the Windows 2000 operating system.

The virtual environment is contained within a two-dimensional box. The limits of the environment are shown with dark lines, and forces displayed to the user by the haptic device prevent the tool tip from moving outside these boundaries. In addition to the tool, the environment contains a moving target (a thin rectangle) and a ball that can be picked up, carried, and thrown with the gripper. The target moves continuously up and down in a regular sinusoidal pattern. The ball behaves much like a ball in the real world: it is subject to a constant downward acceleration from gravity, viscous damping in air, and it will bounce off of the target and the

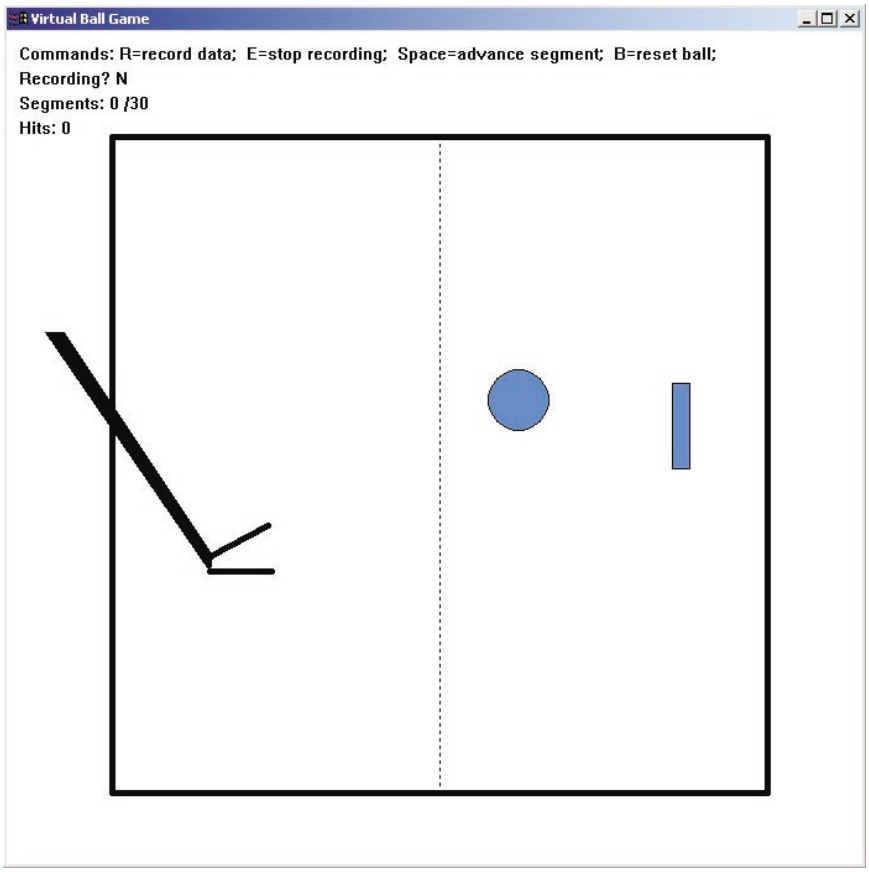

Figure 2.3: The virtual environment. Using the tool, subjects grasp and throw the circular ball at the small rectangular target

floor. However, if it strikes either the left or right wall, it "sticks" to the wall and falls to the floor. The goal of the task is to hit the moving target three times by throwing the ball from behind the dotted line drawn vertically through the middle of the environment. If the ball misses the target, it strikes the right wall and falls to the floor, where it must be retrieved for another try. If the ball hits the target it bounces back to the left, where it will eventually settle in the corner. All users are instructed to refrain from catching the ball in mid-air but, rather, to wait until the ball had settled after each throw before retrieving it. This constraint was developed to simplify the number of potential motions to be recognized.

2.2.1 Recognition System

An experiment was performed to test the performance of an HMM recognition system. The HMM algorithms are implemented with the Hidden Markov Model Toolkit (HTK) from the Cambridge University Engineering Department [17, 56]. Details regarding our use of the HTK can be found in Appendix B.

Use of an HMM system for motion recognition is preceded by a process of training models for each of the motions we desire to recognize. Which motions we desire to recognize is an important choice (see Chapters 1 and 3 for more discussion on this topic), and almost any group of motions could have been selected for this experiment. An analysis of the motions executed during the recorded sessions of an expert user and several test subjects resulted in the definition of ten basic gestures (chosen by the experimenter) that are used to classify all the observed motion. These ten gestures define the "vocabulary" of our recognition system and are described in Table 2.1. The training data was formed from 14 recordings of an experienced user executing these motions. Not every recording contained every motion; each motion had a minimum of seven examples in the training data. The data was used to train a plain, single mixture, single stream, five-state HMM for each of the basic motions.

To assess the performance of the recognition system, it is necessary to have a standard for comparison. As with speech recognition systems, our standard is a transcription detailing the motions used and times of transition between motions. This transcription was identified manually by using the capability of the virtual environment to replay recorded sessions. As the recording is replayed, the data is labelled and

Table 2.1: Motion Vocabulary for the Ball and Target Task

Label	**Description**
A	Moving downward to retrieve ball, ends after ball is grasped
B	Moving primarily upward with ball.
C	Throwing the ball. Ends at time the major components of motion in the direction of the throw cease.
D	Horizontal movement to the left without the ball.
E	Moving forward and down to retrieve ball. Ends after ball is grasped.
F	Moving left and up with ball in gripper.
G	Moving backward and down to retrieve ball. Ends after ball is grasped.
H	Moving forward and up with ball in gripper.
I	Wasted motion-low magnitude in any direction, does not result in major position change or end by retrieving or throwing the ball.
J	No motion; silence.

segmented manually by the experimenter. (In previous work [31, 34], we allowed the users to segment the task during execution by pressing a key on the control computer when they intended to change motions, but for the dynamic task presented here, this method generates significant errors in transcription due to increased mental load.) During each recording, data was collected at 100 Hz. In all, twelve observations were recorded: position (x_{pos}, y_{pos}, z_{pos}) and velocity (x_{vel}, y_{vel}, z_{vel}) of the tool tip in three dimensions, position of the ball (x_{ball}, y_{ball}), the distance separating the ball and the tool tip (ds), the status of the gripper (g)—either open or closed—and the magnitude of forces (f_x, f_y) being exerted on the tool tip by objects in the environment.

2.3 Validating Experiment

The algorithms in the HTK have several parameters which require tuning to obtain the best performance. Thus, our experimental process began with nearly 60 test runs used to adjust these parameters to baseline values that produced reasonable results. Among these parameters were the model transition penalty, the pruning threshold, and the number of states in each model. The transition penalty affects the Viterbi-like algorithm used for recognizing the most likely sequence of models, known as the Token Passing Model. The algorithm works by passing tokens through the network of possible models and discarding tokens that travel low probability paths. The transition penalty is a fixed value that is added to the log probability of each token as it jumps to a new model. The pruning threshold defines the width of search used during the forward-backward procedure for model estimation. Both the transition penalty and the pruning threshold have a strong effect on performance of the system. In general, a lower transition penalty results in a greater number of insertions. Insertions describe places where the system "recognizes" a motion that was not actually performed. Conversely, a higher transition penalty leads to more deletions, when the system does not recognize motions that were performed. The effect of the pruning threshold is less dramatic (the main benefit is decreased computation), but making it larger tends to increase the number of insertions and vice versa. Both parameters require fine-tuning for optimal performance and different data sets. The first batch of tests also verified that the quantity of training data was sufficient for robust model estimation by using only half of the data and obtaining

Table 2.2: Word accuracy percentages for recognition of training data

Test	**Observations**	**Accuracy %**
1	x_{pos},y_{pos}, z_{pos}, x_{vel}, y_{vel}, z_{vel}	73.83
2	x_{pos}, y_{pos},x_{vel}, y_{vel}	73.83
3	x_{vel}, y_{vel}, g	71.96
4	x_{vel}, y_{vel}	71.96
5	x_{vel}, y_{vel}, f_x, f_y	71.96
6	x_{pos}, y_{pos}, x_{ball}, y_{ball}, ds, f_x, f_y	73.83
7	x_{ball}, y_{ball}, ds, f_x, f_y	81.31
8	x_{ball}, y_{ball}, ds	81.31
9	x_{ball}, y_{ball}	81.31

comparable results to tests using twice as much data. With the values of these parameters settled, we set out to determine which observations were most important to achieving accurate recognition.

2.3.1 Results

Table 2.2 shows the results of nine different tests including various combinations of observations in the training data. The recognition was performed on the training data. For all tests, the transition penalty and pruning threshold parameters of the HTK system were set at -200 and 1000, respectively. Accuracy is computed using a common formula from the speech recognition literature (also the one automatically computed by the HTK): $(N - D - S - I)/N$, where N is the number of motions in the transcription, D is the number of deletions, S is the number of substitutions, and I is the number of insertions. This is an appropriate metric because it captures the number of each type of error during recognition.

The results show there is considerable room for improvement before we achieve the

success of other systems based on the same techniques. (Successful speech recognition systems typically have recognition accuracies greater than 95%.) However, they also highlight the flexibility of this method. Even when using nine different combinations of input observation vectors—some with more than three times as many components as others—the recognition rate remains relatively flat. The small variance in recognition rate prevents any sweeping conclusions, but some variables do appear to have advantages over others. As shown in Table 2.2, Test 1 represented a typical sampling of observations that would naturally be selected for a motion task. This combination also included the z-axis position and velocity. Despite the fact that the virtual environment is only two-dimensional, the haptic device is not constrained to this plane, and the possibility existed that movement along that axis could be of use in recognition. When compared to Test 2, though, we see that the recognition is unaffected by the loss of the z-axis information, and we declined using it further. The observations in Tests 3 and 4 were selected because these were most closely related to how the motions were defined (Table 2.1). The results suggest that despite this primary role, recognition can be improved with the inclusion of more information. Test 4 indicates that knowing the gripper status contributes negligibly. Test 5 was the first to include the forces displayed to the user and shows that, although these forces improve the reality of the environment and may be beneficial to the user for completion of the task, they do not appear to have a useful effect on recognition. The results of Tests 7 and 8 support this hypothesis. Test 6 used only observations that are not measured outside of the virtual environment. A small improvement in recognition encouraged Tests 7–9, and these observations, particularly the position of the ball, produce the

highest recognition rates. However, these results do not tell the complete story. First, the state of objects in a real environment may not be available for use in evaluation. Also, further analysis reveals that only one of the 13 examples of motion J (silence) in the data was correctly recognized in these tests. For that reason, this group of observations may not be the best choice in the context of our overall goal of skill evaluation.

This preliminary work also explored the use of a list of motions for skill evaluation. These methods are discussed in Chapter 5.

2.3.2 Conclusions

In our preliminary work we demonstrated the viability of a motion-recognition system using HMMs. Trained motion models were used to automatically recognize the motions utilized during performance of a dynamic, unstructured task. We learned several important things through this work. First, several parameters of the HMM system were found to have important effects on the recognition percentage. Second, the recognition performance is also affected by the observations used for training and recognition with the motion models. Perhaps most importantly, however, we gained insight regarding the tremendous importance of the motion vocabulary and its role in both the ability to recognize the motions which comprise it, and in the type of assessment that may be made using this list of motions.

Chapter 3

Motion Segmentation

3.1 Introduction

In Section 1.4 it was noted that the use of a continuous HMM recognition system did not produce acceptable results in our application. Under the continuous approach, the algorithms incorrectly identified both the times of transition as well as the total number of transitions. Subsequently, the fraction of motions correctly recognized was quite low. An alternate method for automatically segmenting the motion data enables use of the HMM framework for a more simple, isolated recognition process described fully in the next chapter. Inspired by the use of an algorithm for "episode extraction" in [43], we consulted the original work [20], in which the authors present an algorithm for motion segmentation based on the sum of squares of the angular velocity at each joint in a serial chain manipulator (a human arm). This algorithm has proven to be an effective tool for motion segmentation in our application.

Implied in any discussion of an automatic segmentation technique is the existence

of a standard by which to judge the performance of the technique. In our case the standard is obtained through a manual process to identify times of transition between separate motions.

The segmentation process is intricately tied to the definition of the motion vocabulary through the criteria used to identify the beginning and end of each motion. This chapter presents the results of working with data from two different tasks. For each of these, as for any task, it is possible to define any number of segmentations and attempt to manipulate the parameters of the sum of squares (or another) algorithm to produce the desired result. This chapter describes the data collection methods, each of the tasks and the varying segmentations used thus far, and methods and techniques for both manual and automatic segmentation.

3.2 Data Collection Methods

Collecting motion data from the da Vinci® system is done through a software framework known as the Application Programming Interface (API). This arrangement guarantees the data collection process will not affect the operation of the robot by publishing the data under controlled circumstances. A program running on a computer inside the da Vinci® acts as a server, and sends data over a serial communications line to a client computer at a nonconfigurable rate of approximately 10 Hz. The source code for both server and client programs was provided by Intuitive as part of the API installation. The client application was then modified to facilitate recording of the data to logfiles used for all later analysis. Further information about

the API can be found in [32].

Table 3.1 summarizes the data available through the API framework. There are 192 separate values in total. As they appear in the table many of these values are self-explanatory, but others require additional interpretation. The rotation matrix for the master manipulators represents the rotation from the fixed base frame on the console to the coordinate system attached at the end of the manipulator. The Cartesian velocities of the master relate to this final frame. The six values represent the three linear velocities as well as the rotational velocity around each axis.

The data for the three patient side arms (the two manipulators and the camera) has some interesting features. Most notably, the Cartesian endpoint of the slave tools is not available. The Cartesian position that is provided is the location of the remote center of motion (RCM) relative to a frame attached to the tower at the patient's side. The shaft of the surgical tool will change orientation during operation, but will always pass through this point, which is aligned with the patient's abdominal wall during setup. The rotation matrix provides the orientation of the frame located at the tool tip. There are 12 values for the setup joints. These joints are used during initial setup of the robot, but do not move during surgery. There are six joints, with two values (one is redundant) for each joint. The position and velocity values for the final joints on the camera manipulator have no meaning, as the camera lacks a gripper.

Table 3.1: da Vinci® API Data Organization

Master Telemanipulators (MTMs)

Label	Data Points	Organization
Cartesian position	3	x, y, z
Rotation matrix	9	(1,1), (1,2), (1,3), (2,1), (2,2),. . .,(3,3)
Cartesian velocity	6	x_{vel}, y_{vel}, z_{vel}, x_{rot}, y_{rot}, z_{rot}
Joint position	8	joint 1, joint 2,. . ., joint 7, gripper
Joint velocity	8	joint 1, joint 2,. . ., joint 7, gripper

Patient Side Manipulators (PSMs)

Label	Data Points	Organization
Joint position	7	joint 1, joint 2,. . ., joint 6, gripper
Joint velocity	7	joint 1, joint 2,. . ., joint 6, gripper
Cartesian position of RCM	3	x, y, z
Rotation matrix	9	(1,1), (1,2), (1,3), (2,1), (2,2),. . .,(3,3)
Set-up joint values	12	Two values each for joint 1, joint 2, . . ., joint 6

Other

Label	Data Points	Organization
Servo Times	5	left master, right master, left slave, right slave, camera slave
Console Buttons	5	Head in, Master clutch, Camera control, Standby, Ready

3.3 Ring Transfer Task

The first data used for analysis came from a simple, pseudo-surgical task using the da Vinci® surgical system. The task was done on a training board designed for robotic minimally invasive surgery [27], and was designed to use very simple, deliberate motions that would be amenable to segmentation and recognition. The training board has synthetic rubber cones of varying sizes. The larger cones used for the task measure approximately 30 mm in height and 15 mm in diameter at the base.

An 8 mm diameter rubber ring was placed on a cone on the left side of the surgical field. The purpose of the task was to transfer this ring to another cone on the right half of the training model. All test subjects were instructed to retrieve the ring from the cone with the left hand, transfer it to the right hand, and place it on the new cone with the right hand. The task was always initiated and completed with the da Vinci® tools held motionless, grippers closed, in the middle of the viewable workspace.

Figure 3.1 shows the robotic tools and the test apparatus during performance of the task. After several practice runs to get acquainted with the system, the movements of each subject were recorded for up to 20 repetitions of the task. On average, each trial required approximately 15 seconds for completion.

3.3.1 Six Motion Segmentation

As stated, any arbitrary motion vocabulary and segmentation could be defined for this task. The first segmentation divided the task into six intuitively-selected motions:

1. Move the left tool toward the left cone where the ring is located
2. Retrieve the ring from the left cone
3. Move the left tool back to the center to transfer the ring to the right tool
4. With the right tool, move with the ring toward the right cone
5. Place the ring on the right cone
6. Move the right tool back to center to end the trial

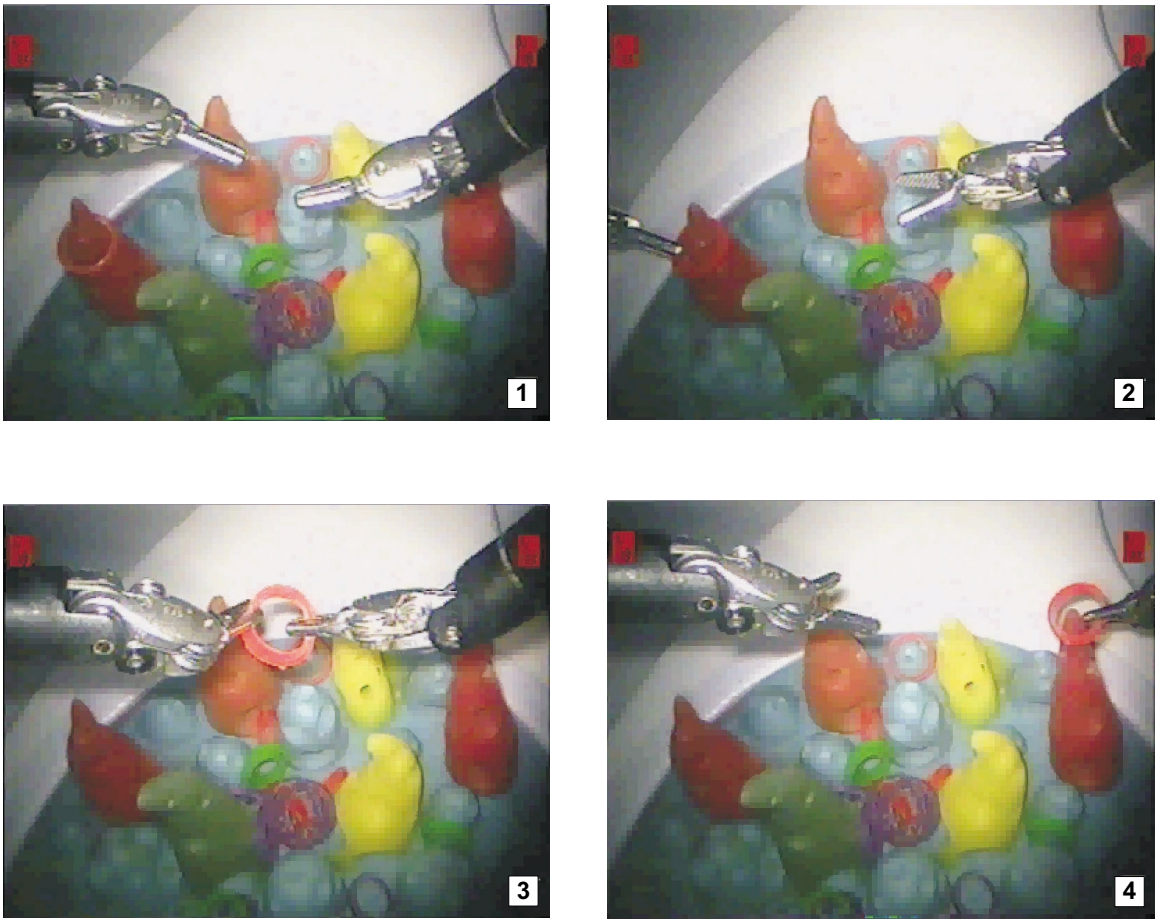

Figure 3.1: The ring transfer task. 1) The starting and ending position. 2) Retrieving the ring from the left cone. 3) Transferring the ring to the right tool. 4) Placing the ring on the right cone.

Criteria were specified to define the start and end of these motions. Motion 1 begins when the grippers open and movement towards the left cone is initiated. Motion 1 ends when the tool first makes contact with the cone. Motion 2 ends when the ring, held in the gripper, clears the cone. Motion 3 was defined to end at the middle of the time when the ring was held by both the left and right gripper during handoff. Motion 4 ends when the right cone first penetrates the center of the ring. Motion 5 ends when the ring is fully placed. Motion 6 ends when the tool is back in

the center of the workspace with the gripper closed.

3.3.2 Alternate Segmentations

Two alternate segmentation schemes were tested with the ring transfer data, one with seven motions and another with four motions. Both schemes are essentially modifications of the original six motion arrangement.

Seven Motion Segmentation

The seventh motion in the seven motion segmentation comes from redefining the third and fourth motions of the six motion segmentation. Whereas previously the handoff was treated as an event which occurred at the transition between two motions, in this scheme it is treated as a motion itself. Also, in this approach, less significance was given to external events and more significance was given to the appearance of motion at the tool tip. The seven motions are:

1. Move the left tool toward the left cone
2. Retrieve the ring from the left cone
3. Move the left tool back to the center
4. Hand the ring from the left tool to the right tool
5. Move the right tool toward the right cone with the ring
6. Place the ring on the right cone
7. Move the right tool back to center

The criteria are modified accordingly. Motions 1 and 2 remain unchanged. Motion 3 concludes as general motion towards the handoff location ceases. Motion 4 is the handoff. Motion 5 begins when the right hand initiates movement away from the handoff location and ends when motion ended near the right cone. Motion 6 ends when the ring was fully placed. Motion 7 ends when the tool was back in the center of the workspace with the gripper closed.

Four Motion Segmentation

The four motion segmentation arose from a combination of the motions in the seven motion segmentation. This shift was inspired by the basic surgical motions identified by Cao and MacKenzie in [6, 37]. One of these motions was labelled "reach & orient." Applying this methodology to our own task, Motion 1 can be characterized as a reach, while Motion 2 can be roughly characterized as an orient, so the two are combined into a single "reach & orient" motion. A similar rule was applied for the remainder of the task to arrive at the following four motions:

1. Move the left tool toward the left cone and retrieve the ring from the cone
2. Move the left tool back to the center to transfer the ring to the right tool and make the handoff
3. With the right tool, move to the right cone and place the ring
4. Move the right tool back to center to end the trial

The transition criteria are modified once again. Motion 1 concludes as the general motion towards the handoff location ceases. Motion 4 is the handoff. Motion 5

begins when the right hand initiates movement away from the handoff location and ends when the right cone first penetrates the center of the ring. Motion 6 ends when the ring is fully placed. Motion 7 ends when the tool is back in the center of the workspace with the gripper closed.

3.4 Manual Segmentation

In addition to recording data through the API framework during performance of the ring transfer and suture tasks, the movements of the da Vinci® tools were simultaneously recorded on videotape using the feed from one of the two cameras in the da Vinci® laparoscope. Each recording was then carefully examined to determine the start and stop time of each of the six motions used in completion of the task.

The manual segmentation procedure was initially done using the slow-motion replay feature on a VCR. The complete details regarding this process are outlined in Appendix A.1, but can be summarized as follows. During playback, the timing of transition events were noted using a stopwatch. To mitigate the effect of errors in this process, each recording was viewed three times and the transition times were averaged.

We believe the intervals of the transition times to be accurate to within 0.1 s of the events as viewed on video. However, as the data and video recording systems are completely independent, it is necessary to correlate these times with the data. Again, the details of all the steps taken to correlate these times are included in Appendix A.1 but will be summarized here. Most importantly, each task contained intentional,

well-defined events at the beginning and end. These events are clearly observable in both the API data and the video. There was a discrepancy in the elapsed time for each recording as derived from the data and from the video review, so a constant scaling factor was used to adjust the transition times for each file to match the total number of data points. Because the scaling was administered to the entire recording, the total change for each transition time was less than one data point on average. It is likely that this method produced segment times that group only a handful of data points in the wrong motion.

In later work an alternate manual segmentation method was devised. In the revised procedure, video from the task is digitized using a video capture card in a PC. With the video in this format it is then possible to step through each recording frame by frame and record the transition times. In addition to being much more efficient, this procedure eliminates the error derived from starting and stopping the stopwatch while viewing the recording. It does not, however, eliminate any errors induced by interpretation of the motions viewed.

3.5 Automatic Segmentation

It is possible for an algorithm to perform motion segmentation just as well or perhaps better than a human segmenter. A human may assign significance to events that are not observable in the data without force measurement or environmental knowledge. For example, in the six-motion ring transfer segmentation, the definition of motion 1 ends when the tool touches the cone, but due to the pliable nature of the

cones, no changes in position or velocity may be evident at this event. However, we do want our system to closely replicate the analysis performed by human surgeons, so it desirable to define the motions in an intuitive way and work to design an algorithm that will function in a similar fashion.

The sum of squares algorithm was proposed in [20], in which the authors used it to segment motions of a human arm. The algorithm is:

$$z = \dot{\theta_1}^2 + \dot{\theta_2}^2 + \dot{\theta_3}^2 + \ldots + \dot{\theta_n}^2$$

In general, during motions, $z \gg 0$, but during transitions the sum drops to $z \approx 0$. By empirically identifying an appropriate threshold, the sum of squares may therefore be used for segmenting motion data. As implied by the notation, in the original paper the variables in the summation were joint velocities. Following suit, we applied the sum of squares algorithm to the data collected from the da Vinci®, where θ_i represented the joints on the left and right master manipulators. Plotting the sum of squares for a typical task produced graphs like Figure 3.2. The continuous line shows the sum of squares of the joint velocities, while the vertical bars show the location of manually-identified motion transitions. In general, these results gave little hope that the sum of squares would be a reliable method for identifying motion transitions in our application.

We then used the same algorithm with the Cartesian velocities of the master manipulators instead of the joint velocities. That is,

$$z = \dot{x_l}^2 + \dot{y_l}^2 + \dot{z_l}^2 + \dot{x_r}^2 + \dot{y_r}^2 + \dot{z_r}^2$$

This approach produced graphs like that shown in Figure 3.3 (for the same trial as

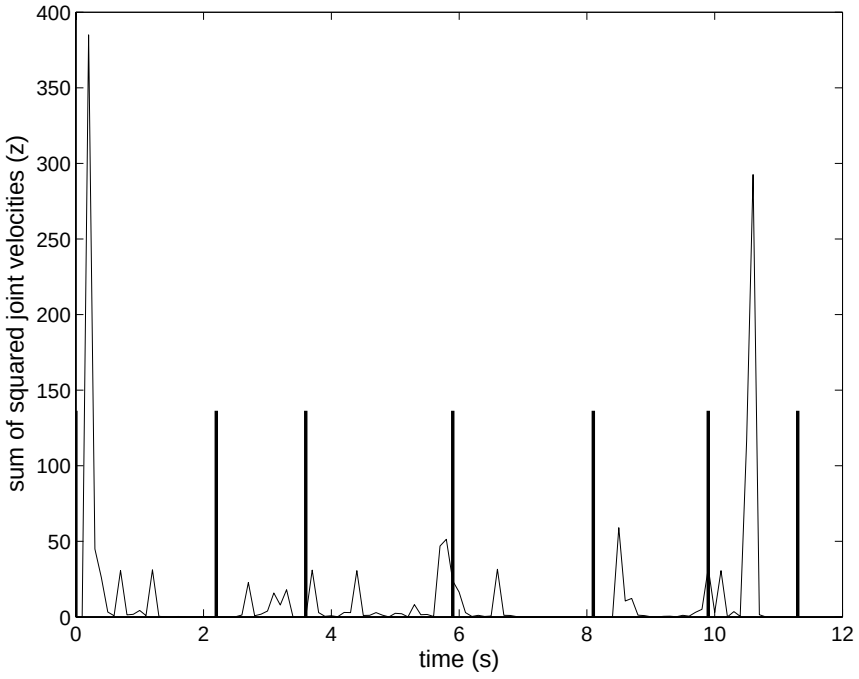

Figure 3.2: Sum of squared joint velocities. The continuous line represents the sum of squares, the vertical bars represent manually-identified motion transitions.

above), where the continuous line once again indicates the sum of squares and the solid vertical bars indicate motion transitions. Although the "peaks" and "valleys" do not line up exactly with the manual segmentation, the algorithm clearly identifies distinct segments in the data which agree with the manual segmentation.

Why does this algorithm work poorly with joint variables but perform admirably with Cartesian variables? We surmise that, during operation of the da Vinci®, surgeons do not explicitly control the joint positions of either the master or slave devices but rather, they control the Cartesian position of these. It stands to reason, then, that the joint positions and velocities would not necessarily be reliable indicators of the surgeon's input, while the Cartesian positions and velocities contain a valuable record of the motions used during surgery. Bobrow, et al [3] provide some interest-

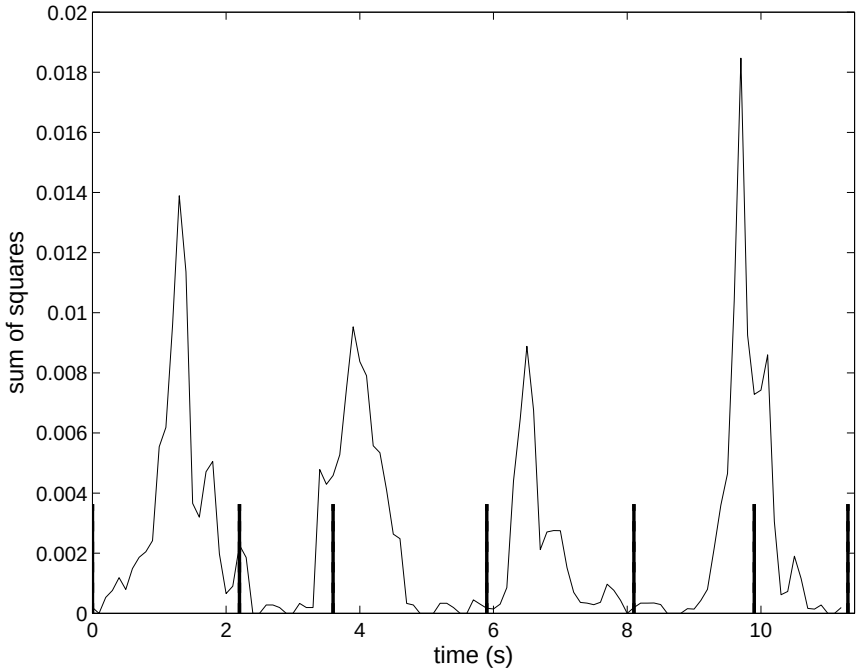

Figure 3.3: Sum of squared Cartesian velocities. The continuous line represents the sum of squares, the vertical bars represent manually-identified motion transitions.

ing insight on this topic associated with their goal of generating human-like robot motions. It is worth noting that the ICSAD system also uses a Cartesian velocity magnitude for segmenting surgical motions [13].

3.5.1 Implementation

Starting with the promising, preliminary results shown in Figure 3.3, the task was to produce an algorithm that robustly and accurately identifies transitions across multiple recordings. In general, the approach has been to iteratively search for peaks in the sum of squares. If the peak is above the higher of two threshold values, it is assumed a motion is in progress. The algorithm then searches for the nearest points on either side of the peak which are below the lower threshold; these points indicate

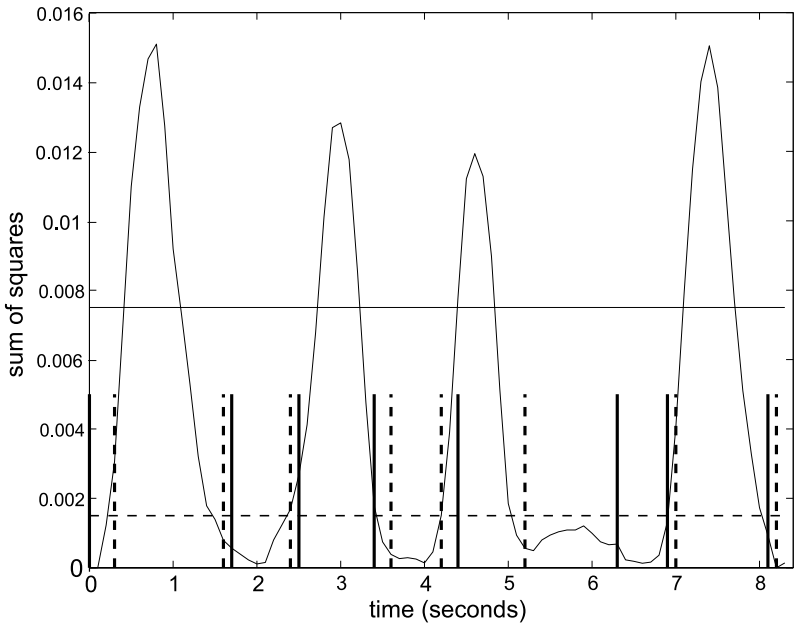

Figure 3.4: Segmentation with the sum of squares. The continuous line represents the sum of squared Cartesian velocities. Vertical bars show times of motion transitions: solid lines are manually-identified, dashed lines are automatically identified with an algorithm. The horizontal lines show the thresholds used in the segmentation algorithm.

motion transitions. Variations on this general strategy include smoothing the data, limiting how close two transitions may be, and using velocities from the left and right hands individually. These varying implementations can be best understood by examining the code directly; short descriptions of each are included in Section A.2. An example of an automatic segmentation is shown in Figure 3.4.

The multitude of variations on the sum of squares algorithm made qualitative assessment of each approach a difficult task. An objective scoring system was developed to help compare the performance of the segmentation algorithm for different parameter values and approaches. The scoring system works by first gathering the automatic segment times for each trial. These automatic segment times are then

compared to the manual segment times. If an automatically-identified transition is sufficiently close (± seven data points) to a manual transition, the program says the automatic transition "corresponds" to the manual transition.

To generate a score for an automatic segmentation, the transition time between corresponding automatic and manual transitions are subtracted. The score is the sum of the absolute values of all these differences. For example, if an automatic transition was placed two data points before or after the manual transition, two points are added to the score. A high score indicates that the automatic segmentation for the trial does not agree well with the manual segmentation.

The automatic segmentation may also miss some transitions or insert others where no transition was manually identified. These transitions typically do not correspond to any of the manual transitions. The scoring system accounts for these types of errors by adding .01 to the score. Thus, a final score of 7.02 indicates a total of seven points of difference between corresponding manual and automatic transitions and two transitions that were either missed or inserted by the automatic segmentation. The score for the automatic segmentation shown in Figure 3.4 is 11.01. This method has proven to be an effective way to compare different versions of the automatic segmentation algorithm.

3.6 Suture Task

Promising results in both segmentation and recognition for the ring transfer task led to a second, more realistic surgical task. This task was performed on a different

portion of the same training board used for the ring transfer task [27]. This portion of the training board contains a piece of simulated tissue with a large incision roughly 75 mm in length. The purpose of the task was to pass a needle through the tissue on one side of the incision. The needle was placed sticking out of the tissue prior to the task, where it is retrieved, positioned, and pushed partway through the tissue with the right tool. The left tool is then used to pull the needle through the tissue and place it near its starting point to complete the task. As before, the task was always initiated and completed with the da Vinci® tools held motionless, grippers closed, in the middle of the viewable workspace. Each trial of the task required approximately 15 seconds to complete. Figure 3.5 shows the robotic tools and the test apparatus during performance of the suturing task.

3.6.1 Seven Motion Segmentation

This data was manually segmented into seven distinct motions:

1. Move the right tool to the needle
2. Pluck the needle from its place in the tissue
3. Position needle for insertion
4. Push the needle into the tissue
5. Using the left tool, pull the needle and attached suture material through the tissue
6. Place the needle on the tissue near where it started

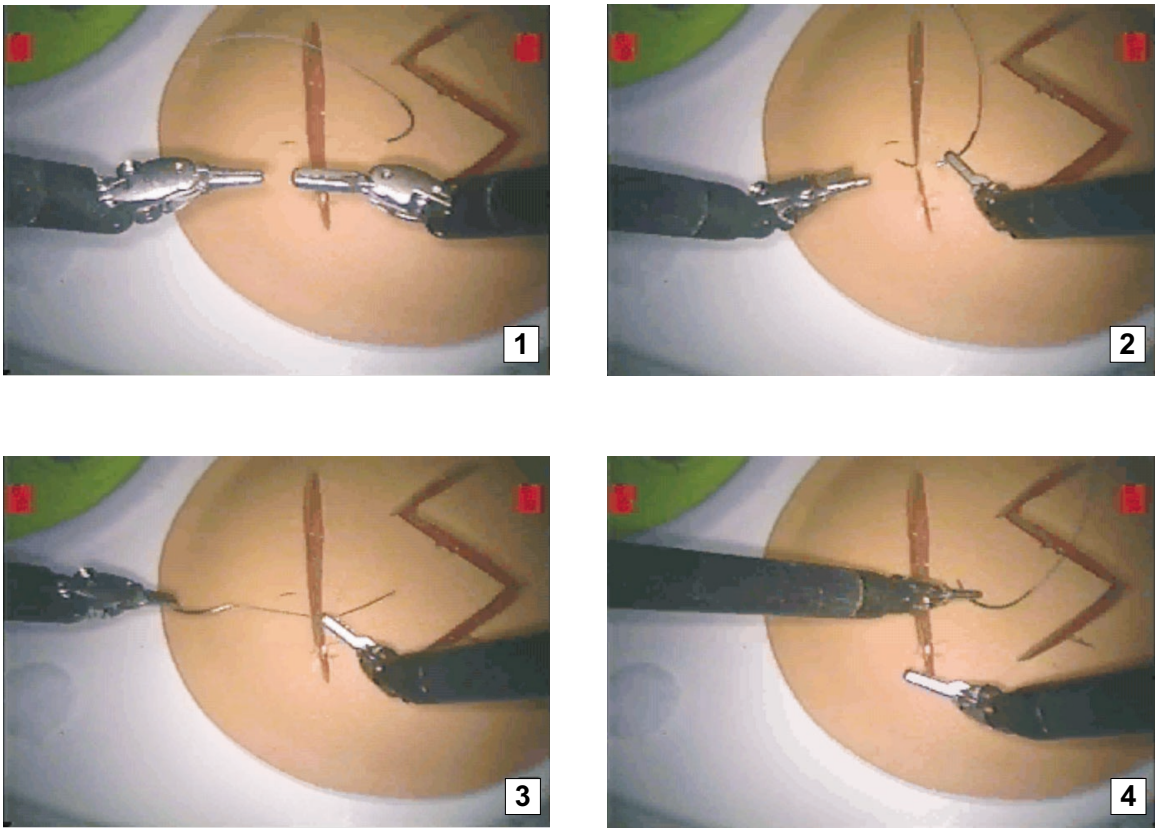

Figure 3.5: The suture task. 1) The starting and ending position. 2) Inserting the needle. 3) Pulling the suture through the tissue. 4) Placing the needle.

7. Return the left tool back to center to end the trial

The criteria defining the start and end of these motions are as follows. Emphasis was given to the appearance of motion at the tool tip in defining the start and end of each motion. Motion 1 begins when the grippers open and the right tool initiates movement towards the needle. Motion 1 ends as the tool stops motion at the needle site. Motion 2 begins when movement begins with the needle in the right gripper. This motion was typically away from the insertion site. Thus, Motion 3 begins with movement towards the insertion site with the needle. Motion 4 begins when the needle punctures the simulated tissue. Motion 5 begins when the left tool initiates movement

away from the insertion site with the needle in the gripper. Motion 6 begins with movement of the left tool back towards the insertion and needle placement site and ends once it arrives at that location. Motion 7 ends when the tool is back in the center of the workspace with the gripper closed.

The suture task data was manually segmented using the digitized video process described earlier in this chapter and several variations on the sum of squares algorithm were used for automatic segmentation. The best-performing of these was selected and used to segment the data for the results reported in Chapter 4.

3.7 Conclusions

This chapter described the methods for collecting motion data via the da Vinci® API and two tasks for which data was recorded. Each of the tasks was the subject of both manual and automatic analyses to determine the times of transition between motions. The sum of squares algorithm has proven itself as an effective tool for automatic segmentation in both tasks. There are many possible variations on this algorithm, a unique scoring system was devised for evaluating the performance of each variation. The segmentation process implicitly defines the motion vocabulary for the system. Each of the varying schemes has at least some of the desirable characteristics we identified for a motion vocabulary in Section 1.2. The next step in the process is to assess our ability to recognize these motions, and this is the topic of Chapter 4.

Chapter 4

Automatic Motion Recognition

4.1 Introduction

The experiments described in Chapter 2 validated our approach to using hidden Markov models for automatic motion recognition. However, as discussed in Sections 1.4 and 3.1, when the continuous HMM approach was applied to data taken from the da Vinci®, the recognition rate was very low. For this reason we adopted an isolated system architecture. Even with the isolated approach the recognition rate was still lower than desirable, so this chapter also discusses the effects of three additional techniques we have explored to increase the rate: interpolation, normalization, and linear discriminant analysis (LDA). Notably, we have found that the effect of each technique varies according to the order and combination in which it is used with other techniques. The results show that a combination of LDA and normalization, in that order, yield the best results.

4.2 Methods

The data collection procedures are discussed in Section 3.2. Significant work has been invested in developing a framework that enables implementing various techniques like those presented in this chapter and preparing the data files for use in the Hidden Markov Model Toolkit [17, 56]. Details regarding procedures for using this framework are included in Appendix B.

4.3 Isolated Motion Recognition

In Section 2.1 we identified three basic problems associated with using HMMs for practical tasks. Problem 2 in this list asked the question: given a sequence of observations, how do we identify the "optimal" state sequence which most likely produced these observations? We say "optimal," because for most HMM architectures there are many different state sequences which could have produced a given set of observations. Much like a multiplicity of state sequences which could have produced the same observations, there exists a chance that different model sequences could have produced the same observations.

Ideally, the model for each motion in the system vocabulary would produce observations distinctly different than those from any other model, as this would make distinguishing the likely motion from the observations much easier. Unfortunately this is not always the case, either as a result of insufficient model training or simply because the motions themselves are similar. This means that when one motion ends and another begins, there may not always be a significant change in the observations,

making it difficult to identify the transition. In a continuous system, segmenting and recognition occur simultaneously. We have adopted an isolated approach. The idea behind switching to isolated motion recognition is to simplify the task we are asking the recognition algorithms to perform. By first segmenting the motions with an alternate means, the recognition system is used only to tell us which of the motion models in the system vocabulary most likely produced the observations for this unknown motion.

4.4 Interpolation

As detailed in Section 3.2, data is collected from the da Vinci® system through the API software framework. The API publishes the data at a nonconfigurable, fixed rate of approximately 10 Hz. For the deliberate, controlled motions typically used by surgeons, the 10 Hz rate is most likely sufficient for capturing the essence of the signals. However, for the purposes of motion recognition with a statistical technique such as HMMs, we would ideally collect data at a much higher rate. To understand why this is, recall that HMM recognition relies on the calculation of the probability that a given set of observations were produced by one of the trained models in the system. A low data collection rate is therefore problematic at two different stages. First, with limited data it is difficult to robustly estimate the observation distributions and transition probabilities that define the models. Second, at a low data collection rate, each motion we wish to recognize may be represented by only a relatively small number of data points (15–25), and calculating the probability that these observations

came from any particular model can only be done with low confidence.

One possible solution to this problem is to interpolate the data we have. In order for this to be a valid approach, we make the assumption that the data we collected at 10 Hz is not aliased. Aliasing describes a condition that might occur if a dynamic signal is sampled at a rate too low to capture the frequency of the transients and the recorded data appears to have a lower frequency. In our system, we can rely on some knowledge of the task to validate this assumption. That is, we know that in general, surgical motions are performed in a relatively slow, deliberate manner and it is unlikely that any significant motions were not captured at the 10 Hz rate. Also, an examination of the 10 Hz data shows "smooth" position trajectories. This suggests that if aliasing did occur, it resulted because of oscillatory motions occurring at almost exactly integer multiples of 10 Hz, an unlikely scenario. Figure 4.1 shows the x position for the left master during a typical recording of the ring transfer task.

The interpolation is done by fitting a third-order polynomial to each pair of positions. The four coefficients in the third-order polynomial are solved for using the pair of positions and the velocities recorded at each of those positions. Originally we intended to use the velocity data recorded from the robot. However, this velocity is often noisy enough that the resulting fitted polynomial is not smooth. Instead, we perform a rudimentary numerical differentiation on the position data using a backward difference calculation. The resulting velocity trace has all the key features of the velocity data recorded from the robot, as seen by comparing Figures 4.2 and 4.3.

The resulting interpolated position data is shown in Figure 4.4. The dots on the line show the original data points. In this example, three additional points were added

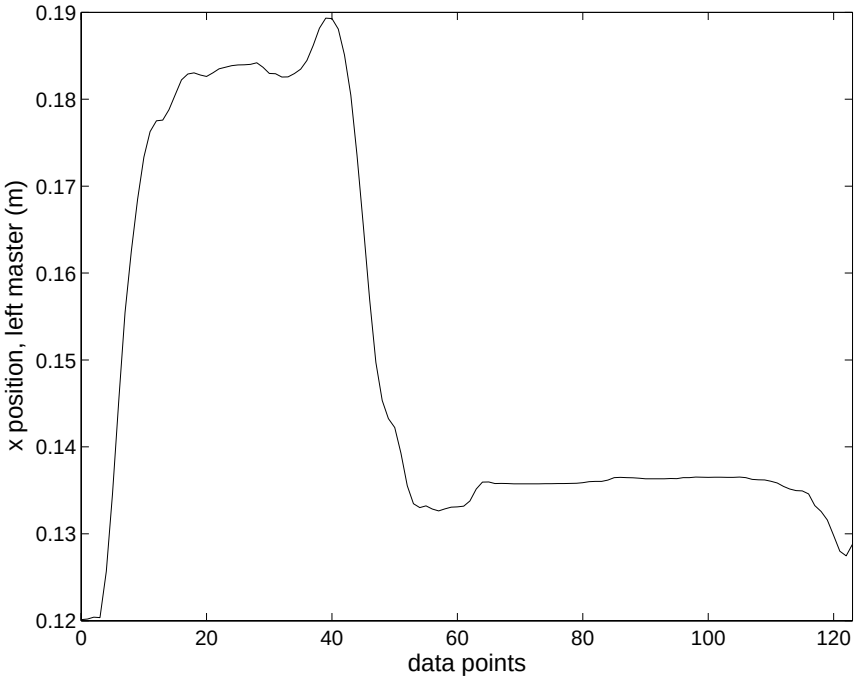

Figure 4.1: Cartesian x position of the da Vinci® left master manipulator during performance of the ring transfer task.

between each original pair. There is no theoretical limit to how many points could be added.

4.4.1 Effects of Interpolation

Although using interpolated data for model training and recognition appears to be a valid approach, the real value of this or any technique is evaluated solely by how it affects the ability of the system to automatically recognize motions. In order to determine this effect, an analysis of variance (ANOVA) study was conducted using data from the ring transfer task and the six motion segmentation described in Section 3.3.

The dependent variable in this study was the percentage of motions correctly

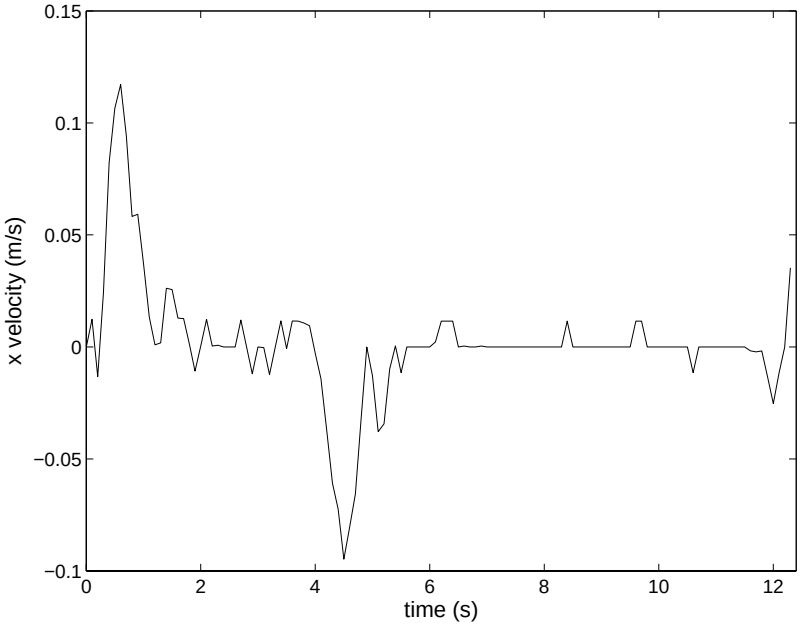

Figure 4.2: x velocity taken directly from the da Vinci®. This data was too noisy to use with the interpolation procedure.

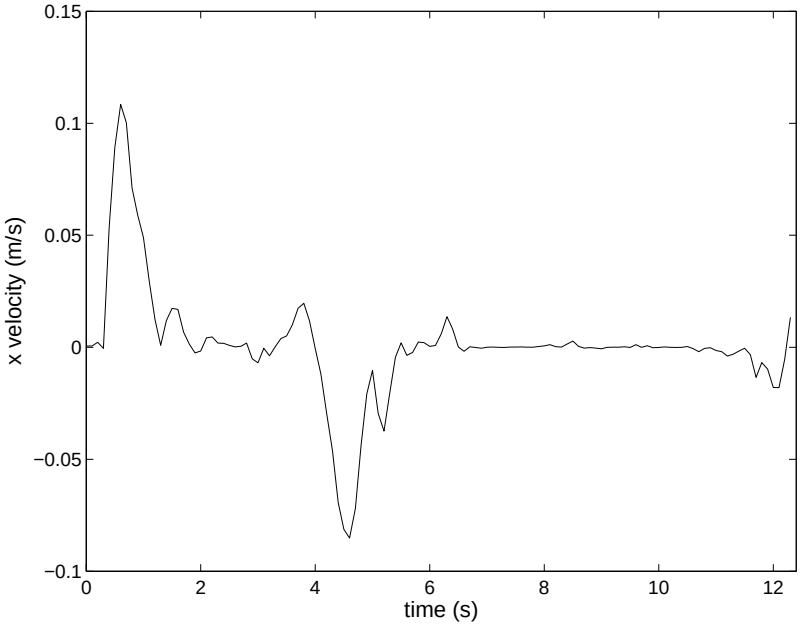

Figure 4.3: x velocity calculated with backward difference from da Vinci® position data. This data mimics the qualities of the raw da Vinci® data while being smooth enough to use for interpolation.

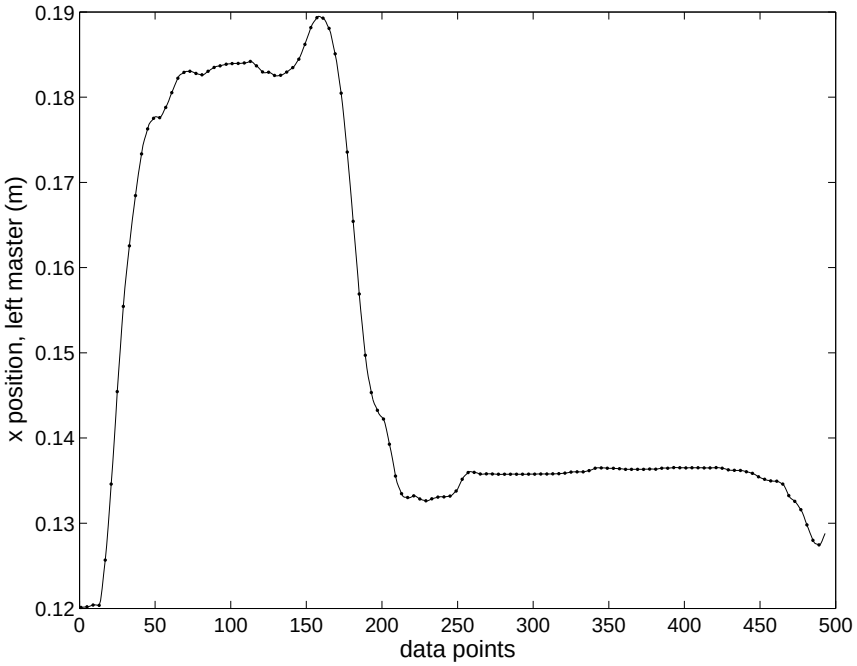

Figure 4.4: Interpolated Cartesian x position; there are three additional points between each original pair.

recognized for the isolated motion recognition procedure. There are many factors that could contribute to variation in the dependent variable, such as the surgeon, the data set used for training the HMMs, the experimental setup, the parameters in the recognition algorithms, and other methods of post-processing the data. We sought to hold all of these factors constant while recording the results of the automatic motion recognition process for 20 combinations of two other factors: 1) the level of interpolation and 2) the number of states in the hidden Markov model. We also chose to consider the surgeon as a blocking factor.

The interpolation levels represent the number of additional points generated between each pair of points in the original data. The levels were 0, 1, 3, and 5 points. We considered five possible values for number of states in the model: 1, 2, 3, 4, and

Table 4.1: Percentage of motions correctly recognized

Points of Interpolation	Number of States									
	1		2		3		4		5	
	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2
0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0
1	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	55.6
3	50.0	50.0	50.0	50.0	11.1	33.3	11.1	33.3	16.7	33.3
5	50.0	50.0	11.1	27.8	11.1	27.8	11.1	27.8	11.1	27.8

5. Data from two surgeons were recorded and used in the experiment. Ten recordings from each surgeon were used to train the models, and isolated recognition was attempted on a total of 36 motions taken from three additional recordings done by each surgeon (six recordings with six motions each).

The results are presented in Table 4.1. The recognition percentage for each of the two subjects (S1 and S2) is shown for each combination of factor levels. Using the isolated motion recognition procedure, the percentage reflects the fraction of the 36 motions that were recognized correctly.

The results of the ANOVA study showed that increasing the number of interpolated points has a negative effect on the recognition percentage. All pairwise-comparisons between the levels of interpolation showed no significant difference ($p = 0.95$) between no interpolation and adding one point, but all other increases in the level of interpolation significantly worsened the recognition rate. This shows that interpolation will not generally improve recognition rates, likely because it does not add truly new information.

Similarly, increasing the number of states also leads to a decrease in the recognition rate. Here the all pairwise-comparisons showed significant differences for all levels

except for levels 3, 4, and 5. This result is not completely unexpected, and the mechanisms are better understood. Adding states to the model creates a need to estimate additional parameters. With a limited amount of data for training, the estimates for all parameters become less robust, which could lead to a decrease in the recognition rate. Also, if the underlying system being modeled by an HMM does not exhibit a sufficient level of complexity, adding additional states creates a model that is not properly matched with the system. This condition may also lead to a low recognition rate. These results seem to indicate that the underlying process is simple enough that it is best modeled with a single-state HMM.

4.5 Linear Discriminant Analysis

As implied by the preliminary experiments discussed in Section 2.3.1, the choice of features included in the observation vector has a definite effect on the performance of an HMM-based recognition system. It is easy to imagine that two or more distinct motions could have some similarities in the values of particular variables. Other variables may differ a great deal, and the ones that differ significantly are more useful for discriminating between motions and ought to be included in the observation vector. Another less obvious issue is that too much information in the observation vector may also be problematic. If we create a more complex model via a larger observation vector, we require more data to estimate the model parameters. However, because we have a limited amount of training data, the result is often poor parameter estimation and reduced recognition capability.

In short, for HMM-based motion recognition to work reliably, each model must have distinct parameters from all other models, a condition which will arise only if each of the motions we wish to recognize are somehow distinct from all other motions in the data we use to train the models. This separation criteria is part of the motivating idea behind performing a linear discriminant analysis (LDA). LDA will transform our data into a reduced dimension space in such a way as to maximize the between-motion separation while minimizing the within-motion variation. The transformed data is then used to train models; test data must also be transformed to this new "feature space" before we can attempt recognition. The reduced dimension of the feature space is an ancillary benefit that will reduce the complexity of the models and the computing requirements for training and recognition. LDA has been applied in speech recognition systems for many years; [5, 18, 57, 58] are some of the earliest works reporting this application.

In order to demonstrate the effectiveness of LDA we considered several test cases. These test cases, much of the following discussion, and our own LDA implementation are drawn heavily from the work of Geirhofer [26]. A formal treatment can be found in [22]. Figure 4.5 shows data from two different "classes" (the different motions in our application). There is very little separation between these classes in either the x or y dimensions, as demonstrated by the general distribution curves for each class on the x-axis (y-axis distributions are not shown). If we attempted to classify a new point into one of the two groups using the x and y values independently, there are regions in which the probability of the new point belonging to either class are roughly equal, and the chance of making an error is therefore quite high.

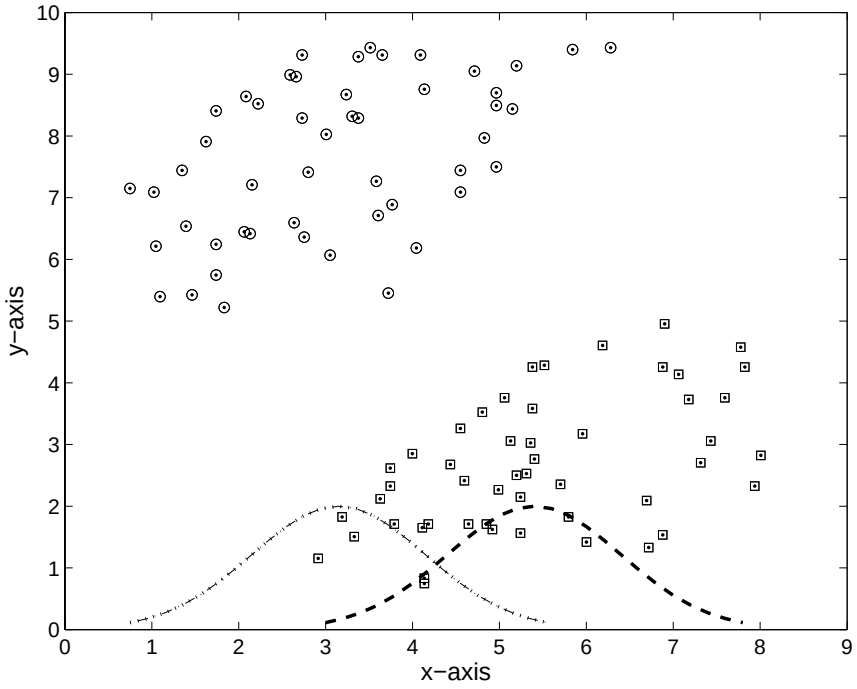

Figure 4.5: Example of data from two different classes with characteristic distributions shown on the x-axis. The data can not be accurately classified using either the x or y axes separately.

To improve this classification problem, we would like to transform the data so that we maximize the separation between these classes while minimizing the within-class variance. Any linear transformation can be represented as $y = \theta^T x$. Here the vector x represents a single sample with dimension m, so $x \in \Re^m$. We will use this transformation to reduce the size of the feature space from m to p, so $y \in \Re^p$ and therefore θ is an $m \times p$ matrix. In practice, we will have n samples of x, forming an $n \times m$ matrix X. Each sample is transformed to produce an $n \times p$ matrix Y.

As described by Geirhofer, the process to determine the appropriate θ begins by calculating the mean covariance matrix for each of the J classes $\overline{W}_j$ and the mean covariance matrix for the complete data set $\overline{T}$, each defined as

$$\overline{W}_j = \frac{1}{N_j}\sum_{i=1}^{N_j}(x_i - \overline{x}_j)(x_i - \overline{x}_j)^T$$

$$\overline{T} = \frac{1}{N}\sum_{i=1}^{N_j}(x_i - \overline{x})(x_i - \overline{x})^T,$$

where $\overline{x}_j$ is the mean vector for each class and $\overline{x}$ is the mean vector for the complete data set. N_j denotes the number of samples of data for class j, and N simply denotes the total number of samples for all classes.

The optimization criteria for $\hat{\theta}$ is formulated as

$$\hat{\theta} = \arg\max_{\theta_p} \frac{|\theta_p^T \overline{T} \theta_p|}{|\theta_p^T \overline{W} \theta_p|},$$

where

$$\overline{W} = \frac{1}{N}\sum_{j=1}^{J} N_j \overline{W}_j.$$

It can be shown that, in order to satisfy the criteria, $\hat{\theta}$ is formed from the eigenvectors corresponding to the p largest eigenvalues of the matrix $\overline{W}^{-1}\overline{T}$. The value of p is chosen based on the desired dimension of the reduced space and a somewhat subjective assessment of the magnitudes of the eigenvalues. For this example, we have no choice but to transform the data to a single dimension, effectively projecting the points onto a line. The original data and the line are shown in Figure 4.6. Histograms showing the distribution of the data from each class on this line are shown in Figure 4.7. From this plot it is clear that there is no overlap between the two classes in the new feature space, and a classification can now be done with much greater accuracy.

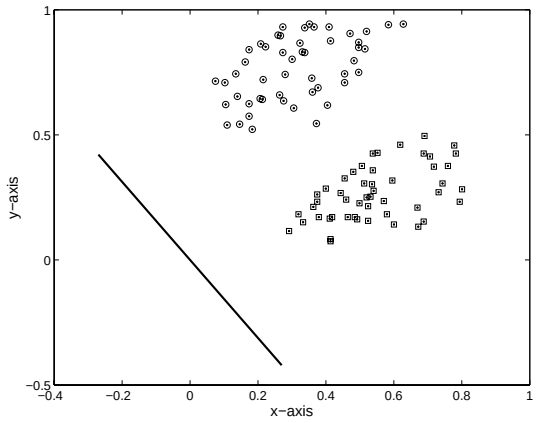

Figure 4.6: Data from two classes and a line representing the reduced-dimension space to which the data will be transformed.

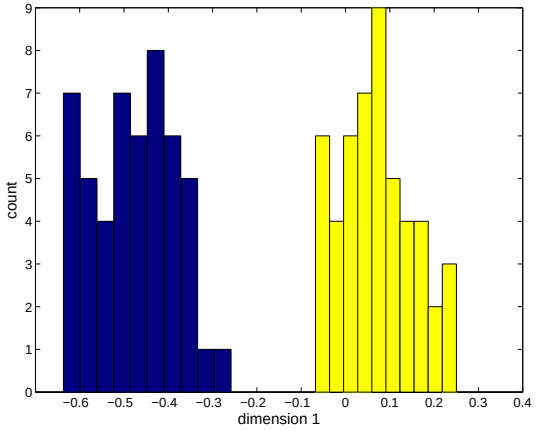

Figure 4.7: A histogram showing the distribution of data from the two classes in Figure 4.5 projected onto the line shown in Figure 4.6.

We have applied this same technique to the data collected from the da Vinci® during the two-handed ring transfer task. During that task we recorded 83 of the 192 possible variables detailed in Section 3.2. Of these 83, the 5 servo times were discarded immediately, leaving 78 variables related to the motion of the master and slave manipulators. These 78 variables were the Cartesian positions, Cartesian velocities, joint positions, and joint velocities for both the left and right master manipulators, and the joint positions and joint velocities for left and right slave arms. To apply LDA to this data we calculated the necessary elements to form the $\overline{W}^{-1}\overline{T}$ matrix and plotted the 78 eigenvalues, shown in Figure 4.8.

The plot indicates there are only a handful of eigenvectors which, when used in a transformation matrix, would have the largest effect. There is a sharp "elbow" in the magnitudes of the eigenvalues after the sixth one, and a more subtle change after ten; for this reason we selected $p = 10$ and formed our transformation matrix $\hat{\theta}$ from the 10 eigenvectors associated with the 10 largest eigenvalues.

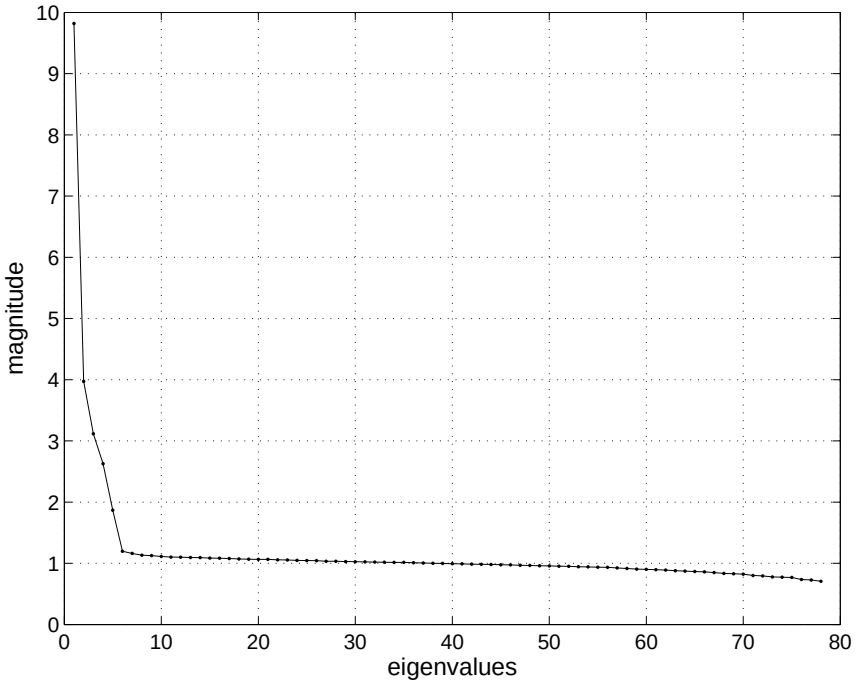

Figure 4.8: Eigenvalues of $\overline{W}^{-1}\overline{T}$ plotted in order of decreasing magnitude. The eigenvectors associated with the ten largest eigenvalues are used to form a transformation matrix.

It is impossible to view the data and visually assess separation between the classes in the reduced 10-dimensional space. However, as with other techniques, we are primarily concerned with the impact of LDA on our motion recognition capability. In order to assess this impact we tested the HMM system's recognition rate under seven different conditions, some with LDA and some without. One of the varying factors in these tests was whether or not the data was normalized at some point in the process. The normalization procedure is intended to compensate for the different units of measurement used for different variables. For example, the joint positions are measured in radians and may vary as much as 3 rad during the task. However, the Cartesian position of the master is measured in meters, and would rarely vary more

Table 4.2: Effect of LDA and normalization on recognition rate

Case	Data	Normalized	Recognition Rate
1	All data (78 columns)	No	51.28%
2	All data (78 columns)	Yes	44.23%
3	Master Cartesian velocities and gripper joint position	No	57.69%
4	Master Cartesian velocities and gripper joint position	Yes	49.36%
5	LDA reduced (10)	No	33.33%
6	LDA reduced (10)	Yes (pre-LDA)	78.85%
7	LDA reduced (10)	Yes (post-LDA)	85.26%

than 0.15 m during the task. During model training and recognition, variables with a smaller magnitude of variation may assume a less significant role, regardless of their true importance to the task. By normalizing each variable we hope to eliminate this effect. Normalization for each variable is done by subtracting each sample from that variable's global mean and dividing by that variable's global standard deviation.

In each of the test cases, we used data from 26 trials of the two-handed ring transfer task to train hidden Markov models for six different motions. Once the models had been trained, the recognition was tested using the isolated motion recognition process. As there were six motions in each recording and 26 recordings, the recognition percentage indicates the fraction of the 156 total motions the system correctly identified. The results are presented in Table 4.2.

The results for Case 7, with an 85% recognition rate, are by far the best results achieved to date. It is interesting to note that, individually, both normalization and LDA had a negative impact on the recognition rate, but when done together they had a positive effect. This indicates an interaction between the two factors, not unlike the

interaction observed between interpolation and the number of states in the ANOVA testing discussed earlier. Although in this case the positive result is a welcome one, the presence of these interactions makes the design of a generally applicable recognition method difficult to achieve. It suggests that for each new technique we try, we can not assess the significance of that technique alone. Rather, it must be assessed in combination with all other techniques in every possible combination. This creates an enormous possible solution space, and it is not clear if there is a more intelligent or efficient method to use beyond a brute-force searching of this space for the best combination of treatments.

To gain a further appreciation for the interaction between different techniques, recall that the ANOVA analysis in Section 4.4 showed that interpolating the data had a negative effect on the recognition rate. Our experiments with LDA came later, and so interpolation had not been considered with this factor. Recent experiments, however, showed some cases where a 1-point interpolation produced marked improvement when used in conjunction with LDA and normalization. This interaction will be explored further.

Chapter 5

Motion-based Skill Evaluation

Chapters 2–4 discussed the concept and implementation of a system to automatically recognize motions for the purposes of surgical skill evaluation. One question remains: that is, once the motions have been recognized, how can they be used to evaluate skill? Answering that question is the focus of this chapter. In short, an examination of the number of motions, the distribution of motions, and the sequence of motions all have potentially valuable roles for this purpose.

5.1 Preliminary Results

Concurrent with our preliminary work to validate the HMM-based motion recognition approach, we also began to explore appropriate ways for evaluating skill from motions. Three different subjects, all with no prior experience using the virtual environment described in Chapter 2 participated in an experiment. Each subject completed the dynamic task described in that chapter three times and their performances

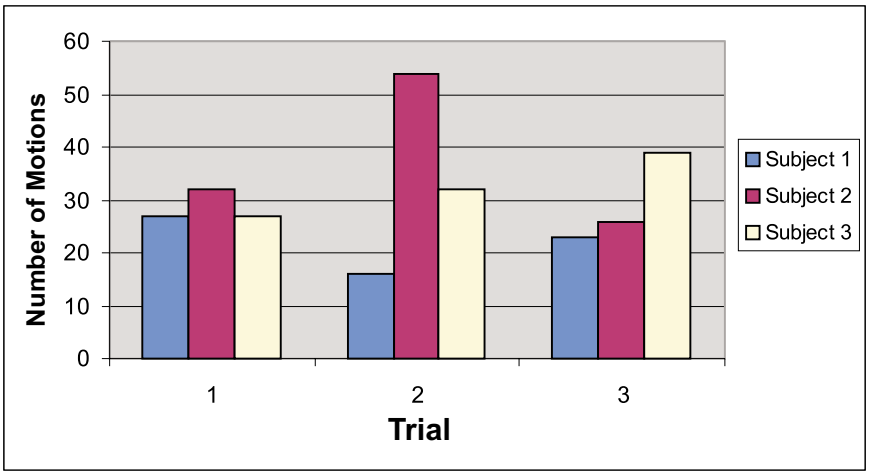

Figure 5.1: Total number of motions used by each subject in each repetition of the ball task. Subject #1 used the fewest motions in each attempt and overall.

were recorded. Our intent was to automatically recognize the motions executed and use this list to draw conclusions about the user's skill. In this case we used a simple method comparing the total repetitions between users over multiple sessions to obtain a relative measure of skill.

Recognizing that our recognition system had not yet been refined to the point that it would provide accurate, reliable results, the results of this experiment refer to the manually identified sequence of motions utilized by each subject, rather than results from the HMM recognition. Figure 5.1 shows the total number of motions used by each subject in each of three attempts to complete the task described. The plot shows that test subject #1 used fewer motions than the other two subjects in all three trials.

As shown in Figure 5.2, we are also able to compare the time of usage of motions I (wasted motion) and J (pause). Wasted motion accounts for 34% ±3% of the total time for all three subjects. More revealing is the usage of pauses, found to be 7.4%

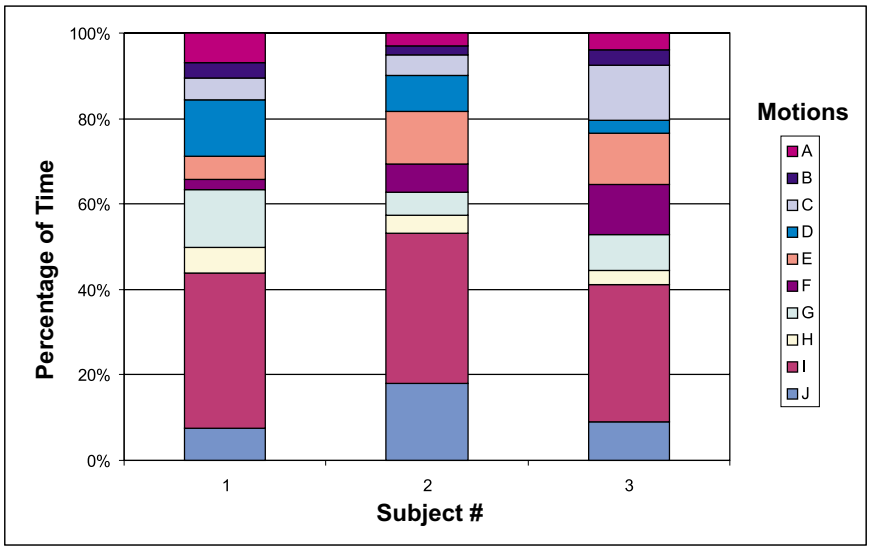

Figure 5.2: Average time usage distribution for all motions. All three subjects spent approximately 1/3 of the time using motion I ("wasted motion")

for subject #1, 18.0% for subject #2, and 9.1% for subject #3.

From these results we conclude that subject #1 was the most skilled of the group. We make this judgement using the assumption that a skillful user will require the use of fewer motions to complete the task than a novice user and that this reduction will come, in part, from more efficient execution. Such "economy of motion" is often subjectively gauged for surgical skill evaluation.

5.2 Methods of Assessment

A motion recognition system enables numerous methods for assessing technical skill. In general, there are three ways a list of motions may be used to evaluate skill. These are examinations of

- the number of motions used

- the distribution of motions used, and
- the sequence of the motions.

In general, these criteria do not have universal standards associated with them, but rather, are most useful when used to compare individual surgeons, groups of surgeons, or the progression of an individual's performance over time. For example, how many motions should be used to complete a bowel anastomosis on a training model? Clearly this is heavily dependent on how we define each motion, and perhaps the best standard we have is the average number of motions used by a group of highly experienced surgeons to complete the task.

We would expect the number of motions used by a surgeon to correlate heavily with the time used to complete the task. Time has been used as a metric for skill in many previous studies, although there is almost universal agreement that time alone does not tell the complete story. Knowing the number of motions used during this time provides useful additional information that enables a reviewer to assess a surgeon's style. In general, it is agreed that fewer motions are better. Counting the number of motions is accomplished by the segmentation algorithm in our system, and closely follows the implementation of the ICSAD system discussed in Section 1.3. The amount of time spent using each motion, as in our preliminary analysis, may also give useful comparisons.

The distribution of motions refers to the relative use of each motion in the system vocabulary. We expect to see two characteristics emerge using this analysis. First, novices will likely use some motions more frequently than experts as a result

of repetition. For example, if the task required positioning of a needle in a specific orientation, novices may need to pick up and set down the needle multiple times in order to achieve this goal, whereas an expert may be able to accomplish this on the first attempt. Second, there may be some motions that are more "advanced," perhaps because they are a hybrid of multiple, more simple motions, which would be used more frequently by expert surgeons because novices have not acquired the necessary dexterity.

The discussion regarding the distribution of motions highlights the important fact that these three criteria (number, distribution, and sequence) are not at all independent from one another. The repetitions which would affect the distribution of motions would also necessarily show up in the number of motions. That scenario would also be precisely the type of characteristics we would expect to find through analysis of the sequence of motions used. In this context, the term sequence refers to patterns of motions. To date, we have not performed any analysis of this type, but it holds some promise for identifying higher-level differences between experts and novices in their completion of a task.

A comparative analysis using any of these criteria, like the one used in our preliminary work, would be particularly useful if a recognized expert was included in the test group. Another possibility would be to track these measures over time or over many repetitions of the same task by a single user in order to evaluate the user's learning curve. Over a large group of subjects trained with different methods, such analysis could yield valuable insight regarding the efficacy of different teaching techniques.

5.3 The Role of Recognition Rate

One obvious concern about this method of evaluation is that, if the recognition system incorrectly identifies some of a user's motions, any skill assessment based on this recognition may be flawed. One way to justify the situation would be to view errors in the recognition system like noise in a more conventional measurement. With acceptably high recognition rates (perhaps greater than 95%), the recognition system gives us a flawed picture of reality, but one that represents reality closely enough that it may still be used effectively to evaluate skill.

A more compelling validation could be achieved through a comparison of the results from manual and automatic motion identification. If the conclusions reached from a skill analysis using manually identified motions are consistent with the conclusions reached using automatically identified motions, then the system based on automatic motion recognition has proven its worth. In actual use, knowing that recognition errors almost certainly exist, it would be incorrect to draw conclusions from small differences in the motion usage by a surgeon.

Regardless of whether we view errors as noise or choose to ignore them based on prior validation, correlation of automatic skill evaluation with external metrics, such as functional outcomes, would provide the best standard.

5.4 Application to a Surgical Task

5.4.1 Task Description

An experiment was conducted to demonstrate the use of our methods for objective surgical skill evaluation. Three surgeons completed a simple suturing task using the da Vinci® surgical system. The goal of this task was to form a continuous suture line by passing the needle through a series of dots marked on a sheet of GORE-TEX® acting as simulated skin. Prior to the start of the procedure the needle was placed sticking out of the sheet near the first entry point. After the surgeon had completed four throws, the needle was laid on the sheet near the last exit point and the grippers were returned to the starting position near the middle of the workspace. Other than the needle entry and exit points and the procedure for the start and end of the task, no constraints were imposed on the surgeon. Each surgeon completed twelve repetitions of the task using the same set of eight holes in the sheet. The experimental setup and performance of the task are shown in Figure 5.3.

The three surgeons participating in the experiment had varying levels of experience and training in both traditional and robotic surgery. The first subject was a senior cardiac surgeon who had participated in training offered by the Intuitive Surgical, Inc., the makers of the da Vinci® system. Following training, this surgeon had used the da Vinci® in approximately 30 procedures over the year and a half prior to the experiment. The second subject was a resident in cardiac surgery with less than one hour of total exposure to the da Vinci® system prior to the experiment; none of the second subject's exposure was formal training for robotic surgery. The third subject

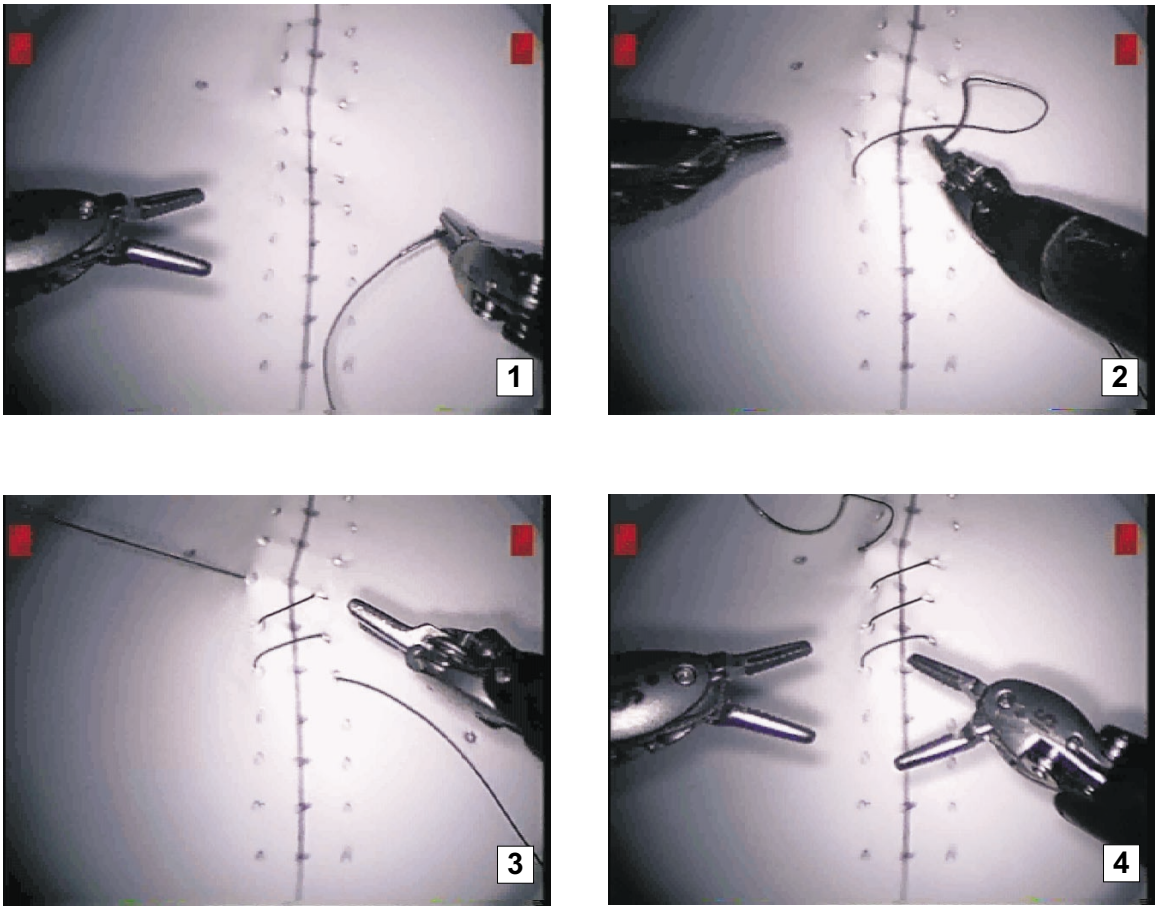

Figure 5.3: The suture task. 1) Retrieving the needle from the starting position. 2) Inserting the needle with the right tool. 3) Pulling the suture through the sheet with the left tool. 4) The starting and ending position; the task is complete.

was an individual with no formal medical training and less than 20 minutes of robot use prior to the experiment.

After reviewing the task performances, an eight-motion vocabulary was identified for the task. The eight motions can be used to classify all the observed motion and are defined as follows:

1. Motion of the right tool to retrieve the needle

2. Motion to position the needle for insertion

3. Needle insertion

4. Motion of the left tool towards the middle or top of the workspace

5. Motion of the right tool towards the middle or top of the workspace

6. Pulling the suture through with the left tool

7. Pulling the suture through with the right tool

8. Orienting the needle with both left and right tools

5.4.2 Manual Evaluation

Using the procedures described in Section 3.4, a transcription for each trial was manually identified. The transcription contains the times of transition between motions and a motion label for each segment. With these transcriptions we can make an analysis of each surgeon's performance.

Two general statistics, time and number of motions, immediately reveal differences between the three subjects. Subject #1, the most experienced of the three, required an average of 56.2 seconds to complete the task, while subject #2 used an average of 77.5 seconds and subject #3 (the least experienced) completed the task in an average of 82.5 seconds. Not surprisingly, there is a correlation between time and the number of motions used. The average number of motions used to complete the task were 19.0, 21.2, and 23.8 for subjects #1, #2, and #3, respectively. In fact, surgeon #1 used the same sequence of exactly 19 motions for all twelve repetitions of the task.

One observation of the experimenters was that the less experienced surgeons (subjects #2 and #3) typically used more time passing the needle (motion 3) than surgeon #1, particularly as they sought to guide the needle tip to emerge through the point marked on the sheet. Indeed, the manual analysis reveals that surgeon #1 spent a per-trial average of 23.5 seconds (41.8% of the total) using motion 3. Surgeon #2's per-trial average was 32.4 seconds (also 41.8% of the total) in use of motion 3. Surgeon #3 used 34.1 seconds (41.3% of the total) using motion 3 on average.

Yet another indicator of skill can be gathered from the transcriptions. Notably, surgeon #1 never resorted to using motion #8 (orienting the needle with both tools). When retrieving the needle from the starting position and when handing the needle from one tool to the other between suture throws, this surgeon was always able to grasp the needle in an orientation that allowed immediate procession to the next phase of the task. Surgeon #2 used this orienting motion twelve times; surgeon #3 used it seven times.

The transcriptions also reveal differences in the approach taken by each surgeon. Surgeon #1 used one particular pattern of motions repeatedly throughout the task. After pushing the needle through the sheet, this surgeon pulled the suture taught with the left tool and then handed the needle to the right tool for another round of positioning and insertion (motions 6, 4, 2, 3). The other two surgeons generally opted to pull the suture a portion of the way with the left tool and then hand the needle to the right tool, which was then used to pull the suture taught (motions 6, 7, 2, 3). This stylistic choice has performance ramifications. The average total time of use for motions 4, 6, and 7 on a per-trial basis for surgeon #1 was 16.1 seconds. This same

statistic for surgeons #2 and #3 was 24.7 and 27.8, respectively. It appears that choosing to pull the suture in two steps is a less time-efficient method. Additionally, by choosing to pull the suture across the "wound" with the right tool, surgeons #2 and #3 placed undue stress on the simulated tissue that ought to be avoided.

5.4.3 Automatic Evaluation

The first step in the automatic evaluation process is segmentation. The sum of squares algorithm, using combined velocity data from left and right hands and a variation which places transitions on only one side of peaks in the sum of squares, was used to produce an automatic segmentation. The average number of motions automatically identified for each surgeon was 19.5 for surgeon #1 (2.6% error), 16.4 for surgeon #2 (22.5% error), and 19.2 for surgeon #3 (17.5% error). The higher error rates for the less-experienced surgeons is not completely surprising—the motions executed by these surgeons were noticeably less deliberate and controlled than those of surgeon #1.

In practical use of a fully-automatic motion segmentation and recognition system the automatically segmented data would be sent to the HMM recognition, which would identify the most likely motion model to have produced each segment of data. This list of automatically-identified motions would then be used for analysis. In testing of such a system, however, this process is not helpful. If we wish to evaluate the performance of the recognition system we must have a complete transcription, which is not produced by the automatic segmentation. For this reason, in this experiment

the manual segmentation of the data was used for model training and recognition.

The data was prepared for use with the HTK using the scripts and procedure described in Section B.1 for an isolated recognition approach. Only data from surgeons #1 and #2 were used to train the models, as the motions from surgeon #3 were deemed too erratic to be useful. The training data included 15 total trials of the task by surgeon #1 and twelve by surgeon #2. All motions from surgeon #1's 15 recordings and twelve recordings each from surgeons #2 and #3 formed the training set (a total of 739 motions). Overall the system correctly recognized 58.05% of the motions. More detail regarding common errors can be understood by consulting the "confusion matrix" composed by the HTK. This matrix has been reproduced below.

```
------------------- Confusion Matrix -------------
      A   B   C   D   E   F   G   H  \% correct
   A 30   6   0   1   1   0   0   1  [76.9]
   B  2  83   2  14  10   0   0   0  [74.8]
   C  4   0  83   0   8   0  44  18  [52.9]
   D  0   0   0  93   0  36   0  32  [57.8]
   E 27  11   2   0  20   0   0   1  [32.8]
   F  0   0   0   8   0 104   0   5  [88.9]
   G  8   0   0  50   0   2   0  13  [ 0.0]
   H  1   0   0   2   0   0   1  16  [80.0]
------------------------------------------------
```

The labels A–H correspond to the motions 1–8. Reading across the matrix rows indicates how many times each motion was correctly recognized and how many times it was mistaken for another motion. For example, motion 1 (label A) was correctly recognized 30 out of 39 times, while it was mistaken for motion 2 (label B) six times and once each for motions 4 and 5 (labels D and E). Motion 7 (label G) was never correctly recognized—it was most often mistaken for motion 4 (label D). Although on first examination these two motions do not appear to share similar characteristics,

the definitions of motions in this vocabulary primarily focus on the movement of one tool. It is possible that the combination of motions from the left and right tools produced similar observations for these two motions and are thus easily confused.

5.4.4 Validation

A recognition rate of 58% was deemed too low for use in automatic evaluation, but the results of the manual analysis provide useful insight. These results suggest that surgeon #1 is the most skilled of the group—on average this surgeon used the fewest motions in the least amount of time to complete the task. Using the same criteria, surgeon #2 would be ranked as the second-most skilled of the group and surgeon #3 as the least skilled. This type of motion-based evaluation can be validated with the use of external metrics. The structure of this experiment provides for several metrics of this type. First, and probably most importantly, this analysis correlates perfectly with the amount of training and experience in both traditional and robot-assisted surgery for each surgeon. Experience is, in many ways, the *de facto* standard for judging surgical skill today. Second, although this task did not have a functional outcome associated with it, the GORE-TEX® sheet used as simulated tissue provides some objective clues to each surgeon's performance. It can be seen that surgeon #1 was able to target the needle exit points much more accurately than both surgeons #2 and #3. The exit points for surgeons #2 and #3 are surrounded by numerous holes created by the needle; two of the four exit points on surgeon #1's sheet have only one hole which was used for all repetitions of the task.

One interesting result of this analysis is that, although surgeon #2 has far more medical training than surgeon #3, surgeon #2 appears to fall somewhere in the middle between surgeon #1 and surgeon #3. This suggests skills gained from traditional surgical techniques are helpful in robot-assisted MIS, but do not directly correspond to efficient execution without additional training.

Chapter 6

Conclusions

Modern surgical training programs rely on the subjective assessment of senior surgeons to assess the technical skill of trainees. Clinicians and researchers alike desire a method for objective evaluation; the emerging field of robot-assisted surgery is an especially appropriate application for such evaluation. This essay describes a system that accomplishes the goal of objective evaluation by automatically segmenting a surgical procedure into separate motions, then recognizing these motions using statistical motion models acquired from training data. Motions used in the completion of a surgical task are used to assess skill; differences in the number, distribution, and sequence of motions may all be used to discriminate between more and less skilled surgeons.

We have accomplished automatic motion recognition by addressing two sub-problems of segmentation and recognition separately. Segmentation—identifying the transitions between motions—is done with algorithms using the sum of squares of the Cartesian velocity of the master manipulators. During motion the sum of squares is

generally much greater than zero, while during periods of inaction and motion transition it falls to approximately zero. The "peaks" and "valleys" in the sum of squares align well with the observed transitions between intuitively-defined motions within a task. The complete set of motions we would like to recognize form the motion vocabulary. Unlike analogous speech recognition systems, there is no predefined or standardized surgical motion vocabulary. We seek a vocabulary that is appropriately general, portable, meaningful, and amenable to automatic segmentation and recognition. The vocabulary defined for the suturing task described in Chapter 5 has many of these qualities.

Recognition, the process of identifying the motions between transitions, is accomplished with hidden Markov models (HMMs). We have applied HMMs at the motion level. That is, we train one HMM for each motion. Automatically segmenting the motions with the sum of squares algorithm enables use of an isolated motion recognition approach which has improved recognition rates in comparison to simultaneous segmentation and recognition under the HMM framework. Further gains in recognition rate have been achieved through the use of several additional techniques. Interpolation was applied to overcome problems associated with a low data collection rate imposed by the da Vinci® API. Linear discriminant analysis is used to map the motion data into a reduced-dimension feature space which maximizes the separation between data from different motions. Data transformed in this way has proven to be superior for model training and recognition. Normalizing the data reduces the effects of noise and measurement unit discrepancies and further enhances recognition when used together with LDA.

For fully objective evaluation, our approach relies first on an effective, accurate automatic segmentation, then on a recognition system with an acceptably-high recognition rate. Although many gains have been made, typical recognition rates are not (and likely never will be) a perfect 100%. Nonetheless, motions recognized by the system may still be used for skill evaluation. The system can be validated if differences in skill identified by the system agree with the amount of prior surgical experience, objective performance measures, and a subjective skill assessment the surgeons in question.

6.1 Future Work

There are many aspects of this topic that could be fruitful areas for future research. Under the HMM framework there are nearly countless variations, extensions, and techniques that could be applied in search of more consistent and accurate motion recognition. Other techniques could also be applied that may result in improved recognition. Principal component analysis (PCA) is one of these. PCA is often used in classification problems, and would most likely take the place of LDA if utilized here. If an isolated recognition approach is to be used, more sophisticated methods for automatic motion segmentation could also be investigated.

Perhaps more interesting would be further investigation regarding a generalized motion vocabulary. Additionally, identification of a grammar for this vocabulary could yield at least two valuable results. First, following the pattern of speech recognition systems, the grammar could be used to improve the results of the motion

recognition task. Second, farther down the road, identifying a surgeon's use of certain "grammatical idiosyncrasies" might be revealing of their skill or style.

A well-defined grammar could also play a role in a potential, far-flung application of this work that would enable surgical robots to function autonomously using motion models trained by human surgeons. A robot could potentially carry out oft-repeated surgical tasks such as knot-tying, or perhaps even greater portions of surgical procedures, particularly if enhanced with vision and other environmental sensors. If attempting to accomplish a higher level task, a robot would rely on a grammar to construct the proper sequence of motions.

Exploring alternate recognition techniques is another area of potential research. HMMs assume that each observation is independent from the last—an assumption that is clearly not true for observations such as positions in our motion data. Segmental Markov models [23] incorporate knowledge of dependence from observation to observation and thus model the trajectory of observations without independence assumptions. Yet another method, graphical models, are a generalization of hidden Markov models [59]. Graphical model systems can define arbitrary dependencies and learn parameters automatically from the data. Although this approach may not produce a meaningful motion vocabulary, improved recognition could justify such a choice.

Regardless of the recognition technique, there is a great deal of work to be done in surgical skill assessment using automatically recognized motions. Many of the motivations driving this research remain unfulfilled. Major initiatives include: further validation of this approach to skill evaluation by correlating with functional outcomes

of skilled tasks; integration with virtual reality surgical training systems; identification of learning curves for surgeons new to surgical robotic systems; and comparisons of skill between groups of surgeons trained differently. Our hope is that research in this area will yield practicable systems for objective skill evaluation that, when integrated with surgical training and feedback, will result in an improved standard of care for all surgical patients.

Appendix A

Segmentation Details

A.1 Manual Segmentation Methods, Techniques, and Results

The first time this manual segmentation procedure was carried out, it was done using the slow-motion replay feature on a VCR. This feature on the particular VCR used played back video at approximately 1/7 normal speed. Using a stopwatch, the timing of events marking a transition from one motion to another were recorded during the viewing. There is some room for interpretation in the timing of these events, and there is also room for error in the starting and stopping of the stopwatch. To mitigate the effect of these errors, each recording was viewed three times and the times obtained from each viewing were then averaged and to obtain a good estimate of the actual timing. If, after three viewings of a recording, the three transition times had a standard deviation greater than one, then the process was repeated for that

recording until all transition times agreed for favorably. This process seems to work well: although the error factors lent as much as 0.5 s of variation in the recorded time of events from viewing to viewing, this error translates to a mere $0.5/7 = 0.071$ s in real time. Recalling that the data is recorded at 10 Hz, this corresponds to an error that is less than the period between individual data points. Because in some cases the errors are positive and negative in others, averaging the three trials most likely balances these effects and produces the best estimate.

While this process leads to a good estimate of the timing of transition events as viewed on the video, it is necessary to correlate these times with the actual data. During recording of each task we took steps to try and make these things line up. Each task started with the da Vinci®tools held motionless with the grippers closed. Data recording was preceded with a countdown, and at the moment recording was initiated the grippers were rapidly opened by the test subject. Likewise, each task ended by closing the grippers and holding still. A stopwatch was also used to record the total time of each trial. After the video review process was complete, the total time for each recording was compared to the time extrapolated from the number of data points and the data collection rate. In all cases, the time extrapolated from the the data was larger than the time derived from adding the times obtained from the slow-motion observations. Moreover, the time recorded during the session and the time of the recording observed on the tape do not agree. On average, the difference was 0.59 s, or about 4%.

The reason for these discrepancies has not been conclusively determined. There were two primary hypotheses: 1) The motion did not start and stop at the same

time as the data recording 2) the VCR used did not play the video at real speed. To test the first hypothesis, the data was plotted and examined. In general, changes in position and velocity were often observed to lag the start of the recording by one or two data points (0.1-0.2 s). However, it also appeared that approximately the same number of data points were inappropriately clipped at the end of the data, yielding a zero net time addition. If the second hypothesis were true, we would expect to see a constant scaling between the time observed using the stopwatch during recording and the the time observed on the tape. No such scaling exists.

In the end, a constant scaling factor was used to adjust the transition times for each file to match the total number of data points. Because the scaling was administered to the entire recording, which contained six motions, the total change for each segment time was, on average, less than one data point. Therefore it is likely that, even in the worst-case, overall this method produced segment times which group only a handful of data points in the wrong motion.

A.2 Automatic Segmentation Algorithms

This section contains short descriptions of the Matlab script files (m-files) developed for automatic segmentation.

All of these scripts are variations on a central strategy for automatic segmentation using the sum of squares algorithm discussed in Section 3.5. The scripts with "combined" in the name combine the Cartesian velocities from the left and right master manipulators into a single sum of squares measure. Accordingly, scripts with "sepa-

rate" in the name consider the sum of squared Cartesian velocities from the left and right hands individually. In the tasks we have tested, motions are often performed with one hand at a time. Thus, any motion of the "inactive" hand may adversely affect the rise and fall of the sum of squares. The separate approach gave the best scores for the ring transfer task discussed in Section 3.3.

A second classification defines these algorithms. The "altered" tag refers to the style of segmentation in which "reach and orient" are defined as one motion. This style of motion definition manifests itself in the automatic segmentation essentially as a segment at the front of each peak, rather than in front and behind.

All of these scripts make use of smoothed data and thresholds for determining "peaks" and "valleys." These parameters can be varied to maximize performance with a particular data set.

combined_evaluation.m This script uses a smoothed sum of squares of the master Cartesian velocities to estimate segment times. The algorithm finds local maxima and then looks for minima below some threshold. This version considers the velocities of the left and right masters together and places a transition on the front and back of each peak.

combined_evaluation_altered.m This script uses a smoothed sum of squares of the master Cartesian velocities to estimate segment times. The algorithm finds local maxima and then looks for minima below some threshold. This version considers the velocities of the left and right masters together but, unlike combined_evaluation.m, places a transition only on the front of each peak.

separate_l_and_r_evaluation.m This script uses a smoothed sum of squared Cartesian velocities from each of the master manipulators individually to finds local maxima and minima. Transitions are identified on the front and back of each peak.

separate_l_and_r_evaluation_altered.m This script uses a smoothed sum of squared Cartesian velocities from each of the master manipulators individually to finds local maxima and minima. Transitions are identified only on the front of each peak.

combined_evaluation_fourth_power.m This script is essentially the same as combined_evaluation.m, but instead of the sum of squared velocities, it sums the velocities after taking them to the fourth power. Because the velocities are typically less than one, this has the effect of nearly eliminating low-magnitude velocities from consideration. This version considers the velocities of the left and right masters together and places a transition on the front and back of each peak.

Appendix B

Recognition Details

B.1 Procedures

B.1.1 Primary Requirements

In the system framework we have created there are four basic phases required to go from the raw data collected during a task performance to the recognition results obtained from the HTK [17, 56]. These steps are 1) segment the data, 2) manipulate the data and create files about the data using a collection of Matlab scripts, 3) format the data files and compile additional information about the data using HTKWrite, 4) use a Perl script to execute the HTK tools. The following is a more detailed outline for each of these steps.

- Segment the data (Chapter 3)
  - Decide which motions are to be recognized
  - Determine the place at which each motion starts and stops in the data

- Create files containing information about the data (details about the structure of each file follow this section)
  - Each recording needs three files
    - `data.dat` – one for each recording; contains the raw data such as positions, velocities, forces, etc. that form the observation vector for the HMM. Could also be transformed data from a process like linear discriminant analysis.
      - determine which variables will be included in the observation vector
      - use Matlab to manipulate the data, select the proper columns, and write the file
    - `seg_data.dat` – one for each recording; has information about which motions were used and times of transition between them
    - `data_points.dat` – one for each recording; refers to the number of rows in the file
  - Each data set gets one additional file
    - `dimensions.dat` – one of these for each set of data
  - Put the files into the proper structure in the `Input_data` directory in the HTK folder
    - one folder for each training/test file, labeled consecutively (tr1,tr2,tr3,etc., or te1,te2,te3, etc.), containing all three .dat files for that data

- Format the data (both training and test) and create files needed for the HTK using the HTKWrite program (written in C++). It does the following:
  - reads in information regarding the total quantity of data
  - iterates through each of the **training** files
    - determines total number of segments and transition times from `seg_data.dat`
    - reads in recorded data from `data.dat`
    - writes the .usr file containing properly formatted data and the HTK header
    - writes the .lab file containing segment labels and transition times
  - repeats the above process for each of the **test** files
  - creates `bcplist` files
  - creates `bcpvocab` files
  - for both test and training files, writes file locations and file stubs (filename string without extension) for perl script
  - creates `protoconfs` files
  - creates network files
  - creates lattice file – However, there is something wrong with the HTKWrite code and the network and lattice files are not created with HTKWrite, they must be done by hand. And actually this may be for the best, since I'm not sure how it would deal with the isolated setup anyway.

        * the lattice file defines the inter-model transitions that are allowed. By doing so, it defines the grammar for task. See the HTK documentation for the proper format.

    - creates the network file: not sure what it does. I believe it is not used in the current configuration.

- Run the HTK – uses a Perl script to call each of the separate HTK tools

    - a Perl interpreter must be installed on the machine for this to work. I have been using the most recent one available from http://www.activestate.com/

    - the script may be modified to call each of the HTK tools with different parameters, which can result in different model topologies, different thresholds, different output being printed, etc.

        * Some of the most important parameters that I have varied in the past have been the number of states in each model, the pruning threshold, and the model transition penalty. The number of states can be changed by modifying the `proto_s1_m1_dc.pcf` file (or whatever file the Perl script is called with, which contains this information in the proper structure) in the `\HTK\protoconfs` directory. The pruning threshold is changed by altering the way the HVite tool is called under the '-t' parameter. The model transition penalty can be changed by altering the "$pOptStr" parameter in the "TESTING" section of `rundemo.pl`.

B.1.2 Detailed Procedure

A collection of Matlab scripts and C++ programs have been written to manipulate and create the files and folders necessary to actualize the four phases described above. This procedure is visualized in the flowchart of Figure B.1 and outlined in further detail here:

- If the data is to be interpolated, run `dv_spline_interp.m`
  - creates new logfiles with the interpolated points in a new directory
  - creates a new `segment_times.dat` file with the recalculated segment times
- If the data is to be normalized, run `dv_normalize.m`
  - creates new logfiles with normalized columns using the global mean and standard deviation
- if the dimension of the observation vector is to be reduced with a linear discriminant analysis (LDA), run the appropriate version of the `dv_lda` scripts. For tasks with a defined, invariant sequence of motions use `dv_lda.m`, tasks with a variable sequence of motions require the use of `dv_lda_variable.m`
  - creates new logfiles with the data having been transformed and reduced in dimension
- run the appropriate version of the `dv_create_files` scripts (depends on system architecture, structure of training data, and segmentation method).
  - specify which files are train/test/neither/both

  - for each file used, copies logfile to `data_all.dat` in appropriate `Input_data` directory
  - uses transition times from segmenting process and creates `seg_data.dat`
  - creates `dimensions.dat` for the data set

- run the `dv_select_columns.m` script.
  - asks for desired columns and writes `data.dat` for all test and training files
  - creates `data_points.dat` for each file

- run HTKWrite (located in the HTK folder)
  - edit/create the `monLattice` file by hand

- run the HTK Perl script
  - evaluate results, get disappointed, throw up hands in exasperation

B.2 Data Preparation Algorithms

This section contains short descriptions of the Matlab script files (m-files) and C++ programs most commonly used to prepare data for automatic recognition with the HTK. Other scripts exist in the library with these; if a script described here doesn't meet the needs of a future task, it would be worth checking to see that a script which performs the desired task doesn't already exist in some form. The files here are listed in alphabetical order.

dv_create_files.m This script queries the user for a designation for each logfile (either train, test, or neither) and then creates the `seg_data.dat` file for all the files to be used. This step requires a transcription of motions; this version of the script assumes an invariant sequence of motions in the task. It then copies the logfile into the appropriate `tr*` or `te*` directory as `data_all.dat`. It also creates a file in this directory, `source.txt`, which records which logfile was copied into this directory. Lastly, it creates the `dimensions.dat` file that is needed for every data set—it identifies the number of training and test files in the set. This version is used for continuous motion recognition.

dv_create_files_autoseg.m This version of `dv_create_files.m` is designed to deal with an automatic segmentation. A manual segmentation is used for the training files and the automatic segmentation is used for the test files. The script is able to cope with the fact that the automatic segmentation may not have the same number of motions for each file. This version is used for continuous motion recognition.

dv_create_files_isolated.m This version of `dv_create_files.m` is designed for use with isolated motion recognition. In the isolated case the transcription is used first to create the `seg_data.dat` file for all the files to be used. Second, it uses the transcription to break up the original logfiles and use one-motion portions to create the `data_all.dat` files in the `tr*` and `te*` folders. It still creates a file in this directory, source.txt, which records which motion in which logfile was used to create the `data_all.dat`.

dv_create_files_variable.m This version of `dv_create_files.m` is for dealing with data which has many repetitions of each motion in the recordings. The HTK allows for only one example of each model in any training file, but we want to use the extra repetitions for training too, so this script breaks up these recordings to form multiple training files. The script can also handle transcriptions with varying number of motions. This script prepares files for isolated motion recognition.

dv_lda.m This script performs a linear discriminant analysis on the data to enable a reduction in dimensionality and a mapping to a new feature space. The script calculates the necessary elements and displays an ordered plot of the eigenvalues of the $\overline{W}^{-1}\overline{T}$ so the user can make a judgement regarding the dimension of the new feature space. It then creates new, transformed logfiles that can be used in other scripts in the same way as the original logfiles.

dv_lda_variable.m This version of `dv_lda.m` has been modified to deal with data with a varying number of motions in each recording; it is otherwise identical to

the original version.

dv_normalize.m This script normalizes all of the data in a set and creates new, normalized logfiles which may be used as input to other scripts in the library the same way as the original files.

dv_select_columns.m This script deals with three files needed for the HTK process. A `data.dat` file is created that contains only the observations of interest from the complete set contained in `data_all.dat`. Since the introduction of LDA into our process, `data.dat` is often a direct copy of `data_all.dat`, as the LDA process essentially selects the desired observations. The `dimensions.dat` file for the entire set is modified to include the number of columns in the data. Lastly, the `data_points.dat` file is also created for each recording.

dv_spline_interp.m This script is used to interpolate daVinci data with a third order cubic polynomial. It calls a subroutine, `dv_spline_interp_sub.m` which performs a simple numerical differentiation in order to have enough values to solve for the coefficients of the polynomial. It creates new, interpolated logfiles which can be used as input to other scripts.

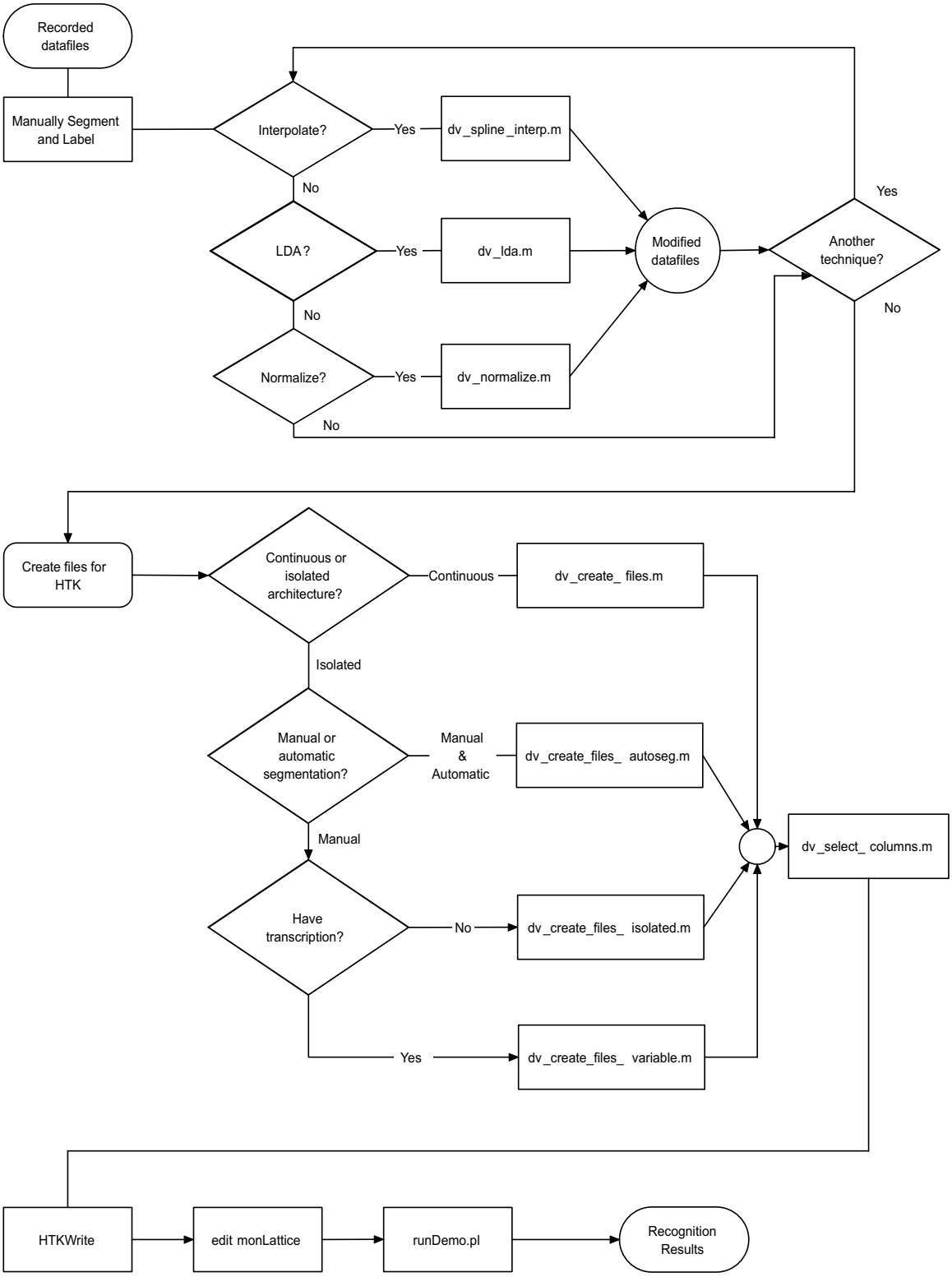

Figure B.1: System Procedure

Bibliography

[1] L. E. Baum and J. A. Egon. An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology. *Bulletin of the American Meterology Society*, 73:360–363, 1967.

[2] L. E. Baum and G. R. Sell. Growth functions for transformations on manifolds. *Pacific Journal of Mathematics*, 27(2):211–227, 1968.

[3] J. E. Bobrow, B. Martin, G. Sohl, E. C. Wang, F. C. Park, and J. Kim. Optimal robot motions for physical criteria. *Journal of Robotic Systems*, v 18(n 12):785–795, December 2001.

[4] M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. pages 994–999, 1997.

[5] P. F. Brown. *The acoustic-modeling problem in automatic speech recognition.* PhD thesis, Carnegie Mellon University, Department of Computer Science, 1987.

[6] C. Cao, C. MacKenzie, and S. Payandeh. Task and motion analyses in endoscopic surgery. In *ASME IMECE Conference Proceedings: 5th Annual Symposium on*

Haptic Interfaces for Virtual Environment and Teleoperator Systems, pages 583–590, 1996.

[7] P. Cohen, B. Heeringa, and N. Adams. Unsupervised segmentation of categorical time series into episodes. In *Proceedings IEEE International Conference on Data Mining*, pages 99–106, December 2002.

[8] S. Cotin, N. Stylopoulos, M. Ottensmeyer, P. Neumann, D. Rattner, and S. Dawson. Metrics for laparoscopic skills trainers: the weakest link! In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 2488 of *Lecture Notes in Computer Science*, pages 35–43, 2002.

[9] A. Darzi and S. Mackay. Assessment of surgical competence. *Qual Health Care*, 10(Suppl. 2):pp. ii64–ii69, December 2001. ISSN: 0963-8172.

[10] A. Darzi and S. Mackay. Skills assessment of surgeons. *Surgery*, 131(2):121–124, February 2002. ISSN: 0039-6060.

[11] A. Darzi, S. Smith, and N. Taffinder. Assessing operative skill: needs to become more objective. *British Medical Journal*, 318:887–888, 1999.

[12] V. Datta, S. Mackay, A. Darzi, and D. Gillies. Motion analysis in the assessment of surgical skill. *Computer Methods in Biomechanics and Biomedical Engineering*, 4:515–523, 2001.

[13] V. Datta, S. Mackay, M. Mandalia, and A. Darzi. The use of electromagnetic motion tracking analysis to objectively measure open surgical skill in the laboratory-

based model. *Journal of the American College of Surgery*, 193:479–485, November 2001.

[14] V. Datta, M. Mandalia, S. Mackay, A. Chang, N. Cheshire, and A. Darzi. Relationship between skill and outcome in the laboratory-based model. *Surgery*, 131(3):318–323, March 2001. ISSN: 0039-6060.

[15] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing (see also IEEE Transactions on Signal Processing)*, Vol.28(Iss.4):357– 366, August 1980.

[16] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39 (Series B):1–38, 1977.

[17] Cambridge University Engineering Department. The hidden Markov model toolkit (HTK v3.1). online at http://htk.eng.cam.ac.uk/, 2004.

[18] G. R. Doddington. Phonetically sensitive discriminants for improved speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 556–559, May 1989. ICASSP-89.

[19] P. Dubois, Q. Thommen, and A. C. Jambon. In vivo measurement of surgical gestures. *IEEE Transactions on Biomedical Engineering*, 49(1):49–54, January 2002.

[20] A. Fod, M. J. Mataric, and O. C. Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 12(1):39–54, January 2002.

[21] G. D. Forney. The Viterbi algorithm. In *Proceedings of the IEEE*, volume 61, pages 268–278, March 1973.

[22] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Computer Science and Scientific Computing. Academic Press, Inc. Harcourt Brace Jovanovich, Boston, 2nd edition, 1990.

[23] M. J. F. Gales and S. J. Young. The theory of segmental hidden markov models. Technical Report Technical Report CUED/F-INFENG/TR.133, Cambridge University Engineering Department, June 1993.

[24] A. G. Gallagher, N. McClure, J. McGuigan, I. Crothers, and J. Browning. Virtual reality training in laparoscopic surgery: a preliminary assessment of minimally invasive surgical trainer virtual reality (MIST VR). *Endoscopy*, 31(4):310313, May 1999.

[25] A. G. Gallagher and R. M. Satava. Virtual reality as a metric for the assessment of laparoscopic psychomotor skills. *Surgical Endoscopy*, 16(12):1746 – 1752, December 2002. ISSN: 0930-2794.

[26] S. Geirhofer. Feature reduction with linear discriminant analysis and its performance on phoneme recognition. Available online at http://www.ifp.uiuc.edu/speech/pubs/2004/geirhofer.pdf, May 2004. Uni-

versity of Illinois at Urbana-Champaign, Department of Electrical and Computer Engineering.

[27] The Chamberlain Group. Robotic trainer (items 4008 and 4014). 934 South Main Street Great Barrington, MA 01230 USA. http://www.thecgroup.com/.

[28] G. S. Guthart and J. K. Salisbury. The Intuitive telesurgery system: Overview and application. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 618–621, April 2000.

[29] J. D. Hernandez, S. D. Bann, Y. Munz, K. Moorthy, V. Datta, S. Martin, A. Dosis, F. Bello, A. Darzi, and T. Rockall. Qualitative and quantitative analysis of the learning curve of a simulated surgical task on the da Vinci® system. *Surgical Endoscopy*, 18(3):372–378, March 2004. ISSN: 1432-2218.

[30] J. Hu, M. K. Brown, and W. Turin. Handwriting recognition with hidden Markov models and grammatical constraints. In *Proceedings of the 4th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 195–205, 1994.

[31] C. S. Hundtofte, G. D. Hager, and A. M. Okamura. Building a task language for segmentation and recognition of user input to cooperative manipulation systems. In *10th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 225–230. IEEE, 2002.

[32] Intuitive Surgical, Inc. The da Vinci® surgical system Application Programming

Interface (API) reference guide, 2001. 950 Kifer Road; Sunnyvale, CA; 94086. ph: 888-868-4647.

[33] M. S. Khan, S. D. Bann, A. Darzi, and P. E. Butler. Assessing surgical skill. *Plast Reconstr Surg*, 112(7):1886–1889, December 2003. ISSN: 0032-1052.

[34] M. Li and A. M. Okamura. Recognition of operator motions for real-time assistance using virtual fixtures. In *11th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 125–131. IEEE, March 2003.

[35] A. Liu and D. D. Salvucci. Modeling and prediction of human driver behavior. In *Proceedings of the 9th International Conference on Human-Computer Interaction*, pages 1479–1483. Lawrence Erlbaum Associates, 2001.

[36] S. Mackay, V. Datta, M. Mandalia, P. Bassett, and A. Darzi. Electromagnetic motion analysis in the assessment of surgical skill: relationship between time and movement. *ANZ Journal of Surgery*, 72(9):632–634, September 2002. ISSN: 1445-1433.

[37] C. L. MacKenzie, J. A. Ibbotson, C. G. L. Cao, and A. J. Lomax. Hierarchical decomposition of laparoscopic surgery: a human factors approach to investigating the operating room environment. In *Minimally Invasive Therapy and Allied Technologies (MITAT)*, volume 10 of *3*, pages 121–127, 2001. Special Issue on endoscopic surgery: Materials-Functions-Ergonomy.

[38] J. A. Martin, G. Regehr, R. Reznick, H. MacRae, J. Murnaghan, C. Hutchison,

and M. Brown. Objective structured assessment of technical skill (OSATS) for surgical residents. *British Journal of Surgery*, 84:273–278, 1997.

[39] K. Moorthy, Y. Munz, S. K. Sarker, and A. Darzi. Objective assessment of technical skills in surgery. *British Medical Journal*, 327(7422):1032–1037, November 2003.

[40] T. E. Murphy, C. M. Vignes, D. D. Yuh, and A. M. Okamura. Automatic motion recognition and skill evaluation for dynamic tasks. In *Eurohaptics*, pages 363–373, July 2003.

[41] T. E. Murphy, R. J. Webster, and A. M. Okamura. Design and performance of a two-dimensional tactile slip display. In *Eurohaptics*, pages 130–137, June 2004.

[42] R. V. OToole, R. R. Playter, T. M. Krummel, W. C. Blank, N. H. Cornelius, W. R. Roberts, W. J. Bell, and M. R. Raibert. Measuring and developing suturing technique with a virtual reality surgical simulator. *Journal of the American College of Surgeons*, 189(1):114–127, July 1999.

[43] A. R. Peters, C. L. Campbell, W. J. Bluethmann, and E. Huber. Robonaut task learning through teleoperation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2806–2811, September 2003.

[44] P. K. Pook. *Teleassistance: Using Deictic Gestures to Control Robot Action*. PhD thesis, Department of Computer Science, University of Rochester, 1995.

[45] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings Of the IEEE*, 77(2):257–286, February 1989.

[46] J. Rosen, B. Hannaford, C. G. Richards, and M. N. Sinanan. Markov modeling of minimally invasive surgery based on tool/tissue interaction and force/torque signatures for evaluating surgical skills. *IEEE Transactions on Biomedical Engineering*, 48(5):579–591, May 2001.

[47] J. Rosen, M. Solazzo, B. Hannaford, and M. Sinanan. Task decomposition of laparoscopic surgery for objective evaluation of surgical residents' learning curve using hidden Markov model. *Computer Aided Surgery*, 7(1):49–61, 2002.

[48] J. Shah and A. Darzi. Surgical skills assessment: an ongoing debate. *BJU Int*, 88(7):655–660, November 2001. ISSN: 1464-4096.

[49] J. Solis, C. A. Avizzano, and M. Bergamasco. Teaching to write Japanese characters using a haptic interface. In *10th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 255–262, March 2002.

[50] T. Starner and A. Pentland. Visual recognition of american sign language using hidden Markov models. In *Proceedings International Workshop on Automatic Face Gesture Recognition*, pages 189–194, 1995.

[51] L. Verner, D. Oleynikov, S. Holtman, H. Haider, and L. Zhukov. Measurements of the level of expertise using flight path analysis from da Vinci® robotic surgical system. In R. Phillips, J.D. Westwood, R.A. Robb, D. Stredney, H.M. Hoffman, and G.T. Mogel, editors, *Medicine Meets Virtual Reality 11*, volume 94 of *Studies in Health Technology and Informatics*. IOS Press, 2003.

[52] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269, 1967. 1967.

[53] D. Wilhelm, K. Ogan, C. G. Roehrborn, J. A. Cadeddu, and M. S. Pearle. Assessment of basic endoscopic performance using a virtual reality simulator. *Journal of the American College of Surgeons*, 195:675–681, 2002.

[54] Y. Yamauchi, J. Yamashita, O. Morikawa, R. Hashimoto, M. Mochimaru, Y. Fukui, H. Uno, and K. Yokoyama. Surgical skill evaluation by force data for endoscopic sinus surgery training system. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 2488, pages 35–43. Lecture Notes in Computer Science, 2002.

[55] G. Ye, J. J. Corso, and G. D. Hager. Gesture recognition using 3D appearance and motion features. In *Proceedings of CVPR 2004 Workshop on Real-Time Vision for Human-Computer Interaction*, 2004.

[56] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. The HTK book (HTK v3.1). Online at http://htk.eng.cam.ac.uk/docs/docs.shtml, 2002.

[57] G. Yu, W. Russell, R. Schwartz, and J. Makhoul. Discriminant analysis and supervised vector quantization for continuous speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 685–688, April 1990. ICASSP-90.

[58] S. A. Zahorian, D. Qian, and A. J. Jagharghi. Acoustic-phonetic transformations for improved speaker-independent isolated word recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 561–564, April 1991. ICASSP-91.

[59] G. Zweig, J. Bilmes, T. S. Richardson, K. Filali, K. Livescu, P. Xu, K. Jackson, Y. Brandman, E. Sandness, E. Holtz, J. Torres, and B. Byrne. Structurally discriminative graphical models for automatic speech recognition - results from the 2001 Johns Hopkins 2001 summer workshop. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002.

Curriculum Vita

Todd Murphy was born on July 24, 1978 in Pasadena, California and raised in Loveland, Colorado. After graduating valedictorian of his class at Loveland High School in 1997, he matriculated at Rensselaer Polytechnic Institute in Troy, New York and went on to receive a Bachelor of Science *cum laude* in Mechanical Engineering in December of 2001. Between 1999 and 2002 Mr. Murphy gained practical experience interning first at Honeywell in Fort Collins, CO, and later at iRobot Corporation in Somerville, MA.

Mr. Murphy began full-time graduate studies at The Johns Hopkins University in Baltimore, Maryland during the fall of 2002. At Hopkins, Mr. Murphy's primary focus was researching methods for objective evaluation of surgical skill, performed under the guidance of Dr. Allison Okamura in Mechanical Engineering and Dr. David Yuh from the Department of Cardiac Surgery at the Johns Hopkins Hospital. Also during his tenure at Hopkins, Mr. Murphy was responsible for the design and construction of a novel 2 DOF tactile slip display [41], a device with potential applications in psychophysics, virtual reality, and teleoperation.