Chapter 1

Introduction

1.1 Motivation

Programming by demonstration can be used to transfer human skills to a robot. These skills can be directly used to repeat a task as it was observed or combined with other similar performances of the task in order to learn a more general account of the skill. The first step involves identifying primitives that can be used to represent the task. Clustering and Hidden Markov Models (HMM) [19] are used to identify skill primitives, as a sequence of Qualitative States (QS), or sensory clusters in the NASA Robonaut [2][42] for a reach and grasp task. Our motivation is to gain a better understanding of the patterns, which aids in the selection of appropriate primitive and skill representations.

1.2 Overview

Skill acquisition is performed by identifying mappings from sensory signals into QSs, the construction of a QS automaton, and the detection of output or motor commands that are used to transition the system from the present QS to the next desired QS. These states may reflect primitive behaviors, complex combinations of behaviors, deliberate decisions and actions occurring at higher mental levels, or other internal processes. The types of skills that we are modeling reflect a teleoperated service and maintenance task performed on the NASA Robonaut. It is not likely that the QSs are known ahead of time, but rather are assumed to emit observations that can be exploited in order to discover and compute them as patterns.

The Fuzzy C-Means (FCM) [11], Possibilistic C-Means (PCM) [38], and the Robust Competitive Agglomerate (RCA) [9] clustering algorithms are used to identify skill primitives

in the NASA Robonaut sensory feature space. These clustering algorithms were selected for two reasons, (1) to look at different types of patterns and (2) to avoid specifying the number of clusters for the FCM and PCM. The PCM is a mode seeking algorithm, while the FCM and the RCA look for partitions. These are fundamentally different types of patterns, which is significantly different from just changing the proximity metrics inside the clustering algorithms. The second aspect involves how one must specify the number of clusters for the FCM and PCM, while the RCA looks for the number of clusters while the algorithm is running.

Another approach to robot skill learning involves the application of HMMs. The Baum-Welch [13] algorithm, a generalization of the Expectation Maximization procedure [1], was used to re-estimate the parameters for an HMM. The Baum-Welch identifies skill primitives, which are now probability distributions, and the state transition model in a single unified framework. More importantly, they estimate all of these parameters from the present model, which is significantly different from clustering. This means that HMMs take into account the temporal aspect of the problem when estimating the model parameters. HMMs were used in order to help validate the earlier robot skills that were obtained through clustering.

Multiple techniques have been applied in order to support our findings. Cluster validity, a procedure that is separate from clustering, was used to help support the number of cluster prototypes. The QS automaton is presently being manually scored, but is compared to earlier segmentation results found by Peters[40]. The clustering and HMM results are extremely similar, in terms of QS transition points and task symmetry, and under certain circumstances are shown to correspond with Peters Sensory Motor Coordination (SMC) segmentation results. This investigation into skill learning has been successful, but still has two primary steps before it can become a fully automatic process. These steps involve a way to automatically perform feature selection and a method to score the extracted QS automata.

Chapter 2

Background

2.1 Domain Information

2.1.1 NASA Robonaut

Robonaut is a robotics project collaboration undertaken by a team of engineers at the Dexterous Robotics Laboratory (DRL) of the Automation, Robotics, and Simulation Division at Johnson Space Center in Houston, Texas [2][42]. One of the goals for Robonaut is to eventually work in outer space as a substitute for a human, performing maintenance functionality on the outside of the space shuttle. Robonaut, for the moment, is fully controlled and manipulated by a teleoperator.

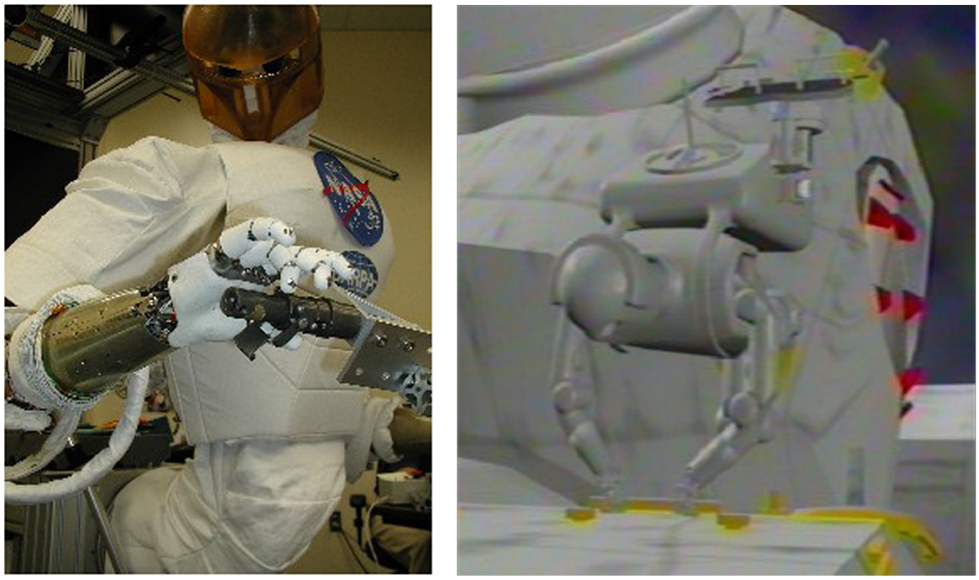

Figure 2.1: (left) Robonaut performing a grasping task and (right) a virtual simulation of Robonaut performing space maintenance on the outside of a space shuttle.

Teleoperation refers to the process of controlling a robot by exchanging sensory and motor information from a remote location. The Roboanut teleoperator is fully immersed in a virtual environment and can see and hear what the robot senses through a helmet. Arm and body sensors on the operator are translated into similar controls on Robonaut's body. Gloves with sensors are used to designate Robonaut's finger positions. However, there is no haptic, force, torque, or proprioceptive feedback for the teleoperator. Figure 2.2 demonstrates how a teleoperator controls the Robonaut.

Figure 2.2: (left) Robonaut grasping an object and (right) a teleoperator emerged in the Robonaut virtual environment.

The context of the work described here was for a teleoperated reach and grasp task. The reach and grasp a wrench task was performed through teleoperation 9 times on the NASA Robonaut, where each time the wrench was located in a different location. Each location was performed 5 times (R repetitions). The 9 locations (L) corresponded to the 8 vertices and center of a hypercube. This volume is the workspace that is used later to combine repetitions to create a general representation for the task. The p^{th} location and r^{th} repetition can now be represented as a sequence of T sensor and motor signals, $V_{(p,r)} = (\vec{x_1} \ldots \vec{x_T})$. $V^t_{(p,r)}$ denotes the t^{th} $(1 \leq t \leq T)$ sample in the r^{th} repetition of the p^{th} location. Each $\vec{x_t}$ is the product of K different sensor or motor signals.

The reach and grasp task consists of 6 episodes: (1) the reaching of the right hand towards the wrench, (2) envelop its handle with the wrench, (3) grasping of the handle, (4) tug on it (pulling it back towards the chest slightly), (5) release of the grasp, and (6) the

withdrawal of the hand back to the right side of the robot [40]. We used this high level description of the task initially to select the number of clusters for our algorithms.

There are two versions of the teleoperated reach and grasp task that we analyzed. The first data set, segmented and time-normalized by Peters (described in detail in the related work section), represents the task as a series of Sensory Motor Coordination (SMC) events and episodes. SMC episodes denote the state of the robot between consecutive events, which are motor actions that occur in response to some sensory stimilus. This transformed data provides a starting place to understand and verify our findings. The second data set is the raw sensory-motor recordings of the task. The raw data results demonstrate our ability to directly take the recordings and generate tangible and useful QS sequences.

2.1.2 Skill Representation

The primitives that we acquire through clustering or with HMMs are used to model robot skills. Our approach relies on pattern recognition to identify these primitives, but represents skills as finite state automata. Finite state automata are a simple, classical, and popular way to model a variety of processes and computational models. Exactly how to represent, acquire, and refine skills is a topic that varies greatly in the domain and literature. Some methods include programming by demonstration[26], learning through imitation[23], sensor to motor function approximation with neurological networks, reinforcement learning models in a connectionist representation[48], and even temporal pattern recognition with HMMs[14]. Teleoperation can be used as a learning by demonstration method of transferring skills from a human to a robot.

Our assumption is that a skill is a broad concept that encapsulates a multitude of static or dynamic processes that have the possibility of being refined, combined, or updated through experience. Skills can range from low level motor processes to higher level cognitive and mental routines. In the case of the NASA Robonaut, we are studying skills that relate to service and maintenance functionality. We take a three step approach that skills can be represented and acquired mathematically through (1) identifying mappings from sensory signals to QSs, (2) constructing a QS automaton, and (3) learning the output or motor

commands associated with transitioning the system from the present QS to the next desired QS [27].

We place the skill acquisition emphasis on more mathematical and quantitative procedures. Yan, Xu, and Chen[14] note a few important aspects that impact the representation and selection of procedures in the area of skill acquisition. The following is a summary of these key points:

- Is the skill learnable in terms of linear or non-linear relationships between stimulus and response?
- Is the skill time invariant or not
- Can the skill be broken down and represented as a discrete and stochastic process?
- How generalizable is the skill?
- To what degree, if any, is the skill decomposable into smaller subdivisable units?

Projects in this area can typically be divided into one of the following three learning categories: supervised, unsupervised or reinforcement learning. These techniques are also categorized by whether the learning process is more automatic and data driven or preprogrammed. Our technique can be labeled as an unsupervised method that is data driven.

2.1.3 Sensory-Motor Coordination

Sensory-motor coordination (SMC) is a method that can help categorize sensory patterns in an environment without relying on high-level input processing procedures. Pfeifer discusses how sensory and motor information can be coupled to form descriptions called SMC events for robot control strategies [33]. He shows four separate robot control architectures that utilize SMC at different levels to couple sensing and acting for navigation situations.

SMC is not limited to robotics, but rather has its roots in psychology and human development theory. This methodology can be useful in building intelligent agents from experience and the process of embodiment. This way of thinking is somewhat different from the traditional, symbolic views conventionally held in Artificial Intelligence. The traditional, symbolic views have powerful application, but suffer from their intrinsic inability to understand the concepts that they represent. SMC is concerned with acquiring a more empirical

and experience-driven notion of a category, which is how it internally distinguishes between objects from different classes. The category being learned is derived through the contribution of different internal motor, sensor, and possibly mental sources.

Pfeifer, in his initial paper on SMC[33], illustrates this subject by describing the difference between the classical method of learning and developing a category for the concept of an apple, and the change in behavior of an agent associated with sensing and interacting with an apple. This initially appears to only be a slight conceptual shift, but its effects have a larger impact in terms of underlying agent or model structure and learning procedures. This approach helps to build intelligent agents that have distributed knowledge systems and behavior networks that are refined through embodiment. Pfeifer used this in a control theory context by learning and refining behaviors based on sensor to motor mappings through a Hebbian learning scheme with bidirectional active forgetting [33].

Peters defines an SMC event as "a motor event which occurs in response to a sensory stimulus, or a sensory stimulus that occurs in response to a motor action" [33](p.2). Peters also defines an SMC episode as a "state of the robot's sensors and actuators between consecutive SMC events" [33](p.2). These definitions are best understood through an example of an agent in its environment. Figure 2.3 demonstrates an example of a wall following task and its corresponding SMC episodes and events according to the sensory-motor signals of an agent. The goal of the agent is to stay as close to the wall without hitting it. The agent is moving and following the wall (episode 0) until it comes across the new hall (event 1). The agent is not forced to turn around and try to follow the new wall (episode 1). The agent finally detects the wall on its right again (event 2) and at this point goes back to the wall following behavior (episode 2).

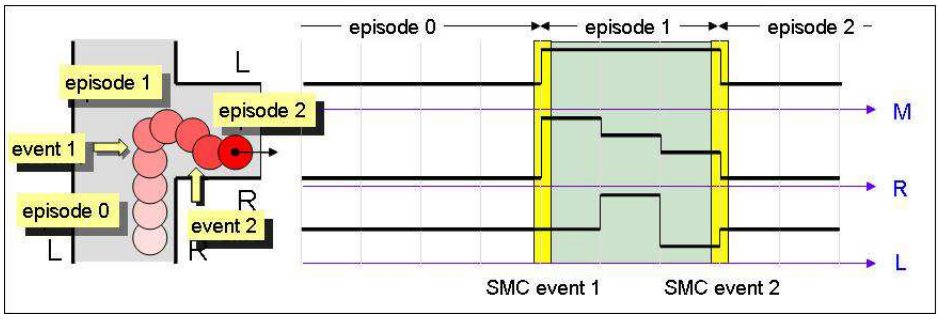

Figure 2.3: SMC illustration provided by Peters [36]. This picture shows a robot wall following task (left) and the robots corresponding sensor and motor recordings (right)

2.1.4 Feature Space

Many pattern recognition techniques search for patterns in what has been coined feature space[46][41]. Our approach takes this popular assumption that features can be combined to construct a cross product space, where patterns are exposed and may be investigated. Problems that do not fit under this classification can be approached through other well known and respected techniques[24]. Some of these techniques include language and grammar theory, linear regression, time series analysis of trends, and even algorithms based on genetic and biological processes.

The following notations will be used throughout the remainder of this work to describe feature space. Assume that the problem of interest is composed of N different sources of information. Each of these individual sources of information, which we will call features, will be denoted as $f_i, (i = 1 \ldots N)$. A subset of these features will be selected and denoted in vector notation as $\vec{v}$ ($\vec{v^t} =< f_1 \ldots f_d >$ where $1 \leq d \leq N$). Figure 2.4 demonstrates a few typical examples of two dimensional patterns in an arbitrary feature space.

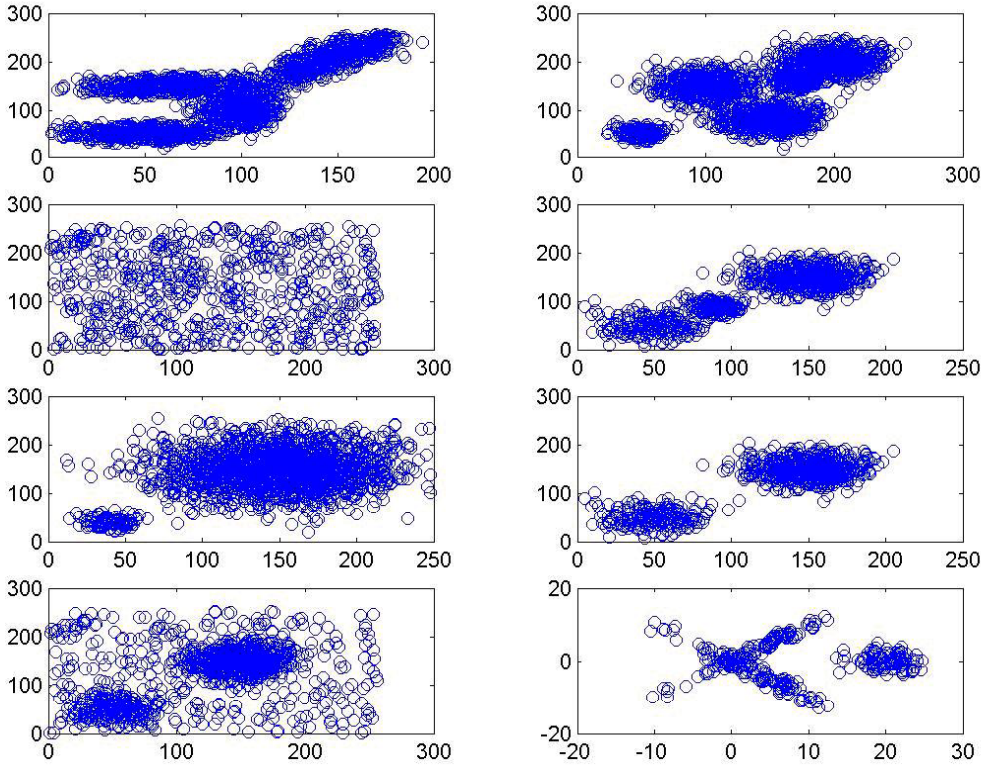

Figure 2.4: Two dimensional synthetic patterns in feature space.

The examples shown in Figure 2.4 illustrate the difficulty related to formulating a set of criteria about what constitutes a pattern. If even a type of pattern for the data sets in Figure 2.4 can be identified, such as a quadratic elliptical shape, there is always the counter example or unforseen case that will not work under the present formulation. For example, the cross pattern in Figure 2.4 illustrates this concept. It is not clear if the cross pattern is made up of two overlapping ellipses, or is a single pattern, a cross. Popular types of patterns include: linear formations, clouds or elliptical distributions, shells, and general quadratic surfaces. Data sets that exhibit a high degree of pattern space separation and small inner pattern scatter are conventionally considered "well behaved." This is not a requirement, but a nice property to have. Data sets that are not well behaved might be due to noisy features or other sources acting on the sampling process or algorithms. Visual inspection can be used to analyze patterns, but is not a reliable process because in many cases the dimensionality is too high to be visualized meaningfully with modern techniques. This increases the difficulty in assessing and defining a clear and unambiguous definition for a pattern.

Robonaut Sensory-Motor Feature Space

The Robonaut is an advanced humanoid robot with forty-seven degrees-of-freedom. The combination of the arm and hand data comprises a possible feature space that has over 100 dimensions. The following list enumerates the arm and hand sources available.

Arm Data: (62 data streams)

(1) 1-16: 4x4 transformation matrix from chest to hand, actual

(2) 17: orbit angle, actual

(3) 18-33: 4x4 transformation matrix from chest to hand, command

(4) 34-36: orbit angle, command

(5) 37-39: force at the wrist (x, y, z)

(6) 40-42: moment at the wrist (x, y, z)

(7) 43-45: force at the shoulder (x, y, z)

(8) 46-48: moment at the shoulder (x, y, z)

(9) 49-55: joint positions (shoulder roll, pitch, yaw; elbow; wrist roll, pitch, yaw)

(10) 56-62: joint torques, corresponding to positions.

Hand Data: (48 data streams)

(1) 1-5: force at each finger (thumb to pinkie)

(2) 6-17: joint positions of the fingers: (thumb yaw, pitch, curl; index yaw, pitch, curl; middle yaw, pitch, curl; palm cup; ring curl, pinkie curl)

(3) 18-29: joint torques of the fingers, corresponding to above.

(4) 30-48: tactile sensors: (thumb front base, front tip, side base, side tip;

(5) index base, middle, tip; middle base, middle, tip; palm ring, middle, index; ring base, middle, tip; pinkie base, middle, tip)

2.1.5 Qualitative States and Qualitative State Sequences

We hypothesize that the number of clustering prototypes or HMM hidden states has a direct one to one relation to the notion of a qualitative state (QS). In the case of a mixture density HMM, multiple probability densities map to one QS. We assume that QSs, which symbolize skill primitives, are an equilivence class in the sensory-motor feature space. However, this statement says nothing about the kind of pattern, or if patterns are allowed to overlap. QSs can be used to construct a finite state automaton, which we use to represent a skill. The sequence of QSs that a robot encounters while performing a task can be visually demonstrated in a QS diagram. Figure 2.5 demonstrates a sample QS diagram and its conversion to a QS sequence.

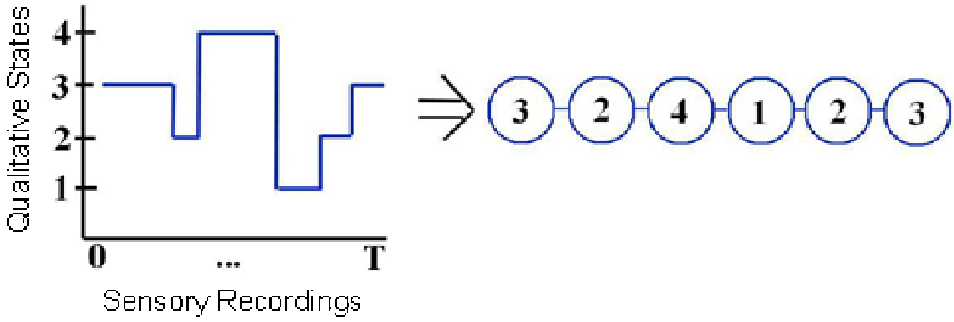

Figure 2.5: Conversion from a QS diagram to a QS sequence. Vertical axis denotes cluster ID's, which is an arbitrary assignment, and the horizontal axis represents the sampling time steps.

Identification of patterns in a robot's sensory feature space leads to a natural description of a task segment derived from the ground up. These descriptions are acquired and shaped by experience, instead of having processes in place and using experience to refine the structure. Our application of clustering involves learning a set of prototypes and the associated membership matrix. QS sequences are identified by hardening and making the fuzzy membership matrix crisp. The maximum membership value of every sample is then singled out and used to represent the cluster that point most likely belongs to. There are some circumstances where the maximum membership value is indeterminate, which means that the value is extremely low or too close to another membership value. If this is the case,

then we just accept and assume the last known state. This filtering technique was used in order to clean up the QS sequences in uncertain regions that have a lot of activity and jumping between states.

Around 700 sensory samples have been recorded and labeled in Figure 2.6 according to their highest cluster membership value. This graph represents one repetition of a reach and grasp task performed on the NASA Robonaut. The vertical axis has a unit for each QS. The horizontal axis denotes the samples collected over time. The temporal (left to right) analysis of this piecewise defined graph demonstrates the transitions over time from QS to QS. This QS sequence represents the skill being acquired.

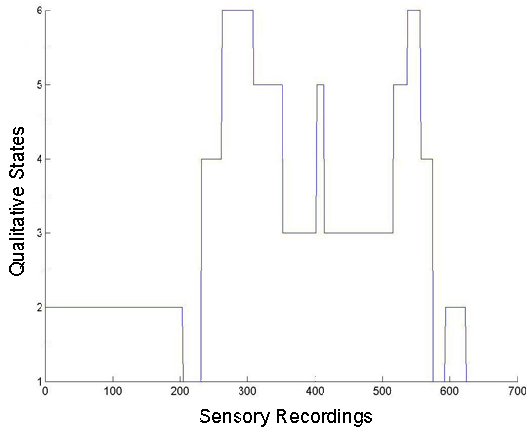

Figure 2.6: Example QS sequence with 6 states found by the RCA algorithm. The x-axis represents the sampled time steps and y-axis units are the arbitrary ID numbers.

2.2 Related Work

2.2.1 Single Ended Contact Formations

The notion of a contact formation (CF) was originally proposed by Desai and Volz to represent a qualitative discrete state that describes how two or more objects are in contact[37]. Skubic and Volz have extended this notion to Single Ended Contact Formations (SECF)[25][28][29][26][27][20], which are a one-sided description of how an object makes contact with its environment. This SECF research was important in initially defining and building the hypotheses from which this thesis is built.

SECFs are computed as patterns in a normalized force and moment sensor feature

space. The significance of using normalized signals corresponds to the direction of the vector in a geometric context. Skubic and Voltz used two different techniques to learn SECFs, which were not reliant on geometric or positional data. The first approach involved the application of fuzzy set theory[26], while the other relied on a neural network[29]. The first approach involved the design of a fuzzy logic rule based classifier. The goal was to learn a set of fuzzy logic rules, one rule for each class. The antecedent part of each rule represents a conjunction of fuzzy membership functions for each feature, which are automatically generated through supervised learning techniques. The consequent part of a logic rule represents the classification into a particular desired class. Classification is therefore the product of fuzzy analytical reasoning.

The neural network approach is an attempt at designing a classifier that learns mappings from normalized force and moment signals to SECF confidence values. There is a network input for each of the six sensory features and a network output node for each desired SECF class. The number of hidden nodes in the network depends on the geometric shape of an object. There is a hidden node for each vertex that can make contact with the environment. The popular back propagation algorithm was used to represent and train the network. The resulting network is capable of identifying the mapping and classification of a new sample point into an output SECF class.

A significant part of this work was the skill representation as a sequence of QS, where each QS is a SECF, identified from sensory signals. This idea involves the notion that SECFs are clusters in the sensor feature space. These are the guiding assumptions for this thesis.

2.2.2 Hidden Markov Models and Skill Learning

One of our approaches to skill acquisition involves the application of HMMs. The following bodies of work have also approached the topic of skill learning in a similar fashion. The following methods are primarily interested in how to model and transfer skills from a human to an agent. These bodies of research do not exactly match what we have done or set out to investigate, but are similar enough through the application of statistical and temporal task modeling with HMMs.

Hovland, Sikka, and McCarragher [8] set out to discover a discrete way to model assembly tasks using HMMs. The resulting states are used in a discrete event controller (DEC) to model skills at a task level. Their training data represents a set of human recordings of an assembly task, each task performed several times. They recorded and studied the position and velocity readings of the human demonstration. This observation sequence was used to train a set of HMM's, one for each task. The transition probability matrix was used to determine the probability of corresponding future events occurring. They finally compute and use the most likely state sequence through the Viterbi algorithm, which comes from the trained model coupled with their monitoring process to decide what next state to transition the system to.

This next body of work comes extremely close to our own in the category of tasks, setup, and intent. Yan, Xu, and Chen [14] use HMMs for skill learning in a teleoperated setting. What distinguishes these authors and their work is the rationale and intent behind the application of an HMM to the task of skill learning. These authors state that their intent was to use an HMM in order to model measurable actions and immeasurable mental states. This is analogous to attempting to learn and model intention through observations and physical processes. If this inverse mapping exists, it is still to be ultimately demonstrated. This work had been applied in the teleoperation control of a space station robot system (Self-Mobile Space Manipulator, SM^2), developed at the Robotics Institute of Carnegie Mellon University. These authors transferred human skills to a robot through analyzing and modeling teleoperated recordings of maintenance and inspection related tasks. They have collected the trajectories of the position and velocity in both joint space and Cartesian space. Using HMMs, the system learned (1) position trajectory in Cartesian space, (2) position trajectory bearing in joint space, (3) and velocity trajectory learning in Cartesian space.

2.2.3 Neural Networks and Reinforcement Learning of Skills

This topic has been included to give a quick exposure to other views and methods for performing skill learning. In general, skill learning refers to the process of acquiring and

modeling skills, but says nothing in particular about how or in what framework this has to be done. A large body of research attempts to leverage the human biological and neural system in an attempt to generate a powerful and general learning framework. Artificial neural networks (ANN) are powerful machines that can be used for function approximation. These structures are distributed information systems which are capable of learning and generalizing from experience even when the data is imprecise and noisy[45].

In the context of learning robot skills, there are an extremely large number of publications which leverage ANNs for control and classification strategies. For example, Skubic has used the popular back propagation algorithm for training a classifier that learns SECF mappings [29]. Pfeifer has also applied a variety of ANN techniques in his initial paper on SMC[33] which include: back propagation, self organizing feature maps (SOFM), and even auto associative memory models with bi-directional active forgetting. One of Pfeifer's models uses neural networks and the Extended Braitenberg Architecture to learn mappings from sensory signals to behaviors or motor commands for the control of mobile robots.

Another field in the area of neural modeling is hybrid and connectionist systems. Connectionism generally refers to methods that leverage ideas from the fields of cognitive psychology, cognitive science, engineering, philosophy, and other disciplines that help in modeling mental phenomenon typically through ANNs. Hybrid systems do not limit themselves to only neural networks, but also employ a variety of symbolic and explicit knowledge throughout the system. Procedural or implicit representations are a form of knowledge that are not locally centralized, but rather spread out through a system of smaller and more general processing units. Symbolic or declarative knowledge represents a single concept or entity as an individual discrete quantity. Combinations of these procedural and declarative knowledge are what are modeled and used for reasoning and acting in connectionist and hybrid systems.

One cognitive and connectionist system, developed by Sun, is for learning and acquiring skills from the ground up (procedural) is CONSYDERR. CONSYDERR stands for CONnectionist System with Dual representation for Evidential Robust Reasoning[43]. The procedural level deals with acquiring neural representations of skills through reinforcement

learning. Low level skill acquisition happens in a reinforcement environment in which learning is performed through the Q-learning algorithm[51][44]. Q-learning can be realized and trained in a neural network so skills are procedurally captured. The input part of the network represents the incoming stimulus, while the hidden layers deal with computing the next state and output action. Information is then extracted from these low level skills, through knowledge extraction algorithms, in order to construct and update a higher level and symbolic set of nodes and rules for different domains. The system is simultaneously attempting to learn procedural and declarative representations of the domain and skills. This system is intrinsically related in terms of links from the procedural and declarative components. This allows the processes of the system to be affected through the procedural and declarative knowledge. This co-interaction between the declarative and procedural components is to allow for more common-sense reasoning about the knowledge and skills that have been acquired.

2.2.4 Episode Extraction

Another approach to skill acquisition involves the sequential segmentation of episodes and events from a time series of sensor and motor recordings. Peters has extracted episodes that correspond to SMC episodes and events for the same reach and grasp task and NASA Robonaut[40] data as used in this thesis. For each object location, the task is performed multiple times. Peters first sequentially analyzes the sensory signals in a single repetition to discover where the episode boundaries occur. Episode extraction is performed for each repetition, and they then combine the segmented episodes to compute an average episode length. This average episode length is used to resample and time normalize an episode to a fixed uniform number of signals. The following two sections describe the specifics behind their techniques.

Segmentation

The guiding principle behind this section is how to segment a time series of vectors into episodes. The following notation is consistent with the repetition and location terms defined

in the section on the NASA Robonaut. It is assumed that each repetition is composed of the same number of episodes. Let K denote the number of episodes in the task. Define $E_{(k,i,j)}$ to be the k^{th} episode in the j^{th} repetition ($1 \leq j \leq R$) from the i^{th} location ($1 \leq i \leq L$). Let $|E_{(k,i,j)}|$ denote the length of the k^{th} episode for repetition j of location i. Since no task can ever be performed the exact same twice, the length of the k^{th} episode in the j^{th} repetition for location b does not have to match the length of the k^{th} episode for the j^{th} repetition at location s ($|E_{(k,j,b)}| \neq |E_{(k,j,s)}|$).

Peters segmented the vector time series according to two different techniques. The first technique generated the segmentation through a visual inspection, while the other used Mataric's technique based on extrema found when analyzing the mean squared magnitude of the joint velocities. They report that this second procedure has generated better segmentations than their earlier manual inspections. They started by looking for peaks in the elbow joint angle for episode boundary detection. However, their initial method missed events that occurred at low velocity moments. When they incorporated the joint position maxima of the elbow, they report that they capture all the events that occur at the moments of low velocity.

Time Normalization

Time normalization is the procedure of constructing new time series descriptions ($E_{tn(p,i,j)}$), where the number of samples in episode p is the same across different repetitions at different locations, i.e. $|E_{tn(p,l,m)}| = |E_{tn(p,i,j)}|$ ($\forall(i,l)$ and $\forall(j,m)$). The t^{th} sample in episode p for the time normalized data is $E^t_{tn(p,i,j)}$. Time normalization is performed by resampling each episode to a fixed number of samples. This step occurs after segmentation, which is the identification of the episode boundary points. The new length of each episode, $|E_{tn(p,i,j)}|$, is computed as the average length for that episode over all repetitions at different locations. The following shows how to compute the number of samples in the p^{th} episode for the j^{th} repetition at location i.

$$|E_{tn(p,i,j)}| = \lfloor \tfrac{1}{R*L} * \textstyle\sum_{l=1}^{L} \sum_{r=1}^{R} (|E_{(p,l,r)}|) \rfloor$$

Chapter 3

Methodologies

3.1 Pattern Recognition

3.1.1 Introduction to Pattern Recognition

Pattern recognition is performed naturally and automatically in every aspect of our daily lives. This natural process has evaded and confused those that look to artificially and computationally replicate these processes. This discipline has been used in the design of classifiers, used to discover underlying model structure, create index structures into a data base or the World Wide Web, analyze and predict sequences of protein structures, and build discrete and stochastic temporal finite automaton. Representations and tools vary across each domain and include: statistical modeling, optimization and calculus techniques, graph theory, neural networks or learning machines, and even logic languages or expert systems. A large misconception about pattern recognition is that it should solely be used for the task of classification. Pattern recognition is more than classification, and can be subdivided according to the learning model: supervised, unsupervised, or reinforcement learning. Bayesian classifiers operate in a supervised setting, while clustering algorithms look for patterns in an unsupervised and unlabeled environment.

There is no one definition for a pattern due to its application specific and subjective nature. A pattern can exist in, but is not restricted to, the spatial, frequency, or temporal domains. A pattern can be a statistical distribution, geometric shape or object in an image, time series trend, sub-sequence in a string, and even a set of associations or logic rules. In this work, we define and work with patterns that are expressed as clusters in feature space[41][46]. Features, or sources of information, are combined to construct a cross product

space where clusters are expressed. Figure 3.1 illustrates a pattern recognition framework that we leverage. Pattern recognition is viewed as a sequence of (1) the phenomena being performed (2) our observation of the phenomena (3) generation of features to describe the phenomena (4) selection of a subset of these features in order to maximize classification and discrimination power (5) application of the PR algorithm or classifier (6) optional post processing (7) and final evaluation of the systems results.

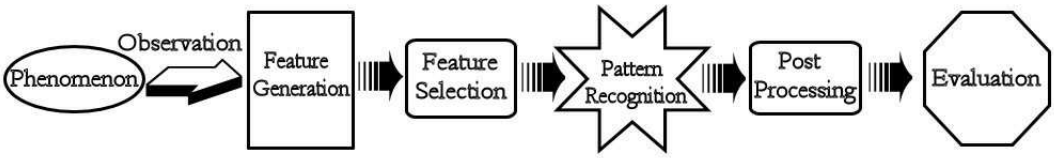

Figure 3.1: Conventional pattern recognition process

3.1.2 Introduction to Clustering

Human intelligence is a marvelous phenomenon in how hard tasks can be performed naturally and with so little effort. Clustering is one of these tasks which appears simple to a human, but extremely difficult to a machine, statistician or mathematician. Theodoridis and Koutroumbas give a loose definition of clustering as, "... to reveal the organization of patterns into sensible clusters (groups), which will allow us to discover similarities and differences among patterns and to derive useful conclusions about them."[46]. A common misconception is that clustering can be used for classification. Clustering is not a classification procedure, but rather a method to reveal structural information in a data set.

Clustering procedures can be grouped into the following categories: sequential algorithms, hierarchical algorithms (agglomerative and divisive) and cost function minimization procedures[46][41]. Sequential clustering algorithms operate by presenting the data one sample at a time. This makes the algorithms highly sensitive to both initialization of cluster parameters and also the presentation order of the data. Hierarchical clustering algorithms typically work in a single or complete linkage fashion to continuously subdivide or merge clusters. Cost function minimization techniques express some notion of error or optimality through a function which needs minimization. This research involves the application of cost

function minimization techniques in combination with dissimilarity measures.

In order to understand the following few sections, several definitions are necessary. Each sample denotes the cross product of K unique features. Assume that we are clustering the following data set, $X = [\vec{x_1} \dots \vec{x_N}]$. The goal of clustering is to partition the data set into M disjoint groupings. The j^{th} cluster will be denoted by C_j. Clusters should follow the next 3 conditions.

(1) $C_j \neq \phi, j = 1, \dots, m$ (no empty clusters)

(2) $\cup_{i=1}^{m} C_i = X$ (union of all clusters is the data space)

(3) $C_i \cap C_j = \phi, i \neq j, j = 1, \dots, m$ (intersections of clusters are empty)

A cluster is represented by a set of parameters, which is typically a prototype vector ($\theta_i = \vec{p_i}$). Prototype vectors are single points in the feature space that, in combination with the proximity metric, separate the feature space into clusters. Another important clustering quantity is the membership matrix, which reflects the grade of membership of an element in the cluster. In the Hard K-Means (HKM) algorithm, the membership value of the i^{th} data sample with the k^{th} cluster, denoted by $u_{(i,k)}$, is 1 if that sample is the most similar to cluster k. Otherwise, the membership value is 0. In the FCM algorithm, the membership values of the i^{th} sample to all clusters have to sum to 1, but the $u_{(i,k)}$ values now are assumed to come from the interval $[0, 1]$. This means that every data sample can actually belong to each cluster to a different "degree".

Proximity Metrics

Clustering is performed by organizing entities into groups that reveal similarity or dissimilarity. Proximity metrics are the means by which similarity and dissimilarity are conventionally measured. Let us first define a few quantities that appear in the following metrics. Each dissimilarity metric, $d^{metric}(\vec{x}, \vec{y})$, is a function of two vectors $\vec{x}$ and $\vec{y}$ that come from X. The covariance matrix is $\sum$ and $\sum^{-1}$ is the inverse of the covariance matrix. The determinant of the covariance matrix is $|\sum|$.

The number of proximity metrics that can be devised is infinite. There is not one metric that works for all cases. Below are a few common dissimilarity metrics that frequently appear, of which we use the last three.

Weighted l_p metric (WLP): $d_p^{WLP}(\vec{x}, \vec{y}) = (\sum_{i=1}^{d}(w_i|x_i - y_i|^p))^{\frac{1}{p}}$

Where w_i and p are scalar parameters

and d is the feature space dimensionality

Euclidean Distance Function (ED): $d^{ED}(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$

Mahalanobis Distance Function (MD): $d^{MD}(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^T(\sum^{-1})(\vec{x} - \vec{y})$

Gustafson-Kessel (GKD), or Scaled MD (SMD): $d^{GKD}(\vec{x}, \vec{y}) = \frac{1}{|\sum|^{1/d}} * d^{MD}(\vec{x}, \vec{y})$

3.1.3 Fuzzy C-Means

One large twentieth century paradigm shift, led by Lofti A. Zadeh, involves the representation of a particular kind of uncertainty. Zadeh introduced the notion of fuzzy sets as a method of relaxing the classical crisp set theory boundaries[17]. This allows membership values to come in degrees. Fuzzy set theory deals with a different kind of uncertainty outside of randomness and frequency. This method of allowing a sample to belong to different sets in varying degrees helps in the development of clustering algorithms. This assumption is made in conjunction with the popular belief that one should never over commit or do something which will later have to be undone, the Principle of Least Commitment [6]. This now allows for the search of the most optimal partition while simultaneously never over committing each point's membership value to a cluster. The Fuzzy C-Means (FCM) has enjoyed a great degree

of success within the category of cost minimization based clustering. The FCM algorithm was proposed by Bezdek in an attempt to relax the way that membership values of elements within clusters are considered[11]. This has the effect of allowing a sample to belong to each individual cluster according to some "degree" (membership value). The FCM seeks a set of prototypes ($P = \{\theta_1 \dots \theta_m\}$) which result in an assignment of membership values (U, where $u_{(i,j)}$ represents the membership of the i^{th} vector to the j^{th} cluster) that can be determined iteratively through minimizing the following cost function.

$$J(\theta, U) = \sum_{i=1}^{n} \sum_{j=1}^{m} (u_{(i,j)}^{q} * d(\vec{x_i}, \theta_j))$$

$$\text{Where } \sum_{j=1}^{m} u_{(i,j)} = 1$$

$$q \text{ is the Fuzzy Factor, typically } q \approx 1.5$$

$$u_{(i,j)} \in [0, 1],\ \vec{x_i} \in X \text{ and } |X| = n$$

The minimization of the above cost function, with respect to the membership sum constraint, leads to convenient iterative update formulas. This makes the basic algorithm fairly simple in design and implementation. A very popular example of the need for fuzzifying the membership grades is illustrated through the classical butterfly set, illustrated in figure 3.2. In particular, look at the points in the middle of the feature space. What cluster do these points belong to? How many clusters should exist? At what point does a sample stop belonging to one cluster and start belonging to another? It is this type of uncertainty which underlies many problems within the pattern recognition field.

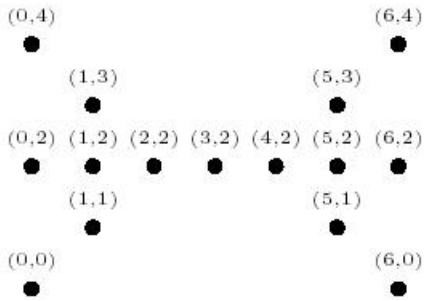

Figure 3.2: Butterfly Data Set

3.1.4 Possibilistic C-Means

The FCM has shown to be a good general clustering algorithm for many settings. The relaxation regarding the assessment of membership grades can lead to great improvements over the HKM, but still leaves many undesirable and unavoidable inherent problems. One very notable problem involves the constraint that the sum of membership values for a point within all clusters must equal 1. This constraint in effect makes the memberships reflect a degree of sharing of the point between clusters. These points can be part of the pattern or noise acting on the sampling process. If the goal is to search for a prototype placement relatively centralized within the cluster, these undesirable FCM clustering artifacts can have the effect of pulling the prototypes out and away from an expected central location.

The Possibilistic C-Means (PCM)[38], developed by Krishnapuram and Keller, can be used to address a few of these noted issues. The PCM algorithm looks for dense regions in the feature space. The first step in deriving the PCM equation requires the removal of the constraint that membership values must sum to 1. However, if this constraint is removed, and is the only change, then the optimization process will simply seek a trivial solution where membership values are all assigned to zero. An additional cost term has been added in order to bypass this trivial solution. This now allows membership values to reflect the possibility or degree of typicality that a point belongs to a cluster[38]. The PCM cost function is defined as:

$$J(\theta, U) = \sum_{i=1}^{n} \sum_{j=1}^{m} (u_{(i,j)}^{q} * d(\vec{x_i}, \theta_j)) + \sum_{j=1}^{M} \alpha_j \sum_{i=1}^{N} (1 - u_{(i,j)})^q$$

where α_j are the "bandwidth" parameters

and $\sum_{j=1}^{m} u_{(i,j)}$ does not have to equal 1

This algorithm does, however, come at the cost of approximating a few more parameters, namely the α_i values, which can be a trick. The original publication on the PCM[38], and the follow up correspondence paper[38], discuss methods to estimate these values. The most traditional way is to first run the FCM in order to get a staring place to estimate these α "bandwidth" parameters. This is the technique that has been used in this work.

Another potential problem is coincidental clustering, i.e., prototypes can tend to be

drawn to the same regions. There is no property that makes prototypes move towards different areas. This artifact is not necessarily a bad thing. An over specification of the number of clusters can be helpful and provide additional and useful information. Statistical testing or other methods can be used to check for prototypes within the same cluster if desired. Improper initialization of parameters can lead to this problem.

In order to bypass the coincidental clustering delima, the PCM was slightly modified. The algorithm is ran multiple times and allowed to converge each time. Each time it converges, the densest cluster (determined by cluster cardinality checks) is extracted and its members are removed from the present working data set. This is done by finding the largest GOM value for each point, making sure its GOM is distingishibly the highest for the cluster under investigation, i.e. not possibly belonging to more than one cluster, and using an optional cut off threshold value to accept these points. The algorithm is then rerun and allowed to now find a new best cluster from the remaining data set. This has empirically been found to work well and act in a reliable fashion.

3.1.5 Robust Competitive Agglomerate

The clustering algorithms presented so far (FCM, PCM and HKM) assume that the system knows the number of clusters to search for ahead of time. This can be a large assumption and can make determining the number of clusters an entirely different problem from the clustering algorithm. A group of methods exist to address this issue, and one of the most popular approaches cluster validity. Cluster validity can be used to find the number of clusters, although there are potential problems. One problem is the cost associated in rerunning the clustering algorithms multiple times to generate the cluster validity index values. Another problem arises in the interpretation of the index values. In order to sidetrack these issues, this project looked for alternative methods to learn the number of clusters during the execution of the algorithm. Performing the learning of the model and the parameter value selection simultaneously is a difficult and sensitive problem.

The Robust Competitive Agglomerate (RCA) algorithm is another cost minimization algorithm that is derived by modifying the conventional FCM cost equation[9]. It is not the

cost function that drops clusters, but rather the algorithm can be used to indicate when and where this might be done. The RCA cost equation is:

$$J(\theta, U) = \sum_{i=1}^{n} \sum_{j=1}^{m} (u^q * d(\vec{x_i}, \theta_j)) - \alpha \sum_{j=1}^{M} \sum_{i=1}^{N} (u_{(i,j)})^2$$

where α is an important competition term

and $\sum_{j=1}^{m} u_{(i,j)}$ has to sum to 1

The key to understanding what this has done to the FCM equation, the additional RCA second cost function term, is inside the newly derived update equations. These update equations do not directly make it possible to determine the number of clusters, but prove useful to help in such a procedure. The update equations for the prototypes in the RCA are:

$$u_{(s,t)} = u_{(s,t)}^{FCM} + u_{(s,t)}^{Bias}$$

$$\text{where } u_{(s,t)}^{FCM} = \frac{\frac{1}{d^2(\vec{x_t}, \theta_s)}}{\sum_{k=1}^{C} \frac{1}{d^2(\vec{x_t}, \theta_k)}}$$

$$\text{and } u_{(s,t)}^{Bias} = \frac{\alpha}{d^2(\vec{x_t}, \beta_s)}(N_s - \Gamma_t)$$

$$N_s = \sum_{j=1}^{N} (u_{(s,j)})$$

$$\Gamma_t = \frac{\sum_{k=1}^{C} \frac{1}{d^2(\vec{x_t}, \beta_k)} N_k}{\sum_{k=1}^{C} \frac{1}{d^2(\vec{x_t}, \beta_k)}}$$

N_s denotes the cardinality of cluster s

Γ_t is the weighted average of cluster cardinalities

The resulting update equation acts in favor of higher than average cardinality clusters. The update equation is the combination of the original FCM update equation, plus a second additional term. The second term has the dual effect of acting negatively and reducing the membership values in weakly expressed clusters (lower than average cardinality). If a cluster is not strongly expressed, then points from that cluster will tend to compete with other surrounding clusters that have a higher than average cardinality. This can be used to isolate weak clusters that will then be dropped during the algorithms execution. The simple approach to dropping clusters in this setting includes picking a threshold value to test against. Discovering these problem-specific values and hard coding them into the algorithm restricts the algorithm and makes it brittle. In this project, the RCA still requires some empirical testing and selection a parameter, but we try to use a more general and relative check based

on the present set of cluster cardinalities. The user specifies a minimal acceptable relative (0% to 100%) cluster cardinality value. This is used to determine when a cluster's cardinality value goes below an acceptable relative level at any time step.

I have also introduced another algorithm parameter that allows the program to go into "harsh" or "soft" mode. In the "harsh" mode a cluster is dropped whenever its relative cluster cardinality goes below the user defined relative threshold. In the "soft" mode a weak cluster is allowed a few iterations to improve before it is simply dropped. The soft mode was designed to minimize the effects of dropping a cluster and allowing the remaining parameters to have a few extra iterations to possibly move into a more representative location. However, alteration of the original equations and philosophy void all the algorithm guarantees.

The RCA also requires an estimate for the α value during the algorithm. When α is proportionally high, relative to the first term, there is competition. When the α value is low in the second term, it in essence drives the competition out of the cost function and acts like the FCM. This work has used the proposed alpha equation within the RCA literature[9]. The proposed method is to use an exponential decay learning rate in order to control the learning through the number of iterations which the algorithm has taken. This rate begins low, increases to a point of maximum competition, and then is allowed to decrease in order to allow the algorithm to stabilize and converge. As the rate decreases and attempts to converge, the RCA equation becomes the FCM. I used the α equation that was put forth in the original paper[9], where $\eta_0 = 1$, $k_0 = 5$, and $t = 10$.

$$\alpha \propto \eta(k) \frac{\sum_{i=1}^{C} \sum_{j=1}^{N} (u_{(i,j)})^2 * d^2(\vec{x_j}, \beta_i)}{\sum_{i=1}^{C} [\sum_{j=1}^{N} u_{(i,j)}]^2}$$
$$\text{where } \eta(k) = \eta_0 e^{-k/t}$$

A few different techniques in estimating the initial parameters for the prototypes were investigated. Determining the initialization parameters is a very classical problem that has the effect of inducing a different set of results depending on the optimization terrain (extrema). Random initialization of the beginning parameter values was avoided. Random initialization does nothing to utilize information in the data set that can help to reduce the number of overall algorithm iterations and improve on convergence locations. Experimentation started with using a global mean and a global covariance. All prototypes, which are

computed after the first global mean prototype, come as a random combination of diagonal and off-diagonal values from the covariance matrix (appropriately scaled), applied to the global mean. It was noted, through experimentation, that the parameter initialization was highly important, so another method of initialization was sought after. The RCA requires that you over specify the number of prototypes to start. One idea to initialize these parameters, if computational time is not as much of an issue, is to use a self organizing feature map (SOFM). If the problem is expected to have around 5 clusters, it is proposed[9] that starting with somewhere around 3 to 4 times that number is a good empirically discovered starting place. The RCA initialization prototypes are generated by a SOFM of size 5x5 (25 elements). In order to get around deficiencies inherent within any estimation technique, extra random prototypes are generated.

3.1.6 Cluster Validity

The number of clusters in the Robonaut's sensory-motor feature space is not known ahead of time. It is not even known if any meaningful clusters exist. A method is therefore needed to discover and validate how many clusters exist in each feature space for every particular task and repetition. This can be a tedious manual process and presently requires a great degree of human intervention and subjectivity. This naturally led the project towards cluster validity measures. Cluster validity measures take the resulting cluster parameters and compute a single numerical value. If the clustering algorithm is executed with a different number of clusters each time, then the resulting cluster validity values can be used in an attempt to learn the clustering parameters (typically the number of clusters). No method to automatically analyze cluster validity results arose from this work. Interpreting cluster validity measures can be a difficult and tricky procedure. The analysis process can be easy as long as the data set is well-behaved and the cluster validity measure is appropriately selected. However, non-synthetic data sets do not always exhibit such desirable properties. In our particular case, we have identified a few possible numbers of clusters, which is explained in more detail in the results section.

It was difficult to determine exactly which cluster validity index measure to apply. This

is partially because our data set comes from sensor and motor equipment that can generate erroneous descriptions at points such as contact. We are also plagued with the realization that, as the robot transitions between clusters, the movement from one cluster to another is a continuous progression with respect to sensor and motor recordings. This creates "trails" of data samples from one cluster to the other. Beyond increasing the difficulty of the pattern recognition algorithm, this also affects a good degree of the popular cluster validity measures which compute cluster separation and intra-cluster scatter values. Cluster separation had to be computed as the distance of two prototypes, not some variation of minimum linkage, i.e. the closest two points in the two clusters and their reflective distance. The intra-cluster separation values are also difficult to compute because a conventional approach of discovering the distance of the maximally distant pair of points is not necessarily a good approach here. The Davies-Bouldin's (DB) measure uses an inner cluster separation value based on the average of all distances from the prototype. A few different cluster validity measures were investigated, but the DB measure was the most consistent in its findings. The Davies-Bouldin measure can be computed as:

$$DB = \frac{1}{c}\sum_{i=1}^{c}(max_{(j\in c, j\neq i)}\frac{S_i+S_j}{d_{i,j}})$$

where c is the number of clusters

A_i is the i^{th} cluster

$|A_i|$ is the cardinality of the i^{th} cluster

$\vec{x}$ is some sample vector

$\vec{v_i}$ is the i^{th} prototype

$$S_i = (\frac{1}{|A_i|}\sum_{(\vec{x}\in A_i)}||\vec{x} - \vec{v_i}||^2)$$

$$d_{(i,j)} = ||\vec{v_i} - \vec{v_j}||$$

3.1.7 Hidden Markov Models

Recognizing patterns that fold out over time require a procedure that has the ability to model the temporal aspects of the process, such as previous decisions or actions that affect the present state and action. Hidden Markov Models (HMM) denote a way to represent and recognize temporal patterns by learning from one or more sequences of observations, such as

a time series of robot sensor and motor signal recordings. A classical example of what HMMs have been used for is the task of speech recognition. Rabiner has applied the Baum-Welch algorithm to learn the parameters for an HMM for the task of recognizing speech patterns as a sequence of phonemes [19]. The Baum-Welch algorithm is used to iteratively re-estimate the parameters for a HMM from a training set of observation sequences. HMMs are stochastic finite state automata, where the states are assumed to be not directly observable, but can be discovered through observations that the states emit. The main problem involves how to take one or more observation sequences that the process emitted and learn a set of HMM parameters that maximize the likelihood that these observations were observed from the model.

The features that the NASA Robonaut generates are the product of a task performed over a finite interval of time. Clustering does little to nothing to incorporate the temporal aspects involved in the skill. Temporal problems have to deal with the reality that their patterns do not only exist within some static and time independent feature space, but they might also have to incorporate previous time information directly into the structure that they seek to learn. The Baum-Welch algorithm provides a method to learn patterns that incorporate a pre-designated number of previous time steps (the model order) into each decision making step. Many HMMs are generally just of the first-order, which means that they consider only one previous time step at each moment. We have applied a first-order continuous observation HMM, shown in Figure 3.3, which can be formally represented through the following notation[22][13]:

- N: The number of pre-designated hidden states. It is assumed that these states are not directly observable, but can be discovered through observations which the system emits
- T: The length of the observation sequence
- S: The hidden states $S = \{S_1, S_2, \ldots, S_N\}$
- O: The observation sequence $O = \{O_1, O_2, \ldots, O_T\}$
- K: This represents the number of emission distributions per hidden state. Many models only assume 1 distribution per hidden state. A more powerful approach is to increase this number and capture models which are composed of multiple distributions per hidden state
- Q: The hidden state sequence
- q_t: The hidden state at time t
- A: The state transition probability distribution matrix, where $\{a_{(i,j)}\}$ represents $a_{(i,j)} = P(q_{t+1} = S_j | q_t = S_i)$
- θ_i: probability density functions parameters for state S_i
- π: $\{\pi_i\}$ represents $P(q_1 = S_i)$, i.e. the initial state distributions

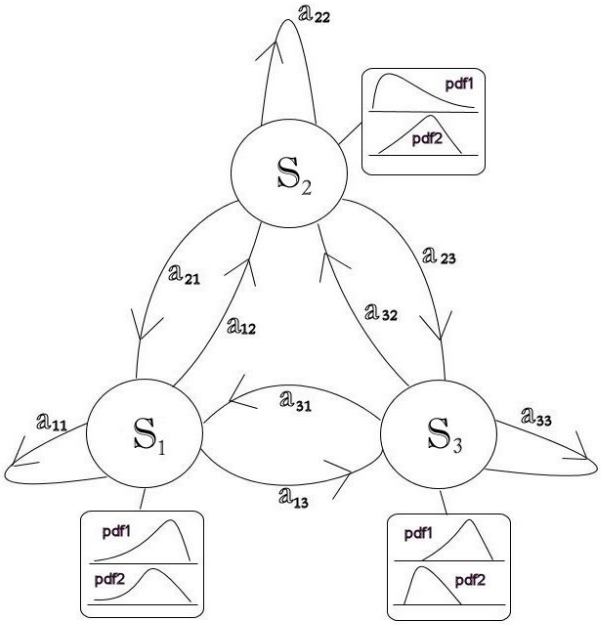

Figure 3.3: Illustration of a continuous observation HMM (COHMM)

These parameters define a COHMM model as the following three touple, $\lambda =< A, \theta, \pi >$. The three classical problems associated with an HMM are described below.

(1) HMM Classification Problem

The classification problem can be addressed by discovering the most likely, if one exists, model (λ^*) that is believed to have generated some observation sequence O. This is done by finding the model that maximizes the following probability equation $P(O|\lambda_i)$.

$$P(O|\lambda) = \sum_{\forall Q}(P(O,Q|\lambda) = \sum_{\forall Q}((\pi_{q1}b_{q1}(O_1))(\prod_{t=2}^{T}(a_{(q_t,q_{t-1})}b_{q_t}(O_t))))$$

The simplest way to solve this equation is to use brute force and consider every possible path. This method does not scale well, taking TN^T calculations. We have used the conventional forward-backward algorithm to make this problem computable[13][22]. This algorithm uses recursive definitions of two useful quantities, the forward and backward values, in order to eliminate many unnecessary calculations and ensure a greatly reduced time of TN^2.

(2) Most Optimal State Sequence

There are a number of possible ways to identify and compute the most optimal state sequence[22]. The problem specifically depends on how one defines the most likely sequence of states. One method is to compute and identify the individually most likely states. However, we find it of particular interest to compute and consider the most likely overall, not individual, sequence of states. We have used the popular Viterbi algorithm[13][22] to approach the problem in this fashion. This algorithm computes the overall most likely sequence of states according to $P(O,Q|\lambda)$. This is done in a two stage process. The first stage requires a forward propagation and calculation of a few important quantities, which is followed by a backward scan and selection of the most likely state sequence.

(3) HMM Training

The third problem is typically regarded as the most difficult. In order to approximate the parameters for an HMM, a more powerful technique than the conventional Maximum-likelihood Estimator (MLE) is needed. We have used the Baum-Welch algorithm, a generalization of the Expectation Maximization (EM) procedure, for learning and training an ergodic first-order continuous observation Hidden Markov Model (COHMM). The next section provides a short overview of the Baum-Welch procedure.

Baum-Welch Procedure

The Baum Welch algorithm is used to iteratively re-estimate the HMMs parameters from a set of observation sequences. The Baum Welch procedure is simply an instance of the EM algorithm in which the Q update formula represents the expected value of the new model (λ^*) according to the present model (λ) and the data set (X).

$$Q(\lambda^*, \lambda) = E[\lambda^* | X, \lambda]$$

The derivation of the Baum-Welch procedure is left out of this thesis primarly for length reasons, but can be found in Rabiners [19] or Bilmes [13] papers. The Baum-Welch procedure leads to a set of iterative update equations for the hidden states and transition model. The following assumes that a variable, $\gamma_i(t) = p(Q_t = i|O, \lambda)$, has been computed that represents the probability of being in state i at time t. The term, $\phi_{(i,j)}(t) = p(Q_t = i, Q_{t+1} = j|O, \lambda)$, is also computed and represents the probability that the model is in state i at time t, and is in state j at time $(t + 1)$. We can now define and express the update equations for the transition parameters as:

$$\pi_i = \gamma_i(1)$$
$$a_{(i,j)} = \frac{\sum_{t=1}^{T-1} \phi_{(i,j)}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

From the above two basic parameters, γ and ϕ, the continuous observation density parameters can be estimated. In the case of a density mixture and normal distribution, we seek to compute the mixture terms $c_{(i,l)}$, means ($\mu_{(i,l)}$), and covariance matrices ($\sum_{(i,l)}$).

$$c_{(i,l)} = \frac{\sum_{t=1}^{T} \gamma_{(i,j)}(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$
$$\mu_{(i,l)} = \frac{\sum_{t=1}^{T} \gamma_{(i,j)}(t) o_t}{\sum_{t=1}^{T} \gamma_i(t)}$$
$$\sum_{(i,l)} = \frac{\sum_{t=1}^{T} \gamma_{(i,j)}(t)(o_t - \mu_{(i,l)})(o_t - \mu_{(i,l)})^T}{\sum_{t=1}^{T} \gamma_i(t)}$$

3.2 Other Tools and Techniques

3.2.1 Normalization of Features

We have disregarded the magnitude and only considered the direction of the force and moment signals. The force is normalized separately from the moment signals. Skubic has conducted work in the past in the area of discovering single ended contact formations (SECF) through normalized force and moment data [27]. We have continued this methodology on these two features, but applying this technique to any of the other features is not appropriate.

$$\text{Normalization} = norm(\vec{v}) = \frac{\vec{v}}{||\vec{v}||}$$
$$\text{where } \vec{v} = \text{vector and } ||\vec{v}|| \text{ denotes the scalar magnitude}$$

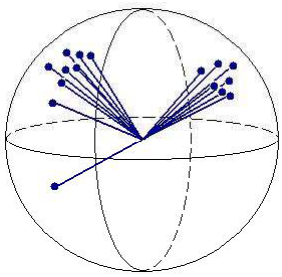

Figure 3.4: Projection of samples onto the unit hyper-sphere

3.2.2 Sliding Window Filter

High-dimensional robots, such as the Robonaut, are plagued by the unavoidable predicament of erroneous sensor and motor recordings. These problems can be generated from the sampling process or from problems directly related to the hardware. In order to compensate

and attempt to filter out the noise, we have applied a linear 5 step averaging window. Assume that v_t represents the present vector to be filtered. Let v_t^* denote the vector after the sliding window has been applied.

$$\vec{v_t^*} = \frac{1}{5}\sum_{i=-2}^{2}(\vec{v}_{t+i})$$

for $2 < t < T - 2$, where T is the total number of samples

3.2.3 Resampling

The raw sensor data, not time-normalized, typically has around 3 times as many hand to arm recordings. There is also a problem in that $V_{(k,i)} \neq V_{(k,j)}$ (where i and j, $i \neq j$, are different repetitions). We need a uniform number of samples to precede with our pattern recognition algorithms. To handle this, we have applied two different techniques. We started off by noting that there was, most of the time, a 3 to 1 ratio of hand to arm readings. The first technique was to generate a stair-step representation of the under sampled data. This means that every arm reading has been expanded out by a factor of 3. This algorithm has a deficiency in that it does not necessarily capture all that actually happened in the newly formulated task description because we are sampling up. This basic stair step algorithm will, however, not handle problems which violate this 3 to 1 relation. We extended this approach to an algorithm that tries to determine where and how to insert for resampling, but it was extremely computationally expensive and appeared to be an overkill. The second process which we ended up using was the matlab resample function. The resample function is an implementation of a polyphase filter.

Chapter 4

Experiments

4.1 Overview

The first round of experiments involved taking a vector time series description for one reach and grasp location, which had been segmented and time-normalized into episodes by Peters, and compare the number of SMC episodes and the location of SMC events to our QS sequence transition points and number of clusters. This involved analyzing the data for one repetition at a fixed location and also looking at combining multiple repetitions to get a more characteristic description of a task. It was later, after interpreting the outcomes from the segmented and time-normalized data, that we reapplied all the procedures on the raw descriptions of the task. Three different clustering algorithms, cluster validity measures, different proximity metrics, and HMMs were used to support the following findings.

The initial problem involved the discovery of clusters in the sensory feature space with the Fuzzy C-Means (FCM). We did not know if the clusters would overlap, so both the Euclidean and Gustafson-Kessel dissimilarity (GKD) measures were applied. The focus then shifted from the FCM to the Possibilistic C-Means (PCM) in order to seek clusters that reflect dense regions. We knew that the robot task recordings would reflect noise, fluctuations at contact points, and other artifacts that could generate features that can affect clustering algorithms as sample points which appear as outliers. The process of a robot moving from one state to another is a continuous process in the sensory and motor feature space that has the effect of leaving trails of samples from one state to the next. With potential outliers and not clearly separable clusters, the PCM provided the ability to minimize the effects of these artifacts and perform more of a mode seeking role.

Both the FCM and PCM algorithms required that we specify the number of clusters. In an attempt to overcome this downfall, the Robust Competitive Agglomerative (RCA) algorithm was used in order to automatically discover the number of clusters as the algorithm progresses. This procedure saved a tremendous amount of computational time, reduced the complexity involved in manually interpreting cluster validity measures, and helped in discovering the best feature subset in a realistic amount of time. The outcome of the RCA algorithm, in terms of QS diagrams and the number of clusters, correspond with the earlier FCM findings and our cluster validity measures.

After applying the above three clustering algorithms and cluster validity measures, we decided to try one last method outside of clustering. This last step involved the application of the Baum-Welch algorithm for HMM parameter estimation to account for the temporal aspects involved in the task. The vector time series is a temporal description, where each recording is dependent on previous time information. The Baum-Welch can capture the spatial density information, i.e. our clusters, and the QS automata in a single framework. The resulting QS automata are the output from the Viterbi algorithm, given the present observation vector time series description and a trained model. The HMM results further validate and reinforce our skill learning results found in the prior clustering stages.

Before clustering or the Baum-Welch algorithm, a few pre-processing steps need to be performed. This involves normalizing the force and moment signals, application of a sliding window filter, and then resampling of the raw data. The following results are shown with 5, 6, and 7 clusters, which is expanded further in the cluster validity section. We also experimented with both the $< SF, SM >$ and $< SF, SM, T >$ groups, as described in the next section. Figure 4.1 demonstrates where each transformation is placed in the experimentation process.

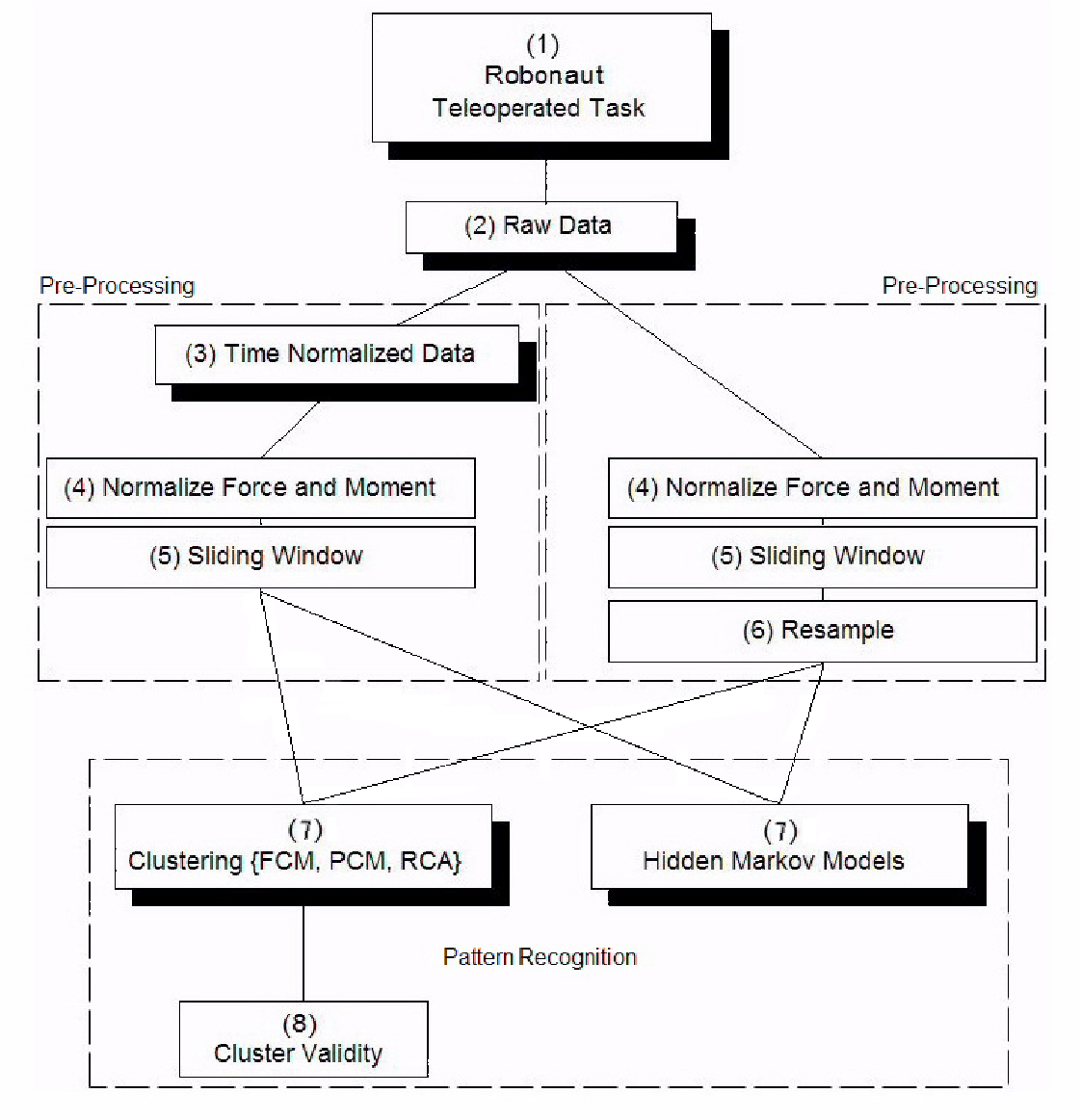

Figure 4.1: Testing procedure flow diagram. Diagram is separated into data acquisition {1,2}, time normalization {3}, pre-processing stages{4,5,6}, and pattern recognition {7,8}

4.2 Feature Selection

Feature selection is currently being manually performed through the inefficient inspection of all feature subset combinations. There are presently no built in mechanisms to perform feature selection as part of the pattern recognition process or in a post algorithm manner. This is primarily due to the inability to automatically score QS sequences. The RCA algorithm, which automatically detects the number of clusters, has made this procedure viable for the moment. The Robonaut has up to 100 dimensions, which have been grouped into 8 sensory categories: arm force (AF), arm moment (AM), shoulder force (SF), shoulder

moment (SM), joint torque arm (JTA), joint torque shoulder (JTS), tactile (T), and finger force (FF). This manual procedure is working well because each feature category has acted independently in an overall positive or negative way, which greatly reduces the number of subsets to analyze. Given the 8 sensory categories, there is the following number of possible subsets:

$$\text{Combinations} = Comb(N) = \sum_{i=1}^{N} \frac{N!}{i!(N-i)!}$$
$$\text{Robonaut Combinations} = Comb(8) = 255 \text{ subsets}$$

From all possible feature category combinations, only the SF, SM, and T categories resulted in constructing QS sequences that transitioned at expected time points and had little regions of indeterminate grade of membership values. Other feature groups did not exhibit meaningful QS sequence behavior. The information that these other groups hold are either not expressed and captured in this representation, or contain partial pieces of information at only selective moments. I believe that if we could perform automatic QS sequence scoring, a forward sequential search for the best feature group could be utilized. This is because, as mentioned above, each category appeared to act independently in a positive or negative fashion. Focus on these three groups reduces the number of subsets to search to 7. Only 2 feature groups, $< SF, SM >$ and $< SF, SM, T >$, from the 7 subsets were found to make a significant impact. We only studied one robot task, but other tasks may have a large number participating features. This places a large burden on the pattern recognition algorithms and the user in manually having to inspect all of the generated results. This is the one of the two principal deficiencies that needs refinement if this approach is expected to work autonomously and scale well.

4.3 Repetition Averaging

A reach and grasp a wrench task was performed through teleoperation at a total of 9 locations. Each location was performed 5 times (repetitions). Repetition averaging is the process of combining repetitions at the same location to generate a more general and characteristic description of the task. The transformed t^{th} sample in the p^{th} episode for location i is defined as $E^{t}_{avg-tn(p,i)}$.

$$E^t_{avg-tn(p,i)} = \frac{1}{R} * \sum_{r=1}^{R} (E^t_{tn(p,i,r)})$$

The default configuration for testing included the application of clustering and the Baum-Welch algorithm to individual repetitions. This was selected initially for simplicity and later for uniformity reasons, due to the fact the raw data was never combined to produce an average representation. Figure 4.2 demonstrates the general behavior related to averaging time-normalized repetitions.

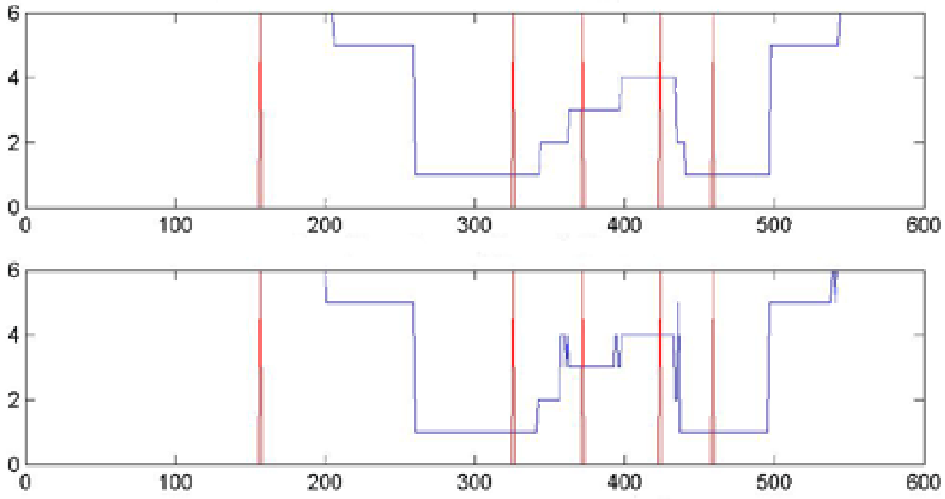

Figure 4.2: SF feature group for the time-normalized data with the FCM for the (top) average of location 1 and (bottom) an individual repetition five at location one.

Averaging time-normalized repetitions has acted like a filter. The goal of the repetition averaging procedure was to remove un-necessary and repetition specific artifacts. It is a basic attempt at a more general description of the overall task. The effect of this averaging has been an all around smoothing in the piecewise linear structure of the QS sequences. There is also far less sporadic transitional activity found.

We have not combined repetitions for the raw data because there is not a direct temporal correspondence of activity across different repetitions. The reason this correspondence exists in the time-normalized data is because of Peters' method of episode detection and time-normalization. The assumption is that each vector in the time-normalized data is correlated with the same activity being performed in another repetition. Because activities will never be performed exactly the same way twice, indices into the raw data time series do not

guarantee that the same activity is being performed at corresponding moments.

4.4 Indeterminate Grade of Membership Region Filtering

The goal of skill learning in this setting required the identification of piecewise defined linear QS sequences that do not exhibit periods of sporadic transition activity. Figure 4.3 demonstrates an example of a period of high transition activity, time 200 to 400, occurring in the SF, SM, and T feature group.

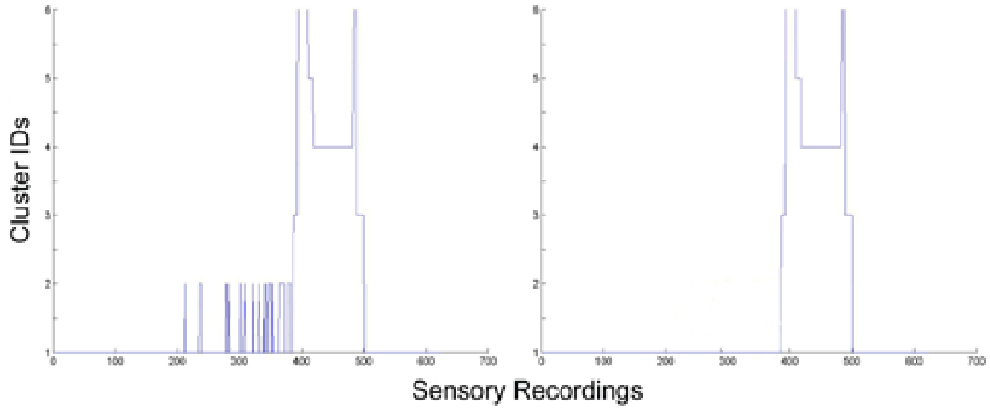

Figure 4.3: (left) QS sequence found by the PCM for the raw data with the (SF,SM,T) feature groups (right) and the indeterminate GOM values identified and removed. Data came from repetition three at location two.

These sporadic regions typically appear in areas of indeterminate GOM values, i.e. where the maximum cluster membership value is extremely low or very close to another value. This usually occurs when our model transitions between QSs, which makes determining the exact moment in time where the model transitions not deterministic. Figure 4.3 shows a more extreme case, but a more typical behavior is demonstrated in Figure 4.4.

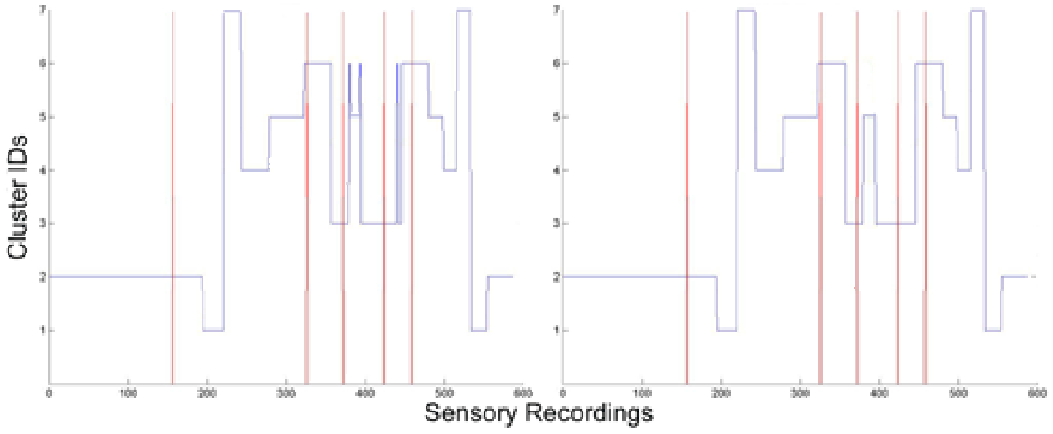

Figure 4.4: (left) QS sequence found by the RCA for the time normalized data with the (SF,SM) feature groups (right) and the indeterminate GOM values identified and removed. Data came from repetition four at location three.

The QS sequence in Figure 4.4 is the typical kind of behavior which this filtering has helped to remove. Transitioning between states is more of a fuzzy concept instead of a crisp concept, i.e. the model is presently transitioning to the next state to some degree. It is hard to believe that any computational technique could identify an exact point in time when the robot transitioned between qualitative states, which are a way to externally model what is internally occurring.

Our criterion is that a GOM value must be clearly identifiable and distinctly maximal over the next highest GOM value. If the robot data set had clusters that were well separated, then it is fair to assume that it would be simple to determine when the model had indeed transitioned between QSs. However, robot data that comes from continuous sensors does not lead to well separated clusters. There are samples that lie between clusters, which represent the gradual and continuous progression from one state to another.

The result of indeterminate GOM values is an inability to clearly determine which cluster the sample should belong to. We approach these indeterminate GOM values by simply retaining the last known good cluster. We consider this a filtering technique. The indeterminate region in Figure 4.3 was completely removed, i.e. recognized just as cluster 1 through this method. The RCA and FCM algorithms had substantially less sporadic transition activity on average when compared to the PCM algorithm. If we used the amount of sporadic transition activity in a QS sequence as an evaluation criterion for the performance

of the clustering algorithm, it reinforces our final project findings that the FCM and RCA algorithms have performed the best for this task.

4.5 Qualitative State Transition Symmetry

One of the top criteria used in evaluating QS sequences is that they should be "well behaved" linear piecewise models. We wanted sequences that contained little or no amounts of indeterminate GOM regions or sporadic transition activity after filtering. As the FCM and PCM results were being analyzed, we noticed that both procedures identified a form of task level sequence symmetry. Our methods identify that the reaching episodes are similar to, in terms of QSs and transition sequences, the retraction episodes. Figure 4.5 shows this symmetry occurring in a FCM generated QS sequence. The cluster identifiers on the QS sequence figure are arbitrary.

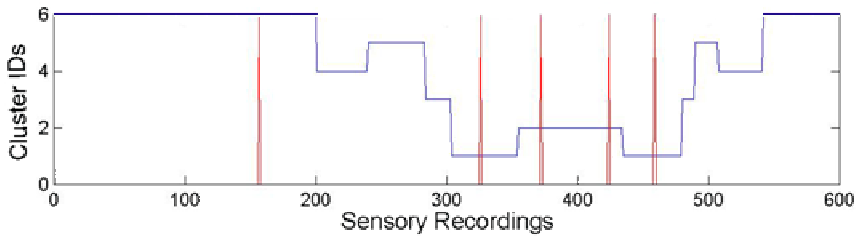

Figure 4.5: QS sequence found by the FCM for the time-normalized data with the (SF,SM) features for repetition three of location two. Our extracted QS sequence is {6, 4, 5, 3, 1, 2, 1, 3, 5, 4, 6}

The QS sequence in Figure 4.5 starts off in QS 6. The model transitions through the sequence of 4, 5, 3, 1, up to QS 2. The model then proceeds to move in reverse order through the same sequence of states, back to QS 6. Our method reveals this symmetry in the task that was previously not captured through Peters' approach. Identifying and representing patterns in a feature space can lead to a more general representation than a sequential scan of extrema in a time signal series. Generalizability is an important part of learning. For a large set of complex skills, a generalized learning method should not only be able to recognize and re-perform tasks just as they were performed as observed, but also be able to abstract and identify other similar sequences not performed in exactly the same fashion. Sequentially

segmenting a signal can identify important moments or events, but does nothing to compare episodes. Our approach compares and captures episode or state similarity by their distance in a feature space. The FCM, PCM, RCA, and Baum-Welch algorithms have all identified this QS sequence symmetry.

4.6 Time-Normalized Data

The following section illustrates the results discovered when we analyzed Peters time-normalized representations of the reach and grasp task. Using the time-normalized data denotes an attempt on our behalf to validate our findings by comparing what was found through clustering and with HMMs to Peters SMC events and episodes. The reason why we consider Peters findings to be of significance is because he has used these SMC events and episodes to reenact the learned skills on the NASA Robonaut. Therefore, Peters SMC results provide us with a starting place to compare and contrast different task aspects while taking into account their usefulness towards the robot. Figure 4.6 shows four QS sequences which demonstrate the most characteristic behaviors in our four different approaches for the time-normalized data.

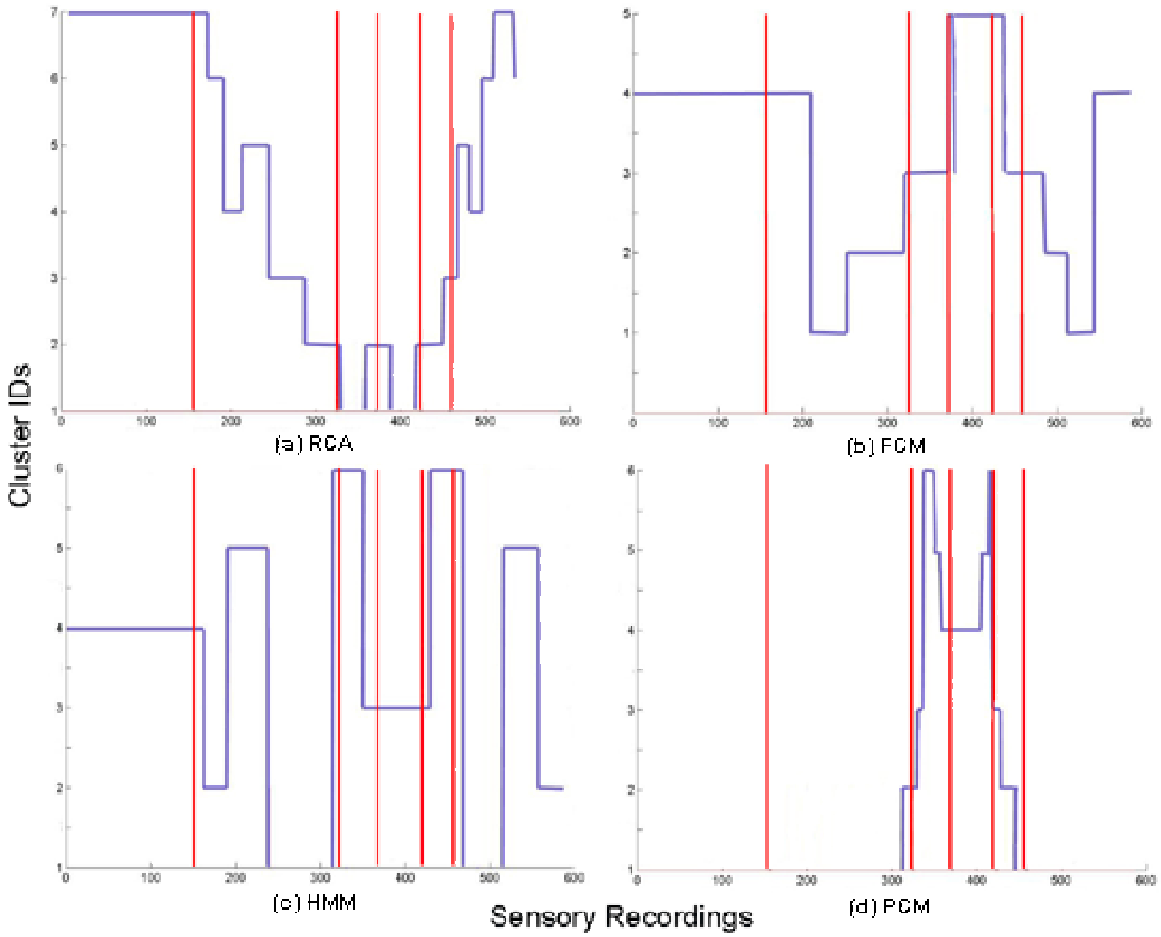

Figure 4.6: QS sequences found with the repetition averaged time-normalized data at location four according to the (SF,SM,T) feature groups through the (a) RCA, (b) FCM, (c) HMMs, and (d) with the PCM. The red vertical bars signal the SMC events found by Peters and the QS labels are arbitrary.

The remainder of this section discusses the results of each algorithm in detail, but there are a few general points that can be made about the QS sequences in Figure 4.6. The PCM algorithm almost always had difficulties with identifying transitions throughout the entire activity. The PCM identified only particular regions in the feature space that do not always reflect the entire task. The FCM algorithm has continuously found results that meet our criteria and are similar to Peters, but presents a difficulty in having to specify the number of clusters to the algorithm. The QS sequence that was identified with the RCA does not exactly match what Peters discovered and found more activity in terms of transitions and number of states. Depending on the RCA settings, which include the learning rate and cluster dropping criteria, different setups lead to four different possible numbers of clusters, which is supported in the cluster validity section. The larger the number of clusters, the

more activity segmentation is observed. In the RCA sub-section, I show that we can acquire results that are extremely similar to those found by the FCM or Peters' SMC segmentation. Lastly, the HMM results closely correspond to the episode points discovered by Peters. HMM results are the most consistent and similar of all the algorithms over different repetitions. However, HMMs required the algorithm to be ran many times in order to get around sensitive problems such as initialization of parameters.

The following tables summarize the transition points that are identified by each algorithm for the first three repetitions at location 1. The standard deviation is computed for each transition point from the first three FCM repetitions. Transition points have been adjusted in each row so that they corresponded to Peters SMC points. Dashes denote no transition relative to that time point.

Peters	157	-	-	326	372	424	459	-	-
Repetition1	-	206	261	322	373	432	472	522	553
Repetition2	-	204	269	325	374	431	469	521	545
Repetition3	-	203	267	321	376	436	473	522	555
StandardDev	-	1.53	4.16	2.08	1.53	2.65	2.08	0.58	5.2

Table 4.1: Transition time points found by Peters and by the FCM algorithm for first three repetitions at location 1. FCM setup was (SF,SM,T) with 5 clusters.

Peters	157	-	326	-	-	372	-	-	424	-	459
Repetition1	-	313	331	339	356	364	409	420	424	436	448
Repetition2	-	311	329	332	355	365	411	418	423	432	449
Repetition3	-	315	328	338	359	364	414	422	425	431	452
StandardDev	-	2	1.53	3.79	2.08	0.58	2.51	2	1	2.65	2.08

Table 4.2: Transition time points found by Peters and by the PCM algorithm for first three repetitions at location 1. PCM setup was (SF,SM,T) with 6 clusters.

Peters	157	-	-	-	-	326	372	-	424	-	459	-	-	-	-
Repetition1	172	184	209	245	288	327	356	385	423	450	462	473	505	522	543
Repetition2	174	185	207	244	285	329	353	388	424	448	460	473	501	526	541
Repetition3	171	184	210	244	283	326	355	385	421	448	465	476	509	524	543
StandardDev	1.53	0.58	1.53	0.58	2.52	1.53	1.53	1.73	1.53	1.15	2.52	1.73	4	2	1.15

Table 4.3: Transition time points found by Peters and by the RCA algorithm for first three repetitions at location 1. RCA setup was (SF,SM,T) and found 7 clusters.

Peters	157	-	-	326	372	424	459	-	-
Repetition1	163	193	238	320	351	432	472	522	565
Repetition2	162	192	237	318	353	430	475	518	566
Repetition3	163	195	239	319	353	429	472	515	560
StandardDev	0.58	1.53	1	1	1.15	1.52	1.73	3.51	3.21

Table 4.4: Transition time points found by Peters and with HMMs for first three repetitions at location 1. HMM setup was (SF,SM,T) with 6 clusters.

4.6.1 Fuzzy C-Means Results

The FCM algorithm was the first and most simple clustering algorithm applied. These results supported our initial hypothesis and gave rise to the idea of looking at other kinds of patterns. While the nature of the sensory patterns are of interest to this project, the QS sequences are of equal or greater value. There was no modification to the original FCM algorithm, only the selection of the Euclidean distance measure and a fuzzy factor of 1.5. The Euclidean distance measure identifies non-overlapping clusters that partition the data set. The GKD metric was attempted, in combination with different fuzzy factor values, but what was identified in this scenario as the best fuzzy factor, a value of around 1.42, lead to QS sequences that had more transition activity at non-meaningful points in time, had a greater number of indeterminate GOM regions, and had the tendency to jump back and forth a lot between states around transition points (which occurred in the indeterminate GOM regions). However, having access to the performance of only one task makes it difficult in general to conclude if all tasks should be allowed to have patterns that overlap in their feature space.

The FCM with a Euclidean measure identifies clusters in this task that give rise to nice linear piecewise QS sequence behavior, which end up closely corresponding with many of Peters' SMC events. The $< SF, SM >$ feature group captures the majority of activity in the task. However, the resulting QS sequences possibly miss some of the contact events and appear to be slow to identify the transition points that are discovered by Peters. If the tactile feature group is added, then more activity in these regions is noticed and the QS sequences more closely correspond with Peters. Only 5 of the tactile sensors had any activity, and only in episodes that involved contact. However, using the $< SF, SM, T >$ group increased the dimensionality of the feature space from 6 to 25. Figure 4.7 illustrates one sequence extracted by the FCM in the $< SF, SM >$ feature space.

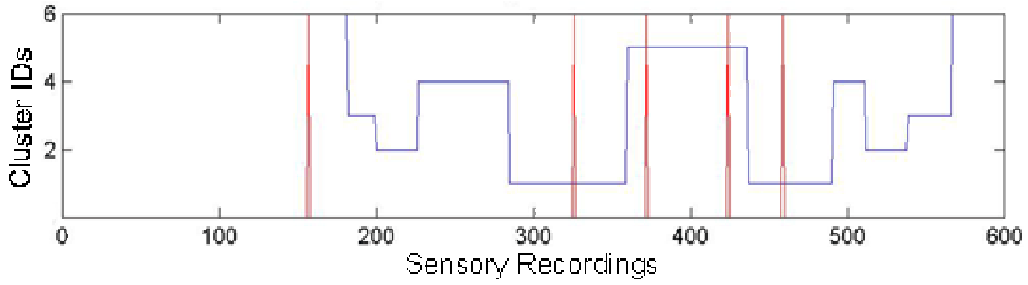

Figure 4.7: FCM QS sequence for the time-normalized data with the (SF,SM) features, repetition one at location one.

Our QS transition points identified in Figure 4.7 are similar to Peters, marked with red vertical lines, but do not occur at the exact locations identified as SMC events. The SMC event markers were taken as a point of reference, but not as a goal or single indicator of success. Our methods were never expected to match perfectly, but the similarity we found in terms of different feature sets and procedures is interesting. We miss the contact events identified by Peters if we only use the $< SF, SM >$ feature categories. This figure and feature category was intentionally selected to demonstrate the effect of leaving out the tactile groups. Figure 4.6 showed the effects of including the tactile features, which almost looked identical to Peters SMC events.

4.6.2 Possibilistic C-Means Results

Approaching this topic with the PCM represented a way to identify a different kind of pattern. The PCM is a mode seeking algorithm and not a partition seeking algorithm like the FCM. We believe that it was an important algorithm to apply, but it did not ultimately help generate useful QS sequences. The algorithm had to be ran too many times (initialization of the prototypes), coincidental clustering worked out to be more of a nuisance than a benefit (i.e. where multiple prototypes converge to the same cluster), and the resulting QS sequences did not have the expected transition activity that was being observed in the other algorithms. When the number of clusters was over specified to the algorithm, more activity in the regions previously missed was occasionally captured. However, in these cases there was too much overall transition activity for the results to be used. In many situations the over specification of the number of clusters ended up converging to the contact regions, and

the other transition were still missed.

The PCM algorithm appeared to do its job in detecting dense regions in the feature space, but the resulting prototype locations ended up either delaying or speeding up our expected QS sequence transition points. In many cases it missed every episode except for intervals that involved contact. It comes at no surprise that higher activity time intervals are not always reflected as dense feature space regions. When realized as a FCM or RCA clustering problem, these regions are identified with a Euclidean distance metric as a partition that can be used in constructing a meaningful QS sequence. Figure 4.6 demonstrated how the PCM failed to capture any of the reach and retract episodes with the $< SF, SM, T >$ features. The resulting prototypes are all found in the contact episodes. This behavior occurred almost always with the PCM.

4.6.3 Robust Competitive Agglomerate Results

As mentioned above, the RCA algorithm automatically works to determine the number of clusters from an initial over specification in the number of prototypes. This technique relaxes some of the burden associated with identifying good initialization locations of the prototypes. As the RCA algorithm works to converge, as α goes to 0, the algorithm turns into the original FCM. The RCA algorithm has proven to provide excellent results in far less computational time then attempting to run the FCM algorithm and analyze cluster validity results. In the majority of our test cases, the RCA algorithm has converged to the number of clusters that we manually interpreted through cluster validity. This algorithm has made our enumerative approach to discovering the best subset of features to use in skill acquisition feasible. The only burden at this point is the manual inspection of QS sequences. Figure 4.8 shows the result of applying the RCA to a time-normalized repetition.

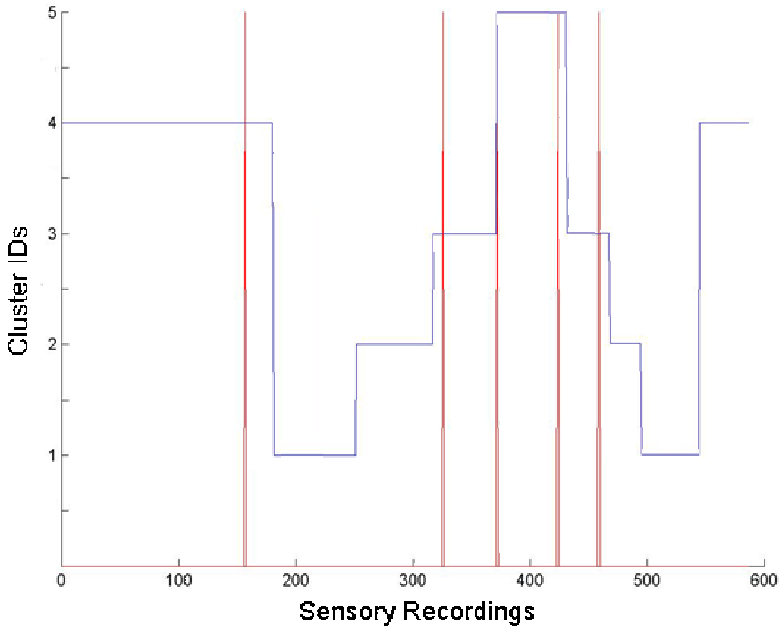

Figure 4.8: RCA results for the time-normalized data with the (SF,SM,T) feature groups. The data is from repetition one at location two.

The QS sequence in Figure 4.8 reveals almost identical results to those that Peters found (vertical red bars). The difference is that our first and last two transitions occur a little later than Peters event points. The late responses may be due to a delay in the movement of samples through the feature space between clusters, which is related to where the 50% GOM boundary is calculated. These regions of low indeterminate GOM values are hard to absolutely place in either cluster without any other outside information. Our method also identifies one additional transition point that Peters does not discover. This transition occurs before the second event. This is almost half way during the reach phase, before the contact is made with the object. The reach and retract episodes appear to be a sequence of smaller sub states.

Furthermore, the RCA results in Figure 4.8 look like those found by the FCM in Figure 4.6. The RCA results are a little closer to Peters SMC events. When the cluster dropping criteria in the RCA is made easier, i.e. the algorithm has a simpler check to drop clusters, the results tended up more closely corresponding with Peter's results. The learning rate and the test criteria to drop clusters are the two primary parameters to the RCA and their effect on this task has been to further partition the clusters and show an increased amount of transition activity. This increase in transition activity identifies more detail in

the performance of the task, but all of the same symmetry and general transition points show up (look at Figure 4.6(a)). Overall, the RCA algorithm has performed very well and is our choice of method because it has simplified many steps through automating the need to specify the number of clusters.

4.7 Raw Data

Up to this point, we have been clustering Peters time-normalized versions of the reach and grasp task. The next procedure involves reapplying all of our procedures to the raw data. Before this can happen, the raw data must be resampled so that the number of arm samples is equal to the number of hand samples. These are the two primary categories that the Robonaut records, which do not occur at the exact same moments in time. We are looking for the same transition activity and relative time points of QS transitions between the raw and time-normalized outcomes. There are no ground truths for the raw data. We did not have access to the time points that were extracted in the time-normalized data. Figure 4.9 shows characteristic results of all four algorithms for the raw data.

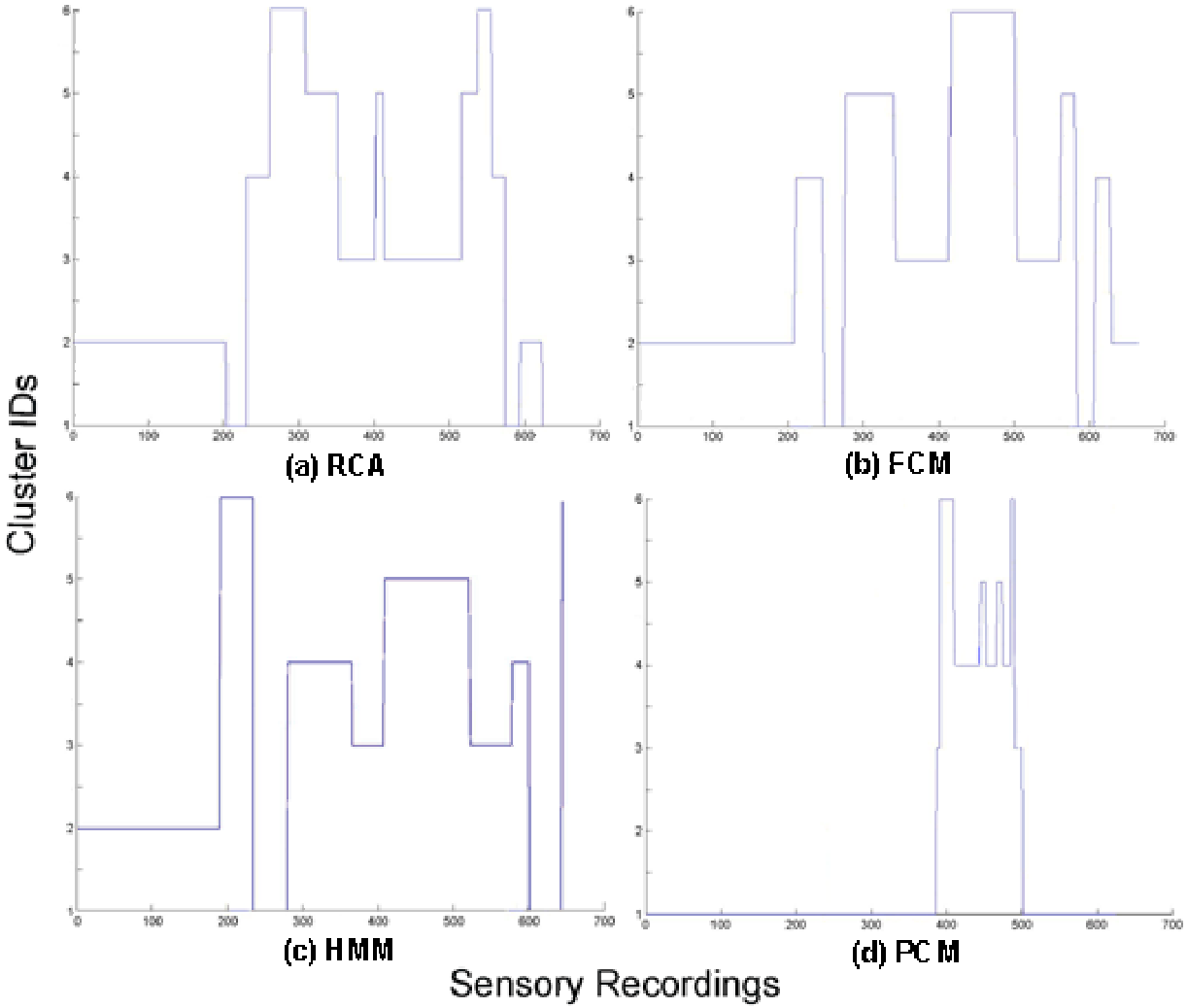

Figure 4.9: QS sequences found in the raw data from repetition two at location five according to the (SF,SM,T) feature groups by the (a) RCA, (b) FCM, (c) HMM, and (d) with the PCM.

We discover essentially the same QS sequences with the same transition symmetry behavior for the raw data as the time-normalized data. In some cases the system is in identical states for relatively the same amount of time. This means that the length that the system is in the first QS in one repetition is the same length as in another repetition. In the remaining cases, the same number of clusters and symmetry exist, but the transition points do not occur near the same time point. This is do to the fact that the same task is never performed exactly the same twice, so the system is in the different states for different periods of time. The majority of differences in repetitions depends on the time point at which the system transitions between states, which was a fixed pre-computed number in the time-normalized data.

4.8 Hidden Markov Model Results

The above sections demonstrated our ability to discover sensory patterns as clusters, and then manually perform a hardening (picking the maximum membership value) of the membership matrix to generate a QS sequence. One fear was that clustering would not incorporate enough of the temporal aspects to accurately reflect the task. From the looks of the clustering results, this is less of a factor than what was originally thought. The application of the Baum-Welch algorithm for HMM parameter estimation allows us to learn the sensory patterns as well as the transition model in one framework. The results are similar symmetric QS sequences found earlier through clustering. The following sections show the QS sequence outcomes, but also go into greater detail into understanding the learned task according to the HMM estimated parameters.

The earlier clustering results benefited by being able to combine multiple repetitions of time-normalized data. The raw data is not combined because there is not a temporal correspondence between repetitions. One benefit of the Baum-Welch algorithm is that an ergodic model may be estimated from multiple observation sequences of different lengths, i.e. our individual raw repetitions. I have applied a continuous observation HMM (COHMM), which assumes that each hidden state is only composed of one probability density function (pdf). A downfall in using HMMs instead of clustering is that there are more initial parameters to specify, because of the state transitional model. This has resulted in the need to rerun the Baum-Welch algorithm multiple times and select the maximum model according to the likelihood of the observation sequences. Figure 4.10 shows three QS sequences that are generated from the Viterbi algorithm, from a model that was estimated from all five repetitions at location 1 (each observation sequence was resampled to a length of 600).

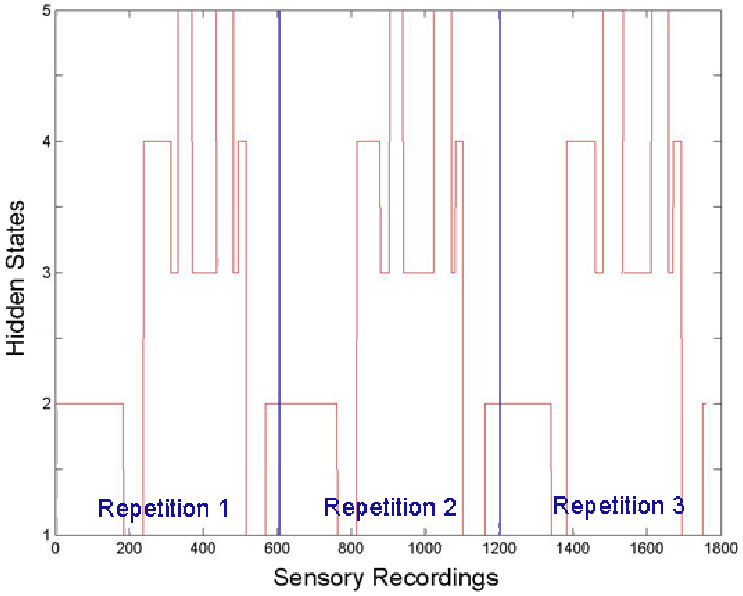

Figure 4.10: Three HMM QS sequences identified by the Viterbi algorithm for the (SF,SM,T) feature groups. The model was learned by training on all five repetitions at location one for the raw data, each observation sequence was resampled to 600 recordings.

The QS sequences that are shown in Figure 4.10 are extracted by the Viterbi algorithm. This is done by presenting a repetition as an observation sequence to the trained model, and using the model to compute the most likely sequence of hidden states that generated this sequence. Repetition 2 begins at time point 600, while repetition 3 starts at time point 1200. The results have been shown in this format to demonstrate that one useful model was learned that uniformly reflects all 3 repetitions. These results are for the raw data, so the resulting transition points can not be directly compared to our earlier or Peters transition time points. The HMM results for the time normalized results were shown in Figure 4.6. The QS sequences all appear to generate the same general transition description. Note how the HMM method has also picked up this task level symmetry that we found earlier in clustering. HMMs have continuously generated QS sequences that are more similar over multiple repetitions. More likely than not, this has to deal with the way that the model learns by presenting multiple observation sequences. When only a few repetitions from the same location are used for training, it is shown that this still holds.

4.8.1 Hidden Markov Model Transition Matrix

One of the parameters that the Baum-Welch algorithm estimates is the state transition matrix. The state transition matrix can be used in a structural fashion to analyze what transitions are likely, which are possible, and those that cannot occur. The matrix values range in the probabilistic interval $[0, 1]$, and are colored in the below figure to indicate their probabilistic value (where black is denoted by a value of zero, within working precision, and white indicated a probabilistic value of one). The matrix has the same number of rows and columns, where each row and each column represents a single hidden state. Therefore, each cell denotes the transition probability of going from state i to state j. The diagonal of the matrix denotes self transitions. Figure 4.11 is an illustration of the HMM transition matrix obtained from Figure 4.10.

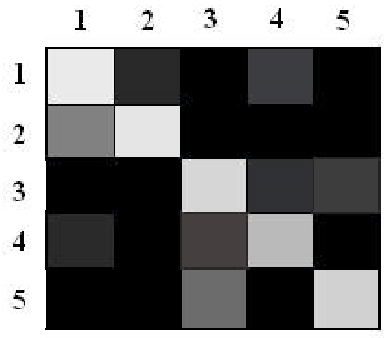

Figure 4.11: Transition matrix for the five state HMM model in Figure 4.10, which was trained on repetitions one, two, and three at location three for the raw data.

The transition matrix in Figure 4.11 is reflective of most HMM transition matrices. Most HMM transition matrices are sparse, i.e. they do not have many non-zero (within working precision) entries. This is one particular reason why most programming libraries for HMMs use sparse data structures to save storage space for the transition matrix. This is not a problem in the case of just one transition matrix, but when a system maintains a set of HMMs that have a large number of hidden states, this storage space adds up quickly.

A large advantage related to using HMMs instead of clustering involves classification. The transition matrix that is obtained by the Baum-Welch procedure is a vital component

in recognizing future similar task performances. Clustering only produces a partition matrix that can be used to generate a single QS sequence. HMMs internally work to approximate the model transition matrix as the algorithm progresses. The following table shows the classification results obtained when using a model that was trained on repetitions 1 and 2 at location 1 and repetitions 2 and 3 at location 2. Due to the lack of another task to compare the acquired models against, the following results show the classification performance on testing and training data from the single task for the first two locations, which will be used to show the versatility and some of the limitations related to HMMs.

	Classification Results
All Repetitions from Locations 1 and 2	.90
Training Data	1
Modified Repetitions	1
Sub-sequence Repetitions	1

Table 4.5: Classification results from a model trained on the first two repetitions at location one and repetitions two and three from location two, and tested on all ten repetitions at location one and two for the raw data with the (SF,SM,T) features. Also shown are the results from 2 observation sequences that should not be recognized, generated by reordering the sensory recordings from repetition two and three from location one and randomly inserting a few randomly generated sensory vectors. The last row indicates the results of taking a sub-sequence of observations from repetition one and five at location two and performing classification.

The results in the above table show that the estimated model correctly classified 90% of the repetitions from location one and two (only missing repetition 4 at location 2). The modified repetitions show that when a different observation sequence is analyzed, which is expected to not be recognized, the approximated model was able to reject these observations. Different types of tasks need to be recorded in order to generate classification results worthy of demonstrating the accuracy of HMMs in this domain. What these results do is show the feasibility and applicability of this technique to the domain. The sub-sequence example shows that as long as part of the task is performed it will be classified. The task does not have to performed in its entirety, which relates to the order of the HMM.

4.9 Cluster Validity

A different validation procedure for our clustering algorithms, outside what can be a subjective process of analyzing QS sequences, involves cluster validity measures. These measures can be used to find the number of clusters. As mentioned above, the majority of cluster validity measures generate an index value that reflects some relation or ratio of cluster separation against inner cluster scatter. We have selected the Davies-Bouldin (DB) measure because of how it handles the calculation of inner cluster scatter and cluster separation. This section presents a few of the most salient and illustrative examples to show what was discovered. Figure 4.12 illustrates what the DB found for repetition five at location three with the raw data.

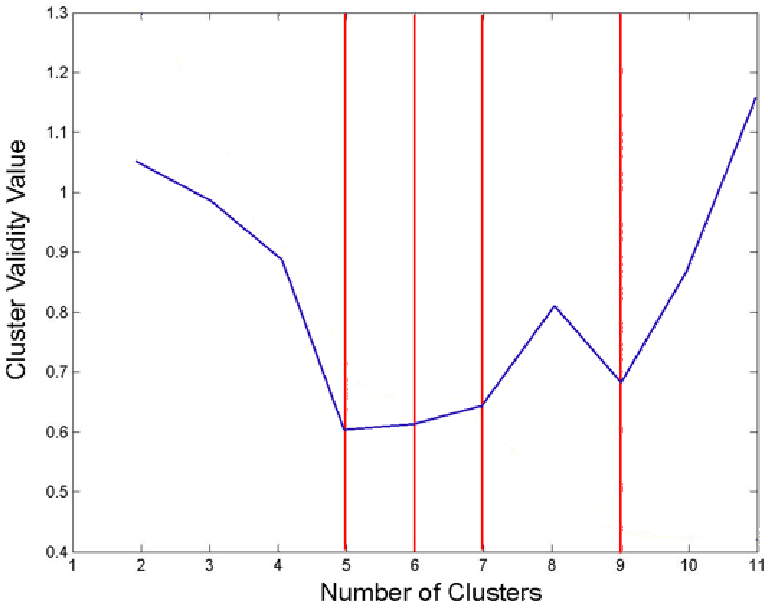

Figure 4.12: RCA result for the (SF,SM,T) feature group for repetition five at location three. Vertical bars are superimposed on the graph to indicate important number of cluster points. These points are identified as important because they represent where the graph levels out or is flat. Multiple points appear to be identified in this figure.

The vertical bars in Figure 4.12 denote points that are interpreted to be indications of the number of clusters. These diagrams are assumed to be reflective of the number of clusters, i.e. that clusters do exist in the data for this particular feature combination, due to the results which have emerged from this research, the overall good general behavior of the following CV graphs, and the correspondence of the QS sequence and CV results.

For well-behaved data sets that exhibit nicely separated and compact clusters, a CV graph should indicate a distinct trend that clearly indicates the number of clusters. However, we have noticed that there are instances in this research where DB graphs indicate multiple numbers of clusters. The graphs indicate that there are possibly 5 or 9 clusters. Cluster numbers 5 and 9 are important because they reflect major points in the graph where the curve levels out. Cluster numbers 6 and 7 are where the graph is flat for a period of time, which could be a further partitioning of the five clusters. Because of the positive results that have emerged for the different numbers of clusters that CV appears to indicate, our interpretation of this phenomenon is that a hierarchical structure exists in which large clusters can be subdivided into smaller partitions. Figure 4.13 shows an example RCA result if we assume that there are 9 clusters.

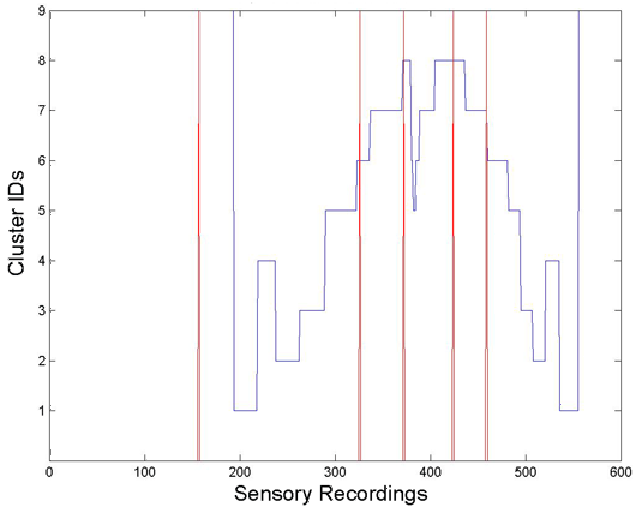

Figure 4.13: RCA results for the (SF,SM,T) feature groups for repetition four at location one of the time-normalized results.

Notice that the resulting QS sequence still shows the transition symmetry, but there is an increased amount of activity. We experimented with the cluster drop criteria in the RCA algorithm and found that we can more or less control which of these cluster numbers the algorithm discovers. The particular values for these parameters are empirically found and do not appear that they would hold up in general for an automated system, i.e. they may have to be rediscovered for each task. However, there was a general rule of thumb that we followed when trying to identify these numbers. We found what might be an obvious relation between

making the algorithm have a easier drop criteria and finding smaller numbers of clusters. The two easiest numbers of clusters to identify were the cluster numbers five and nine, which ended up representing the two extremes of our picked parameters, i.e. the most easy and the harshest drop criteria values (the cluster cardinality comparisons). It is important to note that there were no other numbers of clusters identified when we varied these two parameters. Figures 4.8, 4.9, 4.6, and 4.13 respectively represent the results found with the RCA for five, six, seven, and nine clusters. The ability to favor QS sequences with more or less transition activity through modifying the drop criteria in the RCA could be a very useful tool.

4.10 Discussion

When robot skill acquisition is performed by identifying primitives as sensory patterns in a high dimensional spatio-temporal feature space, the selection of procedure and ultimately the type of pattern can drastically affect the resulting QS sequences. The goal of applying different procedures is to gain a better understanding of the patterns, which aids in the selection of appropriate primitive and skill representations. The FCM algorithm has generated very useful QS sequences that correspond in many ways with the SMC episode points which Peters identified. The PCM algorithm was not useful for identifying each and every expected QS transition point in the task. However, this does not mean that the PCM is not applicable to this domain. The dense regions that were identified by the PCM could be useful if one only wants groupings in the feature space that have a high density. The PCM was successful in finding activity in the contact episodes, which loosely corresponded with Peters SMC points.

The RCA algorithm was extremely helpful because it relieved the need to specify the number of clusters. Different criteria for dropping clusters in the RCA effects the final results, which correspond to what was identified by cluster validity measures. When the drop criteria is made to be simple, i.e. the algorithm drops clusters more often, the RCA algorithm converges to five clusters, whose results correspond to Peters SMC results.

In an attempt to see if the clustering algorithms did not capture enough temporal aspects of the task, HMMs were applied. HMMs have the disadvantage of having to specify

the number of hidden states. Initialization of the HMM parameters appeared to impact the results in a more profound fashion than the three clustering algorithms applied. The HMM results closely corresponded to Peters findings, and was the most similar results found over all the repetitions at different locations. The main advantage of using HMMs involves the acquired transition matrix and the ability of the framework to perform classification.

The patterns in the robot's sensory feature space appear to be a combination of dense and sparser regions. The PCM showed which regions were dense, but the RCA and the FCM, partition generating algorithms, identified clusters in the areas outside of these dense contact regions. The most useful QS sequences involved using all of the acquired sensory recordings and treating the patterns as partitions of the recordings. The resulting partitions picked up the dense areas that the PCM found, but also identified useful QSs and transition points in other areas. The PCM found clusters in the contact and object grasping episodes, while the FCM and the RCA identified activity in these episodes and in the reaching and retracting episodes. The reaching and retraction episodes reflect sensory recordings that are being sampled as the robot is undergoing a lot of change. It is not surprising that these episodes do not show up as fairly dense regions in the feature space. HMMs also identify all of the activity throughout the entire task. HMMs are very useful because of the way that multiple repetitions can be used in approximating the model, the classification power of HMMs, and the most likely hidden state sequence.

The time-normalized data provided a way to compare SMC events against our QS transition points. There was no ground truth for the raw data. We were never provided access to the raw time points that corresponded with Peters' time-normalized SMC events. This makes it difficult to determine which technique is the most accurate for the raw data. The number of samples in the clusters are different for the raw data versus the time-normalized data. This affects the FCM and the RCA algorithms, which perform partition seeking. The contact episodes that the PCM algorithm identified in the raw data were observed in all of the other three algorithms. However, the time points that the FCM, RCA, and HMMs found outside of the contact regions were different. The most consistent and similar to the time-normalized results were the PCM and the HMMs. The probability density functions in

the HMMs and the mode seeking properties of the PCM looked to be the least effected by the density change in the clusters.

Chapter 5

Conclusions

5.1 Summary

Robot skill acquisition can be viewed as identifying sensory patterns in a spatio-temporal feature space. Sensory patterns are primitives that can be used for learning and executing skills. Our investigation of robot skills as finite state automata, with HMM probability distributions or QS cluster sequences as the skill primitives, has lead us to conclude that teleoperated tasks performed on the NASA Robonaut can be learned from patterns in the robot's sensory feature space. The application and usefulness of the acquired skills is a future problem that can be demonstrated by performing the skills on the robot, or through using these skill primitives in a control or intelligent robot setting.

Clustering and HMM parameter estimation are two pattern recognition procedures that can be used to identify skill primitives as patterns in a robots sensory feature space. The FCM algorithm, with a Euclidean distance metric, has been useful for discovering symmetric QS sequences that compare to Peters method of SMC episode and event extraction. The PCM algorithm has not proven useful in continuously discovering QS sequences that do not exhibit sporadic transition periods, do not have regions of indeterminate GOM values, and transition at meaningful points. The RCA clustering algorithm has produced the best results and been helpful in reducing the computational time, algorithmic complexity, and degree of manual human intervention involved in analyzing and interpreting cluster validity results. The RCA is presently making the enumerative procedure of clustering all feature subsets feasible. For these such reasons, the RCA is the preferred clustering algorithm in this work.

Our second approach to skill learning involved the estimation of parameters for an

HMM. The Baum-Welch algorithm was used to learn the robot skills directly. This approach helps to truly take into account the temporal nature of the task. Multiple repetitions of a task, which may be of different lengths, can be directly combined and used in HMM parameter estimation. The results from this procedure come close to Peters method of SMC episode extraction. The HMMs have also discovered this symmetric behavior for the reach and grasp task that was found in clustering. The only downfall behind using an HMM over clustering is that the RCA estimated the number of clusters automatically, and an HMM presently has to have its parameters hand coded.

5.2 Possible Areas of Application

A discrete and stochastic QS automaton has a variety of possible application in the robotics domain. These areas include control theory and possibly even a hybrid or cognitive system. In a control theory setting, the QS sequence may be used to drive an entity such as a discrete event controller (DEC). The controller could incorporate local sensory information and the finite automaton to make decisions about which state to transition the system into next. A hybrid or cognitive application might specify how to use the acquired skills as a form of low level implicit knowledge. This provides the ability to reenact the skills as observed, model decision-making at a low, reactive level, or extract symbolic domain information and relations that may be used in a higher level analytical decision making framework. The majority of present work in this area involves the application of artificial neurological networks (ANN) for low level implicit knowledge and First Order Predicate Logic (FOPL) for explicit knowledge and decision making. The difficulty lies in identifying procedures to extract symbolic information from sub-symbolic networks such as an ANN. ANNs are typically viewed internally as black boxes, but an HMM is a more state full and symbolic representation that might be more exploitable.

5.3 Problem Areas

The following two sections illustrate critical areas that are limiting this approach from being a fully automated process. This is where the majority of time and human subjective

interpretation presently comes into play. These topics include feature selection and the identification of scoring criteria for QS sequences.

5.3.1 Feature Selection

Feature selection (FS) is an important part of this work that is limiting the procedure from scaling well and being automated. FS refers to processes of identifying and singling out features, or combinations of features, that are salient to the problem or task being learned. We have shown that Robonaut's original 100 dimensional feature space is not adequate for discovering useful sensory patterns. This 100 dimensional space is also very computationally intense on the pattern recognition algorithms. At a particular point in time, only a subset of sensors and motors are participating in the task. The SF data provided promising results, and when coupled with the SM group we identified the majority of the task. The tactile group was then added and many of the contact episodes captured. If we had a method to automatically score the QS sequences and generate something similar to a classification result, we could possibly apply an inefficient forward or backward sequential feature selection algorithm. However, my intuition tells me that this sort of a procedure will not scale well. The ultimate goal is to be able to perform a kind of feature selection or evaluation process during the pattern recognition algorithm. If this is not possible in the algorithm frameworks that we have selected, then maybe a pre-processing procedure that individually analyzes features to determine when a sensor was not active during a task, or contains mutual and correlated information with another sensor, might help in directly removing features. Another option involves using something down the path of particle swarm optimization (PSO) or ant colony optimization (ACO) to perform feature selection. These types of algorithms might be the most appropriate when trying to search for a feature subset from a robot that has a large feature search space which can render calculus and other types of sequential or hill climbing techniques inoperable.

5.3.2 Lack of Scoring

If there is one central problem embodying this research, in terms of automating and further validating the work, it involves the lack of scoring criteria for evaluating QS sequences. This problem presently requires human intervention. I do not expect this procedure to scale well, in terms of the number of results to analyze and manually score. However, I am skeptic about generating a general set of criteria to automatically score anything involving robot behaviors or skills. This does not appear to be an area in which there is one correct answer. The proof is more empirical and will probably lie in the application and usage of the algorithm outcomes.

5.4 Future Work

There are four primary areas from this work that I would like to investigate in the future. These subjects do not necessarily require that I continue this particular research in this context of skill learning. The majority of these problems are related to the procedures that I applied and could be used in a variety of different domains for other applications. The first subject is very important and is a limiting factor of this work. For this particular procedure of skill learning to succeed, I need to identify an automatic way to perform feature selection or feature reduction inside the context of the different pattern recognition algorithms that were applied. This will require the modification or construction of a new algorithm possibly based on clustering or HMMs. I would like to avoid any procedure that requires too much domain specific filtering or knowledge initially, as well as procedures that run after the pattern recognition algorithm is finished. Due to the size of the search space in this robot project, as it relates to feature selection, procedures based on evolutionary computing or swarm optimization may prove to be helpful, compared to classical gradient or hill climbing methods.

A second topic involves the constraint that a user has to presently specify the model or clustering parameters. The RCA clustering algorithm proved to be extremely helpful in removing the need to specify the number of clusters. However, I am not sure that the RCA incorporates enough of the temporal aspects in the task by just identifying points in

a spatio-temporal feature space. Our project findings show that these different algorithms identify similar results, but I did not study a variety of different robot tasks. This was one of the initial pushes for formalizing and casting the task as an HMM problem. One goal would be to identify a procedure to learn the HMM parameters during the estimation of the model or though refinement of the model.

Matthew Brand has performed research in the area of HMM structure discovery through using entropy to learn how to maximize model structure while minimizing the entropy of the joint distribution [52] [30]. Brand claims that this procedure helps to produce models that are more understandable and able to be analyzed by a human. He makes a few grand claims, but the technique and his results appear to be promising. The models which he learns had far fewer hidden states and the transition matrices were more sparse. Brand also introduced a way to use a single HMM to track multiple events in traffic activity. This appears to be an interesting way to use HMMs for tracking multiple simultaneous processes, which can be a combination of external or internal processes. One problem, depending on how you look at it, with Brands approach is that the entropy and criteria to remove hidden states from the model appears to be relatively complicated and tied to the HMM.

An alternative approach, which is much simpler in concept, design, and execution involves optimization by Evolutionary Computing (EC) [3]. Learning model structure through EC can be done outside of the Baum-Welch algorithm. This means that one can use and continuously change the type of HMM and not have to modify the structure learning procedure. Brands method will require re-deriving the majority of the mathematics and the learning procedure whenever the probability distribution function type of HMM internals change. There are presently a few published techniques that deal with learning HMM structure through using EC [16] [31] [50]. Each of these works appears to be quite simple and only work for the most simple of models, a DOHMM. This area appears to be under researched and might be a good area to investigate.

The third matter involves applying Hierarchical HMMs (HHMM) to skill learning or any other general temporal process. Conventional HMMs consist of a single flat layer, in terms of model structure. A variety of processes contain many common substructures, which

would be redundancy learned and modeled in a single layer finite state automaton. Many domains, such as activity monitoring [18], may be composed of multiple levels of hierarchical temporal structure. Robot skills could also be viewed as possessing multiple levels of hierarchical structure, which could be of great benefit to model. As a future extension, I would like to look into whether multiple temporal hierarchical layers exist in skill learning or in activity monitoring and how much benefit there is in learning them.

The last area of possible future extension involves taking the learned QS sequences and discovering a method to generate new future similar skills. The skills represent a way to describe and reproduce a particular skill, but their results may be able to be combined or broken down into smaller or larger reusable components for new skills. One would desire a reach and grasp task that is capable of performing a variety of different reach and grasp procedures as long as the most elementary pieces in such a general skill have been learned.