VERBS AND ADVERBS AS THE BASIS FOR MOTION GENERATION IN HUMANOID ROBOTS

By

Albert William Spratley II

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements for

the degree of

MASTER OF SCIENCE

in

Electrical Engineering

August, 2006

Nashville, Tennessee

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

The goal of this thesis is to evaluate the suitability of the Verbs and Adverbs interpolation algorithm in the context of motion generation for humanoid robotic systems. The Verbs and Adverbs algorithm, as will be presented in greater depth in later sections, was originally devised as a method for generating realistic animated figures by attempting to generalize example motions within a defined Adverb space. These examples were then used to re-create similar motions with a near infinite degree of variability.

History of Robotics

The first use of the term “robot“, in its modern context, was in a 1921 play entitled “Rossum’s Universal Robots“ (R.U.R.) by Czechoslovakian playwright Carel Kapek. The play depicted mechanical servants who, once endowed with emotion, destroyed their master. Despite the fact that robotics was not a recognized field until a later date, there are many recorded examples of what would later be considered robots created prior to *R.U.R.*. These examples include a patent filed with the US Patent Office in 1893 for a mechanical, steam-powered horse and a demonstration of a remote-controlled submersible boat by Nikola Tesla in Madison Square Garden at the World’s Fair in 1898.

The next major recorded innovations in the field occurred during one of the most tumultuous times in this world’s history. During the Manhattan Project, in 1942, scientists devised the “telemanipulator“ in an attempt to distance themselves from the dangerous radiation of the materials they were using. The telemanipulator consisted of a series of mechanical linkages that translated the motions of the operator to a crude manipulator while the operator remained at a safe distance. The telemanipulator left much to be desired when compared to modern robotic manipulators. For one, the motion resulting from the operator's commands was often unintuitive. The operational difficulty of the telemanipulator was further complicated by a severe lack of tactile feedback.

Another important event that occurred in 1942 was the publication of “Runaround“ written by Isaac Asimov who would become known as the “father of robotics“. In this story he published the famous Three Laws of robotics [3]:

1. A Robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A Robot must obey orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Later, as Asimov's fictional robots evolved and gained responsibility over human society, it became necessary to include the Zeroth Law which stated *A Robot may not injure humanity, or, through inaction, allow humanity to come to harm.*

Soon after the appearance of Asimov's first story, the field of robotics began to explode in the form of articles, books, applications, and research into the paradigms and architectures being employed. The creations and theories developed during this period led to the introduction of the industrial manipulator in the mid-1950's. Driven largely by the work of

Norbert Weiner, one of the pioneers of modern control theory, these robots could perform a pre-determined operation many times with a higher reliability and accuracy than their human counterparts.

> *...it has long been clear to me that the modern ultra-rapid computing machine was in principle an ideal central nervous system to an apparatus for automatic control; and its input and output need not be in the form of number or diagrams, but might very well be, respectively, the readings of artificial sensors such as photoelectric cells or thermometers, and the performance of motors or solenoids.*
>
> *- Norbert Weiner, Electronics, 1949*

Also during this period, the first Artificial Intelligence (AI) robot,“Shakey“, was developed at Stanford by Nils Nilssen and Charles Rosen. The research conducted during the development of “Shakey“ resulted in the formalized Hierarchical Paradigm that was commonly used until the late 1980’s.

Hierarchical Paradigm

The Hierarchical Paradigm proved a major milestone in the development of robotics. This architecture, also known as the SENSE-PLAN-ACT (SPA) architecture was the first formalized architecture that allowed robots to determine the appropriate action to perform based on its internal goals, external percepts, and known capabilities. This paradigm, however, was not without its downside. The sensing and planning phases required a stable environment. This precluded the robot from performing any actions while the environment was being sensed and a plan being formed. This is actually why “Shakey“ got it’s name. The robot would perform a simple action and then quickly stop to begin sensing what changed and planning the next course of action. The robot was rather tall and top-heavy and tended to shake when it abruptly stopped. As stated previously, despite its shortcomings this paradigm was in regular use until the idea of the Reactive Paradigm took hold in the late-1980's.

Reactive Paradigm

As the robots continued to grow in complexity, researchers began to become very dissatisfied by the limitations of the Hierarchical Paradigm and began to explore methods to improve on the robot's ability to act in a more dynamic environment. Similarly, recent advances in the field of ethology (the study of animal behavior) suggested that many behaviors are almost reflexive in nature. This led robotic researchers in the late 1980's to formalize the Reactive Paradigm still commonly used in simple systems. The Reactive Paradigm is based on the idea that in many cases, the performed action is directly related to the perceived state of the environment. This tightly coupled sense-act paradigm is easily observed in nature. Any person that has ever accidentally placed their hand on a hot stove knows that through no conscious effort, their body commands them to pull their hand away. This clearly indicates a very tight relationship between the sensation created by the hot stove and your body's reflex of removing your hand.

Rodney Brook's seminal paper "Intelligence Without Representation" [7], published in 1991, proposed an extension of this concept in which higher level behaviors can amplify or suppress the action of lower level behaviors. By organizing the behaviors in vertical layers of increasing complexity, he believed that while the individual behaviors in each level do not contain any significant intelligence in and of themselves, the action performed by the constructive and destructive interference of the layers would create seemingly intelligent behaviors. This architecture is often referred to as the "Subsumption Architecture". While, in many cases, a reactive paradigm such as the Subsumption Architecture can prove sufficient and significantly more responsive to a dynamic environment than the SPA architecture, this methodology is not without its drawbacks.

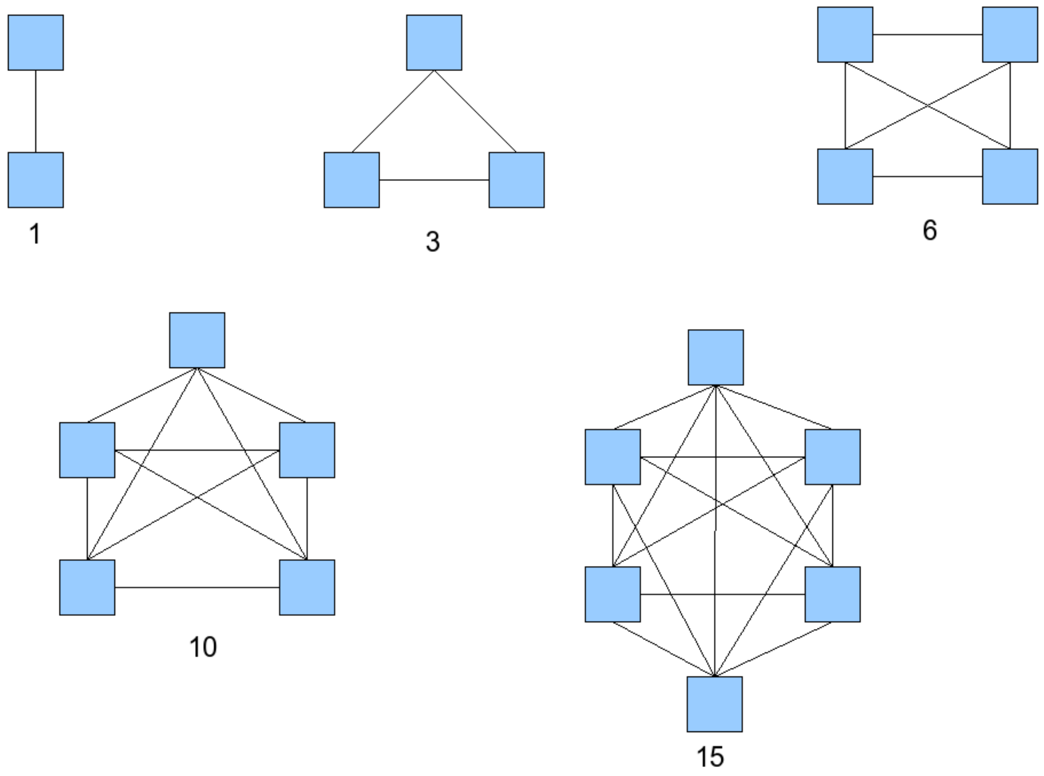

Figure 1: Increasing Complexity

As shown in **Figure 1**, as the number of behaviors increases, the number of interconnections required to connect all the components increases by $\frac{n!}{2!(n-2)!}$. Additionally, it was determined that the coefficients used for determining the constructive and destructive nature of the interaction between behaviors was extremely subjective. Due to these problems, researchers began looking for a suitable compromise between planning and acting. Additional information on the behavior-based architecture can be found in [2].

Hybrid Hierarchical-Reactive Paradigm

The quest to re-integrate the planning phase of the SPA architecture eventually led to the development of the aptly named Hybrid Paradigm in the late 1990's. This paradigm attempted to give the robot a basic vocabulary of behaviors that could act either independently or through constructive or destructive interference as determined by a higher-level process. This process typically involved the more deliberative functions of robotic systems such as planning, map-making, localization, etc. While much work is still on-going in this paradigm, most architectures allow the planning process to override the reflexive behaviors or activate them in a sequence to achieve a desired goal. The architecture used in this thesis is based on a hybrid design and will be discussed more in Section II. Additional historical background on robotics can be found in [16], [28], [14], and [10].

Organization of this Document

So far this work has introduced the primary goals of this thesis and introduced some historical background to further familiarize the reader with the context in which this document is written. Additional terms that apply to the more specific nature of the test platform and the larger goals of the lab will be introduced and explained as necessary. In the next section, the goals and platform will be introduced in detail along with brief synopses of other on-going work. The relevance of this work will be made clear in those terms. Section III will also describe the Verbs and Adverbs algorithm and the particulars of this implementation.

Section IV will focus on the experimental setup and attempt to identify the potential sources of error and the justification for the approach taken. Four example motions will be used in the experimental data set and the justification for the examples and the number of exemplars used will be presented. The results from each experiment will also be presented.

CHAPTER II

RELATED THEORY AND RESEARCH

Intelligent systems should possess the ability to learn, record learned information, represent its environment, “focus“ on task related information, and perform tasks on self-generated motions. “Focus“ is the ability to select from the world and from learned information what is important for a given basic task. Albus [1] proposed the necessity of having a world model, short-term and long-term memory, and task planners. The CIS has made a concerted effort over the last few years to develop an architecture that integrates some of the emerging ideas from neuroscience. The following sections will present some of these ideas as well as information regarding the implementation in use by the CIS.

Memory Models

Current Research suggests that humans have three primary types of memory storage: Long-Term Memory (LTM), Short-Term Memory (STM), and Working Memory (WM). These memory models are each further divided into different types of memory. The following sections will address each type of memory, its memory subcategories, and how they fit into the current ongoing research. Additional, more in-depth information regarding the various human memory models can be found in [27], [24], [23], [19], and [11].

Long-Term Memory (LTM)

Long Term Memory (LTM) is where humans store information that is persistent. While research suggests that information stored in LTM is almost permanent in nature, the ability to easily retrieve that information is dependent on how often it is used. A common expression in neuroscience states ”Neurons that fire together, wire together”. Every time a pair of neurons fire together, the connection between them is reinforced. Similarly, the connections can degrade over time making the associated memories more resistant to retrieval. The LTM stores three types of memory, Procedural Memory (PM), Semantic Memory (SM), and Episodic Memory (EM). Each of these memories encodes different types of information.

Procedural Memory encodes information concerning actions and tasks that we perform. The memory is further divided into two categories: Motion Primitives (MP) and Behaviors. Motion Primitives encode the most basic motions that we, as humans, can perform. They are almost instinctual in nature and can be performed without any significant conscious thought. Behaviors, on the other hand, encode much more complicated types of motions. The simplest way to describe Behaviors is that they encode sequences and combinations of MPs in order to accomplish a task. There is no limit in the number or manner of MPs that are encoded in a behavior.

Behaviors also typically require a more significant level of thought relative to MPs. For example, a MP may be grasping an object on a counter (not reaching to the object, simply grasping it), while a Behavior would be playing the piano. Again however, ”neurons that fire together, wire together”; the more a Behavior is performed, the more strongly encoded that sequence of MPs becomes. It will eventually approach a near MP level. Tying your shoes is a great example of this. At a very young age, learning to tie your shoes is a very difficult, thought-intensive process. However, after tying your shoes every morning for your entire life, it eventually gets to the point where you don't even have to pay attention to what you are doing.

Episodic Memory (EM) is very different from Procedural Memory in that it encodes events. For example, your 10th birthday party would be encoded in Episodic memory. Everything from the smell of the cake and candles, the color of the wrapping paper, to your friends and family singing you "Happy Birthday". One way of describing this is that it is a time-series of sensory information of the important events in our lives. It takes more than an event just occurring to make it important enough to encode in LTM. For instance, you probably don't remember the specifics of your last commute to work. However, you probably remember the last time someone rear-ended your car or you witnessed an accident during your last commute to or from work. This is because the extra emotion of those extraordinary events make the event significant enough for your brain to encode it in long term memory.

Semantic Memory (SM) is yet another beast in and of itself. Semantic memory encodes hard facts about the world in which we live. Semantic memory is what tells us that trees are green, the sky is blue, and the sun is bright. These memories are encoded every time we interact with a new place, a new object, a new person, or see something we don't recognize. One way to describe semantic memory is that it contains data structures with all sensory information relative to things and people we perceive. Unlike EM, there is no time context associated with SM.

Short-Term Memory (STM)

The human Short Term Memory (STM) is very similar to the LTM except it is not persistent. Short term memory uses the senses to store information regarding both the properties and relative orientation of objects in our environment. The sensory information could be words, colors, shapes, smells, etc. This encoding process begins when a human perceives a stimulus. It is retained in this stage for a short period of time before being passed to STM. At this stage, the determination as to whether or not the information should be passed into the STM is made based on attention. Information in the STM can be further transitioned to the LTM via repetition. This process is depicted in **Figure 2**.

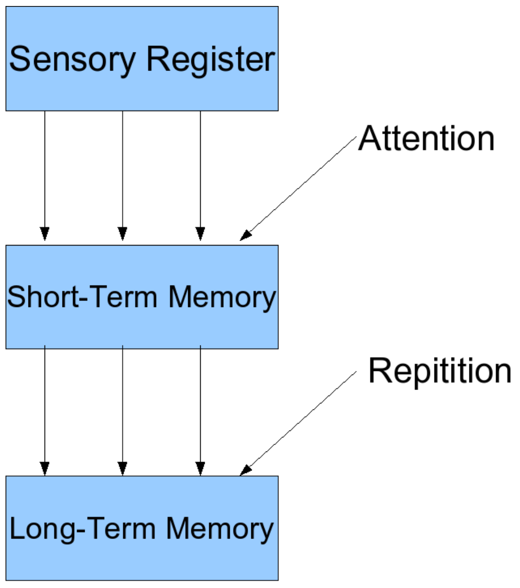

Figure 2: STM Encoding Process [4]

Working Memory (WM) and the Central Executive (CE)

Working Memory (WM) is a very unique and complex type of memory. It is defined as a limited-capacity store for retaining information over the sort-term and of performing mental processes on the contents of this store [12]. Working memory only contains a few discreet “chunks“ of memory at any given point. Early research suggested the size of this store is seven (7) chunks plus or minus two (2) [21] while more recent research has placed the number closer to four (4) [9]. While this may seem limited, WM is what gives humans the ability to focus on a specific task.

Closely associated with the WM is a mechanism called the Central Executive (CE) [5] that, among other tasks, determines which pieces of information from LTM and STM are relevant to the current task and should be brought into the WM. The CE must also be capable of generating plans and monitoring their process, maintaining and updating goal information, inhibiting distractions, shifting between different levels of cognition ranging from unique actions to complex deliberation, and learning new responses to novel situations [22]. According to some research, the CE consists of two pieces: the phonological loop, which is responsible for the storage and processing of linguistic information, and the visuo-spatial scratch-pad, which stores and processes information about objects in our environment relative to each other and to ourselves. This functionality is depicted in **Figure 3**. As we are presented with new tasks, this CE must learn which pieces of information are needed in the WM. In humans, this learning process is believed to be performed, in part, by dopamine projections into the Dorsolateral Prefrontal Cortex (DL-PFC). Other related work in the CIS is attempting to mimic this process using Temporal-Difference Learning (TD-Learning) and

Neural Networks. TD-Learning is a reinforcement learning technique that makes decisions based on expected reward in the future [13].

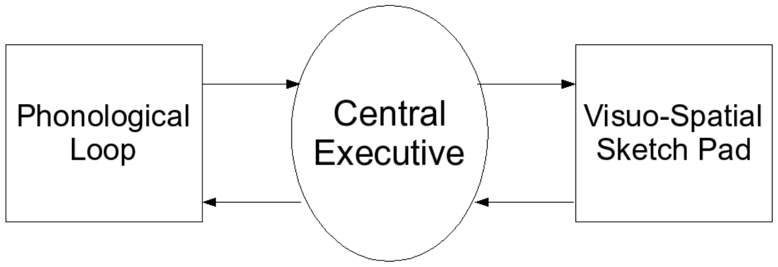

Figure 3: Baddely's Model of Working Memory

Now that a basic understanding of human memory models has been presented, it is important to explain the role of the Verbs and Adverbs alogrithm in this context. The Verbs and Adverbs algorithm will act as a tool for the encoding, generation, and manipulation of motion primitives in the LTM. Once a "library" of motion primitives has been created, it becomes a relatively simple task to string a series of motion primitives together to create behaviors. Behaviors that are used multiple times could also potentially be stored to create more complex MPs if so desired.

CIS Platform and Architecture

At the Vanderbilt University Center for Intelligent Systems (CIS), an upper-torso humanoid robot named ISAC (Intelligent Soft Arm Control) was used for our experimentation. ISAC incorporates a head with two pen-cameras for stereo imaging, microphones for stereo sound pickups, and two arms with basic hand-shaped manipulators. The underlying control architecture used is an in-house architecture called the Intelligent Machine Architecture (IMA).

The following sections will discuss both the platform, ISAC, and the architecture, in further detail.

Intelligent Soft-Arm Control (ISAC)

ISAC, shown in **Figure 4**, is an acronym for Intelligent Soft Arm Control and was originally developed in 1992 as an aid for the elderly and handicapped.

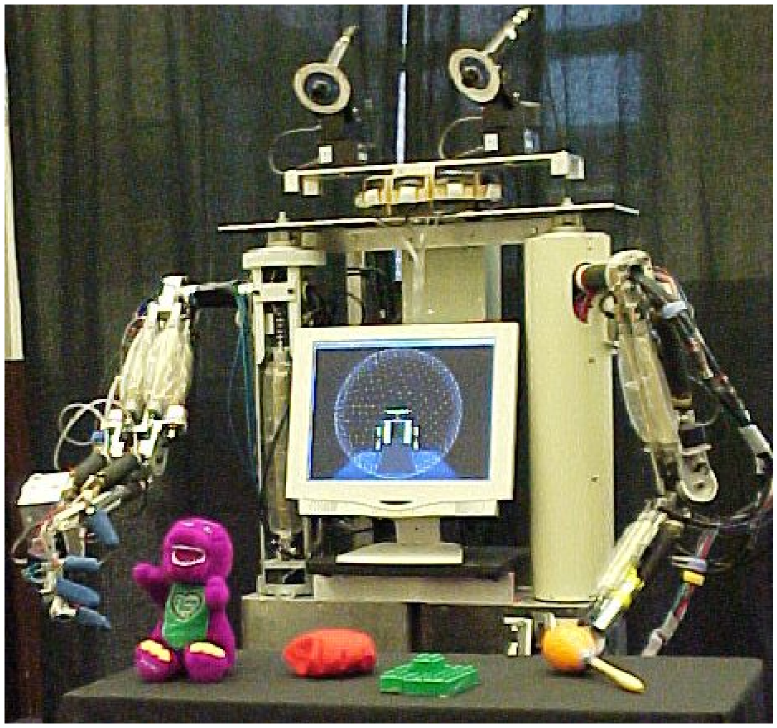

Figure 4: ISAC

In an attempt to develop a safer system for this demographic, ISAC was designed to use pneumatic muscles in its arms that allow the arms to flex upon collision thereby minimizing damage or injury. While this does allow for a safer robot, the highly non-linear McKibben actuators have proven very difficult to model and control. While the CIS has developed a control-law for the arms, creating smooth natural-looking motions has proven a difficult challenge through traditional inverse-kinematic techniques. At the end of each arm is a three-fingered hand with an opposable thumb. At the palm of each hand is a simple touch sensor which denotes contact with an object. Additional sensors include 6 encoders on each arm for feedback control and data logging. As previously mentioned, ISAC also has two microphones on either side that can determine the direction of a noise. Finally, ISAC incorporates two pen-cameras that are capable of tracking objects in the environment.

IMA

The underlying control software for ISAC is the Intelligent Machine Architecture, IMA, developed in-house. During the mid-1980s, Behavior-Based robotics was at a peak. However, many researchers were having difficulty developing large, complex systems. At the same time, the field of Computer Science was undergoing the Object-Oriented Programming revolution. These two technologies turned out to be a perfect mate for each other. The principles and technologies of Object Oriented Programming were specifically designed to address the the problems encountered when developing a large, complex, system. As such, some revolutionizing technologies were starting to emerge in order to facilitate development using this new programming model. More specifically, Microsoft's Common Object Model (COM) and Object Linking and Emedding (OLE) tools were designed to address the need for integration and design. Additionally, the Common Object Request Broker Architecture (CORBA) was gaining popularity among the open-source Linux community. Structured programming was also being formalized by Carnegie Mellon in the form of the Capability Maturity Model

(CMM). IMA was designed to incorporate these technologies using Microsoft's Visual Studio and C++ programming languages. IMA supports the creation, communication, and reuse of agents. In this context, the word agent means simply something that acts [26]. In IMA, this can take the form of almost anything from a high-level agent for social interaction to an agent that computes the distance to an object in the environment based on a perceived sound or image. IMA provides a template for the agents the lab designs as well as defining the form and transmission protocol of the data that can be communicated between them. IMA is an ongoing project that continues to evolve and support additional tools for agent design, testing, and development. Additionally, the lab has been exploring ways to cross the operating system divide and incorporate other platforms into the IMA network. This could potentially mean the inclusion of Linux and other Unix-like operating systems that are sometimes more cost-effective in an academic lab environment.

The CIS has been developing a Cognitive Robot Architecture, using IMA that incorporates many of the ideals presented in the previous sections. Agent representations of many of the memory models presented here have been develop as part of their on-going research. **Figure 5** shows a conceptual view of how theses ideas all fit together.

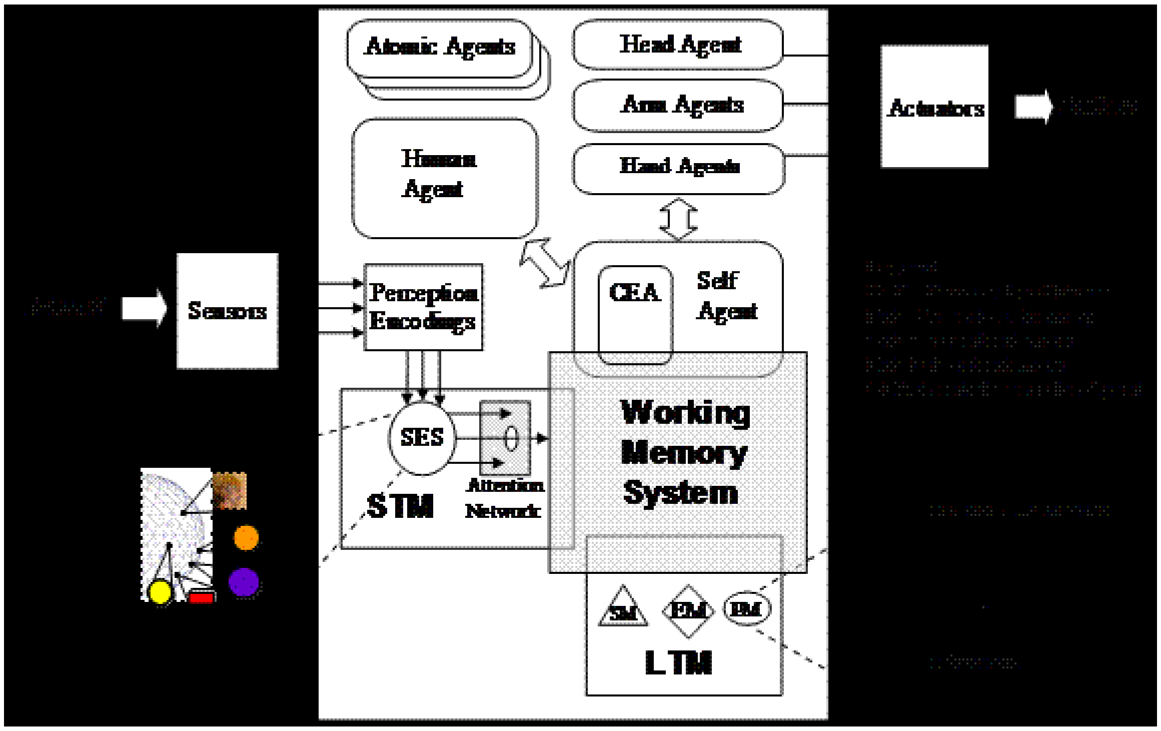

Figure 5: IMA Based Cognitive Architeture for ISAC

CHAPTER III

VERBS AND ADVERBS

In 1998, Doctors Charles Rose, Bobby Bodenheimer, and Michael F. Cohen published a new technique for computer animation called Verbs and Adverbs. In their work, “verbs“ represent the behaviors and have associated “adverbs“. Similar to grammatical adverbs, the adverbs in this technique are used to parameterize a motion. In practice, this algorithm has proved to be a very accurate [8] and fast non-linear interpolation technique. The algorithm, as presented here, has been slightly modified to better suit the nature of the work being conducted in the CIS although the basic components have remained the same. Basically, the technique encodes a set of example motions into a set of linear and non-linear coefficients based on their associated adverbs. In this work the encoding process is split into two steps and the interpolation process is split into an additional two steps. Basic methods and techniques used in the Verbs and Adverbs algorithm can be found in [15], [6], and [20]. **Table 1** defines the symbols used throughout the following sections.

Motion Segmentation and KeyTimes

The first phase of the encoding process centers around identifying a motion’s KeyTimes and segmenting and normalizing the motion about those KeyTimes. A KeyTime, in this context, is defined as a critical structural element in the motion’s trajectory that is common to all motions of that type. In other words, a set of examples for a given motion should all have

Table 1: Important Symbols

Symbol Name	**Description**
NumDOF	Number of Degrees of Freedom in the trajectory
NumAdverbs	Number of Adverbs that parameterize the motion
NumExamples	The number of example motions
NumPoints	Total number of data points in the trajectory
NumKeyTimes	Number of KeyTimes
k	$k \in Z^{nonneg} < NumKeyTimes$, the KeyTime Number
KT	point in the resample trajectory corresponding to a keytime
A	Adverb matrix, independent variable in a least squares fitting
B	Trajectory matrix, dependent variable in a least squares fitting
$\bar{B}$	error matrix after least squares fitting
$\tilde{B}$	least squares approximation of trajectory
x	matrix of linear coefficients
r	matrix of radial-basis function coefficients
D	matrix of gaussians
tt	time index

similar KeyTimes that occur at different instances during the trajectories. While the exact location and magnitude of the KeyTimes may vary, the number and structural significance of the KeyTimes should be the same. A graphical depiction of this relationship is in **Figure 6**.

For the purposes of this work, a technique called Kinematic Centroid Segmentation, from [17] and [18], was used to identify the KeyTimes. This method computes the center of the robotic manipulator's trajectory relative to the base and then finds the KeyTimes by looking for the local maxima. This is a relatively simple process that uses the Davenitt-Hartenburg parameters of the manipulator with the recorded trajectory to compute the end-effector's position in terms of (X, Y, Z). The distance from the home position, (assumed to be the first point in the trajectory), is computed and divided in half, thus identifying the kinematic centroid of the trajectory. This calculation is presented graphically in **Figure 7**. A windowing process is then used to find the local maxima with constraints on t he minimum distance between maxima. The first and last points of the trajectory are automatically denoted as KeyTimes.

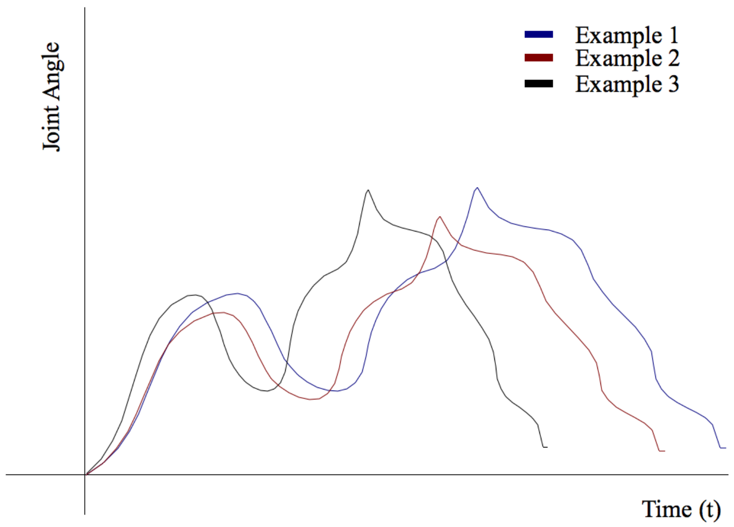

Figure 6: Example Set

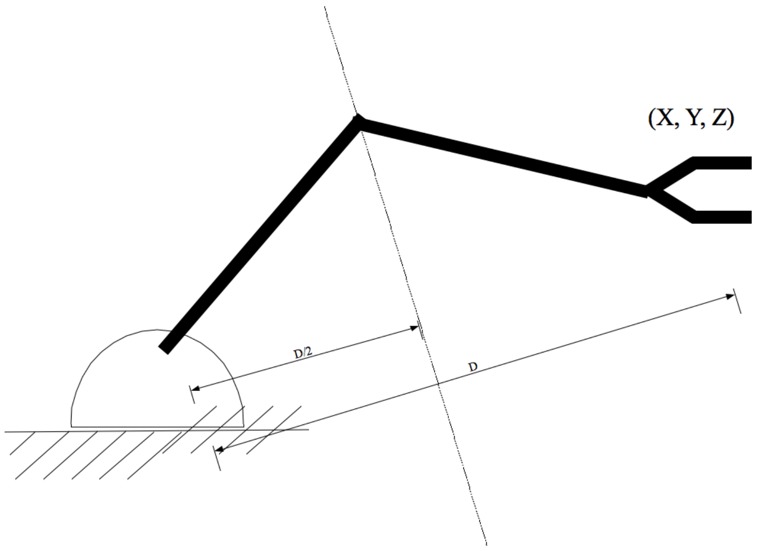

Figure 7: Computation of the Kinematic Centroid

Once the KeyTimes for a motion have been identified, it is necessary to resample the segments bounded by the KeyTimes. The goal is to align the KeyTimes so that they occur at even intervals along the trajectory. If, for example, the motions are resampled to a fixed total length of $NumPoints$ points, and there are five KeyTimes ($NumKeyTimes = 5$), the KeyTimes will be points at 0, $\lfloor \frac{(NumPoints-1)}{NumKeyTimes-1} + .5 \rfloor$, $\lfloor \frac{2*(NumPoints-1)}{NumKeyTimes-1} + .5 \rfloor$, $\lfloor \frac{3*(NumPoints-1)}{NumKeyTimes-1} + .5 \rfloor$, and $(NumPoints - 1)$. This placement is given by Equation 1 where $k \in Z^{nonneg} < NumKeyTimes$. The resampling is computed using a simple point-wise linear interpolation algorithm.

$$KT = \lfloor \frac{k}{NumKeyTimes - 1} \times (NumPoints - 1) + 0.5 \rfloor \tag{1}$$

A side-effect of this process is that some motions in the example set may be discarded because either too many or too few KeyTimes are discovered during this process. The algorithm, as implemented for this work, looks at the number of KeyTimes found in each example motion and keeps the most examples with the highest number of KeyTimes. This is a built-in sanity check that ensures that all the example motions used in future steps are similar in structure.

Encoding of Examples

Once the motions have been pre-processed, the motions are ready for encoding. The first step is to perform a least-squares linear fitting of the data. A least-squares linear fitting typically solves for the slope and intercept of a line that best approximates the general trend of the data. Knowing the trend of the data allows the Verbs and Adverbs algorithm to extrapolate outside of the space defined by the example set. Consider **Figure 8**.

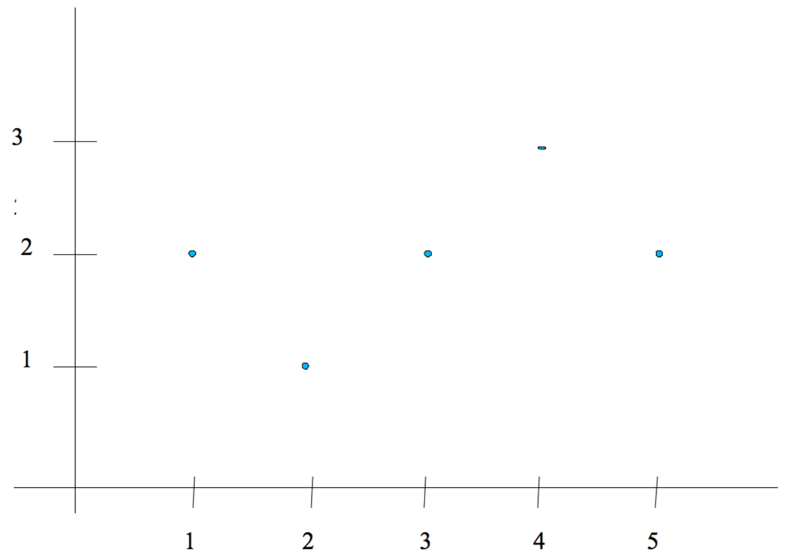

Figure 8: Least Squares Example Data

In order to find a line that approximates this data, we must solve Equation 2 for the coefficient matrix x. In this case, x will be a 2×1 matrix containing the coefficients for x^0 and x^1.

$$Ax = B \tag{2}$$

The matrix, A, becomes the values of the independent variable for the data. Equation 3 shows this matrix.

$$A = \begin{bmatrix} a_0^0 & a_0^1 \\ a_1^0 & a_1^1 \\ a_2^0 & a_2^1 \\ a_3^0 & a_3^1 \\ a_4^0 & a_4^1 \\ a_5^0 & a_5^1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \tag{3}$$

The matrix B becomes the values of the dependent variable as shown in Equation 4.

$$B = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 3 \\ 2 \end{bmatrix} \tag{4}$$

This system can then be solved, in a general sense, for the coefficients through the simple algebraic manipulations of Equations 5 and 6. Equation 7 shows the calulation for the example presented.

$$A^T A x = A^T B \tag{5}$$

$$x = (A^T A)^{-1} A^T B \tag{6}$$

$$x = (\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix})^{(-1)} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 2 \\ 3 \\ 2 \end{bmatrix} \tag{7}$$

The Verbs and Adverbs technique uses the same least squares technique presented except that the linear coefficients must be computed for each set of points in the trajectory. The motions however, are completely defined in the adverb space. In other words, the form of the least squares fitting used in our technique is shown in Equation 8 where $x[tt]$ is a $NumAdverbs + 1 \times NumDOF$ matrix of linear coefficients at index tt, A is a

$NumExamples \times NumAdverbs + 1$ matrix of the adverb values of the example motion, and $B[tt]$ is a $NumExamples \times NumDOF$ matrix of the original trajectory at index tt. An important point here is that the first adverb value for any example motion must be one (1). In other words, the first column of any A matrix is all ones, representing the x^0 term in our previous explanation of a least squares fitting.

$$Ax[tt] = B[tt] \tag{8}$$

Once the linear coefficients have been computed, we can solve Equation 9 for a linear approximation of the original data.

$$Ax[tt] = \tilde{B}[tt] \tag{9}$$

$\tilde{B}[tt]$ denotes an approximation of $B[tt]$. By subtracting this linear approximation, $\tilde{B}[tt]$, from the original trajectory $B[tt]$, as shown in Equation 10, we are left with the residual error of the linear approximation. The linear approximation of the data set in this algorithm is essential to the method's ability to extrapolate beyond the area defined by the example motions.

$$B[tt] - \tilde{B}[tt] = \bar{B}[tt] \tag{10}$$

The next phase of the encoding process will use the error, $\bar{B}[tt]$, to compute the residual non-linearities of the original motion. The residual error will then be approximated using a radial basis function. For simplicity we will use the same notation as [25]. The radial basis function in the encoding process is computed using only the distance, $d_{i1}(p_{i2}) = ||p_{i1} - p_{i2}||$, between a point in the adverb space corresponding to example $i1, p_{i1}$, and the point in adverb space that corresponds to the example motion $i2, p_{i2}$. In this phase of the encoding process, the radial basis function used is a simple gaussian, shown in Equation 11 where $\alpha = 1000$.

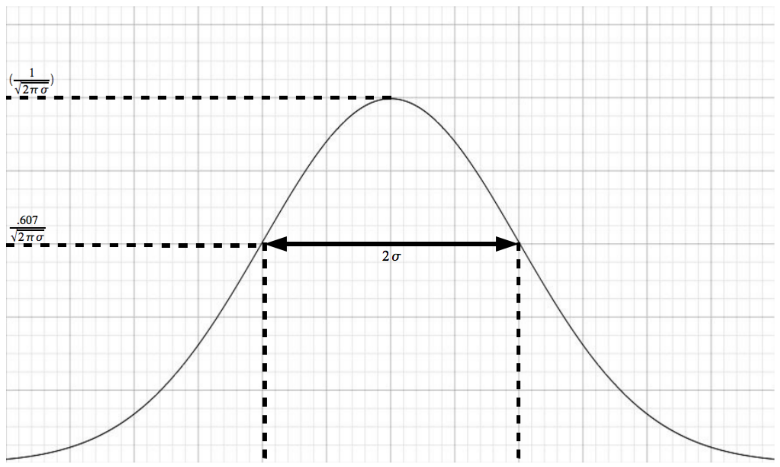

Figure 9: Gaussian Curve

$$D_{i1,i2} = e^{d_{i1}(p_{i2})/\alpha} \tag{11}$$

To solve for the radial basis coefficients we must solve Equation 12 for each time tt.

$$r[tt] = D^{-1}\bar{B}[tt] \tag{12}$$

Interpolation of New Motions

Once the example motions have all been decomposed into the linear and radial basis function coefficients, it is possible to interpolate a new motion by defining a new A and D matrix for the desired motion. Let's walk through a basic example assuming that all the coefficients have already been computed. We will assume that our motion is defined by three adverbs; for now we will denote those values by a_1, a_2, a_3. The new A matrix is simple to create, shown in Equation 13.

$$A = \begin{bmatrix} 1 & a_1 & a_2 & a_3 \end{bmatrix} \tag{13}$$

The D matrix, in our example, is determined by computing the gaussian using the distance between the adverb definition of the new motion, denoted p_{new} and the adverb values of the original examples. We will continue to use the values a_1, a_2, a_3 to denote the new adverbs values. Assuming there were originally 3 example motions, the new D matrix is shown in Equation 14.

$$D = \begin{bmatrix} e^{\frac{d_0(d_{new})}{\alpha}} & e^{\frac{d_1(d_{new})}{\alpha}} & e^{\frac{d_2(d_{new})}{\alpha}} \end{bmatrix} \tag{14}$$

Finally, to interpolate, we solve equation 15 for each index tt using the previously computed coefficient matrices $r[tt]$ and $a[tt]$.

$$B[tt] = Dr[tt] + Aa[tt] \tag{15}$$

Resampling of the New Motion

Once the new interpolated motion has been created it is necessary to resample the motion so that the final motion has the appropriate duration and sample rate for playback. This is done using a similar process as in Section . The resulting motion is basically resampled using a point-wise linear interpolation algorithm that determines the total number of points to be computed based on the duration and sample rate desired by the user.

CHAPTER IV

EXPERIMENTATION AND RESULTS

The experimental data was recorded relative to a point centered on ISAC's right shoulder. A polar coordinate system was used with the angle, measured in degrees, and the height of the group and distance from the body being measured in inches. This setup is depicted in **Figure 10**.

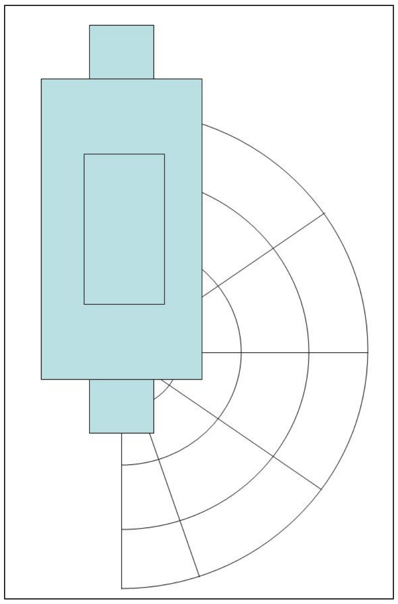

Figure 10: Experimental Setup

Each trial consisted of four motions, and each motion had at least 5 examples. The motions selected for this experiment include: reaching to a point, returning from a point, handshaking, and bicep-curls. These motions were chosen because, with the exception of the bicep-curl, they represent motions that ISAC will probably use with some regularity. The bicep-curl was selected as a counter example. By varying the number of repetitions of the curling motion, it was expected that the Verbs and Adverbs algorithm would fail to accurately interpolate or extrapolate new motions. The angle of each motion was measured using a protractor taped into place, centered about the point of rotation and a string that would extend out over the workspace. There is an estimated 2° error in the angle measurement. The distances and heights were measured with a yard stick and marked on a microphone stand and on the floor with duct-tape. The estimated error is approximated to be the width of a piece of duct-tape, ±1 inch. Data was recorded by manually manipulating the arm through the desired motion while the encoder values were being logged. Section will explain how the generated motions were analyzed.

Data Analysis

The collected data was analyzed using MATLAB. Three .m files were written in an attempt to analyze the various data. The first file computed the kinematic centroid of each example and plotted each of the joint trajectories. The second plotted the corresponding joints from each of the examples and a newly generated motion against each other for comparison. The third computed the RMS error for the generated motions that should match an examplar. In each set of generated motions, at least two of the new motions used the same adverbs as a recorded motion. This was done as a check that the algorithm was indeed accurately encoding and generating the motions.

Reaching

The first motion selected for use in my experiments, was for ISAC's arm to reach from rest to a point in ISAC's workspace. This is probably the most common motion that ISAC will ever perform. It serves as a precursor motion to other motions like grasping, pointing, etc. ISAC's arm was manually manipulated to six (6) points in its workspace while encoder data was recorded. The adverbs were the angle relative to ISAC's shoulder (degrees), the distance (inches) from ISAC's torso, and the height of the point relative to the floor (inches). **Figures 11** through **16** show the joint angles for the first set of recorded motions.

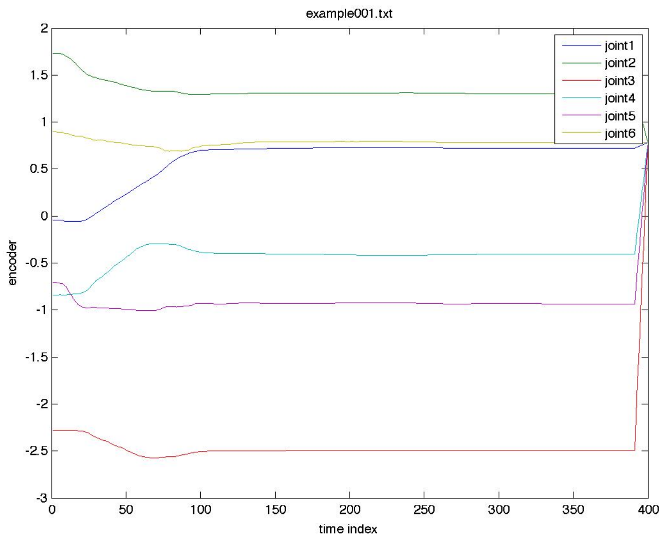

Figure 11: Reaching 1

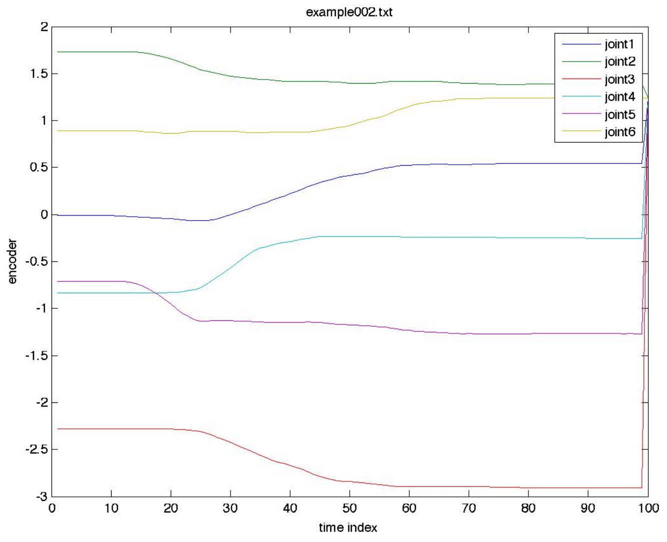

Figure 12: Reaching 2

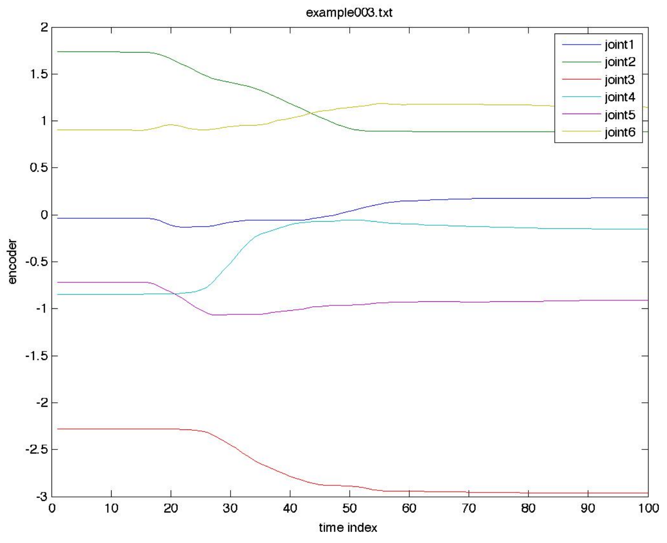

Figure 13: Reaching 3

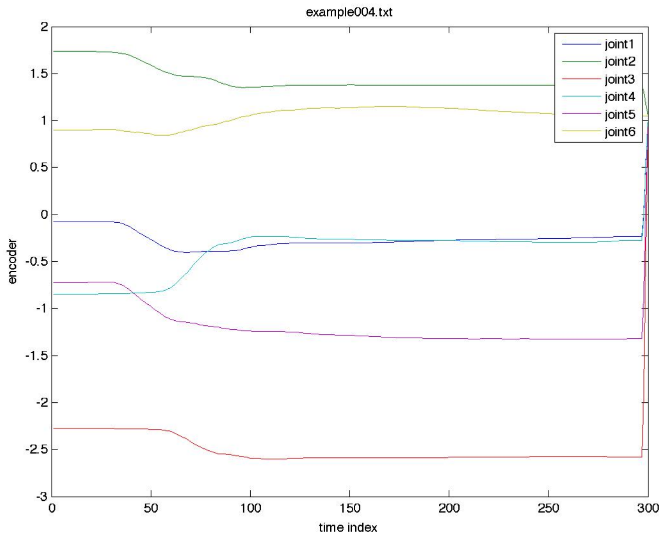

Figure 14: Reaching 4

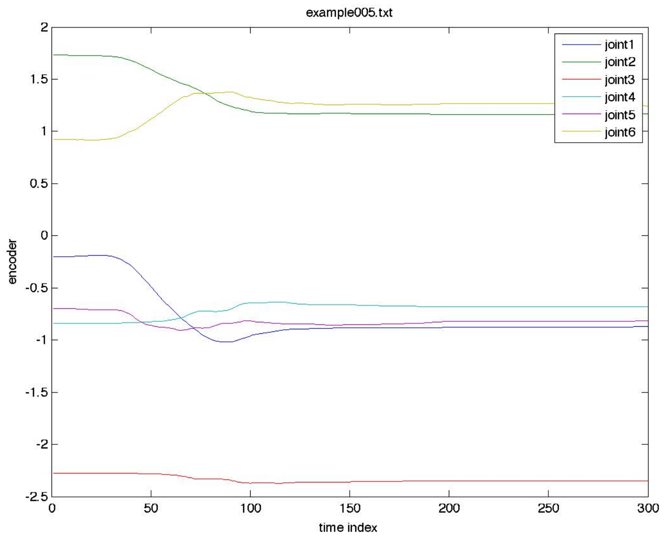

Figure 15: Reaching 5

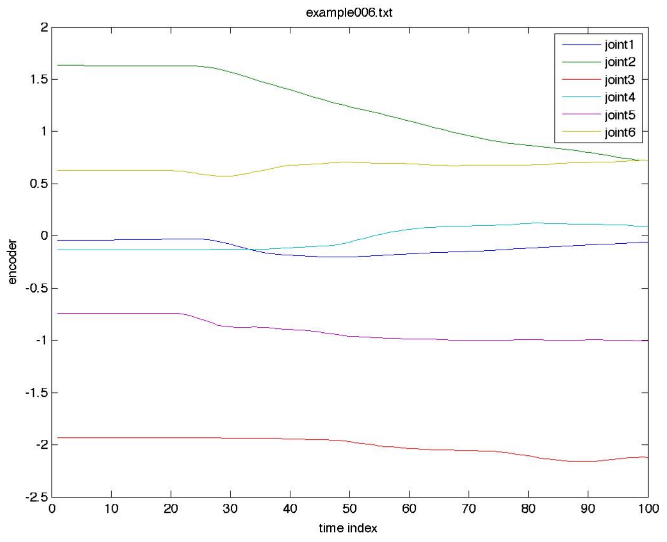

Figure 16: Reaching 6

Out of these recorded motions, three examples were selected. As mentioned in Section , the other motions were discarded because of a lack of similarity. **Figures 17** through **19** shows the plot of the centroids of the selected motions. **Figures 20** through **22** shows the centroid of the motions that were not selected.

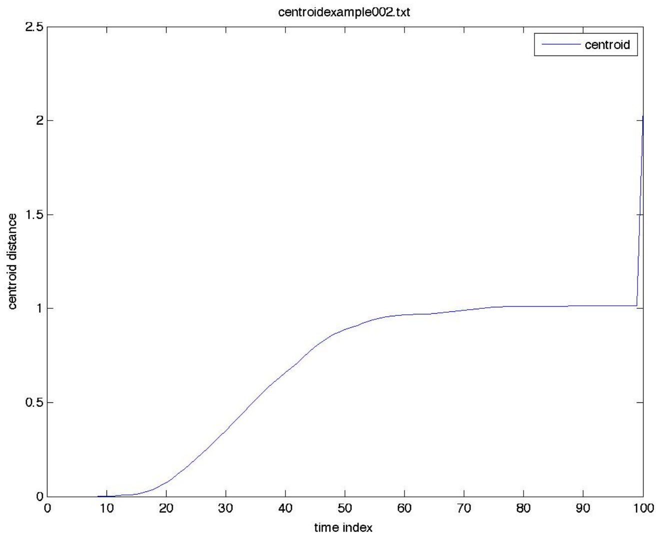

Figure 17: Reaching Example 2

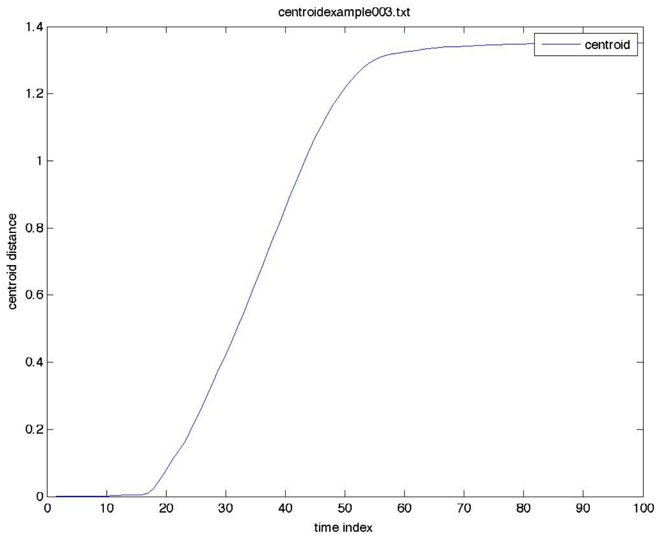

Figure 18: Reaching Example 3

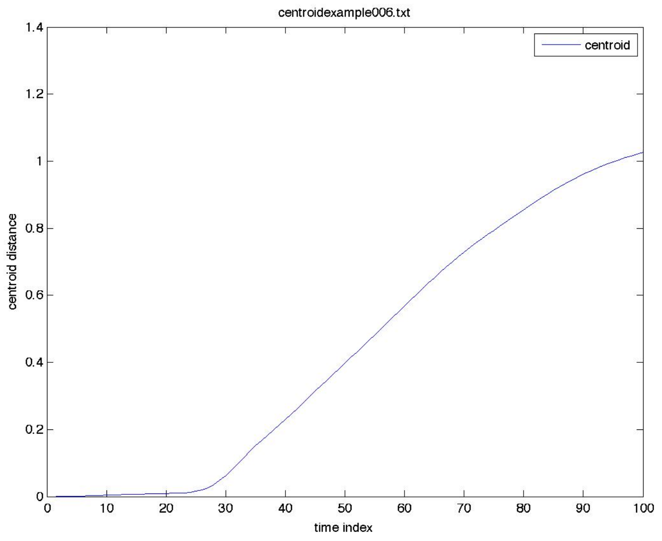

Figure 19: Reaching Example 6

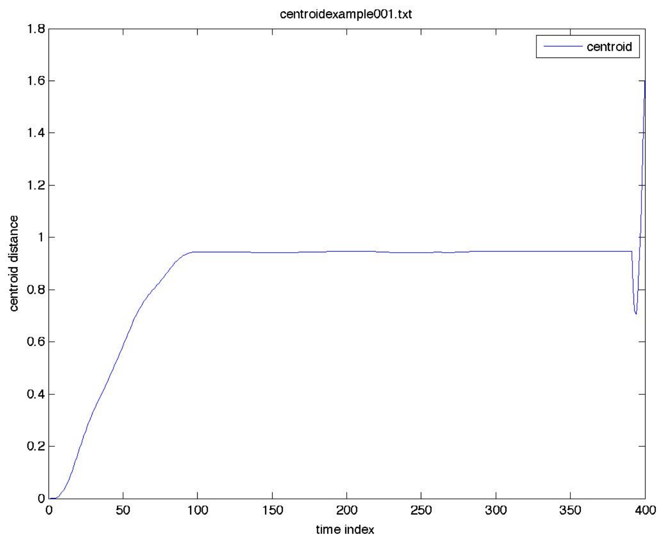

Figure 20: Reaching Example 1

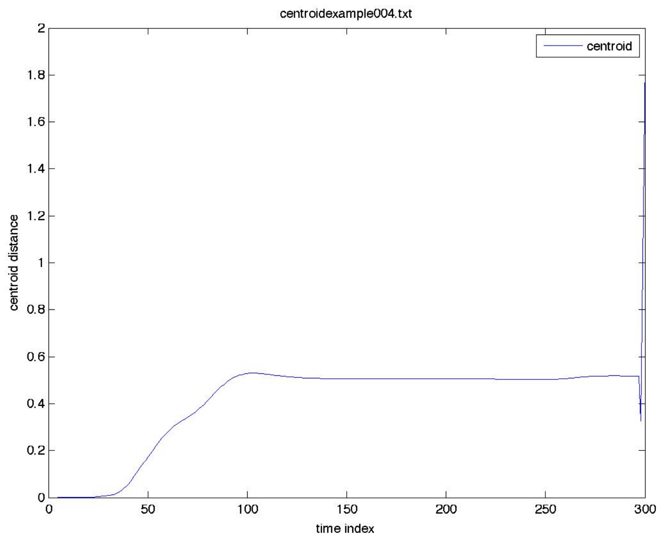

Figure 21: Reaching Example 4

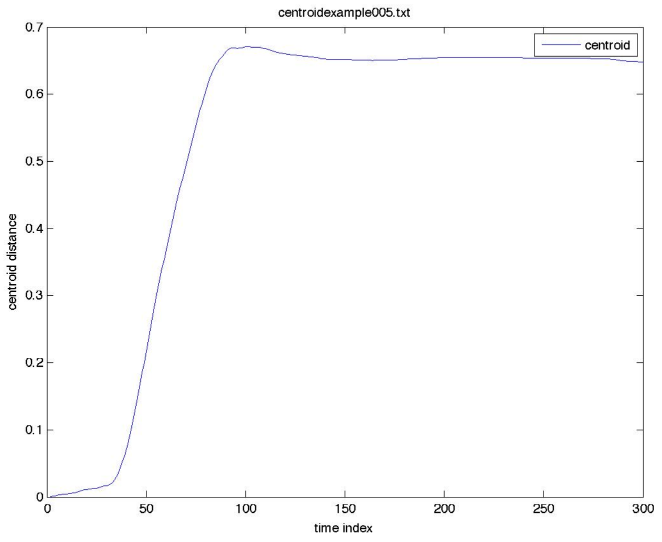

Figure 22: Reaching Example 5

In this particular case, the centroid plots aren't very revealing in terms of why three of the motions were not selected. While visual observation seems to suggest that these motions are all very similar in structure, it is important to realize that the rigid nature of the selection and segmentation process requires only minor deviations in structure to eliminate a particular motion in favor of another. More obvious differences will be shown in future examples.

Reaching, Results

After the motions were segmented, resampled, and selected, they were encoded into the coefficient matrices mentioned in Section . New motions were generated using the values tabulated in Appendix B. The motions that were designed to match examples were analyzed by computing the Root-Mean-Square (RMS) Error between the generated motion and the originally recorded motion. The equation used for computing this error is given in Equation 16.

$$RMS = \sqrt{\frac{1}{N}\sum_{n=0}^{N} f[n]^2} \tag{16}$$

Where $f[n]$ is simply a point-wise difference between the joint trajectories being compared. For example, if $x[n]$ is equal to the trajectory of the first joint of motion 1, and $y[n]$ is equal to the trajectory of the first joint of motion 2, then $f[n] = x[n] - y[n]$. In this set of examples and generated motions, there were two motions generated that should have matched encoded examples. In fact, the plots lined up so well that the matching example motions are not visible upon inspection. **Figures 23** through **28** show one set of these plots. Additionally, **Table 2** shows the computed RMS values for all the generated motions and the corresponding example. Because the plots of the generated motions are so difficult to see, they will be omitted in future sections.

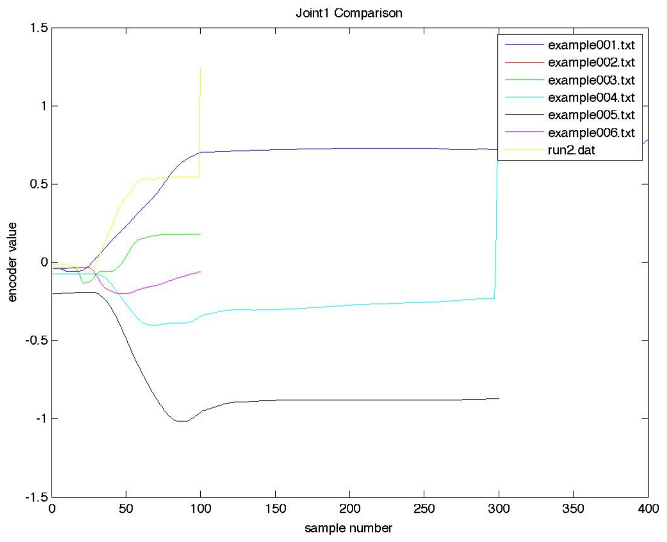

Figure 23: Reaching, Generated Motion 2, Joint 1

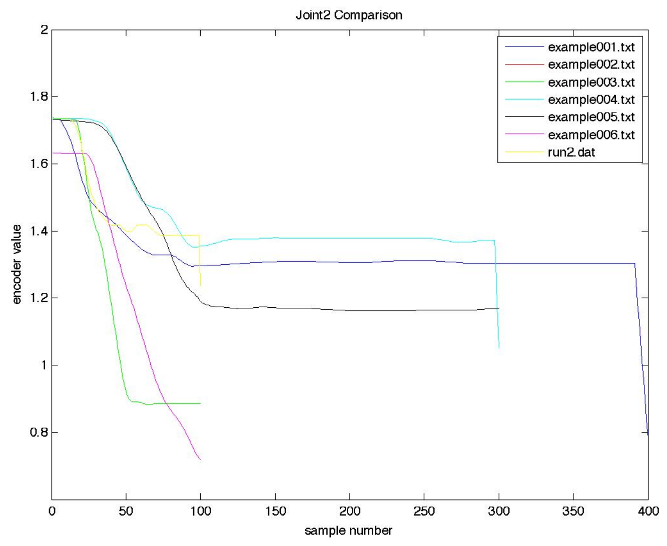

Figure 24: Reaching, Generated Motion 2, Joint 2

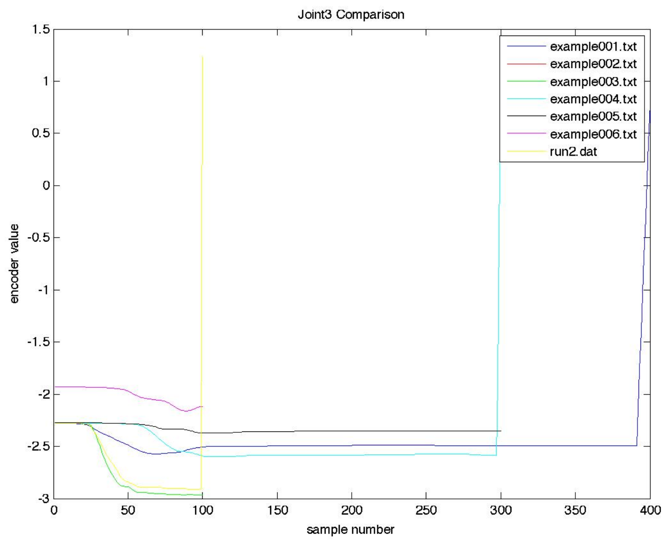

Figure 25: Reaching, Generated Motion 2, Joint 3

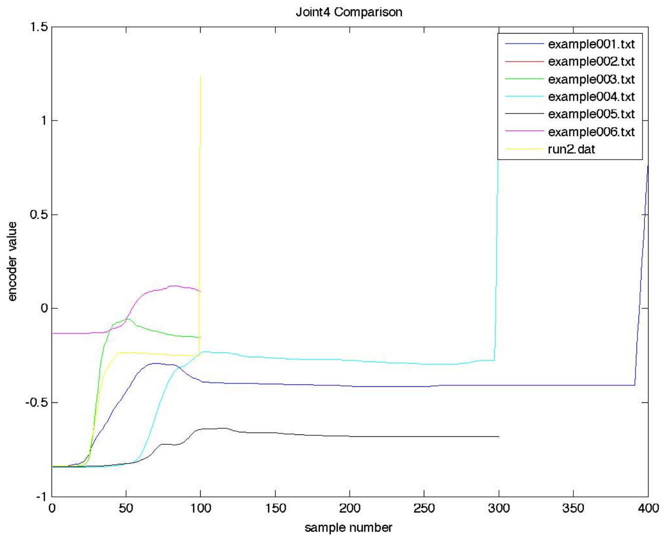

Figure 26: Reaching, Generated Motion 2, Joint 4

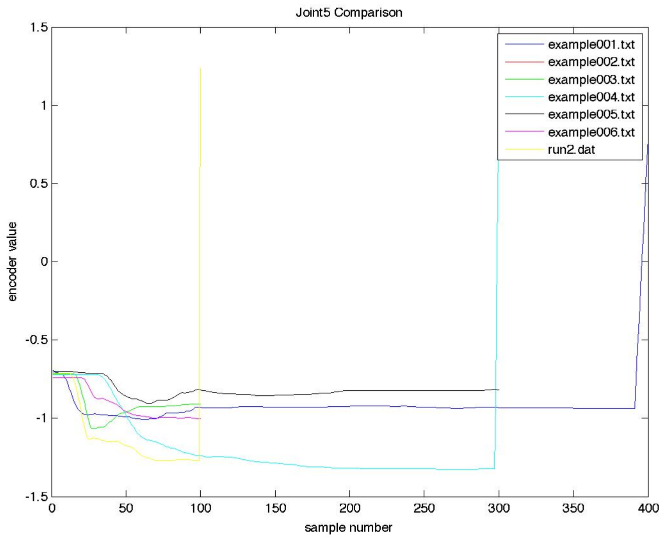

Figure 27: Reaching, Generated Motion 2, Joint 5

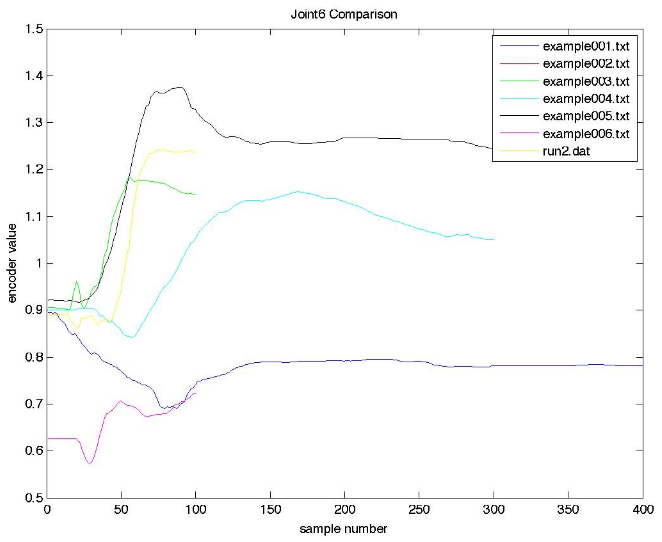

Figure 28: Reaching, Generated Motion 2, Joint 6

Table 2: Reaching, RMS Values

RMS Values for Generated Motions and Corresponding Examples							
Example	Gen. Motion	J1 RMS	J2 RMS	J3 RMS	J4 RMS	J5 RMS	J6 RMS
6	3	$5.58e-6$	$2.721e-5$	$3.789e-5$	$7.10e-6$	$1.994e-5$	$6.65e-6$
2	2	$4.22e-6$	$2.853e-5$	$3.937e-5$	$5.40e-6$	$1.991e-5$	$5.93e-6$

Handshake

The second motion selected for ISAC in this experiment was Handshaking. This is an interesting example because the ”shaking” motion introduces some interesting structure to the example motions. It is also often used in lab demos as part of ISAC’s routines for human interaction. As in the other motions, ISAC’s arm was manually manipulated to a point in ISAC’s workspace with a fixed height, at a fixed distance, but varying angle relative to ISAC’s shoulder. The hand was “pumped“ three times at each point with a fixed amplitude. The adverb was the angle to the handshake point relative to ISAC’s shoulder. **Figures 29** through **33** shows the joint trajectories for the motions used. Only five (5) examples were recorded for this behavior.

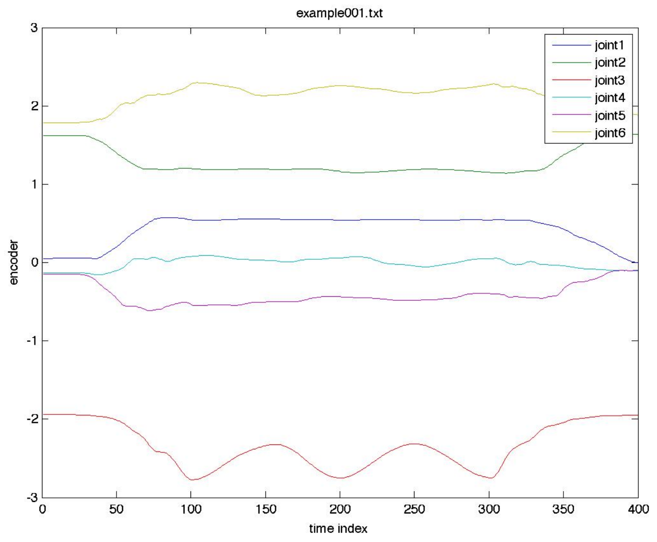

Figure 29: Handshaking 1

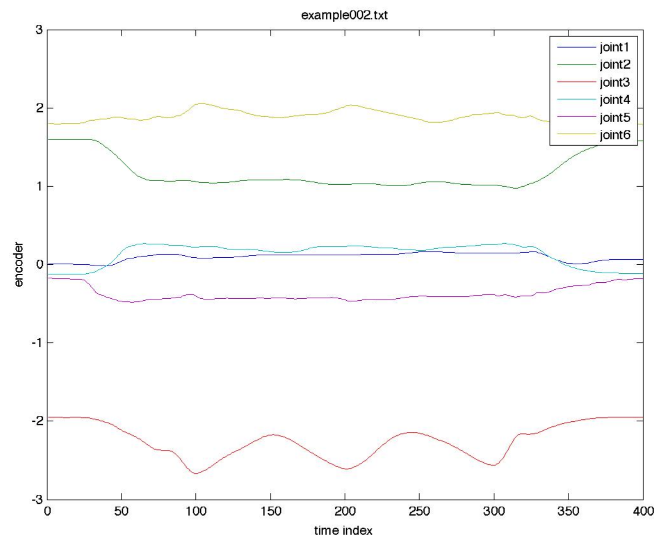

Figure 30: Handshaking 2

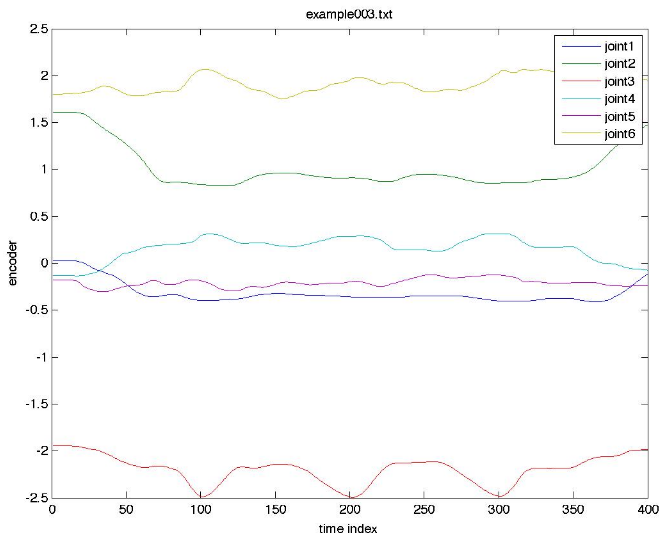

Figure 31: Handshaking 3

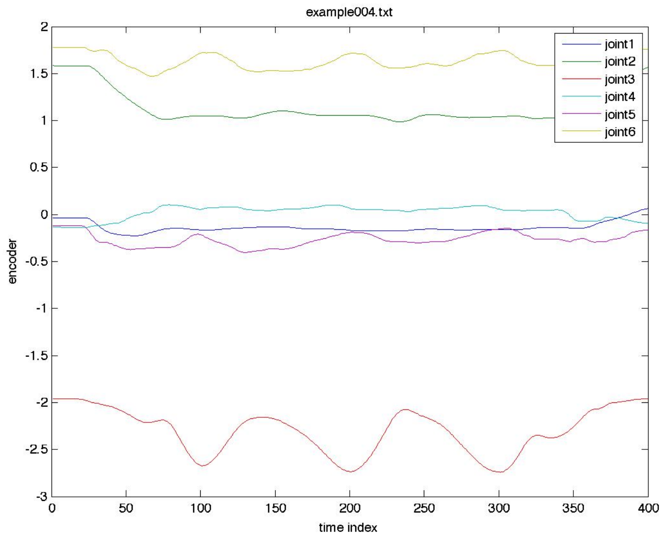

Figure 32: Handshaking 4

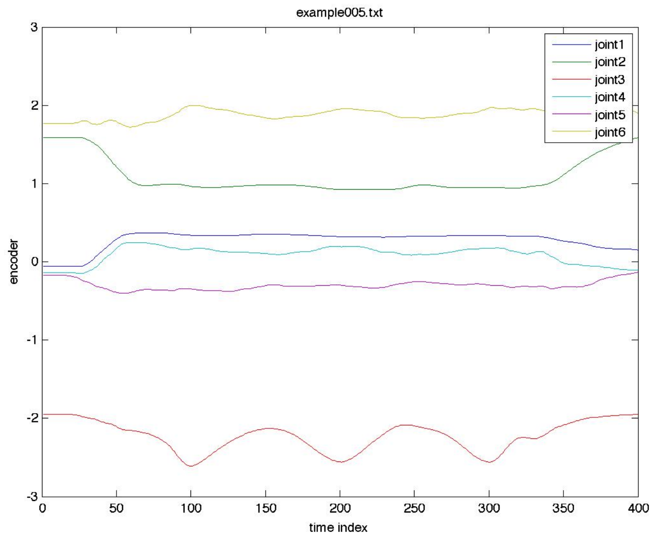

Figure 33: Handshaking 5

The centroid plots of these motions are of particular interest as they directly affect the output of the segmentation and selection process. These plots are shown in **Figures 34** through **38**.

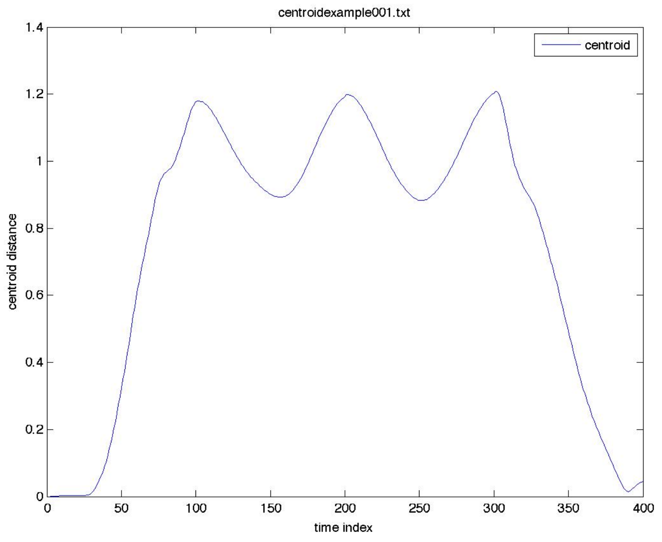

Figure 34: Handshaking Example 1

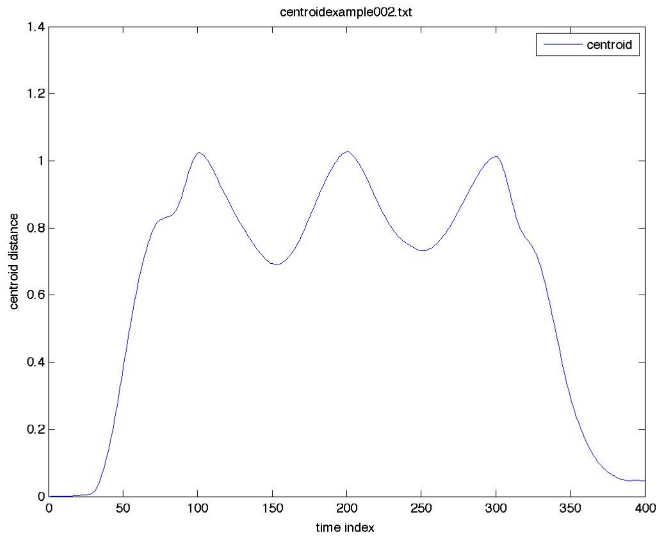

Figure 35: Handshaking Example 2

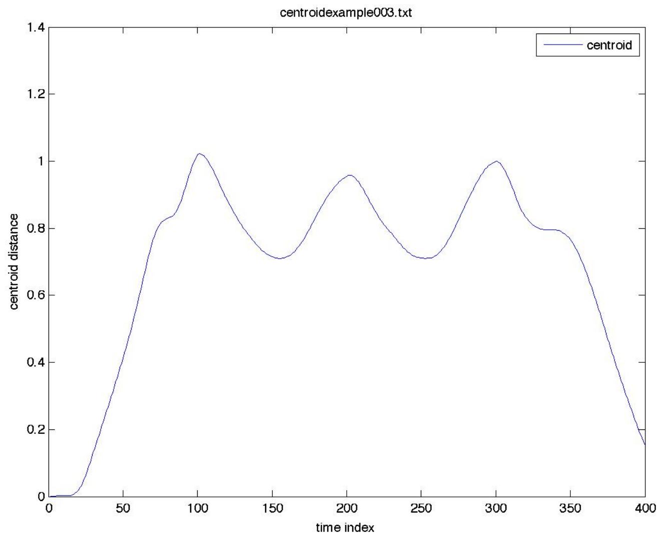

Figure 36: Handshaking Example 3

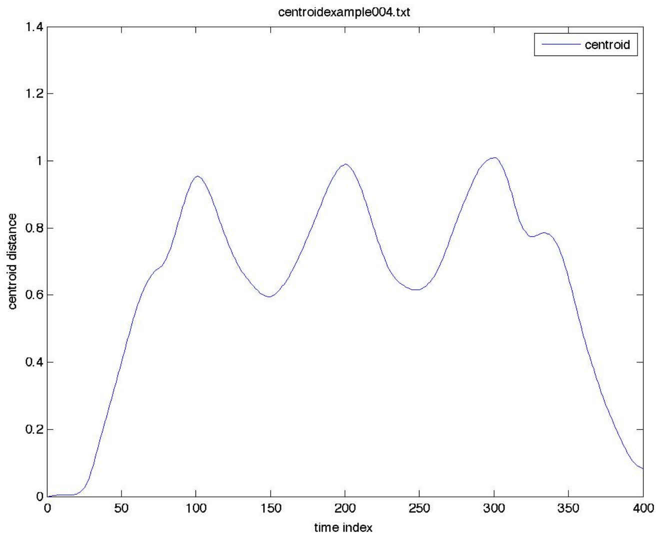

Figure 37: Handshaking Example 4

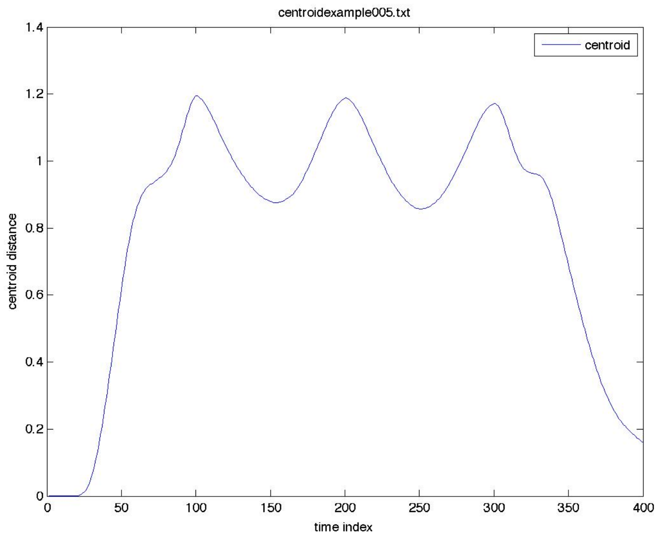

Figure 38: Handshaking Example 5

In this case, the centroid plots all have very similar. All five motions were encoded in this trial.

Handshaking, Results

As mentioned previously, these motions were analyzed, segmented, resampled, and encoded. Five motions were generated from these examples. Two of the generated motions match the examples. The adverb values used for generating these motions can be found in Appendix B. The RMS values indicating the error between the generated motions and their corresponding example are found in **Table 3**. The plots of the generated motions are not useful for inspection and are therefore not included.

Table 3: Handshaking, RMS Values

RMS Values for Generated Motions and Corresponding Examples							
Example	Gen. Motion	J1 RMS	J2 RMS	J3 RMS	J4 RMS	J5 RMS	J6 RMS
1	1	$3.761e-5$	$4.044e-5$	$2.953e-5$	$3.460e-5$	$2.843e-5$	$4.332e-5$
5	3	$3.836e-5$	$4.138e-5$	$2.997e-5$	$3.529e-5$	$2.901e-5$	$4.411e-5$

Return

The third motion chosen was to return from a point in ISAC's workspace back to rest. As usual, ISAC's arm was manually manipulated while encoder data was recorded. The starting points for these examples were designated by the angle of the point (degrees) relative to ISAC's shoulder, the distance (inches) relative to the torso, and the height (inches) relative to the floor. The arm was returned, as close as possible, to the same resting point for each example. There were six example motions recorded. **Figures 39** through **44**.

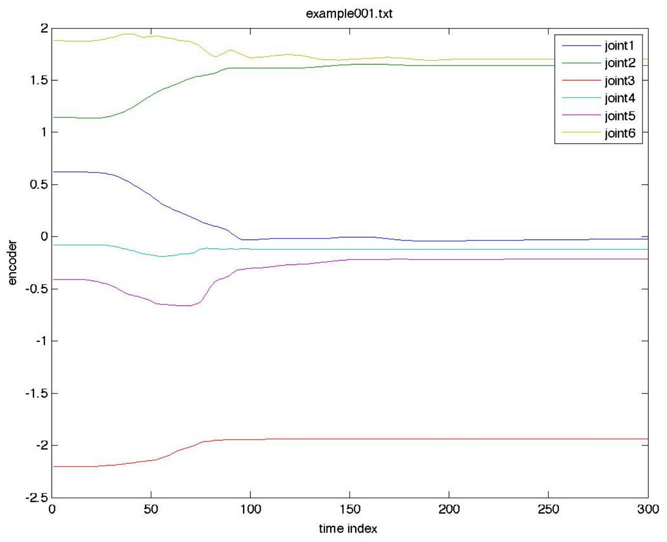

Figure 39: Return 1

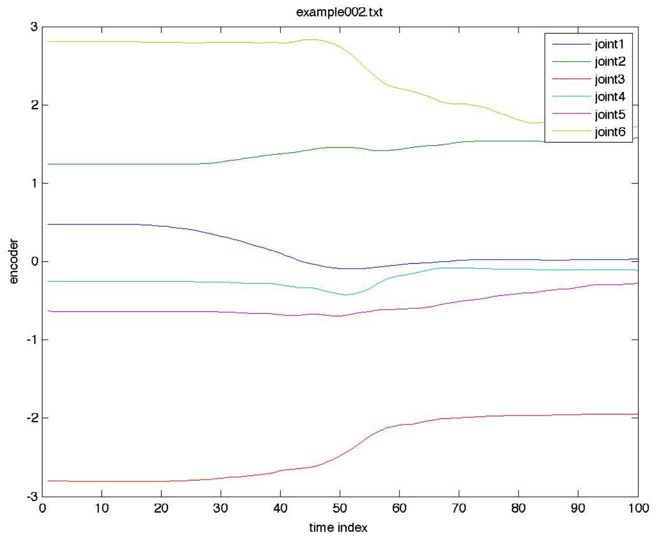

Figure 40: Return 2

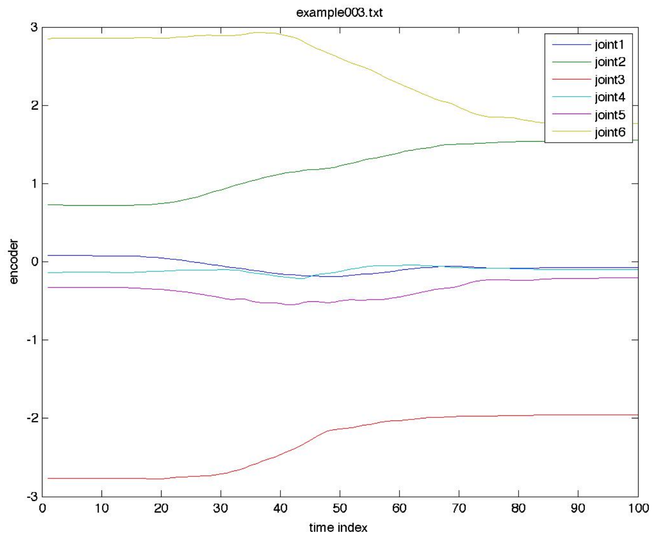

Figure 41: Return 3

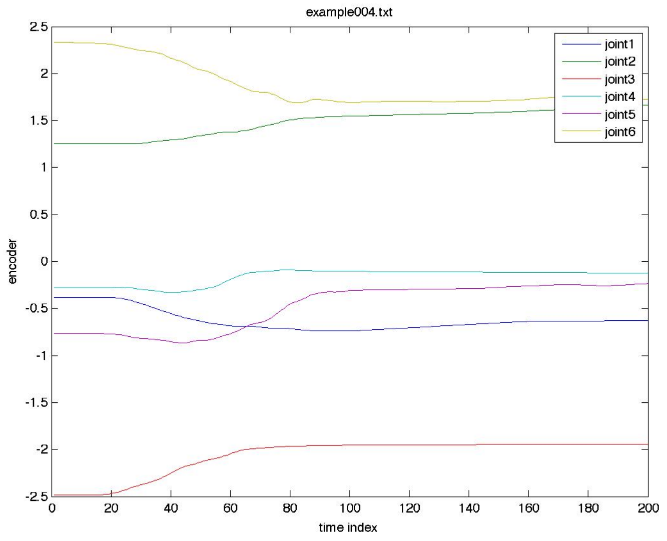

Figure 42: Return 4

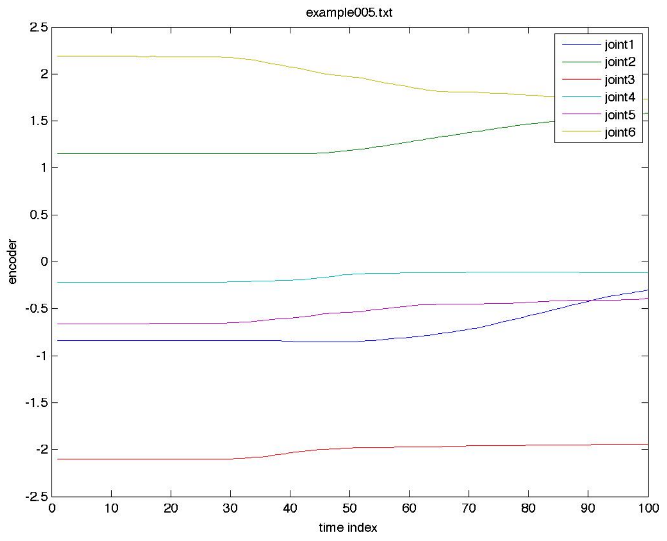

Figure 43: Return 5

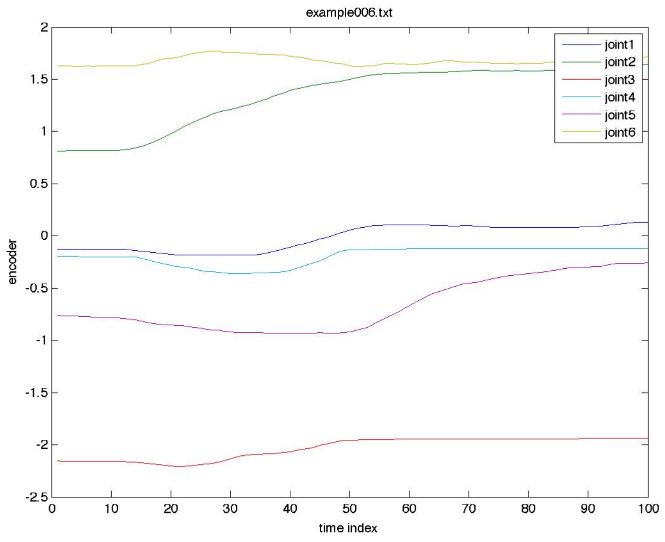

Figure 44: Return 6

For this set of example motions, four out of the six recorded examples were encoded. Unfortunately, the plots of the centroids of these examples don't give much insight as to why some were selected while others were discarded. The bicep-curl motion presented next will provide better insight into this phenomenon.

Return, Results

These results were analyzed in the same fashion as the others. **Table 4** shows the RMS values of the generated motions and their corresponding examples.

Table 4: Return, RMS Values

RMS Values for Generated Motions and Corresponding Examples							
Example	Gen. Motion	J1 RMS	J2 RMS	J3 RMS	J4 RMS	J5 RMS	J6 RMS
2	2	$2.833e-5$	$3.0e-5$	$2.236e-5$	$1.585e-5$	$2.966e-5$	$2.646e-5$
6	3	$3.392e-5$	$2.419e-5$	$2.449e-5$	$1.591e-5$	$2.926e-5$	$2.449e-5$

Bicep-Curl

The final motion selected for this experiment was the bicep curl. This motion provides a great example of how the algorithm analyzes and segments the motion data. For this motion, the arm was manually manipulated to a fixed point in ISAC's workspace and "curled" a variable number of times. The magnitude/amplitude of the curling motion was controlled by moving the hand to a fixed point above the end position. The adverb was the number of times that the arm was curled. This motion was specifically chosen because it should cause the algorithm to fail. In other words, the goal was specifically to vary the structure of the motion so that the algorithm would throw out all examples except for one. The hypothesis was that if only one motion is encoded, the algorithm should not have enough data to generate any new motions that even closely resemble the examples. **Figures 45** through **49** shows the centroids of the different example motions. Note the clear structural differences. Each of the “bumps“ in the plots represents one ”curl” of the motion. Since the segmentation and resampling portion of this approach rely on local maximums, it is apparent that only one motion will be selected.

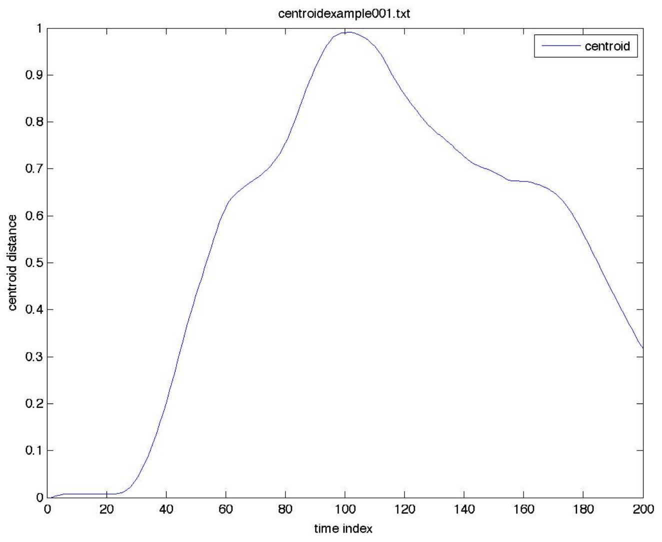

Figure 45: Bicep Curl Example 1

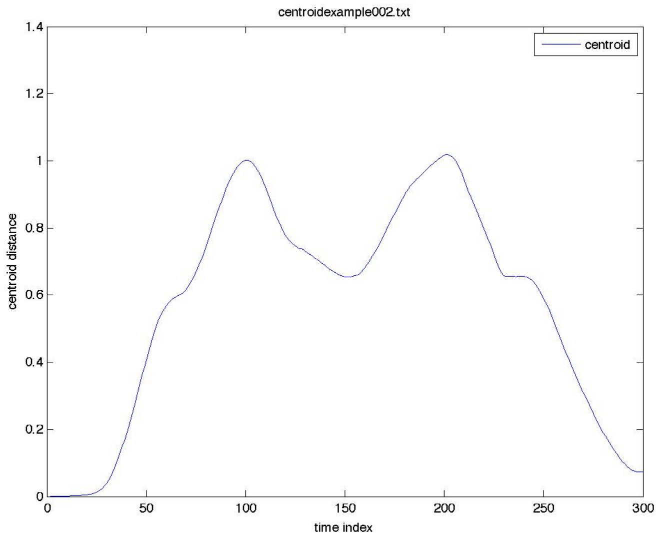

Figure 46: Bicep Curl Example 2

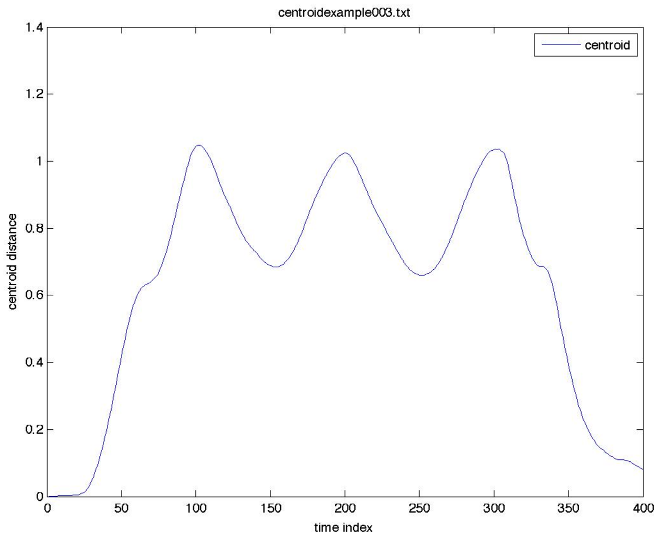

Figure 47: Bicep Curl Example 3

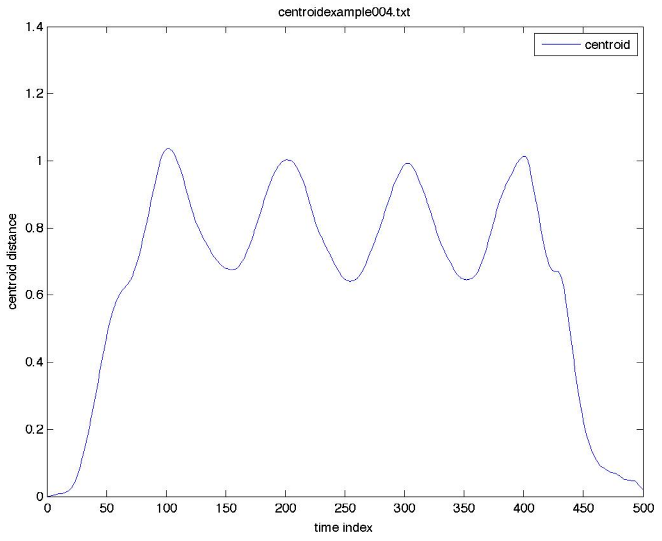

Figure 48: Bicep Curl Example 4

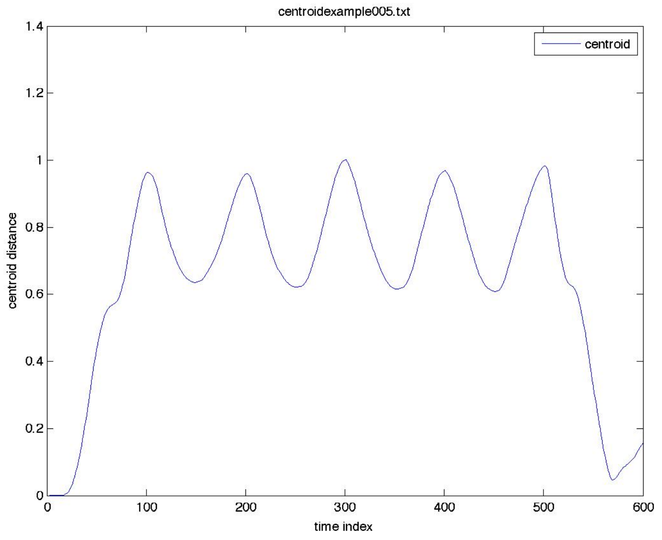

Figure 49: Bicep Curl Example 5

Bicep-Curl, Results

As mentioned previously, the theory was that because only one motion is selected in this example set, the Verbs and Adverbs algorithm cannot encode enough information to accurately represent the structure of the input motion. This seemed to hold true. **Figures 50** through **55**.

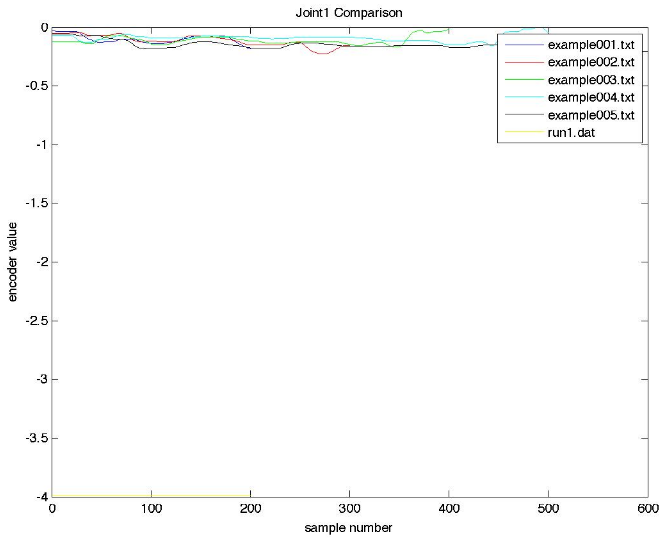

Figure 50: Bicep Curl, Generated Motion 1, Joint 1

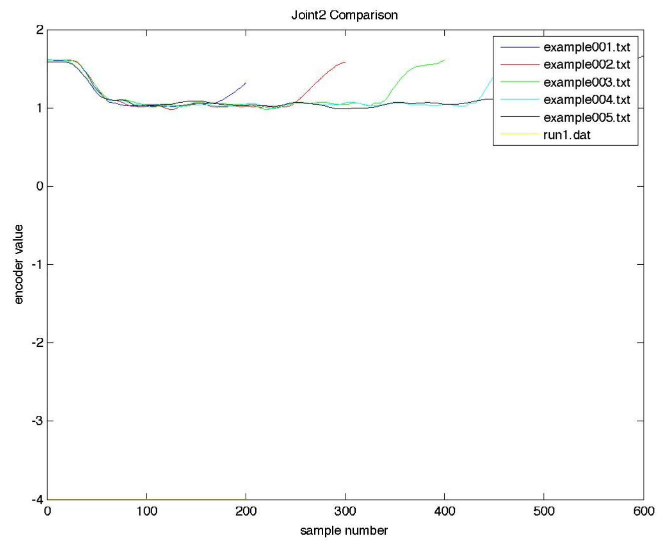

Figure 51: Bicep Curl, Generated Motion 1, Joint 2

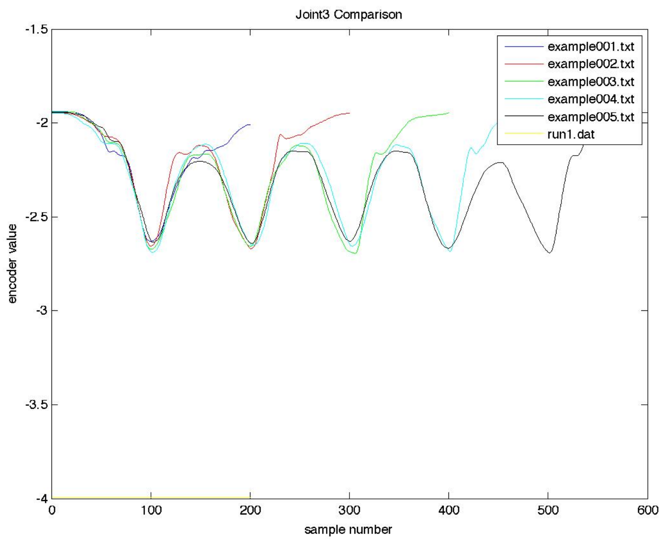

Figure 52: Bicep Curl, Generation Motion 1, Joint 3

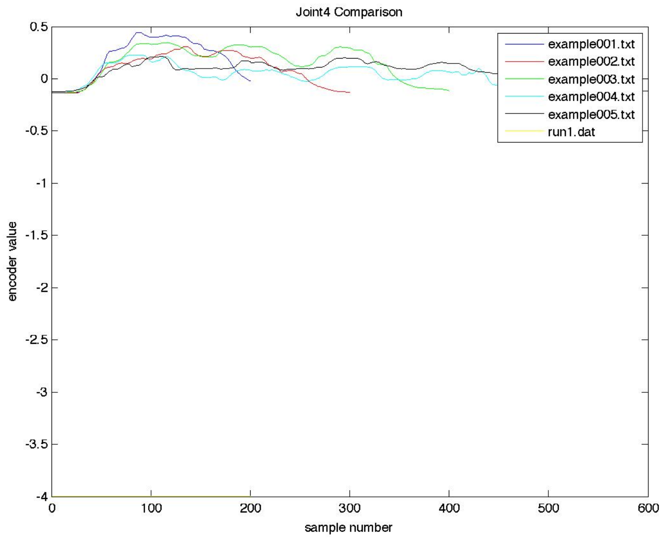

Figure 53: Bicep Curl, Generated Motion 1, Joint 4

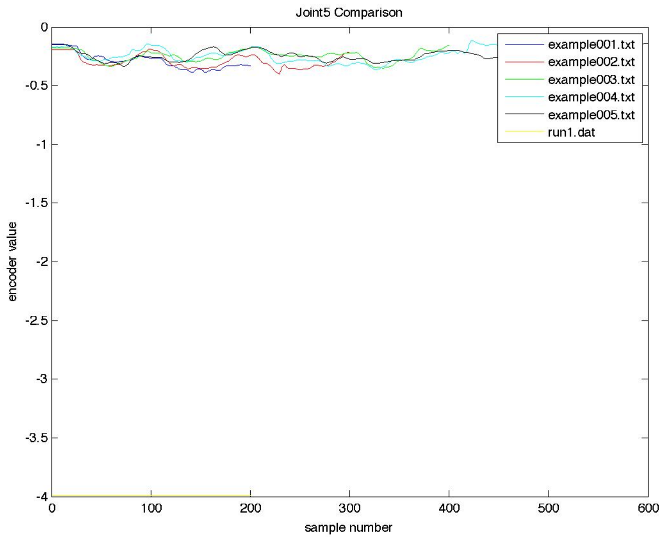

Figure 54: Bicep Curl, Generated Motion 1, Joint 5

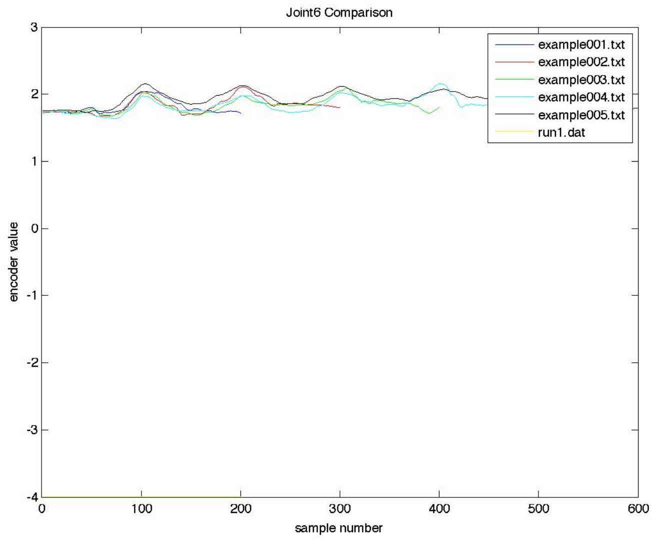

Figure 55: Bicep Curl, Generated Motion 1, Joint 6

As you can see, the resulting motion from interpolation does not match the input motions very well at all. In fact, unlike the previous examples, the RMS Error for this motion was considerably worse. **Table 5** shows the RMS Error for the generated motion and the corresponding example motion.

Table 5: Bicep-Curl, RMS Values

RMS Values for Generated Motions and Corresponding Examples							
Example	Gen. Motion	J1 RMS	J2 RMS	J3 RMS	J4 RMS	J5 RMS	J6 RMS
1	1	3.8951	5.1874	1.8225	4.1980	3.7067	5.8019

CHAPTER V

CONCLUSIONS

When appropriate inputs are given to the Verbs and Adverbs algorithm, it is capable of recreating the examples surprisingly well. It can also generate new motions, within reasonable limits, that accurately represent the structure of the input motions. The algorithm does, however, have limitations. Recall Equation 15, shown again in Equation 17. Since the D and r matrices are dependent on $NumExamples$ and tt, and the A and a matrices are dependent on $NumExamples$, $NumAdverbs + 1$, and tt, the complexity of the algorithm is given in Equation 19 and Equation 21. In other words, the ability of this algorithm to quickly produce results is inversely proportional to the number of examples, the number of adverbs, and the number of samples (which is dependent on duration and sample rate).

$$B[tt] = Dr[tt] + Aa[tt] \tag{17}$$

$$NumAdverbs + 1 > NumExamples \tag{18}$$

$$O(tt \times NumAdverbs + 1 \times NumDOF) \tag{19}$$

$$NumAdverbs + 1 \leq NumExamples \tag{20}$$

$$O(tt \times NumExamples \times NumDOF) \tag{21}$$

For most motions this algorithm will be more than adequate. Some preliminary timing was done on the encoding and decoding time of the algorithm on a G4 processor running OSX (v10.4) with 256 MB RAM. **Table 6** shows these results. These numbers are anecdotal only and should be used only for a rough order of magnitude timing estimates. According to the “time“ utility manpage, and the POSIX.2 standard (IEEE Std. 1003.2-1992), system time is defined as the amount of time consumed by system overhead. User time is defined as the amount of time used to execute the application to the standard error stream. Real time is defined as the total elapsed time. Times reported for system and user time can have a ±1 second resolution.

Table 6: Timing Information

	Encode Time			Interpolation Time		
Trial/Motion	User Time (s)	System Time (s)	Real Time (s)	User Time (s)	System Time (s)	Real Time (s)
Trial 1, Bicep-Curl	.799	.060	1.023	.102	.221	.03
Trial 1, Handshake	1.253	.136	2.486	.492	.04	.844
Trial 1, Reaching	.567	.054	.883	.176	.032	.528
Trial 1, Return	.474	.05	.65	.19	.032	.533

There are a couple other minor issues regarding this implementation of the Verbs and Adverbs algorithm on ISAC. As mentioned previously, the segmentation method is extremely susceptible to even minor fluctuations in the kinematic centroid trajectory that result in the detection of a local maxima. This is in part, due to an arbitrarily sized windowing process for finding the local maxima. A technique that could be used to “filter“ some of these maxima would be to examine the magnitude of a potential local maxima in terms of relative percent increase. However, this process would also be somewhat subjective and would have limited effectiveness in all situations.

Additionally, manually manipulating the arm through a motion is prone to human error. I have not yet determined a mitigating approach to this problem. One possible solution would be to compute a trajectory in advance and use that to “feed“ the algorithm. However, this approach seems to defeat the purpose of having a motion generation algorithm in the first place. Controls need to be put into place in order to more accurately and repeatably record motions. Overall, the Verbs and Adverbs algorithm, implemented on a humanoid robot performs extremely well.

APPENDIX A

RECORDED DATA

Reaching			
Example	Adverb 1 (range in.)	Adverb 2 (elevation in.)	Adverb 3 (azimuth deg.)
Ex1	9	30	65
Ex2	4	40	69
Ex3	10	50	90
Ex4	5	35	121
Ex5	0	30	159
Ex6	17	40	114

Bicep Curl	
Example	Adverb 1 (reps)
Ex1	1
Ex2	2
Ex3	3
Ex4	4
Ex5	5

Handshake	
Example	Adverb 1 (azimuth deg.)
Ex1	60
Ex2	90
Ex3	120
Ex4	115
Ex5	80

Return			
Example	Adverb 1 (range in.)	Adverb 2 (elevation in.)	Adverb 3 (azimuth deg.)
Ex1	9	30	65
Ex2	4	40	69
Ex3	10	50	90
Ex4	5	35	121
Ex5	0	30	159
Ex6	17	40	114

APPENDIX B

INTERPOLATION ADVERB VALUES

Reaching			
Example	Adverb 1 (range in.)	Adverb 2 (elevation in.)	Adverb 3 (azimuth deg.)
Ex1	9	30	65
Ex2	4	40	69
Ex3	17	40	114
Ex4	7	45	80
Ex5	12	30	110

Bicep Curl	
Example	Adverb 1 (reps)
Ex1	1
Ex2	2
Ex3	3
Ex4	4
Ex5	10

Handshake	
Example	Adverb 1 (azimuth deg.)
Ex1	60
Ex2	120
Ex3	80
Ex4	107
Ex5	160

Return			
Example	Adverb 1 (range in.)	Adverb 2 (elevation in.)	Adverb 3 (azimuth deg.)
Ex1	9	30	65
Ex2	10	50	90
Ex3	70	40	114
Ex4	12	45	110
Ex5	7	30	80

BIBLIOGRAPHY

[1] J. A Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):473 – 509, 1991.

[2] Ronald C. Arkin. *Behavior Based Robotics*. The MIT Press, 1999.

[3] Isaac Asimov. *I, Robot*. Bantam Spectra, 1991.

[4] R. C. Atkinson and R. M. Shriffin. Human memory: A proposed system and its control processes. *Psychology of Learning and Motivation*, 2:89 – 195, 1968.

[5] A. Baddely. Working memory. *Oxford Psychology Series*, 11, 1986.

[6] Otto Bretscher. *Linear Algebra with Applications*. Prentice Hall, 1997.

[7] Rodney A Brooks. Intelligence without representation. *Aritificial Intelligence*, 47:139–159, 1991.

[8] Christina L. Campbell. Learning Through Teleoperation on Robonaut. MS thesis, Vanderbilt University, 2003.

[9] N. Cowan. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 1(24):87 – 185, 2001.

[10] John J. Craig. *Introduction to Robotics Mechanics and Control*. Addison-Wesley Publishing Company, Inc., 1989.

[11] Will Dodd and Ridelto Gutierrez. The role of episodic memory and emotion in a cognitive robot. ROMAN, 2005.

[12] M. S. Gazzania, R. B. Ivry, and G. R. Mangum. *Cognitive Neuroscience: The Biology of the Mind*. New York: Norton, 2002.

[13] Stephen Gordon and Joe Hall. System integration with working memory management for robotics behavior learning. Submitted to ICDL, 2006.

[14] Dudek Gregory and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.

[15] R. W. Hamming. *Numerical Methods for Scientists and Engineers*. Dover Publications, Inc., 1973.

[16] John Iovine. *Robots, Androids, and Animatrons*. McGraw-Hill, 1998.

[17] Odest Chadwicke Jenkins. *Data-Driven Derivation of Skills for Autonomous Humanoid Agents*. PhD thesis, University of Southern California, 2003.

[18] Odest Chadwicke Jenkins and Maja J. Mataric. Automated Derivation of Behavior Vocabularies for Autonomous Humanoid Motion. Submitted to Autonomous Agents and Multi Agent Systems, 2003.

[19] K. Kawamura, W. Dodd, P. Ratanaswasd, and R.A. Gutierrez. Development of a robot with a sense of self. CIRA Conference, 2005.

[20] Clare D. McGillem and George R. Cooper. *Probabilistic Methods of Signal and System Analysis*. Oxford University Press, 1999.

[21] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81 – 97, 1956.

[22] Palis Ratanaswasd. *Design and Implementation of Cognitive Control in a Humanoid Robot*. PhD thesis, Vanderbilt University, 2006.

[23] Palis Ratanaswasd, Will Dodd, Kazuhiko Kawamura, and David C. Noelle. Modular behavior control for a cognitive robot. ROMAN Conference, 2005.

[24] Palis Ratanaswasd, Stephen Gordon, and Will Dodd. Cognitive control for robot task execution. ROMAN Conference, 2005.

[25] Charles Rose, Bobby Bodenheimer, and Michael F. Cohen. Verbs and Adverbs: Multidimensional Motion Interpolation Using Radial Basis Functions. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.

[26] Stuart Russel and Peter Norvig. *Artificial Intelligence A Modern Approach*. Prentice Hall, 2003.

[27] D.M. Wilkes, Mert Tugcu, J.E. Hunter, and David Noelle. Working memory and perception. ROMAN Conference, 2005.

[28] Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, Ltd., 2002.