Advanced Therapy Learning Algorithm for Spinal Cord Stimulation

by

Amanda Dawn Gaudreau Balderrama
B.S. Electrical Engineering
Massachusetts Institute of Technology (2009)

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

Master of Engineering in Electrical Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

Author .....
Department of Electrical Engineering and Computer Science
August 20, 2010

Certified by .....
Dr. Collin M. Stultz, MD
W. M. Keck Associate Professor of Biomedical Engineering
Thesis Supervisor

Certified by .....
Eric Panken
Scientist, Medtronic, Inc. - Neuromodulation
Thesis Supervisor

Accepted by .....
Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Advanced Therapy Learning Algorithm for Spinal Cord Stimulation

by

Amanda Dawn Gaudreau Balderrama

B.S. Electrical Engineering

Massachusetts Institute of Technology (2009)

Submitted to the Department of Electrical Engineering and Computer Science
on August 20, 2010, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Science and Engineering

Abstract

Spinal Cord Stimulation (SCS) is a technique used to treat chronic pain and has been shown to be an effective method of treatment, both financially and socioeconomically. Stimulating electrodes are surgically implanted into the epidural space, outside the dura, a protective sac filled with cerebral spinal fluid (CSF) surrounding the spinal cord. The thickness of the CSF changes according to body orientation, causing the distance between the stimulating electrodes and the spinal cord to vary. This phenomenon has been reported to cause painful or ineffective stimulation. In order to detect postural behavior and adjust SCS parameters accordingly, a tri-axial accelerometer based algorithm has been developed. The algorithm enables patients to adjust stimulation therapy parameters real-time, associates the patient indicated parameters with a vector, and stores them in a therapy library. Stimulation therapy parameters are then automatically selected by classifying incoming TA data according to the vectors in the therapy library, providing individualized, closed-loop stimulation therapy.

Thesis Supervisor: Dr. Collin M. Stultz, MD
Title: W. M. Keck Associate Professor of Biomedical Engineering

Thesis Supervisor: Eric Panken
Title: Scientist, Medtronic, Inc. - Neuromodulation

Acknowledgments

Medtronic's commitment to improving the quality of life for people suffering with life-threatening diseases and conditions is extremely prevalent and inspiring. Their contribution to my experience at MIT has been invaluable. I would like to thank the corporation as a whole for giving me this opportunity and for supporting the VI-A program, which has made a huge impact on my aspirations, achievements, and future.

I am in debt to the many who made my experience at Medtronic so memorable and productive, especially my advisor Eric Panken. Eric's abundant support and insightful guidance were vital to the development of this thesis. In addition, I would like to extend my gratitude to Lynn Davenport, Tim Dennison, Elizabeth Fehrmann, Justin Kemp, Dave Linde, Nicholas Mairs, and Dennis Skelton, who each offered me their own wealth of knowledge. These people contributed immensely to the development of this project and are fondly remembered for their continual support and willingness to help.

I would also like to thank my most influential professors at MIT, my thesis advisor Professor Collin Stultz, and my academic advisor Professor George Verghese. Professor Stultz first ignited my passion for Biomedical Engineering in 6.022 – Organ Transport Systems. Since then, his expertise and understanding have gone a long way in my career at MIT. Professor Verghese's unending support and encouragement were undoubtedly one of the essential elements in helping me achieve my goals. I could not count the times he helped me nor can I measure my gratitude for all of his benefaction.

Finally, and most importantly, I would like to thank my family, friends, and boyfriend for all they have done to support me. Mom, Christina, Jeromey and Jessica, I could not have made it through MIT without my periodic visits home filled with games, laughter and love. David, *if* any of this were possible without you, I know it would not have been worth as much, or have been as enjoyable. To my friends in "Toys–R–Us" and Ame, thank you for filling my life with more than just problem sets. Thank you all for giving my work meaning and for loving me through the difficulties.

Contents

List of Figures

Chapter 1

Introduction

Implantable Neurostimulators (INSs) are a broad class of devices used to treat a variety of neuropathological conditions using electrical stimulation. By stimulating different target nerves or regions in the brain, INSs can be used to improve a variety of conditions including sleep apnea [1], chronic headaches [2], and chronic pain [3]. The most basic understanding of the therapeutic effect induced by an INS is that the electrical pulse generated disrupts or blocks pathological nerve signals thereby reducing symptoms associated with the disease being treated. Therapeutic effects of neurostimulation include reduction of tremor in Parkinson's Disease patients, prevention of seizures in epilepsy patients, and pain relief by induced paresthesia in chronic pain patients.

INSs require a defined set of stimulation therapy parameters (STPs) to specify various aspects of the electrical stimulation, such as pulse width, frequency, voltage (or current) amplitude, and electrode configuration. The exact correlation between stimulation therapy efficacy and STPs is still not completely understood for many INS applications, such as with deep brain stimulation [4] and spinal cord stimulation [5]. Often times, programming of these devices is done via trial and error [6], and is an arduous task for both clinician and patient [7]. In the field of spinal cord stimulation (SCS), various theoretical models [8] and automated methods [7] have been designed to aid with initial STP selection and electrode configuration based on surgical lead position and patient feedback. Selecting appropriate lead configurations such that the

paresthesia induced by the device is concordant with painful regions is the definition of success for this therapy therapy [9].

The "single most important factor" which dictates the magnitude of the pulse required to produce a therapeutic effect (in terms of energy delivered to the spinal cord) is the distance between the stimulating electrodes and the neural structures being targeted in the spinal cord [5]. Various researchers have found that the difference between the stimulation amplitudes required to elicit therapeutic responses in upright versus supine postures to be statistically significant [10] [11] [12]. Methods developed for compensating for these variations in distance between the stimulating electrodes and the neural target include constant current models [12] and ultrasonic distance detection [13]. The former has been deemed an ineffective method, as impedance does has no statistically significant dependence on posture [9], and the latter has not been fully developed and decreases battery life of the device by approximately 20%.

A SCS device which 1) accommodates for changes in STPs over time and 2) automatically adjusts STPs according to distance between the stimulating electrodes and the spinal cord, had not be developed until recently. A device designed to meet these unmet needs has been underway at Medtronic. Since the position of the spinal cord in the spinal column has been found to vary based on body orientation [14], Medtronic assessed the feasibility of detecting five basic postures using a single, tri-axial accelerometer (TA). The TA provides information about a patient's posture, which is then used to automatically select appropriate STPs. This concept has lead to the development of a newly emerging, closed-loop SCS, currently commercially released in Europe and under clinical investigation in the US.

The work described in this thesis is an extension of the TA based posture detection algorithm developed by Medtronic. Instead of using five, predefined postures to select STPs, the proposed algorithm allows patients to indicate STPs at any given time. The algorithm associates the information with a vector in three-space, and uses the information to automatically administer stimulation therapy for future TA data. This method allows patients to develop their own closed-loop SCS therapy such that it uniquely suits their therapy needs.

Chapter 2 provides a basic background on SCS systems and spinal cord physiology. It also describes the relevant research supporting the development of body orientation based SCS systems. The body orientation based algorithm for stimulation (BOBAS) developed by Medtronic is outlined, and details regarding the data available for assessing the performance of the algorithm developed in this thesis is also described. Chapter 3 explains the functional aspects of the Advanced Therapy Learning Algorithm for Stimulation (ATLAS) and the parameters involved in defining an individualized stimulation therapy plan. The performance of ATLAS in terms of defined objectives and comparisons with BOBAS are presented in Chapter 4. A summary of the optimal algorithm implementation is given in Chapter 5, along with suggestions for future work.

Chapter 2

Background

This chapter gives the basic information necessary for understanding the development of the Adaptive Therapy Learning Algorithm for Stimulation (ATLAS). First, the history, basic functional components, and implantation of a spinal cord stimulation system will be presented. A discussion of the research which motivated the development of a posture detection algorithm will follow. The current closed-loop algorithm developed by Medtronic will then be described in detail. Finally, the study data used to design the algorithm developed in this thesis will be discussed.

2.1 Spinal Cord Stimulators

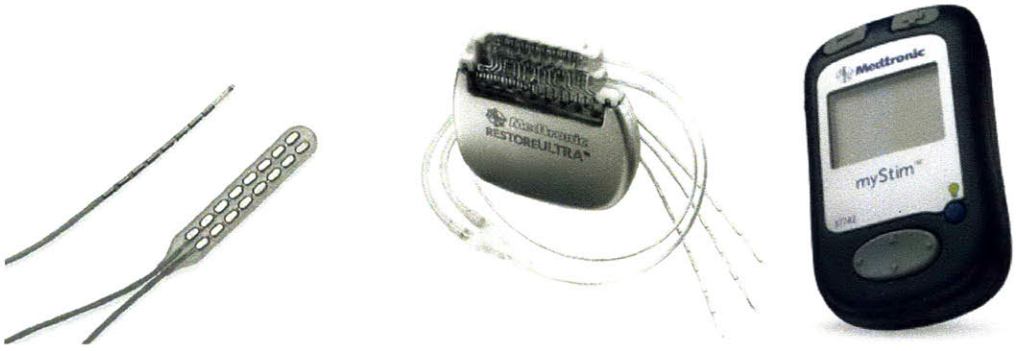
Spinal cord stimulation (SCS) is an effective technique used to treat a variety of chronic pain conditions including failed back syndrome, degenerative disk disease, and complex regional back syndrome [15] [16] [17]. In 1965, Melzack and Wall's *Gate Control Theory of Pain* proposed that transmission of pain is performed by central transmission cells which relay pain signals to the brain [18]. This theory lead to the first experiment attempting to treat pain by means of percutaneous epidural spinal cord stimulation in 1967, conducted by Shealy et al [19]. This experiment, though followed shortly by death of the patient due to an undiagnosed bacterial infection in the brain, demonstrated that SCS was a feasible and effective technique for pain management. Since then, advancements in surgical techniques, stimulation methods,

and device designs have transformed SCS therapy from an extremely risky procedure to an effective alternative for pain management [3].

Despite the risks associated with the therapy [20], it has been found to improve quality of life in the long term [21]. With proper patient selection and surgical techniques, SCS has been identified as an effective method for pain management, leading to a reduction in pain and increased patient satisfaction when compared to conventional medical management alone [22]. In addition, SCS has been found to reduce the net per patient cost when compared to conventional pain management treatments [23].

As shown in Figure 2-1(b), a SCS is an IPG with one or more insulated conductive lead wires connected to it. At the end of the lead wire, a lead with exposed conductive electrodes delivers the electrical impulse from the IPG to the spinal cord. As shown in Figure 2-1(a), the number, shape and size of lead and electrodes varies. The lead is implanted beneath the lamina, into the epidural space (Figure 2-2(a)), either via a laminectomy or through percutaneous needles depending on the lead design [24][25]. The other end of the lead wire is attached to the IPG (Figure 2-2(b)). The IPG is typically implanted in the lower abdominal or gluteal region (Figure2-2(c)), and generates an electrical pulse which is then sent to the spinal cord via the stimulating electrodes. The electrical pulse interrupts the pain signal to the brain and replaces it with a tingling sensation known as paresthesia [12].

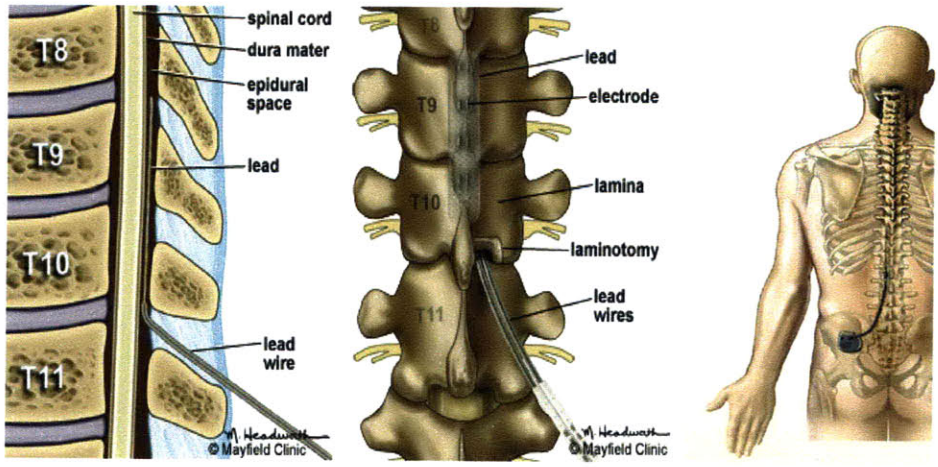
The parameters of the electrical impulse generated by the IPG are called stimulation therapy parameters (STPs). STPs include pulse width, rate, voltage or current amplitude, and electrode configuration. In addition, the SCS can be programmed to cycle through multiple programs, where each program may have electrondes configured to target different pain regions. The anode-cathode configuration of the electrodes can also be specified as a means of steering the electric field generated to achieve different therapeutic effects. STPs are set by a clinician shortly after implantation [24][12]. Recent SCS systems come with a patient programmer (PP), shown in Figure 2-1(c). The PP allows patients to adjust their STPs outside of the clinic. While the PP does improve the overall efficacy of a patient's SCS therapy, it still requires



(a) (b) (c)

Figure 2-1: (a) Percutaneous type lead with eight-electrodes (top left) and paddle type lead with two, eight-electrode arrays (bottom right) - Courtesy of Boston Scientific. (b) SCS device, RestoreUltra - Courtesy of Medtronic. (c) MyStim Patient Programmer - Courtesy of Medtronic.

patients to manually change STPs whenever the SCS therapy is either not masking pain, or worse, causing uncomfortable or painful stimulation. Medtronic was among the first companies to incorporate closed-loop functionality into their SCS system, aiming to reduce patient burden and improve patient's quality of life.



(a) (b) (c)

Figure 2-2: The "**T#**" identifies the Thoracic Vertebrae. (a) Sagittal view of electrode placement - Courtesy of Mayfield Clinic. (b) Posterior view of electrode placement - Courtesy of Mayfield Clinic. (c) Overall view of SCS device placement - Courtesy of Neuron Medical Art.

2.2 Physiological Motivation for Body Orientation Based Stimulation

Originally, SCS leads were placed subdurally, resulting in a high degree of risk and morbidity associated with the therapy. To mitigate these detrimental factors, leads were instead placed in the dorsal epidural space [18]. Coupled with the prevalent use of percutaneous electrodes rather than paddle type electrodes, this meant that a less invasive surgery could be used to implant the electrodes using a percutaneous needle [26]. While this made SCS a more viable technique for treating chronic pain, it also introduced other technical challenges. Because of the distance separating the electrodes from the neural target – namely, the transmitter systems in the dorsal horns [27] – the varying thickness of the dorsal cerebral spinal fluid layer changes the effective electric field received at the target location. In addition to the varying thickness of the cerebral spinal fluid depending on spinal level [8], body orientation has been shown to change the position of the spinal cord in the spinal column [9][13][28].

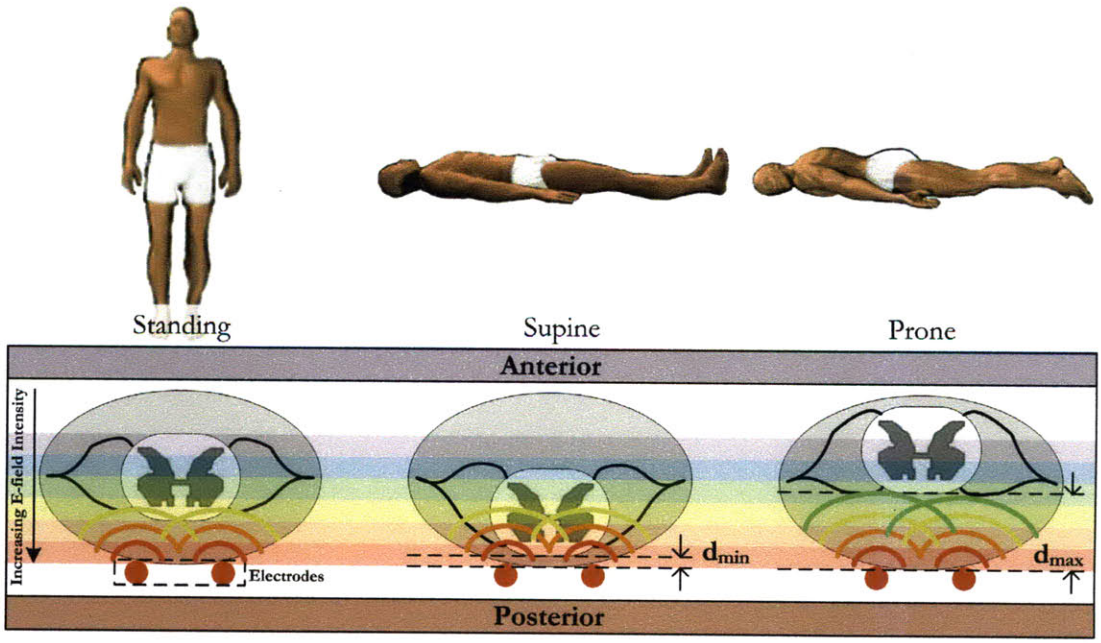


Figure 2-3: Spinal cord movement in the spinal column due to body orientation.

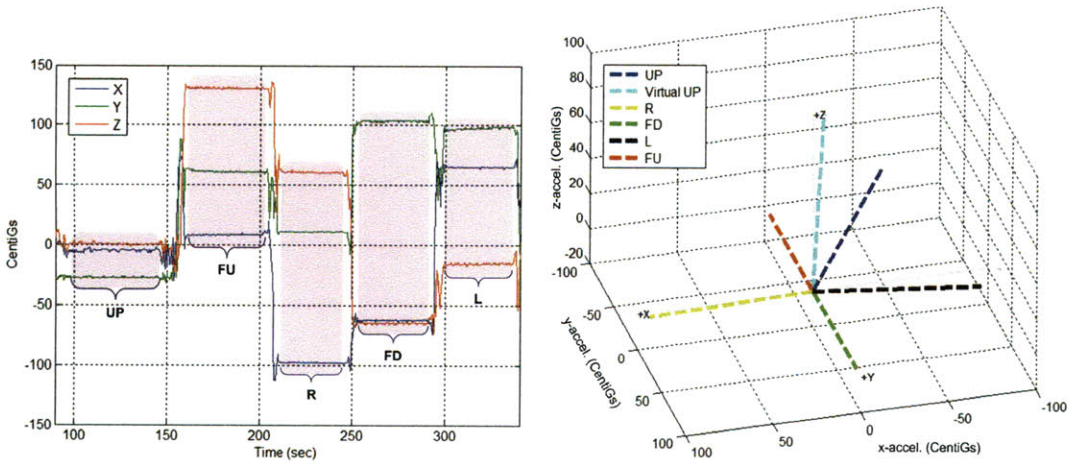
The diagram in Figure 2-3 depicts the movement of the spinal cord dorsally when

going from standing to supine. Since the distance between the spinal cord and the stimulating electrodes is decreased, this causes an effective increase in the electric field intensity and could lead to a painful or shocking sensation [12]. The difference between the therapeutic stimulation voltage required for upright versus supine postures has been found to be statistically significant [11][10]. To compensate for the adjusted stimulation level required to elicite a therapeutic effect, Medtronic developed a Body Orientation Based Algorithm for Stimulation which uses a tri-axial accelerometer to detect five basic postures. Based on these posture detections, the stimulation therapy parameters (STPs) are automatically adjusted to patient preferred levels for each posture, mitigating the likelihood of over-stimulation.

2.3 Body Orientation Based Algorithm for Stimulation (BOBAS) using an Accelerometer

Motivated by the literature suggesting a statistically significant difference between standing and supine stimulation voltages, Medtronic developed a body orientation based algorithm for stimulation (BOBAS) which could reliably detect five basic postures: standing upright (UP), lying face up (FU), lying face down (FD), lying on the right side (R), and lying on the left side (L). The algorithm uses five orientation vectors (v_{OR}s) corresponding to UP, FU, FD, R, and L to partition three space and sort TA data according to its spatial location relative to the v_{OR}s. The five vectors are defined during an initialization period, during which data is collected while the patient assumes each of the five postures. Stable data within the start and end times of an orientation, indicated by shaded purple boxes in Figure 2-4(a), is used to compute a mean vector for each orientation. Ideally, the v_{OR}s would be orthogonal; however, factors such as implant location and physical constraints result in the relative vector positions that are far from ideal. In order to clearly differentiate upright postures from lying ones, a virtual upright (v_{virtUP}) orientation vector is synthetically created as the vector normal to the plane approximated by the lying vectors. The v_{OR}s, which

include v_{virtUP}, are depicted spatially in Figure 2-4(b).



(a) (b)

Figure 2-4: (a) Temporal representation of v_{OR}s. Shaded purple regions indicate data that was used to aggregately equal the final v_{OR}. (b) Spatial representation of v_{OR}s transformed into xyz-coordinates.

A set of partitioning parameters are used to uniquely divide three-space according to the position of the v_{OR}s. Two parameters, θ_{UP} and θ_{LD}, are used to form cones around v_{UP} and v_{virtUP}. As shown in Figure 2-5(a), two cones θ°_{UP} around both v_{UP} and v_{virtUP} define upright postures. Another cone θ°_{LD} from v_{virtUP} divides upright from lying therapy space. Anything greater than θ°_{LD} from v_{virtUP} is considered a lying posture. Data that fall between these two defined regions are classified as hysteresis postures. Therapy space regions are shown in Figure 2-5(b). Planes that are equidistant between two adjacent lying vectors further divide the lying therapy space into four regions, as shown in Figure 2-5(c) and (d).

For a pain patient, these six v_{OR}s define the basis of the therapy space. Optimal STPs are selected for each of the five basic postures and are used for stimulation therapy whenever the posture corresponding to the v_{OR} is detected by BOBAS. Both the v_{OR} data and the associated STP values are stored in a therapy library. Each column of the therapy library contains the xyz-coordinates of one of the v_{OR} and the associated STP values. The information in the column of the therapy library is referred to as an entry. Since BOBAS uses five basic postures, a BOBAS therapy

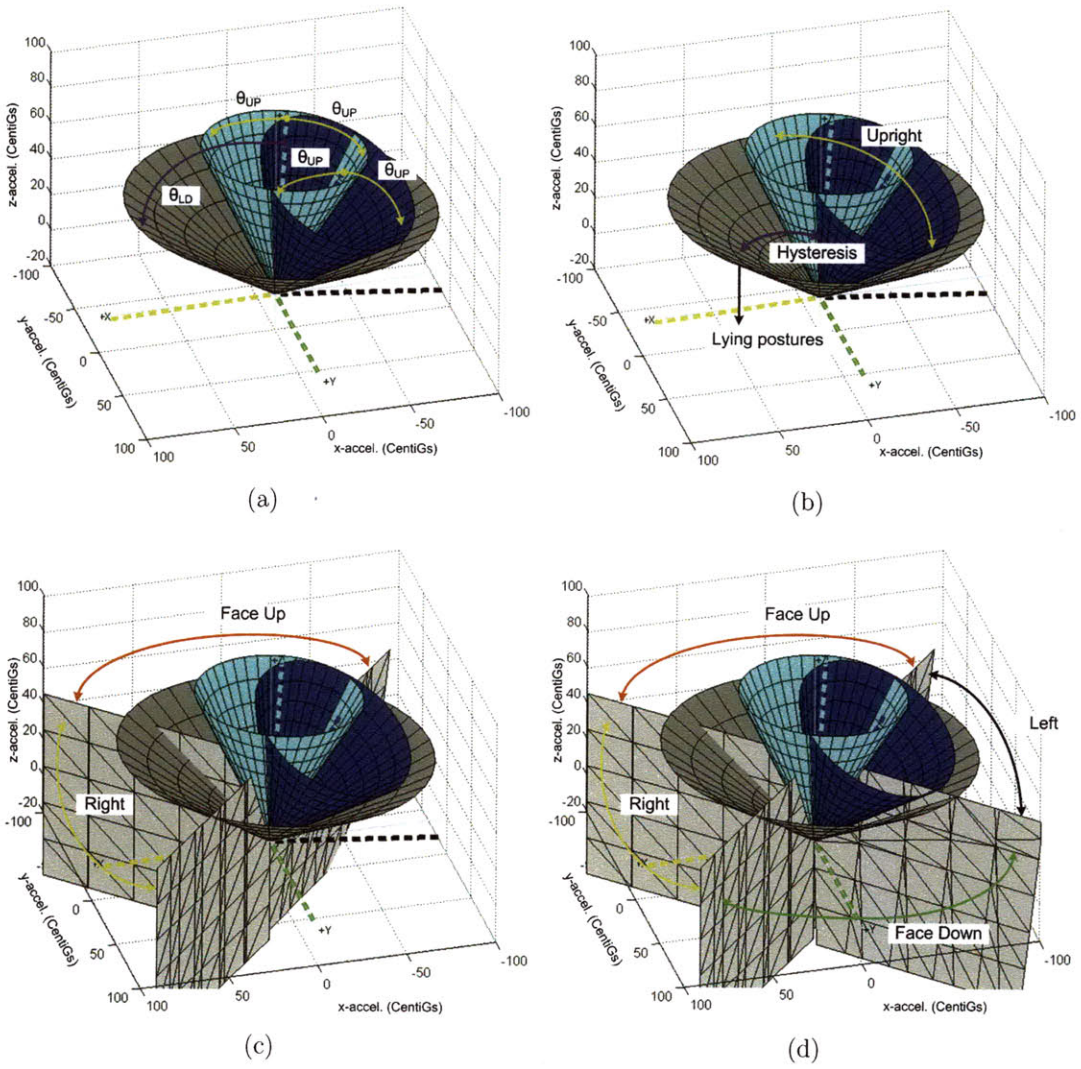


Figure 2-5: (a) Two cones defined θ°_{UP} from v_{UP} (blue) and v_{virtUP} (cyan) specify the upright region shown in (b). A third cone (grey) θ°_{LD} from v_{virtUP} divides the upright therapy-space from the lying down therapy-space, with a hysteresis region between the two broad posture regions, shown in (b). (c) and (d) Planes equidistance between two neighboring lying vectors divide the lying therapy space into four regions.

library always has five entries. In addition to the five static posture detected, BOBAS may also classify a TA signal as "Upright and Active" using an concurrently running activity detection algorithm. Since the algorithm developed in this work only develops a detection algorithm for static postures, the activity algorithm will not be discussed.

2.4 Description of Study Data

The data used to design and validate the closed-loop adaptive algorithm developed in this work was collected in a two-part, Medtronic run research study called the PRS Study. The first protocol of the study was a three-day monitoring period designed to identify subjects who made frequent STP adjustments due to changes in posture or activity. In order to participate in the second protocol of the study, enrolled subjects were required to make at least two amplitude adjustments per 24 hour period during the first part of the study. The first portion of the study will be called the "qualification protocol" since subjects were required to meet certain criteria in order to qualify for the second part of the study. The second part of the study will be called the "validation protocol" since the objective of this portion of the study was to evaluate the performance of BOBAS and subject's satisfaction with algorithmically derived stimulation therapy adjustment. The validation protocol was a 4 hour, in-clinic protocol. The objective and description of each protocol and the constraints of the data sets collected will be discussed in detail in Sections 2.4.1 and 2.4.2.

The two-part study was conducted over an eight month period from August 2008 to April 2009. A total of 19 subjects completed the qualification protocol and 16 completed the validation protocol. Of these data sets, 16 and 11 were available from the qualification and validation protocols respectively for development of the algorithm described in this thesis. Subjects who participated in the study were ambulatory chronic pain patients with a Medtronic Restore or Restore Advanced SCS device implanted for at least three months. Subjects were required to be at least 18-years-old, have a BMI of 40 or less, and implanted with percutaneous thoracic leads. Hardware for each protocol of the study will be described in each subsection.

2.4.1 Three-day Qualification Protocol

In the qualification protocol, subjects were instructed to change their SCS STPs as they normally would throughout the day and were asked to maintain an activity log. Subjects were given a personal data assistant (PDA) to keep their activity log. The

device was specially formatted for the qualification protocol. An ambulatory data recorder, called a μADR (mircoADR) (Figure 2-6), is an external, battery powered device which is adhered to the hip or abdomen of a subject before beginning the study. It encases a TA (ADXL 330, Analog Devices) and onboard memory used to record data over the three-day period. The μADR itself does not have the capability of delivering, monitoring or augmenting therapy, but rather simply records TA data. To change STPs, subjects were given a research PP identical to the one pictured in Figure 2-1(c) with modified software to record all programming changes and associated timestamps made by the subject.

Figure 2-6: The Micro Ambulatory Data Recorder (μADR) is a small device approximately 1.5 x 2 inches which is adhered to a subject's gluteal region. It encases a triaxial accelerometer (ADXL-330, Analog Devices) as well as a data recording device.

The objective of the qualification protocol was to summarize the time course and type of STP changes made relative to TA data collected. No algorithm was used to automatically administer therapy during the course of the qualification protocol. Before beginning the study, orientation vectors were defined and used for post-processing purposes to broadly characterize daily activity for pain patients. These data were also used to validate BOBAS posture detection accuracy. Data sets resulting from the qualification protocol are referred to as three-day study data ($3data$). Each data set included the following information:

- Two to four days worth of TA xyz-data
- PP data indicating changes to STPs and the timestamp associated with each change (required modification to commercial software available for the device)

- Patient activity log information recorded using a PDA data. The activity being performed and the timestamp associated with the activity log entry was recorded.
- Five orientation vectors recorded in-clinic
- Post-processed BOBAS posture and activity classifications for each TA data point

Subjects were able to have four active programs at any given time. In each program, the simulation voltage amplitude, pulse width, or rate can be adjusted. Additionally, the device may be turned on or off. A string indicates the STP adjustment and the corresponding timestamp. This information is then saved in a PP data file. These changes are identified by 'Inc Amp #', 'Dec Amp #', 'Inc PW #', and 'Dec PW #' where # is an integer 1–4 representing the program which was changed. Changes in rate are identified by 'Inc Rate' and 'Dec Rate' and turning the SCS on or off is indicated by strings 'Stim On' and 'Stim Off'. Along with the string descriptions and timestamps, the final STPs after each PP input are given for each program.

The intention of the subject annotated activity log from the PDA was to validate the basic BOBAS functionality; however, actual daily activity and precision of any subject's activity log could not be asserted, as subjects were not visually monitored. Activity annotations were only loosely considered as a validation tool. Subjects annotated their activity by selecting from a predefined list: Bath, Car Ride, Go To Bed, Lie Down, On Back, On Left Side, On Right Side, On Stomach, Other, Shower, Sit, Stand, Wake Up, and Walk. On average, subjects made 20.5 activity annotations per day with a standard deviation of 13.3 and minimum and maximum annotations per day of 6.3 and 49.3, respectively.

Based on post-posture classifications from BOBAS and observation of the data in three space, subjects frequently annotated an activity before or after physically performing the indicated activity. An example would be a subject indicating "Go To Bed" while remaining in a posture classified as upright for several minutes before ultimately assuming lying postures for several hours. These deviations between activity

log annotations and true engagement of the annotated action made it difficult to assign a true start and end time to any given activity log annotation. Similarly, subjects would make STP changes before or after assuming one of the five BOBAS postures for an extended period of time, making it difficult to attribute the STP change to a specific posture or posture transition. A common phenomenon was changing STPs from high, "awake amplitudes" to lower, "sleeping amplitudes" before going to bed. Many subjects would lower stimulation amplitudes in active programs 2-30 minutes before assuming a lying posture for several hours, indicating the while a STP change was being made, it was actually intended for a posture somewhat far in the future.

Figure 2-7 depicts the two scenarios described above. The subject indicates "Go to Bed" as the current activity at 1900.2 minutes (indicated by the cyan line); however, as shown by the BOBAS classification of the TA data, his posture is still classified as UP until about 1924.2 minutes (indicated by the magenta line). In addition, the stimulation amplitude voltage is changed from 5 V or greater to 2 V at 1912.7 minutes (indicated by the green line). A stimulation amplitude of 5 V or greater was used for about 13 hours before the subject turned the amplitude down to 2 V, which was the stimulation voltage level he used for the following 7.7 hours. Primarily lying postures were assumed from 1924.2 to 2364.5 minutes of this particular subject's qualification protocol data, indicating that he was sleeping and that the 2 V stimulation level was intended for the sleeping period.

Other problems which made the data difficult to interpret were (1) using a wide range of stimulation amplitudes in a single posture (or region of three space), and (2) using the same stimulation amplitude for a variety of postures (or a vast region of three space). While the relationship between the activity annotations, STP changes and the BOBAS basic posture classifications may be reasonably inferred by examining the data, these discrepancies are difficult to resolve without artificially creating a data set and limited the extent of analyses possible using the *3data*.

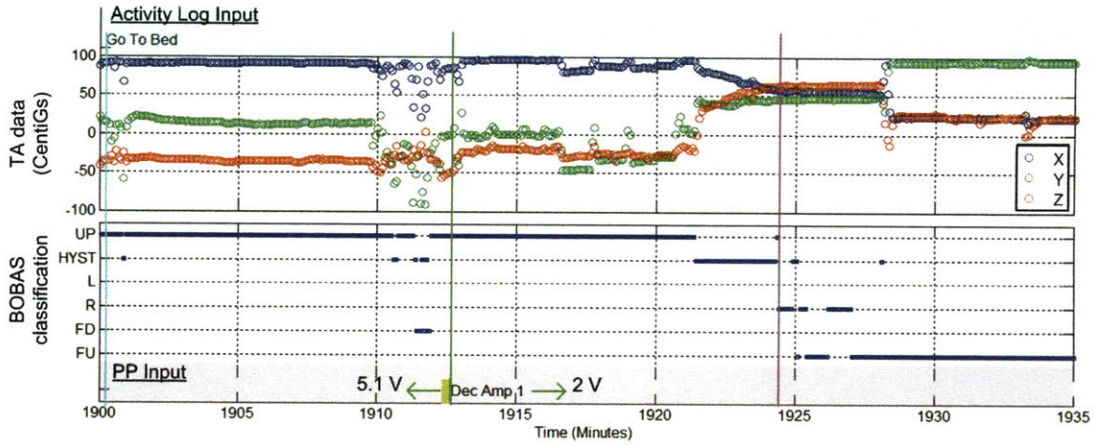


Figure 2-7: Activity log input versus changes in posture and stimulation therapy parameter levels. The cyan line indicates time associated with the "Go to Bed" activity log input, the green line marks the time associated with stimulation voltage adjustment from 5.1 V to 2 V, and the magenta line indicates the time when the subject goes from primarily UP BOBAS classifications to primarily lying classifications.

2.4.2 In-clinic Validation Protocol

The data from this portion of the study is referred to as in-clinic study data ($ICdata$). As mentioned, to qualify for the validation protocol, subjects were required to make at least two posture or activity related programming changes per 24-hour period in the qualification protocol. The objective of the validation protocol was two-fold: to demonstrate a statistically significant difference between standing and supine therapeutic stimulation amplitudes, and to compare the satisfaction of algorithmically versus manually changed STPs. In the verification protocol, a clinician instructed subjects to perform a sequence of physical tasks, such as stand, lie supine, lie prone, ect., while the BOBAS algorithm ran real-time to adjust STPs based on detected posture.

For the validation protocol, subjects were given a μADR capable of wireless communication with a separate memory module. The memory module, which is a separate, battery powered unit, connects to a lab programmer (LP) via USB. As TA data is collected from the μADR, it is wirelessly sent to the memory module and interpreted by the LP. The LP runs BOBAS and performs real-time STP selection based on the posture classification of the given TA data. The stimulation information

is then telemetered to the implanted SCS and is used to adjust stimulation therapy. Various metrics were collected in order to asses how satisfied subjects were with the algorithmically derived STPs and how comfortable the stimulation therapy was for each physical task in the protocol.

Each data set from the validation protocol included the following:

- About 15 minutes of TA xyz-data
- Event timestamps and labels which correspond to the time and type of activity the subject was instructed to perform
- Truth postures associated with the BOBAS classification for each of the physical tasks in the protocol
- Five orientation vectors recorded in-clinic, as well as a calculated virtual upright vector

Since the $ICdata$ was collected in a controlled environment and at a higher sampling frequency than the $3data$, it was used to characterize the signal features of stable postures, noise, and transitions between postures. Values for typical point to point distance variation for each these TA features were derived using these data sets. This information was used to define the preliminary parameter values for the algorithm described in this work. These data sets were also used to compare the classification accuracy between BOBAS and ATLAS. Results from this comparison will be detailed in Section 4.2.1.

Chapter 3

Adaptive Therapy Learning Algorithm for Stimulation (ATLAS)

As discussed in Section 2.3, given a set of TA data, BOBAS will determine appropriate stimulation therapy parameters (SPTs) using a therapy library seeded with five orientation vectors (v_{OR}s). While the STPs associated with each v_{OR} may be redefined by the patient at any time, the number of library entries is fixed at five and the xyz-coordinates of the v_{OR}s used by BOBAS must be defined in a clinical setting. BOBAS has the following limitations:

1. The practical inter-postural variation from the v_{OR}s recorded in clinic. A posture assumed outside of the clinic may correspond to a somewhat different area in three-space relative to the v_{OR} defined in the therapy library.

2. The utility of any given v_{OR} outside of the clinic. For instance, it is common for patients to avoid lying down on the side that the IPG is implanted on. If the IPG is implanted on the right side, the patient may rarely lie on that side, thus making the library entry unnecessary.

3. The possible need for a greater density of vectors to adequately describe the STPs needed in a given basic orientation region.

For these reasons, an algorithm which allows patients to define arbitrary reference vectors (v_{ref}s) without associating them with any body orientation or fixing the number of library entries would allow patients to dynamically customize and fine tune their therapy space. Using a patient programmer (PP), patients can indicate preferred STPs real-time and the Adaptive Therapy Learning Algorithm for Stimulation (ATLAS) will progressively build a therapy library containing valid v_{ref}s and their associated STPs. If v_{OR}s are required for diagnostic purposes, two or more v_{OR}s can be defined as part of the clinical initiation process.

Section 3.1 will give a high level overview of the interaction between ATLAS's three functions. Sections 3.2– 3.4 will discuss the data association function ($f(assoc)$), library modification function ($f(libmod)$), and data classification function ($f(class)$) of ATLAS in detail.

3.1 System Overview

ATLAS operates within a global environment where information from the therapy library, TA, and PP dictate execution of functions. There are three functions: data association, library modification, and data classification. Initially, a clinician will seed the therapy library with at least one default entry. Until a PP input is registered, the data classification function selects STPs based on the distance between the incoming TA data and the v_{ref}s currently stored in the therapy library.

Once a PP input is registered, the data association function ($f(assoc)$) is executed. In the $f(assoc)$, association criteria is used to search for a series of stable data to generate a valid association vector (v_{assoc}). The v_{assoc}, along with the STPs indicated by the PP input, are collectively called association data. If a stable series of data is identified, the association data will be passed into the $f(libmod)$ and the library will be updated according to similarity criteria imposed within the function. Figure 3-1 shows the interaction between the three ATLAS functions and the events which result in a change in the functional state.

The $f(assoc)$ and $f(libmod)$ perform various high level tasks. In the $f(assoc)$,

time, distance, and noise criteria are used to evaluate whether the STPs indicated by the PP input can be reliably associated to a v_{assoc}. If a time series of data passes the stability and same posture distance criteria for the duration of the stable timer (T_{stable}) within a search timer (T_{search}) period, a v_{assoc} will be generated and the association data will be passed into the $f(libmod)$. The $f(libmod)$ then assesses whether the new v_{assoc} is sufficiently similar to any of the existing v_{ref} in the therapy library or whether the new v_{assoc} is different enough to create a new library entry with the association data. A detailed state diagram with the high level processes performed during transitions from one state to another is shown in Figure 3-2.

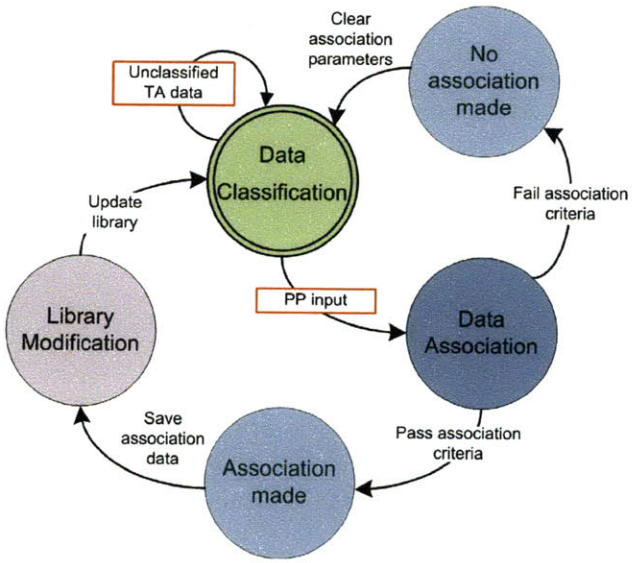


Figure 3-1: Basic state diagram depicting relationship between the Data Classification (green), Data Association (blue) and Library Modification (purple) functions. The boxes outlined in red indicate incoming input data which will change the current functional state.

The algorithm was implemented in MATLAB where a main script[1] reads in data files containing time-series TA and PP data. Figure 3-3 describes the main function of ATLAS in flow chart format. The variable naming scheme used in Chapter 3 is given on Table 3.1. The time-series data is read in one data point at a time at a frequency of f_{TA} Hz. For each sample, the xyz-coordinates of the current TA data point (v_t), the time when the last PP input was recorded (t_{PP}) and the current sample time (t)

[1]The terms script and function are used interchangeably.

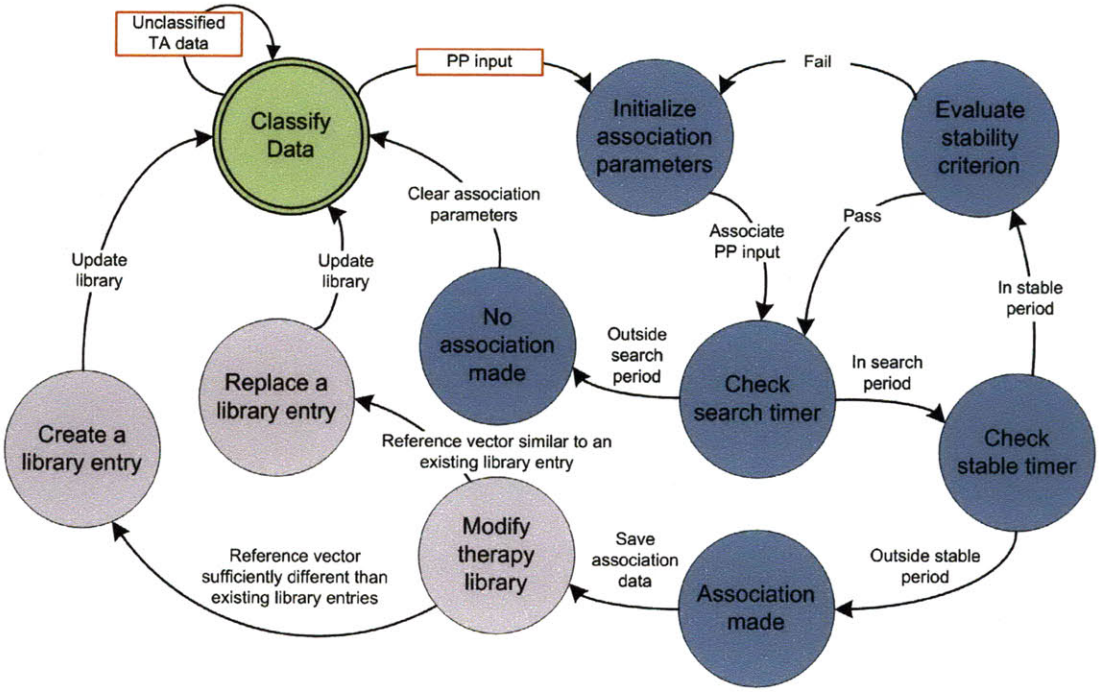


Figure 3-2: Detailed state diagram showing states of the Data Classification, Data Association and Library Modification functions in green, blue and purple respectively.

are known. In addition, a collection of flag variables (F_{type}) indicate the current state of the algorithm.

Abbreviation	**Type of Variable**	**Units**
T_{type}	Period of time	seconds
t_{type}	Time keeping variable	seconds
F_{type}	Flag	binary value (0 or 1)
D_{type}	Distance threshold	metric dependent
d^2	Distance between two vectors	metric (D, E, or S, see Table 3.2)
v_{type}	xyz-data points	centi-g
$variable_{t-n}$	Value of $variable$ at time $t-n$ where n corresponds to number of samples	variable dependent

Table 3.1: General reference for variable and parameter symbolic nomenclature and their definitions and units of measurement.

By default, the $f(class)$ is executed. Until an input is received from the PP, STPs will be selected for each incoming TA data point using the therapy library. ATLAS begins executing the $f(assoc)$ when a new PP input is received (F_{new}=1).

The registered PP input prompts the initialization of the time keeping variables t_{search} and t_{stable}, as well as other data keeping variables which control the generation of a v_{assoc} within the $f(assoc)$. The $f(assoc)$ is executed as long as $F_{associate} = 1$. As shown in Figure 3-2, the $f(assoc)$ will continue to be executed with each new sample until either the search timer has expired ($t - t_{search} \geq T_{search}$) or an association vector, v_{assoc}, has been formed (which occurs when $t - t_{stable} \geq T_{stable}$). The former case occurs in the main script when $F_{associate}$, F_{new}, and $F_{library}$ all equal zero and the latter case occurs when the the library modification flag is set ($F_{library} = 1$). The method by which these flags get set and the v_{assoc} gets formed will be discussed in detail in Section 3.2.

All three functions require that the distance between two points be measured. Three distance metrics are used throughout ATLAS to measure the relative position of two vectors: Angular (A[2]), Squared-Euclidean (E) and Sum-of-Differences (S[3]). The distance metrics are defined on Table 3.2. Each time ATLAS is run, the same distance metric is used to measure all require distances throughout the algorithm. The three methods for calculating distance were chosen to explore the performance between different methods of partitioning three-space. In addition, the computational complexity of each of the distance metrics ranges from very complex (requiring a trigonometric operation as well as two square root operations for D) to very simple (requiring only simple addition and absolute value for S).

Distance Metric	Abbreviations	Formula, $d^2(v_1, v_2)$
Angular (Degrees)	D	$\cos^{-1}\left(\frac{\sum_{i=1}^{n} v_1(i)v_2(i)}{\|v_1\|\|v_2\|}\right)$
Squared Euclidean	E	$\sum_{i=1}^{n} (v_1(i) - v_2(i))^2$
Sum-of-Differences	S	$\sum_{i=1}^{n} \|v_1(i) - v_2(i)\|$

Table 3.2: Distance metrics used in ATLAS to compute the distance between two vectors, v_1 and v_2. The vectors are three dimensional ($n = 3$) where the first, second and third elements correspond to x, y, and z components of the vector.

[2]While the angle between two vectors is not necessarily a distance, it will be referred to as such in this work. Vectors will always be formed as lines extending from the origin to the vector's xyz-coordinates.

[3]Also known as Manhattan or City-block distance [29].

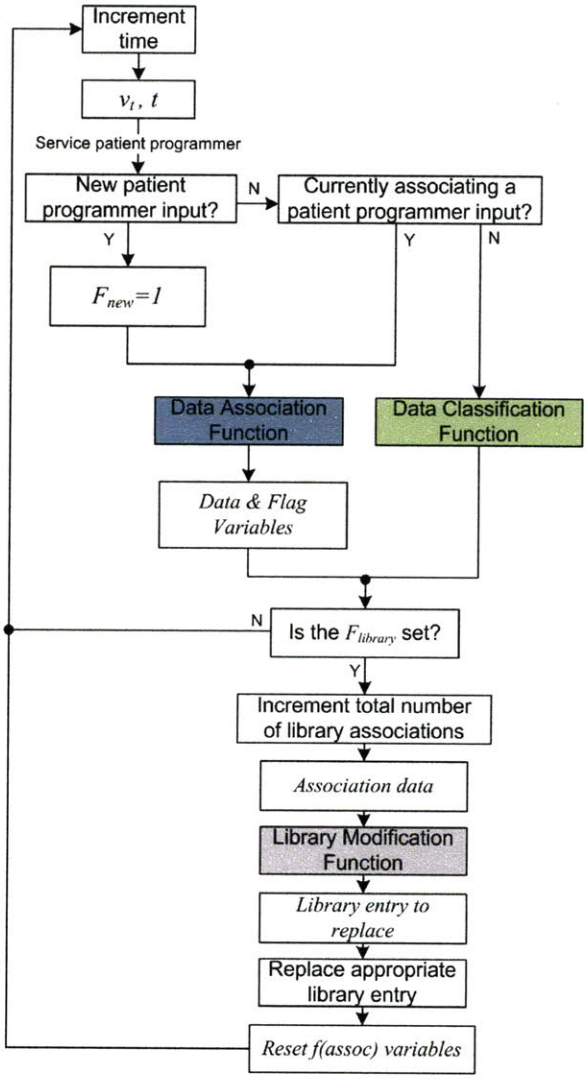


Figure 3-3: Functional layout of the ATLAS main script implemented in MATLAB. The data association, data classification and library modification functions are called from this script. Variable abbreviations and definitions can be found on Table 3.1.

3.2 Data Association: Mapping of Therapy Parameters to a Position Vector

The $f(assoc)$ is the most computationally intensive function of ATLAS. This function must identify a segment of data that corresponds to the postural behavior causing a patient to make a STP change. A natural technique for a problem of this nature would be to implement a machine learning algorithm and collect training data

over some period of time. Unsupervised learning techniques were explored, but were considered unsuitable since they require large data storage capacity and impose significant patient burden during the training period. In addition, since machine learning algorithms tend to be sensitive to factors such as initial conditions [30], there is no guarantee that the output would accurately reflect a patient's stimulation therapy needs.

The method used to associate a segment of data to a PP input in the $f(assoc)$ minimizes the computational complexity and gives patients the ability to refine their therapy space whenever necessary. While the capability to update STPs associated with a v_{OR} is possible using BOBAS, the $f(assoc)$ of ATLAS allows patients to define STPs in whatever way best suits their individual therapy needs. Presumably, patients change their STPs upon experiencing inadequate or painful stimulation therapy. For instance, if a patient were to go from FU to UP[4], he may require a higher stimulation amplitude. Upon standing, he might therefore increase the stimulation amplitude, and continue doing whatever activity he stood up to do.

The main objective of the $f(assoc)$ is to generate a representative vector which can be used to identify the STPs indicated by the PP input. Due to the variation in the position of the TA vector, this presents a major challenge. The distance between successive TA data points is constantly changing. Even when a patient is reasonably still, some variation in the location of the TA data points is expected as a result of insignificant patient movement or measurement error. When a patient changes his STPs to suit the new body orientation he is in, the $f(assoc)$ attempts to identify a concise region in three-space which is representative of the body orientation which caused the patient to change his STPs. The concise region is identified by a vector called an association vector (v_{assoc}). If a v_{assoc} is generated in the $f(assoc)$, it is passed into the $f(libmod)$ along with the indicated STPs and is incorporated into the therapy library.

The generation of a v_{assoc} is triggered by a PP input. Once a PP input is registered in the main function, the new PP input flag is set ($F_{new} = 1$) which causes the $f(assoc)$

[4]Posture references used for conceptualization, but are not actually defined in ATLAS.

to get executed (see Figure 3-3). The $f(assoc)$ uses six data association parameters and a variety of data variables to generate a v_{assoc}. There are two distance thresholds, two timers, and two noise allowance parameters. Collectively, the parameters are used to select a segment of data following a PP input which is suitable for generation of a v_{assoc}. The parameters and their function will be described below and the various operations performed during each iteration of the $f(assoc)$ are detailed in flow chart format in Figure 3-4. The data variables keep track of the state of the $f(assoc)$ and maintain data upon each iteration of the function. The function uses the following variables to dictate the generation of a v_{assoc}:

- Three time keeping variables
  - t_{search} : Time associated with the receipt of a PP input
  - t_{stable} : Time indicating the beginning of a stability period
  - t_{buffer} : A data vector with N elements which saves the last N times from previous iterations
- Two TA data recording variables
  - v_{stable} : xyz-data of the v_{assoc} being generated
  - v_{buffer} : A data vector with N sets of xyz-data corresponding to the last N TA data points from previous iterations
- A logical buffer
  - B_{p2p} : A logical buffer with N elements corresponding to the result of the stability criterion (explained below)
- Three flags
  - F_{new} : Indicates the detection of a new PP input
  - $F_{associate}$: Indicates whether a PP input is in the process of being associated to a v_{assoc}

– F_{libmod} : Indicates whether the $f(assoc)$ has successfully generated a v_{assoc} for a registered PP input or not

Using a point to point threshold (D_{p2p}), a same posture threshold (D_{assoc}) and two parameters which control the amount of noise acceptable out of N points ($MofN$), data with an acceptable amount of noise and spatial variation is identified and used to create a v_{assoc}. The four parameters impose three criteria which must be met to create a v_{assoc}: (1) a stability criterion, (2) a noise criterion, and (3) a same posture criterion. In order for an association to be made, all three criteria must be met within a specified time frame. The stable timer, T_{stable}, specifies the amount of data required to generate the v_{assoc}, and T_{search} limits the span of time during which a v_{assoc} can be generated after the receipt of a PP input. At the beginning of each iteration of the function, these time period requirements are checked. First, the search period it checked ($t - t_{search} \leq T_{search}$, Figure 3-4 **B.0**). If the current time is outside of the search period, then the STPs indicated by the PP input are not incorporated into the therapy library (Figure 3-4 **B.1**). If the current time is within the search period, a check to determine whether the stability period has expired is conducted ($t - t_{stable} \leq T_{stable}$, Figure 3-4 **C.0**). If it has, v_{stable} is associated with the STPs indicated by the PP input, and is referred to as an association vector (Figure 3-4 **C.1**). Both the vector and the STPs information are then passed into the $f(libmod)$.

The stability criterion is used to identify "stable" data which has very little variation in three-space. On the second iteration of the $f(assoc)$, the distance between the previous TA data point, v_{t-1}, and the current TA data point, v_t, is measured and compared to D_{p2p} (Figure 3-4 **D.0**). If the distance between v_{t-1} and v_t is less than D_{p2p} ($d^2(v_{t-1}, v_t) \leq D_{p2p}$), the two vector components are averaged together to generate a single stable vector, v_{stable} ($v_{stable} = mean(v_t, v_{t-1})$, Figure 3-4 **H.0**). New data points that pass the stability criterion continue to get averaged into the value of v_{stable} ($v_{stable} = mean(v_t, v_{stable})$). The first time the distance between v_{t-1} and v_t exceeds D_{p2p}, then the TA data point is flagged as noise (Figure 3-4 **F.2** and **F.3**).

A logical vector (B_{p2p}) with N elements is used to record the result of the stability

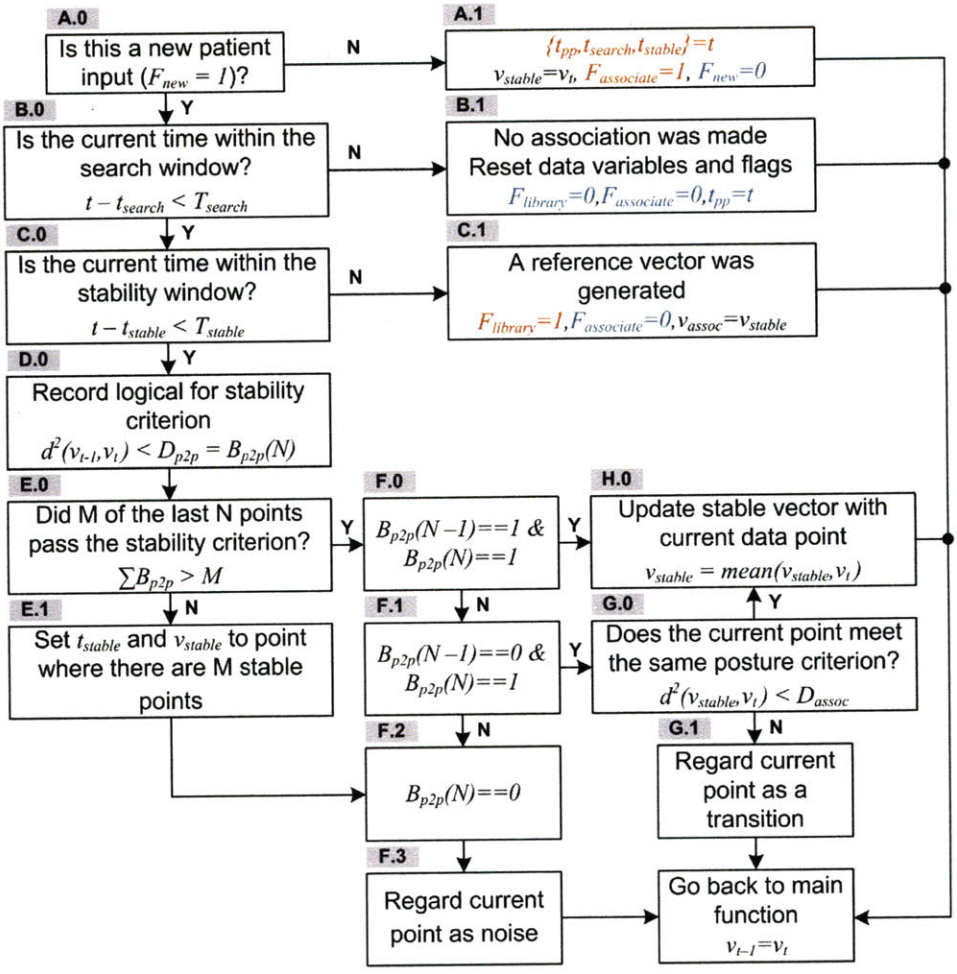


Figure 3-4: Data association function flowchart.

criterion for the last N points. A value of 1 or 0 is stored in the N^{th} element depending on whether the criterion is passed or failed. Initially, the B_{p2p} is padded with N ones. After the stability criterion is checked (Figure 3-4 **D.0**), the noise criterion is verified by summing the elements of B_{p2p} to ensure that at least M of the last N data points were considered suitable for inclusion in the v_{stable} (Figure 3-4 **E.0**). If greater than $N-M$ of the last N points failed the stability criterion ($\sum_{k=1}^{N} B_{p2p}(k) < M$, Figure 3-4 **E.1**), it indicates that the patient is either active or that the signal has too much variation to specify a concise region in three-space. If this occurs, the stability time keeping variable, t_{stable}, is reset so that the maximum number of noisy points are included and v_{stable} is recalculated accordingly.

If v_t fails the stability criterion, as long as the noise criterion is met, the point is regarded as noise. Once the noisy data subsides (as indicated by $B_{p2p}(N) = 1$, Figure 3-4 **F.1**), the distance between v_t and v_{stable} is measured and compared to a same posture threshold, D_{assoc} ($d^2(v_t, vstable) \leq D_{assoc}$, Figure 3-4 **G.0**). This is called the same posture criterion. Passing the same posture criterion indicates that the the point to point deviation in three-space between v_{t-2} and v_{t-1} was caused by noise, or an insignificant movement. The current data point is therefore incorporated into the calculation for v_{stable} ($v_{stable} = mean(v_t, v_{stable})$). If v_t fails the stability criterion, the noisy data is considered to represent a transition rather than noise (Figure 3-4 **G.1**), and the v_{stable} and t_{stable} are reset to the current data and time ($v_{stable} = v_t$ and $t_{stable} = t$).

3.3 Library Modification: Redefinition of Therapy Vector Space

Once an association between a PP input and a vector is made in the $f(assoc)$, the $f(libmod)$ determines how the new association data should be incorporated into the therapy library. The objective of this function is to maintain a therapy library which accurately describes a patient's stimulation therapy space according to changes they have made to their STPs over time. To minimize the amount of computation and storage space necessary, it is important to keep only sufficiently different information. This function first measures the distance between every existing v_{ref} in the therapy library and the v_{assoc} from the $f(assoc)$. The minimum distance between any of the v_{ref} and v_{assoc} is compared to the $f(libmod)$'s same posture threshold, D_{libmod}. If the distance is less than D_{libmod}, then the library entry is considered sufficiently similar, and is replaced by the association data corresponding to the associated PP input. If v_{assoc} is greater than D_{libmod} away from all existing therapy library v_{ref}, then a new entry is created and filled with the association data.

Both the $f(assoc)$ and $f(libmod)$ have same posture thresholds, D_{assoc} and D_{libmod}

respectively. While the same posture thresholds serve similar purposes, their values are independently defined and affect the properties of the final therapy library differently. The current implementation of the $f(libmod)$ uses only the D_{libmod} to determine whether two entires are "sufficiently similar". Suggested improvements for determining whether to replace a library entry or not will be addressed in the conclusion (Chapter 5).

Entry	1	2	3	4	5	6	...
Association #	1	2	3	4	5	6	...
X	$v_{FU}(1)$	$v_{FD}(1)$	$v_R(1)$	$v_L(1)$	$v_{UP}(1)$	$v_{ref1}(1)$	...
Y	$v_{FU}(2)$	$v_{FD}(2)$	$v_R(2)$	$v_L(2)$	$v_{UP}(2)$	$v_{ref1}(2)$	...
Z	$v_{FU}(3)$	$v_{FD}(3)$	$v_R(3)$	$v_L(3)$	$v_{UP}(3)$	$v_{ref1}(3)$	...
Time	t_0	t_0	t_0	t_0	t_0	t_{stable}	...
Rate (Hz)	60	60	60	60	60	60	...
Amp1 (V)	A_1	A_1	A_1	A_1	A_1	A_1	...
Amp2 (V)	A_2	A_2	A_2	A_2	A_2	A_2	...
Amp3 (V)	A_3	A_3	A_3	A_3	A_3	A_3	...
Amp4 (V)	A_4	A_4	A_4	A_4	A_4	A_4	...
PW1 (ms)	PW_1	PW_1	PW_1	PW_1	PW_1	PW_1	...
PW2 (ms)	PW_2	PW_2	PW_2	PW_2	PW_2	PW_2	...
PW3 (ms)	PW_3	PW_3	PW_3	PW_3	PW_3	PW_3	...
PW4 (ms)	PW_4	PW_4	PW_4	PW_4	PW_4	PW_4	...

Table 3.3: Data contained in a therapy library.

Each library entry has 15 components and is uniquely identified by the entry number and the xyz-coordinates. All the necessary STPs required to specify a stimulation in the SCS are stored in the therapy library. Table 3.3 shows the components of a therapy library and an example of information a therapy library may contain. In the table, an entry corresponds to a column of data. The first five entries are the basic v_{OR}, which may be a typical initial library set up. As a patient increasingly makes

programming changes, the library may grow and the v_{OR} entires may be replaced by arbitrary v_{ref} (like Entry 6). The association number is assigned by finding the maximum association number and adding one to it. It uniquely identifies the therapy library entry at a given time. The order which a library entry was added is indicated by both the association number and by the time. If sufficient storage capacity exists, entry data which get replaced by new associations can be saved and used to consolidate the therapy space periodically. This suggestion is mentioned in more detail in the conclusion (Chapter 5).

3.4 Data Classification: Selection of Stimulation Therapy Parameters

The $f(class)$ uses the therapy library to select an appropriate set of STPs to deliver for a given TA data point. While it is important to deliver appropriate therapy at any point in time, it is equally important to filter out noisy data and to avoid delivering sporadic or painful stimulation therapy. For this reason, the $f(class)$ not only measures the relative position of the incoming TA point to existing v_{ref}, but also requires that the data exhibit a trend in its relative position before delivering stimulation therapy associated with a therapy library entry. The method of filtering noise is identical to the method used in the $f(assoc)$ where two variables, K and L, limit the number of extraneous library entry classifications allowed in a segment of L classifications. The parameters K and L dictate how often the STPs will change. Smaller values for L and K enforce a looser requirement for a library entry classification trend. Larger values lead to a delayed response when changing STPs.

The distance between the incoming data point and all existing v_{ref} is measured. The v_{ref} which is closest to the TA data point location is identified and the association number is extracted. The current library association number is compared to a library association number buffer which contains the association numbers of the last L classification made. If the current association number is greater than or equal to K

of the last L association number classifications made, then the STPs for library entry corresponding to that association number are used for the stimulation therapy for the current time, t. Otherwise, the STPs of the last valid library entry classification will be used.

Chapter 4

Assessment of Algorithm Functionality

Since collecting data for a clinical trial was not feasible for the scope of the project, ATLAS performance was characterized using the *3data* and *ICdata*. In addition, outputs from ATLAS were quantitatively and qualitatively compared to BOBAS outputs. Section 4.1 will evaluate the effect of the data association and library modification parameters on the final properties of the therapy library. Section 4.2 analyzes various aspects of the performance of ATLAS relative to BOBAS.

4.1 Effects of Parameters on Data Association Vectors and Therapy Library

As mentioned in Section 2.4.1, optimal stimulation parameters for a given point in three space cannot be asserted using *3data*. Instead, quality of a particular therapy library generated by a patient's PP usage is assessed using several guiding principles. First, we assume that a patient changes his STPs because the current STPs are unsuitable for the posture or activity that the patient is or will be in. While it is sometimes difficult or impossible to know exactly what posture or activity a given programming change was intended for in the *3data*, it is assumed that the change

was not made without reason and that the subject intended the change for a present of future posture or activity, not a previously assumed one. Therefore, for most PP inputs, it is desirable to associate the indicated STPs to some representative vector and to save the information in the therapy library.

A therapy library entry can be replaced several times throughout the course of the study. As an entry gets replaced over time by new v_{assoc}, the spatial similarity between the v_{assoc} and the original v_{ref} is more likely to decrease. For example, in the worst case scenario, after two replacements of a library entry, the final v_{ref} can be a distance of up to $2 \cdot D_{libmod}$ away from the original reference vector, placing it significantly far from it's original position and risking spatial overlap with the v_{ref}s from other library entires. To avoid spatial drifting or spreading of the v_{ref}s pertaining to a given library entry over time, it is important to minimize the spread between vectors which corresponded to the same library entry throughout the course of the study. An identical case can be made for the individual data points which collectively define a v_{assoc}. Ideally, the individual points should be confined to a small region of space, thus increasing the specificity of a given v_{assoc}.

Based on the above arguments, the following therapy space objectives will be used to assess the quality of a therapy library:

O1) Maximize the number of PP inputs which get associated

O2) Minimize the percentage of points labeled as noise within the stable window

O3) Minimize the spread between data points which make up an association vector

O4) Minimize the spread between vectors used to define a single library entry over the course of the study

4.1.1 Parameter Screening Using Design of Experiment

The properties of the data which contribute to the generation of v_{assoc} and of the positions of the v_{ref} within a therapy library are dictated by the parameters of the $f(assoc)$ and $f(libmod)$. To identify the main and interaction effects between

$f(assoc)$ and $f(libmod)$ parameters, a design of experiment (DoE) was used to evaluate the effects of ATLAS parameter values on the content of the final therapy library and the properties of the v_{assoc} generated within the $f(assoc)$. The DoE data was used to select suitable parameter values for subsequent evaluations of ATLAS. The parameter values and responses metrics used to conduct the DoE will be introduced followed by conclusions of the analysis conducted based on the data collected.

Factors

A total of four parameters are used in the DoE to analyze their effect on the final therapy library properties. The parameters, formally referred to in DoE as factors, and are described below:

F1) D_{p2p}: Point to point distance threshold used to check the stability criterion in the $f(assoc)$

F2) D_{assoc}: Same posture distance threshold used to check the same posture criterion in the $f(assoc)$

F3) $MofN$: Two parameters which collectively are used to check the noise criterion in the $f(assoc)$

F4) D_{libmod}: Same posture distance threshold used to check the distance between an new v_{assoc} and existing therapy library entries in the $f(libmod)$

The two timers used in the $f(assoc)$, T_{stable} and T_{search}, were not included in the DoE since the concept and values of these parameters had been previously verified in the creation of BOBAS. The values of the timer parameters were not considered significant factors to vary. For each factor listed above, a minimum, nominal and maximum value was specified. Values for each of three distance metrics were used for the three distance threshold parameters D_{p2p}, D_{assoc}, and D_{libmod}. Since M and N collectively specify the percentage of noisy points allowed in an N-point segment of data, three values (a minimum, a nominal, and a maximum) are used the specify N and values of M are chosen such that the overall fraction is minimal, nominal or

maximal. The numeric values for the $f(assoc)$ and $f(libmod)$ used in the DoE are given on Table 4.1. A random subset of seven of the 15 $3data$ sets were selected to run the analysis on. A full factorial DoE was conducted, where the responses resulting from each permutation of parameter value combinations was measured. Evaluating all permutations for values of the four factors results in 243 (3^5) iterations of the algorithm per data set per distance metric.

	Variable	Units	Minimum	Nominal	Maximum
$f(assoc)$ **Parameters**	D_{p2p}	Angular	3	6	10
		Squared-Euclidean	30	100	400
		Sum-of-Differences	8	18	28
	D_{assoc}	Angular	10	15	20
		Squared-Euclidean	400	700	1000
		Sum-of-Differences	28	40	50
	M	# of points	7 11 14	8 12 16	9 13 18
	N	# of points	12 18 24	12 18 24	12 18 24
	t_{stable}	Seconds	120		
	t_{search}	Seconds	420		
$f(libmod)$ **Parameter**	D_{libmod}	Angular	10	15	20
		Squared-Euclidean	400	700	1000
		Sum-of-Differences	28	40	50

Table 4.1: Design of experiment values. Final factor values used for subsequent analyses are highlighted in yellow. Selection of these values is discussed in the DoE conclusions section.

Responses

To evaluate the four objectives listed in Section 4.1, the following seven responses were collected:

R1) Percentage of associations made: The percentage of search triggers which ultimately get associated. Both components of this response are explained in R1a and R1b. The response is measured as a percentage of R1b with respect to R1a.

R1a. Total number of search triggers: Once a search period has ended (either because an association is made or because the search timer has expired), the search period will be recorded as a search trigger. Initiation of the search

timer is triggered by a new PP input; but, if a new PP input is registered before the search timer expires and before the PP input was associated, then the previous PP input will not be counted as a search trigger. Instead, the two inputs will be considered part of the same programming period. Therefore, the number of search triggers depends both on how quickly associations are made and how long the search and stable timers are.

R1b. Number of associations made: If the association criterion has been fulfilled, the data is used to generate a v_{assoc}. This response counts the number associations made in the $f(assoc)$.

R2) Time required to make an association: In order to make an association, the stability timer must expire; however, depending on how many times the stability timer is re-initiated due to failure of the association criteria, the time between the detection of the PP input and the creation of a v_{assoc} can vary. Only associated PP inputs were considered for this response.

R3) Percentage of noisy points within a stable period: Any point that exceeds the D_{p2p} is automatically flagged and can potentially be classified as either noise or a transition. The percentage of points classified as "noise" within a stable period will be presented. Only associated PP inputs were considered for this response.

R4) Distance between v_{assoc} and each of the data points used to create the vector: The average distance between all non-flagged points and the final v_{assoc} generated will be measured in degrees.

R5) Number of library entries: The number of library entries depends on how similar a new v_{assoc} is to existing library v_{ref}s. The final number of library entries depends on the value of D_{libmod}, the number of associations made and loosely on the position of the v_{assoc}.

R6) Number of associations per library entry: Throughout the study, information in the library can change depending whether a new association replaces an existing

library entry or creates a new entry. This number not only depends on the D_{libmod}, but also on the number of associated PP inputs.

R7) Library entry spread: For library entries that have more than one v_{ref} throughout the duration of the study, the average vector will be calculated. The distance between the average library entry vector and each v_{ref} pertaining to that entry will be measured in degrees. A depiction of this response metric is shown in Figure 4-1.

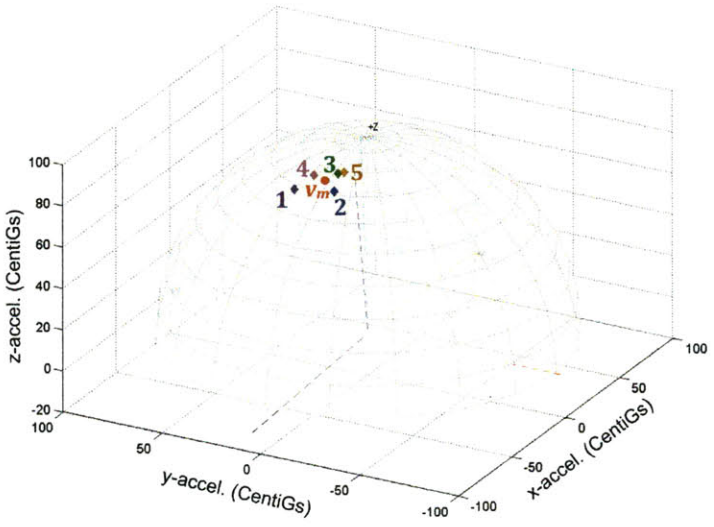


Figure 4-1: Entry spread calculation using a therapy library entry with five vectors over time. The mean vector, indicated by the red circle labeled v_m, is defined as the mean of all the previous v_{ref} corresponding to a particular library entry (points labeled $1 - 5$). The distance between each vector and v_m is then measured, and the metric presented per library entry is the average of these distances.

These seven responses were collected per ATLAS distance metric mode (3) per subject data set (7) per parameter combination (243). Responses are measured after all three ATLAS distance metric modes are run sequentially on one subject data set using one parameter combination. This process is repeated on the remaining data sets. After data responses are collected from each data set for a given parameter combination, the response values across all subjects are averaged together. The average responses from all seven data sets for each distance metric mode using one parameter

combination constitutes one run. A total of 243 runs are required to collect response data for each of the parameter combinations.

Conclusions

Differences between response values due to the unique factor combinations were explored using the MATLAB Statistical Toolbox. Both main effects and interaction effect plots were analyzed. As mentioned in Section 4.1, the performance of an ATLAS output is measured according to whether each of the objectives is appropriately maximized or minimized with respect to the factor value and distance metric used. Objective performance is measured using the values of the responses collected in the DoE. Factor values which produced desirable performance were identified and used for the subsequent ATLAS evaluations described in this chapter.

The effects of the four factors on the responses were observed to have the same general trend for the three distance metrics used. These trends are summarized on Table 4.2. The rows of Table 4.2 indicate the relationship between an objective number and the response number(s) used to evaluate that objective's performance. The arrows next to the objective (O) number indicate the desired direction for improved performance of that objective. Similarly, the arrows next to the response (R) numbers indicate how the value of the response must change in order to achieve the indicated direction of the objective. Double hashed arrows are used for responses which are most relevant for the evaluation of the objective performance and which have a clear direction of change to support an improvement of the objective. The arrows in the F# columns indicate how factor values need to be changed to illicit the indicated change in response value. Factors which have an insignificant effect on the response value have a tilde ($\sim$) symbol instead of an arrow.

Objectives O1– O3 — which deal with the percentage of associated search trigger, the percent noise within an association period, and the distance between data points which make up the final v_{assoc} — depend directly on the performance of the $f(assoc)$. As expected, the value of D_{libmod} (F4) was found to have an insignificant effect on the final values of the responses used to measure the first three objectives. The main effect

O#	R#	F#: Main Effects			
		F1	F2	F3	F4
O1⇑	R1⇑	↑	↑	↓	∼
	R2⇓	↑	↑	↓	∼
O2⇓	R3⇓	↑	↓	↑	∼
O3⇓	R4⇓	↓	↓	↑	∼
O4⇓	R5⇓	∼	∼	∼	↑
	R6↑	↑	↑	↓	↑
	R7⇓	∼	∼	∼	↓

Table 4.2: Relationship between ATLAS therapy space evaluation objectives and the responses used to quantitatively evaluate performance of a given parameter set relative to the objective number. Objectives number descriptions can be found on page 44, response numbers on page 46, and factor numbers on page 45.

plots for responses R1, R3 and R4 are shown in Figure 4-2(a)–(i). The D_{p2p} (F1) has the most significant main effect on the final value of the first four responses; however, as shown in Figure 4-3, there are interaction effects between the first three factors which also must be considered. Regardless of the distance metric used to measure distances in ATLAS, there is a trade off between percentages of search triggers which get associated (R1) and the spread of the data points used to generate a v_{assoc} (R4).

To increase R1 for better performance according to O1, D_{p2p} should be increased. While this decreases the percentage of noisy points included in the association period (R3) and augments performance according to O2, it does so deceptively since the increased D_{p2p} makes it less likely that the stability criterion will fail. As D_{p2p} is increased, points which are further from each other still constitute a stable signal and therefore decrease the specificity of the points which make up a v_{assoc}. This degrades the quality of the v_{assoc} according to O3 and is indicated in Figure 4-2(i)–(g) by the increased association spread due to increasing D_{p2p}. This same trade off is observed across distance metrics. For instance, the maximum percentage of search triggers associated using D is 96.37%, which is the average value for R1 when D_{p2p} is the maximum value (10 degrees). The maximum value achieved for R1 using the S is 98.22% when D_{p2p} is 28. This increased maximum R1 value achieved using S is countered by increased association spread compared to equivalent factor values using D.

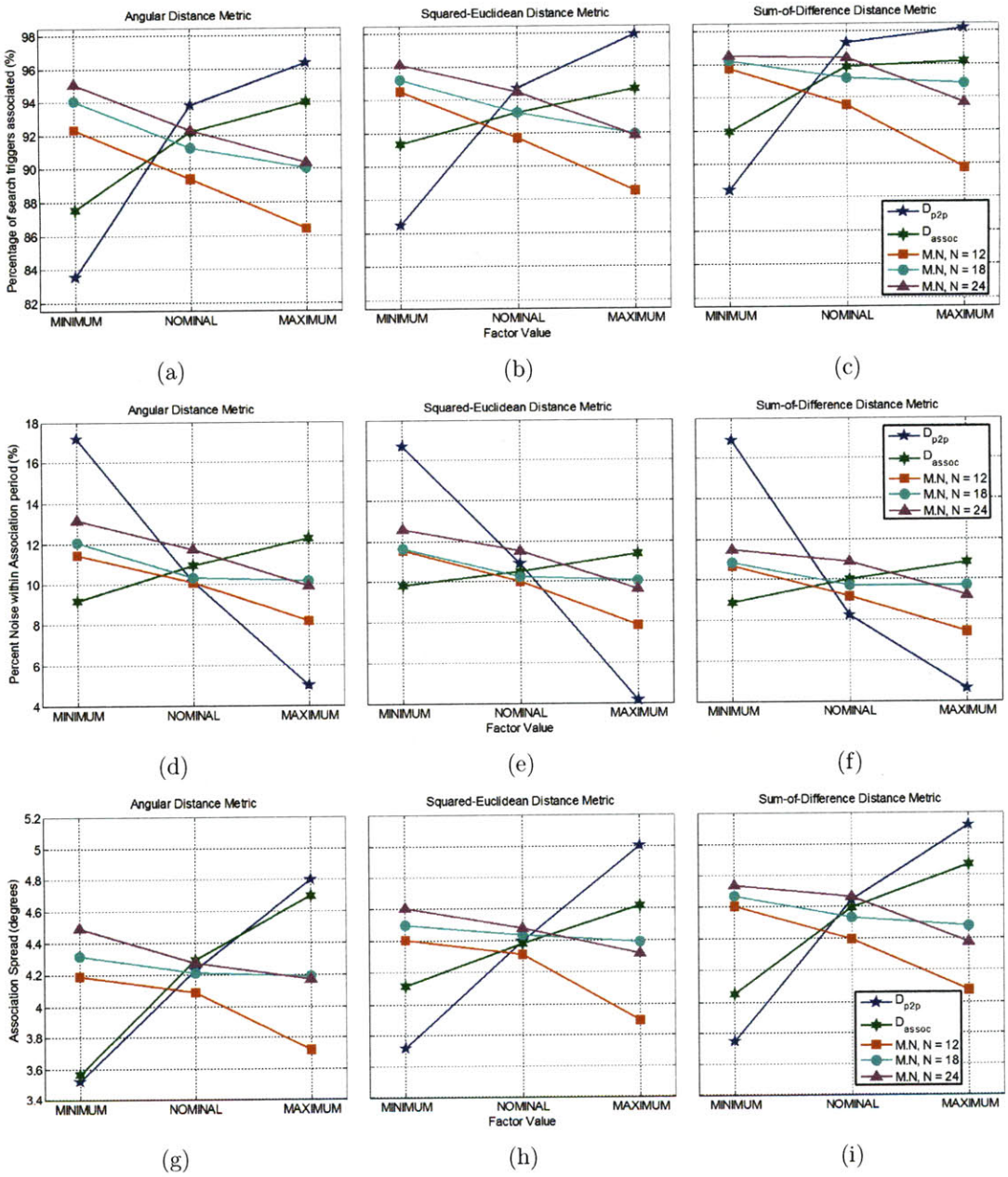


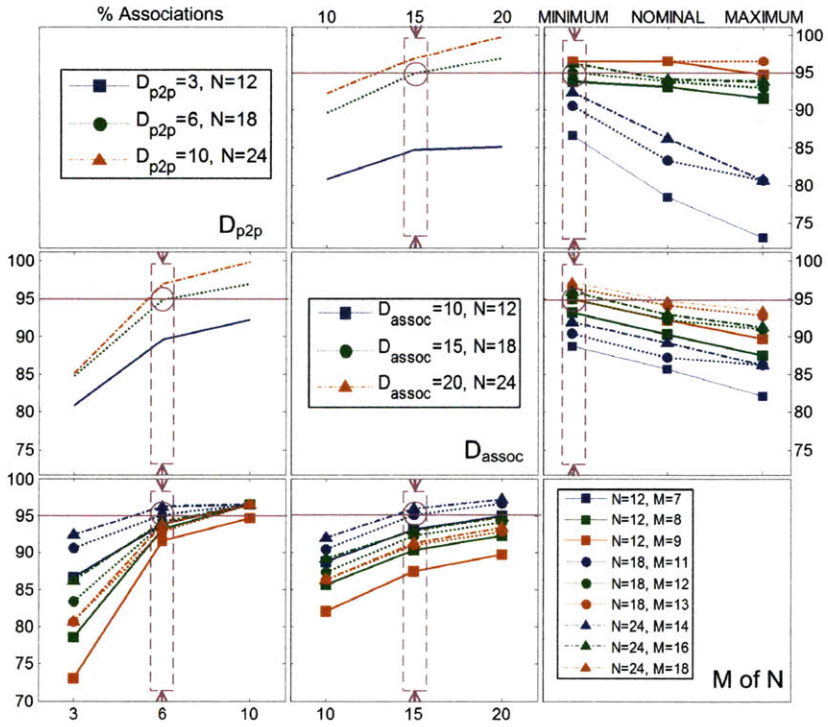
Figure 4-2: Main effect of factors on: (a-c) percent associations (R1), (d-f) percent noise (R3) and (g-i) association spread (R4). The columns correspond to the different distance metrics used (a,d,g)–D, (b,e,h)–E, and (c,f,i)–S. For the minimum, nominal and maximum factor values, see Table 4.1.

Because R1 and R4 are the most straight-forward $f(assoc)$ performance indicators, the interaction plots for these two responses are shown in Figure 4-3(a) and (b) respectively using angular distance metric results. Basic matrix notation will be used

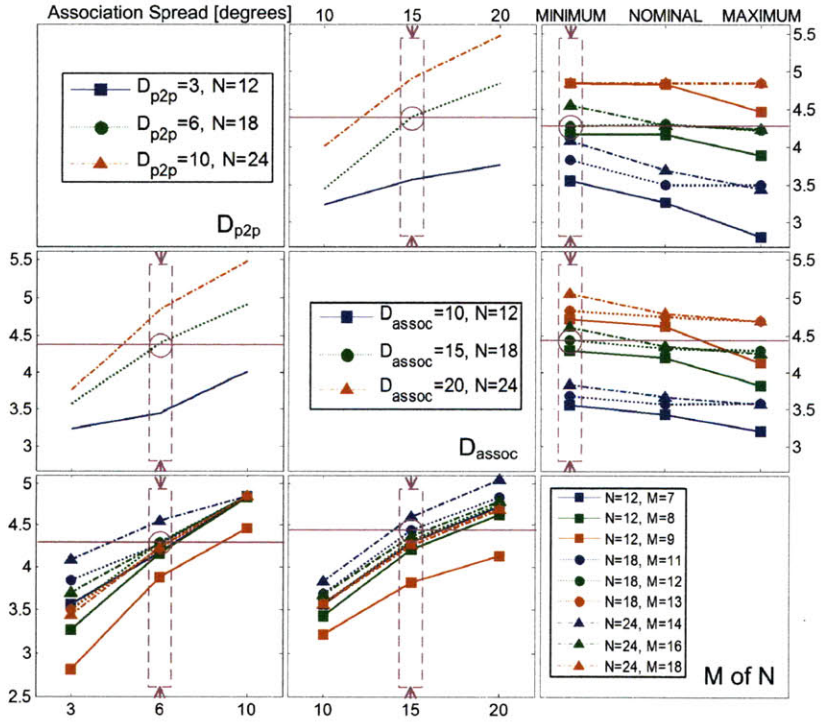
to identify an interaction plot were Figure 4-3$[i, j]$ will refer to the plot in the i^{th} row and j^{th} column. The row numbers indicate that the factor values (Fi) in that row are distinguished using different line styles. Columns corresponding to a factor number (Fj) indicate that the x-axis specifies the three possible factor values. The interaction plots are constructed such that $[i, j]$ and $[j, i]$ have the same information, but with the x-axis and line properties as means of indicating a factor's value switched. The dashed boxes indicate the factor value chosen as optimal for the j^{th} column, and the circles indicate the optimal value corresponding the i^{th} row.

As shown in Figure 4-3(a)$[1, 2]$ and $[1, 3]$, there is significant interaction between D_{p2p} and the other two factors, particularly when considering the difference in response values when D_{p2p} goes from 3 to 6. In $[1, 3]$, we see that as D_{p2p} assumes the nominal and maximum values, the effect of $MofN$ on R1 diminishes. Both plots show that as D_{p2p} is set to its nominal and maximum values, the percent association performance drastically improves. Since the nominal value for D_{p2p} greatly increases the value of R1 while still resulting in relatively good performance for R4, it was selected as the most appropriate value for subsequent evaluations of ATLAS (D_{p2p} = 6 using D, 100 using E and 18 using S).

The minimum value of D_{assoc} which still yielded acceptable response values was chosen assuming the nominal value of D_{p2p}. As shown in Figure 4-3(a)$[2, 1]$, the difference between the value of R1 achieved when the minimum and the nominal D_{assoc} value is used is greater than the difference of R1 achieved between the nominal value and the maximum. A similar relationship between $MofN$ and D_{assoc} is observed in Figure 4-3(a)$[3, 2]$. While the value of D_{assoc} significantly dictates the value of R4, a nominal value of D_{assoc} was chosen as a compromise between an increase in the percentage of association and an increase in the association spread; however, a convincing argument could be made for choosing the minimum value of D_{assoc}, depending on the desired level for R1.



(a)



(b)

Figure 4-3: Interaction effects of the first three factors on (a) percent associations (R1) and (b) association spread in degrees (R4). Interpretation description on page 52.

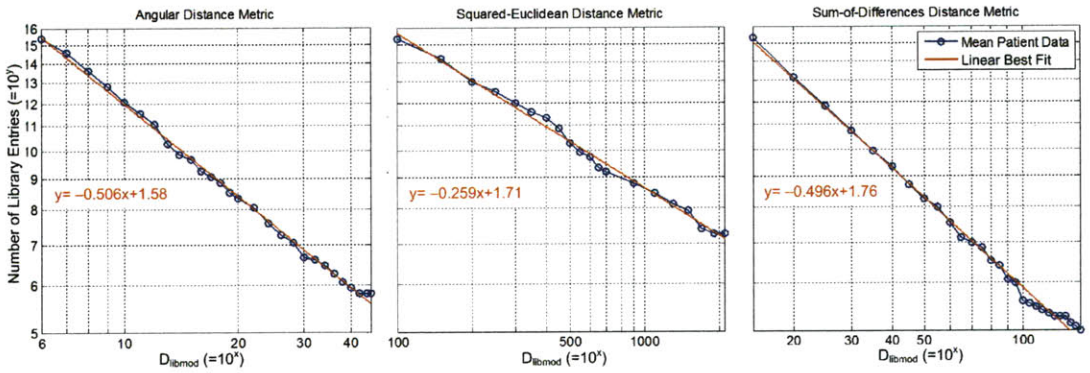
The $MofN$ values were chosen by first comparing the performance between the minimum, nominal and maximum fraction values. When D_{p2p} and D_{assoc} are set to their nominal values, there is only a small gain in performance when going from minimum to maximum fraction values; therefore, only the minimum fraction values were considered since they increase R1 while only marginally affecting R4. The minimum value of N (12) had notably worse performance compared to the nominal (18) and maximum (24) values. Comparing the difference between $N = 18$ (blue line with circular markers) and $N = 24$ (blue line with triangular markers) in Figures 4-3(a) and 4-3(b) [3,2], it is apparent that while there is only a very small difference between the R1 values achieved, the differences between the R4 values are noticeable. By using $N = 18$ (the nominal value), there is a reduction in the association spread with a very small reduction in the percentage of points associated.

Upon analyzing the main and interaction effects on R7, which is the primary performance indicator for O4, it was found that D_{libmod} was the only factor with a significant effect on the final value of the response. The number of library entires in the final therapy library (R5) decreases as the value of D_{libmod} increases since a v_{assoc} and v_{ref} can be further apart, but still be considered "sufficiently similar." This causes the number of associations per library entry (R6) to increase, and in turn the library entry spread increases as D_{libmod} increases. Without knowing the position of the library entires in the therapy space, it is difficult to determine the optimal direction of change for R5 and R6. Therefore, the value of D_{assoc} was selected for nominal performance of library entry spread for each of the distance metrics. The final values selected for each of the factors are highlighted on Table 4.1.

4.1.2 Effect of D_{libmod} on Therapy Library

Based on the DoE results, the D_{libmod} parameter is primarily responsible for the definition of the stimulation therapy space by determining how new associations relate to existing v_{ref} and dynamically redefining the therapy library. To explore the effect of D_{libmod} on the final therapy library content, two of the response metrics measured in the DoE, R5 and R7, were used to evaluate the effect of D_{libmod} on the final properties

of the stimulation therapy space for a range of values. R5 is the resulting number of library entries after using running ATLAS on a 3*data* set. R7 is measured as the average distance between each v_{ref} corresponding to a given library entry and the average vector defined by mean of all the v_{ref} corresponding to a given library entry throughout the course of the study. Responses were collected using each distance metric mode for all subject 3*data* sets.



(a) (b) (c)

Figure 4-4: Effect of D_{libmod} on the final number of library entries in the therapy library for (a) D, (b) E, and (c) S distance metrics.

The average number of library entries produced by ATLAS for all 3*data* sets ($\overline{R5}$) versus D_{libmod} is shown in Figure 4-4. The data is plotted on $\log_{10}$–$\log_{10}$ scaled axes with the linear best fit line (in red) fitted to the mean patient data line (in blue). The linear best fit line, having the standard form of $y = mx + b$, yields the relationship between $\overline{R5}$ and D_{libmod} described in Eqn (4.1).

$$
\begin{aligned}
x &= log_{10}(D_{libmod}) \\
y &= log_{10}(\overline{R5}) \\
log_{10}(\overline{R5}) &= mlog_{10}(D_{libmod}) + b \\
\overline{R5} &= 10^{mlog_{10}(D_{libmod})+b} \\
\overline{R5} &= D_{libmod}^{m} \mid_{d^2} 10^{b}
\end{aligned}
\tag{4.1}
$$

The number of library entries was found to be inversely proportional to approximately

$D_{libmod}^{-1/2}\,|_{D,S}$ for both D and S distance metrics. For E, $D_{libmod}\,|_E$ was raised to the approximately the $-1/4^{th}$ power, due to the fact that squared-euclidean distance was being used. The final relationships between the value of D_{libmod} and the average number of library entries are described by Eqn (4.2), (4.3), and (4.4) for D, E, and S respectively.

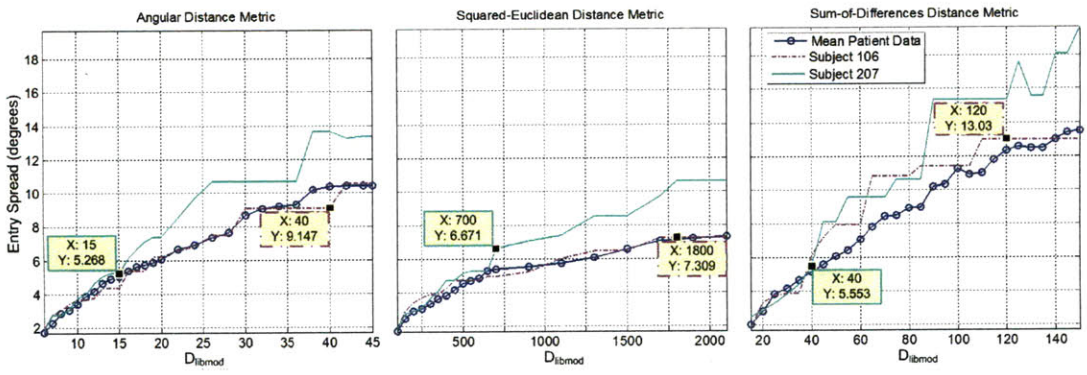
$$\overline{R5} = D_{libmod}^{-0.506}\,|_D\; 10^{1.58} \tag{4.2}$$

$$\overline{R5} = D_{libmod}^{-0.259}\,|_E\; 10^{1.71} \tag{4.3}$$

$$\overline{R5} = D_{libmod}^{-0.496}\,|_S\; 10^{1.76} \tag{4.4}$$

While the number of library entries versus D_{libmod} is comparable for all distance metric modes, the entry spread versus D_{libmod} exhibits very different behavior depending on the distance metric used. In Figure 4-5, the average entry spread, measured as the average of each $3data$ set's individual entry spread, is shown as a blue line with circular markers for D, E, and S distance metrics. The mean patient data curves in Figure 4-5 show that as D_{libmod} increases, the entry spread also increases, however, the rate of increase and the final "saturation values" of the entry spread differ widely across distance metric modes. The final D_{libmod} values used for each of the distance metric modes (45, 2100 and 150 for D, E, and S respectively) are larger than the advised D_{libmod} values; however, the effect on $R5$ as D_{libmod} assumes non-practical levels sheds light on how robust each distance metric mode is. As D_{libmod} approaches 45 using D, the entry spread begins to level off to about 9.15 degrees. Using E, the entry spread begins to level off to 7.31 degrees as D_{libmod} goes to 2100. The entry spread using the S distance metric does not exhibit this same "saturation" behavior. The increase in entry spread as a function of D_{libmod} is approximately linear, with a maximum measured value of 13.55 degrees.

An increase in entry spread indicates a degradation of a given library entry's specificity in three-space and increases the ambiguity of data classification. To explore the fundamental effects of D_{libmod} on the therapy library content over time, two subjects



(a) (b) (c)

Figure 4-5: Effect of D_{libmod} on the entry spread using (a) D, (b) E, and (c) S distance metrics. Entry spread data for two subjects (106 and 207) is shown on each graph. Probed points indicate the values for entry spread (Y) and D_{libmod} (X) which correspond to data shown in Figures 4-6 (subject 106) and 4-7 (subject 207). See page 47, for the definition of "entry spread."

were chosen to conduct a qualitative analysis. Subject 106 was chosen to compare the effects of the different distance metrics at high D_{libmod} values, since the entry spread for this subject's therapy library closely followed the mean entry spread across all subjects. The fundamental differences between the distance metrics as D_{libmod} assumed higher values were explored using Subject 207's $3data$ set. This subject's data set went on to have one of the highest entry spread values as the value of D_{libmod} increased, and was selected to show the differences between a "problematic" subject's therapy library evolution for the operating parameter values highlighted on Table 4.1. The curves for the entry spread versus D_{libmod} for both subjects are shown in Figure 4-5 with data point indicating the entry spread (Y) for each D_{libmod} (X) value used to produce Figures 4-6 and 4-7.

Figures 4-6 and 4-7 show the location of the therapy library entry v_{ref} throughout the course of the qualification protocol for Subjects 106 and 207 respectively. In each figure, the orientation vectors are rotated into the Cartesian coordinate system with R, FD and UP being roughly aligned with the xyz axes. The basic posture abbreviations are shown on the plot for reference. The method used to make these transformations is shown in Section 4.2.1, Eqn. (4.5). This step is done simply to make the 3-D plots

of the data easier to view and does not change any relative properties of the data. The hemisphere roughly defines the location of the TA data points in 3-space, and is meant to provide spatial context to the data. Finally, v_{ref} corresponding to the same library entry are identified by color and marker shape. While the color of a marker always maps to a single library entry, the shape may be repeated since total of eight distinct markers were used. Except for six v_{ref} on Figure 4-7(a), the temporal information of the v_{ref} is not included in Figures 4-6 and 4-7.

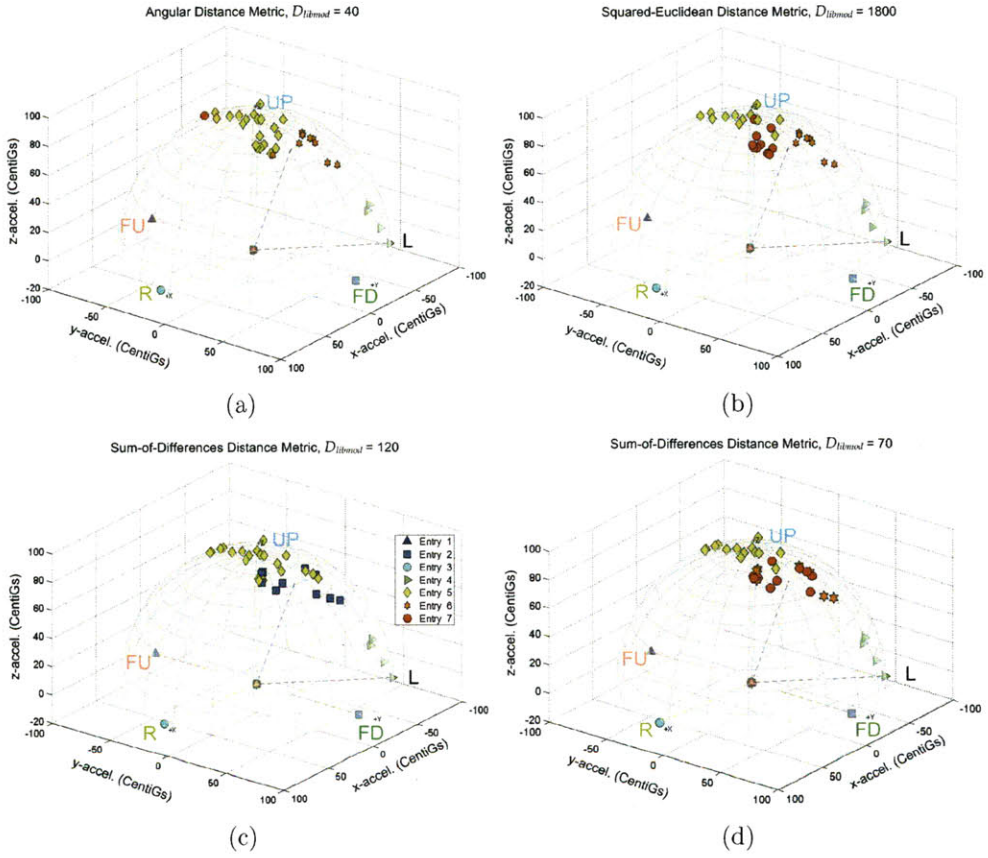


Figure 4-6: Three-dimensional depiction of ATLAS therapy library development on Subject 106 $3data$ using nominal values for all parameters except D_{libmod}. Graphs are shown for (a) $D_{libmod}\,|_D = 40$, (b) $D_{libmod}\,|_E = 1800$, and (c) $D_{libmod}\,|_S = 120$. To compare a value of D_{libmod} using S which yields seven therapy library entires, (d) shows the resulting therapy library using $D_{libmod}\,|_S = 70$.

Figures 4-6(a), 4-6(b), and 4-6(c) correspond to the data stored in the therapy library using $D_{libmod}\,|_D = 40$, $D_{libmod}\,|_E = 1800$, and $D_{libmod}\,|_S = 120$ respectively. These

particular values were chosen since they exhibit the therapy library development as entry spread reaches its near maximum value. The D_{libmod} values mentioned for D and E yield a final therapy library with seven entries; however, the seventh entry using D has only one corresponding v_{ref}. In contrast, the therapy library using the indicated value for $D_{libmod}\,|_S$ results in a final therapy library with only five entires, which is the number of entries the algorithm is initially seeded with (the five basic v_{OR}s). In Figure 4-6(c), we see that v_{ref}s corresponding to Entry 2 extend into the "UP" region, near the Entry 5 v_{ref}s. Comparing the resulting therapy libraries using the different modes of ATLAS, we see that the partitioning of the therapy space by the library entries is done uniquely for each metric. In addition, the differences between the entry spread curves in Figure 4-5 are put into context.

Aside from the fundamental difference between the distance metric modes, the high degree of entry spread when using $D_{libmod}\,|_S$ of 120 could have resulted because the D_{libmod} value was relatively higher than the nominal D_{libmod} value for S than the large values were to the nominal values using D or E. To compare an S therapy library with seven entries for Subject 106, the resulting library with $D_{libmod} = 70$ was generated (Figure 4-6(d)). While the therapy library generated using this value for $D_{libmod}\,|_S$ was much more similar to the therapy libraries generated using D and E, the v_{ref} belonging to Entry 6 and Entry 7 were substantially more mixed than the v_{ref}s belonging to the same entries using E.

Figure 4-7 shows the components of the therapy library throughout the course of the qualification protocol for Subject 207. For all distance metrics, the nominal parameter values highlighted on Table 4.1 were used. A total of 17, 15 and 15 library entries resulted using the D, E, and S distance metrics respectively from the 30 PP inputs made by this subject. For the most part, the v_{ref}s corresponding to a given library entry are the same for the different distance metric used. Discrepancies include the lack of an Entry 14 v_{ref} using E and content of E's Entry 9 compared to D and S. The v_{ref} representing Entry 11 does not exist in the S therapy space since the contents of Entry 7 and Entry 5 differ from the same entries using D or E. Finally, D generates an extra entry, Entry 17, near the "L" Entry 3 v_{ref}s. This association is

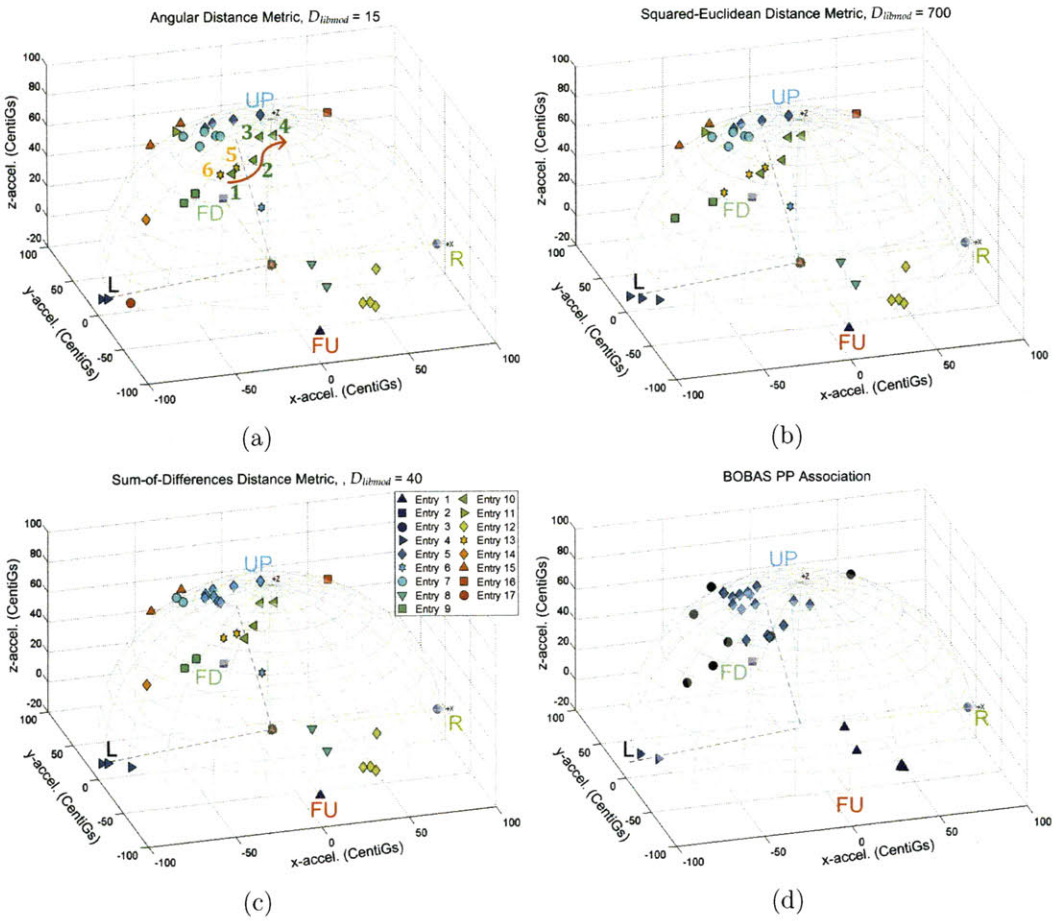


Figure 4-7: Three-dimensional depiction of ATLAS therapy library development on Subject 207 *3data* using nominal values for all parameters except D_{libmod}. Graphs are shown for ((a) $D_{libmod}\ |_D = 15$, (b) $D_{libmod}\ |_E = 700$, and (c) $D_{libmod}\ |_S = 40$. The numbered points shown in (a) indicate the relative temporal order of the six points. (d) The PP input associations made by BOBAS. The black circles represent PP inputs that were not associated. Five of the seven not associated were "classified" as hysteresis.

included in Entry 3 for both E and S. For comparison, the classification of PP inputs into five library entries corresponding to the basic body orientations using BOBAS is shown in Figure 4-7(d).

Looking at the figures, the necessity for some of the library entries may not be clear. For instance, in Figure 4-7 (a–c), Entry 10 and Entry 13 appear to occupy the same region in the therapy space. The order of the PP inputs and the movement of a library entry location upon incorporation of new v_{ref} results in a v_{ref} for a given

library entry that may be distant from the entry's original v_{ref}. The red arrow in Figure 4-7(a) and the numbers along the arrow near the entry points indicate the relative order of the incorporation of that v_{ref} into the therapy library. While the points numbered 1 and 2 are very close to the points labeled 5 and 6, the addition of v_{ref} 3 and 4 move the therapy library v_{ref} sufficiently far from its original position, resulting in the formation of a new library entry once v_{ref} 5 gets incorporated. This phenomenon is known as "entry drift" and occurs because the only criterion used to modify the therapy library is based on distance from the existing library v_{ref}. Strategies to mitigate entry drift will be discussed in the conclusion, Chapter 5.

Based on this analysis, for the D_{libmod} range of interest (see Table 4.1), the performance of the three algorithms is similar. The E distance metric results in the lowest entry spread and S in the greatest. Depending on constraints for number of therapy library entries that can be stored on the SCS device, the relationship between D_{libmod} and number of entries ($\overline{R5}$) described in Eqns. (4.2), (4.3) and (4.4) can be used to help select an appropriate value for D_{libmod}.

4.2 ATLAS versus BOBAS Comparisons

While the four objectives discussed in Section 4.1 were based on reasonable assumptions concerning PP usage, the true performance of the therapy library generated by ATLAS cannot be assessed without programming a pain patient's SCS with the algorithm and monitoring his satisfaction with the resulting stimulation therapy. In order to objectively evaluate the performance of ATLAS, various characteristics of the algorithm's performance will be compared directly to BOBAS. In Section 4.2.1, posture classification performance of ATLAS using a therapy library constructed based on PP inputs will be compared to that of BOBAS using the *ICdata*. Section 4.2.2 will quantify of the number and type of operations required to make an association using ATLAS verses BOBAS.

4.2.1 Posture Detection Accuracy Using In-Clinic Study Data

Since known postures are being assumed in the validation protocol, the *ICdata* provides a means of objectively evaluating the data classification accuracy of a posture detection algorithm. A therapy library will first be generated by ATLAS using the *3data*. All spatial data will then be rotated from *3data* space into *ICdata* space. Each entry of the transformed therapy library will then be identified with a basic posture, and the TA data from the *ICdata* will be classified by ATLAS's $f(class)$ using the therapy library generated from the *3data*. Each data point will be classified to a library entry, which in turn is mapped to a basic orientation. The accuracy of the data classifications made by ATLAS will then be evaluated.

Different accelerometer devices were used in the qualification protocol and the validation protocol. As an initialization step for each protocol, v_{ORS} were defined. Since the v_{ORS} were defined at different times and were generated using different devices, the position of the v_{ORS} defined in the qualification protocol cannot be directly compared to the v_{ORS} defined in the validation protocol. In order to compare posture detection using the therapy library generated by ATLAS on *3data*, the *3data* v_{ORS} need to be mapped with minimal error to the v_{ORS} recorded from the *ICdata* for the same subject. Singular value decomposition [31] is used so that the norm of the difference between the *ICdata* v_{ORS} (A) and the transformed *3data* v_{ORS} (BQ) is minimized (Eqn 4.5).

$$\begin{aligned} min\,\|A - BQ\| &= trace(A^TA) + trace(B^TB) - 2 \cdot trace(Q^TB^TA) \\ U^T(B^TA)V &= \Sigma \\ Z &= V^TQ^TU \\ trace(Q^TB^TA) &= trace(Q^TU\Sigma V^T) = trace(Z\Sigma) \\ Q &= UV^T \end{aligned} \tag{4.5}$$

The order of the orientation vectors and the number used affects how well the two

vector sets are aligned. The mapping is done so that the sum of the angular distance between the $3data$ v_{OR}s and the $ICdata$ v_{OR}s for each of the five orientations is minimized. This is done to optimize the alignment of orientation vectors that are relatively more alike across both the $3data$ and $ICdata$. Different permutations of v_{OR} order and number are tested and the set of vectors which minimizes the distance between the same postures using $3data$ v_{OR}s and the $ICdata$ v_{OR}s is determined. A rotation matrix (Q) is found to minimize the distance between the subset of vectors, and is used to rotate all the v_{ref} within the therapy library into the $ICdata$ vector space.

Once $3data$ v_{OR}s and therapy library v_{ref} have been transformed into the vector space defined by the $ICdata$ v_{OR}s, each association made during the qualification protocol is mapped to a posture: (lying) face up, face down, left, right, hysteresis, or upright. Each of these posture associations are made using the $ICdata$ v_{OR}s. All v_{ref} within θ_{UP} (30°) of either the v_{virtUP} or v_{UP} are mapped to upright. All v_{ref} greater than θ_{LD} (60°) from the v_{virtUP} are considered lying postures and are then mapped to the nearest lying v_{OR}. All associations greater than θ_{UP} but less than θ_{LD} of the v_{virtUP} are classified as hysteresis. After the v_{ref} are mapped to an $ICdata$ orientation, posture detection accuracy is assessed by assigning acceptable postures to each physical task in the validation protocol. Table 4.3 contains a list of all the physical task descriptions and their numbers along with the acceptable postures.

Physical Task (#)	Acceptable Postures
Sit (1, 3, 15)	UP, HYST
Stand (2, 4, 13)	UP,HYST
Treadmill (14)	UP, HYST
Recline (16)	UP, HYST, FU

Physical Task (#)	Acceptable Postures
Face Up (5, 12, 17)	FU
Face Down (7, 10)	FD
Right Side (6, 9)	R
Left Side (8, 11)	L

Table 4.3: Validation protocol physical tasks and associated acceptable postures.

For all distance metric modes used by ATLAS, one classification error occurred, which was the misclassification of physical task (PT) 17 (FU) performed by Subject 101. As shown in Figure 4-8, the library entries used to classify the data corresponding to PT17 (associations 9 and 11, indicated by magenta stars), were both located

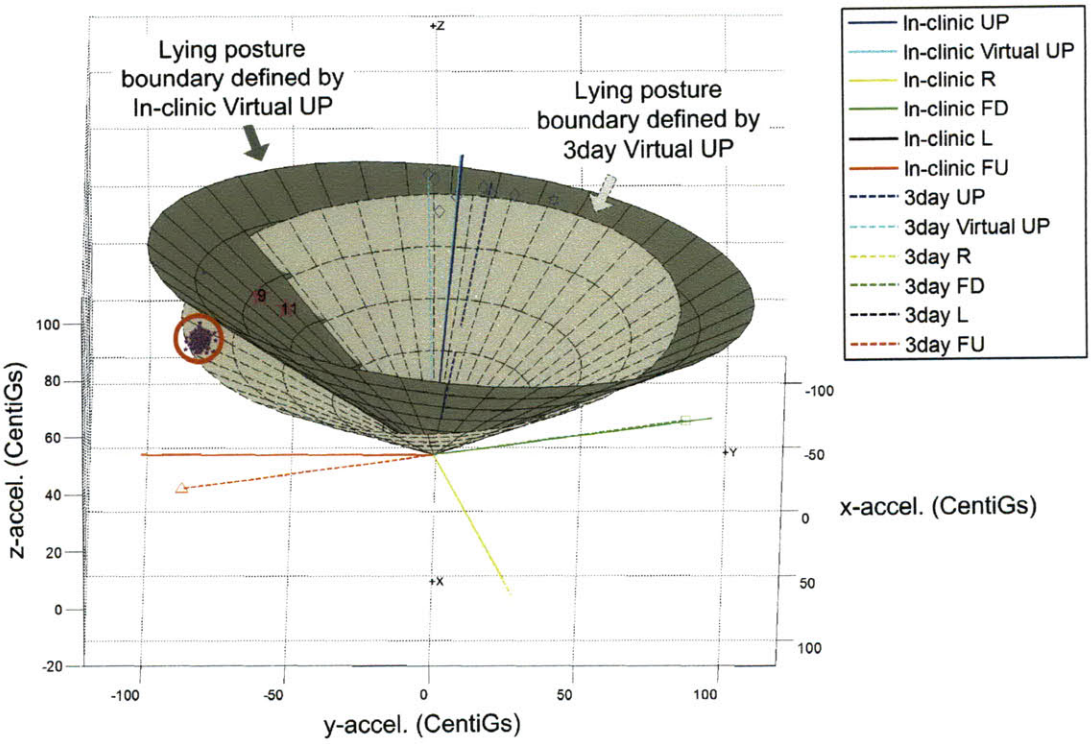
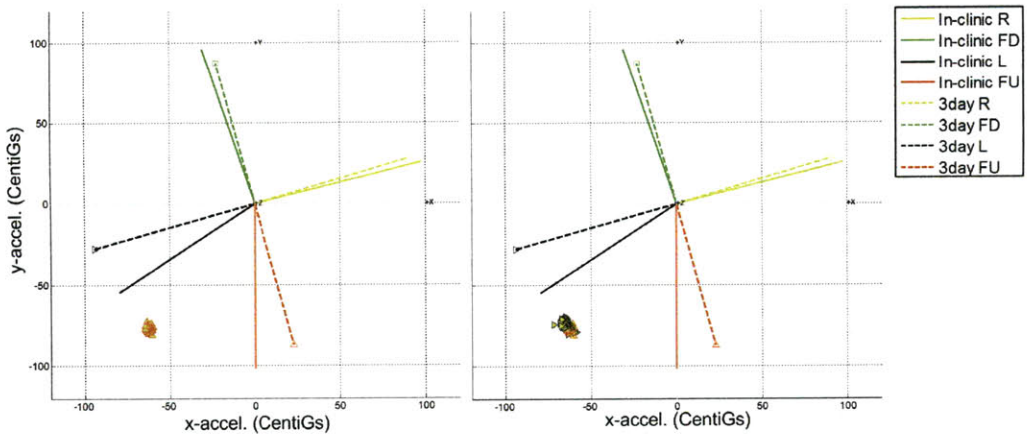


Figure 4-8: Misclassification of PT17, which should be classified as FU according to the validation protocol truth. TA data corresponding to PT17 is circled in red, and the corresponding therapy library entries which were used to classify the data are indicated by their association numbers, 9 and 11, and are represented by magenta stars.

in the hysteresis region, according to the three-space partitioning method used in BOBAS. While not necessarily identified by the same association numbers, all three ATLAS distance metrics modes produced a library entry which had two vectors in the same relative location as those shown in Figure 4-8. The PT17 data points were all associated with this library entry and resulted in a misclassification of PT17 as HYST instead of FU. While the angular distances between PT5 and PT12 and the $ICdata\ v_{FU}$ were 10.35° and 2.63° respectively, the average angle between the PT17 data, circled in red in Figure 4-8 and the $ICdata\ v_{FU}$ was 38.5°. The relatively large distance between $ICdata\ v_{FU}$ and the mean PT17 data indicates that PT17 was in some way different than PT5 and PT12. Therefore, this misclassification is considered insignificant when comparing the posture detection performance of ATLAS and BOBAS.



(a) (b)

Figure 4-9: (a) Shows the fraction of points corresponding to PT8 which were incorrectly classified as FU instead of L using the S distance metric, (b) shows all the data corresponding to PT8. Data classified as FU has a red outline a $\triangle$ marker and data classified as L has a black outline and a $\triangleright$ marker.

One additional misclassification which occurred for the same subject was the classification of 47.9% of the points corresponding to PT8 as FU instead of L using the S distance metric. Figure 4-9 shows the lying $ICdata$ and $3data$ v_{OR}s and PT8 data points projected onto the xy-plane. The PT8 data points which were incorrectly classified as FU rather than L are shown in Figure 4-9(a). All the data corresponding to the PT are shown in Figure 4-9(b). While this misclassification can be partially explained by the transformation error between the $3data$ v_{OR}s and the $ICdata$ v_{OR}s, this same misclassification was not observed in either of the other two distance metric modes (D and E) used by ATLAS. While further analysis would be necessary, this partial misclassification of PT8 observed only when using S could indicate a disadvantage of using S over D or E. Other than the misclassification of PT17 for all distance metric modes of ATLAS and of PT8 using S, the posture classification performance using therapy library v_{ref} was as good as BOBAS.

4.2.2 Computational Complexity

The number of mathematical operations required in the $f(class)$ and $f(libmod)$ functions increases with the number of library entries. This leads to an increase in the

current draw of the microprocessor in the SCS and ultimately depletes the battery life of the device. In order to operate within power dissipation budges, it is important to balance the objectives for system performance and the practical considerations of the system. The computational complexity of each algorithm will be presented as a count of logical, addition, multiplication and trigonometric operations which must be performed by each function in the worst case. Depending on the microprocessor being used, the power cost of each of these operations varies.

Currently, BOBAS is implemented such that the angular distance metric is used to measure the angle between each of the incoming TA data points and the upright vectors, v_{virtUP} and v_{UP}. A "posture code" is assigned to each of the TA data points and depending on whether a PP input is being associated or not, the posture code information is used by either the data association function or the data classification function. Because the number of library entries is fixed and this same computation is performed regardless of algorithm state, BOBAS requires 7 logical, 24 addition, 27 multiplication, and 5 trigonometric operations per TA data point in the worst case. A few minor logical operations are also required in BOBAS $f(assoc)$.

The associated mathematical operation counts for each distance metric used by ATLAS are given on Table 4.4. The computational complexity of D > E > S. The number of operations required to complete each of the ATLAS functions are shown on Table 4.5. Depending on how the distance between two vectors is measured (d^2), the number of operations required varies. While the number of operations in $f(class)$ and $f(libmod)$ depend on the number of library entries (ℓ), the $f(assoc)$ depends only on whether certain criterion have been passed or failed. The worst case number of operations for $f(assoc)$ is shown on Table 4.5, where all three stability, noise, and same posture criteria must be checked.

d^2	Logicals	Additions	Multiplications	Trigonometric
D	0	6	11	1
E	0	5	3	0
S	3	5	0	0

Table 4.4: Number of mathematical computations required for each distance metric.

	Logicals	Additions	Multiplications	Trigonometric	d^2
$f(assoc)$	5	5	1	$\sim$	2
D	5	18	23	1	$\sim$
E	5	16	7	0	$\sim$
S	8	16	1	0	$\sim$
$f(libmod)$	$\ell - 1$	0	0	$\sim$	ℓ
D	$\ell - 1$	6ℓ	11ℓ	ℓ	$\sim$
E	$\ell - 1$	5ℓ	3ℓ	0	$\sim$
S	$4\ell - 1$	5ℓ	0	0	$\sim$
$f(class)$	$\ell - 1$	0	0	$\sim$	ℓ
D	$\ell - 1$	6ℓ	11ℓ	ℓ	$\sim$
E	$\ell - 1$	5ℓ	3ℓ	0	$\sim$
S	$4\ell - 1$	5ℓ	0	0	$\sim$

Table 4.5: Computational operations required for each of the ATLAS functions. The $f(assoc)$ and $f(libmod)$ depend on the number of library entires, ℓ.

While the actual specifications for current draw per mathematical operation depends on the microprocessor used, to quantify the differences between the BOBAS and ATLAS algorithms, MATLAB (v7.8) functions `tic` and `toc` were used to measure the time required to run BOBAS and ATLAS on each 3*data* set. Data was collected using an HP Pavilion tx2500 machine with an AMD Turion™ X2 Dual-Core Mobile RM-70, 2GHz, 2.75 GB of usable RAM, and 32-bit Windows 7 operating system. The results, though somewhat surprising, were encouraging. The average times required to run each algorithm on the 3*data* sets was 48.27, 17.51, 17.08, and 17.49 seconds per data set using BOBAS, and ALTAS D, E, and S algorithms respectively. It was expected that ATLAS would require a longer time for completion, given that there would be more library entries than when using BOBAS. The result, however, was that ATLAS compiled the 3*data* at least 2.76 times faster than BOBAS. This result could be explained by the fact that BOBAS was written by a previous author and was not optimized for use in this thesis. ATLAS was written originally and used

different methods to compute distances and structure data. A result somewhat more difficult to explain is the difference between the D, E, and S run-times. This analysis, however, could have been affected by many factors, such as CPU usage variation, and optimization of certain mathematical functions in MATLAB. With these considerations in mind, the run-time results for each of the ATLAS distance metric modes are very similar indicating a minimal cost associated with choosing one metric over another.

Chapter 5

Conclusion

The concept of body orientation based stimulation, long acknowledged as an unmet functional necessity in SCS devices [28], was not incorporated into a SCS device until Medtronic's development of BOBAS. While patient satisfaction with algorithmically derived STPs was higher than manual STP adjustment, the opportunity to even further improve the closed-loop functionality was investigated. In this work, development of an Advanced Therapy Learning Algorithm for Stimulation (ATLAS), funded by Medtronic, has been proposed. ATLAS allows patients to individualize their stimulation therapy so that they receive comfortable, therapeutic stimulation at all times according to their individual needs. ATLAS develops the fundamental aspects needed to implement such an algorithm. From determining how to identify and save new PP input to investigating the effect of different distance metrics, the tri-axial accelerometer based algorithm proposed in this work has the potential to improve the current five-posture-based detection method currently used.

The following work has been detailed in this thesis:

1. Development of a tri-axial accelerometer based algorithm used to define a stimulation therapy space for a SCS device using patient programmer input

2. Characterization of three distance metrics – angular, squared-euclidean, and sum-of-differences – and their affects on the final properties of a therapy library developed when used by ATLAS

3. Method for comparing data acquired from two different studies

It was found that by using the highlighted values on Table 4.1 for the six data association function parameters and the library modification parameter, an average of 9.67, 9.20, and 9.33 library entries are created with entry spreads of 5.10, 5.41, and 5.17 using angular, squared-euclidean, and sum-of-differences distance metrics respectively. Due to nature of the data available for analysis, the quality and accuracy of the STPs selected using the therapy library produced by ATLAS could not be verified. When compared to the validated posture detection accuracy of BOBAS, however, ATLAS performed as well as BOBAS in most cases.

Based on the analysis conducted, the squared-euclidean distance metric is suggested due to its robust design in terms of therapy library entry spread and low computational complexity with respect to the angular distance metric. The highlighted values indicated on Table 4.1 yielded desirable results for subsequent analyses, however, it should be assessed whether the specificity of the association vectors generated in the data association function or the percentage of associated patient programmer inputs should be optimized. As mentioned in Section 4.1.1, there is a trade off between these two objectives when setting D_{libmod} to the minimum value versus the nominal value.

The major limitation of the data available was the inability to make a correlation between posture and preferred stimulation therapy parameters (particularly stimulation voltage). An extensive study either requiring subjects to continually optimize their therapy real-time or using an algorithm to select suitable STPs for a given subject state (such as posture) and assessing whether changes in STPs are required would be beneficial in determining the therapeutic ranges associated with arbitrary regions in three-space. The $ICdata$ did provide information of this nature for the five-basic orientations, but since ATLAS proposes to more finely and arbitrarily partition three-space, this data could not be used for validation. Other complications include accounting for day-to-day variations in preferred stimulation parameter levels. To move forward, it would be beneficial to collect data implementing both ATLAS and BOBAS for an extended period of time and assessing patient satisfaction during that

period.

Depending on the memory and computational capacity of the microprocessor, more advanced methods for library modification and data classifications are suggested. For the library modification function, these strategies may be used to mitigate proliferation of redundant therapy library entries, decrease entry spread over time, and group entries that are more alike in multiple dimensions (not merely spatial). Other suggestions are made in regards to STP selection. Improvements are listed below:

- Rather than completely replacing a "similar" therapy library with association data, the previous v_{ref} and the new v_{assoc} could be averaged together to lessen the entry drift.

- Use additional parameters to dictate therapy library refinement after an association has been made. Similarity could be assessed on the basis of stimulation therapy parameter similarity (such as stimulation voltage amplitude) or the amount of time a given library entry was used without being modified. Cost functions could potentially be developed in order to decide how a new set of association data should be incorporated into the therapy library.

- Keep a record of previous PP inputs and after some elapsed time period (on the order of days), aggregated the data into a compressed therapy library which interprets the relationship between PP inputs and defines the therapy space accordingly.

- Classify data according to its relative distance to n neighboring therapy library reference vectors. The STPs could be calculated as a fraction of the STPs designated by the v_{ref}s in the same vicinity as the TA data point.

- The therapy library can be organized such that library entries that are spatially close together are grouped. When data classification is performed for a new TA data point, instead of calculating the distance between the data point and each library entry v_{ref}, different parts of the library, corresponding to distant regions in three-space, could be checked sequentially. As soon as the relative region of

the data point is determined, the STPs can be selected from the library entries within that region.

Appendix A

Glossary

3data	Three-day qualification protocol data (Section 2.4.1)
association	*xyz*-coordinates of an association vector and the associated set of stimulation therapy parameters
association criteria	Criteria that must be met in order for a valid reference vector to be generated in the data association function. Individual stability, noise, and same posture criteria must be met.
ATLAS	Advanced Therapy Learning Algorithm for Stimulation
BOBAS	Body Orientation Based Algorithm for Stimulation
CSF	Cerebral Spinal Fluid
D	Angular distance metric (Table 3.2)
D_{assoc}	Data association function same posture distance threshold
D_{libmod}	Library modification function same posture distance threshold
D_{p2p}	Data association function point to point distance threshold
E	Squared-Euclidean distance metric (Table 3.2)
FD	Lying Face Down (prone)

FU	Lying Face Up (supine)
$f(assoc)$	Data Association Function
$f(class)$	Data Classification Function
$f(libmod)$	Library Modification Function
$ICdata$	In-clinic validation protocol data (Section 2.4.2)
IPG	Implantable Pulse Generator
INS	Implantable Neural Stimulator
L	Lying Left Side
noise criterion	Requires that M of the last N points in the data association function pass the stability criterion, or the same posture criterion if separated by noise. Essentially imposes a noise ceiling for an N-point segment of data.
PP	Patient Programmer
R	Lying Right Side
S	Sum-of-Differences distance metric (Table 3.2)
same posture criterion	$d^2(v_{stable}, v_t) \leq D_{assoc}$
SCS	Spinal Cord Stimulator
SPT	Same Posture Threshold
STP	Stimulation Therapy Parameters (amplitude, pulse width, rate, and electrode polarities)
stability criterion	$d^2(v_{t-1}, v_t) \leq D_{p2p}$
TA	Tri-axial Accelerometer
therapy library	Data structure which contains information used to select STPs.
therapy space	The spatial position of reference vectors within a therapy library which influence how three-space is partitioned.
therapy space objectives	See page 44
UP	Upright (standing)
v_{ref}	Reference Vector (xyz-coordinates)

Appendix B

MATLAB Scripts

B.1 ATLAS Main Function

```
function [ATLAS] = ATLAS_MainFctn(subj_num,varargin)

% Script Requirements : get_patient_data, group_valid_pp_data,
% seed_library, initialize_ClassifyParams, initialize_ATLASparams,
% initialize_AssocVars, LibraryMod_MainTool,
% fassoc, classify_data, modify_libary

%% Initialize Libraries
d2 = [1,2,3];
library = cell(0,3);
distances = {'degrees','edist','sdiff'};

%% Initialize Sensor Data Variables
patient_data = get_patient_data(subj_num);
pp_data = group_valid_pp_data(subj_num);
all_pp_data = pp_data.all(2:end,:);
pptimes = cell2mat(pp_data.all(2:end,1));
pptype = pp_data.all(2:end,2);

%% Initialize Sensor Data Variables
pos_data = patient_data.sensor_data.pos_data;
data_times = patient_data.sensor_data.rsdate;
npnts = length(data_times);

for e = 1:length(d2)
  %% Initialize Library Parameters and Variables
  ATLASparams.distmeasure = distances{d2(e)};
  [ATLASparams,searchtimer,stabletimer,Dp2p,Dassoc,Dlibmod,M,N,distmeasure]...
    = initialize_ATLASparams(ATLASparams);
  saveParams.searchtimer = searchtimer;
  saveParams.stabletimer = stabletimer;
  saveParams.Dp2p(d2(e)) = Dp2p;
  saveParams.Dassoc(d2(e)) = Dassoc;
  saveParams.Dlibmod(d2(e)) = Dlibmod;
  saveParams.M = M;
  saveParams.N = N;

  if strmatch('seed',PropertyNames)
    library_seed_instructions = PropertyVal{strmatch('seed',PropertyNames)};
  else
    library_seed_instructions  = [1,1];
    % default instructions indicate to seed the library with the
```

```
    % orientation vectors using virtual upright instead of the
    % recorded/actual upright
  end

  library_seed = seed_library(subj_num,library_seed_instructions);
  %seeds the library with the five basic orientation vectors
  all{d2(e)} = library_seed;
  nassociations = all{d2(e)}(end,1);
  library{d2(e)} = library_seed;
  entry_history{d2(e)} = cell(1,30);
  for l = 1:size(library{d2(e)},1)
    entry_history{d2(e)}{l} = library{d2(e)}(l,1);
  end


  %% Initialize Data Classification Parameters and Variables
  ClassifyParams.distmeasure = distances{d2(e)};
  [ClassifyParams,Dlibmod,L,K,thigh2low,tlow2high] = ...
    initialize_ClassifyParams(ClassifyParams);

  ClassifyData{d2(e)}.association_ID = zeros(npnts,6);
  ClassifyData{d2(e)}.LibEntry_ID = zeros(npnts,6);
  ClassifyData{d2(e)}.LibEntry_ID(:,1:2) = 1;

  %% For each incoming data point
  current_data = [];
  current_time = [];
  i = 0;

  %% Initialize AssocVars
  % Data Variables
  AssocVars = initialize_AssocVars;
  while i < npnts
    i = i + 1;
    current_data = pos_data(i,:);
    current_time = data_times(i);
    t0ppinput = AssocVars.t0ppinput;
    [Fnew,ppID] = serviceppinput(current_time,t0ppinput);
    if length(ppID) > 1
      ppID = ppID(end);
    end
    AssocVars.Fnew = Fnew;
    if Fnew
      %CALLS TO DATA ASSOCIATION FUNCTION----------------------------------------------------------
      if ¬isempty(ppID)
        if strmatch(pptype(ppID),'Stim On')
          %disp('Stim turned back on')
          if ppID > 1 & strmatch(pptype(ppID-1),'Stim Off')
            AssocVars.ppID = ppID;
            [AssocVars] = ...
              fassoc(ATLASparams,AssocVars,current_time,current_data,i);
          elseif ppID == 1
            AssocVars.ppID = ppID;
            [AssocVars] = ...
              fassoc(ATLASparams,AssocVars,current_time,current_data,i);
          else
            Fnew = 0;
            AssocVars.t0ppinput = current_time;
            AssocVars.Fnew = Fnew;
            [AssocVars] = ...
              fassoc(ATLASparams,AssocVars,current_time,current_data,i);
          end
        else
          if strmatch(pptype(ppID),'Stim Off')
            % Once stim is turned off, there should be a routine
            % that checks for the following stim on command AND
            % disables all output stim_params sent to the device.
            % Once the following stim on command is recieved, it
            % should immediately begin classifying data again
            % according to exisiting library entries.
```

```
            all_pp_data(ppID,5:12) = num2cell(zeros(1,8));
          end
          AssocVars.ppID = ppID;
          [AssocVars] = ...
            fassoc(ATLASparams,AssocVars,current_time,current_data,i);
        end
      else
        disp('ppID is empty')
      end
    elseif AssocVars.Fassoc
      [AssocVars] = ...
        fassoc(ATLASparams,AssocVars,current_time,current_data,i);
    end
    %-------------------------------------------------------------------------

    if ¬AssocVars.Fassoc & ¬AssocVars.Fnew
      %% If a new association was made (which occurs if stable
      % criteria is fulfilled AND the stability timer has EXPIRED
      % (which means the current point does not contribute to the
      % data that comprises the association reference vector))
      if AssocVars.Flibmod
        % First decide what to do with the new association => replace a
        % library entry to create a new one
        nassociations = nassociations + 1;
        nLibEntries = size(library{d2(e)},1);
        newstate.data = AssocVars.stabledata;
        newstate.time = AssocVars.t0ppinput;
        ppID = AssocVars.ppID;
        newstate.group = cell2mat(all_pp_data(ppID,3));
        newstate.amplitude = cell2mat(all_pp_data(ppID,5:8));
        newstate.pulse_width = cell2mat(all_pp_data(ppID,9:12));
        newstate.rate = cell2mat(all_pp_data(ppID,4));
        newstate.ppID = ppID;
        newstate.entry = [nassociations,newstate.data,0,newstate.time,...
          newstate.group,newstate.rate,newstate.amplitude,...
          newstate.pulse_width,newstate.ppID];
        current_state_vector = newstate.data;
        if nassociations == 1
          library{d2(e)} = [];
          LibEntry2replace = 1;
        else
          existing_state_vectors = library{d2(e)}(:,2:4);
          all_state_vectors = all{d2(e)}(:,2:4);
          % CALL TO LIBRARY MODIFICATION FUNCTION-------------------------------
          [LibEntry2replace,entries_within_posthresh] =...
            flibmod(all_state_vectors,existing_state_vectors,...
            current_state_vector,ATLASparams);
          %-------------------------------------------------------------------
        end

        all{d2(e)}(nassociations,:) = newstate.entry;

        if isempty(entry_history{d2(e)}{LibEntry2replace})
          %means a new library entry was created
          entry_history{d2(e)}{LibEntry2replace} = ...
            nassociations;
        else
          %means that the association replaced information in an existing
          %library entry
          entry_history{d2(e)}{LibEntry2replace} = ...
            [entry_history{d2(e)}{LibEntry2replace},nassociations];
        end

        library{d2(e)}(LibEntry2replace,:) = newstate.entry;
        assoc_ind = find(data_times == AssocVars.t0stable):i;

        AssocVars = initialize_AssocVars;
        AssocVars.t0ppinput = current_time;

      elseif ¬skip_fclass
```

```
        %% Classify Current Data Point
        if i > L
          bufferpnts = ClassifyData{d2(e)}.LibEntry_ID(i-L:i-1,1);
        else
          bufferpnts = [ClassifyData{d2(e)}.LibEntry_ID(1:i,1);1;1;1;1;1];
          bufferpnts = bufferpnts(1:5);
        end

        if i == 1
          prev_stim_LibEntry = 1;
          prev_stim_assoc = 0;
        else
          prev_stim_LibEntry = ClassifyData{d2(e)}.LibEntry_ID(i-1,2);
          prev_stim_assoc = ClassifyData{d2(e)}.association_ID(i-1,2);
        end

        %CALL TO DATA CLASSIFICATION FUNCTION----------------------------------------
        [library,dataclass_assocID,dataclass_libID,stim_assocID,stim_libID,...
          ramptime] = fclass(library,ClassifyParams,bufferpnts,...
          prev_stim_LibEntry,prev_stim_assoc,current_data);
        %---------------------------------------------------------------------------
        ClassifyData{d2(e)}.association_ID(i,1) = dataclass_assocID;
        ClassifyData{d2(e)}.association_ID(i,2) = stim_assocID;
        ClassifyData{d2(e)}.association_ID(i,3:6) = ramptime;
        ClassifyData{d2(e)}.LibEntry_ID(i,1) = dataclass_libID;
        ClassifyData{d2(e)}.LibEntry_ID(i,2) = stim_libID;
        ClassifyData{d2(e)}.LibEntry_ID(i,3:6) = ramptime;
        if stim_assocID ~= 0
          all{d2(e)}(stim_assocID,5) = all{d2(e)}(stim_assocID,5)+1;
        end
      end
    end
  end

  %% Make the output Variables more and store as cells instead of
  % matrices with columns labeled appropriately
  nentries = size(library{d2(e)},1);
  entry_history{d2(e)} = entry_history{d2(e)}(1:nentries);

  lib_data = library{d2(e)};
  library{d2(e)} = cell(nentries+1,17);
  library{d2(e)}(1,:) = {'ENTRY','X','Y','Z','CNT','TIME','GROUP','RATE',...
    'AMP1','AMP2','AMP3','AMP4','PW1','PW2','PW3','PW4','ppID'};
  library{d2(e)}(2:end,:) = num2cell(lib_data);

  all_data = all{d2(e)}(1:nassociations,:);
  all{d2(e)} = cell(nassociations+1,17);
  all{d2(e)}(1,:) = {'ENTRY','X','Y','Z','CNT','TIME','GROUP','RATE',...
    'AMP1','AMP2','AMP3','AMP4','PW1','PW2','PW3','PW4','ppID'};
  all{d2(e)}(2:end,:) = num2cell(all_data);

  class_data_assoc = ClassifyData{d2(e)}.association_ID;
  class_data_lib = ClassifyData{d2(e)}.LibEntry_ID;
  ClassifyData{d2(e)}.association_ID = cell(npnts+1,6);
  ClassifyData{d2(e)}.LibEntry_ID = cell(npnts+1,6);
  ClassifyData{d2(e)}.association_ID(1,:) = {'Nearest Association ID',...
    'ID for Stimulation Parameters',...
    'Ramp time (seconds) for program 1','Ramp time (seconds) for program 2',...
    'Ramp time (seconds) for program 3','Ramp time (seconds) for program 4'};
  ClassifyData{d2(e)}.LibEntry_ID(1,:) = {'Nearest Entry ID',...
    'ID for Stimulation Parameters',...
    'Ramp time (seconds) for program 1','Ramp time (seconds) for program 2',...
    'Ramp time (seconds) for program 3','Ramp time (seconds) for program 4'};
  ClassifyData{d2(e)}.association_ID(2:end,:) = num2cell(class_data_assoc);
  ClassifyData{d2(e)}.LibEntry_ID(2:end,:) = num2cell(class_data_lib);
```

```
end

  function [Fnew,ppID] = serviceppinput(t,t0ppinput)
    % if a button has been pressed, return 1 for newppinput; otherwise,
    % return the past value of newppinput
    % This routine should be serviced every for everysample
    ppID = find(pptimes > t0ppinput & pptimes ≤ t);
    if isempty(ppID)
      Fnew = 0;
    else Fnew = 1;
    end
  end

ATLAS.library = library;
ATLAS.all = all;
ATLAS.ClassifyData = ClassifyData;
ATLAS.entry_history = entry_history;
ATLAS.subj_num = subj_num;
ATLAS.virtualup = library_seed_instructions(2);
ATLAS.ATLASparams = saveParams;

end
```

B.2 Data Association Function

```
function [updatedAssocVars] = ...
  fassoc(ATLASparams,AssocVars,current_time,current_data)

% Input variables ATLASparams, AssocVars, current_time, current_data,
% detailed_data, and data_pnt_ind are passed in from the ATLAS main
% function script and indicate the current state of the f(libmod).
%
% Script Requirements : initialize_AssocVars, initialize_ATLASparams,
% dist_from_vector

%% Load previous state data and split up parameters
[stabledata,prevdata,Bp2p,p2pbuffertimes,bufferdata,Flibmod,...
  Fnew,Fassoc,insearchperiod,instableperiod,t0ppinput,t0search,...
  t0stable,ppID] = initialize_AssocVars(AssocVars);
[ATLASparams,searchtimer,stabletimer,Dp2p,Dassoc,Dlibmod,M,N,distmeasure]...
  = initialize_ATLASparams(ATLASparams);
fs = 5.125; %sampling frequency for 3data

if Fnew
  %% Service new pp input and appropriatly set time variables
  t0ppinput = current_time;
  t0search = current_time;
  t0stable = current_time;
  stabledata = current_data;
  bufferdata = nan(N,3);
  bufferdata = [bufferdata(2:end,:);current_data];
  Bp2p = ones(1,N);
  p2pbuffertimes = ones(1,N);
  p2pbuffertimes = [p2pbuffertimes(2:end),current_time];
  Flibmod = 0;
  Fnew = 0;      %**SHOULD BE REDUNDANT
  Fassoc = 1;
elseif Fassoc
  %% Associate or continue associating new pp input
  insearchperiod = (current_time - t0search) < searchtimer;
  if insearchperiod %search time HAS NOT expired
    instableperiod = (current_time-t0stable) < stabletimer;
    if instableperiod %stable time HAS NOT expired
      %% This means that stability timer has not expired, therefore,
      % incoming accelerometer data should be tested for association
      % criterion
      p2pdist = dist_from_vector(current_data,prevdata,distmeasure(1));
```

```
      stablecrit = p2pdist ≤ Dp2p; %STABILITY CRITERION
      Bp2p = [Bp2p(2:end),stablecrit];
      p2pbuffertimes = [p2pbuffertimes(2:end),current_time];
      bufferdata = [bufferdata(2:end,:);current_data];
      if sum(Bp2p) ≥ M %NOISE CRITERION
        %% M of N points are within the point to point threshold
        if Bp2p(end-1) == 1 & stablecrit
          % the last point also fulfilled the stable criterion, so we can
          % automatically add current data point to the stable data vector
          stabledata = mean([stabledata;current_data]);

        elseif Bp2p(end-1) == 0 & stablecrit
          % the last point failed the stability criterion so we must
          % check if the new point is sufficiently similar to the last
          % recorded data points to be added to the stable data variable
          LTp2pind = find(Bp2p(1:end-1)==1);
          stablepntdist = ...
            dist_from_vector(stabledata,current_data,distmeasure(1));
          sameposcrit = stablepntdist ≤ Dassoc; %SAME POSTURE CRITERION
          if sameposcrit
            % Indicates the two portions of accelerometer signal separated
            % by noise are similar enough to be considered the same posture
            stabledata = mean([stabledata;current_data]);
          elseif ¬sameposcrit % Indicates a "transition"
            Bp2p = [ones(1,N-1),stablecrit];
            p2pbuffertimes = [ones(1,N-1),current_time];
            bufferdata = [ones(N-1,3);current_data];
            stabledata = current_data;
            t0stable = current_time;
          end %EO same posture criterion IF
        elseif ¬stablecrit
          % Do nothing, simply regard the point as noise.
        end %EO noise criterion passed IF
      else
        %% M of N points were not "stable", but we will try to save
        % as much stable data as possible
        noise_ind = find(Bp2p == 0);
        save_ind = find(Bp2p == 1);
        save_ind = save_ind(find(save_ind > noise_ind(1)));

        % allow the maximum number of noisy data points (N - M) and save as
        % much "stable data" as possible

        if isempty(save_ind)
          stabledata = current_data;
          bufferdata = [bufferdata(2:end,:);current_data];
          Bp2p = ones(1,N);%nan(1,N);%
          p2pbuffertimes = ones(1,N);
          p2pbuffertimes = [p2pbuffertimes(2:end),current_time];
        else
          Bp2p = [ones(1,save_ind(1)-1),Bp2p(save_ind(1):end)];
          p2pbuffertimes = ...
            [ones(1,save_ind(1)-1),p2pbuffertimes(save_ind(1):end)];
          bufferdata = ...
            [nan(save_ind(1)-1,3);bufferdata(save_ind(1):end,:)];
          stabledata = mean(bufferdata(save_ind,:),1);
          current_time = p2pbuffertimes(save_ind(1));
        end %EO salvage usable data IF
        t0stable = current_time;
      end %EO noise criterion IF
    elseif ¬instableperiod
      % The current patient programming input indicated by ppID was able to
      % be associated to a set of data (stabledata) and should be used to
      % modify the library.
      Flibmod = 1;
      Fassoc = 0;
    end %EO stability timer check IF
  elseif ¬insearchperiod
    % Search timer expired before an association could be made.
    Flibmod = 0;
```

```
    Fassoc = 0;
    t0ppinput = current_time;
    stabledata = zeros(0,3);
  end %EO search timer check IF
end %EO f(libmod) state check IF

prevdata = current_data;
updatedAssocVars = initialize_AssocVars(stabledata,prevdata,Bp2p,...
  p2pbuffertimes,bufferdata,Flibmod,Fnew,Fassoc,insearchperiod,...
  instableperiod,t0ppinput,t0search,t0stable,ppID);
end %EOF
```

B.3 Library Modification Function

```
function [entryID,posthresh_entries] = flibmod(ALLrefparam,LIBrefparam,...
  new_vector,ATLASparams);

% Purpose : Giving the all the vectors that have already been saved and the
% vectors that currently compose the library, using criterion given in
% parameters, this function determines whether a new library entry needs to
% be created or which library entry to new_vector should replace.

[ATLASparams,searchtimer,stabletimer,Dp2p,Dassoc,Dlibmod,M,N,distmeasure]...
  = initialize_ATLASparams(ATLASparams);

%% Automatically add data to the all_entry library
if isempty(ALLrefparam)
    posthresh_entries = 0;
else
    nassoc = size(ALLrefparam,1);
    dist_from_entries = ...
      dist_from_vector(ALLrefparam,new_vector,distmeasure);
    dist_from_sameposthresh = ...
      dist_from_entries - ones(nassoc,1)*Dlibmod;
    posthresh_entries = find(dist_from_sameposthresh <= 0);
end

%% Decide whether to create a new library entry or replace an exisiting one
if isempty(LIBrefparam)
    entryID = 1;
else
    nentries = size(LIBrefparam,1);
    dist_from_entries = ...
      dist_from_vector(LIBrefparam,new_vector,distmeasure);
    dist_from_sameposthresh = ...
      dist_from_entries - ones(nentries,1)*Dlibmod;
    [dist_from_closest_entry,closest_entry_ID] = ...
      min(dist_from_sameposthresh);
    if dist_from_closest_entry >= 0 %CREATE a new entry
        nentries = nentries + 1;
        entryID = nentries;
    else %REPLACE an existing entry
        entryID = closest_entry_ID;
    end
end
end
```

B.4 Data Classification Function

```
function [library,data_association_ID,data_LibEntry_ID,...
  stim_association_ID,stim_LibEntry_ID,ramp_time] = ...
  fclass(library,ClassifyParams,bufferLibEntries,...
  last_stim_LibEntry,last_stim_assoc,current_data);
```

```
5
6  % Purpose: Classifies data on a point to point basis and determines the
7  % closest library entry, the appropriate stimulation output and the
8  % appropriate ramp time required
9
10 distance_measure = ClassifyParams.distmeasure;
11 [ClassifyParams,sameposthresh,L,K,tlow2high,thigh2low] = ...
12   initialize_ClassifyParams(ClassifyParams);
13
14 if strmatch(distance_measure, 'degrees')
15   d2 = 1;
16 elseif strmatch(distance_measure, 'edist')
17   d2 = 2;
18 elseif strmatch(distance_measure, 'sdiff')
19   d2 = 3;
20 end
21
22 library_ref_params =library{d2}(:,2:4);
23 dist_from_entries = ...
24   dist_from_vector(library_ref_params,current_data,distance_measure);
25 [minval,min_libID] = min(dist_from_entries);
26 closest_entryID = min_libID;
27 closest_association_num = library{d2}(closest_entryID,1);
28 data_association_ID = closest_association_num;
29 data_LibEntry_ID = min_libID;
30
31 if sum(bufferLibEntries == min_libID) == K
32   stim_association_ID= closest_association_num;
33   stim_LibEntry_ID = min_libID;
34   lastEntry = last_stim_LibEntry;
35   currentEntry = stim_LibEntry_ID;
36   if lastEntry == currentEntry
37     ramp_time = zeros(1,4);
38   else
39     prevAmps = library{d2}(lastEntry,9:12);
40     currentAmps = library{d2}(currentEntry,9:12);
41     AmpDiff = prevAmps - currentAmps;
42     h2l = find(AmpDiff ≥ 0);
43     l2h = find(AmpDiff < 0);
44     if ¬isempty(h2l)
45       ramp_time(h2l) = thigh2low;
46     end
47     if ¬isempty(l2h)
48       ramp_time(l2h) = tlow2high;
49     end
50   end
51 else
52   stim_association_ID = last_stim_assoc;
53   stim_LibEntry_ID = last_stim_LibEntry;
54   ramp_time = zeros(1,4);
55 end
56
57 library{d2}(stim_LibEntry_ID,5) = library{d2}(stim_LibEntry_ID,5) + 1;
58
59 end
```

Bibliography

[1] Schwartz et al. Therapeutic electrical stimulation of the hypoglossal nerve in obstructive sleep apnea. *Arch Otolaryngol Head Neck Surg*, 127(10):1216–1223, 2001.

[2] Schwedt TJ, Dodick DW, Hentz J, Trentman TL, and Zimmerman RS. Occipital nerve stimulation for chronic headachelong-term safety and efficacy. *Cephalalgia*, 27(2):153–157, 2007.

[3] Cameron T. Safety and efficacy of spinal cord stimulation for the treatment of chronic pain: A 20-year literature review. *Journal of Neurosurgery: Spine*, 100(3):254–67, 2004.

[4] McIntyre CC, Savasta M, Goff LKL, and Vitek JK. Uncovering the mechanism(s) of action of deep brain stimulation: activation, inhibition, or both. *Clinical Neurophysiology*, 115(6):1239 – 1248, 2004.

[5] Barolat G. Epidural spinal cord stimulation: Anatomical and electrical properties of the intraspinal structures relevant to spinal cord stimulation and clinical correlations. *Neuromodulation*, 1(2):63–71, 1998.

[6] Gordon AT, Zou SP, Kim Y, and Gharibo C. Challenges to setting spinal cord stimulator parameters during intraoperative testing: Factors affecting coverage of low back and leg pain. *Neuromudulation: Technology at the Neural Interface*, 10:133–141, 2007.

[7] North et al. Automated, patient-interactive, spinal cord stimulator adjustment: a randomized controlled trial. *Neurosurgery*, 52(3):572–80, March 2003.

[8] Holsheimer J and Barolat G. Spinal geometry and paresthesia coverage in spinal cord stimulation. *Neuromodulation*, 1(3):129–136, 1998.

[9] Abejon D and Feler CA. Is impedance a parameter to be taken into account in spinal cord stimulation? *Pain Physician*, 10:533–40, July 2007.

[10] Holsheimer J, Barolat G, Struijk JJ, and He J. Significance of the spinal cord position in spinal cord stimulation. *Acta Neurochir Suppl*, 64:119–124, 1995.

[11] Olin JC, Kidd DH, Miller QL, and North RB. Postural changes in spinal cord stimulation perceptual thresholds. *Neuromodulation*, 1:171–75, 1998.

[12] Moffitt MA, Lee DC, and Bradley K. *Spinal Cord Stimulation: Engineering Approaches to Clinical and Physiological Challenges.* Springer New York, June 2009.

[13] Dijkstra EA, Holsheimer J, Olthuis W, and Bergveld P. Ultrasonic distance detection for a closed-loop spinal cord stimulation system. In *19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 5, pages 1954–1957. IEEE, 1997.

[14] *American Journal of Neuroradiology*, 15.

[15] Medtronic Inc. Spinal cord stimulation, June 2010.

[16] Struijk JJ, Holsheimer J, Barolat G, He J, and Boom HBK. Paresthesia thresholds in spinal cord stimulation: a comparison of theoretical results with clinical data. *Rehabilitation Engineering, IEEE Transactions on*, 1(2):101–108, June 1993.

[17] Alo KM. Lead positioning and programming strategies in the treatment of complex pain. *Neuromodulation*, 2(3):165–170, 1999.

[18] Melzack P and Wall PD. Pain mechanisms: A new theory. *Science*, 150(3699):971–978, Nov 1965.

[19] Shealy C, Mortimer J, and Reswick J. Electrical inhibition of pain by stimulation of the dorsal columns: A preliminary report. *Anesthesia and Analgesia*, 46(3):489–491, July 1967.

[20] Deer TR and Stewart CD. Complications of spinal cord stimulation: Identification, treatment and prevention. *Pain Medicine*, 9(S1), 2008.

[21] Vulink et al. The effects of spinal cord stimulation on quality of life in patients with therapeutically chronic refractory angina pectoris. *Neuromudulation: Technology at the Neural Interface*, 2:33–40, 1999.

[22] Kumar et al. Spinal cord stimulation versus conventional medical management for neuropathic pain: A multicentre randomised controlled trial in patients with failed back surgery syndrome. *Pain*, 132(1-2):179 –188, 2007.

[23] Mekhail NA, Aeschback A, and Stanton-Hicks M. Cost benefit analysis of neurostimulation for chronic pain. 20:462–468, 2004.

[24] Bradley K. The technology: The anatomy of a spinal cord and nerve root stimulator: The lead and the power source. *Pain Medicine*, 7(s1):1348–1356, 2006.

[25] North RB. Neural interface devices: Spinal cord stimulation technology. *Proceedings of the IEEE*, 96(7):1108–1119, July 2008.

[26] North RB. Spinal cord stimulation for chronic,intractable pain: superiority of multi-channel devices. *Pain*, 44(2):119–30, 1991.

[27] Linderoth B and Foreman RD. Physiology of spinal cord stimulation: Review and update. *Neuromudulation: Technology at the Neural Interface*, 2(3):150–164, July 1999.

[28] Abejon et al. Effect of posture on spinal cord stimulation in patients with chronic pain syndromes: analysis of energy requirements in different patient postures. *Rev Esp Anestesiol Reanim.*, 56(5):S27–S34, May 2009.

[29] Improved Outcome Software. Distance metrics overview, July 2010.

[30] Jain AK, Murty MN, and Flynn PJ. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[31] Golub GH and Van Loan CF. *Matrix Computations*, pages 425–26. The Johns Hopkins University Press, 1983.