Dynamic Discriminant Analysis with Applications in Computational Surgery

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Rodney Lee Dockter II

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Timothy M. Kowalewski Ph.D.

May, 2017

Acknowledgements

The author would like to thank the Medical Robotics and Devices lab members including John O'Neill, Mark Gilbertson, Trevor Stephens, Anna French, and Darrin Beekman for their support. The University of Minnesota Informatics Institute (MnDRIVE - UMII) Fellowship for financial support. The University of Minnesota Graduate School Interdisciplinary Fellowship for financial support. Professors Will Durfee, Max Donath, and Nikos Papanikolapolous for their encouragement and advisement. And finally Timothy Kowalewski for his immense support.

Dedication

To my wife Rachel. Without her support, encouragement, and patience this work would have never been finished.

Abstract

Background: The field of computational surgery involves the use of new technologies to improve surgical safety and patient outcomes. Two open problems in this field include smart surgical tools for identifying tissues via backend sensing, and classifying surgical skill level using laparoscopic tool motion. Prior work in these fields has been impeded by the lack of a dynamic discriminant analysis technique capable of classifying data given systems with overwhelming similarity.

Methods: Four new machine learning algorithms were developed (DLS, DPP, RELIEF-RBF, and Intent Vectors). These algorithms were then applied to the open problems within computational surgery. These algorithms are designed with the specific goal of finding regions of data with maximum discriminating information while ignoring regions of similarity or data scarcity. The results of these techniques are contrasted with current machine learning algorithms found in the literature.

Results: For the tissue identification problem, results indicate that the proposed DLS algorithm provides better classification than existing methods. For the surgical skill evaluation problem, results indicate that the Intent Vectors approach provides equivalent or better classification accuracy when compared to prior art.

Interpretation: The algorithms presented in this work provide a novel approach to the classification of time-series data for systems with overwhelming similarity by focusing on separability maximization while maintaining a tractable training routine and real-time classification for unseen data.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Executive Summary

The ultimate goal of this work is to address adverse surgical events through advancing computational surgery. The specific technical problem requires discriminating data from overwhelmingly similar datasets, the lack of successful algorithms in this field is impeding surgical progress e.g. adverse error mitigation. The strategy to achieve this goal was the development of novel machine learning algorithms that exploit class-discriminant dynamical features via information theory. If successful these algorithms may enable safer surgery by discriminating between similar system types, conditioned on the class, such as two surgeons of comparable skill or identifying grasped tissues during a robotic surgery, given data streams arising from a surgical robot.

The design of this research is two-fold, 1) the design of a dynamic discriminant analysis algorithm exploiting statistical and information-theoretic criteria, 2) The application of these algorithms to solve open problems in computational surgery. These classification problems include:

- Simulated linear and non-linear data sets.
- Identifying specific tissue types via minimally invasive surgical grasping.
- Identifying skilled vs. unskilled surgeons.

We developed four distinct algorithms which enable classification of dynamic time series data found in computational surgery. These algorithms are specifically designed to target data regions which contain maximal discriminating information yet avoid drawing conclusions about data-sparse regions. This is achieved through comparing probability densities, conditioned on the class. Each algorithm is then capable of classifying data for a single time-step (online for real-time reporting), as well as classifying entire trajectories using weighted classification estimates.

- For linear simulated data, the proposed algorithms are equivalent to algorithms from prior art. For non-linear data the proposed DLS algorithm exceeds ordinary least squares.
- For the tissue identification problem, the three proposed algorithms obtain better classification results than neural networks or random forests. Additionally, the DLS approach achieves the best accuracy (90%).

- For surgical skill classification from raw motion data, none of the proposed general algorithms nor the benchmark comparison algorithms adequately classify surgical skill. We conclude that raw surgical motion data is inseparable.

- However, using a novel derived feature (Intent Vectors), an overall classification accuracy of 96.9% was observed for three laparoscopic surgical tasks. This approach exceeds results in prior art.

A secondary contribution of this work has been the development of the Minimally Acceptable Classification (MAC) Criterion for surgical skill evaluation research. This criterion sets a minimal benchmark for algorithms developed in the field of objective assessment of surgical skill.

1.2 Motivation

The last decade has seen tremendous growth and advancement in the field of surgery along with the symbiotic proliferation of new technologies. A prime example of this advancement has been the da Vinci Surgical System (Intuitive Surgical Inc.). Despite these technological advancements, surgery still remains relatively risky. It is estimated that surgical errors account for at least 32,000 deaths each year in the United States [1–3]. These errors resulted in an extra $282 billion in costs and approximately 2.4 million days spent by patients in the hospital [4]. Additionally, adverse surgical outcomes were reported in 5% of robotic hysterectomies [5, 6]. This high instance of dangerous errors caused by surgeons results in an economic burden on the health care system. Additionally surgery has become extremely common with an average of 50 million surgeries performed each year, this corresponds to 7 surgeries per lifetime for the average American [7].

The relatively new field of computational surgery is centered around both the development of new surgical technologies and ensuring they are safely utilized. Computational Surgery is defined by Garbey et al. as a "new discipline that focuses on the application of medical imaging, robotics, biological modeling, simulation, and information technology in surgical treatment of patients" [8]. Despite recent advances in the tools available to surgeons, surgery is still ranked as one of the top 15 leading causes of death in United States [1–3]. One of the key goals of computational surgery is to utilize technology to make surgery safer. This can be achieved by making surgical tools more autonomous and capable of predicting errors such as tissue crush injury or accidental vessel puncturing, as well as by improving training and certification of surgeons to ensure adequate skill levels.

One of the key components of computational surgery is the processing and analysis of the large data sets generated by new surgical technology. In particular, data analysis and informatics can be utilized to improve surgical safety and patient outcomes. Several studies have noted that one of the largest problems with laparoscopic and other Minimally Invasive Surgical (MIS) procedures is the lack of dexterity and force feedback [9, 10]. The lack of force feedback in MIS procedures can lead to tissue crush injuries and vessel damage [11]. Several research groups have developed instrumented minimally invasive surgical tools capable of sensing force at the distal end [12, 13]. These instruments have generated large data sets of force and strain data correlated with specific tissue types [14]. However, there has been limited success in using

this data to accurately identify tissues.

Tholey et al. performed a study to evaluate the effects of vision, force, and combined feedback in regards to identification of tissue stiffness [9]. Their study indicated that vision feedback resulted in a 52% tissue classification rate while force feedback and combined feedback resulted in a 67% and 83% classification rate, respectively. Sie et al. performed online tissue identification in-vivo using an Extended Kalman Filter (EKF) for parameter estimation [14]. This group was able to perform estimation for the most delicate of their four tissues within 300ms of the grasp start. However using this method, it was not possible to discern between some tissue types (liver and small bowel) or handle unknown tissue types. The shortcomings witnessed in prior art have all suffered from the lack of a robust system identification method for the noisy, non-linear, time-series signals present in tissue that can cope with the overwhelmingly similar data streams. For this reason, tissue identification in-vivo is a key application for a dynamic discriminant analysis method. These results provide a baseline for evaluation of this method.

Objective surgical skill evaluation has also been a key area in surgical research. Several research groups have developed laparoscopic and robotic surgical skills trainers such as the Electronic Data Generation and Evaluation (EDGE) laparoscopic trainer [15, 16]. Along with theses trainers and simulators, substantial research has investigated tasks and metrics that best discriminate between skill levels [17]. Research has also demonstrated direct correlation between surgical skill level and complication rates [18]. However attempts at objective discrimination between expert and novice surgeons have been only mildly successful [19], not reaching an expected 100% discrimination for obviously different subjects.

The primary gap in prior surgical skill evaluation has been a 100% classification using motion analysis under leave-one-subject-out cross validation. Every study reviewed indicated a classification rate of 95% or below for binary and ternary skill classification. Additionally these classification rates focused on discrimination of static motion parameters (i.e. overall path length) which results in a loss of key dynamic data (e.g. what is happening and when during a procedure). Since motion analysis has been shown to discriminate skill level [20], the missing component is thus a robust discriminant analysis method capable of using the dynamic information available via laparoscopic and robotic tracking devices. By focusing on the key but subtle difference in expert and novice motion dynamics, a dynamic discriminant analysis method should be able to more accurately classify surgical skill level when compared with prior art, if it were to be deemed successful.

A common obstacle in both tissue type force identification as well as surgical skill evaluation has been the lack of a robust discriminant analysis algorithm capable of handling the non-linear dynamic data sets that are generated within these research fields. While discriminant analysis and system identification have been the subject of research for many years, an ideal solution for use within computational surgery has not yet been identified.

In order to eventually address surgical errors, improve surgical skill evaluation, and ultimately improve surgical outcomes, the proposed algorithms were designed to discriminantly classify similar systems given non-linear, noisy, time series data. This method will be applicable to a variety of time series data sets commonly found in computational surgery. Prior art has revealed a major gap in the field of discriminant dynamic analysis. Namely no algorithm exists which can simultaneously provide these key components:

- Dynamic: Directly operate on continuous (or discrete) time-series, non-linear class labeled data. Possibly with an indeterminate number of states.
- Discriminant: Effectively ignore attributes common or absent to all classes and emphasize the between class differences, preferably via tunable parameters.
- Tractable: Provide a tractable training algorithm i.e. training can be performed on a standard computer.
- Fast: Provide real time evaluation and classification of new, unseen data. e.g. to provide information for online adaptive control algorithms.
- Honest: Provide a means to report confidence or probability of correct classification.

1.3 Specific Contributions

The specific contributions presented in this work are outlined as follows:

- A modified Least Squares approach which computes discriminant parameters given class based data. This Discriminant Least Squares (DLS) approach generalizes Least Squares and allows amplification of discriminating information inherent in dynamic data.
- A novel grid based approach which identifies regions of maximum separability for use in online classification called Discriminant Phase Portrait (DPP).

- A modified version of the RELIEFF feature weighting algorithm capable of considering multi dimensional data via radial basis functions (RELIEF-RBF).
- An extension of RELIEF-RBF to sub-sample data in order to include only separable data for use in training a Gaussian Process classification.
- A novel motion metric for use in surgical skill classification (Intent Vectors).
- A new benchmark for use in surgical skill evaluation algorithm development which stipulates a minimum acceptable threshold for classification accuracy (MAC criterion).
- Application of these algorithms to experimental data from two open problems in computational surgery: tissue identification via backend sensing with minimally invasive surgical tools, and surgical skill level classification via surgical tool motion data. In both cases, the algorithms provide superior performance to prior art.

A copy of all source code is available via Git repository (`https://github.umn.edu/labmrd/Dockter_Thesis_2017_Code`) or by request.

1.4 Outline

The presented work will adhere to following outline:

- Chapter 2 Presents prior work in the field of discriminant analysis as well as computational surgery.
- In Chapter 3 the proposed algorithms for dynamic discriminant analysis are presented.
- In Chapter 4 the hardware used for data acquisition is detailed.
- In Chapter 5 the data sets and experimental procedures are outlined.
- Chapter 6 demonstrates the implementation of the Dynamic Discriminant algorithms on the core applications and their data sets.
- Chapter 7 presents a discussion and comparison of the algorithms presented.

Chapter 2

Background

2.1 Background on Computational Surgery

Minimally Invasive Surgery (MIS) has become a mainstay of modern surgical techniques. MIS surgery had initial success in the mid 1980's with rudimentary techniques used for laparoscopic cholecystectomy [21]. The 1990's saw tremendous growth in the number of MIS procedures performed as well as the safety and efficacy. MIS procedures were found to improve recovery time, reduce blood loss, and shorten hospital stays when compared with traditional open surgery [22].

Figure 2.1: Common RMIS End-Effectors. (Source: Intuitive Surgical)

The 2000's brought the introduction of Robotic-assisted Minimally Invasive Surgery (RMIS) (Fig. 2.1). The ZEUS Robotic Surgical System (Computer Motion, Sunnyvale, CA) and the da

Vinci Surgical System (Intuitive Surgical Inc.) both had initial success, with the da Vinci ultimately dominating the market. The RMIS system allows surgeons a more ergonomic position from which to perform the surgery and higher quality, 3-dimensional optics. RMIS systems also afford the surgeon additional dexterity through the use of a multi degree of freedom wrist at the distal end of the tool [23].

Systems such as the da Vinci have instituted a new step in the medical and surgical fields. The merger of technology and surgery has produced a commodity previously unavailable to researchers and clinicians; large, easily attainable, data sets. These data sets contain information not just on clinical outcomes but also surgeons performance, tissue properties, and procedural structure. The analysis and utilization of this information is the subject of research in a field known as computational surgery. Computational surgery is concerned with the intersection of computational science and medical technologies such as advanced imaging, laparoscopy, endoscopy, novel sensor, and virtual reality simulators [8].

Computational surgery is a relatively new field but has already seen successful application in several clinical areas. Garbey et. al have applied mathematical modeling coupled with image registration for use in post lumpectomy breast reconstruction [24]. Another application has been the fusion of MRI imaging with robotic surgery to allow real time visualization and force feedback guidance for beating heart surgery [25]. Another application of computational surgery which is somewhat less obvious is the design of a computational desk for preoperative planning [26]. Such a system is required for planning of complex surgeries such as endovascular surgery. This desk is required to simultaneously allow a surgeon to visualize a patients medical imaging, 3D renderings, segmentation, and model tool tissue interactions. The common thread for all computational surgical endeavors is the fusion of medical imaging, clinical data, and systems information to make surgery safer and more efficient.

2.2 Background on Surgical Skill Evaluation

While minimally invasive methods such as laparoscopic and robotic surgery have certain advantages, these procedures also present certain challenges and dangers not typically found in traditional open surgery. As such, these methods require more advanced training and evaluation criteria for surgeons. The primary issues for laparoscopic surgeons are the lack of tactile

feedback, loss of natural hand-eye coordination, lack of dexterity, and visual spatial perception [17, 27, 28]. Over time laparoscopic surgeons improve via muscle memory and improved visual spatial perception. However this takes time and experience. Robotic surgery similarly presents challenges to the surgeon. These challenges stem from the lack of force feedback and narrow viewing angle, in addition to the visual spatial perception issues found in laparoscopic procedures [23, 29]. While robotic surgery does improve the dexterity and 3D visualization when compared with laparoscopic surgeries, this technology still presents challenges to the surgeon not encountered in conventional open surgery. The issues associated with both laparoscopic and robotic surgery have inspired the development of both simulated training devices and skill evaluation methods. The purpose of these devices is to prepare surgeons for the difficulties encountered in minimally invasive surgery as well as develop evaluation criteria for surgeons before allowing them into the operating room.

Several training and evaluation systems have been developed in recent years for both laparoscopic and robotic techniques. These surgical training systems come in two primary designs; physical 'box' trainers and virtual reality simulators. Physical box trainers were originally designed for laparoscopic tools and allow users to insert actual tools into trocar ports and then perform various tasks in a shell. The user then views inside the shell via a camera and monitor. These tasks can include peg transfer, suturing, and cutting tasks in order to train psychomotor skills. Common box trainer systems for laparoscopy include the McGill Inanimate System for Training and Evaluation of Laparoscopic Skills (MISTELS), the Advanced Dundee Endoscopic Psychomotor Tester (ADEPT), and the Electronic Data Generation and Evaluation (EDGE) Trainer (Simulab Corporation, Seattle, WA) (Fig. 2.2) [15]. Research has also gone into the development of inanimate box trainers for robotic surgery as well. The Fundamentals of Robotic Surgery (FRS) consortium has proposed a psychomotor skill evaluation system which is currently under development [30].

Figure 2.2: EDGE Laparoscopic trainer. (Source: Simulab Corporation)

Virtual reality (VR) systems use mock surgical tool handles which are then fed into a computer system. The motions of these synthetic tools are then reproduced in a virtual world which the user views on a computer screen. These virtual worlds can be comprised of simple psychomotor tasks as well as human anatomy and actual procedures. Virtual reality trainers have been used for both laparoscopic and robotic procedures. For robotic surgery their exist commercially available options such as the Mimic dv-Trainer (Mimic Technologies, Seattle, WA) and the da Vinci Skills Simulator (Intuitive Surgical, Sunnyvale, CA). Virtual reality simulators also exist for laparoscopy. The MIST-VR [31], LapSim (Surgical Science, Gothenburg, Sweden) (Fig. 2.3) and LapMentor (Simbionix Corp, Cleveland, OH) are all commercially available simulators.

Figure 2.3: LapSim virtual reality trainer. (Source: Surgical Science)

Minimally invasive surgical trainers however do not inherently provide skills feedback and evaluation criteria. Research has gone into the development of objective surgical skills metrics. These metrics have different basis depending on the simulation system in use. In the field of traditional open surgery, the objective structured assessment of technical skill (OSATS) was

developed as a comprehensive objective rating system for surgical skill [32]. Using this system expert surgeons rate de-identified videos of residents or peers using a set of operation specific checklists and global rating forms. These checklists include eight parameters; respect for tissue, time and motion, instrument handling, suture handling, flow of operation, knowledge of procedure, overall performance, and quality of final product. For laparoscopic surgery, the global assessment tool for evaluation of intraoperative laparoscopic skills (GOALS) was created using similar objective evaluation criteria [33]. The GOALS criteria consist of a 5-item global rating scale. These items are based on depth perception, bi-manual dexterity, efficiency, tissue handling, and autonomy.

Both OSATS and GOALS are generally used as the gold standard when comparing other skill metrics in surgery [34]. However, this is not necessarily ideal given they typically have intraclass correlation coefficients of 0.89 and inter-rater reliability of 0.72 [32, 33]. However, objective skill measures have been found to correlate well with surgical outcomes. Birkmeyer et al. found that the bottom quartile of surgical skill (scored via OSATS) was found to have a complication rate of 14.5% compared with a 5.2% complication rate for the top quartile of surgical skill [18]. As such, OSATS as well as GOALS are considered de facto to be the available gold standard of surgical skill.

Recent research has investigated other systems and metrics for gauging surgical skill. In addition to scores and metrics, researchers have also explored classification methods to discriminate expert from novice surgeons. One such method has been the Fundamentals of Laparoscopic Surgery (FLS). FLS is comprised of both a cognitive knowledge and technical skills portion. The technical skills portion includes five main tasks intended to gauge skill level: peg transfer, pattern cutting, ligating loop, suturing with an intracorporeal knot, and suturing with an extracorporeal knot [17]. Initial studies found high degrees of correlation between objective FLS scores and subjective operating room performance [35].

Several groups have explored the use of Motion Analysis in order to gauge surgical skill. Initially three motion parameters were proposed by Datta et al. for skill evaluation: path length, number of movements, and task completion time [36]. Motion analysis has been applied by several researchers for gauging laparoscopic skill. Chmarra et al. explored the use of motion parameters coupled with the use of Linear Discriminant Analysis (LDA) to classify surgeons skill level between expert, intermediate, and novice [19]. In order to record motion parameters, the TrEndo tracking system was used [37]. For this study the parameters used were time, path

length, depth perception, motion smoothness, angular area, and volume. Using this method, classification was correct for 75% of the participants. Other research groups have further validated these motion metrics [38, 39].

Lin et al. applied the motion analysis concept to the physical limbs of a surgeon using inertial measurement units attached to the surgeons arms [16]. In order to perform classification this group calculated the average power spectrum density for each arm joint velocity, reduced the dimensionality using principal component analysis (PCA) and LDA to classify. In this study surgical skill level was correctly classified with a rate of $88-94\%$.

Rosen et al. explored the objective evaluation of skill using force information coupled with tissue interaction [40]. This study utilized a custom endoscopic grasper outfitted with force and torque sensors located on both the grasper handle and on the outer shaft of the grasper near the distal end. Using this tool, subjects of varying levels of reported skill performed a laparoscopic cholecystectomy on a pig. Surgical skill was then characterized using a Hidden Markov Model (HMM) based on the force torque patterns observed in various states of tissue interaction. This evaluation show a significant difference between all skill levels.

Jog et al. proposed a novel motion statistic: 'Ribbon Area' to evaluate surgical skill [41]. In this approach the Cartesian motion data of the surgical tool tip is transformed into a ribbon surface by tracing the area swept by a "brush" consisting of a line between the clevis of the tool tip and a point 1mm from the clevis along the the instrument axis. Using this line at subsequent time steps allows the calculation of a quadrilateral (A_t) formed by two brush lines. Summing the total Ribbon Area during a given task segment is used as a measure of efficient pose management. A threshold on the total Ribbon Area is used to classify surgical skill level. This approach resulted in 80% accuracy for binary skill level classification with k-fold cross validation.

One approach to surgical skill evaluation has been to decompose a surgical procedure or task into smaller subtasks or 'Surgemes' [42]. This study utilized kinematic data recorded using the da Vinci robot API. Using a data set of 72 features ($N(k)$) from the robot kinematics, this group transformed this feature set into lower dimensional data ($Y(k)$) using LDA. Using the $Y(k)$ features they utilized Bayes chain rule to compute the probability of being within a Surgeme given the kinematic data. For this study they attempted to classify motions from a suturing task into 8 Surgemes: 1) Reach for needle. 2) Position needle. 3) Insert and push needle 4) Move to middle (left hand). 5) Move to middle (right hand). 6) Pull Suture (left hand). 7) Pull Suture

(right hand). 8) Orient needle. In theory a dictionary of Surgemes could be developed for any surgical task. This study utilized only one expert and one novice surgeon and resulted in a Surgeme classification accuracy of 91%.

Reiley et al. applied Hidden Markov Models (HMM's) to the evaluation of surgical skill [86]. This approach used HMM's to model surgical motion within a given Surgeme for a suturing task. This work again utilized kinematic data via the da Vinci API in a feature vector consisting of 14 states. For this study they manually segmented the kinematic data into the corresponding Surgemes via video review. A separate HMM model was trained for each Surgeme and each skill level (Novice, Intermediate, and Expert) resulting in a total of 24 HMM models. Given a skill model and the observed output of the HMM, a maximum log likelihood was used to estimate the most probable model for a given set of a data. This approach resulted in a leave-one-trial-out classification accuracy of 100%, however this work did not report a leave-one-user-out accuracy. There is critical importance in regards to evaluating these algorithms using a leave-one-user-out validation scheme. In the leave-one-trial-out scheme, the classification model is trained with all trials from all subjects except for one trial. Therefore the model has prior knowledge of a given subject in the case where a subject performs more than one trial. In this case algorithms provide acceptable results because some trials from all subjects are used for training a model. The true test for real world performance is the leave-one-user-out scheme, an algorithm must be able to correctly classify a never before seen subject in a high stakes testing and evaluation scenario.

In a similar approach, Tao et al. utilized Sparse HMM's to model skill level based on the sequence of Surgemes used [44]. In this study manually segmented Surgemes were used for training the models. Given the known Surgeme at a given time step, the transition probability between each Surgeme is directly computed. These transition probabilities are the basis for each HMM. Using this approach, an HMM is trained for each skill level. A new trial is classified by finding the model which yields the highest log likelihood probability. In this study three tasks were utilized: Suturing, Needle Passing, and Knot Tying. This approach resulted in 97.4% accuracy for leave-trial-out but only 59% for leave-user-out cross validation in ternary classification. The low results for HMM's in leave-user-out cross validation indicates that these approaches have a high degree of over fitting.

Ahmidi et al. developed the Descriptive Curve Coding (DCC) method as means to perform gesture recognition and skill assessment [45]. The goal of the DCC approach is to assign a

local coordinate system to window of Cartesian position data. Then for consecutive coordinate frames, the relative change in motion is encoded as a string of 7 possible direction changes. Accumulated Frenet Frames [46] are used to assign a local coordinate system (W_i) to a small window of tool tip positions. Using subsequent frames, a change in direction is encoded as an integer (0-7) representing cardinal directions in 3-dimensional space. Given a trajectory of encoded strings, they trained a Common String Motif (CSM) dictionary for both Experts and Novices. For online classification they compute pseudo similarity metrics between online strings and the trained CSM to compute the probability of being either a Novice or Expert. For skill classification this approach provided an accuracy of 98% for k-fold cross validation and 91% for leave-user-out.

In [47], stroke-based features were used to assess motion consistency within septoplasty procedures. The septoplasty procedure requires the surgeon to remove the mucosal flap off the underlying cartilage and bone. In order to asses skill level in this procedure, tool motion data was transformed into a coordinate frame relative to the septal plane. Then for each removal stroke, they compute several features related to the efficiency of the stroke: trajectory length, stroke length, duration consistency, height distribution, and task time. Using these features they trained a kernel Support Vector Machine (kSVM). Under binary classification this approach gave an accuracy of 90.9% for leave-trial-out and 74% for leave-user-out cross validation.

The majority of studies exploring the use of motion parameters have all shown the construct validity motion analysis for evaluating laparoscopic skills [38]. Additional studies have also shown face and concurrent validity using motion analysis. Macmillan et al. demonstrated face validity using the Advanced Dundee Endoscopic Psychomotor Tester (ADEPT) for skill evaluation [48]. This study utilized task time, plate error score, and probe error score as motion parameters. Moorthy et al. were able to show concurrent validity using the Imperial College Surgical Assessment Device (ICSAD) [49]. This group used time and path length as parameters, however the overall score also included a custom observer checklist. The use of observer checklists inherently detracts from the objective nature.

Prior art has shown that objective measures of skill, specifically motion metrics, have the potential to accurately predict surgical skill level. Surgical skill level has also been shown to correlate well with decreased complication rates. However the primary gap in the prior art has been a 100% classification rate using motion analysis under leave-one-user-out cross validation. Every study reviewed indicated a classification rate of 95% or below for binary and

ternary skill classification. Additionally these classification rates focused on discrimination of static motion parameters (ie overall path length). Since motion analysis has clearly been shown to discriminate skill level, the missing component is thus a robust discriminant analysis method capable of using the dynamic information available via laparoscopic and robotic tracking devices. By focusing on the key but subtle difference in expert and novice motion dynamics, a dynamic discriminant analysis method should be able to more accurately classify surgical skill level compared with prior art.

2.3 Background on Tissue and Force Sensing

Another topic of research related to improving minimally invasive surgery has been the development of surgical tools capable of force and tactile feedback. A key weakness in both robotic and laparoscopic surgery is the lack of haptic feedback. This can prevent surgeons from using their sense of touch in order to assess potential complications [9]. Additionally the lack of sensing can lead surgeons to grasp tissue too hard, thus causing tissue crush injuries [10, 50]. In laparoscopic cholecystectomies, laparoscopic graspers have been shown to increase tissue crush injuries significantly [51, 52]. Similarly, laparoscopic gynecological procedures result in a 1.5% rate of injury to the ureter, resulting in inflammation, cellular death and fistula formation [53, 54]. For colorectal surgeries one of the most common instrument induced injuries is inadvertent tearing of the bowel from grasping too hard [55]. Colorectal laparoscopy results in a 0.13% incidence of bowel injury. The tissue crush injuries revealed in these studies is largely attributed to the lack of force feedback available to surgeons.

Several studies have focused on the quantitative benefits that force feedback can provide. MacFarlane et al. designed a custom Babcock grasper with force feedback and a haptic control console in order to test how well subjects could identify the compliance (firmness) of sample tissue [13]. When compared with a standard grasper, the force feedback grasper reduced the mean square error in compliance identification approximately four fold. Tholey et al. performed a similar study to evaluate the effects of vision, force, and combined feedback in regards to identification of tissue stiffness [9]. Their study indicated that vision feedback resulted in a 52% tissue classification rate while force feedback and combined feedback resulted in a 67% and 83% classification rate, respectively. While these results were only marginally statistically significant ($\alpha = 0.052$, Tukey's method), they still indicate that force feedback improves tissue

sensing capabilities.

Okamura et al. reviewed various applications of force feedback within minimally invasive surgery and the benefits of haptic feedback [56]. Their results indicated that a lack of force information resulted in an increase in fine suture breakage. Results also showed that sutures tied with some method of force feedback (auditory or visual) resulted in suture knot tension that approximated ideal tension. Wagner et al. evaluated the amount of force applied to tissue using a surgical robotic system during a mock blunt dissection and compared the results using various amounts of force feedback [10]. The results of this study showed that without force feedback, subjects applied an average force magnitude which was 50% greater than with force feedback. The peak force magnitude similarly increased by 100% without force feedback. Additionally, the number of errors that resulted in damaged tissue increased three fold. This study overwhelmingly demonstrated the change in grasping force caused by force feedback and in turn the need for force sensing to avoid excessive force application to tissue.

Several research groups have explored methods for the sensing of grasping force as well as systems for relaying force information back to the user. Primarily these methods have related to the sensing of force and position on the grasper in order to determine the tissue type. The key purpose of identifying tissue type in-vivo is to allow the automatic thresholding of force levels in order to prevent crush injuries. The basis of the identification of tissue type stems from the classic non-linear tissue model derived by Y.C Fung [57]. This model was expanded upon by Yu et al. to include mass and damper terms [58]. The dynamics of tissue using this nonlinear model can be expressed in terms of position and force variables:

$$u = m\frac{\partial^2 p}{\partial t^2} + d\frac{\partial p}{\partial t} + \alpha\left(e^{\beta\varepsilon} - 1\right) \tag{2.1}$$

Here ε is the tissue strain. The most commonly used sensing modality for force feedback has been a combination of strain gauges on the grasper and optical encoders for measuring jaw position (Fig. 2.4). Some of the initial work done in this field was by Bicchi et al. who developed a prototype sensorized laparoscopic tool capable of sensing tissue properties [12]. This prototype utilized strain gauges on an aluminum ring inserted in the grasper joint at the distal end. The position sensing was accomplished with a optical position sensor attached to the jaws. By fitting the force-angle relationship to a third order polynomial, this group saw promising disparity in the coefficient values. However, this research did not utilize the full nonlinear model developed in [57] and also did not report quantitative identification results.

Rosen et al. developed the Force Feedback Endoscopic Grasper (FREG) for the dual purpose of examining tissue properties in-vivo and assessing the benefits of haptic feedback in surgery [59]. This design used encoder wheels with 1400 quadrature position counts over the 34.4° grasp motion. The force sensing was achieved using flat coil actuators taken from a modified hard disk drive. This encoder-actuator combination was also use on the master side in order to provide force feedback to the user. Using this system, grasper force and position were fit to the α and β values in the tissue model (Eq. 2.1). Their results indicated a high quality of the numerical fit between experimental tissue values and model output ($R^2 > 0.99$).

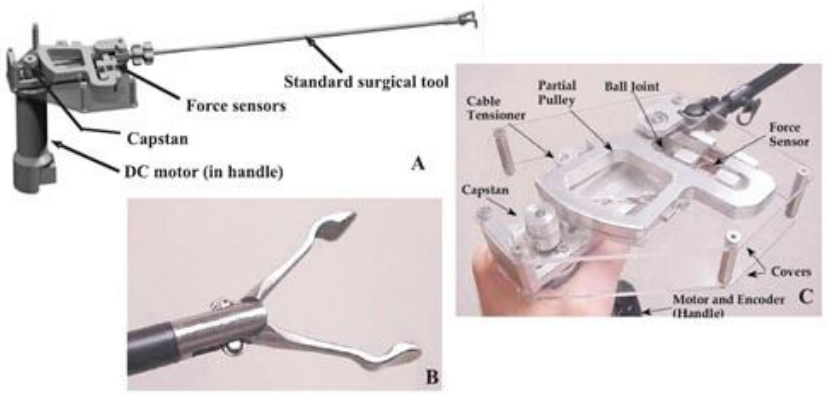


Figure 2.4: Motorized Endoscopic Grasper (MEG) (Source: [50])

Brown et al. improved upon the FREG design with the Motorized Endoscopic Grasper (MEG) for the purpose of in-vivo tissue identification [60, 61]. In comparison with the coil actuators used in the FREG, the MEG utilizes DC motors to actuate the grasping. The sensing in this implementation was achieved through the use of strain gauges attached to the pulley mechanism at the proximal end of the tool. Additionally position sensing was achieved with a digital encoder affixed to the motor. Using this system α and β values for liver and small bowel were recorded along with error bars. While the curve fits for the non-linear model were not perfect, this study provides baseline values for the model parameters.

Using a different sensing modality, namely an aspiration device, Hollenstein et al. were able to improve upon tissue sensing capabilities in vivo [62]. However in this device suction was used to suck the tissue into an aspiration hole where the deformation is then recorded with cameras. Using the deformation and the hole size, the stress strain relationship of the tissue

is estimated. While considerably different than the laparoscopic grasping modality, this study indicated that the strain modulus can be determined with a high degree of accuracy.

More recently, the MEG device has been used to characterize the acceptable levels of grasping force before tissue injury occurs [50]. Using the MEG, varying levels of grasping force were applied to liver, these tissues were subsequently analyzed for tissue and vascular damage via histological staining . This study concluded that a proper threshold for tissue grasp force existed around 180 kPa compressive force. This was based on the level of force where a statistically significant amount of neutrophils occurred. Using this methodology and a highly sensorized grasper, surgeons could be warned when damaging levels of force are applied.

Within robotic surgery, Yamamato et al. have developed a method for gauging tissue properties using the da Vinci Surgical System [63]. Instead of grasping the tissue, this design utilizes palpitation of the tissue via the robotic arms. The force applied to the tissue is then measured with a load cell situated below the tissue, therefore this system is not implementable in a real surgical setting. The tissue properties were then estimated using recursive least squares (RLS). While not a particularly useful system design, this group did develop a custom visualization technique for overlaying tissue properties on real time stereo images. Such a system is a novel method for providing surgeons with visual cues related to tissue grasping force.

Sie et al. have utilized the Mechanical Smart Endoscopic Grasper (MSEG) in order to perform online tissue identification in-vivo [14]. The MSEG is variation of the MEG with the addition of touch sensor on the grasper surface in order to sense the beginning of a tissue grasp. Using an Extended Kalman Filter (EKF) for parameter estimation, this group was able to perform estimation for the most delicate of their four tissues within $300ms$ of the grasp start. However using this method, it was not possible to discern between some tissue types (liver and small bowel) or handle unknown tissue types. This is assumed to be a result of the estimation technique used.

In [64], Li et al proposed the use of Gaussian Process Regression (GPR) to estimate the grip force at the distal end of a robotic surgical tool. In this work they recorded the motor position and torque (estimated via motor current) and used it to train an estimation model. Ground truth force on the gripper was sensed via Honeywell force sensors as the tool was tele-operated to grab synthetic rubber. The GPR approach was chosen since it allows implicit modeling of non-linear systems without a prior model. The GPR method learns the model directly from the training data. The GPR method outperformed standard dynamic modeling approaches, achieving a force

estimation error of $0.07N$. While not explicitly related to tissue classification, this method does allow a more accurate force estimation at the distal end of the tool.

Prior art has clearly shown the negative impacts stemming from the lack of haptic feedback within minimally invasive surgery. Research groups have also demonstrated that laparoscopic tools can be successfully modified to perform force sensing. Furthermore, using sensorized tools it has been shown that tissue identification can be performed in-vivo using a nonlinear tissue model. However the shortcomings witnessed in prior art have all suffered from the lack of a robust system identification method for the noisy, non-linear signals present in tissue. For this reason, tissue identification in-vivo is another key application for a dynamic discriminant analysis method.

2.4 Background on Machine Learning

The broad purpose of system identification and discriminant analysis is to determine the underlying mathematical structure of system given a data set. The simplest method of identification is a data fitting algorithm to determine the equation parameters that best describe a data set. A common example would be using $y = mx + b$ as a model and fitting the parameters m and b using a least squares fit. These parameters can then be utilized to classify the system.

An important distinction should be made concerning system identification versus discriminant analysis. While these two fields have similar purposes, system identification focuses on a generative model that best fits the data for a single class. In contrast discriminant analysis focuses on the classification of data into two or more classes. For discriminant analysis, a training set with labeled classes is required so that the differences in data sets can be computed. System identification methods are unlikely to succeed in dynamic discriminant analysis applications where different classes have trajectories with overwhelming similarity.

The majority of discriminant analysis methods and system identification techniques have focused on static data sets. A few recent exceptions have expanded into dynamic data sets within system identification. A review of two potential applications, tissue sensing and surgical skill evaluation has clearly indicated the need for a dynamic discriminant analysis method capable of handling the non-linear and potentially noise prone signals.

2.4.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is perhaps the most common discriminant analysis method; discriminating between classes using a single linear discriminant threshold [65]. LDA is typically attributed to Fisher's work on a linear discriminant. For the basic binary classification problem, this requires N_1 samples $[x_1^1, ..., x_n^1]$ belonging to class C_1, and N_2 samples belonging to class C_2. The samples are projected on to a single line $y = w^T x$ so as to maximize the separability. The optimal direction of this line can be solved using Fisher's linear discriminant. This derivation stems from a general multivariate Gaussian function:

$$F_k(x) = \frac{1}{(2\pi)^k |\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)} \tag{2.2}$$

Here k is the number of samples. In order to discriminate the two classes, the log ratio of the class Gaussian functions is used to find an optimal projection vector where separability is maximized. This is done by maximizing Fisher's linear discriminant function:

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \tag{2.3}$$

Where S_B is the between class scatter and S_w is the within class variance given by:

$$S_W = \sum_{i=1,2} \sum_{j=1}^{N_i} (x_j^i - \mu_i)(x_j^i - \mu_i)^T \tag{2.4}$$

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \tag{2.5}$$

This calculation requires calculating the scatter of the data (Eq. 2.6).

$$S_i = \sum_{j}^{N} (x_j^i - \mu_i)(x_j^i - \mu_i)^T \tag{2.6}$$

Where μ_i represents the average of all samples from class C_i.

$$\mu_i = \frac{1}{N_i} \sum_{j}^{N} x_j \tag{2.7}$$

These scatters sum to the within-class scatter (Eq. 2.4). The difference in these scatters is similarly the between-class variance (Eq. 2.5). The optimal LDA projection can then be obtained by solving the eigenvalue decomposition (Eq. 2.8) or directly via (Eq. 2.9).

$$\left|S_W^{-1}S_B - \lambda I\right| = 0 \tag{2.8}$$

$$W = S_W^{-1}(\mu_1 - \mu_2) \tag{2.9}$$

The W vector yields the optimal projection for maximizing separability of the classes. The threshold for separating the classes is finally given by the parameter T (Eq. 2.10).

$$T = W\frac{1}{2}(\mu_1 + \mu_2) \tag{2.10}$$

Where μ is the mean for each class. After computing the projection vector and threshold for the class labeled training data set, future samples can thus be classified based on the existing vector and threshold. Ideally new unlabeled samples will be correctly classified however new classifications are dependent on the separability of the scalars and thus the quality of the projection vector.

This derivation assumes binary classification. However, the extension of LDA to multi-class classification problems is completed by using a more generalized form. The between- and within- class scatters are simply computed between all class variations. Thus the projection vectors are compiled into a projection matrix as columns. The optimal projection vector is again computed as that which maximizes the ratio of the between- class to within- class scatter. This is found using a similar eigen decomposition to that of Eq. 2.8.

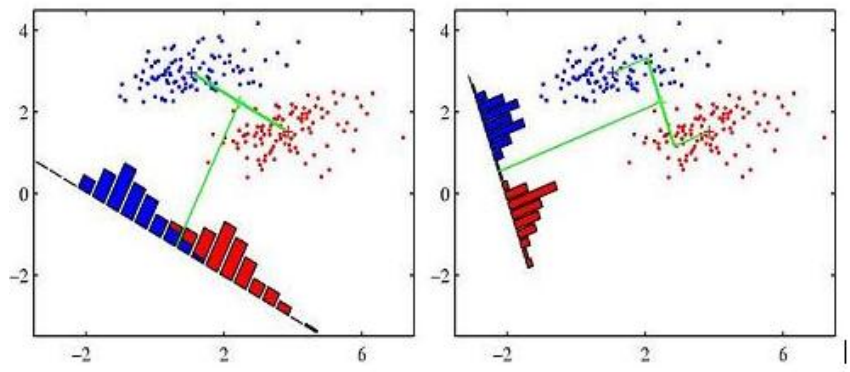


Figure 2.5: Left: classification using the mean squared projection. Right: binary classification using the LDA projection. (Source: [83])

LDA has also been extended to multiple dimensions with reasonable ease [66]. The primary difference in 2-dimensional LDA is the use of matrix representation for the scatter. The same type of eigen decomposition is still used to find the optimal projection. Other implementations of LDA have focused on overcoming the gaussian assumptions made in traditional LDA. Yan et al. proposed the use of graph embedding within LDA in order to determine the interclass and intraclass scatter, they have dubbed this method Marginal Fisher Analysis [67].

LDA has been utilized as the primary discriminant analysis method for several applications in computational surgery. LDA can only discriminate between static data which is not suitable for the dynamic data sets found in surgical tool motion and tissue data. As a result, research groups using LDA in these applications have found ways to aggregate dynamic motion data into static psuedo-metrics. For skill evaluation these aggregate values can include total time, path length, motion in depth, and motion smoothness [19]. However these aggregate values have the potential to lose key information present in the dynamic signal.

2.4.2 Quadratic Discriminant Analysis

A logical extension of LDA occurs when the classes do not have a common covariance matrix. In this case the normalization factors (quadratic terms) of the scatter ratios do not cancel nicely. This is the basis for Quadratic Discriminant Analysis (QDA). QDA deals with classification problems where the boundary between class pairs is modeled by a quadratic function. In the classical derivation the likelihood of each class is modeled as a Gaussian distribution.

For QDA the equation follows a similar fitting to that of LDA, except that the covariance matrices are computed for each class [65]. The discriminant projection vector is taken from the generalized Gaussian function (Eq. 2.2), without the common covariance matrix, the class likelihood ratio becomes

$$\delta_k(x) = -\frac{1}{2} log\,|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + log(\pi_k) \tag{2.11}$$

Here Σ_k represents the covariance matrix. In a similar manner to eigen decomposition found in the LDA derivation, the eigen values for QDA are found by taking the eigen decomposition of the diagonalized covariance matrix.

$$\Sigma_k = U_K D_K U_K^T \tag{2.12}$$

Here D_k is a diagonal matrix of eigenvalues. With this decomposition the solution for the quadratic function can be found by first multiplying by the center sample matrix

$$(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) = [U_k^T(x-\mu_k)]^T D_k^{-1}[U_k^T(x-\mu k)] \tag{2.13}$$

While both LDA and QDA can handle static classification problems very well, the quadratic approach allows classification for slightly more complex decision boundaries. Even for situations where the data distributions are not Gaussian, both LDA and QDA perform very well. This reason is likely that the Gaussian models are stable and provide a balanced bias-variance trade off [65]. A side by side comparison of the linear classifier and the quadratic classifier reveals the key difference in the boundary descriptor. The data used to examine the classifiers was the Fisher Iris data set, this consists of three different classes of flowers, Setosa, Versicolor, and Virginica. The data for each flower consists of petal length and petal flower. Both classifiers were computed in Matlab (The Mathworks Inc., Natick, MA). The Linear Discriminant classifier shown clearly demonstrates the two linear thresholds between each class (Fig. 2.6).

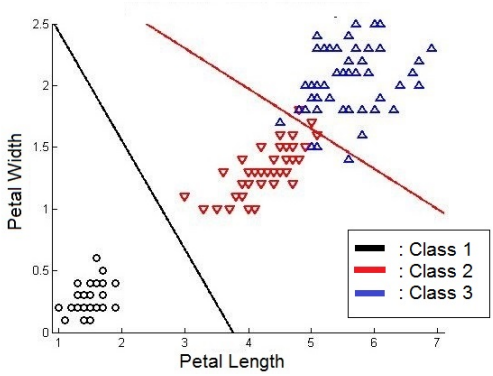


Figure 2.6: Fisher Iris classified with LDA (data set from Matlab (Mathworks Inc.))

The QDA classifier demonstrates the curved thresholding between each class (Fig. 2.7). This type of classifier certainly excels in cases where class boundaries are not straight lines but instead one class surrounds a portion of another.

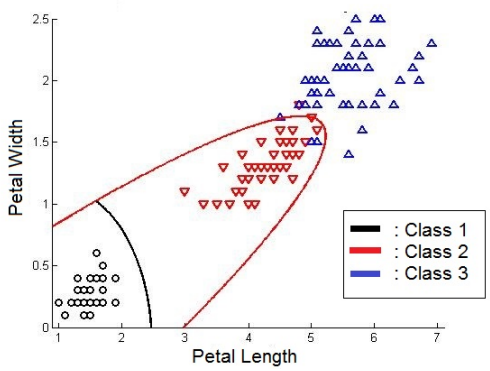


Figure 2.7: Fisher Iris classified with QDA (data set from Matlab (Mathworks Inc.))

While QDA is a commonly used tool in the literature, research has gone into various improvements and variations of the QDA algorithm. Friedman proposed Regularized Discriminant Analysis (RDA) as a compromise between LDA and QDA [68]. In regularized discriminant analysis, the individual covariance matrices while distinct are pooled so as to shrink the distinct matrices down to a common covariance as in LDA. This regularized covariance matrix is found

by 2.14.

$$\Sigma_k(\alpha) = \alpha\Sigma_k + (1-\alpha)\dot{\Sigma} \tag{2.14}$$

Where $\dot{\Sigma}$ represents the pooled covariance matrix as used in LDA. α is taken as a value between 0 and 1 which serves as a scaling factor between distinct covariances and common covariances. RDA has become a common classification method especially for data sets with ill-posed covariance matrices.

Srivastava et al. developed a variation of QDA wherein the prior (the probability distribution that expresses this estimate before observations are made) is computed using a rough covariance estimate [69]. This formulation was termed Bayesian Discriminant Analysis 7 (BDA7). The prior referenced is a probability distribution of the Gaussian functions used in the training set. The prior is computed as:

$$p = \gamma_o \frac{exp[-\frac{1}{2}tr(\Sigma_h - 1 B_h)]}{|\Sigma_h|^2} \tag{2.15}$$

Here γ_o is a normalization constant and B_h is a matrix that determines the value of the maximum prior probability distribution. Srivastava proposed changing this B_h from the common formulation and instead using

$$B_h = q\ diag(\dot{\Sigma}_{ML}) \tag{2.16}$$

Where $\dot{\Sigma}_{ML}$ is the maximum likelihood covariance matrix. This formulation was compared against both the classic QDA and the RDA methods and found a significant decrease in classification error rates relative to LDA and QDA and approximately the same error rates as RDA.

While QDA and the related RDA methods do have certain advantages over LDA, they still represent static classifiers and as such cannot directly handle dynamic data sets.

2.4.3 Kernel Discriminant Analysis

Kernel Discriminant Analysis (KDA) is yet a further variation of the classic discriminant analysis formulation. Multiple variations of the Kernel Discriminant Analysis method have been proposed in the literature. A discriminant analysis method using kernels for non-parametric class conditional distributions was proposed in the 1970's and is covered in [73]. However a latter implementation of KDA focused on the use of the Kernel trick for performing a non-linear

mapping into some feature space. This formal derivation was proposed by Mika et al. [74]. The summary presented here will focus on the KDA method for non-linear mappings.

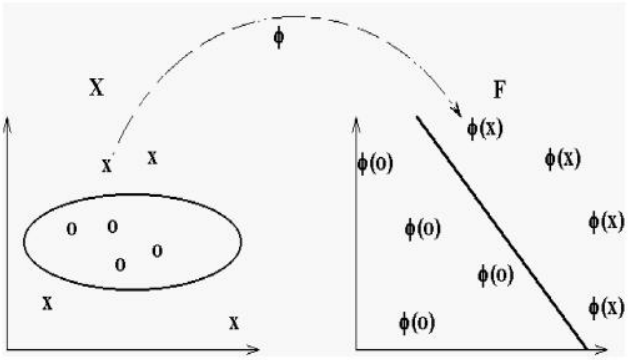


Figure 2.8: Possibly non-linear data (left) mapped to feature space using the kernel trick (right)

In the formulation of the KDA algorithm, the focus lies on the non-linear mapping to some feature space F. This feature space is a numerical representation of an object from the data (Fig. 2.8). This non-linear mapping is achieved via the 'kernel trick'. The basis of the 'kernel trick' is the ability to perform an inner dot product in a higher-dimensional space. Thus for $x_i, x_j \in \mathbb{R}^N$ the inner dot product in a higher dimension space $\mathbb{R}^M$ $(M > N)$ can be computed as $K(x_i, y_j) = (\Phi(x_i) \cdot \Phi(y_j))$ where $K(x)$ is the kernel and $\Phi(x_j)$ maps x to $\mathbb{R}^M$. One such common mapping is $\Phi = \frac{1}{2} exp(|X_1 - X_2|^2)$. This method is a computationally inexpensive means to map data to a non-linear feature space.

As in the LDA case the optimal discriminant solution is obtained by maximizing the following ratio:

$$J(w) = \frac{w^T S_B^\Phi w}{w^T S_W^\Phi w} \tag{2.17}$$

Where S_B^Φ and S_W^Φ represent the within-class and between-class scatter matrices in feature space. The vector $w \in F$ is assumed to have the form of 2.18.

$$w = \sum_i^\ell \alpha_i \Phi(x_i) \tag{2.18}$$

Here $\Phi(x_i)$ is the non-linear mapping into feature space. α_i is a normalization constant. ℓ is the size of the data vector. This expansion can be simplified and rearranged to a familiar form

from numerator of the ratio maximization (Eq. 2.17).

$$w^T S_B^{\Phi} w = \alpha^T M \alpha \tag{2.19}$$

Here M is represented as

$$M = (M_1 - M_2)(M_1 - M_2)^T \tag{2.20}$$

and

$$M_j = \frac{1}{\ell_i} \sum_{k=1}^{\ell} \Phi(x_j, x_k^i) \tag{2.21}$$

Similarly the denominator of 2.17 is found as 2.22

$$w^T S_W^{\Phi} w = \alpha^T N \alpha \tag{2.22}$$

Here N is defined as

$$N := \sum_{j=1,2} K_j \, (I - 1_{\ell_j}) K_j^T \tag{2.23}$$

Where K_j is the kernel matrix for class j and 1_{ℓ_j} is a matrix with all entries $1/\ell_j$.

By substituting 2.19 and 2.19 into 2.17, this results in a tractable maximization problem (Eq. 2.24).

$$J(\alpha) = \frac{\alpha^T M \alpha}{\alpha^T N \alpha} \tag{2.24}$$

By differentiating this ratio and setting equal to zero, the α value is readily obtained.

$$\alpha = N^{-1}(M_2 - M_1) \tag{2.25}$$

Finally the projection vector in feature space (w) can be mapped back to our input space in order to find a hyper-plane classifier (Eq. 2.26).

$$y(x) = (w \cdot \Phi(x)) = \sum_{i=1}^{\ell} \alpha_i \, k(x_i, x) \tag{2.26}$$

This summary of equations gives the simplified derivation of the KDA algorithm. The key component however is the use of the dot product kernel to map the discriminant to feature space in order to compute a linear solution there before mapping back to the input space.

Similar derivations have been proposed by other groups. Baudat et al. published a work on a Generalized Discriminant Analysis (GDA) method using kernels around the same time as Mika [75]. This work similarly proposes the use of a dot-product in feature space in order to solve the non-linear mapping. Then a ratio similar to the form of 2.24 is solved using eigen decomposition. The importance of this work is the extension to multiple classes compared with the binary derivation in [74]. The multi-class implementation primarily requires extending the summation of S_B^Φ and S_W^Φ to include all combinations of class scatter. More recently Cai et al. proposed a KDA method which using a spectral regression technique in order to make the discriminant solution a regularized regression problem as opposed to the eigen decomposition method [76]. To do this, the eigenvector problem is solved by substituting in a regularized regression solution. Then kernel matrix is similarly solved by compiling the set of orthogonal vectors spanned in the eigenvectors. The purpose of this work was generally to improve the computational efficiency associated with KDA.

While the KDA method does improve the classification of non-linear systems, this method again presumes a static data set and as such will lose key information found in dynamic systems. However the use of the kernel trick is particularly interesting as it allows for non-parametric and perhaps non-linear mappings which will play a role in the formulation of this work.

2.4.4 Kernel Density Estimation

The concept of non-parametric density estimation using kernels is originally attributed to Rosenblatt [70] and Parzen [71] around the year 1960. While the work of Rosenblatt and Parzen was not directly in an effort to develop a discriminant method, their ideas shaped the concept of the Kernel Density Estimator (KDE).

The primary concept behind the Kernel Density Estimator (KDE) is analogous to a histogram computed in continuous space. Instead of using discrete bins to determine the probability of a particular outcome, the kernel method acts as a moving average over each data point by centering a cell at each x (also called a Parzen window). The kernel density estimation takes the

form of 2.27.

$$f(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h}) \tag{2.27}$$

Here n represents the number of data points and h represents a smoothing parameter termed the bandwidth. The bandwidth has an optimal solution which can be computed using regression techniques [72]. K represents the kernel function which is any non-negative function which integrates to one. The kernel function predominately used in KDE implementations is the uniform kernel or the Gaussian kernel.

$$K_1(u) = \begin{cases} 1 & : \quad |u| < h \\ 0 & : \quad |u| >= h \end{cases} \tag{2.28}$$

While the KDE is not in itself a discriminant analysis method, this technique is immediately useful in computing analytic probability distributions from data for subsequent use in information theoretic algorithms.

2.4.5 Maximum Entropy

A variation of discriminant analysis has been developed which is based on the concept of maximum entropy [77–79]. The basis of entropy-based discriminant analysis is the identification of the within-class compactness and between-class separability. Entropy is a topic commonly used in the field of informatics and is broadly a measure of the uncertainty of a random variable. This is expressed mathematically as:

$$H(x) = -\sum_{x \in X} p(x) log(p(x)) \tag{2.29}$$

Here $p(x)$ is the probability function of variable x. The entropy definition can be extended to represent the relative entropy between two distributions of two random variables. This relative entropy is called Mutual Information (Eq. 2.30).

$$I(X;Y) = \sum_{x \in X}\sum_{y \in Y} p(x,y) log(\frac{p(x,y)}{p(x)p(y)}) \tag{2.30}$$

Where $p(x,y)$ is the joint probability function. This can be rewritten as $I(X;Y) = H(X) - H(X|Y)$. In other words the mutual information corresponds to the reduction in uncertainty of X due to the knowledge of Y. Of keen interest to this work is the fact that research groups have

successfully used entropy as measure of system complexity in biological time varying signals. Pincus originally proposed Approximate Entropy (ApEn) as a means to gauge the entropy of a time-series with changing complexity, such as biological system series [80]. The premise behind ApEn is the use of a window template which is passed over a time series of data, the width of this window is set by a parameter m. For each window of data, groupings of neighboring data points are used to determine the regions with the sharpest changes in value. The total number of regions where the gradient of change is above a certain threshold parameter r, is used to determine a difference term. By summing these difference terms, the approximate entropy can be computed at each time step. ApEn is advantageous in regards to its reduced computational requirements. Other variations of the ApEn method have included Sample Entropy [81] and Multi-Scale entropy [78]

ApEn has been applied by several researchers in regards to biological time signals. Richman et al. proposed the use of a variation of ApEn, termed Sample Entropy (SampEn), in order to measure system complexity for cardiovascular signals [81]. Sample Entropy is formulated as follows. For a time series data set of length N, $[u_1, u_2, ..., u_n]$, a window template vector (similar to that of ApEn) is defined for any given time i as $X_m(i) = [u_i, u_{i+1}, ..., u_{i+m-1}]$. Where m is the window size parameter. Then a distance function is defined between two such window vectors as the maximum difference between corresponding scalar components.

$$D[X_m(i), X_m(j)] = max(|u(i+k) - u(j+k)| : 0 \leq k \leq m-1) \tag{2.31}$$

Then the total count of pairwise distances is computed. B_i is the total number of vectors such that $D[X_m(i), X_m(j)] < r$ and A_i is the total number of vectors where $D[X_{m+1}(i), X_{m+1}(j)] < r$. Using these counts, SampEn is computed as a log ratio (Eq. 2.32).

$$SampEn = -log(\frac{A_i}{B_i}) \tag{2.32}$$

SampEn overcomes a shortcoming of ApEn which stems from bias that assumes additional non-existent similarities. The work of Richman et al. used SampEn to evaluate the similarity of two distinct cardiovascular time series. This time set involved sleeping patients heart rate and chest volume. SampEn was able to consistently determine the synchrony of each set in order to discriminate the two.

Costa et al. also explored the use of entropy for evaluating physiological time series, namely heart rate [78]. Costa developed a variation of SampEn called Multi-Scale Entropy (MSE)

which utilizes a time series scale factor τ such that the coarse grain time series window is defined by

$$y_j^\tau = \frac{1}{\tau} \sum_{i=(j-1)_\tau+1}^{j\tau} x_i,\ 1 \leq j \leq \frac{N}{\tau} \tag{2.33}$$

The value of τ allows SampEn to be calculated for a variety of time scales. When applied to heartbeat time series, Costa et al. found high values of entropy separation for young and old age groups when using MSE. This study showed that the weakest separation was between the two groups when using the unit time scale (that of SampEn), thus indicating the superior performance of MSE for use in discriminant applications.

Early work indicated the potential for the use of entropy and approximate entropy methods for discriminating time series based on the synchrony of the data. These results inspired the development of variations of Approximate and Sample entropy. The use of these tools for discriminant analysis is based on determining the expected entropy for two different classes and then using a distance metric in order to determine classification. However this simple calculation is not ideal for variable or noisy entropy. He et al. proposed the use of entropy principals in conjunction with a classic Discriminant Analysis approach in a method called Maximum Entropy Robust Discriminant Analysis (MaxEnt-RDA) [79]. In MaxEnt-RDA the feature extraction for optimal projection finding is achieved with the use of Parzen probability matrices in order to characterize the within class and between class variation. The Parzen window (Sec. 2.4.4) can be represented as

$$f_{x:\sigma}(x) = \frac{1}{n} \sum_{i=1}^{n} G(x - x_i, \sigma) \tag{2.34}$$

Here G represents the kernel function of bandwidth σ.

$$G(x - x_i, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x - x_i)^2}{2\sigma^2} \tag{2.35}$$

This Parzen window is then substituted into the standard quadratic entropy function (Eq. 2.36).

$$H(X) = -log \int f_x^2(x) dx \tag{2.36}$$

This substitution yields the following entropy estimate:

$$H(X) = -log(\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} G(x_j - x_i), \sigma) \tag{2.37}$$

This results in an eigen decomposition problem to locate the maximum entropy such that:

$$max(H(U^T X)\ s.t.\ H(U^T X|C) = c_1\ \&\ U^T U = I \tag{2.38}$$

Where $H(X|C)$ is the conditional entropy. The optimal solution for this maximization is given by an eigen decomposition (Eq. 2.39) similar to that of LDA.

$$XL_t(u)X^T u = \lambda XL_w(u)X^T u \tag{2.39}$$

He et al. solved this eigen decomposition by instead linearizing the kernel term (Eq. 2.35) using a first order taylor expansion. Using this formulation the maximization solution is reduced to a graph embedding problem. This group found that MaxEnt-RDA was able to outperform LDA in three separate classification problems.

The use of entropy for evaluating time varying signals as well as entropy based discriminant analysis are among the most promising approaches for discriminant analysis of time series data sets. While the discriminant analysis entropy methods have not been extended to time series, the incorporation of time series entropy measures such as SampEn is a logical progression.

2.4.6 Information Theoretic Techniques

While not traditionally a discriminant analysis technique, information theory has the potential for use in identifying and exploiting dynamic discriminant information. Information theory is traditionally used to identify and quantify distributions and relations between data. These methods are variations on the standard entropy measure discussed in section 2.4.5.

One common tool in information theory is the relative entropy of a system. Relative entropy is defined as the distance between two probability distributions. For example for two PDFs $p(x)$ and $q(x)$, the relative entropy is

$$D(p||q) = \sum_{x\in\chi} p(x) log\frac{p(x)}{q(x)} \tag{2.40}$$

Additionally relative entropy can be viewed as the inaccuracy of assuming that the data distribution is q when the true distribution is p. The relative entropy pseudo-measure is identical to the Kullback-Leibler (KL) divergence which is common in statistical analysis.

Another common information theoretic tool is the conditional entropy which is the expected value of the entropies of a conditional distribution. This measure can be thought of as the amount of information required to defined a random variable, given knowledge of a different random variable:

$$H(Y|X) = \sum_{x \in \chi} p(x) H(Y|X = x) \tag{2.41}$$

Where $H(Y|X = x)$ is the entropy of a variable Y for a known value of X. Another information theory measure is the mutual information. Mutual information is a measure of the reduction in the uncertainty of X given knowledge about Y. The most common expression of mutual information is:

$$I(X;Y) = \sum_{x,y} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{2.42}$$

Both mutual information and KL divergence can be used as measures of the more general concept; information gain. Information gain is the expected change in information entropy from a prior distribution to a posterior distribution. In other words this is the change in entropy from before an observation to entropy after.

$$IG(X,y) = H(X) - H(X|y) \tag{2.43}$$

The approaches mentioned here have the potential to provide key measures regarding separability and discriminant capability for dynamic discriminant analysis. Information theoretic techniques have previously been applied to certain machine learning techniques. However no prior art has been identified which utilizes KL divergence or information gain techniques in order to identify regions of maximum separability.

2.4.7 Feature Weighting and Dimensionality Reduction

In many machine learning applications, a data set may consist of a large number of features or dimensions. It is often the case that not all features provide relevant discriminating information. Some features may have high degrees of similarity between classes while others may

have complete separability. The goal of feature weighting and dimensionality reduction is to identify the best features in terms of discriminant potential. These features can then be used in subsequent machine learning techniques. One method for feature selection is the RELIEFF algorithm [82]. This is used in binary classification to rank features based on their ability to separate the data effectively. For each point, the K-nearest neighbors belonging to the true class (hit) and the opposite class (miss) are found. Using these nearest neighbors, a mean distance to both the hit neighbors (D_{hit}) and the miss neighbors (D_{miss}) is computed. The weights for a particular feature (W_f) are updated according to the difference between mean hit distance and mean miss distance (computed using that particular features data) (Eq. 2.44).

$$W_f = \sum_{i=1}^{N} \left(D_{hit_i} - D_{miss_i} \right) \tag{2.44}$$

Once weights for each feature have been computed, the features are sorted based on weight. Features with the highest weights are considered the most relevant features for classification. RELIEFF and its variants are limited to considering each feature separately and do not consider combinations of features simultaneously.

2.4.8 Principal Component Analysis

Principal Component Analysis (PCA) is another common tool used to reduce dimensionality in complex data sets while maintaining variance. PCA can also be used to perform feature extraction. PCA is defined as the principal subspace, such that the variance of the projected data is maximized [83]. A brief summary of the formulation is provided.

Given a vector of N samples $[x_1, ..., x_n]$, the data is projected into a maximum variance projection as follows. First the mean of the projected data is computed as $u_1^T \bar{x}$ where $\bar{x}$ is given by:

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{2.45}$$

The variance of the projected data is similarly

$$\frac{1}{N} \sum_{n=1}^{N} (u_1^T x_n - u_1^T \bar{x}) = u_1^T S u_1 \tag{2.46}$$

Where u_1 represents the projection vector and S represents the covariance matrix:

$$S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T \tag{2.47}$$

In order to maximize the variance from the vector, a Lagrange multiplier is utilized:

$$u_1^T S u_1 + \lambda_1(1 - u_1^T u_1) \tag{2.48}$$

Taking the derivative and setting equal to zero yields:

$$Su_1 = \lambda_1 u_1 \tag{2.49}$$

This then allows a solution for the eigen decomposition. u_1 is set as the eigenvector with the largest eigenvalue. In the general case of an N dimensional space, the optimal projection is represented by N eigenvectors $[u_1 ... u_N]$ which correspond to the N largest eigenvalues. Using these eigenvectors as coefficients allows the computation of each of the k principal components (Eq. 2.50.

$$Y_k = e_{k1} * x_1 + e_{k2} * x_2 + \cdots + e_{kN} * x_N \tag{2.50}$$

Where e_{ki} is the i^{th} coefficient from the k^{th} eigenvector u_k. Y_1 is thus the 1^{st} principal component and therefore the subspace with the highest variance.

While not a discriminant analysis method in itself, PCA can be used to find the optimal linear combination of dimensions which maximizes variance. As indicated in Section 2.2, PCA has been used extensively in order to reduce dimensionality in motion metrics. The major shortcoming of PCA is that it is blind to class. Notably, if the between class variance is small compared to the within-class variance, PCA will effectively ignore class data.

2.4.9 System Identification

The discriminant analysis methods mentioned previously have all focused on identifying optimal thresholds and projections for discriminating between classes. However, a second type of analysis, System Identification (SI) can also be used for classification of data sets. Most SI algorithms focus on a generative approach to solve for a set of parameters that fit a known model. These models can be either linear or non-linear. These parameters can then be used to

identify a particular class. The primary difference between discriminant analysis and SI is that identifying a particular system does not guarantee that two similar systems will be distinguishable. However, SI methods have been the subject of much research in recent years and given the similarity to discriminant methods, these concepts have the potential to aid in the development of a dynamic discriminant algorithm. Additionally, certain SI algorithms have been extended to dynamic signals.

One classic system identification method is the Ordinary Least Squares (OLS). This method is the best linear unbiased estimator for linear systems [84]. Other methods include Volterra, Wiener, and NARMAX methods, among others. Additionally, filtering methods such Kalman Filters and Particle Filtering can be used for system identification. These methods all focus on non linear series expansions based on system inputs and outputs. However they all focus on a generative model for system identification, not a discriminant model for classification.

2.4.10 Volterra and Wiener Methods

A classic approach for dynamic non linear system identification has involved the use of series expansions of nonlinear models. In the Volterra series, the system output at time t ($Y(t)$) depends on the input to the system ($x(t)$) at all previous times $[1,...,t-1]$. This expansion, originally derived by Vito Volterra in the 1800's, can be represented as

$$y(n) = \sum_{i=0}^{\infty} Y_i(x(n)) = Y_0(x(n)) + Y_1(x(n)) + ... + Y_j(x(n)) \tag{2.51}$$

Where $Y_j(x(n))$ is the jth-order functional:

$$Y_j(x(n)) = \sum_{k_1=0}^{\infty} ... \sum_{k_j=0}^{\infty} h_j(k_1,...,k_j)x(n-k_1)...x(n-k_j) \tag{2.52}$$

And $h_j(k)$ is the jth-order Volterra kernel:

$$h_j(k_1,...,k_j) = \sum_{m_1=0}^{\infty} ... \sum_{m_j=0}^{\infty} a_m(m_1,...,m_j)b_{m_1}(k_1)...b_{m_j}(k_j) \tag{2.53}$$

Here $a_m(m_i)$ are constant parameters and b_{mj} is the set of orthonormal basis.

The Volterra series was revisited by Norbert Wiener (around 1960) in order to perform non-linear circuit component design and identification. In the Wiener model, a non-linear system

is assumed to be in series with a linear, time invariant system. The output of the series in the Wiener model was rearranged to be:

$$y(n) = \sum_{i=0}^{\infty} G_i[k_i; x(n)] \tag{2.54}$$

Where G_m for even values of m is given by

$$G_m[k_m; x(n)] = g_m[k_m, k_{m-2(m)}, k_{m-4(m)}, ..., k_{0(m)}; x(n)] \tag{2.55}$$

And for odd values of m:

$$G_m[k_m; x(n)] = g_m[k_m, k_{m-2(m)}, k_{m-4(m)}, ..., k_{1(m)}; x(n)] \tag{2.56}$$

Here k_j represents the jth-order Wiener kernel:

$$k_{m-2r(m)}(i_1, ..., i_{m-2r}) = \frac{(-1)^r m!(\sigma_x^2)^r}{(m-2r)!r!2^r} \sum_{j_1=0}^{\infty} ... \sum_{j_r=0}^{\infty} k_m(j_1, ..., j_r, i_1, ..., i_r) \tag{2.57}$$

Using either the Volterra or Wiener formulation, system identification can be performed on these non-linear series expansions. In order to perform system identification, the output at time t and all previous inputs are required. For the Volterra series, identification is achieved by minimizing the square of the error in 2.59.

$$J(n) = e^2(n) \tag{2.58}$$

Where

$$e(n) = d(n) - \hat{d}(n) \tag{2.59}$$

Here $\hat{d}(n) = y(n)$ is the current output. The next step in the identification process is an update step for both the error and weights:

$$e(n) = d(n) - H^T(n)X(n) \tag{2.60}$$

$$H(n+1) = H(n) - \mu X(n)e(n) \tag{2.61}$$

Here μ represents a step-size parameter which influences convergence. $H(t)$ represents a weight vector comprised of all kernel weights as in Eq. 2.53. For each time step this kernel vector will ideally converge to values which minimize the error between the estimate and the actual output (Eq.2.59). These parameters can thus be used to identify a particular system. The identification of a Wiener expansion series can be found via a similar error minimization technique where instead the k_j kernels converge to an optimal solution.

The obvious benefit of the Volterra and Wiener series system identification methods is the ability to handle time-varying systems. In contrast with the discriminant methods mentioned above, the SI methods described here are not limited to static data sets. However, the parameters estimation and error minimization focuses on a best fit, not on the best discriminative features. Nevertheless, the Volterra and Wiener formulations are widely covered in the literature and as such deserve acknowledging.

2.4.11 NARMAX Methods

The Nonlinear Autoregressive Moving Average Model (NARMAX) is a common non-linear system identification. NARMAX is a variation on the classic Volterra method for parameter estimation. The NARMAX method excels in systems with large or non-linear noise. The NARMAX model is a function not only of past inputs and outputs ($y(k), u(k)$), but also noise sequences ($e(k)$).

$$y(k) = \theta_0 + \sum_{i_1=0}^{n} f_{i_1}(x_{i_1}(k)) + \sum_{i_1=0}^{n}\sum_{i_2=i_1}^{n} f_{i_1 i_2}(x_{i_1}(k), x_{i_2}(k)) + \\ \ldots + \sum_{i_1=0}^{n} \ldots \sum_{i_\ell=i_{\ell-1}}^{n} f_{i_1 \ldots i_\ell}(x_{i_1}(k), \ldots, x_{i_\ell}(k)) + e(k) \quad (2.62)$$

Here ℓ represents the degree of the polynomial expansion and $f(i)$ is a vector of model parameters:

$$f_{i_1 \ldots i_\ell}(x_{i_1}(k), \ldots, x_{i_\ell}(k)) = \theta_{i_2 \ldots i_\ell} \prod_{k=1}^{\ell} x_{i_\ell}(k) \quad (2.63)$$

$x(k)$ is vector of system inputs, outputs and noise:

$$x(k) = [y(k-1), \ldots, y(k-n_y), u(k-1), \ldots, u(k-n_u), e(k-1), \ldots, e(k-n_e)] \quad (2.64)$$

Where n_y, n_u, and n_e represents system output lags. System identification using the NARMAX model is performed by identifying the parameters in Eq. 2.63 such that the error $e(k) = y(k) - \hat{y}(k)$ is minimized. This requires identifying a nonlinear mapping:

$$F[y^{k-1}, u^{k-1}, e^{k-1}] \tag{2.65}$$

With this non-linear mapping , the model parameters θ_{i_m} can be used to identify a given system. In many ways the NARMAX formulation is similar to the Volterra series in that a nonlinear series expansion and the corresponding parameter values are used to determine a system. However the NARMAX formulation differs in the use of the noise-dependent model terms $e(k)$, these terms allow for nonlinear or biased noise. Furthermore, the NARMAX formulation still allows for time varying (dynamic) systems. However, it suffers from the drawbacks of other system identification algorithms in that discriminative features are not the focus. This detracts from the usefulness of such methods in dynamic discriminant analysis applications where different classes have very similar trajectories.

2.4.12 Hidden Markov Models

A Hidden Markov Model (HMM) is a statistical method for determining a model representation of a nonlinear system. HMMs are a variation on a basic Markov Model or Markov Chain. In a Markov chain the state space representation of a system is used and the system is characterized by the probability of transitions between various states. In an HMM, the states are not directly observed and instead of using the state transition probabilities, the state transitions are inferred by the sequence of output probabilities. The HMM theory was originally proposed by Baum et al. in the 1960's as method for recovering a state matrix via a set of observations [85]. HMM's have been applied extensively to speech recognition problems but have also been implemented in certain surgical skill evaluation methods [44, 86]. Additionally, some groups have utilized the more simplified Markov Chain for surgical skill evaluation [87].

The core definition of the HMM involves an underlying stochastic process in which the states are unobservable and only the output is observable. The elements of an HMM include the number of states in the model M, the number of distinct observation symbols per state M, and the state transition probability A which embeds the probability of transition between any two distinct states. Additionally, an HMM representation requires a probability distribution for the observations symbols $B = b_i(k)$ and an initial state distribution estimate π.

Using good choices for the HMM parameters should allow the HMM to compile an observation sequence which is given by $O = [O_1, O_2, ..., O_t]$. This observation sequence is computed via the following steps [88]:

1. Choose an initial state estimate according to π.
2. Choose $O_t = v_k$ based on the observation symbol probability ($b_i(k)$)
3. Transition to a new state $q_{t+1} = S_i$ according to the state transition probability distribution.
4. Move to the next time step $t = t + 1$ and repeat.

These steps require two main probability distributions, the probability of the observation sequence $P(O|\lambda)$, and an optimal estimate of the state sequence $Q = [q_1, q_2, ..., q_t]$ given the observation sequence. The observation sequence probability has several possible procedures. The simplest estimate of $P(O|\lambda)$ is given directly by summing the joint probability of all possible state sequences:

$$P(O|\lambda) = \sum_{q_1 \cdots q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T) \tag{2.66}$$

While Eq. 2.66 is not particularly computationally efficient it is the most straightforward solution. Other more efficient procedures also exist. Given the observation sequence probability, the state sequence estimate can be expressed as the probability of being in state S_i given the observation sequence and the model parameters $\lambda = (A, B, \pi)$.

$$P(q_t = S_i|O, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \tag{2.67}$$

Where $\alpha_t(i)$ and $\beta_t(i)$ account for portions of the observation sequence. The solution to Eq. 2.67, i.e. the likely estimate for the state at time t (q_t), can be solved as:

$$q_t = argmax(P(q_t = S_i|O, \lambda)),\ 1 \leq t \leq T \tag{2.68}$$

The final consideration in designing a HMM is the selection of optimal parameters $\lambda = (A, B, \pi)$ such that $P(O|\lambda)$ is maximized. Several methods exist for determining these parameters including Baum-Welch [89], gradient techniques, and expectation modification [88]. The

majority of these algorithms are not analytical and require iterative numerical approaches and maximum likelihood estimates.

The use of HMMs in discriminant analysis and system identification has been considered in several research fields. HMMs excel in systems with a known finite number of states and where system identification (generative model) is the goal. However HMM methods are ill-suited for systems with an indeterminate number of states. Conventional HMM methods are also not specifically designed for use as a discriminant model. The use of maximum likelihood estimates for solving for parameter values explicitly causes HMM algorithms to concentrate on general similarities in data as opposed to inherently discriminant algorithms which focus on regions of maximum separability between classes.

Recent research has focused shifting the use of HMMs to discriminant analysis settings. In order to achieve maximum discriminative ability the HMM must use a training set which focuses on maximizing the separability of classes and there the optimal parameter selection. Bourlard et al. proposed an initial discriminant based HMM which consisted of a hybrid approach with the use of Artificial Neural Networks (ANN) [90]. The goal of the HMM-ANN hybrid is to improve discrimination by training each model parameter set with consideration given to all other models, thus identifying maximum separability. In the work of Bourlard the training method is performed such that the HMM can estimate the probability of the observed data vector, given a hypothesized HMM state. Using a modified version of this probability, namely the posteriori probability of an HMM state given a data vector $P(q_k|x_n)$, the HMM parameters can be estimated by minimizing the Mean Square Error (MSE).

The formulation of the HMM with neural networks allows an initial extension of HMMs to the discriminant problem. Other groups have explored similar avenues for the use of HMMs in discriminant settings. Quan et al. utilized a similar HMM-Neural Network approach in order to improve separability in Signature Verification applications [91].

A different extension of the HMM is the discriminative model HMM which involves the use of Maximum Mutual Information (MMI) when designing the model parameters [92]. The joint MMI-HMM approach involves training the HMM model parameters while considering all other observations and models. This is in order to maximize the discriminative abilities of each model using the Bayesian discriminant function [93]. The Bayes discriminant function is the probability of a correct classification minus the probability of an incorrect classification (Eq.

2.69).

$$g_0(x) = p(C_1|x) - p(C_2|x) \tag{2.69}$$

The mutual information comes into focus when choosing parameters that optimally distinguish between observations generated by the appropriate model and observations generated by incorrect models; i.e. the parameters are chosen in order to maximize the mutual information I between the set of observation sequences $O = O^1O^2...O^T$ and the set of all models $\lambda = \lambda^1\lambda^2...\lambda^v$. This can be expressed as summing over all observations given all possible model parameters (Eq. 2.70).

$$I = \max_{\lambda} \sum_{v=1}^{V} [logP(O^v|\lambda_v) - log \sum_{w=1}^{V} P(O^v|\lambda_w)] \tag{2.70}$$

The hybrid HMM-MMI approach allows for an improved method of training an HMM to focus on the discrimination of particular classes. This method however cannot be implemented analytically and must be approached with numerical methods. However, the concept of maximized discrimination of incorrect classification is appealing.

While HMMs do present certain benefits including time series model estimation and no required knowledge of the internal states, HMMs also impose undesirable drawbacks. These drawbacks include the requirement of feature extraction in order to convert a time series to a finite number of states. This feature extraction can result in the possible loss of key information. Other drawbacks are presented in systems with indeterminate number of states. Additional drawbacks can include classification difficulties when dealing with systems that contain extremely similar signals and only moderate amounts of differentiating time series information.

2.4.13 Particle Filters

Particle Filters, otherwise known as Sequential Monte Carlo Methods, are a common numerical approach to system identification. The basic formulation of the particle filter considers an approximate solution to the optimal recursive Bayesian filter.

The particle filter is largely based on the Monte Carlo simulation, proposed in the 1940's by Ulam [94]. A Monte Carlo simulation is based on the concept of random samples of data in multiple dimensions. The sampling of this data is usually centered around an initial value with the sampling probability distribution encompassing random values around that point. Monte

Carlo simulations were initially conceived as a numerical approximation to solve difficult combinatorics problems and have since seen applications in physics and math. The particle filter in contrast utilizes the random sampling as weighted particles to approximate a probability density of a state space dynamical model. This implementation was first proposed by Gordon et al. and was termed the 'bootstrap filter' [95]. The basic formulation of the particle filter considers an approximate solution to the optimal recursive Bayesian filter. Normally the prediction density function for this filter is given by Eq. 2.71.

$$p(x_{t+1}|Y_t) = \int p(x_{t+1}|x_t)p(x_t|Y_t)dx_t \tag{2.71}$$

Here the filtering density, an update to the prior is then given by

$$p(x_t|Y_t) = \frac{p(y_t|x_t)p(y_t|Y_{t-1})}{p(y_t|Y_{t-1})} \tag{2.72}$$

The particle filter provides an estimate of the filtering density by using a set of random samples $[x_{k-1}(i) : i = 1,...,N]$. The distribution of these samples are taken from the Probability Density Function (PDF) $p(x_{k-1}|D_{k-1})$. Each sample represents a random estimate or particle of the possible state. Using these particles, the discrete estimate is then given by:

$$p(x_t|Y_t) = \sum_{i=1}^{N} q_t^{(i)} \delta(x_t^{(i)} - x_t) \tag{2.73}$$

Here $\delta()$ represents the dirac delta function. The superscript (i) indicates a particular state particle which is an approximation to the actual state. Similarly $q_t^{(i)}$ are the normalized weights for each distinct particle. The weights allow the particles to approximate the PDF. These weights are updated each time step by Eq. 2.74.

$$q_{t+1}^{(i)} = p(y_{t+1}|x_{t+1}^{(i)})q_t^{(i)} \tag{2.74}$$

This update function means that particles with the largest likelihood will have larger weights. Then as time progresses, the particles and their associated weights will begin to approximate the PDF from the density function for the filter.

In order to implement this particle filter, first the state space model has to be specified (in the classic formulation). The state space model is assumed to follow some parametric functions

$f(\cdot)$ and $h(\cdot)$. The standard model is given by

$$z_{t+1} = f(z_t; \theta_t) + v_t^z \tag{2.75}$$

$$y_t = h(z_t; \theta_t) + e_t \tag{2.76}$$

These functions are dependent both on the state z_t and the function parameters θ_t. Additionally, initial estimates of the densities p_{z_0}, p_{θ_0} and noise densities p_{v_t} must be specified. Finally the particles are initialized as $[x_0^{(i)} : i = 1, ..., N]$ where x is distributed according to p_{x_0}

The next step is measurement update from the current system output. In this step the weights are updated according to

$$q_t^{(i)} = q_{t-1}^{(i)} p(y_t | x_t^{(i)}) \tag{2.77}$$

Here $i = 1, ..., N$. The next step is to re-sample from the previous particles according to their weights. Several methods for re-sampling exist, the most common is called Sampling Importance Re-Sampling (SIR). In this method N samples are picked from the previous set $x_t^{(i)}, \theta_t^{(i)}$. The probability of picking a particular sample i is defined by the weight $q_t^{(i)}$. Hence highly weighted particles are more likely to persist to the next time step, meaning that particle is more likely a true estimate of the state. Notice that re-sampling also propagates parameters estimates for the state space model. Therefore particles and weights can be used to gauge the likelihood of a particular parameter better representing the state space model.

The final step is the prediction step. Here the state is updated according to the state space model and the noise densities (Eq. 2.78). The parameters are also updated according to the parameter noise densities (Eq. 2.79).

$$x_{t+1}^{(i)} = f(x_t^{(i)}; \theta^{(i)}) + v_t^z + w_t^z \tag{2.78}$$

$$\theta_{t+1}^{(i)} = \theta_t^{(i)} + v_t^\theta + w_t^\theta \tag{2.79}$$

This algorithm continues for subsequent time steps until a sufficient condition is met. Usually this condition is taken to be a certain weight threshold is achieved or a certain number of time steps have passed. The purpose of this particle filter formulation in terms of system

identification is for parameter identification. If the general model of a system is known, then by estimating the parameters that best represent the data, the specific underlying system can also be discovered. The use of the particle filters for state and parameter estimation has been proposed by several researcher groups [96–98].

More recently work has proposed explicit system identification via parameter estimation in a particle filter setting. Poyiadjis et al. proposed a more computationally efficient means to compute the score vector for the particle filter with explicit applications in parameter estimation [99]. With the proposed algorithm, model parameters for a stochastic volatility model were estimated with only 50 particles. Schon et al. also presented an explicit derivation of the use of particle filters for system identification [100]. This work highlighted the use of expectation maximization (EM) for parameter estimation in non-linear systems. Using this EM framework, convergence to the correct parameters for a non-linear, time varying system, was achieved with only 50 particles. Limetkai et al. proposed a Conditional Random Field (CRF) filter variation of the particle filter as a discriminative modeling technique [101]. The CRF filter is a discriminative undirected probabilistic model for use in continuous functions. This CRF method was implemented in a robot localization application with average errors of about $7cm$. CRF methods have certain negative traits, namely that the system requires a discrete number of states. For several applications, such as surgical skill evaluation, there is no deterministic method to define the number of necessary states. Another issue is that the training parameters cannot be subsequently used to inform the trainee about necessary improvements or errors. Finally the CRF model is not truly discriminant classifier in that optimal separation projections are not guaranteed.

In general the particle filtering method has the potential for use in a discriminant setting. The particle filter is favorable in situations with non-linear system models. However, in systems with high levels of noise relative to the inter-class separation, parameter estimation becomes increasing difficult. Additionally all PF based classification schemes are based on the closest parameter set to a known system. For situations where a known system model is not available, such as complex tissue models, the traditional particle filter approach will not work.

2.4.14 Random Forests

Random Forests are an ensemble method initially developed in 2001 [102]. Similar to Adaboost and other boosting techniques, Random Forests employ a divide and conquer strategy to create an ensemble of weak learners. Each weak learner alone has limited discriminant ability, however when combined they make up a strong learner.

The basis of training a Random Forest is a simple decision tree [103]. In a decision tree, class labeled L_T training input data X_T (Eq. 2.80) enters the top of the tree and is parsed into smaller subsets at each chance node using a weak threshold (usually a Decision Stump [104]). Once a subset at a given end node has been parsed down to a single data point (or data points from a single class), the corresponding class at that node can be used as a classifier.

$$X_T(t) = [x_1(t), x_2(t), \cdots x_m(t)] \tag{2.80}$$

In the Decision Stump approach, the best threshold is found by testing all values in the data set as a threshold (Eq. 2.81). The threshold T_D which maximizes information gain is taken as the best predictor.

$$T_D = argmax(IG(L_T, L_T(X_T < T_D))) \tag{2.81}$$

A basic calculation of Information Gain (IG) is given in Equation 2.82, where H is the entropy and $H(y|\bar{y})$ is the conditional entropy.

$$IG(Y, \bar{Y}) = H(Y) - H(Y|\bar{Y}) \tag{2.82}$$

In the case of a Random Forest, multiple decision trees are combined to create a 'Forest'. The 'Forest' is trained via taking random samples $\hat{X}_T \in X_T$ of the complete training data set. These samples are generally about 70% of the complete training data. The remaining data is used as the out-of-bag data (initial test data for a given tree). Of the sub training data, a new sample of m variables is used to train the current node in the current tree. At the next node in the tree a new sample of m variables is used to train that node. This process repeats until the end of the first tree is reached. The same process is repeated for all subsequent trees.

For online classification, new data is sent through each tree, the resultant classification from each tree can then be averaged to arrive at the aggregate classification estimate. Random Forests

have the benefit of not requiring any aggregate model. Additionally the training runtime is primarily a function of the number of trees hyper-parameter.

2.4.15 Neural Networks

Neural Networks (NN), also called Artificial Neural Networks, are a common pattern recognition method in the field of machine learning. Neural networks were originally proposed in the 1960's as mathematical representation of biological information. Broadly a neural network is an attempt to simulate the way the human brain processes information. In order to create this artificial brain, all potential inputs to the system are scaled by independent weights and fed into a 'neuron'. This 'neuron' then sums all scaled values to compute an activation value. The output activation value is then fed forward to the next layer. The most common NN approach is called a feedforward network. In the feedforward NN, this activation function is created by M linear combinations of the input values $[x_1,...,x_D]$, this is called the input layer of the network.

$$\alpha_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \ : \ j = 1,...,M \tag{2.83}$$

Here the superscript (1) indicates the 'layer' of the network. The parameters $w_{ji}^{(1)}$ are called weights and $w_{j0}^{(1)}$ are biases. These activation values are then transformed by an activation function $h(\cdot)$ in order to give the output of a basis function (similar to linear regression basis functions)

$$z_j = h(\alpha_j) \tag{2.84}$$

The result of the activation functions makes up a hidden layer in the network. The activation function in the hidden layer is often a $tanh()$ or sigmoid function. The output value of this function is then used as an input to subsequent layers in the the network. Next the hidden units are again linearly combined to compute the output unit activations:

$$\alpha_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \ : \ k = 1,...,K \tag{2.85}$$

Here K is the total number of outputs, $w_{kj}^{(2)}$ represent weights, and $w_{j0}^{(2)}$ are the biases. Note here that the superscript (2) now represents the second 'layer' of the network. Finally, the output

unit activations are subjected to another activation function in order to compute the networks outputs:

$$y_k = \sigma(\alpha_k) \tag{2.86}$$

The output activation function is commonly either the identity or a sigmoid function [83]. By combining the input layer, hidden layer and output layer, the complete network function can be created and used for generic multi-class classification problems. A simple diagram of the network with $D = 5$ input variables and $K = 1$ output variables is given in Fig. 2.9.

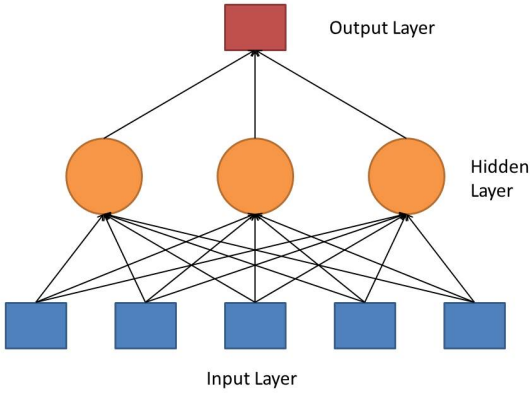


Figure 2.9: A simple Neural Network with five inputs and one output.

It should be noted that any finite number of input variables and output variables can be utilized in a single layer of a neural network. Each output will then return a value based on the activation function used. The output nodes can be trained to represent a variety of quantities including correct or incorrect classifications in a discriminant analysis application.

The key step in the development of a neural network is to train the various network parameters such as weights, $w_{ji}^{(1)}$, for each activation function. The weights must be trained so that a particular output node will return one when a correct classification is present and zero otherwise. The bias terms $w_{j0}^{(1)}$ act as the threshold offsets and also need to be trained. There are multiple methods to train these weights. These training methods can be either supervised or unsupervised learning techniques depending on the application. The most common of which is backpropogation. Other methods include iterative minimization, and gradient approaches.

Most parameter estimation routines are based on the concept of minimizing a sum of squares error function given by Eq. 2.87.

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}(y(x_n, w) - t_n)^2 \tag{2.87}$$

Where t_n represents the target output vector and $y(x_n, w)$ represents the output vector. In order to minimize this error using an iterative method, an initial value for the weight vector is chosen $w^{(0)}$ is used. Then the weight vectors are computed in succession using the formula:

$$w^{(\tau+1)} = w^{(\tau)} + \Delta w^{(\tau)} \tag{2.88}$$

Where $\Delta w^{(\tau)}$ is the update term and can be determined using various techniques including gradient descent. Several other weight training techniques are discussed in the literature [105, 106].

Neural Networks (NN) are a common machine learning technique. This approach focuses on static data or in some instances, time series with discrete input vectors at each time set. Furthermore, while learning the optimal parameters for a NN system can be beneficial in classifying a particular output, the parameters do not generally have a direct correlation to the physical system.

Recent research has been aimed at overcoming some of the issues present in traditional neural networks. One approach in this field is called the Recurrent Neural Network (RNN). The basis of the RNN approach is the output activation value for any layer is propagated back to the hidden layer as a sort of modified feedback loop [107]. RNN methods can be used either in discrete time or continuous time.

In this learning algorithm, the error at a particular instant is defined to be:

$$e_k(t) = d_k(t) - y_k(t) \tag{2.89}$$

Where $d_k(t)$ is a specified target class that the output should correspond to at time t. The then overall error at time t including past errors is:

$$J(t) = \frac{1}{2}\sum_{k\in U}[e_k(t)]^2 \tag{2.90}$$

Using this formulation, the change in weights for any layer is given by:

$$\Delta w_{ij} = \sum_{t=t_0+1}^{t_1} \Delta w_{ij}(t) \tag{2.91}$$

Then the actual change in weights relative to change in error is given by:

$$\Delta w_{ij} = \alpha \sum_{k \in U} e_k(t) p_{ij}^k(t) \tag{2.92}$$

Where $p_{ij}^k(t) = \frac{\delta y_k(t)}{\delta w_{ij}}$ is the change in output given weight changes. The optimal weights can then be solved in manner similar to the traditional NN in order to minimize the error in Eq. 2.90.

Both the NN and RNN techniques allow for non-linear, stochastic processes to be identified and modeled. In fact, the binary output values for the final layer can be trained to perform discriminant analysis wherein a result of 'one' from an output node would indicate that a particular class is represented by the data. However, this discriminant method has certain downfalls specifically if multiple class output nodes return a binary true value at the same instant, requiring further arbitration. The NN framework provides no means to determine which proposed class is actually correct except for the use of subsequent layers. For this reason several layers may be required and even then a single discriminant classification is not guaranteed. Furthermore, the NN framework is dependent on a known, finite number of classes. However for many, highly complicated systems, such as a human performing surgery (Sec. 2.2) there is no way to know how many inputs are required. Reiley et al. proposed using nine states for use in their HMM based surgical skill evaluation research [86]. However that does not indicate that nine input nodes would be sufficient for a similar neural networks based approach. Nevertheless, the NN framework represents an elegant solution for classification in both static and dynamic systems.

2.4.16 Gaussian Process Regression

Gaussian Process Regression (GPR) is a common method of learning non-parametric, kernel-based, probabilistic models [108]. The goal of GPR is to probabilistically estimate the expected output Y given and input X. In Bayesian Linear Regression (BLR) it is assumed that the state space function is linear $f(x) = Xw$, yielding a prediction of the form $P(y|X,w) \sim \mathcal{N}(Xw, \sigma_n^2 I)$. Here X is an $n \times m$ matrix and w is a $m \times 1$ vector of weights. In contrast to BLR, Gaussian

Process Regression makes no inherent assumption about the form of the state space model. Instead GPR uses kernel functions to represent the model directly from the training data.

By definition a Gaussian process is a set of random variables such that a discrete sample of them comprises a joint Gaussian distribution. If $\{f(x), x \in R^d\}$ is a Gaussian Process, then given the observations $x1, x2, \cdots x3$, the joint distribution of the random variables $f(x1), f(x2), \cdots f(x3)$ is also a Gaussian process. Therefore a Gaussian process can be fully represented by a mean function, $\mu(x)$ and a covariance function $K(x,x')$ (Eq. 2.93).

$$p \sim GP(\mu(x), K(x,x')) \tag{2.93}$$

The most common derivation of a GPR model assumes a data set of the form $D = \{(x_i, y_i)_{i=1}^n\} = (X, y)$. It is assumed that the output follows the form of Equation 2.94.

$$y = f(x_i) + \varepsilon_i \tag{2.94}$$

Where f is a latent variable and ε is zero mean Gaussian noise:

$$f \sim GP(0, K) \tag{2.95}$$

$$\varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{2.96}$$

Since the prior on f is a Gaussian process then the posterior on f ($p(f|D)$) is also a Gaussian process. This model is used to make predictions for output estimates (y_*) given new samples (x_*) as in Equation 2.97.

$$p(y_*|x_*, D) = \int p(y_*|x_*, f, D)\ p(f|D)\ df \tag{2.97}$$

Given this distribution, a predictor of the form $p(y_*|x_*, X, y)$ is desired, in other words the estimate of the output is dependent only on the new sample, the training data, and the latent variables. The definition of a Gaussian process yields Equation 2.98.

$$\begin{bmatrix} y \\ y_* \end{bmatrix} = \mathcal{N}\left(0, \begin{bmatrix} K_N & K_{*N}^T \\ K_{*N} & K_{**} \end{bmatrix}\right) \tag{2.98}$$

Where K_N is the covariance matrix of the training data, K_{*N} is the covariance between the training data and the online data and K_{**} is the covariance of the test data. This permits a predictive distribution of the form in Equation 2.99.

$$p(y_*|x_*,X,y) = \mathcal{N}(\mu_*|\sigma_*^2) \tag{2.99}$$

In this case the mean and covariance are given by Equations 2.100 and 2.101.

$$\mu_* = K_{*N}(K_N + sigma_n^2\, I)^{-1} y \tag{2.100}$$

$$\sigma_*^2 = K_{**} - K_{*N}(K_N + sigma_n^2\, I)^{-1} K_{*N}^T \tag{2.101}$$

Where σ_n is a tunable parameter relating to inherent observation noise in the system. This then permits an output estimate y_* for any sample x_* which is based solely on the training data and the covariance kernel K. The choice of covariance kernel is a key component in developing a GPR model. The most common kernel function is the squared exponential kernel (Eq. 2.102) which is very similar to a Radial Basis Function.

$$k(x,x') = \sigma_f^2\, exp(-\frac{||x-x'||^2}{2\ell^2}) \tag{2.102}$$

Where σ_f is a tunable parameter related to the process noise, and ℓ is related to the characteristic length scale or bandwidth of the data. The squared exponential kernel can be thought of as summing a Gaussian at one data point given all other data points. It should be noted that any kernel function can be used for a GPR model. Using this kernel results in a covariance matrix of the form in Equation 2.103.

$$K(X,X') = \begin{bmatrix} k(x_1,x_1') & k(x_1,x_2') & \cdots & k(x_1,x_n') \\ k(x_2,x_1') & k(x_2,x_2') & \cdots & k(x_2,x_n') \\ \vdots & \vdots & \vdots & \vdots \\ k(x_n,x_1') & k(x_n,x_2') & \cdots & k(x_n,x_n') \end{bmatrix} \tag{2.103}$$

In the standard implementation of the GPR model a training data set of the form $D = \{(x_i,y_i)_{i=1}^n\}$ is assumed. The first step in training this model is to find a representative subset of the training data. For large data sets, it is intractable to use a data covariance matrix K_N which is more than a few hundred points, particularly due to the matrix inverse in Eq. 2.100.

Therefore from D, a subset of data points $D_{train} = \{X_{train}, y_{train}\}$ of length $d < n$ is identified. For analysis purposes, a subset of test data with which to check our model $D_{test} = \{X_{test}, y_{test}\}$ of length $k < n$ is also identified.

Given both our training and testing data sets, three separate covariance matrices are computed (Eq. 2.103): $K_{train,train}$, $K_{test,train}$, and $K_{test,test}$, corresponding to the three covariances found in 2.98. These kernels also require explicit parameters which can be estimated from the training data as $\sigma_f = std(X_{train})$ and $\ell = sqrt(range(Y_{train}))$.

Given our covariance matrices, the inverse matrix from Eq. 2.100 is computed. While one could utilize a brute force matrix inversion, a more elegant solution does exist. Since K_N is a covariance matrix it is inherently a hermitian, positive-definite matrix. Since this is the case we can utilize a Cholesky decomposition approach to compute a lower triangular matrix L which satisfies the condition $L^T L = K$. Given L, it is more computationally efficient to compute the matrix inverse of $(L^T L)^{-1}$. This results in Equation 2.104.

$$K^{-1}_{train,train} = (L^{-1})^T L^{-1} \tag{2.104}$$

Where

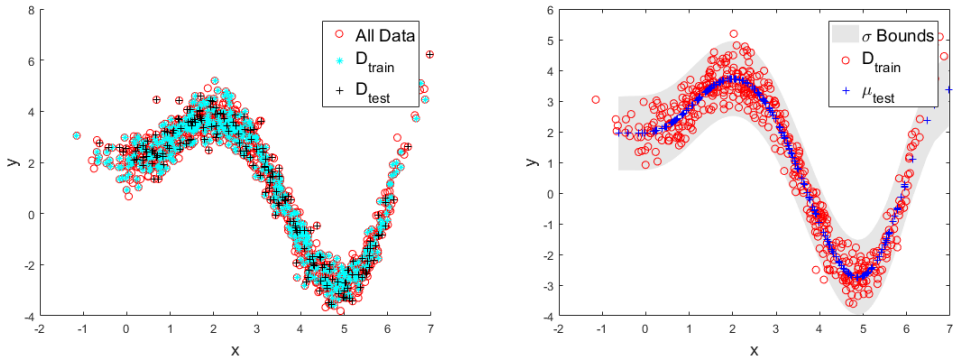
$$L = chol(K_{train,train} + \sigma_n^2 \, I) \tag{2.105}$$

Using the inverse of the covariance $K_{train,train}$, the mean and covariance of the predictive distribution for our test data X_{test} is computed. As in Equations 2.100 and 2.101, the Gaussian process estimates are computed in Equation 2.106- 2.107.

$$\mu_{test} = K_{test,train} \, K^{-1}_{train,train} \, y_{train} \tag{2.106}$$

$$\sigma^2_{test} = K_{test,test} - K_{test,train} \, K^{-1}_{train,train} \, K^T_{test,train} \tag{2.107}$$

This results in a predictive distribution for our test data which is a function of only our training data D_{train}. As an illustrative example, a function of the form $f = xsin(x)$ is used with additive noise applied. A subsample is taken from this data for the training and testing. The results of this test can be found in Figures 2.10 - 2.10b.



(a) Sample Train and Test Data.

(b) Estimated Output for Test Data.

Figure 2.10: Gaussian Process Regression example.

Given this approach, to classify online data one needs to save only the inverse covariance $K^{-1}_{train,train}$ and the training output data y_{train}. Then for online estimation, $K_{test,train}$ and $K_{test,test}$ are computed in order to arrive at the predictive distribution.

The Gaussian Process Regression model provides an elegant solution for the probabilistic modeling of non-linear and non-parametric data. The core problem with GPR models is that for large data sets the covariance matrix K_N becomes very large and thus makes the matrix inverse computationally expense to compute. Additionally for online estimation, this algorithm requires at a minimum $d + d^2$ multiplications ($O(d^2)$) where d is the size of our training data. Furthermore, the naive GPR model provides no inherent logic in sub-sampling the data so that D_{train} is sufficiently representative of the true function shape. Therefore the training data in our model can possibly be data scarce in certain regions. While some work has investigated the use of GPR for classification problems, the standard formulation is designed for regression problems.

2.5 Background Summary

This section has covered a background on both surgical skill evaluation and tissue identification for minimally invasive tools. An overview of machine learning methods has also been covered representing the current state of the art. Based on this review no dynamic discriminant method

exists that can handle the problem of discriminating subtle dynamic differences in overwhelmingly similar data. This problem is prevalent and inescapable in computational surgery. The gaps found in both surgical skill evaluation and tissue identification motivate the evolution of existing machine learning techniques and the development of new techniques to further improve solutions to these open problems in computational surgery. Solving this problem would provide concrete advancements in skill evaluation and tissue identification, which would in turn help ultimately enable safer surgery.

Chapter 3

Proposed Algorithms

In order to fill the gap outlined in Chapter 2, multiple candidate algorithms are explored which satisfy the requirements identified. To do so the following framework is introduced for optimization criteria in a discriminant setting; a generalized discriminant criteria wherein the following error is minimized:

$$\min_{\Phi_1\Phi_2} \quad [(D_u(x_1,\Phi_1)+D_u(x_2,\Phi_2))-\lambda(D_u(x_1,\Phi_2)+D_u(x_2,\Phi_1))] \tag{3.1}$$

Here $D_u(x_i,\Phi_i)$ represents some distance function between data x_i from class i to model Φ_j from class j, given an input u. However, this distance measure can be generalized to probabilities, variances, entropies or other measures distance between data sets. Additionally no requirement is placed on the linearity of this system or the inputs, merely that some distance measurement can be made between the input and system. As is the standard criteria for a distance metric, this measure must meet the following conditions: non-negativity, identity of indiscernibles, symmetry, and the triangle inequality.

Additionally, λ represents a weighting factor to scale between a generative model ($\lambda = 0$) and a discriminative model ($\lambda = large$). This concept can alternately be expressed as ratio between the two components:

$$\min_{\Phi_1\Phi_2} \quad \frac{D_u(x_1,\Phi_1)+D_u(x_2,\Phi_2)}{\lambda(D_u(x_1,\Phi_2)+D_u(x_2,\Phi_1))} \tag{3.2}$$

In either formulation the goal is to identify high density domains, which contain optimal discriminant information, while simultaneously ignoring low density domains. It should be

noted that minimizing Eq. 3.1 reduces to Bayes discriminant function when ignoring overall model fit or bidirectional classification. Bayes discriminant function is given as

$$g_1(x) = p(C_1|x_1) - p(C_2|x_1) \tag{3.3}$$

Where x_1 is the data from class C_1 and $p(\cdot)$ is the conditional probability. This discriminant function can also be stated as the difference between the probability of new data coming from a particular class verse some incorrect class. For discriminant cases, a maximization of this difference is desired. Equation 3.2 is a generalization of LDA, QDA, and KDA methodology, which considers ratios of between-class and within-class scatter. The generalized discriminant criteria is integral to the formulation of the candidate discriminant dynamic algorithms.

The first candidate algorithm focuses on a Discriminant Least Squares (DLS) estimate so as to maximize the probability of correct classification while simultaneously minimizing the instance of incorrect classification by considering all data sets at the same time. The DLS method requires a linear function with parameters that will be estimated over time.

The second algorithm, the Discriminant Phase Portrait (DPP), will use a dynamic state space representation, inspired by phase-portrait derived features, in conjunction with a grid-based probability estimate in order to identify areas of maximum empirical class separability. Then incoming data will be classified based on the new observations location in phase space. The probabilistic classification will be weighted based on the separability in that particular region.

The third candidate algorithm is a variation on the DPP approach. For each class of data to be trained, a Radial Basis Function will be used to get a multi-dimensional probability estimate for all regions of the state space. Using this probability the data will be subsampled to keep only the separable data. Then for online data, a variation of Gaussian Process Regression is used to estimate the class membership for each class. This approach is termed RELIEF-RBF.

A fourth candidate algorithm is a feature specifically derived for use in surgical skill classification. This feature is based on assessing the deviation from a optimal trajectory that a surgeon uses while moving surgical tools. This approach, termed Intent Vectors, is used to classify expert from novice surgeons.

Additionally, for each candidate algorithm a confidence value will be reported. This confidence value will be provided with each classification estimate and correspond to the relative uncertainty and separability of that particular data.

In order to evaluate these algorithm designs and explore their relevance to Computational Surgery and beyond, three initial applications are also specified in Chapter 5.

3.1 Algorithm Development

In order to perform discriminant classification on non-linear and noisy time series systems with high degrees of similarity, a discriminant analysis method is developed which focuses on the discriminant features embedded in dynamic information. Multiple candidate algorithms are proposed which will be used to arrive at an optimal algorithm which achieves all of the requirements listed. Multiple algorithms were developed simultaneously to observe the benefits and downfalls of each unique approach. Each method will require three elements. The system will first require large amounts of class labeled training data sets with which to empirically train the discriminant parameters. For any given application, several time series data sets for each individual class will be recorded and stored along with the classification identifier. The particular elements in each data set will depend on the particular application. Once sufficient data sets for all classes have been compiled the training algorithm will be enacted.

3.1.1 Candidate Algorithm 1: DLS

The first candidate algorithm focuses on a method to maximize discriminability in simple linear models. This approach investigates a parameter vector capable of compromising between a generative and a discriminant model. In order to investigate this, a least squares model is used for a given data vector. The input of this system is then assumed to be a linear combination of the data vector. This approach is termed the Discriminant Least Squares (DLS) method.

The proposed Discriminant Least Squares (DLS) approach focuses on maximizing the probability of correct classification while simultaneously minimizing the instance of incorrect classification by considering all data sets at the same time. The DLS method requires a function linear-in-parameters that will be estimated over time. This algorithm stems from the following optimization for a generalized discriminant criteria wherein the following error is minimized:

$$\min_{\Phi_1\Phi_2} \quad [(D_u(X_1,\Phi_1)+D_u(X_2,\Phi_2)\ldots+D_u(X_n,\Phi_n)) - \lambda(D_u(X_1,\Phi_2)+D_u(X_1,\Phi_3)+\ldots+D_u(X_1,\Phi_n)+\ldots + D_u(X_n,\Phi_1)+D_u(X_1,\Phi_2)+\ldots+D_u(X_n,\Phi_{n-1}))] \quad (3.4)$$

Here $D_u(X,\Phi)$ represents a generic distance metric between data X and a parametric model Φ. This distance metric can be adjusted to use probability or entropy measures. In the case of probability given one data set and two classes (models), (3.4) collapses to Bayes discriminant criteria. For the purpose of demonstration a root mean square error is employed between the data matrix and the input to ensure online tractability (3.5).

$$D_u = \Sigma_t(u(t)-x(t)\Phi)^2 \quad (3.5)$$

This formulation is an attempt to simultaneously minimize the within-class error and maximize the between-class error. The derivation begins with the standard least squares approach. Here a data vector at time t is given by (3.6) for a system with m states.

$$x(t) = [x_1(t), x_2(t), ..., x_m(t)]^T \quad (3.6)$$

Additionally the elements of the data vector may be any arbitrary, possibly non-linear, states $(\dot{x}, \ddot{x}, x^2, etc)$. Then the input at time t is a linear combination of the data using a parameter α_i:

$$u(t) = \alpha_1 x_1(t) + \alpha_2 x_2(t) + ... + \alpha_m x_m(t) \quad (3.7)$$

In matrix form this equation can be expressed for a time series up to time t_n

$$U = X\Phi \quad (3.8)$$

With:

$$U = [u(t_0), u(t_1), ..., u(t_n)]^T \quad (3.9)$$

Where $u(t_i)$ is the system input at time t_i. Here X represents a data matrix where each row is the data vector $x(t)$ as in (3.6) at subsequent time steps:

$$X = \begin{bmatrix} x_1(t_0) & x_2(t_0) & \cdots & x_m(t_0) \\ x_1(t_1) & x_2(t_1) & \cdots & x_m(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_n) & x_2(t_n) & \cdots & x_m(t_n) \end{bmatrix} \tag{3.10}$$

Similarly Φ is the parameter vector comprised of linear parameters α_i for a particular class of data.

$$\Phi = [\alpha_1, \alpha_2, ..., \alpha_m]^T \tag{3.11}$$

In the traditional total least squares generative model the primary goal would be the computation of this parameter vector via matrix pseudo inverse:

$$\Phi = (X^T X)^{-1} X^T U \tag{3.12}$$

However in the discriminant model formulation several potential classes exist, each with unique data matrices $(X_1, X_2, ..., X_w)$, input vectors $(U_1, U_2, ..., U_w)$, and parameter vectors $(\Phi_1, \Phi_2, ..., \Phi_w)$. In this formulation, the goal is not to determine parameter vectors which best fit each class independently but instead to find parameter vectors which jointly maximize separability of each class.

For consistency in naming, the data matrix used in this algorithm is assumed to be given by D_T where each row represents a sample with d dimensions. Similarly υ_T represents an input to the system at each time step. Finally, L_T is a column vector of class labels for each sample in D_T. These class labels take on integer values $L_T \in \{1, 2, 3, \cdots\}$. From this convention, the class specific data matrix can be assigned according to Equation 3.13.

$$X_i = D_T(L_T == i, :) \tag{3.13}$$

For an n^{th} order classification example, the error value is examined for an incorrect classification, i.e., a data matrix from one class mapped by a parameter vector from another class. Per (3.4) for the linear case, the distance function $D(\ \cdot\ )$ is the mean squared error between input and the linear system.

$$
\begin{aligned}
e(t) = & \\
& \sum_t [(U_1 - X_1\Phi_1)^2 + \ldots + (U_n - X_n\Phi_n)^2) \\
& - \lambda((U_1 - X_1\Phi_2)^2 + \ldots + (U_1 - X_1\Phi_n)^2 \\
& + \ldots + \\
& (U_n - X_n\Phi_1)^2 + \ldots + (U_n - X_n\Phi_{n-1})^2)]
\end{aligned} \tag{3.14}
$$

For a simple example, a standard linear system is given by

$$U = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 \tag{3.15}$$

For the sake of brevity, ternary classification is assumed. Using this system and expanding (3.14) results in the following form:

$$
\begin{aligned}
e(t) = & (\bar{X}_1\bar{\Phi}_1 + \bar{X}_2\bar{\Phi}_2 + \bar{X}_3\bar{\Phi}_3) \\
& - \lambda_1(\bar{X}_1\bar{\Phi}_2 + \bar{X}_1\bar{\Phi}_3) \\
& - \lambda_2(\bar{X}_2\bar{\Phi}_1 + \bar{X}_2\bar{\Phi}_3) \\
& - \lambda_3(\bar{X}_3\bar{\Phi}_1 + \bar{X}_3\bar{\Phi}_2)
\end{aligned} \tag{3.16}
$$

Where $\bar{X}_i$ is an element-wise sum (denoted with the Σ subscript) of the following inputs and states over time:

$$
\begin{aligned}
\bar{X}_i = & [u^2, x_1^2, x_2^2, x_3^2, x_4^2, \\
& - 2x_1u, -2x_2u, -2x_3u, -2x_4u, \\
& 2x_1x_2, 2x_1x_3, 2x_1x_4, 2x_2x_3, 2x_2x_4, 2x_3x_4]_\Sigma
\end{aligned} \tag{3.17}
$$

Similarly $\bar{\Phi}_i$ is the expanded parameter vector:

$$
\begin{aligned}
\bar{\Phi}_i = & [1, \alpha_1^2, \alpha_2^2, \alpha_3^2, \alpha_4^2, \\
& \alpha_1, \alpha_2, \alpha_3, \alpha_4, \\
& \alpha_1\alpha_2, \alpha_1\alpha_3, \alpha_1\alpha_4, \alpha_2\alpha_3, \alpha_2\alpha_4, \alpha_3\alpha_4]^T
\end{aligned} \tag{3.18}
$$

$$\begin{aligned} e(t) =& (\bar{X}_1 - \lambda_2\bar{X}_2 - \lambda_3\bar{X}_3)\bar{\Phi}_1 + \\ & (\bar{X}_2 - \lambda_1\bar{X}_1 - \lambda_3\bar{X}_3)\bar{\Phi}_2 + \\ & (\bar{X}_3 - \lambda_1\bar{X}_1 - \lambda_2\bar{X}_2)\bar{\Phi}_3 \end{aligned} \tag{3.19}$$

Here λ_i acts as weighting power to move between a discriminant and generative model for each class. The optimal value of λ in this approach is found empirically by testing a variety of λ values and using the value that yields the highest accuracy when re-classifying the data set. However an analytical solution for the optimal λ value may be obtainable.

To jointly minimize the error (3.19), the partial derivative of all the terms with respect to the parameters (A_i) is taken and each equation is set equal to zero

$$\frac{\partial e}{\partial \alpha_i} = 0 \tag{3.20}$$

Solving these four equations is possible for the linear case since each parameter set is decoupled from other parameters. It can be shown that for a single class this parameter differentiation results in the following solution:

$$\begin{bmatrix} x_1^2 & x_1x_2 & x_1x_3 & x_1x_4 \\ x_1x_2 & x_2^2 & x_2x_3 & x_2x_4 \\ x_1x_3 & x_2x_3 & x_3^2 & x_3x_4 \\ x_1x_4 & x_2x_4 & x_3x_4 & x_4^2 \end{bmatrix}_\Sigma \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} x_1u \\ x_2u \\ x_3u \\ x_4u \end{bmatrix}_\Sigma \tag{3.21}$$

Where the subscript Σ indicates an element wise summation over time for each entry in the matrix of Eq. 3.21. This form is more easily represented with matrix variables:

$$X^*\Phi = U^* \tag{3.22}$$

Where X^* represents the augmented state matrix in (3.21) and U^* represents the augmented input matrix in (3.21). This same differentiation can be repeated for each class in (3.19) resulting in a complete solution for parameters Φ^* which maximize separability between the two classes for a given λ:

$$\Phi_1^* = [X_1^* - \lambda_2X_2^* - \lambda_3X_3^*]^{-1}[U_1^* - \lambda_2U_2^* - \lambda_3U_3^*] \tag{3.23}$$

This solution takes its form directly from the Discriminant criteria reminiscent of Bayes criteria (3.1). This form is thus easily extended to produce optimal discriminant parameters for each class: Φ_n^*.

It is interesting to note that for the m dimensional case this solution approaches a generic form. For a data vector of:

$$X = [x_1,\ x_2, \ldots,\ x_m] \tag{3.24}$$

The generic discriminant solution expands to:

$$X^* = \begin{bmatrix} x_1^2 & x_1x_2 & \ldots & x_1x_m \\ x_1x_2 & x_2^2 & \ldots & x_2x_m \\ \vdots & \vdots & \ddots & \vdots \\ x_1x_m & x_2x_m & \ldots & x_m^2 \end{bmatrix}_{\Sigma} \tag{3.25}$$

In its simplest form this can be stated as a special case of the outer product:

$$X^* = X\ X^T \tag{3.26}$$

Using this generic form, the solution for the discriminant parameters follows the same solution as (3.23) with the corresponding larger input matrix (3.27).

$$U^* = [x_1u, x_2u, \ldots, x_mu]^T \tag{3.27}$$

In addition to identifying discriminant parameters, this training data is also used to identify a noise threshold for online classifications. This noise threshold is used to ignore classification estimates when the separation between the classes is less than the noise within a class. In order to determine this threshold, the following error is computed using the training data and parameter vectors

$$\bar{e}_i = U - D_i\Phi_i \tag{3.28}$$

The quantity $\bar{e}_i$ represents the inherent noise for each class at each time step.

Once a discriminant set of parameters have been identified, the classification can be done online using new data matrix D_x. This matrix is populated at each time step with a new row of

data vectors. Therefore at each time step (τ), classification error values can be computed for each class

$$\Delta_i(\tau) = \sum_{t}^{\tau} U(t) - x(t)\Phi_i^* \tag{3.29}$$

At each time step these classification values can be computed for a given size of D_x. Whichever error is lowest indicates that the corresponding class is the most likely. For a ternary classification example the class estimate is:

$$Class = \begin{cases} 1 & : \quad min(\Delta_i) = \Delta_1 \\ 2 & : \quad min(\Delta_i) = \Delta_2 \\ 3 & : \quad min(\Delta_i) = \Delta_3 \end{cases} \tag{3.30}$$

An additional metric can also be computed via the Δ_i values at any given timestep. To approximate confidence in classification, the α value is computed in (3.31).

$$\alpha = \frac{||\Delta_1 - \Delta_2 - \Delta_3||}{\max(\Delta_1, \Delta_2, \Delta_3)} \tag{3.31}$$

This α value is also used to determine convergence time. Convergence can be assumed to occur when α exceeds a threshold. This threshold can be empirically determined as the value past which classification does not change.

However a second check is performed in order to ensure that the classification at that point has 'good' discriminant ability, i.e. above the noise threshold. This check is based on the separation of the online error values versus the training error (Eq. 3.28):

$$||\Delta_2 - \Delta_1|| > ||\bar{e}_1 + \bar{e}_2|| \tag{3.32}$$

If this check returns true, then a given classification is assumed to have sufficient discriminant weighting. If not then that classification is marked as 'unknown' which is preferable to an unreliable classification and a requirement identified in Section 1.2. Additionally, the confidence value for a classification can be computed as the ratio between the error separations in Eq. 3.32. An algorithm outline for the DLS approach is given in Algorithm Listing 1.

Algorithm 1 Calculate DLS Parameters

Input: Data Matrix D, Input Vector υ, Class Label Vector L, Weighting Parameter λ

Output: DLS Model Parameters Φ

function DLS TRAIN(D, υ, L, λ)
  $classes \leftarrow unique(L)$
  for $i = 1 : classes$ **do**
    $X_i \leftarrow D(L == i, :)$
    $U_i \leftarrow \upsilon(L == i, :)$
  end for
  for $i = 1 : classes$ **do**
    $X_i^* \leftarrow X_i X_i^T$
    $U_i^* \leftarrow U_i X_i^T$
  end for
  for $i = 1 : classes$ **do**
    for $j = 1 : classes$ **do**
      $V1 \leftarrow X_i^*$
      $V2 \leftarrow U_i^*$
      if $i! = j$ **then**
        $V1 -= \lambda X_j^*$
        $V2 -= \lambda U_j^*$
      end if
      $\Phi_i^* = V1^{-1} V2$
    end for
  end for
  return Φ^*
end function

The extension of this algorithm to multi class problems is relatively straightforward. Instead of just three classification errors, a classification error would be computed for each pairwise interaction for all n classes and the corresponding M_i and Φ_i.

While this method does not inherently handle non-linear systems, it does lend useful insight into the discriminant model formulation and subsequent discriminant parameter estimation techniques. Additionally, this formulation does intrinsically account for dynamic features embedded in time series. This method will be used as an initial approach to investigate dynamic discriminant analysis and compared to standard techniques such as LDA or OLS.

3.1.2 Candidate Algorithm 2: DPP

The second candidate algorithm utilizes a dynamic state-space representation of the time-series data. This representation, inspired by phase-portrait derived features allows identification of key discriminating features embedded in the dynamic signal. This method first requires a large training data set in order to identify these key discriminating features. This approach is termed the Discriminant Phase Portrait (DPP) method.

The high level steps for the training algorithm are given as:

- Collect class labeled, time-series dataset for each possible class in the system
- Run empirical training algorithm on generalized phase portrait data for all classes simultaneously.
- Segment N-Dimensional data into grid regions.
- Identify regions which maximize separability in phase space using information gain ratios.
- Using classification parameters from the training, evaluate the same data using a leave-some-out validation scheme.

A large training data set is required to train the grid based model which will subsequently maximize discriminant capability. The specific discriminant function used will be a variation of a weighted probability function in phase portrait space. Therefore one of the key parameters to determine is the weights. These weights will be determined based on the separability between the various classes in a particular range of the phase portrait. In order to identify regions of high separability in the phase portrait first a grid size is specified, then the Probability Density Function (PDF) will be computed for each region.

For each region of the phase portrait a specific weight w_v will be determined which indicates the importance of that particular region in terms of discriminating features. Inspired by Fishers LDA and specifically Eq. 3.2, these weights are based on the ratio of the between-class and within-class probability. When new data comes online, the corresponding region will be identified and the class probability will be weighted according to how separable the classes are in that region. New data which corresponds to a region of low inter-class separation will be given subsequently low weights and data from a region of high inter-class separation will be given relatively high weights.

The training data for this algorithm assumes a data vector at time t given by (3.33) for a system with m dimensions and n samples.

$$x(t) = [x_1(t), x_2(t), ..., x_m(t)]^T \tag{3.33}$$

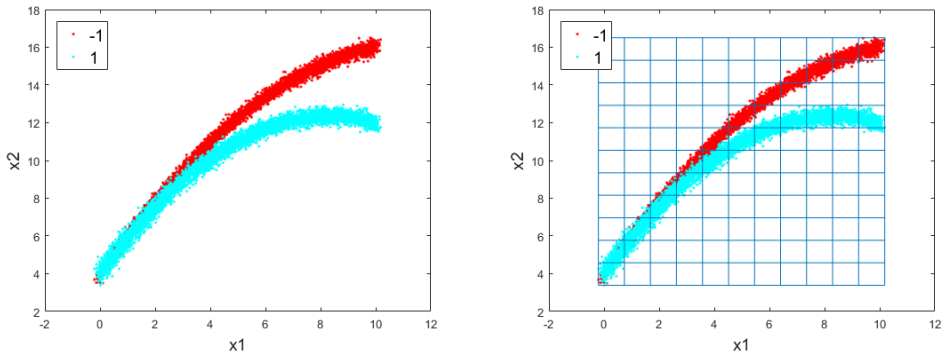
Again the elements of the data vector may be any arbitrary states ($\dot{x}, \ddot{x}, x^2, etc$). The complete data set can be represented as a data matrix (X_T) where each row is a sample $x(t)$ as in (3.33) at subsequent time steps:

$$X_T = \begin{bmatrix} x_1(t_0) & x_2(t_0) & \cdots & x_m(t_0) \\ x_1(t_1) & x_2(t_1) & \cdots & x_m(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_n) & x_2(t_n) & \cdots & x_m(t_n) \end{bmatrix} \tag{3.34}$$

Each sample $X_T(\tau, :)$ in the training set also requires a true class label $C(t)$. The vector of class labels is represented as L_t (Equation 3.35). In the binary classification case, class labels are set to $C = \{-1, 1\}$ which allows for easier computation of classification.

$$L_T = [C(1), C(2), \cdots C(n)]^T \tag{3.35}$$

The first step in training the DPP model involves segmenting the training data X_T into discrete regions via a gridding approach. Given a hyper-parameter *ns* equal to the number of grid elements in each dimension, the N-D training data is divided into distinct regions. For the simulated non-linear data given in Figure 3.1a the data is subdivided into regions as shown in Figure 3.1b. In this example ($ns = 11$).



(a) Non-Linear Class Labeled Data (b) Resultant Grid

Figure 3.1: Sample non-linear data and grid ($ns = 11$)

This grid is automatically populated using the range of the training data and the hyper parameter ns, which controls the coarseness of the grid. The origin of each grid region for the 2D case is computed following Equation 3.36 using indices j,k where $0 < j \leq ns$ and $0 < k \leq ns$

$$G_{j,k} = [min(X_T(:,1)) + jR_{scale}(1), \quad min(X_T(:,2)) + kR_{scale}(2)] \tag{3.36}$$

Where $R_{scale}(i)$ is the width of a single grid region in the i^{th} dimension (Eq. 3.37).

$$R_{scale}(i) = \frac{max(X_T(:,i)) - min(X_T(:,i))}{ns} \tag{3.37}$$

Here $min(X_T(:,i))$ and $max_T(X(:,i))$ refer to the minimum and maximum values of the i^{th} dimension in the training data X_T. The gridding approach is extendable to multiple dimensions via additional indices i.e. $G_{j,k,l,\ldots}$. Given the grid model, the grid index which a sample data point x_o falls into can be identified according to equation 3.38.

$$I[j,k] = \left[\frac{x_o(1) - min(X_T(:,1))}{R_{scale}(1)}, \quad \frac{x_o(2) - min(X_T(:,2))}{R_{scale}(2)} \right] \tag{3.38}$$

In order to compute the weights which will provide the maximum discriminant information, following Eq. 3.2, the between-class separation ratio will be computed. This separability is computed for each region in the training data. The first step to compute these weights is to collect all training data which exists inside a given grid region $G_{j,k}$, this is accomplished by looping through each data point in the training set X_T and determining the grid indices according

to Eq. 3.38. This results in a training data set for a given region ($D_{j,k}$) as well as class labels ($L_{j,k}$) for each data point in the region.

The separability ratio used in this case is a modified Kulback-Leibler (KL) divergence. Whereas the standard KL divergence is one sided ($D_{kl} = \int p\ log(\frac{p}{q})$) we employ a custom variant of KL divergence according to Equation 3.39. Notice here that W_{KL} is not strictly a probability distribution since values range from $-inf$ to inf (Fig. 3.2). This KL variant is beneficial for classification since the values scale equally in the positive and negative direction depending on which class is more likely.

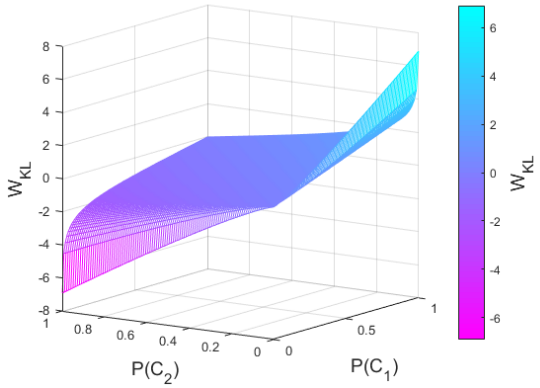


Figure 3.2: W_{KL} weights given probability.

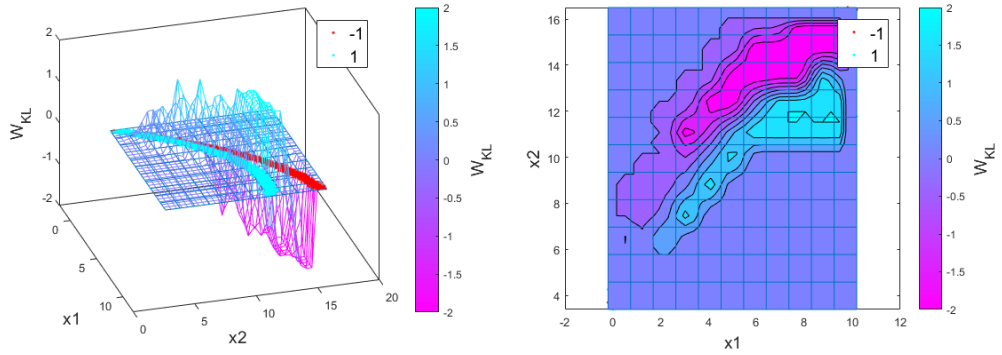
$$W_{KL}(j,k) = log\left(\frac{P(C_1|D_{j,k})}{P(C_2|D_{j,k})}\right) \tag{3.39}$$

Here $P(C_i|D_{j,k})$ is the probability of class C_i given region data $D_{j,k}$ and the corresponding class labels $L_{j,k}$ according to Equation 3.40.

$$P(C_i|D_{j,k}) = \frac{1}{n(j,k)} \sum_{t}^{n(j,k)} (L_{j,k}(t) == C_i(t)) \tag{3.40}$$

Where $n(j,k)$ is the total number of data points in region $G_{j,k}$. Therefore $P(C_i|D_{j,k})$ is the probability of a data point in $D_{j,k}$ being from class C_i. This calculation can be run independently for each region. The effect of the W_{KL} value is to give regions with equal instances of both classes a low weight ($W_{KL} = 0$), whereas regions with high probability of one class and low

probability of other classes will have a high KL weighting. For the sample non-linear data in Fig. 3.1a, we compute the W_{KL} values for each region in Figure 3.3.



(a) W_{KL} Weights (3D) (b) W_{KL} Weights (Contour)

Figure 3.3: Sample non-linear data with W_{KL} weighting for $x1$, $x2$ phase portrait. Note: only regions where there is high between class scatter and low-within class scatter are emphasized.

In Figure 3.3 one observes that the W_{KL} weighting increase as the nonlinear trajectories diverge. Therefore regions are weighted more heavily where class separability is high. However, the W_{KL} weighting does not permit consideration for regions with low density of data. Grid regions may exist where one class has a very high probability but only a very small subset of the training data exists in that region, therefore our confidence of that region's classification quality is diminished. For this reason the relative density of each region $G_{j,k}$ is recomputed as shown in Equation 3.41.

$$\rho(j,k) = \frac{n(j,k)}{max(n)} \tag{3.41}$$

Here $max(n)$ is the maximum number of samples found in any one grid region, therefore this scaling must occur after all regions have been analyzed. The value of ρ is scaled within $0 < \rho < 1$ where $\rho = 1$ occurs at the densest grid.

Given both the W_{KL} weighting and the ρ density, the separability measure $S_{j,k}$ in a given region (j,k) will be computed as the product over sum of these two weightings (Eq. 3.42).

$$S_{j,k} = \frac{W_{KL}(j,k) \cdot \rho(j,k)}{W_{KL} + \rho(j,k)} \tag{3.42}$$

Regions where this ratio is large implies that the KL weighting is high while the density is also relatively high. Similarly, when $S_{j,k}$ is low, either the KL weighting is low or data is scarce, neither of which are optimal regions for discriminant information. In theoretic terms this ensures that only regions with 'good' discriminant capacity are focused on for discriminant weighting. As shown in Figure 3.4, the value of $S_{j,k}$ still assumes both positive and negative values with the magnitude scaled by ρ.

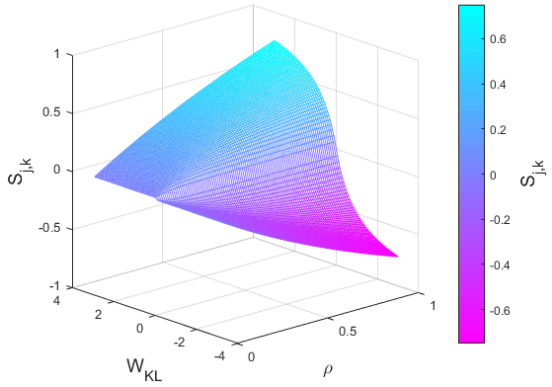
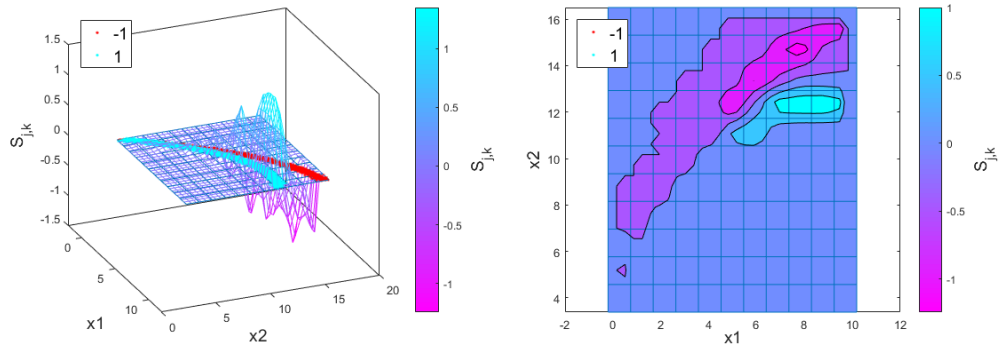


Figure 3.4: $S_{j,k}$ grid element separability given probability ratio W_{KL} and grid confidence based on data presence ρ. Note: as $\rho \to 0$, separability also exhibits $S \to 0$.

The separability measure $S_{j,k}$, is thus used to represent the relative discriminant weighting in each region. Given this weighting, an online classification estimate is computed within each grid region. For the sample non-linear data in Figure 3.1a the $S_{j,k}$ separability is shown in Figure 3.5.



(a) $S_{j,k}$ Separability (3D) (b) $S_{j,k}$ Separability (Contour)

Figure 3.5: Sample non-linear data with $S_{j,k}$ separability for $x1$, $x2$ phase portrait. Note: only regions with *both* a high probability ratio W_{KL} and a high density ρ are emphasized.

Online classification is accomplished using a running sum of $S_{j,k}$ values based on the grid regions that a trajectory passes through. Given the grid model $G_{j,k}$ and the corresponding separability weights, online classification is tractable in real time. For online classification we assume that a time-series trajectory of data (Eq. 3.43) is to be classified.

$$D_{online} = [\chi_1, \chi_2, \cdots \chi_n]^T \tag{3.43}$$

Where χ_i is an n-dimensional sample at timestep i. For each sample in the trajectory, we first determine the corresponding grid indices $I[j,k,...]$ according to Equation 3.38. Once the corresponding grid region $G_{j,k}(\chi_i)$ has been found for a new online sample, the $S_{j,k}$ value for that grid region is added to a running sum to accomplish overall classification. This running sum M_{online} is given in Equation 3.44.

$$M_{online}(t) = \sum_{i=1}^{t} S_{j,k}(\chi_i) \tag{3.44}$$

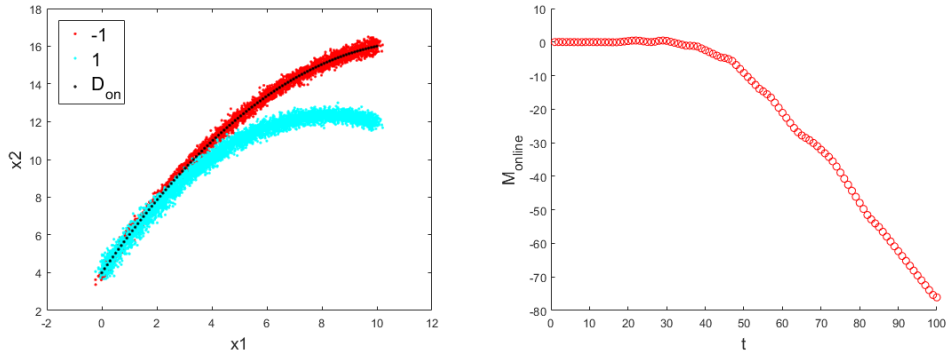
Where $t < n$ is the current timestep index and n is the total number of points in the online trajectory vector. The value of M_{online} will vary depending on the regions the trajectory passes through. Low values of M_{online} indicate that the trajectory has a low classification confidence, i.e. the trajectory is representative of both classes.

Once M_{online} has been computed for all samples in the online trajectory, then the class estimate is taken as the sign of the resultant sum (Eq. 3.45). Similarly the magnitude of the sum

is used to compute the relative confidence of the classification, a core requirement from List 1.2.

$$C_{est} = \begin{cases} -1 & : M_{online} < 0 \\ 1 & : M_{online} > 0 \end{cases} \tag{3.45}$$

For the sample non-linear data given in Figure 3.1a an online classification estimate is computed for a new trajectory D_{online} given in Figure 3.6a. Additionally the running sum of M_{online} values is given in Figure 3.6b. As expected the running sum trends towards negative values resulting in a correct classification ($C_{est} = -1$).



(a) Online Non-Linear Data

(b) $M_{online}(t)$ for all samples in $D_{online}(t)$

Figure 3.6: Online non-linear data classification. The classification score M_{online} tends towards negative values.

A brief summary of the proposed algorithm for candidate 2 (DPP) is provided as pseudo code in Algorithm Listing 2.

Algorithm 2 Compute DPP Model

Input: Data Matrix X, Class Label Vector L, Grid Granularity ns

Output: DPP Model Parameters W, S

function DPP TRAIN(X, L, ns)
  $nclasses \leftarrow unique(L)$
  $ndims \leftarrow size(X, 2)$
  $nsamples \leftarrow size(X, 1)$
  $ngrids \leftarrow ns^{ndims}$
  for $d = 1 : ndims$ **do** ▷ Compute limits and scale for the grid
    $minbound(d) \leftarrow min(X(:, d))$
    $maxbound(d) \leftarrow max(X(:, d))$
    $R_{scale}(d) \leftarrow (maxbound(d) - minbound(d))/ns$
  end for
  for $i = 1 : nsamples$ **do**
    $xtemp \leftarrow X(i, :)$
    $ltemp \leftarrow L(i)$
    $index \leftarrow (xtemp - minbound)/R_{scale}$
    $D(index, ltemp)++$ ▷ Compute counts for each class in each grid
    $n(index)++$
  end for
  for $i = 1 : ngrids$ **do** ▷ Compute class probabilities directly for each grid
    for $c = 1 : nclasses$ **do**
      $P(i, c) = D(i, c)/n(i)$
    end for
    $W(i) = log(P(i, 1)/P(i, 2))$
    $\rho(i) = n(i)/max(n)$
    $S(i) = (W(i)\rho(i))/(W(i) + \rho(i))$
  end for
  $return\ W, S$
end function

The implementation of this algorithm is expandable to multi-dimensional data since the $S_{j,k}$

values are stored in an N-Dimensional matrix with *ns* entries in each dimension. Similarly the grid bounds $G_{j,k}$ are stored in a vector of size (ns^N *x N*). However, this approach is still susceptible to the curse of dimensionality for high dimensional systems with sparse data in a given grid element. It is interesting to note that in the simplest case ($ns = 1$) this algorithm reduces to the naive Bayes classifier, however discriminant capacity increases as the granularity of the gridding is increased. Determination of an optimal *ns* value is currently heuristically determined.

This candidate algorithm achieves the requirements identified in Section 1.2 for dynamic discriminant analysis. This method places no restrictions on the number of states to be evaluated for either the separability measure or the online estimate sum. This method also inherently handles time series data represented in state space form. Additionally, the separability weighting natively constitutes a discriminant model wherein the discriminating features are favored while the common features are ignored. Finally no knowledge of a prior system model or linearity is required.

3.1.3 Candidate Algorithm 3: RELIEF-RBF

The third candidate algorithm is variation on the Algorithm 2 approach. This method utilizes the phase portrait data representation, however instead of analyzing the separability in discrete regions, separability is analyzed across the entire phase portrait. This approach addresses two use cases, first it may be used as feature selection criteria, i.e. identifying the states which contain the most separability. Second, this approach may be used to perform classification for new data, wherein the subset of separable data is used to compute a Gaussian Process model for classification. This model is based on a data clustering technique. This method is termed RELIEF-RBF, noting its similarity to the RELIEFF algorithm (Section 2.4) which inspired it.

It is assumed that this data set will contain a high degree of similarity between classes. If this were not the case, more traditional regression techniques would be used. Because of this similarity, the algorithm should ignore any data that is not separable from the opposite class. In order to do this, regions are identified and ignored that fit the following criteria

- Sample data has a deficient amount data in the surrounding window relative to its own class
- Sample data has a near equal probability from its own class and the opposite class

Similar to DPP, this approach will require a large class labeled training data set. This data must be represented in state space form. This approach allows us to analyze the separability of any subset of d dimensions where $d \leq m$ (m is the total number of dimensions in the data set). For a given class in state space, the training data set can be be represented as X_T (Eq. 3.46) with class labels represented as L_T (Eq. 3.47). In this case class labels can assume any integer value $C = \{1,2,3,...\}$.

$$X_T = \begin{bmatrix} x_1(t_0) & x_2(t_0) & \cdots & x_m(t_0) \\ x_1(t_1) & x_2(t_1) & \cdots & x_m(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_n) & x_2(t_n) & \cdots & x_m(t_n) \end{bmatrix} \tag{3.46}$$

$$L_T = [C(1), C(2), \cdots C(n)]^T \tag{3.47}$$

Here the training data at a given time step (t) is given by a single row of the matrix ($X_T(t,:)$) for a system with m generalized linear-in-parameter states (e.g. $(x, \dot{x}, \dot{x}^2, \ldots)$). It is important to note that for input training data, each dimension must be mean variance scaled before analyzing the separability, this is due to the euclidean distance function used within the squared exponential kernel. Therefore within our model, the mean and variance scaling factors are saved for use in online classification.

Given the class labeled data in state space, the first step is to compute a probability distribution over all training data for a given class. To compute this probability distribution, a multi-dimensional Gaussian Radial Basis Function (RBF) is employed. An RBF is used to estimate a probability distribution given discrete data (Eq. 3.48).

$$\phi(r) = e^{-(\varepsilon r)^2} \tag{3.48}$$

Here r represents the euclidean distance between two m-dimensional data points $r = ||x - x_i||$.

We employ an RBF to estimate the probability density function given within class (hit) and between class (miss) data across any combination of m dimensions. As in the standard RBF all data from all dimensions contribute to the overall probability of that data point. Using the training data set, each point (indexed by i) within the n-sample set is assigned a probability

estimate via RBFs for within class probability (P_{hit}) and between class probability (P_{miss}) (Eq. 3.49-3.50).

$$P_{i,hit} = \frac{1}{N_{hit}} \sum_{j=1}^{N_{hit}} e^{-(\varepsilon \|x_i - x_j\|)^2} \tag{3.49}$$

$$P_{i,miss} = \frac{1}{N_{miss}} \sum_{k=1}^{N_{miss}} e^{-(\varepsilon \|x_i - x_k\|)^2} \tag{3.50}$$

Here x_j represents data points with the same class label as x_i i.e. $\{x_j \in X_T | C_i = C_j\}$. Similarly, x_k represents data points with a different class label than x_i i.e. $\{x_k \in X_T | C_i \neq C_k\}$. The bandwidth variable ε is used to scale the kernel radius given a standard deviation. The value of ε is empirically determined according to Equation 3.51 which is derived from Silverman's rule of thumb bandwidth estimation for Gaussian data.

$$\varepsilon = 1.06\sigma(n^{-1/5}) \tag{3.51}$$

Where $\sigma = std(D_T)$ and n is the number of samples in D_T. Given the class specific probability estimates for each data point, the relative separability of each data point is computed between its hit-class and miss-class. This requires computing the Kullback-Leibler (KL) divergence of each point using both probability estimates (Eq. 3.52).

$$W_{i,rbf} = P_{i,hit} \cdot log(\frac{P_{i,hit}}{P_{i,miss}}) \tag{3.52}$$

Each data point x_i in d-dimensional space ($d \leq m$) is assigned an estimate of separability (i.e. relevance in terms of classification use). The mean relevance weighting from all points in the training data set yields an aggregate estimate of the relevance weighting for that combination of features. This relevance weight can then compared with other combinations to improve feature selection for large, multi-dimensional, numerical data sets.

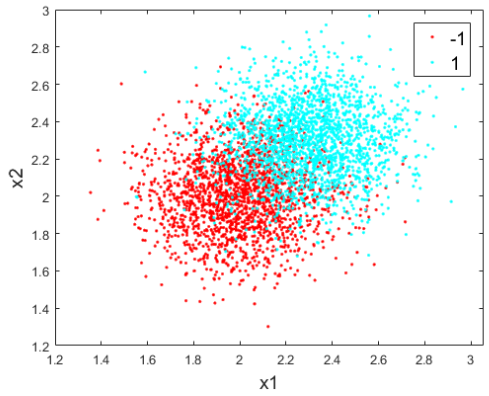
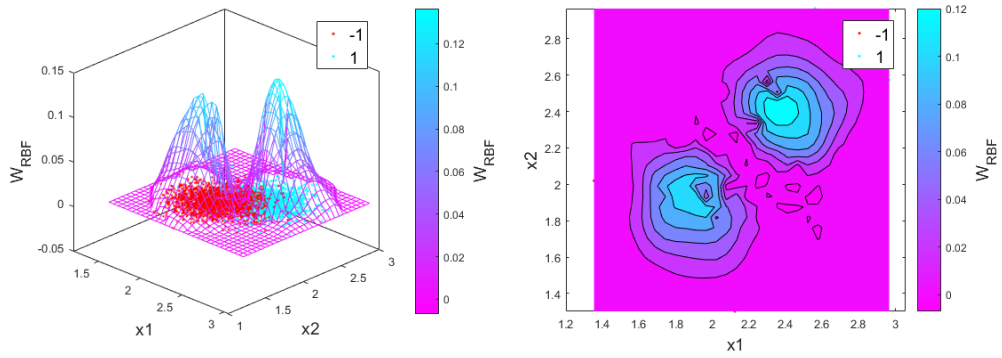


Figure 3.7: Simulated data for two classes.

A two-dimensional example of the relevance weights for two classes of a simulated Gaussian data is given in Figure 3.7. The RELIEF-RBF algorithm rewards only regions with high confidence of separability (high W_{rbf}), while penalizing *both* regions with a prevalence of all classes *and* regions that are data scarce (low W_{rbf}). A plot of the computed W_{rbf} weights for the sample 2D data is given in Figure 3.8.



(a) W_{rbf} Weights (3D)

(b) W_{rbf} Weights (Contour)

Figure 3.8: W_{rbf} weights for sample 2D data. Note: only regions where there is high between class scatter and low-within class scatter are emphasized.

In the simplest sense, this approach can be used to assess the relative separability for a given subset of d dimensions from the training data X_T. By looking at all possible combinations of $d \leq m$ dimensions, a combination can be found that yields the best overall separability (W_{rbf}).

To accomplish this an n-choose-k algorithm is used to determine the possible combinations of states. For a 3 dimensional example, n-choose-k would return the following combinations (Eq. 3.53) where $k = 1 : m$.

$$V = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1,2 \\ 1,3 \\ 2,3 \\ 1,2,3 \end{bmatrix} \tag{3.53}$$

To assess the separability for each combination, each combination $V(k)$ is evaluated and used to isolate the subset of the training data for those dimensions (D_V) as in Equation 3.54.

$$D_V(k) = D_T(:, V(k)) \tag{3.54}$$

Using this subset of the training data, W_{rbf} is recomputed and the mean W_{rbf} is stored. This is repeated for all combinations in 3.53 to find the combination ($\hat{V}$) that gives the maximum separability (Eq. 3.55).

$$\hat{V} = \underset{V}{\operatorname{argmax}}\ mean(W_{rbf} | D_T(:, V(k))) \tag{3.55}$$

While the RELIEF-RBF approach could be limited to determining features with the highest separability, this methodology is also utilized for classification. The classification approach used herein is an extension of the Gaussian Process Regression (GPR) outlined in Section 2.4.16. The only change from feature weighting to classification will be the use of $\{-1, 1\}$ as a our class labels. As previously, noted GPR develops a probabilistic model based on latent functions of the training data. However, GPR provides no inherent data sub-sampling technique and therefore results in intractable models for large training data sets. This gap leads to the use of RELIEF-RBF to effectively subsample the training data to use only data with high separability (i.e. discriminating data).

The overall GPR approach is again derived from the principal that a Gaussian distribution can be completely represented only by a mean and covariance. This distribution can be written

as Equation 3.56.

$$P(f|\Sigma,\mu) = \frac{1}{\sqrt{2\ \pi\ \Sigma^2}}\ exp(-0.5(f-\mu)^T\Sigma^{-1}(f-\mu)) \tag{3.56}$$

Given this form, it is known that the marginal distribution of a Gaussian Process also has a Gaussian Distribution parameterized by a mean function $\mu(x)$ and covariance function $K(x,x')$. The marginal of a Gaussian has the form in Equation 3.57.

$$P(f,g) = \mathcal{N}\left(\begin{bmatrix} a \\ b \end{bmatrix}, \begin{bmatrix} A, C \\ C^T, B \end{bmatrix}\right) \tag{3.57}$$

Furthermore, the conditional of a Gaussian process also has a Gaussian distribution allowing us to write the conditional distribution as Equation 3.58 (see [108] for a complete derivation).

$$P(f|g) = \mathcal{N}(a + CB^{-1}(y-b), A - CB^{-1}C^T) \tag{3.58}$$

Hence the conditional distribution of our data is now a function of the mean and covariance. All that is left to do is compute the mean and covariance functions given the training data X, y. In the standard GPR derivation, a zero mean Gaussian is assumed with a covariance given by a kernel matrix as our marginal likelihood (Eq. 3.59).

$$P(y|X) = \mathcal{N}(0, K_n + \sigma^2 I) \tag{3.59}$$

Therefore our predictive distribution for online data x_* takes the form of Equation 3.60.

$$P(y_*|x_*, y, X) = \mathcal{N}(\mu_*, \sigma_*^2) \tag{3.60}$$

Where now the form of our mean and covariance functions are given by Equations 3.61, 3.62 respectively.

$$\mu_* = K_{test,train}\ (K_{train,train} + \sigma_n I)^{-1}\ y_{train} \tag{3.61}$$

$$\sigma_*^2 = K_{test,test} - K_{test,train}\ (K_{train,train} + \sigma_n I)^{-1}\ K_{test,train}^T \tag{3.62}$$

As in the standard formulation, this allows us to compute the predicted mean and covariance of online data x_* given only the training data X, y. Here again K represents the covariance kernel matrix which takes the form of a squared exponential (similar to Eq. 3.48) in Equation 3.63.

$$K(X,X') = \begin{bmatrix} k(x_1,x'_1) & k(x_1,x'_2) & \cdots & k(x_1,x'_n) \\ k(x_2,x'_1) & k(x_2,x'_2) & \cdots & k(x_2,x'_n) \\ \vdots & \vdots & \vdots & \vdots \\ k(x_n,x'_1) & k(x_n,x'_2) & \cdots & k(x_n,x'_n) \end{bmatrix} \tag{3.63}$$

Where

$$k(x,x') = \sigma_f^2 \, exp(-\frac{||x-x'||^2}{2\ell^2}) \tag{3.64}$$

Here again, σ_f^2, σ_n^2, and ℓ are tunable parameters that effect the marginal likelihood. For this work an estimate for these terms is based off the standard deviation of the training data. For the observation noise, $\sigma_n^2 = std(y_{train})$. For the process noise, $\sigma_f^2 = mean(std(X_{train}))$. And for the characteristic length scale, $\ell = 1.06 * \sigma_n^2 * n^{-1/5}$, which is Silverman's rule of thumb for bandwidth and is a crude approximation to the average distance between peaks.

The key for online prediction is the covariance kernel matrix $K_{train,train}$. This kernel is computed using the training data X_T. However, for large data sets this matrix becomes size $n \times n$ and also requires a matrix inverse. For this reason, the RELIEF-RBF approach is employed to choose a subset of the training data. This subset $\hat{X}_T$ is chosen as follows.

Given the full training data X_T with m features, n samples, and class labels $L_T \in \{-1, 1\}$, we recompute $W_{i,rbf}$ for each point in the training data set according to Eq. 3.52. This results in a vector of relevance weights W_T for the training data (Eq. 3.65). In the case of the simulated data from Figure 3.6, this results in a vector of weights shown in Figure 3.9.

$$W_T = [W_{1,rbf}, W_{2,rbf}, \cdots W_{n,rbf}]^T \tag{3.65}$$

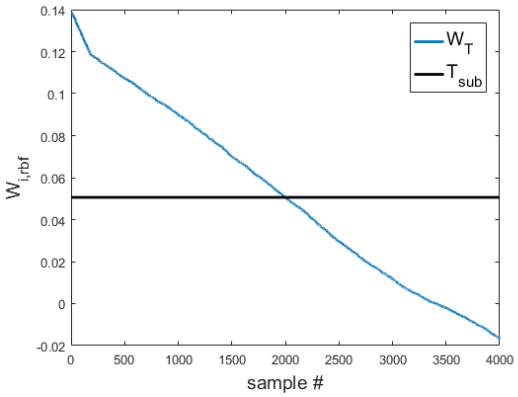


Figure 3.9: W_T weight vector with thresholds ($r_w = 0.5$)

Given the sorted separability weights W_T a subset ratio r_W can be specified and used to assemble $\hat{X}_T$ from the training data. This subset ratio $0 < r_W < 1$ is used to determine the size $s = r_w n$ of $\hat{X}_T$. Once the number of samples to take from X_T has been determined, it is a matter of finding a threshold $T_{sub} = W_T(:,s)$ (Figure 3.9). All samples are taken from X_T for which the separability exceeds this ratio (Eq. 3.66).

$$\hat{X}_T = \{X_T(i) \mid W_T(i) \geq T_{sub}\} \tag{3.66}$$

In the case of the simulated data from Figure 3.6, a subset ratio of $r_W = 0.5$ (which intuitively corresponds to the most separable half of the data) results in a subsampling of the data as shown in Figure 3.10. Conceptually, the new subset of training data consists of the points in the original data with the highest separability.

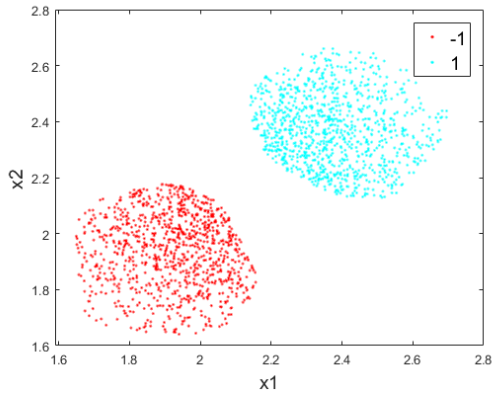


Figure 3.10: $\hat{X}_T$ subset state space data identified to be highly separable ($r_W = 0.5$)

Using this $\hat{X}_T$ subset, and corresponding labels $\hat{L}_T$, the GPR model can now be trained with a significantly smaller $K_{train,train}$ covariance matrix. Additionally, this covariance will only exist in regions with a high confidence of class separability. This can help mitigate issues with online processing speed and curse of dimensionality considerations. In order to train the GPR classification model only the inverse covariance matrix (Eq. 3.67) is computed. The model then consists of storing LK, $\hat{X}_T$, $\hat{L}_T$, and the mean variance scaling factors for online classification.

$$LK = (K_{train,train} + \sigma_n I)^{-1} \tag{3.67}$$

In contrast to typical GPR, our classification approach does not use continuous values for the output y, instead y takes on the class label values $\{-1,1\}$ directly. This approach is sometimes referred to as a Label Regression Method (LR). While LR does not result in a proper probabilistic distribution, it still allows a Gaussian process based classification. For this approach the mean and covariance functions from Equations 3.61-3.62 were used. Where now $X_{train} = \hat{X}_T$ and $y_{train} = \hat{L}_T \in \{-1,1\}$. With these substitutions, the Gaussian Process covariance kernels take the form of Equations 3.68-3.69.

$$K_{train,train} = K(\hat{X}_T, \hat{X}_T) \tag{3.68}$$

$$K_{test,train} = K(x_i, \hat{X}_T) \tag{3.69}$$

This permits the classification of a given online data point x_i into the $-1, 1$ class labels (Eq. 3.70). The output of μ_* will not necessary take on the discrete class labels, however classification will be based on the sign of μ_* with confidence based on the magnitude (Eq. 3.71).

$$\mu_* = K_{test,train}\, LK\, \hat{L}_T \tag{3.70}$$

$$L_i = sign(\mu_*(x_i)) \tag{3.71}$$

In the case of online classification of time series data, a particular trajectory sample can be represented as a vector of samples $X_{on} = (x_{t0}, x_{t1}, \cdots, x_{tk})$. This vector will be classified by the following steps. An individual sample $x_{t=i}$ will be evaluated relative to the GPR predictive distribution to yield an individual classification $\mu_{t=i}$.

For any given sample from an online trajectory, the classification mean will be added to a running sum (Eq. 3.72) which will allow an overall class estimate for the time series (Eq. 3.73). The confidence on this sum can either be extracted from the magnitude of the individual classifications or a weighting factor based on the covariances.

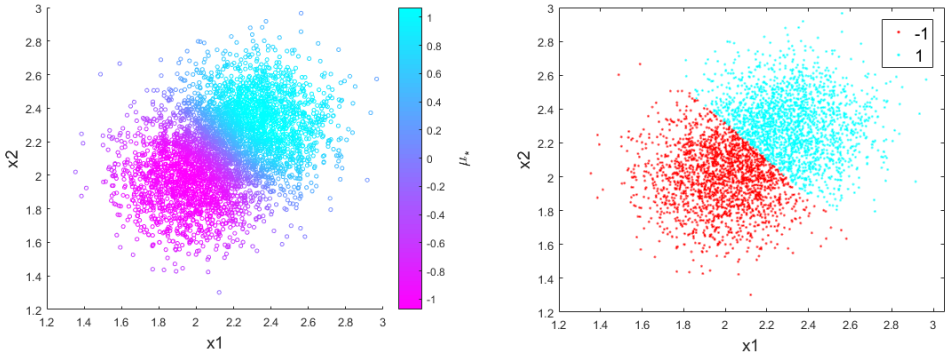
$$L_{on} = \sum_{t=1}^{k} \mu_*(x_t) \tag{3.72}$$

$$C_{on} = sign(L_{on}) \tag{3.73}$$

$$S_{on} = \frac{||L_{on}||}{k} \tag{3.74}$$

Where S_{on} indicates the confidence in these classification based on the magnitude of the running sum. At any given timestep for the online data, the sign of the sum (C_{on}) will be taken as the most likely class estimate.

In the case of the simulated data from Figure 3.6, the classification model was trained using the two class $\hat{X}_T$ data from Figure 3.10. The class estimate value μ_* for each individual data point is given in Figure 3.11a.



(a) μ_* Classification Estimate (b) L_i Discrete Classification

Figure 3.11: RELIEF-RBF classification for sample 2D data.

As evident by Figure 3.11, classification of all data points (including inseparable data) in the full data set X_T is achieved using only the subset of separable points $\hat{X}_T$ as training data. Using the L_i classification for this data set yields discrete classification estimates (Fig. 3.11b). This approach with this simplistic data yields an accuracy of 85%. However, the feasible extension to more complicated and inseparable data is evident since the model would be trained with only the separable portion of the data.

A brief summary of the proposed algorithm for subsampling the separable data (Candidate 3: RELIEF-RBF) is given as pseudo code in Algorithm Listing 3.

Algorithm 3 Subsample with RELIEF-RBF Weights

Input: Data Matrix X, Class Label Vector L, Subset Ratio r_w

Output: Subsampled Data $\hat{X}$

function RELIEFRBF(X, L, r_w)
  $nsamples \leftarrow size(X, 1)$
  $nkeep \leftarrow nsamples \cdot r_w$
  for $i = 1 : nsamples$ **do**
    $S_{hit} = S_{miss} = 0$
    $n_{hit} = n_{miss} = 0$
    $xtemp \leftarrow X(i,:)$
    for $j = 1 : nsamples$ **do** ▷ Compute RBF for hit and miss
      if $L(i) == L(j)$ **then**
        $S_{hit} += exp(-\varepsilon \cdot ||xtemp - X(j,:)||^2)$
        $n_{hit}++$
      else if $L(i) != L(j)$ **then**
        $S_{miss} += exp(-\varepsilon \cdot ||xtemp - X(j,:)||^2)$
        $n_{miss}++$
      end if
    end for
    $P_{hit}(i) \leftarrow S_{hit}/n_{hit}$
    $P_{miss}(i) \leftarrow S_{miss}/n_{miss}$
    $W(i) \leftarrow P_{hit}(i) \cdot log(P_{hit}(i)/P_{miss}(i))$ ▷ Compute KL weight
  end for
  $WSorted \leftarrow sort(W)$ ▷ Sort all weights
  for $i = 1 : nkeep$ **do** ▷ Store the best data points
    $\hat{X} \leftarrow X(W(i) == WSorted(i),:)$
  end for
  $return\ \hat{X}$
end function

This method, while similar to the DPP approach, has certain distinct advantages; namely it avoids the restrictions of a grid based method since the probability distribution is computed

over the entire phase space. Additionally, the RELIEF-RBF weighting ensures only areas with high separability are used in training.

3.1.4 Candidate Algorithm 4: Intent Vectors

The fourth candidate algorithm is departure from the previous algorithms. This algorithm is not a generic machine learning algorithm but instead a derived feature specific to surgical skill evaluation. 'Intent Vectors' is a novel motion statistic for surgical skill classification. The 'Intent Vectors' statistic is based on the overall goal of a motion segment using a surgical tool (Fig. 3.12). Using the starting and ending location of a motion segment as endpoints, a vector is computed which represents the ultimate goal of that segment. It is assumed that this Intent Vector is the ideal line of motion for a given segment, this allows the computation of metrics which represent the amount of deviation from this optimal trajectory.

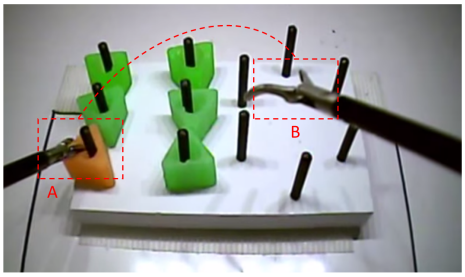


Figure 3.12: FLS Peg Transfer segment.

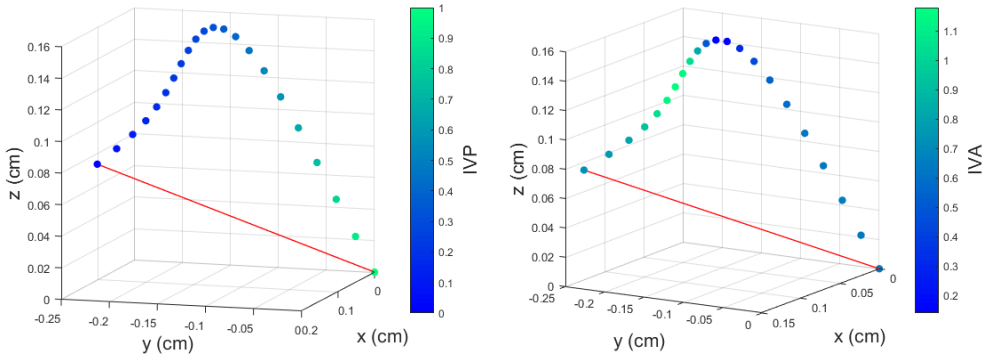
For a segment of Cartesian tool position data of length N, $\Psi = [D_1, D_2, \cdots, D_N]$ where $D_i = [x, y, z]$ represents the 3D location at time $t = i$. The Intent Vector is then computed in Eq. 3.75.

$$\overrightarrow{IV} = \frac{D_N - D_1}{\|D_N - D_1\|} \tag{3.75}$$

From this Intent Vector the progress of each point in Ψ along this line can contextualize other actions relative to the ultimate trajectory. The Intent Vector Progress value (IVP) is computed according to Equation 3.76 using a dot product operator and scaled by the magnitude of the Intent Vector (thus fixing the starting and ending points at 0 and 1). An illustrative example

is given in Figure 3.13a.

$$IVP_i = \frac{(D_i - D_1) \cdot \overrightarrow{IV}}{\|D_N - D_1\|} \tag{3.76}$$



(a) Intent Vector Progress in 3D

(b) Intent Vector Angle in 3D

Figure 3.13: Intent Vector measures for sample Cartesian motion segment of a surgical tool.

The Intent Vector framework also includes the Intent Vector Angle (*IVA*): the angle of motion relative to the overall angle of the Intent Vector. *IVA* is computed for each point in Ψ by taking the difference at a given point in time between the current tool location and the previous location $(D_i - D_{i-1})$ which is then normalized to give a unit vector in 3D space (S_i). This instantaneous unit vector can thus be compared with the overall intention, indicating the degree to which the tool is moving in the correct direction or doubling back (Eqs. 3.77 - 3.78).

$$S_i = \frac{D_i - D_{i-1}}{\|D_i - D_{i-1}\|} \tag{3.77}$$

$$IVA_i = cos^{-1}(S_i \cdot \overrightarrow{IV}) \tag{3.78}$$

The value of *IVA* is bounded between $0 < IVA < \pi$ since it is of no concern which direction the angle differs from the overall intent. This angle can instead be thought of as a heading error. An illustrative example is given in Figure 3.13b. The Intent Vector framework was implemented for all motion segments within the EDGE data set (Section 5.3). For each task the *IVA* and *IVP* measures were compiled into a 2D feature vector with corresponding skill labels. A plot of *IVA* and *IVP* for a surgical Suturing task can be found in Figure 3.14.

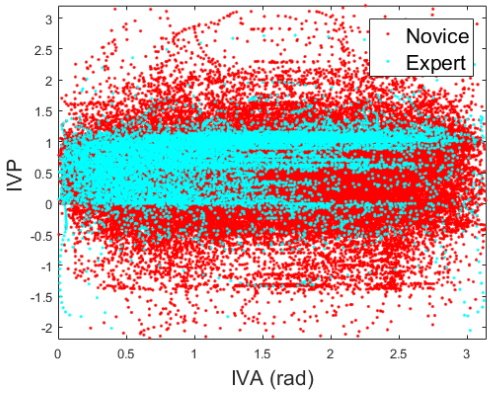


Figure 3.14: IVA, IVP space for a suturing task.

We employ the 'Intent Vectors' feature in a classification scheme intended to classify expert vs novice surgeons based on surgical motion data. This feature was applied to laparoscopic tool motion collected using a dry lab laparoscopic trainer. The details of the EDGE laparoscopic data set can be found in Section 5.3.

Given the high-degree of similarity in the Intent Vector space for expert and novice surgeons, to use the Intent Vector data within a classification scheme a classification approach which focuses on deviations from the region of high expert probability was employed. The region in 2D IVA-IVP space with the highest density of expert surgical motion is identified. A modified version of the RELIEF-RBF algorithm is then employed in order to threshold the relevance weights for the Expert class (Eq. 3.79).

$$W_{i,exp} = P_{i,exp} \cdot log(\frac{P_{i,exp}}{P_{i,nov}}); \tag{3.79}$$

Here $W_{exp} = W_{rbf}$ from Eq. 3.52 where Expert is the *hit* class. All training data is assigned a relevance weight relative to the Expert data. A threshold on $W_{i,exp}$ is computed using an information gain maximization similar to the typical decision stump algorithm [104]. A threshold (T_w) is identified such that classification of the Intent Vector data follows Eq. 3.80 and maximizes the information gain ($IG = H(Y|X) - H(Y)$) for classification ($Y = skilllevel$) given

($X = [IVA, IVP]$).

$$Y = \begin{cases} Novice, & W_{exp}(X) < T_w \\ Expert, & W_{exp}(X) \geq T_w \end{cases} \tag{3.80}$$

Using the relevance weight threshold, all expert data in $[IVA, IVP]$ space above T_w is retained as 'True Expert Data' (Fig. 6.34a) then a Gaussian probability model is trained for online classification ($P_{exp}(X|\mu, \sigma)$) (Fig. 6.34b). A threshold value for this Gaussian model (T_p) is found by taking the $P_{exp}(X)$ at the minimum $W_{i,exp}(X) > T_w$ value.

The next step is to classify each individual time-indexed data point within a given segment for a specific surgeon. For surgeon (g) and segment (s) the time series data is given as $\Lambda_{g,s} = [\lambda_1, \lambda_2, \cdots, \lambda_N]$ where $\lambda_i = [IVA, IVP]$ at time $t = i$. Using $P_{exp}(X|\mu, \sigma)$ each data point is classified as 1 or 0 to signify Novice or Expert, respectively (Eq. 3.81). Values where $y_i = 1$ are considered a 'demerit' for behaving like a Novice and are used in the overall evaluation of the motion.

$$y_i = \begin{cases} 1, & P_{exp}(\lambda_i) < T_p \\ 0, & P_{exp}(\lambda_i) \geq T_p \end{cases} \tag{3.81}$$

Using this y_i value an estimate of the per time-step accuracy can be computed. A correct per time-step classification corresponds to $y_i = 1$ for an 'Obvious Novice' and $y_i = 0$ for an 'Obvious Expert' for any given time-step. Accuracy is then derived from the percentage of time steps where Novices behaved like Novices (received demerits) and Experts behaved like Experts (did not receive demerits).

Overall segment classification requires a secondary threshold on the demerit counts. Given a vector of time-indexed motion demerits $q_{g,s} = [y_1, y_2, \cdots, y_N]$ a mean score for that particular segment $SK_{g,s} = mean(q_{g,s})$ is computed. Given the $1, 0$ labels this score has the effect of being very low for frequent Expert motions and higher if motions fall outside the 'True Expert' model (many Novice demerits). A threshold is trained based on the average SK scores (T_{sk}) for Expert and Novice Surgeons using a decision stump approach. A Leave-One-User-Out scheme per skill group (LOUOpG) is employed (i.e. leave one obvious novice and one obvious expert out

per training) and test each left-out surgeon based on all motion segments (Eq. 3.82).

$$C_g = \begin{cases} Novice, & mean(SK_{g,s}) > T_{sk} \\ Expert, & mean(SK_{g,s}) \leq T_{sk} \end{cases} \tag{3.82}$$

For each LOUOpG iteration, all relevant measures and thresholds are recomputed i.e. W_{exp}, T_w, T_p, and T_{sk} based on the training data set alone, therefore limiting overfitting for the validation data. This Intent Vector approach is applied to the EDGE data set outlined in Section 5.3 for binary classification.

In order to compare the accuracy of our classification approach, previously validated aggregate task metrics as highlighted in [19] were utilized. For this comparison a feature vector was used, comprised of Tool Path Length, Economy of Motion (Eq. 3.83), Motion Smoothness, and Motion Curvature (Eq. 3.84, where $\dot{r} = \|\dot{x}, \dot{y}, \dot{z}\|$) ($\bar{\chi}$ = [PL,EOM,MS,MC]). A Linear Discriminant Analysis (LDA) classifier (class based means and covariances, equal weighting) was trained on this feature vector to classify skill levels. Again, a LOUOpG cross validation was employed with this classifier. Classification was also examined using a combination of Intent Vectors and aggregate metrics with combined feature vector $\hat{\chi} = [\bar{\chi}, mean(SK_{g,s})]$. Again a standard LDA classifier was utilized in a LOUOpG cross validation to classify a complete task.

$$EOM = \frac{Path\ Length}{Task\ Time} \tag{3.83}$$

$$MC = \frac{\dot{r} \times \ddot{r}}{|\dot{r}|^3} \tag{3.84}$$

3.2 MAC Criterion (Surgical Skill Evaluation Only)

As part of developing algorithms for surgical skill level classification, the Minimally Acceptable Classification (MAC) Criterion was also developed for use in surgical skill evaluation. This MAC criterion is intended to act as a minimal benchmark when developing new features and algorithms for surgical skill evaluation. The MAC criterion for computational skill evaluation states that given an obvious Novice and an obvious Expert, the classification accuracy must be 100%. Some misclassification may be acceptable between other skill levels, e.g. Experts vs. Master or Intermediate vs. Expert, but not an *obvious* Novice vs. *obvious* Expert. Here obvious

Novices are defined as subjects who should never be allowed to operate (always disqualified) and obvious Experts as subjects who should never be disqualified from operating. Surgery requires this stipulation given that patently unqualified surgeons endanger lives. Often, such a large difference is very evident via task time or a casual viewer watching a video [111]. Therefore a rigorous motion analysis algorithm should meet this minimal performance benchmark in order to justify its cost and use. While this is not a sufficient criteria, it does provide a *minimal* necessary criterion to use as a baseline in this field. A classification accuracy of 100% must be demonstrable as a minimal criteria for surgical skill classification. This requires stating both the classifier performance under Leave-One-User-Out level cross validation and enumerating its useful benefits over existing methods like summary metrics (e.g. task time). The MAC criterion is explicitly used to evaluate the Intent Vector approach and other algorithms as they are applied to surgical skill evaluation data.

It is herein proposed that the MAC criterion be adopted in surgical skill research as a minimal benchmark for a surgical skill classifier. Otherwise, the cost or complexity of sophisticated algorithms may not be justified. Using MAC also demands more carefully chosen ground truth skill categories to ensure accurate establishment of the ground truth, e.g. combining multiple criteria such as OSATS review, caseload, and procedural metrics. Failure to establish such a clean ground truth may hamper scientific progress in skill evaluation research.

3.3 Benchmark Algorithms for Comparison

While these algorithms represent novel approaches to machine learning given dynamic data, it is also important to understand their performance with respect to proven, established approaches. For this reason two common machine learning algorithms: Neural Networks (NN) and Random Forests (RF) were also applied to the sample data sets to serve as a benchmark for performance comparison. An overview of these two algorithms is given in Sections 2.4.15 - 2.4.14. The use of these existing algorithms in contrast to the proposed algorithms will allow us to observe any potential improvements in performance.

The Neural Network implementation used was the Matlab Neural Network Toolbox (Mathworks Inc. Nattick, MA). The Random Forest implementation used was the Matlab TreeBagger toolbox. For the Neural Network training, the Levenberg-Marquardt back-propagation algorithm is utilized. Unless otherwise noted, the activation functions used were sigmoidal. For

the Neural Network the hyperparameters to be set were the number of hidden layers and the number of nodes. For the Random Forest training, the only hyperparameter to be varied was the number of trees.

As is the standard, the input data for the NN model was mean-variance scaled in order to allow the full range of values in the sigmoidal activation function. Additionally, each NN model was retrained three times in order to avoid local minima in the cost function. The values from the output node for both NN and RF were trained to $Y_{out} = [0, 1]$, corresponding to class labels from each data set. For trajectory classification, the mean of the output values was used for all points in the trajectory. Classification was then performed according to Equation 3.85.

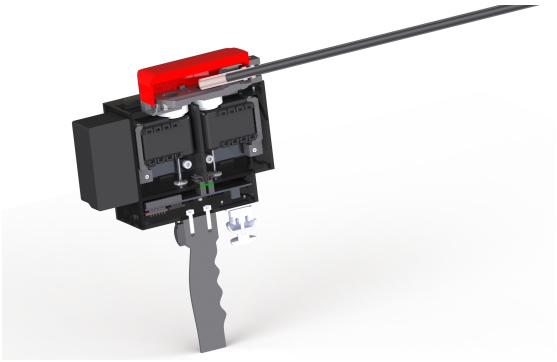
$$C_{est} = \begin{cases} 0 & : mean(Y_{out}) < 0.5 \\ 1 & : mean(Y_{out}) \geq 0.5 \end{cases} \tag{3.85}$$

The NN and RF implementations used allow classification both at each time step in a trajectory, as well as for the trajectory as a whole. For all data sets a leave-one-out cross validation was employed.

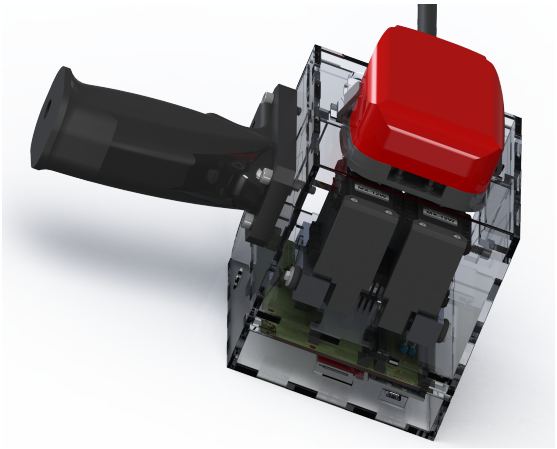
Chapter 4

Hardware Design

In order to perform tissue identification and force estimation in-vivo, a new minimally invasive 'Smart Tool' grasper (Fig. 4.1) was designed in conjunction with ongoing research in the Medical Robotics and Devices Lab. This setup is comprised of a custom hardware unit attached to a da Vinci Si EndoWrist surgical grasper. The mechatronic device is actuated via motors on the proximal end and measures both force and position estimates throughout the grasp using load cells and encoders, respectively.



(a) Smart Tool Overview



(b) Smart Tool with Clear Walls

Figure 4.1: Smart Tool overview.

All measurements are taken at the spindles on the proximal end of the tool as to not disturb the distal grasping end and to provide a surrogate for the torque (motor current) and position (encoder counts) already present in the da Vinci control loop. The mechanical hardware (Fig. 4.2) consists of two Dynamixel MX-12 Servo Motors (Robotis Inc. Lake Forest, CA). These servo motors contain a magnetic encoder which provides 2048 bit resolution. The DC motor within each servo provides 1.5Nm of stall torque.

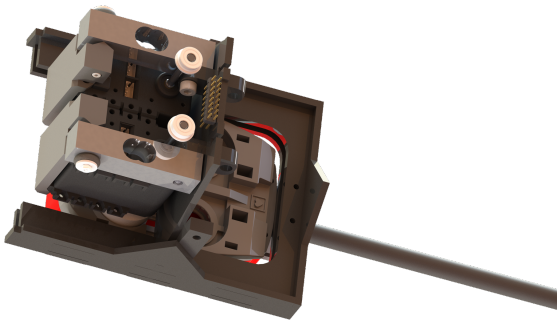


Figure 4.2: Smart Tool motors.

In order to measure the reaction torque at the proximal end, a load cell is utilized to measure the reaction torque on the servo motor casing (Fig. 4.3). The servo motor is affixed only to the spindle on the bottom of the da Vinci tool and a single screw through the moving arm of the load cell. In this fashion an estimate of motor torque is obtained by measuring the reaction force experienced at the free end of the load cell (Fig. 4.4). A 3133 Micro Load Cell (Phidgets, Inc. Calgary, Alberta) was used. This load cell has a range of $0-5kg$, well within the range of expected torques for the cable driven surgical tools.

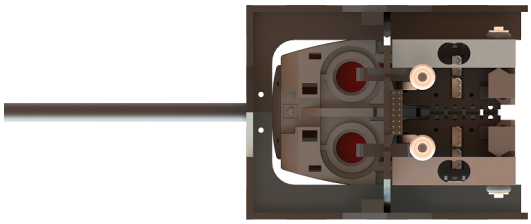


Figure 4.3: Smart Tool load cells.

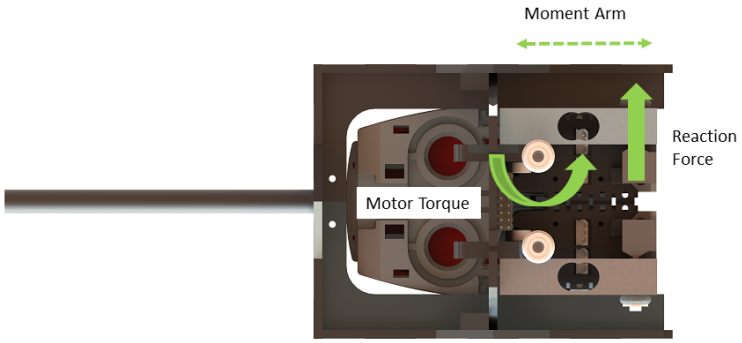


Figure 4.4: Smart Tool load cell force diagram.

The form of this reaction torque is given in Equation 4.1.

$$T_{est} = F_{cell} \times L_{motor} \tag{4.1}$$

Here F_{cell} is the force reading obtained by the load cell and L_{motor} is the length of the servo motor body. According to CAD design and physical measurements the length of this servo motor is $0.037m$. Additionally each load cell was calibrated to Newtons using calibration weights. The specific da Vinci tool used for data collection was the Maryland Bipolar Forceps (Fig. 4.5). The grasper jaws on this tool were measured to have a surface area $A_{jaw} = 33.2mm^2$. This surface area value is used in the computation of estimated stress values given force (Eq. 4.2).

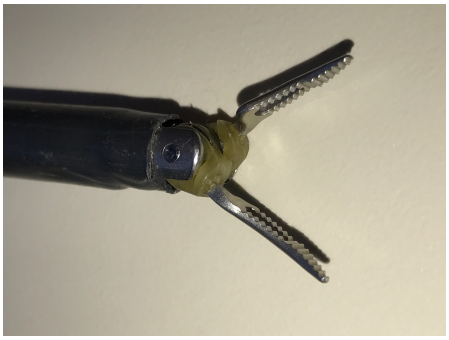


Figure 4.5: Marlyand Bipolar Forceps.

$$\sigma_{est} = \frac{F_{jaw}}{A_{jaw}} \tag{4.2}$$

The supporting structure and casing for the smart tool was designed in CAD and manufactured through a combination of laser cutting acrylic and 3D printing ABS. The finished product with an attached da Vinci grasper is shown in Figure 4.6.

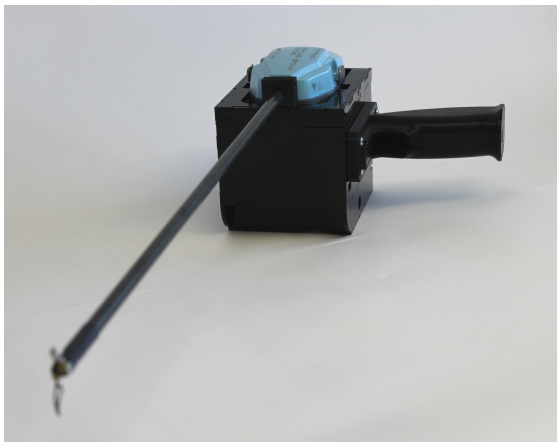


Figure 4.6: Smart Tool physical realization.

The electrical hardware for the smart tool was an integral component for data collection and real time control. The primary functions of the electronic circuit design was the actuation of the servo motors, the reading of the servo motor position, and the reading of the load cell signal. An image of the custom Smart Tool electronics can be found in Figure 4.7.

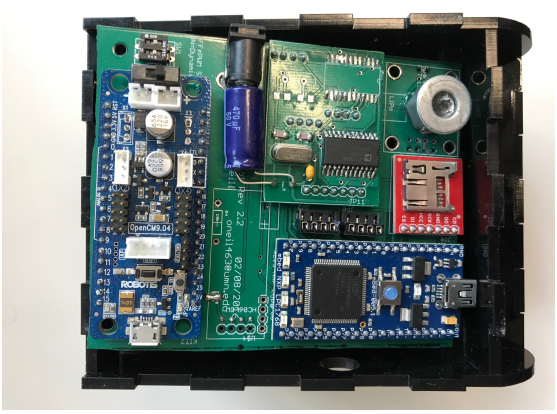


Figure 4.7: Smart Tool electronics.

The electronics board consists of 4 primary components. The first component is an mbed NXP LPC1768 microcontroller (ARM Holdings, Camrbidge, UK). This microcontroller is comprised of an ARM Cortex M4 chip This board is utilized for high level data collection and logic. The second component is a Robotis OpenCM9.04 logic board. This board is used to control the servo motors and report angular position. The third component is an AD7730 Analog to Digital Converter chip (Analog Devices, Norwood, MA). This chip is a 24-bit analog-to-digital converter (ADC) for reading the load cell data. The fourth component is an SPI based micro-sd card reader. This allows for fast data collection at $1kHz$. All code for data collection was written in compiled C for the mbed microcontroller.

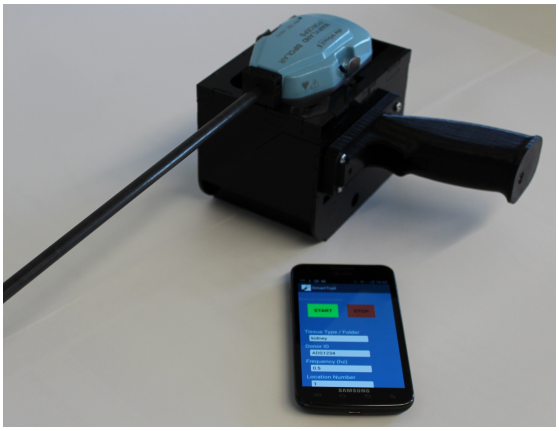


Figure 4.8: Smart Tool with data collection App.

The microcontrollers communicate with a custom Android application (Fig. 4.8) which provides input of the experiment parameters and metadata, as well as giving the command to begin data collection. Input fields include grasp frequency, number of grasps, and tissue type metadata. Communication between the Android application and the microcontrollers is achieved using a serial-over-Bluetooth protocol.

A key element in collecting stress-strain data with the Smart Tool was determining first touch, i.e. the angle of the grasper when it first comes into contact with the tissue sample (θ_i). Strain is determined as in Equation 4.3.

$$\varepsilon_{est} = \frac{\delta\theta}{\theta_i} = \frac{\theta - \theta_i}{\theta_i} \tag{4.3}$$

Since the θ_i is integral in computing accurate strain, a two part system was utilized. The Maryland Bipolar Forceps tool is equipped with electrically isolated electrocautery signals that are connected to the grasper jaws. A resistance sensor was connected to the electrocautery leads on the proximal end of the tool. Then the code can monitor for any change in the resistance. Any change above a threshold is used to indicate a first touch. As a secondary measure the force measurement is also thresholded relative to a baseline force. Any change in force above a threshold indicates a first touch event. For a given time series, which consists of one complete cycle of the jaws closing and opening, the first touch is segmented according to Equations 4.4-4.7. This requires first computing a threshold based on the minimum and maximum resistance values between the jaws (Equation 4.4).

$$\Omega_{thresh} = \frac{max(\Omega_{jaw}) + min(\Omega_{jaw})}{2} \tag{4.4}$$

Using this threshold the index set for the grasp data is taken as all resistance values greater than this threshold (Equation 4.5).

$$I_R = \Omega_{jaw} >= \Omega_{thresh} \tag{4.5}$$

From this index set the last index in I_R is taken as the index of the first touch (Equation 4.6).

$$I_{start} = max(I_R) \tag{4.6}$$

Given the start index, the ending index is taken as the point in time where the force reaches its maximum plus a small delay to allow the grasp to settle (Equation 4.7).

$$I_{end} = (F_{jaw} == max(F_{jaw})) + T_{settle} \tag{4.7}$$

Therefore the angle and force data while the tissue is in contact with tissue is taken as all time steps between I_{start} and I_{end}. This Smart Tool implementation was used to collect the cadaveric grasp data outlined in Section 5.2.

Chapter 5

Experimental Design

In order to examine the effectiveness of the proposed algorithms on classifying dynamic data, three primary data sets from unique applications are presented for evaluation. The first data set involves simulated dynamic data populated using Matlab and Simulink by utilizing various state space models. The second data set consists of dynamic surgical tool grasp data collected with a sensorized minimally invasive surgical grasper. The third data set consists of surgical motion data collected using the EDGE laparoscopic trainer in [15]. The following chapter outlines the experimental algorithm application for each data set.

5.1 Data Set 1: Simulated Dynamic Data

This first data set used to evaluate the proposed algorithms consists of simulated data populated in Matlab and Simulink. This experimental data will serve to investigate initial validity of the proposed algorithms. This data was populated using both a linear and non-linear model with a variety of noise factors and separable parameters to ensure robustness. In the linear simulated data the following state space model is used

$$\begin{pmatrix} \dot{x} \\ \ddot{x} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\alpha_{1,1} & -\alpha_{1,2} \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix} + U \tag{5.1}$$

Here $\alpha_{c,i}$ represents linear parameter i for class c. The input to this system is a ramp. In order to simulate distinct class systems, these parameters will be unique pairs for each class. To control the separation between classes, the linear parameters are set according to a scaling

factor $\kappa \geq 1$. In this way the α_2 parameters are set for class two as a function of α_1 (Eq. 5.2).

$$\alpha_{c,i} = \alpha_{1,i}\ \kappa \tag{5.2}$$

In this case $\kappa = 1$ results in identical systems, as κ increases the separation between data also increases. For the purpose of system testing, a subset of two κ values is utilized to create data. In order to simulate noisy data, several simulations (trajectories) will be run for the same class parameter set with slight parameter noise ξ added to each parameter (Fig. 5.1).

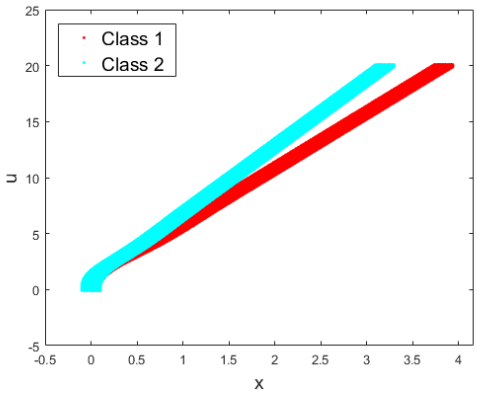


Figure 5.1: Linear model simulated data.

In order to simulate noise, for each simulation, random process noise (λ) is added to the output states of x, $\dot{x}$, and $\ddot{x}$ and random observation noise (ε) to the force input state u. This permits a more realistic simulation of real world data especially in systems such as the smart tool.

For the simulated linear data, the permutations outlined in Table 5.1 will be collected.

Class	C_1	C_2
Parameters	$(\alpha_{11}, \alpha_{12})$	$(\alpha_{21}, \alpha_{22})$
Separation	$\kappa = 1$	$\kappa = 1.1, 1.2$
Parameter Noise	1%	1%
Process Noise	5%	5%
Observation Noise	5%	5%
# Data Points	1000	1000
# Simulations	100	100

Table 5.1: Parameters, noise, and data points for linear model simulated data generation.

The candidate algorithms were trained using the linear simulated data as an input data set. The number of data points per simulation and the number of simulations was chosen in order to provide sufficient training samples. For training and testing, subsets of the total data set will be utilized for the input data set in order to observe any potential degradation of the discriminant ability.

Once each candidate algorithm was trained using the simulated data, the discriminant ability was evaluated using a leave-one-out validation scheme. For example 99 of the original simulations were used for training from each class, the remaining trajectory was individually run as an online sample and subsequently classified. The estimated classification from each candidate algorithm was compared with the known class. The number of correct and incorrect classifications were then recorded and used as a baseline for each algorithms potential discriminant ability.

In addition to the linear model, a variation of a non-linear tissue model [57] was also be used for a second set of simulated data. The non-linear model used is an exponential function:

$$\ddot{x} = \alpha_{11}\dot{x} + e^{-\alpha_{12}x} + U \tag{5.3}$$

Again here α_{ci} represents non-linear parameter i for class c. In order to simulate distinct class systems, these parameters will be unique sets for each class. To control the separation between classes, the model parameters are again set according to a scaling factor $\kappa \geq 1$ (Eq. 5.2). This nonlinear state space system results in a phase portrait similar to that of Fig. 5.2. In

order to simulate noisy data, several simulations will be run for the same class parameter set with slight parameter noise added to each parameter.

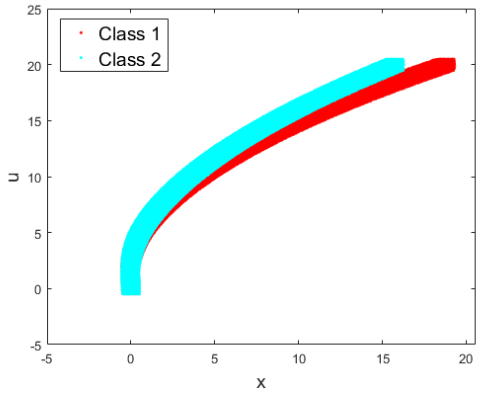


Figure 5.2: Non-linear model simulated data.

The non-linear model simulated data will be evaluated with the same number of classes, data samples, and noise values as the linear model (Table. 5.1). Again each candidate algorithm was evaluated using a leave-one-out validation using a single trajectory from the simulated data as the training sample.

For each test classification, the following data will be recorded

- The estimated classification
- The actual class label
- The time to convergence (as percentage of total trajectory)
- Whether classification converged
- The training set size
- The actual system parameters used

The expected outcome from this experiment will be that the first candidate algorithm (DLS) will be able to accurately discriminate between classes for a variety of parameter sets for the linear system but will have lower accuracy when discriminating data from the non-linear model.

However it is expected that the DPP and RELIEF-RBF algorithms will be capable of discriminating classes for both the linear and non-linear models with $> 95\%$ correct classifications. Additionally it is expected that as the distance between the parameter sets falls below the parameter noise threshold, neither discriminant method will be capable of repeated correct classification.

5.2 Data Set 2: Cadaveric Tissue Grasp Identification

Prior art has focused on the development of tools capable of sensing tissue properties in-vivo in order to address the negative impacts stemming from the lack of haptic feedback within minimally invasive surgery [10, 56]. Research has demonstrated that laparoscopic tools can be successfully modified to perform force sensing. Furthermore, using sensorized tools it has been shown that tissue identification can be performed in-vivo using a nonlinear tissue model [14]. However prior attempts have all suffered from the lack of a robust discriminant classification method for the non-linear, dynamic signals present in tissue. For this reason, tissue identification in-vivo is a key application for the dynamic discriminant analysis method.

A large data set containing human cadaveric tissue grasp data was collected using the Smart Tool hardware described in Chapter 4. This device was developed and constructed in the Medical Devices and Robotics lab and is named the Smart Tool. The Smart Tool is designed around an augmented da Vinci tool. The da Vinci tool cables are driven by a pair of DC servo motors. The housing of these servo motors are then mounted onto load cells. These load cells allow an estimate of the force being applied at the grasper jaws. The position of the grasper jaws is estimated by reading the encoder embedded within the servo motor. At each time step the recorded data vector includes angle, angle derivatives, and force estimates (Eq. 5.4). The data at each time step was stored along with the timestamp to a log file for offline analysis.

$$\chi_t = [\theta, \dot{\theta}, \ddot{\theta}, F] \tag{5.4}$$

The basis of the identification of tissue type stems from the non-linear tissue model derived by Y.C Fung [57]. This model was expanded upon by Yu et al. to include mass and damper terms [58]. The dynamics of tissue using this nonlinear model can be expressed in terms of position and force variables (Eq. 5.5).

$$u = m\frac{\partial^2 p}{\partial t^2} + d\frac{\partial p}{\partial t} + \alpha\left(e^{\beta\varepsilon} - 1\right) \tag{5.5}$$

Only the Discriminant Least Squares (DLS) algorithm will directly utilize this parametric model. It is important to understand that the non-linear terms present in this model will require linearization to be used in the DLS approach.

Experiment 2 will involve tissue classification using the smart tool grasp data on cadaveric tissue samples. For the cadaveric tissue data, tissue samples were collected over a 5 month span at the University of Minnesota. These organs were obtained from LifeSource via the Bequest Program at the University of Minnesota. Tissue samples were refrigerated following crossclamp. Both tissue types were stored in same solution: 0.9% Saline (Baxter International Inc. Deerfield, IL).

The Smart Tool was used to grasp two distinct tissue types: Liver and Pancreas. These tissues represent two distinct ranges of tissue parameters and will also allow direct comparison to prior work in this field.

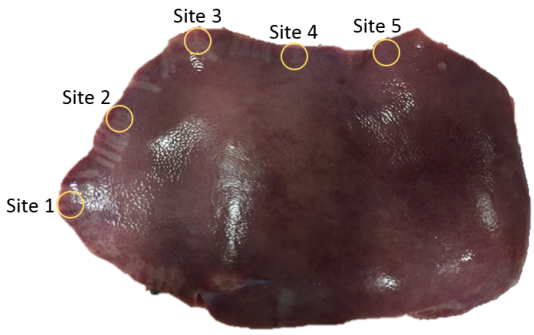


Figure 5.3: Tissue grasp site locations.

For this data set samples were collected from multiple tissue donors. Each donor provided both organs: Liver and Pancreas. Twenty grasps were collected at five different sites for each organ (Fig. 5.3). The five sites were collected to ascertain intra-patient variability of tissues and the five patients were collected to ascertain inter-patient variability of tissues. A detailed experimental protocol for data collection used is outlined:

1. Remove tissue from refrigeration and place it in a water bath (de-ionized water) preheated to 38 °C
2. Once the tissue has reached 38°C, remove it from the water bath and place it on the workbench for data collection
3. Using Smart Tool, grasp the tissue 20 times at a single location using an automated 0.5

Hz trapezoidal trajectory

4. Replace the tissue in the water bath and repeat steps 2-3 at five unique locations around the tissue
5. Repeat steps 1-4 for each tissue type

Tissue collection was completed for 5 unique tissue donors. Donors were only utilized if they provided both the necessary tissues: Pancreas and Liver. Donors were collected over a period of 5 months as they became available. For each donor the relevant information including Cause of Death (COD) and Time Since Death (TSD), the time between death and data collection, were recorded using a de-identified patient ID. Information from the five donors are included in Table 5.2.

Donor No.	**Gender**	**Age**	**Weight [kg]**	**COD**	**TSD [hr]**
ADG1102	M	41	95.4	ICH	170.0
ADG4496	M	24	85.1	HT	101.85
ADIT421	M	54	98.5	HT	124.3
ADLA459	M	51	65.3	Anoxia	89.7
ADLA222	F	44	59	CA	69.8

Table 5.2: Organ donor demographic data.

COD abbreviations used are Intracranial Hemorrhage (ICH), Cardiac Arrest (CA), and Head Trauma (HT). TSD was computed as the time between estimated time of death and time that tissue testing began. Tissue grasping with the Smart Tool required the use of a platform on which to place the tissue sample (Fig. 5.4). In this way the axis of the surgical grasper was aligned with the tissue sample.

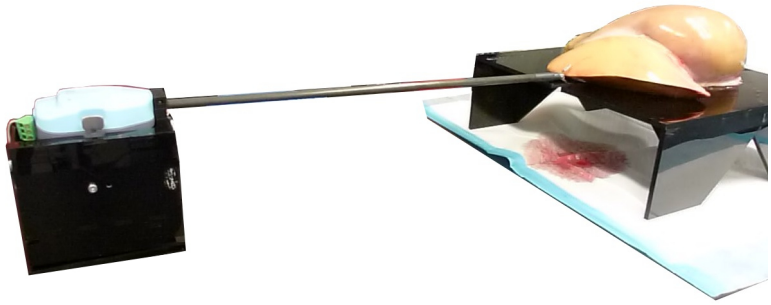


Figure 5.4: Smart Tool tissue data collection setup.

Per the the study outline, 5 donor subjects were obtained with both tissues of interest. For each tissue, 5 locations were grasped with 20 grasps at each location. This should have resulted in 1000 total grasps. For one of the locations on Donor ADLA459 Pancreas and one the locations on Donor ADLA222 Liver, the data files were corrupted which meant those locations' grasps are not included in the data set. Therefore the final grasp total was 960, with 480 pancreas grasps and 480 liver grasps. The average grasp duration was 291 *ms*. This duration is the time between first contact with the tissue and the time when the grasp ends. An overview of the resultant data set is given in Table 5.3.

Tissue	**Donors**	**Locations**	**Grasps**	**Grasp Duration [ms]**
Liver	5	24	480	298 (std = 37)
Pancreas	5	24	480	284 (std = 61)
Combined	5	48	960	291 (std = 50)

Table 5.3: Tissue data set overview.

A representative Stress-Strain plot for the two tissues is given in Figure 5.5. It is clear from this data that the tissue responses have a significant amount of overlapping regions.

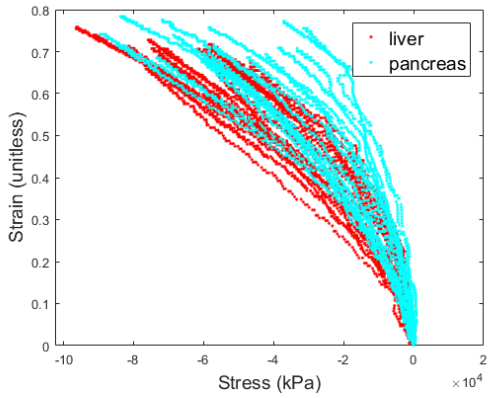


Figure 5.5: Stress-Strain response for liver and pancreas.

The candidate algorithms will be trained using the cadaveric grasp data as a training set. Additionally subsets of the total training data will be utilized for the training set in order to observe potential degradation of the discriminant ability. Once the candidate algorithm has been trained using a subset of the tissue grasp data, the discriminant ability will be evaluated using a leave-one-out validation scheme. The cross validation for this data set is two-fold. First, to examine the intra-patient variability for tissue classification, only the grasp data for both tissue types from within the same patient is used. Within a single patient a Leave-One-Location-Put (LOLO) process is employed to train the models and subsequently classify using the left out validation grasps. Second, to examine the inter-patient variability, a Leave-One-Donor-Out (LODO) scheme is employed. Here the model is trained using all grasps from all donors except one, then the classification is tested on the validation donor. The estimated tissue type classification from each candidate algorithm is compared with the known class to assess classification accuracy.

For each test classification, the following data will be recorded:

- The estimated tissue type
- The actual tissue type
- The time to convergence
- Whether classification converged
- The training set size

In the event of insufficient discriminating information, a tissue type classification of 'unknown' will also be possible. The 'unknown' classification will be an important piece of information for in-vivo applications since this could represent a tissue type which has not yet been observed. Numerically this would indicate that the classification did not converge to a sufficient threshold. In the force limiting grasper application, this classification could result in a marginal amount of force limiting with the assumption that unknown tissues should not be grasped too harshly.

The expected results from this experiment are that the first candidate algorithm (DLS) will not be able to correctly classify tissue type ($< 80\%$ classification) given the non-linear nature of the assumed physical tissue model and the lack of a clear separating hyper-plane. However it is expected that the second and third candidate algorithms (DPP, RELIEF-RBF) will correctly identify tissue types with $> 95\%$ correct classifications. Correct classifications will be achieved for both intra-patient and inter-patient cross validations, however it is likely that intra-patient will be more accurate given consistency among tissue. Additionally it is expected that these candidate algorithm will have a faster classification convergence rate than prior art ($< 300ms$) [14].

5.3 Data Set 3: Surgical Skill Evaluation

Prior art has shown that objective measures of skill, specifically motion metrics, have the potential to accurately predict surgical skill level [19, 38]. Surgical skill level has also been shown to correlate well with decreased complication rates [18]. However the primary gap in the prior art has been a 100% classification using motion analysis under leave-surgeon-out cross validation.

Figure 5.6: EDGE Laparoscopic platform.

A large data set concerning surgical tool motion and surgical skill level has been collected using the Electronic Data Generation for Evaluation (EDGE) (Simulab Corp. Seattle, WA) laparoscopic training platform (Fig. 5.6). Subject enrollment in this study was approved and registered under Western IRB 19125-A/B. The EDGE platform is used in hospitals and medical schools as a training module for laparoscopic skill development and records task video data and tool motion data from participants. The 3D position and orientation of the laparoscopic tools used in EDGE are tracked using a gimbal system. The EDGE platform also records instrument force application. At each time step the surgical tool motion data set from the EDGE system contains the data vector in Equation 5.6.

$$\chi_t^E = [X, Y, Z, F, \theta_{grasp}] \tag{5.6}$$

The EDGE surgical tool motion data set was previously collected by Kowalewski et al. and curated to include segmented trajectories as well as high confidence skill level classification labels [15, 109]. This data set was collected at three different sites and consisted of participants including surgical faculty, residents, and fellows. Participants in the study performed a subset of the Fundamentals of Laparoscopic Surgery (FLS) tasks; Peg Transfer, Pattern Cutting, and Intracorporeal Suturing. Each subject was asked to complete, at minimum, three iterations of the Peg Transfer task, two iterations of the Pattern Cutting task, and two iterations of the

Suturing task. The subject pool consisted of 98 total subjects from a variety of specialties including General Surgery, Urology, and Gynecology spanning three teaching hospitals. Two FLS-certified graders manually recorded task errors and task completion time was automatically recorded. Task errors and completion time were then used to compute an overall FLS score for each iteration.

From this data set the ground truth expert group (determined by a combination of caseload, FLS score, and p-OSATS score) was chosen for our 'Obvious Expert' category and the FLS Novice group (determined by the bottom 15^{th} percentile of FLS scores for trials in each task) for our 'Obvious Novice' category. Individuals with such low scores would fail FLS and thus not be allowed to operate. The complete data set contains 447 recorded trials across three tasks [109]. Only 91 of the original recorded trials were selected to represent the extremes of 'Obvious Experts' and 'Obvious Novices'. Each trial was performed by a different subject (Table 5.4).

Skill Level	**Peg Transfer**	**Pattern Cutting**	**Suturing**
'Obvious Novice'	29	25	13
'Obvious Expert'	6	10	8

Table 5.4: FLS trials by task and skill level.

Training data for the candidate algorithms will be based on segmented trajectories or defined movements (ie point A to point B). Each task was recorded with time synchronized video and tool motion data. This provided time-stamped Cartesian positions (x,y,z in cm) along with tool roll and grasper jaw angle (θ, degrees) at 30 Hz. This allowed subsequent computation of motion derivatives such as velocity and acceleration. In post processing, surgical tool motion was segmented into distinct motions within each task based on information from the tool grasper at the distal end (Fig. 5.7). A segment was considered to begin when the grasper was opened ($\theta > 3\,\text{deg}$) and the force within the grasper jaws fell below a threshold ($F_g < 4N$). The segment was then considered complete when the jaws were closed ($\theta < 3\,\text{deg}$) and the force applied within the grasper jaws rose above a threshold ($F_g > 4N$) for $200ms$ [109]. Each tool is segmented separately, allowing for overlapping segments between each instrument (hand). The mean number of segments per trial is given in Table 5.5.

FLS Task	Mean Segments Per Trial	std
Peg Transfer	27.9	2.2
Pattern Cutting	42.8	10.1
Intracorporeal Suturing	27.2	9.1

Table 5.5: Typical number of segments per trial.

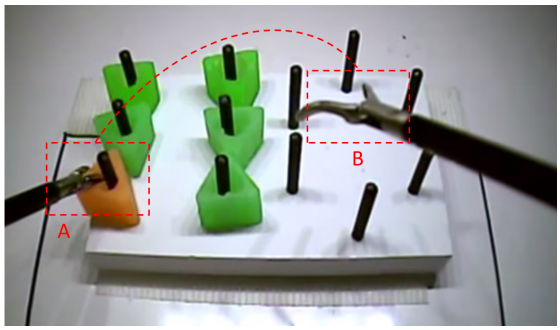


Figure 5.7: A sample left hand tool trajectory in the Peg Transfer task from point A to point B.

Functionally this segmentation scheme results in segments where a tool is moved in a trajectory toward an object, then the jaws are closed around the object to secure it, thus ending the segment. The segments focused only on tool motion where the surgeon is reaching toward an object (eg. before grasping or cutting), a motion which is prevalent in nearly all surgical tasks. The goal of this segmentation scheme was to be generalizable to all surgical tasks as compared to task specific surgical gestures. It is expected that some spurious false positives may occur within segmentation and assumed that these false segments occur equally across skill groups.

For skill level classification, the three tasks are utilized: Peg Transfer (PT), Suturing with Intracorporeal Knot (SIK), and the Pattern Cutting task (PC). These tasks are components of the FLS training and have been shown to correlate with skill level. The existing data set from the EDGE laparoscopic platform includes data for each class and task according to Table 5.4. The reason for the low number of 'Obvious Expert' surgeons available for use in experimental training is their limited availability. It was found that for each task, approximately 30 distinct trajectory segments exist. Therefore this will result in close to 2,800 unique trajectories for each skill level. Each algorithm will be trained and tested on each task independently, therefore some

tasks may have better classification than others.

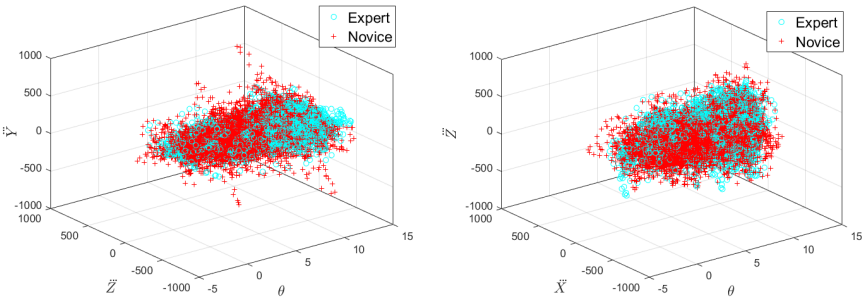
This experiment will involve classification of surgical skill level given tool motion and force data for laparoscopic surgical tools. Given the tool motion data sets as well as the labeled skill level, the candidate algorithms will attempt to classify surgical skill level of the subjects. Surgical skill level will be subjected to binary classification. The two tiers of skill will be 'Obvious Novice' vs. 'Obvious Expert'. The labeled skill values from the EDGE data set have been binned into these tiers for use as low noise class labels.

Using this data set, the candidate algorithms will be trained using a multi-dimensional state representation of the tool motion. Initially, the states to be used were refined in the state space representation. Using both RELIEFF and the RELIEF-RBF (Sec. 3.1.3), the states from the raw EDGE motion data which had the highest separability were investigated. The states used in this study are given in Eq. 5.7 where $\dot{x},\dot{y},\dot{z}$ terms represent derivatives w.r.t. time of the Cartesian location of the surgical tool tip. χ_t is sample at each time step in the data set. The Cartesian position of the surgical tool $[x,y,z]$ was excluded because of its relationship to the present surgical gesture. All resulting feature combinations were investigated.

$$\bar{\chi}_t = \left[\theta\ \dot{\theta}\ \dot{x}\ \dot{y}\ \dot{z}\ \ddot{x}\ \ddot{y}\ \ddot{z}\ \dddot{x}\ \dddot{y}\ \dddot{z}\ \|\dot{x},\dot{y},\dot{z}\|\ \|\ddot{x},\ddot{y},\ddot{z}\|\right] \tag{5.7}$$

.

The relevance of the raw motion states was examined for all states in Eq. 5.7. The three motion states with the highest relevance weights according to RELIEFF were found to be $[\theta,\dddot{z},\dddot{y}]$. The corresponding RELIEFF weights were $[2.3\times10^{-3},2.7\times10^{-3},3.0\times10^{-3}]$. A plot of these three states is given in Figure 5.8a.



(a) Top Three RELIEFF States (b) Top Three RELIEF-RBF States

Figure 5.8: Relevance weightings for raw motion states.

RELIEF-RBF gave slightly different states with high relevance. The motion states with the highest relevance weights according to RELIEF-RBF were found to be $[\theta, \dddot{x}, \dddot{z}]$. The corresponding RELIEF-RBF weight was 6.7×10^{-3} for this combination of states. A plot of these three states is shown in Figure 5.8b. The additional relevance weights for the other motion states are not included for the sake of brevity but were all similarly low.

Given the initial findings from RELIEFF and RELIEF-RBF the following states were chosen to use for the DPP and RELIEF-RBF classification algorithms: $\hat{\chi}_t = [\theta, \dddot{x}, \dddot{y}, \dddot{z}]$. However for the Intent Vector approach the appropriate data is still the segmented Cartesian tool position.

Once a candidate algorithm has been trained using data from each skill level class, the discriminant ability will be evaluated using a leave-one-user-out (LOUO) cross validation scheme. Non-training data from each skill level group will be analyzed individually and classified. The estimated skill level classification for each candidate algorithm will be compared with the known class (taken from labeled skill level).

For each test classification, the following data will be recorded:

- The estimated skill level
- The 'ground truth' skill level
- The time to convergence
- Whether classification converged
- The states used for representation

The use of the skill level labels from the EDGE data is assumed to be completely accurate since the data set has been evaluated by experts using a 'gold standard' of skill evaluation (OSATS). Since this data set has already been collected and requires no additional effort to perform classification tests with, the sample size will be assumed to be sufficient. Therefore no power analysis was performed to guarantee sample size. Instead the leave-one-out cross validation will be utilized to observe whether the sample sizes are sufficient.

The expected results from this experiment are that the second candidate algorithm (DPP) will have reasonably accurate classification rates ($> 85\%$). However given the potentially distinct features embedded in the dynamic motion data, it is expected that the third and fourth methods (RELIEF-RBF, Intent Vectors) will have a higher correct classification rates than prior art ($> 95\%$). Additionally, it is expected that the majority of the discriminating information will be embedded in the Cartesian state information while the grasper force and position data may be less beneficial.

5.4 Summary Experimental Design

This section has outlined the variety of data sets that will be utilized as sample applications for the dynamic discriminant algorithms proposed in Section 3. Given the uniqueness of each proposed algorithms, it is not feasible to use each of these algorithms for all data sets. For example the DLS approach requires a parametric model to train with, for the surgical tool motion data, there is no known parametric model mapping tool motion to skill level. Similarly, the Intent Vector approach is only applicable to the surgical tool motion data. An overview of the algorithms and the applied data sets is given in Table 5.6.

Algorithm	**Linear Simulated**	**NonLinear Simulated**	**Tissue Data**	**EDGE Data**
DLS	X	X	X	-
DPP	X	X	X	X
RELIEF-RBF	X	X	X	X
Intent Vector	-	-	-	X

Table 5.6: Algorithms and applicable data sets.

It is expected that the first candidate algorithm (DLS) will classify linear state space models with $> 90\%$ accuracy given prior training data. It is expected that sufficient training data sizes will be application specific, but large enough to encompass the noise threshold for a given system. It is also expected that the first candidate algorithm will be able to correctly classify $> 80\%$ the non-linear systems, as long the non-linear component is insignificant when compared to the effect of the linear term. However for systems with potentially highly non-linear effects (such as the tissue type classification) the first candidate will be expected to have a low instance of correct classification. The complete expected results, including the convergence rate as a percent of the total trajectory, are listed in Table 5.7.

Data Set	**Correct Classification**	**Convergence Rate [%]**
Linear Simulated	$> 95\%$	20
Non-Linear Simulated	80%	20
Tissue Data	80%	20
EDGE Data	-	-

Table 5.7: Candidate algorithm 1 (DLS) expected results.

The second candidate algorithm (DPP) is expected to handle all the test classification applications quite well. For the simulated data it is assumed that classification will be correct for both linear and non-linear systems $> 95\%$. For the tissue type classification it is expected that tissue type will be correctly identified in $> 90\%$ of tests. For surgical skill level it is expected that skill level will be classified correctly in $> 95\%$ of tests. A summary of the expected results, including the convergence rate as a percent of the total trajectory, is given in Table 5.8.

Data Set	**Correct Classification**	**Convergence Rate [%]**
Linear Simulated	$> 95\%$	15
Non-Linear Simulated	$> 95\%$	15
Tissue Data	$> 90\%$	25
EDGE Data	$< 80\%$	-

Table 5.8: Candidate algorithm 2, 3 (DPP, RELIEF-RBF) expected results.

Similarly, the third candidate algorithm (RELIEF-RBF) is expected to have sufficient classification rates for all sample applications. For the linear and non-linear simulated data a classification rate $> 95\%$ is expected. Similarly for skill level and tissue identification, a classification rate $> 95\%$ is expected. Additionally, given the lesser computational complexity of the third algorithm, it is expected to have faster computation times when compared with DPP.

Given its limited applicability, the Intent Vector approach is expected to produce high classification results for the surgical tool motion data set. Since this feature is specifically designed for surgical gestures a classification rate of $> 95\%$ is expected. However, this feature-based approach is not applicable to the other data sets.

Chapter 6

Results and Analysis

Using the algorithms outlined in Chapter 3 and the sample application data sets outlined in Chapter 5, the classification ability of the proposed algorithms was assessed. In all cases, except where noted, the classification was performed using a leave-one-out cross validation.

6.1 Results: Algorithm 1 (DLS)

As indicated previously, the Discriminant Least Squares algorithm was only tested on the linear and non-linear simulated data, and the tissue grasping data sets. This was due to the requirement of a parametric model for training the discriminant parameters.

6.1.1 Linear Simulated Data

For each cross-validation iteration for linear simulated data set, the discriminant least squares parameters Φ_c^* for each class were computed using only the training data subset according to Equation 3.23. For this simple system the linear model was assumed to take the form of Equation 6.1.

$$U = [x\ \dot{x}\ 1] \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} \tag{6.1}$$

Given this three term model, the augmented state matrix from Equation 3.25 takes the form

of Equation 6.2.

$$X^* = \begin{bmatrix} x^2 & x\dot{x} & x \\ x\dot{x} & \dot{x}^2 & \dot{x} \\ x & \dot{x} & 1 \end{bmatrix}_{\Sigma} \tag{6.2}$$

For this cross validation analysis, one simulation was left out from each class for training the parameters Φ_c^*, the accuracy was then tested according to Equation 3.29 using the left out simulations. This cross validation was repeated for both separabilities $\kappa = 1.1$ and $\kappa = 1.2$. Given the linear nature of this data set, the optimal λ value was found to be $\lambda = 0$.

Both the per time-step accuracy (wherein we attempt to classify at each point in a trajectory) and the per-trajectory accuracy are reported. The mean convergence time as a function of the total trajectory time is also reported. An overview of the classification accuracy for the linear simulated data can be found in Table 6.1.

Separation	**Time-Step Accuracy [%]**	**Trajectory Accuracy [%]**	**Convergence Rate [%]**
$\kappa = 1.1$	92.9	100	13
$\kappa = 1.2$	93.9	100	11

Table 6.1: DLS classification result for simulated linear data.

It is clear from this simple linear data set that the DLS algorithm works as expected when classifying linear data with no need for the λ term to add discriminating power. A plot of the effect of the λ term on accuracy is given in Figure 6.1. The accuracy decreases after $\lambda > 0.5$, which is taken to be the $\lambda_{critical}$ value.

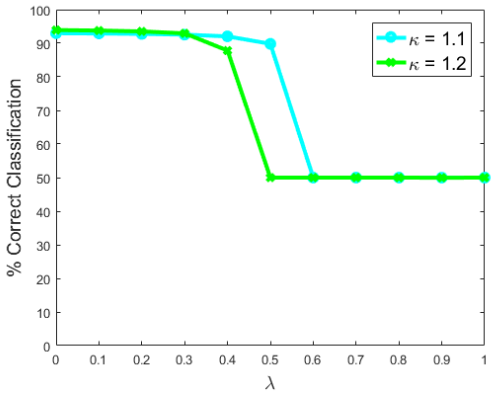


Figure 6.1: λ versus classification accuracy, linear data.

A feature of the DLS algorithm is the confidence value $\alpha(x_i)$ given at every data point (Eq. 3.31). An auxiliary measure of how well the DLS algorithm performs is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.2). For the $\kappa = 1.1$ linear data the mean confidence was 0.32 $(std = 0.27)$ and 0.69 $(std = 0.24)$ for incorrect and correct classifications, respectively. For the $\kappa = 1.2$ linear data the mean confidence was 0.38 $(std = 0.27)$ and 0.78 $(std = 0.21)$ for incorrect and correct classifications, respectively.

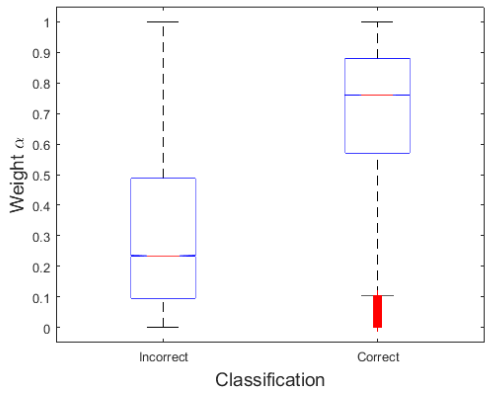


Figure 6.2: Distributions of α confidence, linear data ($\kappa = 1.1$)

6.1.2 Non-Linear Simulated Data

For each cross-validation iteration for the non-linear simulated data set, the discriminant least squares parameters Φ_c^* for each class were computed using only the training data subset according to Equation 3.23. For this non-linear system the required linearized model was assumed to take the form of Equation 6.3.

$$U = \phi_1 \ddot{x} + \phi_2 \dot{x} + \phi_3 x + \phi_4 x^2 \tag{6.3}$$

Given this three term model, the augmented state matrix from Equation 3.25 takes the form of Equation 6.4.

$$X^* = \begin{bmatrix} x^2 & x^3 & x\dot{x} & x\ddot{x} \\ x^3 & x^4 & x^2\dot{x} & x^2\ddot{x} \\ x\dot{x} & x^2\dot{x} & \dot{x}^2 & \dot{x}\ddot{x} \\ x\ddot{x} & x^2\ddot{x} & \dot{x}\ddot{x} & \ddot{x}^2 \end{bmatrix}_\Sigma \tag{6.4}$$

For this cross validation analysis, one simulation was left out from each class for training the parameters Φ_c^*, the accuracy was then tested according to Equation 3.29 using the left out simulations. This cross validation was repeated for both separabilities $\kappa = 1.1$ and $\kappa = 1.2$. Given the nonlinear nature of this data set, the optimal λ value was found to be $\lambda = 0.4$ for the $\kappa = 1.1$ data and $\lambda = 0.2$ for the $\kappa = 1.2$ data.

Both the per time-step accuracy (wherein we attempt to classify at each point in a trajectory) and the per-trajectory accuracy is reported. The mean convergence time as a percentage of the total trajectory time is also reported. An overview of the classification accuracy for the non-linear simulated data can be found in Table 6.2.

Separation	**Time-Step Accuracy [%]**	**Trajectory Accuracy [%]**	**Convergence Rate [%]**
$\kappa = 1.1$	80.9	100	39
$\kappa = 1.2$	86.6	100	19

Table 6.2: DLS Classification result for simulated non-linear data.

It is evident from this non-linear data set that the DLS algorithm works as expected when classifying non-linear data. However, added benefit is observed when utilizing the λ term to

add discriminating power. A plot of the effect of the λ term on accuracy is given in Figure 6.3. One observes that classification accuracy increases up to certain value of λ, and then declines.

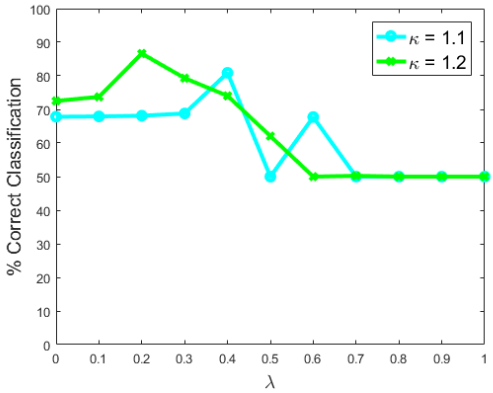


Figure 6.3: λ versus classification accuracy, non-linear data.

An auxiliary measure of how well the DLS algorithm performs is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.4). For the $\kappa = 1.1$ non-linear data the mean confidence was 0.37 $(std = 0.33)$ and 0.50 $(std = 0.32)$ for incorrect and correct classifications, respectively. For the $\kappa = 1.2$ non-linear data the mean confidence was 0.36 $(std = 0.28)$ and 0.61 $(std = 0.28)$ for incorrect and correct classifications, respectively.

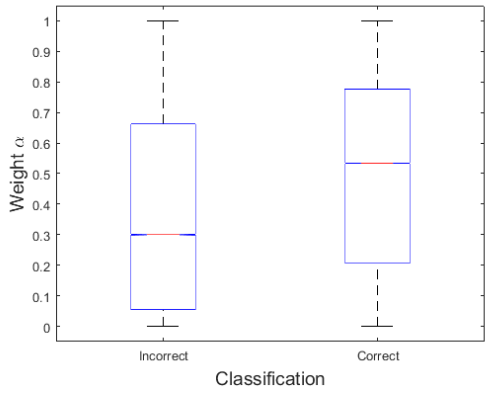


Figure 6.4: Distributions of α confidence, non-linear data ($\kappa = 1.1$)

6.1.3 Tissue Identification

For each cross-validation iteration for the tissue grasp data set, the discriminant least squares parameters Φ_c^* for each class were computed using only the training data subset according to Equation 3.23. For this complex non-linear system, the linearized model was assumed to follow Equation 6.5. Given this three term model, the augmented state matrix from Equation 3.25 takes the form of Equation 6.6 with inputs $U = F$.

$$F = \phi_1 \ddot{\theta} + \phi_2 \dot{\theta} + \phi_3 \theta + \phi_4 \theta^2 \tag{6.5}$$

$$X^* = \begin{bmatrix} \theta^2 & \theta^3 & \theta\dot{\theta} & \theta\ddot{\theta} \\ \theta^3 & \theta^4 & \theta^2\dot{\theta} & \theta^2\ddot{\theta} \\ \theta\dot{\theta} & \theta^2\dot{\theta} & \dot{\theta}^2 & \dot{\theta}\ddot{\theta} \\ \theta\ddot{\theta} & \theta^2\ddot{\theta} & \dot{\theta}\ddot{\theta} & \ddot{\theta}^2 \end{bmatrix}_\Sigma \tag{6.6}$$

For this cross validation analysis, first one patient's tissue data was left out from each class for training the parameters Φ_c^*, then the accuracy was tested according to Equation 3.29 using the left out patient. This cross validation is referred to as Leave-One-Donor Out (LODO) and is used to assess inter-patient variability.

The cross validation was repeated using a Leave-One-Location-Out (LOLO) scheme. In this approach intra-patient variability was assessed by storing all locations for a given donor except one as training data, the model was then trained using this training set. Classification was then performed using the left out location, all for one donor at a time. Given the nonlinear nature of this data set, the optimal λ value was empirically found to be $\lambda = 0.48$ for the LODO validation and $\lambda = 0.16$ for the LOLO validation.

Both the per time-step accuracy (wherein we attempt to classify at each point in a grasp) and the per-trajectory accuracy are reported. The mean convergence time as a percentage of the total trajectory time is also reported. An overview of the classification accuracy for the LODO and LOLO cross validations can be found in Tables 6.3 - 6.4. The per tissue classification accuracy is also included in each table.

Class	Time-Step Accuracy [%]	Trajectory Accuracy [%]	Convergence Rate [%]
Liver	87.4	100	16
Pancreas	59.6	80.0	25
Combined	73.8	90.0	21

Table 6.3: DLS classification result for tissue grasp data (LODO cross validation)

Class	Time-Step Accuracy [%]	Trajectory Accuracy [%]	Convergence Rate [%]
Liver	67.1	76.0	8
Pancreas	90.1	96.0	21
Combined	78.8	86.0	14

Table 6.4: DLS classification result for tissue grasp data (LOLO cross validation)

It is evident from this tissue data set that the DLS algorithm does not work as well for overlapping data sets. However, added benefit is still observed when utilizing the λ term to add discriminating power. A plot of the effect of the λ term on accuracy is given in Figure 6.5. One observes that classification accuracy increases up to approximately $\lambda = 0.8$, and then declines, representing the $\lambda_{critical}$ value.

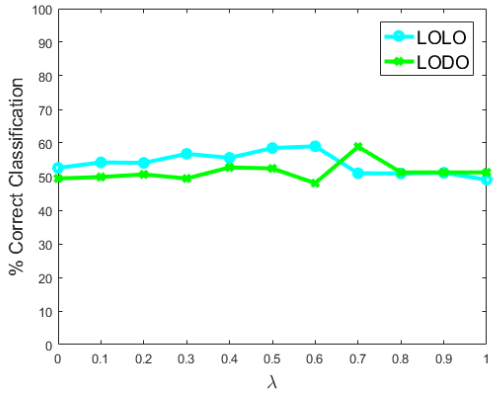


Figure 6.5: λ versus classification accuracy, tissue data.

An auxiliary measure of how well the DLS algorithm performs is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.6). For the LODO cross validation the mean confidence was 0.52 ($std = 0.37$) and 0.54 ($std = 0.25$) for incorrect and correct classifications, respectively. For the LOLO cross validation the mean confidence was 0.59 ($std = 0.25$) and 0.57 ($std = 0.28$) for incorrect and correct classifications, respectively.

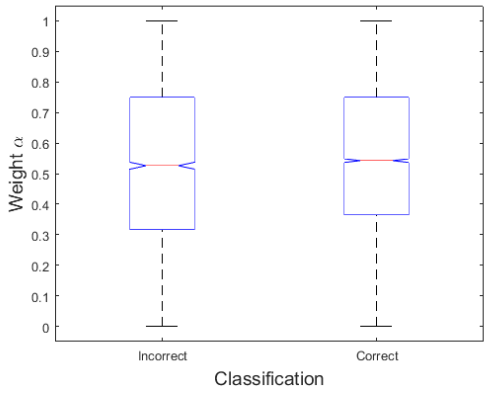


Figure 6.6: Distributions of α confidence, tissue data (LODO)

6.2 Results: Algorithm 2 (DPP)

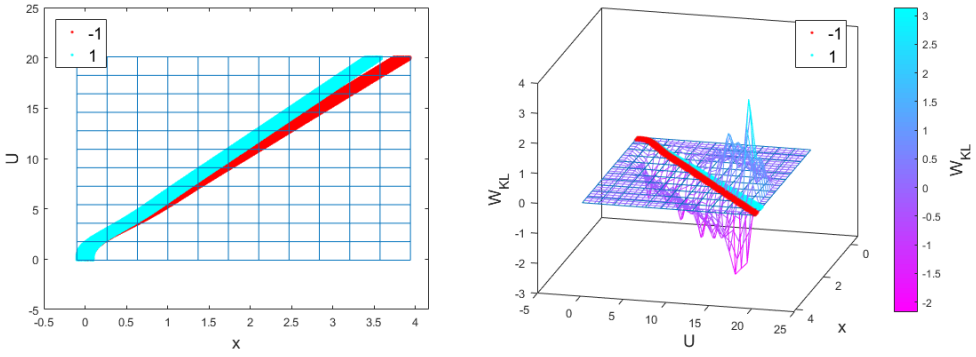
This section contains the results of the Discriminant Phase Portrait (DPP) algorithm. This approach was tested on the linear and non-linear simulated data, the tissue grasping data, and the surgical skill evaluation data sets.

6.2.1 Linear Simulated Data

For each cross-validation iteration for the linear simulated data set, the DPP grid indices were computed using only the training data subset according to Equation 3.36. For this simple linear model the states used for the phase portrait are given in Equation 6.7. These states result in a 3D grid of probability estimates, however for visualization purposes only a two dimensional grid is displayed along with the separability weights.

$$x_T(t) = [x(t), \dot{x}(t), U(t)] \tag{6.7}$$

Given this three dimensional phase portrait, the resultant grid for two states is given in Figure 6.7a. Similarly the W_{KL} weights from Equation 3.39 are given in Figure 6.7b.



(a) Resultant Grid

(b) W_{KL} Weights (3D)

Figure 6.7: Linear simulated data with DPP grid weighting ($\kappa = 1.1$)

For this cross validation analysis, one simulation was left out from each class for training the DPP Grid and separability measure $S_{j,k}$, classification was then tested according to Equation 3.45 using the left out simulations. This cross validation was repeated for both separabilities $\kappa = 1.1$ and $\kappa = 1.2$. For this data a grid element coarseness of $ns = 11$ was used. This value was heuristically determined in order to provide sufficient grid density given the observed separation between class data.

Both the per time-step accuracy, wherein we attempt to classify at each point in a trajectory (Eq. 3.44), and the per-trajectory accuracy, wherein the full resultant sum is used to classify (Eq. 3.45) are reported. The mean convergence time as a percentage of the total trajectory time is also reported. An overview of the classification accuracy for the linear simulated data can be found in Table 6.5.

Separation	**Time-Step Accuracy [%]**	**Trajectory Accuracy [%]**	**Convergence Rate [%]**
$\kappa = 1.1$	92.9	100	10
$\kappa = 1.2$	93.7	100	9

Table 6.5: DPP classification result for simulated linear data.

It is clear from this simple linear data set that the DPP algorithm works as expected when classifying linear data for entire trajectories. The effect of the grid coarseness parameter ns was not fully investigated for this data set, however sufficient classification power was found for values of $ns > 8$. The rate at which the M_{on} score (Eq. 3.44) varies can be used as a secondary measure of convergence time. A plot of this score as a function of trajectory percentage for each left out simulation is given in Figure 6.8.

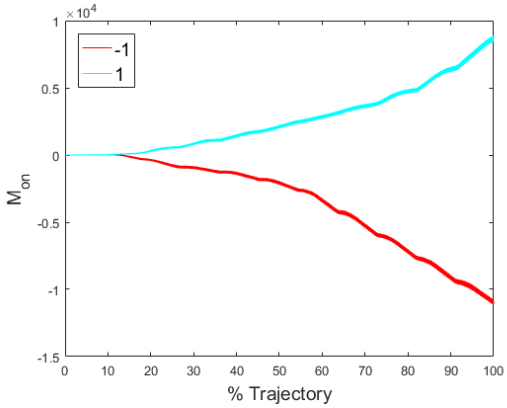


Figure 6.8: M_{on} score convergence versus time, linear data ($\kappa = 1.2$)

A feature of the DPP classification is the confidence value $S_{j,k}(x_i)$ given at every data point (Eq. 3.44). An auxiliary measure of how well the DPP algorithm performs is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.9). For the $\kappa = 1.1$ linear data the mean confidence was 0.04 $(std = 0.07)$ and 0.36 $(std = 0.26)$ for incorrect and correct classifications, respectively. For the $\kappa = 1.2$ linear data the mean confidence was 0.07 $(std = 0.15)$ and 1.12 $(std = 1.01)$ for incorrect and correct classifications, respectively.

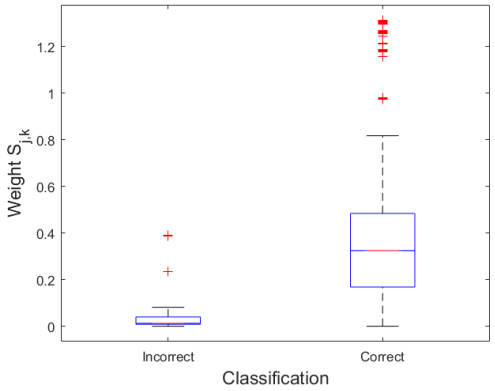


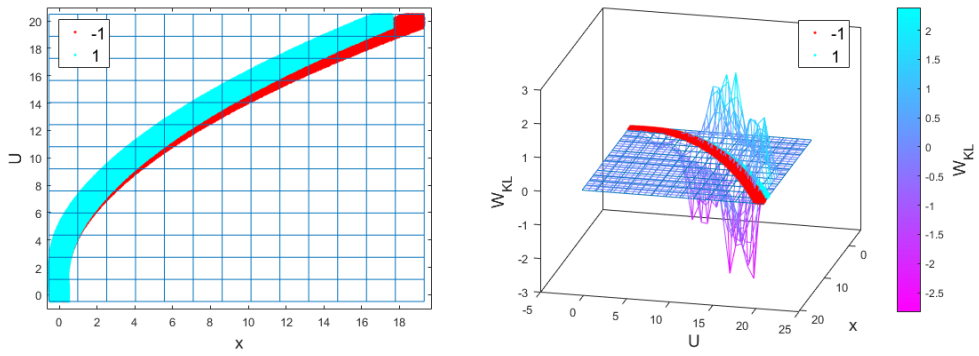
Figure 6.9: Distributions of $S_{j,k}$ confidence, linear data ($\kappa = 1.1$)

6.2.2 Non-Linear Simulated Data

For each cross-validation iteration for the non-linear simulated data set, the DPP grid indices were computed using only the training data subset according to Equation 3.36. For this non-linear system the states used for the phase portrait are given in Equation 6.8. These states result in a 3D grid of probability estimates, however for visualization purposes only a two dimensional grid is displayed along with the separability weights.

$$x_T(t) = [x(t), \dot{x}(t), U(t)] \tag{6.8}$$

Given this three dimensional phase portrait, the resultant grid for two states is given in Figure 6.10a. Similarly the W_{KL} weights from Equation 3.39 are given in Figure 6.10b.



(a) Resultant Grid

(b) W_{KL} Weights (3D)

Figure 6.10: Non-linear simulated data with DPP grid weighting ($\kappa = 1.1$)

For this cross validation analysis, one simulation is left out from each class for training the DPP Grid and separability measure $S_{j,k}$, classification is then tested according to Equation 3.45 using the left out simulations. This cross validation was repeated for both separabilities $\kappa = 1.1$ and $\kappa = 1.2$. For this data a grid element coarseness of $ns = 13$ was used. This value was heuristically determined in order to provide sufficient grid density given the observed separation between class data.

Both the per time-step accuracy, wherein we attempt to classify at each point in a trajectory (Eq. 3.44), and the per-trajectory accuracy, wherein the full resultant sum is used to classify (Eq. 3.45) are reported. The mean convergence time as a percentage of the total trajectory time is also reported. An overview of the classification accuracy for the non-linear simulated data can be found in Table 6.6.

Separation	**Time-Step Accuracy [%]**	**Trajectory Accuracy [%]**	**Convergence Rate [%]**
$\kappa = 1.1$	78.6	100	25
$\kappa = 1.2$	82.8	100	19

Table 6.6: DPP classification result for simulated non-linear data.

It is clear from this non-linear data set that the DPP algorithm works similarly well when classifying non-linear data for complete trajectories. The effect of the grid coarseness parameter *ns* was not fully investigated for this data set, however sufficient classification power was found

for values of $ns > 9$. The rate at which the M_{on} score (Eq. 3.44) varies can be used as a secondary measure of convergence time. A plot of this score as a function of trajectory percentage for each left out simulation is given in Figure 6.11.

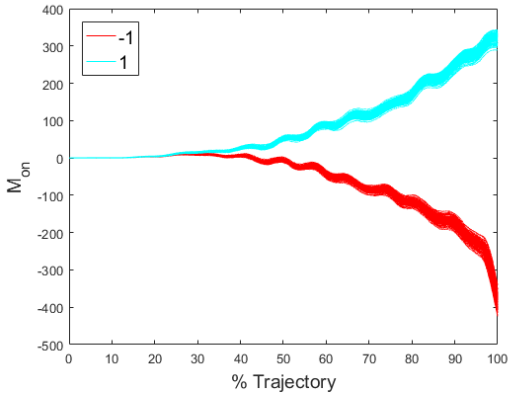


Figure 6.11: M_{on} score convergence versus time, non-linear data ($\kappa = 1.1$)

An auxiliary measure of how well the DPP algorithm performs is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.12). For the $\kappa = 1.1$ non-linear data the mean confidence was 0.02 $(std = 0.03)$ and 0.09 $(std = 0.08)$ for incorrect and correct classifications, respectively. For the $\kappa = 1.2$ nonlinear data the mean confidence was 0.03 $(std = 0.04)$ and 0.20 $(std = 0.19)$ for incorrect and correct classifications, respectively.

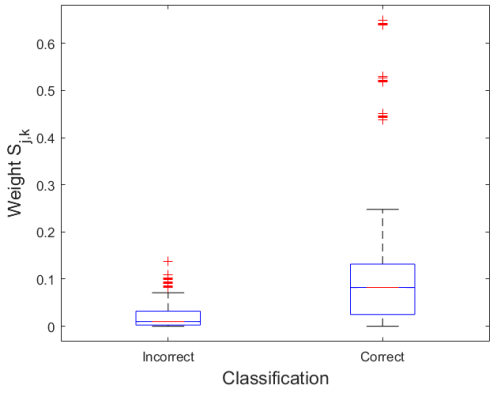


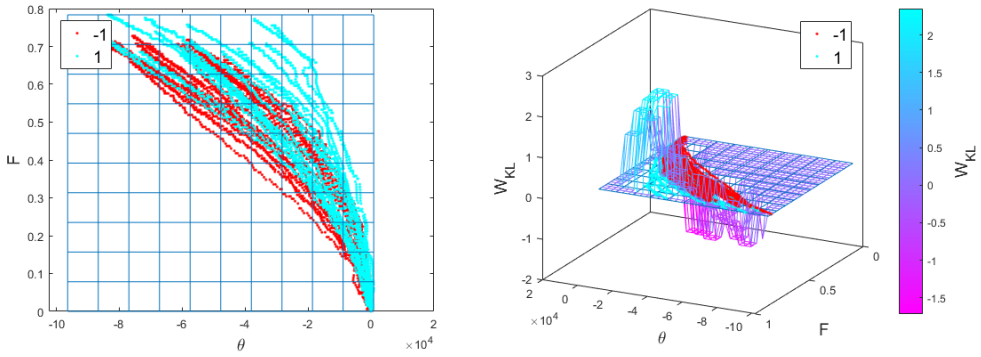
Figure 6.12: Distributions of $S_{j,k}$ confidence, non-linear data ($\kappa = 1.1$)

6.2.3 Tissue Identification

When using the tissue grasp data within the DPP algorithm, the labels for the two tissue types were mapped into ± 1 as follows $[Liver,\ Pancreas] = [-1, 1]$. This is required given the basis of the algorithm. For each cross-validation iteration for the tissue grasp data set, the DPP grid indices were computed using only the training data subset according to Equation 3.36. For this complex non-linear system the states used for the phase portrait are given in Equation 6.9. These states result in a 3D grid of probability estimates, however for visualization purposes only a two dimensional grid is displayed along with the separability weights.

$$x_T(t) = [\theta, \dot{\theta}, F] \tag{6.9}$$

Given this three dimensional phase portrait, the resultant grid for two states is given in Figure 6.13a. Similarly the W_{KL} weights from Equation 3.39 are given in Figure 6.13b.



(a) Resultant Grid

(b) W_{KL} Weights (3D)

Figure 6.13: Tissue grasp data with DPP grid weighting.

For this cross validation analysis, first one patient's tissue data was left out from each class for training the DPP Grid and separability measure $S_{j,k}$, the classification was then tested according to Equation 3.45 using the left out patient. This cross validation is referred to as Leave-One-Donor Out (LODO) and is used to assess inter-patient variability.

The cross validation was repeated using a Leave-One-Location-Out (LOLO) scheme. In this approach intra-patient variability was assessed by storing all locations for a given donor except one as training data, then the model was trained using this training set. Then classification was

performed using the left out location, all for one donor at a time. Given the complex nature of this data set, a grid element coarseness of $ns = 10$ was used. This value was heuristically determined in order to provide sufficient grid density given the observed separation between class data.

Both the per time-step accuracy (wherein we attempt to classify at each point in a grasp) and the per-trajectory accuracy are reported. The mean convergence time as a percentage of the total trajectory time is also reported. An overview of the classification accuracy for the LODO and LOLO cross validations can be found in Tables 6.7 - 6.8. The per tissue classification accuracy is also included in each table.

Class	**Time-Step Accuracy [%]**	**Trajectory Accuracy [%]**	**Convergence Rate [%]**
Liver	67.6	100	39
Pancreas	47.2	57.6	23
Combined	57.6	80.0	31

Table 6.7: DPP classification result for tissue grasp data (LODO cross validation)

Class	**Time-Step Accuracy [%]**	**Trajectory Accuracy [%]**	**Convergence Rate [%]**
Liver	63.3	80.0	27
Pancreas	56.5	60.0	22
Combined	59.9	70.0	25

Table 6.8: DPP classification result for tissue grasp data (LOLO cross validation)

It is evident from this tissue data set that the DPP algorithm, while providing marginal improvement in accuracy, does not perform as well on this tissue data set. This is likely attributable to the Pancreas data which does not appear to follow a consistent trajectory. The effect of the grid coarseness parameter *ns* was investigated for this data set and similar classification power was found for values of $9 < ns < 12$. The rate at which the M_{on} score (Eq. 3.44) varies can be used as a secondary measure of convergence time. A plot of this score over time for each left out grasp is given in Figure 6.14.

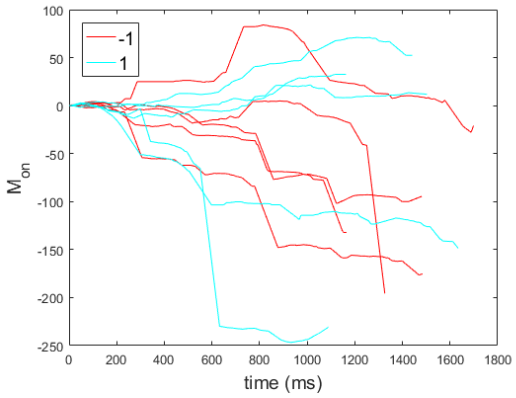


Figure 6.14: M_{on} score convergence versus time, tissue grasp data.

An auxiliary measure of how well the DPP algorithm performs is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.15). For the LODO cross validation the mean confidence was 0.13 $(std = 0.26)$ and 0.12 $(std = 0.24)$ for incorrect and correct classifications, respectively. For the LOLO cross validation the mean confidence was 0.14 $(std = 0.34)$ and 0.17 $(std = 0.32)$ for incorrect and correct classifications, respectively.

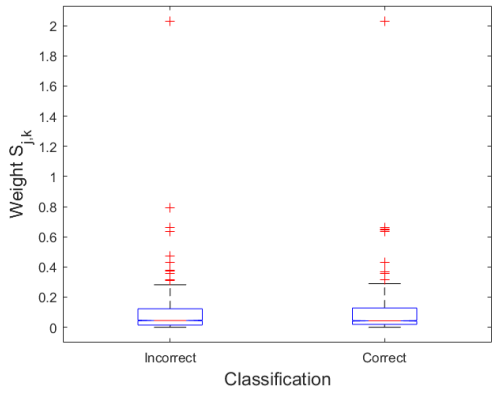
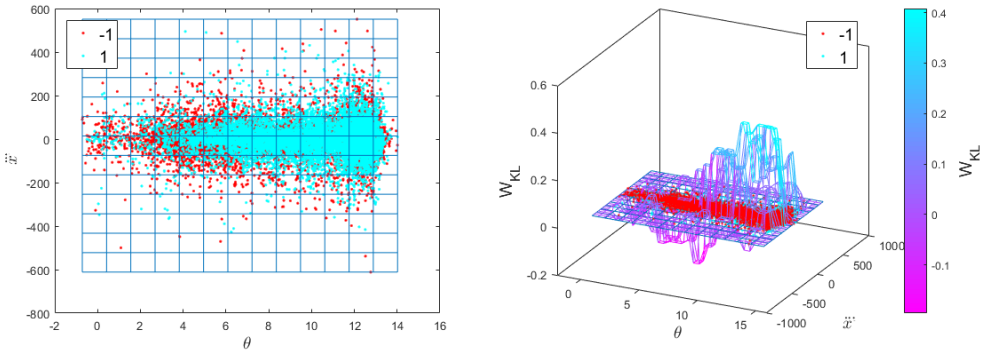


Figure 6.15: Distributions of $S_{j,k}$ confidence, tissue data (LODO)

6.2.4 Surgical Skill Classification

The DPP algorithm was trained using binary classifiers for skill from the raw surgical tool motion of the EDGE data set. As indicated in Section 5.3, the full data set consists of 13 states given in Equation 5.7. Using RELIEF-RBF feature weighting the states with the highest potential separability were reduced to include $\hat{\chi}_t = [\theta, \dddot{x}, \dddot{y}, \dddot{z}]$. A sample plot of these states and skill level is given in Figure 5.8b. These states result in a 4D grid of probability estimates, however for visualization purposes only a two dimensional grid is displayed along with the separability weights. When using the EDGE data within the DPP algorithm, the labels for the two skill level types was mapped into ± 1 as follows $[Novice,\ Expert] = [-1, 1]$. The resultant grid for two states is given in Figure 6.16a. Similarly the W_{KL} weights from Equation 3.39 are given in Figure 6.16b.



(a) Resultant Grid

(b) W_{KL} Weights (3D)

Figure 6.16: EDGE motion data with DPP grid weighting (Peg Transfer task)

For this cross validation analysis, one subject from each skill level group was left out for training the DPP Grid and separability measure $S_{j,k}$, the classification was then tested according to Equation 3.45 using the left out subject. This cross validation is referred to as Leave-One-User-Out-per-Group (LOUOpG). Classification was performed independently for each FLS task; Peg Transfer, Pattern Cutting, and Suturing. For this multidimensional data, a grid element coarseness of $ns = 13$ was used. Given the overwhelming similarity in this data a heuristically determined value for ns was used in order to provide sufficient grid density given the observed separation between class data.

Both the per time-step accuracy (wherein we attempt to classify at each point in a surgical motion segment) and the per-task accuracy, wherein the full resultant sum from all segments is used to classify (Eq. 3.45) are reported. Then mean convergence time as a percentage of the total task time is also reported. An overview of the classification accuracy for surgical skill level for the three FLS task cross validations can be found in Tables 6.9- 6.11. The per skill-level classification rate is also included in each table.

Class	**Time-Step Accuracy [%]**	**Task Accuracy [%]**	**Convergence Rate [%]**
Novice	84.5	86.2	6.9
Expert	35.9	41.4	12
Combined	72.5	63.8	9.5

Table 6.9: DPP classification result for EDGE surgical motion data (Peg Transfer task)

Class	**Time-Step Accuracy [%]**	**Task Accuracy [%]**	**Convergence Rate [%]**
Novice	73.9	60.0	13
Expert	9.7	4.0	20
Combined	62.1	32.0	17

Table 6.10: DPP classification result for EDGE surgical motion data (Pattern Cutting task)

Class	**Time-Step Accuracy [%]**	**Task Accuracy [%]**	**Convergence Rate [%]**
Novice	76.9	76.9	20
Expert	23.1	23.1	36
Combined	67.1	50.0	28

Table 6.11: DPP classification result for EDGE surgical motion data (Suturing task)

It is evident that the DPP algorithm does not perform well on the raw surgical motion data. This is likely due to the high degree of overlap between expert and novice motion, as evidenced by the extremely low RELIEFF and RELIEF-RBF weights given in Section 5.3. However given

the low separation between skill groups seen in Figure 6.16a, a macro classification of 73% per time step is acceptable.

An auxiliary measure of the DPP performance is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.17). For the Peg Transfer task the mean confidence was $3.9e^{-2}$ ($std = 3.7e^{-2}$) and $4.5e^{-2}$ ($std = 4.0e^{-2}$) for incorrect and correct classifications, respectively. For the Pattern Cutting task the mean confidence was $2.7e^{-2}$ ($std = 2.6e^{-2}$) and $2.3e^{-2}$ ($std = 2.2e^{-2}$) for incorrect and correct classifications, respectively. For the Suturing task the mean confidence was $6.1e^{-2}$ ($std = 8.7e^{-2}$) and $4.2e^{-2}$ ($std = 4.1e^{-2}$) for incorrect and correct classifications, respectively.

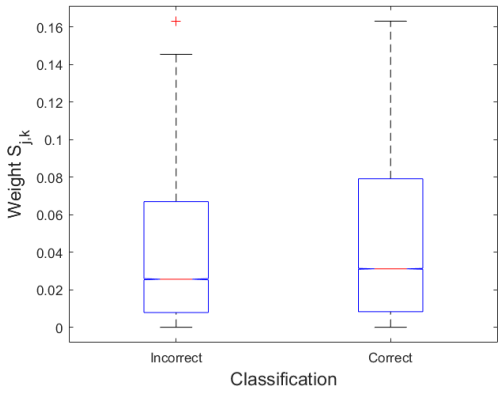


Figure 6.17: Distributions of $S_{j,k}$ confidence, EDGE data (Peg Transfer task)

6.3 Results: Algorithm 3 (RELIEF-RBF)

This section contains the results of the RELIEF-RBF algorithm. This approach was tested on the linear and non-linear simulated data, the tissue grasping data, and the surgical skill evaluation data sets.

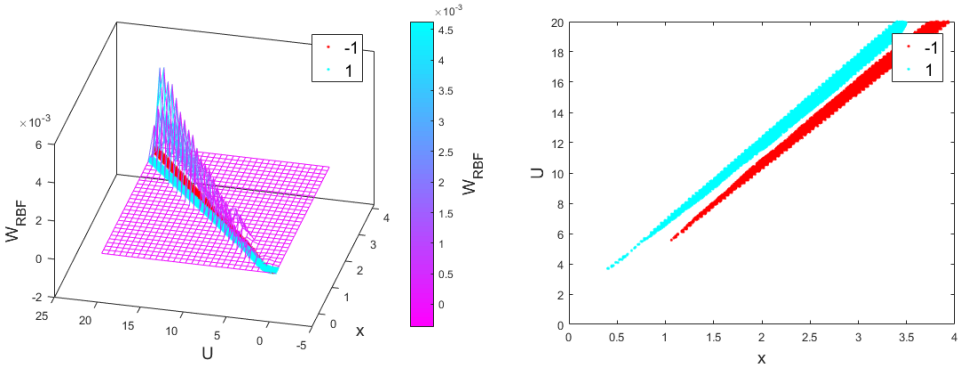
6.3.1 Linear Simulated Data

For each cross-validation iteration for the linear simulated data set, the RELIEF-RBF weights $W_{i,rbf}$ and subsampled data $\hat{X}_T$ were computed using only the training data subset according to Equation 3.65 - 3.66. For this simple linear model the states used for the phase portrait are

given in Equation 6.10. These states result in a 3D probability estimate, however for visualization purposes only a two dimensional phase portrait is displayed along with the RELIEF-RBF weights.

$$x_T(t) = [x(t), \dot{x}(t), U(t)] \tag{6.10}$$

Given this three dimensional phase portrait, the RELIEF-RBF weights ($W_{i,rbf}$) for two states is given in Figure 6.18a. Similarly the subsampled training data $\hat{X}_T$ from Equation 3.66 are shown in Figure 6.18b. A sample plot of the L_{on} value (Eq. 3.72) for the linear data is also given in Figure 6.19.



(a) W_T RELIEF-RBF Weights (b) $\hat{X}_T$ Subsampled Data

Figure 6.18: Linear simulated data with RELIEF-RBF subsampling ($\kappa = 1.1$)

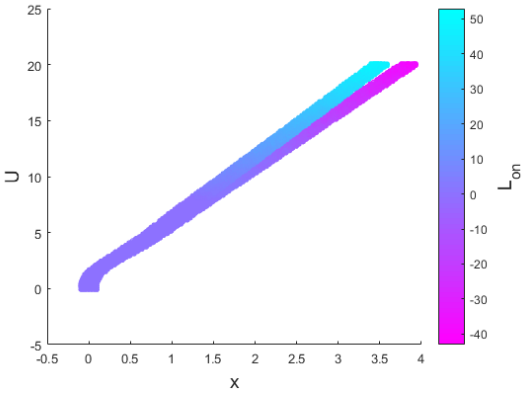


Figure 6.19: L_{on} RELIEF-RBF score, linear simulated data.

For this cross validation analysis, one simulation was left out from each class for training the RELIEF-RBF subsampling and GPR model *LK* (Eq. 3.67), classification was then tested according to Equation 3.73 using the left out simulations. This cross validation was repeated for both separabilities $\kappa = 1.1$ and $\kappa = 1.2$. For this data a subset ratio of $r_W = 0.3$ was used.

Both the per time-step accuracy, wherein we attempt to classify at each point in a trajectory (Eq. 3.71), and the per-trajectory accuracy, wherein the full resultant sum is used to classify (Eq. 3.73) are reported. The mean convergence time as a percentage of the total trajectory time is also reported. An overview of the classification accuracy for the linear simulated data can be found in Table 6.12.

Separation	**Time-Step Accuracy [%]**	**Trajectory Accuracy [%]**	**Convergence Rate [%]**
$\kappa = 1.1$	71.8	100	28
$\kappa = 1.2$	75.2	100	36

Table 6.12: RELIEF-RBF classification result for simulated linear data.

It is clear from this simple linear data set that the RELIEF-RBF algorithm works as expected when classifying linear data for entire trajectories. The subset ratio $r_W = 0.3$ primarily affects the degree of over-fitting in the data and the computational complexity. While the per-time step classification is low, these classifications are made with low confidence (Λ_{on}) at the beginning of the trajectory where separability is low. A plot of this confidence score as a function of trajectory percentage for each left out simulation is given in Figure 6.20.

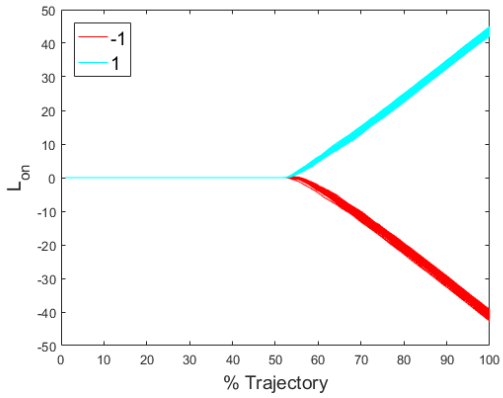


Figure 6.20: L_{on} score convergence versus trajectory completion, linear data ($\kappa = 1.2$)

An auxiliary measure of how well the RELIEF-RBF algorithm performs is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.21). For the $\kappa = 1.1$ linear data the mean confidence was 0.06 $(std = 0.15)$ and 0.59 $(std = 0.41)$ for incorrect and correct classifications, respectively. For the $\kappa = 1.2$ linear data the mean confidence was $7.3e^{-5}$ $(std = 1.3e^{-3})$ and 0.65 $(std = 0.42)$ for incorrect and correct classifications, respectively.

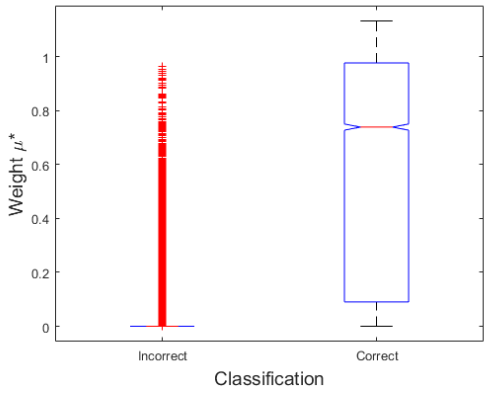


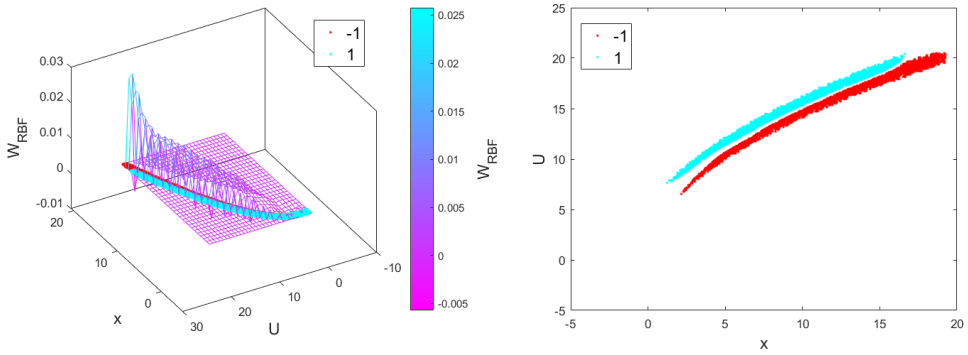
Figure 6.21: Distributions of $\mu*$ confidence, linear data ($\kappa = 1.1$)

6.3.2 Non-Linear Simulated Data

For each cross-validation iteration for the non-linear simulated data set, the RELIEF-RBF weights $W_{i,rbf}$ and subsampled data $\hat{X}_T$ were computed using only the training data subset according to Equation 3.65 - 3.66. For this non-linear model the states used for the phase portrait are given in Equation 6.11. These states result in a 3D probability estimate, however for visualization purposes only a two dimensional phase portrait is displayed along with the RELIEF-RBF weights.

$$x_T(t) = [x(t), \dot{x}(t), U(t)] \tag{6.11}$$

Given this three dimensional phase portrait, the RELIEF-RBF weights ($W_{i,rbf}$) for two states is given in Figure 6.22a. Similarly the subsampled training data $\hat{X}_T$ from Equation 3.66 are shown in Figure 6.22b. A sample plot of the L_{on} value (Eq. 3.72) for the non-linear data is also given in Figure 6.23.



(a) W_T RELIEF-RBF Weights (b) $\hat{X}_T$ Subsampled Data

Figure 6.22: Non-linear simulated data with RELIEF-RBF subsampling ($\kappa = 1.1$)

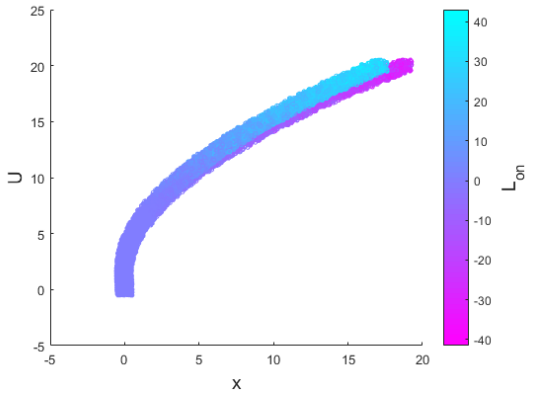


Figure 6.23: L_{on} RELIEF-RBF score, non-linear simulated data.

For this cross validation analysis, one simulation is left out from each class for training the RELIEF-RBF subsampling and GPR model *LK* (Eq. 3.67), classification is then tested according to Equation 3.73 using the left out simulations. This cross validation was repeated for both separabilities $\kappa = 1.1$ and $\kappa = 1.2$. For this data a subset ratio of $r_W = 0.3$ was used.

Both the per time-step accuracy, wherein we attempt to classify at each point in a trajectory (Eq. 3.71), and the per-trajectory accuracy, wherein the full resultant sum is used to classify (Eq. 3.73) are reported. Then mean convergence time as a percentage of the total trajectory time is also reported. An overview of the classification accuracy for the non-linear simulated data can be found in Table 6.13.

Separation	**Time-Step Accuracy [%]**	**Trajectory Accuracy [%]**	**Convergence Rate [%]**
$\kappa = 1.1$	77.4	100	36
$\kappa = 1.2$	74.8	100	33

Table 6.13: RELIEF-RBF classification result for simulated non-linear data.

It is clear from this non-linear data set that the RELIEF-RBF algorithm works equally as well when classifying non-linear data for entire trajectories. The subset ratio $r_W = 0.3$ primarily affects the degree of overfitting in the data and the computational complexity. While the per-time step classification is low, these classifications are made with low confidence (L_{on}) at the beginning of the trajectory where separability is low. A plot of this confidence score as a

function of trajectory percentage for each left out simulation is given in Figure 6.24.

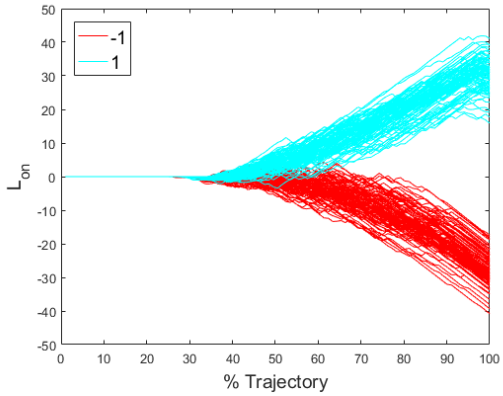


Figure 6.24: L_{on} score convergence versus trajectory completion, non-linear data ($\kappa = 1.1$)

An auxiliary measure of how well the RELIEF-RBF algorithm performs is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.25). For the $\kappa = 1.1$ non-linear data the mean confidence was 0.09 $(std = 0.24)$ and 0.54 $(std = 0.41)$ for incorrect and correct classifications, respectively. For the $\kappa = 1.2$ non-linear data the mean confidence was $2.4e^{-3}$ $(std = 0.03)$ and 0.48 $(std = 0.43)$ for incorrect and correct classifications, respectively.

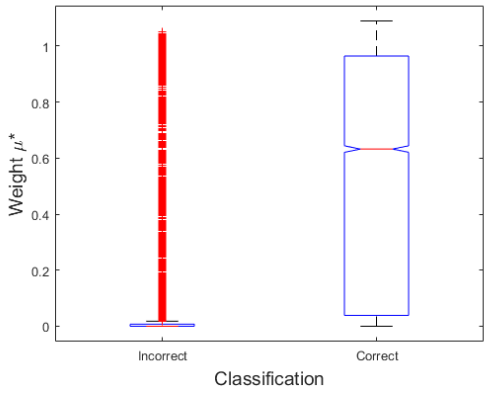


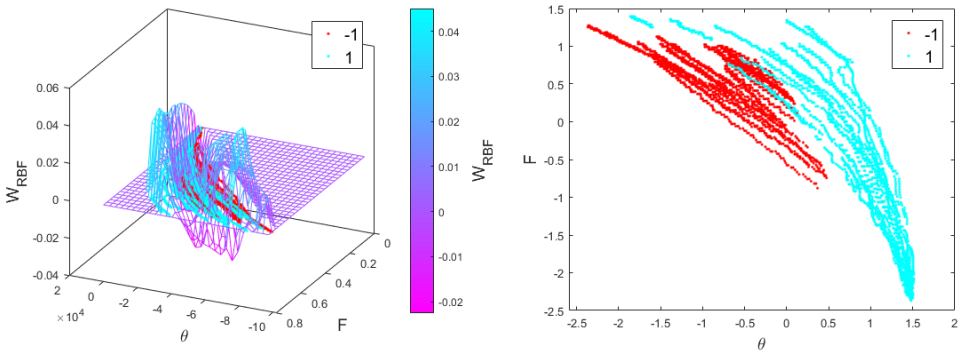
Figure 6.25: Distributions of $\mu*$ confidence, non-linear data ($\kappa = 1.1$)

6.3.3 Tissue Identification

When using the tissue grasp data within the RELIEF-RBF algorithm, the labels for the two tissue types were mapped into ± 1 as follows $[Liver,\ Pancreas] = [-1, 1]$. This is required given the basis of the algorithm. For each cross-validation iteration for the tissue grasp data set, the RELIEF-RBF weights $W_{i,rbf}$ and subsampled data $\hat{X}_T$ were computed using only the training data subset according to Equation 3.65 - 3.66. For this complex non-linear system the states used for the phase portrait are given in Equation 6.12. These states result in a 3D probability estimate, however for visualization purposes only a two dimensional phase portrait is displayed along with the RELIEF-RBF weights.

$$x_T(t) = [\theta, \dot{\theta}, F] \tag{6.12}$$

Given this three dimensional phase portrait, the RELIEF-RBF weights ($W_{i,rbf}$) for two states is given in Figure 6.26a. Similarly the subsampled training data $\hat{X}_T$ from Equation 3.66 are shown in Figure 6.26b. A sample plot of the L_{on} value (Eq. 3.72) for the non-linear data is also given in Figure 6.27.



(a) W_T RELIEF-RBF Weights (b) $\hat{X}_T$ Subsampled Data

Figure 6.26: Tissue grasp data with RELIEF-RBF subsampling.

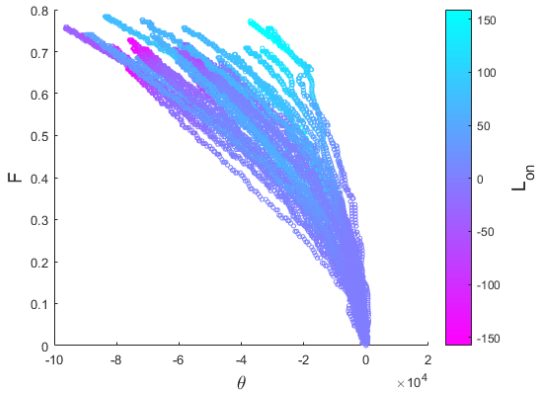


Figure 6.27: L_{on} RELIEF-RBF score, tissue grasp data.

For this cross validation analysis, first one patient's tissue data was left out from each class for training the RELIEF-RBF subsampling and GPR model *LK* (Eq. 3.67), classification was then tested according to Equation 3.73 using the left out patient. This cross validation is referred to as Leave-One-Donor Out (LODO) and is used to assess inter-patient variability. For this data we used a subset ratio of $r_W = 0.6$.

The cross validation was repeated using a Leave-One-Location-Out (LOLO) scheme. In this approach intra-patient variability was assessed by storing all locations for a given donor except one as training data, the model was then trained using this training set. Classification was then performed using the left out location, all for one donor at a time.

Both the per time-step accuracy, wherein we attempt to classify at each point in a grasp (Eq. 3.71), and the per-trajectory accuracy, wherein the full resultant sum is used to classify (Eq. 3.73) are reported. The mean convergence time as a percentage of the total trajectory time is also reported. An overview of the classification accuracy for the LODO and LOLO cross validations can be found in Tables 6.14 - 6.15. The per tissue classification accuracy is also included in each table.

Class	Time-Step Accuracy [%]	Trajectory Accuracy [%]	Convergence Rate [%]
Liver	69.6	100	30
Pancreas	59.5	60.0	15
Combined	64.6	80.0	22

Table 6.14: RELIEF-RBF classification result for tissue grasp data (LODO cross validation)

Class	Time-Step Accuracy [%]	Trajectory Accuracy [%]	Convergence Rate [%]
Liver	57.8	80.0	36
Pancreas	68.4	72.0	23
Combined	62.9	76.0	29

Table 6.15: RELIEF-RBF classification result for tissue grasp data (LOLO cross validation)

It is evident from this tissue data set that the RELIEF-RBF algorithm provides only marginal improvement in classification accuracy for the tissue data set. This is again likely attributable to the significant overlap in this data. The subset ratio $r_W = 0.6$ was found to affect the classification accuracy, accuracy decreased for $r_W < 0.6$ but stayed constant for $r_W \geq 0.6$. The rate at which the L_{on} score (Eq. 3.72) varies can be used as a secondary measure of convergence time. A plot of this confidence score as a function of time for each left out grasp is given in Figure 6.28.

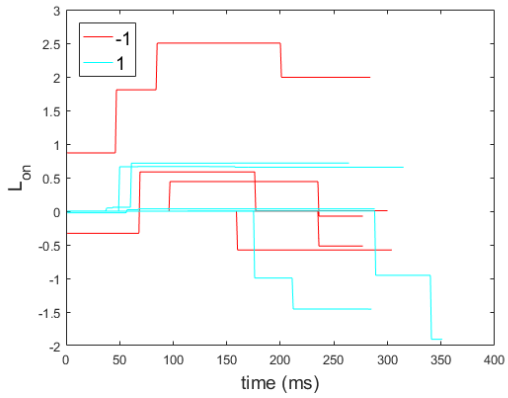


Figure 6.28: L_{on} score convergence versus time, tissue grasp data.

An auxiliary measure of how well the RELIEF-RBF algorithm performs is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.29). For the LODO cross validation the mean confidence was $1.2e^{-2}$ $(std = 9.6e^{-2})$ and $1.9e^{-2}$ $(std = 1.2)$ for incorrect and correct classifications, respectively. For the LOLO cross validation the mean confidence was $4.3e^{-3}$ $(std = 5.6e^{-2})$ and $5.0e^{-3}$ $(std = 6.1e^{-2})$ for incorrect and correct classifications, respectively.

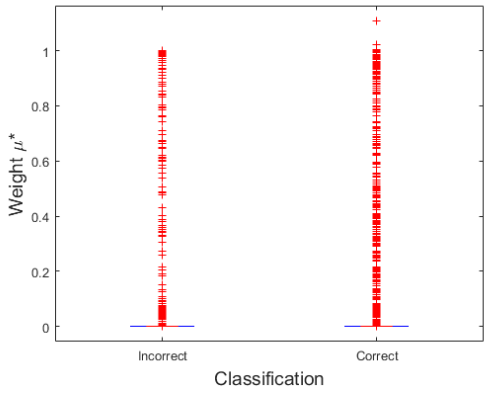
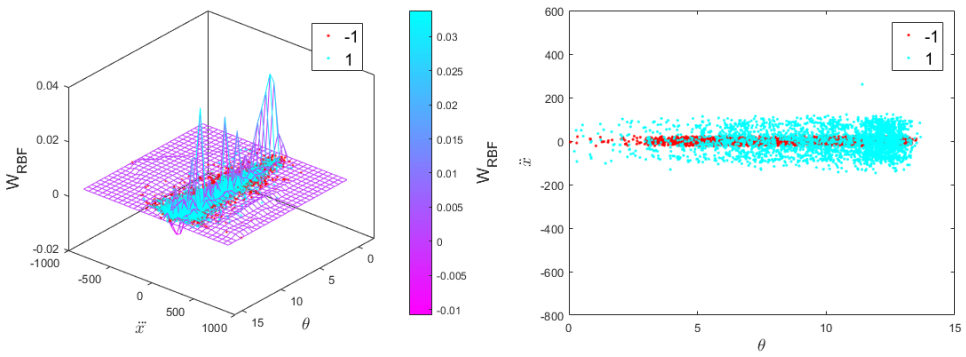


Figure 6.29: Distributions of $\mu*$ confidence, tissue data (LODO)

6.3.4 Surgical Skill Classification

The RELIEF-RBF algorithm was trained using binary classifiers for skill from the raw surgical tool motion of the EDGE data set. As indicated in Section 5.3, the full data set consists of 13 states given in Equation 5.7. Using RELIEF-RBF feature weighting, the states with the highest potential separability were reduced to include $\hat{\chi}_t = [\theta, \dddot{x}, \dddot{y}, \dddot{z}]$. A sample plot of these states and skill level given in Figure 5.8b. These states result in a 4D probability estimate, however for visualization purposes only a two dimensional grid is displayed along with the separability weights. When using the EDGE data within the RELIEF-RBF algorithm, the labels for the two skill level types was mapped into ± 1 as follows $[Novice,\ Expert] = [-1, 1]$. the RELIEF-RBF weights ($W_{i,rbf}$) for two states is given in Figure 6.30a. Similarly the subsampled training data $\hat{X}_T$ from Equation 3.66 are shown in Figure 6.30b. A sample plot of the L_{on} value (Eq. 3.72) for the tool motion data is also given in Figure 6.31.



(a) W_T RELIEF-RBF Weights (b) $\hat{X}_T$ Subsampled Data

Figure 6.30: EDGE motion data with RELIEF-RBF subsampling (Peg Transfer task)

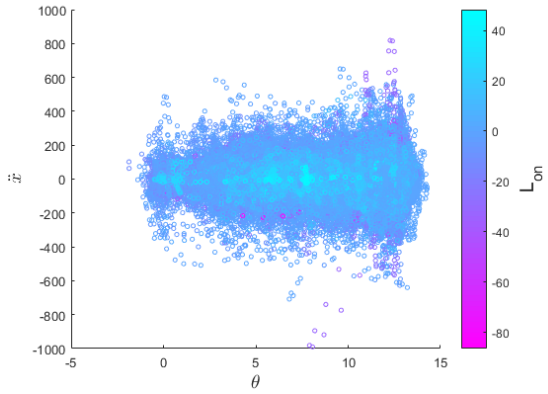


Figure 6.31: L_{on} RELIEF-RBF score, EDGE motion data.

For this cross validation analysis, one subject was left out from each skill level group for training the RELIEF-RBF subsampling and GPR model *LK* (Eq. 3.67), classification was then tested according to Equation 3.73 using the left out subjects. This cross validation is referred to as Leave-One-User-Out-per-Group (LOUOpG). Classification was performed independently for each FLS task; Peg Transfer, Pattern Cutting, and Suturing. For this data a subset ratio of $r_W = 0.5$ was used.

Both the per time-step accuracy, wherein we attempt to classify at each point in a surgical motion segment (Eq. 3.71), and the per-task accuracy, wherein the full resultant sum from all segments is used to classify (Eq. 3.73) are reported. The mean convergence time as a percentage of the total task time is also reported. An overview of the classification accuracy for surgical skill level for the three FLS task cross validations can be found in Tables 6.16- 6.18. The per skill-level classification rate is also included in each table.

Class	**Time-Step Accuracy [%]**	**Task Accuracy [%]**	**Convergence Rate [%]**
Novice	99.4	100.0	1.0
Expert	3.4	0.0	9.5
Combined	75.8	50.0	5.2

Table 6.16: RELIEF-RBF classification result for EDGE surgical motion data (Peg Transfer task)

Class	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
Novice	98.2	100.0	3.5
Expert	4.51	0.0	7.3
Combined	80.1	50.0	5.4

Table 6.17: RELIEF-RBF classification result for EDGE surgical motion data (Pattern Cutting task)

Class	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
Novice	90.2	92.2	2.7
Expert	6.5	4.0	8.9
Combined	79.6	46.4	5.8

Table 6.18: RELIEF-RBF classification result for EDGE surgical motion data (Suturing task)

It is evident that the RELIEF-RBF algorithm results in similar performance to DPP on the raw surgical motion data. This performance is likely attributable to the high degree of overlap between expert and novice motion, as evidenced by the extremely low RELIEFF and RELIEF-RBF feature weights given in Section 5.3. However given the low separation between skill groups seen in Figure 6.30b, a macro classification accuracy of 80% per time step is reasonable.

An auxiliary measure of the RELIEF-RBF performance is the distribution of confidence values when correct and incorrect classifications were made (Fig. 6.32). For the Peg Transfer task the mean confidence was $2.6e^{-3}$ ($std = 2.9e^{-3}$) and $5.4e^{-3}$ ($std = 4.5e^{-2}$) for incorrect and correct classifications, respectively. For the Pattern Cutting task the mean confidence was $4.0e^{-3}$ ($std = 3.6e^{-2}$) and $5.7e^{-3}$ ($std = 4.5e^{-2}$) for incorrect and correct classifications, respectively. For the Suturing task the mean confidence was $5.9e^{-3}$ ($std = 5.1e^{-2}$) and $2.6e^{-3}$ ($std = 3.1e^{-2}$) for incorrect and correct classifications, respectively.

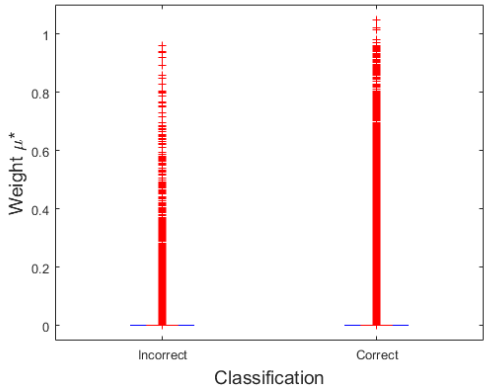


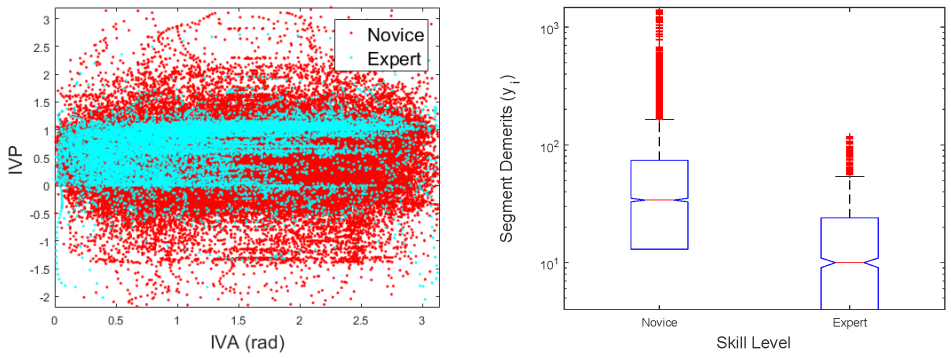
Figure 6.32: Distributions of $\mu*$ confidence, EDGE data (Peg Transfer task)

6.4 Results: Algorithm 4 (Intent Vectors)

This section contains the results of the Intent Vector feature and classification algorithm. As previously indicated, this approach was tested only on the surgical skill evaluation data set since the feature was derived specifically from surgical motion.

6.4.1 Surgical Skill Classification

The Intent Vector framework was trained using binary classifiers for skill from the EDGE data set. A sample plot of the Intent Vectors space is given in Figure 6.33a. This data indicates clear differences between Novices and Experts. Novices spend far more time outside the 0-1 range of the IVP, meaning they often backtrack and overshoot the starting and ending points. Additionally, Experts spend a lot of time with low IVA values meaning they generally head in the correct direction. However, Experts also have varied IVA values around the endpoint of segments ($IVP = 1$), meaning that near the endpoint, experts make fine adjustments to their approach.

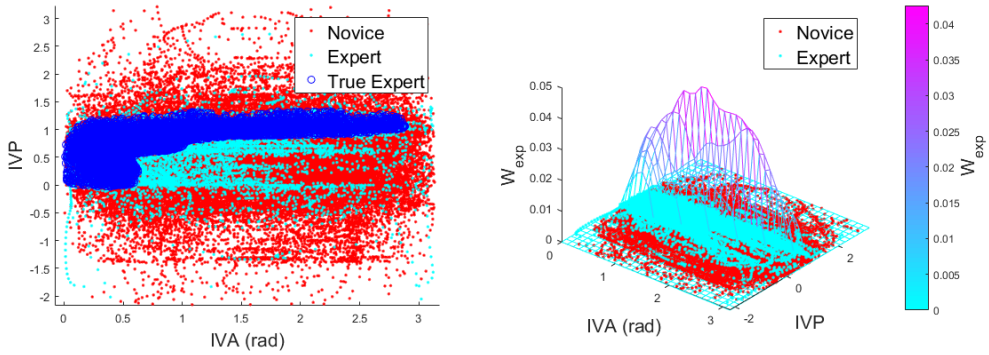


(a) IVA vs. IVP with Class Labels

(b) Per-Segment Demerits (y_i)

Figure 6.33: Intent Vector data (a) and demerit counts (b) (Obvious Novice and Expert) for Suturing task box-plot notch indicates range of 95% confidence for median separation.

The Intent Vector classification yielded a large separation among segment demerit counts (y_i) between Expert and Novice surgeons. A plot of these values for each class is given in Figure 6.33b. The mean segment demerit count was found to be 65.9 ($std = 105.2$) for Novices and 22.6 ($std = 27.7$) for Experts. The relevance weights (W_{exp}) and 'True Expert' data in the Intent Vector space are shown in Figure 6.34.



(a) 'True Expert' Region

(b) 'True Expert' Probability

Figure 6.34: Intent Vector data with 'True Expert' data and RELIEF-RBF weights (Obvious Novice and Expert)

The Intent Vector framework yielded an average classification accuracy of 96.9% between

novices and experts using a LOUOpG scheme for all tasks combined (Table 6.19). The Intent Vector approach fails to pass the MAC criterion for all tasks. However it does achieve this criterion for the Pattern Cutting task.

Skill Level	Peg Transfer	Pattern Cutting	Intracorporeal Suturing
Novice	96.5 [100*] {100*}	100* [96] {96}	100* [92.3] {92.3}
Expert	83.3 [83.3] {86.2}	100* [90] {100*}	92.3 [87.5] {100*}
Macro Accuracy	94.2 [97.1] {97.6}	100* [94] {97.2}	97.1 [90] {95.2}

* Achieves 100% Classification

Table 6.19: Intent Vectors [Aggregate Metrics] {Combined Features} classification accuracy (%)

The per-time step accuracy can also be estimated for the Intent Vector framework by recording the per time-step demerit values y_i for Obvious Novice and Experts (Eq. 3.81). To estimate this per time-step accuracy, the percentage of correct classifications y_i is computed over an entire task. A correct classification corresponds to $y_i = 1$ for an 'Obvious Novice' and $y_i = 0$ for an 'Obvious Expert' for any given time-step. This is analogous to the percentage of time that novices performed motions outside the 'True Expert' region (likewise the percentage of the time that experts performed motions inside the 'True Expert' region). The per time-step classification rate by skill level and task is given in Table 6.20.

Skill Level	Peg Transfer	Pattern Cutting	Intracorporeal Suturing
Novice	12.1	50.5	36.9
Expert	91.9	66.6	82.5
Macro Accuracy	31.7	53.5	42.9

Table 6.20: Percentage of time within each task that was effectively used to classify skill (e.g. 36.9% of novice task data was used to correctly classify them as novices, the remainder was not useful for correct classification, i.e. overlapped with expert data.) (%)

An example plot of Expert versus Novice total segment demerits and the learned thresholds

T_{sk} (Eq. 3.82) from all LOUOpG iterations is given in Figure 6.35 for the Intracorporeal Suturing task. Results suggest the existence of an ideal threshold (obtainable using all available data) that provides clear separation between Novice and Expert data in the Suturing task.

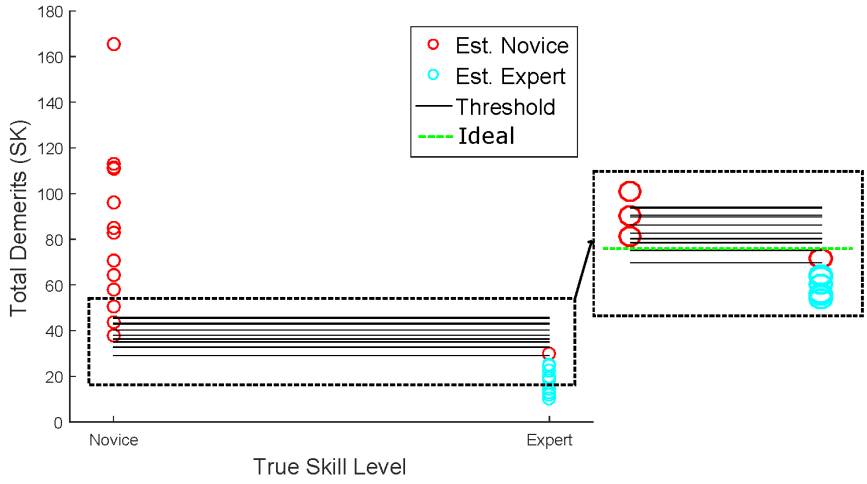


Figure 6.35: LOUOpG classification using Intent Vectors with thresholds (T_{sk}) and ideal separable threshold.

For comparison, the LDA classifier using the aggregate task metric features ($\bar{\chi}$) achieved the classification rates in square brackets in Table 6.19. These measures failed to achieve 100% (macro accuracy) classification for any of the tasks. The Intent Vector approach performed better than aggregate measures for both the Suturing and Cutting tasks, but worse in the Peg Transfer task. The combined feature vector $\hat{\chi}$ achieved equivalent or better macro accuracy than the aggregate metrics alone for all tasks; indicating improved performance through the incorporation of Intent Vectors.

6.5 Results: Benchmark Algorithms

To assess the success of the proposed algorithms classification was performed on the same data sets using common algorithms from prior art; Neural Networks and Random Forests. Prior art has found that both Neural Networks (NN) and Random Forests (RF) provide high classification accuracies compared to other machine learning techniques. These algorithms were tested on the

simulated data, the tissue grasp data, and the surgical skill evaluation data sets. Thus providing context for the classification results given above.

6.5.1 Linear Simulated Data

For the linear simulated data set, both the NN and RF classifiers were trained with the input vector given in Equation 6.13

$$x_T(t) = [x(t), \dot{x}(t), U(t)] \tag{6.13}$$

For the NN a network structure of 1 hidden layer, 5 nodes in the hidden layer, and one output node was chosen. The activation function in both the hidden and output layers was a sigmoid. The values from the output node were trained to $Y_{out} = [0,1]$, corresponding to classes $[1,2]$, respectively. For trajectory classification, the mean of the output values was used for all points in the trajectory. Classification was then performed according to Equation 3.85.

For the RF, an ensemble structure of 20 trees was chosen. The classifier was trained with out-of-bag prediction. Again, the values from the output node were trained to $Y_{out} = [0,1]$, corresponding to classes $[1,2]$, respectively. For trajectory classification, the mean of the output values was used for all points in the trajectory. Classification was then performed according to Equation 3.85.

Both the per time-step accuracy, wherein we attempt to classify at each point in a trajectory, and the per-trajectory accuracy, using the mean classification value are reported. The mean convergence time as a percentage of the total trajectory time is also reported. An overview of the classification accuracy for the linear simulated data can be found in Tables 6.21-6.22.

Separation	**Time-Step Accuracy [%]**	**Task Accuracy [%]**	**Convergence Rate [%]**
$\kappa = 1.1$	92.8	100	10
$\kappa = 1.2$	94.5	100	7.5

Table 6.21: Neural Network classification result for linear simulated data.

Separation	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
$\kappa = 1.1$	90.5	100	13
$\kappa = 1.2$	93.3	100	9

Table 6.22: Random Forest classification result for linear simulated data.

6.5.2 Non-Linear Simulated Data

For the non-linear simulated data set, both the NN and RF classifiers were trained with the input vector given in Equation 6.14

$$x_T(t) = [x(t), \dot{x}(t), U(t)] \tag{6.14}$$

For the NN a network structure of 1 hidden layer, 5 nodes in the hidden layer, and one output node was chosen. The activation function in both the hidden and output layers was a sigmoid. The values from the output node were trained to $Y_{out} = [0,1]$, corresponding to classes $[1,2]$, respectively. For trajectory classification, the mean of the output values was used for all points in the trajectory. Classification was then performed according to Equation 3.85.

For the RF, an ensemble structure of 20 trees was chosen. The classifier was trained with out-of-bag prediction. Again, the values from the output node were trained to $Y_{out} = [0,1]$, corresponding to classes $[1,2]$, respectively. For trajectory classification, the mean of the output values was used for all points in the trajectory. Classification was then performed according to Equation 3.85.

Both the per time-step accuracy, wherein we attempt to classify at each point in a trajectory, and the per-trajectory accuracy, using the mean classification value are reported. The mean convergence time as a percentage of the total trajectory time is also reported. An overview of the classification accuracy for the non-linear simulated data can be found in Tables 6.23-6.24.

Separation	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
$\kappa = 1.1$	81.8	100	28
$\kappa = 1.2$	85.8	100	21

Table 6.23: Neural Network classification result for non-linear simulated data.

Separation	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
$\kappa = 1.1$	74.7	100	36
$\kappa = 1.2$	80.9	100	26

Table 6.24: Random Forest classification result for non-linear simulated data.

6.5.3 Tissue Identification

For the Tissue Grasp data set, both the NN and RF classifiers were trained with the input vector given in Equation 6.15

$$x(t) = [\theta, \dot{\theta}, F] \tag{6.15}$$

For the NN a network structure of 1 hidden layer, 5 nodes in the hidden layer, and one output node was chosen. The activation function in both the hidden and output layers was a sigmoid. The values from the output node were trained to $Y_{out} = [0, 1]$, corresponding to classes [*Liver*, *Pancreas*], respectively. For grasp classification, the mean of the output values was used for all points in the grasp. Classification was then performed according to Equation 3.85. A NN model with 10 hidden nodes was also employed but resulted in worse performance than the 5 node variant.

For the RF, an ensemble structure of 20 trees was chosen. The classifier was trained with out-of-bag prediction. Again, the values from the output node were trained to $Y_{out} = [0, 1]$, corresponding to classes [*Liver*, *Pancreas*], respectively. For grasp classification, the mean of the output values was used for all points in the grasp. Classification was then performed according to Equation 3.85.

For this cross validation analysis, first one patients tissue data was left out from each class for training the NN and RF models, classification was then tested using the left out patient. This cross validation is referred to as Leave-One-Donor Out (LODO) and is used to assess inter-patient variability.

The cross validation was repeated using a Leave-One-Location-Out (LOLO) scheme. In this approach intra-patient variability was assessed by storing all locations for a given donor except one as training data, the model was then trained using this training set. Classification was then performed using the left out location, all for one donor at a time.

Both the per time-step accuracy, wherein we attempt to classify at each point in a grasp, and the per-trajectory accuracy, using the mean classification value are reported. The mean convergence time as a percentage of the total grasp time is also reported. An overview of the classification accuracy for the tissue grasp data can be found in Tables 6.25-6.28.

Class	**Time-Step Accuracy** [%]	**Task Accuracy** [%]	**Convergence Rate** [%]
Liver	66.8	80.0	39
Pancreas	54.8	60.0	41
Combined	60.9	70.0	40

Table 6.25: Neural Network classification result for tissue grasp data (LODO cross validation)

Class	**Time-Step Accuracy** [%]	**Task Accuracy** [%]	**Convergence Rate** [%]
Liver	66.1	76.0	21
Pancreas	61.2	60.0	30
Combined	63.7	68.0	26

Table 6.26: Neural Network classification result for tissue grasp data (LOLO cross validation)

Class	**Time-Step Accuracy** [%]	**Task Accuracy** [%]	**Convergence Rate** [%]
Liver	59.5	80.0	35
Pancreas	51.8	60.0	37
Combined	55.8	70.0	36

Table 6.27: Random Forest classification result for tissue grasp data (LODO cross validation)

Class	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
Liver	77.1	92.0	24
Pancreas	61.2	56.0	29
Combined	69.3	74.0	26

Table 6.28: Random Forest classification result for tissue grasp data (LOLO cross validation)

6.5.4 Surgical Skill Classification

For the EDGE data set, both the NN and RF classifiers were trained using binary classifiers for skill from the raw surgical tool motion of the EDGE data set. As indicated in Section 5.3, the full data set consists of 13 states given in Equation 5.7. Using RELIEF-RBF feature weighting the states with the highest potential separability were reduced to include $\hat{\chi}_t = [\theta, \ddot{x}, \ddot{y}, \ddot{z}]$.

For the NN a network structure of 1 hidden layer, 7 nodes in the hidden layer, and one output node was chosen. The activation function in both the hidden and output layers was a sigmoid. The values from the output node were trained to $Y_{out} = [0,1]$, corresponding to classes $[Novice,\ Expert]$, respectively. For task-level classification, the mean of the output values was used for all points in the tool motion segments. Classification was then performed according to Equation 3.85. A NN model with 10 hidden nodes was also employed but resulted in worse performance than the 7 node variant.

For the RF, an ensemble structure of 20 trees was chosen. The classifier was trained with out-of-bag prediction. Again, the values from the output node were trained to $Y_{out} = [0,1]$, corresponding to classes $[Novice,\ Expert]$, respectively. For task-level classification, the mean of the output values was used for all points in the tool motion segments. Classification was then performed according to Equation 3.85.

For this cross validation analysis, one subject from each skill level group was left out for training the NN and RF models, classification was then tested using the left out subjects. This cross validation is referred to as Leave-One-User-Out-per-Group (LOUOpG). Classification was performed independently for each FLS task; Peg Transfer, Pattern Cutting, and Suturing.

Both the per time-step accuracy, wherein we attempt to classify at each point in a surgical motion segment, and the per-task accuracy, wherein the full resultant sum from all segments is used to classify are reported. The mean convergence time as a percentage of the total task time

is also reported. An overview of the classification accuracy for surgical skill level for the three FLS task cross validations can be found in Tables 6.29- 6.34. The per skill-level classification rate is also included in each table.

Class	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
Novice	100	100	0
Expert	1.0	0.0	0
Combined	75.4	50.0	0

Table 6.29: Neural Network classification result for EDGE surgical motion data (Peg Transfer task)

Class	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
Novice	100	100	0
Expert	0.0	0.0	0
Combined	81.4	50.0	0

Table 6.30: Neural Network classification result for EDGE surgical motion data (Pattern Cutting task)

Class	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
Novice	100	100	0
Expert	0.0	0.0	0
Combined	86.8	50.0	0

Table 6.31: Neural Network classification result for EDGE surgical motion data (Suturing task)

Class	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
Novice	98.8	100	1.9
Expert	26.5	20.1	10
Combined	81.04	60.3	6.3

Table 6.32: Random Forest classification result for EDGE surgical motion data (Peg Transfer task)

Class	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
Novice	99.9	100.0	1.5
Expert	0.21	0.0	3.7
Combined	81.4	50.0	2.6

Table 6.33: Random Forest classification result for EDGE surgical motion data (Pattern Cutting task)

Class	Time-Step Accuracy [%]	Task Accuracy [%]	Convergence Rate [%]
Novice	99.7	100.0	1.8
Expert	6.59	0.0	10
Combined	87.5	50.0	5.1

Table 6.34: Random Forest classification result for EDGE surgical motion data (Suturing task)

6.6 Algorithm Timing

To compare the speed of the training algorithms and online classification code, each of the proposed algorithms and comparison algorithms were run on identical data sets. The computation time for each training approach was evaluated using the tic-toc functions in Matlab. The hyper-parameters used in the timing tests were as follows: $\lambda = 0.1$ for DLS, $ns = 11$ for DPP, $r_w = 0.3$ for RELIEF-RBF, 1 hidden layer and 5 nodes for Neural Network, 20 trees for Random Forest.

For the timing test a training data with 200200 data points and 3 dimensions was used. In order to achieve reliable timing results, a given training routine was run 10 times and the resultant timing was divided by 10 to achieve an average training time. Additionally each 10 epoch training was run 3 times in order to allow Matlab to allocate the necessary memory. For the online timing test an online data set of 1000 data points and 3 dimensions was used. For each algorithm the online classification time was run 10 times and the resultant timing was divided by 10 to achieve an average run time estimate. A plot of the relative timing for both training and online classification for all algorithms is given in Figures 6.36-6.37.

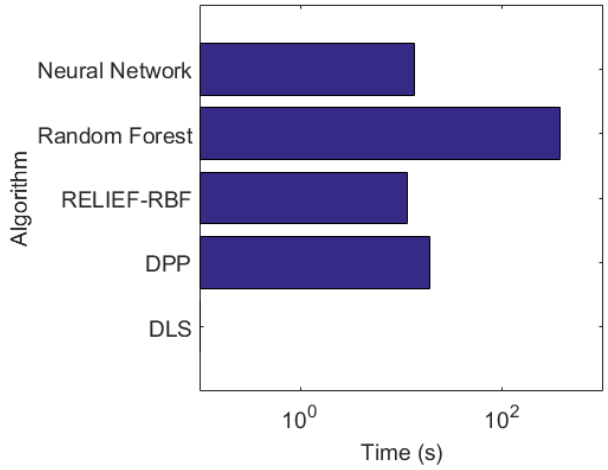


Figure 6.36: Comparison of relative run times for the training routine.

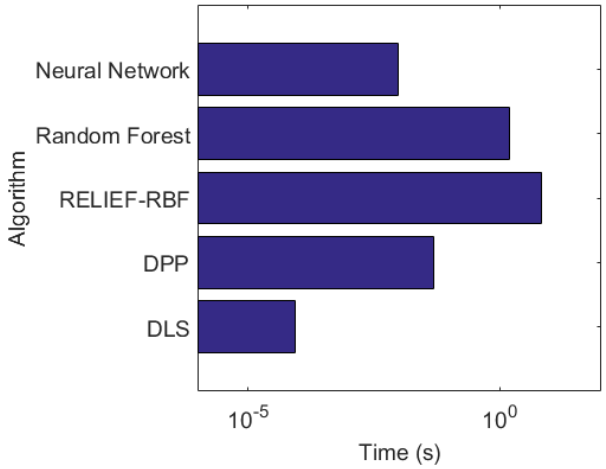


Figure 6.37: Comparison of relative run times for online classification.

6.7 Overview of Results

We have demonstrated the performance of several algorithms using multiple data sets. Both the proposed algorithms (DLS, DPP, RELIEF-RBF, and Intent Vectors) as well as the comparison algorithms (Neural Networks and Random Forests) were applied to the 3 main data sets (simulated, tissue grasping, and surgical tool motion). A comparison of the per-time step classification accuracies across all algorithms is given in Table 6.35. A comparison of the per-trajectory classification accuracy across all algorithms is given in Table 6.36. For the surgical tool motion data set, the macro accuracy across all three FLS tasks is reported. Since the Intent Vectors method relies on a demerit count, the per time-step accuracy estimate is significantly lower.

		Data Sets						
		Linear $\kappa = 1.1$	Linear $\kappa = 1.2$	Non-Linear $\kappa = 1.1$	Non-Linear $\kappa = 1.2$	Tissue Grasp LODO	Tissue Grasp LOLO	Surgical Tool Motion
Algorithms	DLS	**92.9**	93.9	80.9	**86.6**	**73.8**	**78.8**	-
	DPP	**92.9**	93.7	78.6	82.8	57.6	59.9	67.6
	RELIEF-RBF	71.8	75.2	77.4	74.8	64.6	62.9	78.2
	Intent Vectors	-	-	-	-	-	-	43.6
	NN	92.8	**94.5**	**81.8**	85.8	60.9	63.7	79.8
	RF	90.5	93.3	74.7	80.9	55.8	69.3	**82.4**

Table 6.35: Per time-step classification results, all algorithms (best accuracy in bold)

		Data Sets						
		Linear $\kappa = 1.1$	Linear $\kappa = 1.2$	Non-Linear $\kappa = 1.1$	Non-Linear $\kappa = 1.2$	Tissue Grasp LODO	Tissue Grasp LOLO	Surgical Tool Motion
Algorithms	DLS	100	100	100	100	**90.0**	**86.0**	-
	DPP	100	100	100	100	80.0	70.0	49.3
	RELIEF-RBF	100	100	100	100	80.0	76.0	49.3
	Intent Vectors	-	-	-	-	-	-	**96.9**
	NN	100	100	100	100	70.0	68.0	50.0
	RF	100	100	100	100	70.0	74.0	54.45

Table 6.36: Per trajectory classification results, all algorithms (best accuracy in bold)

Chapter 7

Discussion and Conclusion

7.1 Analysis of Results

This work has presented multiple new algorithms for use in dynamic discriminant analysis: Discriminant Least Squares (DLS), Discriminant Phase Portrait (DPP), RELIEF-RBF, and Intent Vectors. These algorithms were applied to sample applications within computational surgery and compared with state of art algorithms. The analysis of these results is presented for each data set separately.

7.1.1 Linear Simulated Data

The linear simulated data set was used as a baseline validation for the proposed algorithms. The first three algorithms were used to classify both linear data set separation values ($\kappa = 1.1$ and $\kappa = 1.2$). As indicated in Tables 6.1, 6.5, and 6.12, the proposed algorithms provided high classification accuracies for the per time-step classification under leave-one-out cross validation. Both the DLS and DPP approaches provided 93% and 94% accuracies for each data separation value (κ) respectively. The RELIEF-RBF approach yielded a slightly lower accuracy of $72-76\%$ for per time-step classification. Additionally, as indicated in Tables 6.21, 6.22 these algorithms achieved similar results to the Neural Network (NN) and Random Forest (RF) approaches. The NN and RF approaches yielded accuracies between 90% - 94%.

For the full trajectory classification, all algorithms yielded an accuracy of 100% indicating that all algorithms were successful at classifying complete time series data sets. This provides initial validity for the use of the 3 proposed algorithms (DLS, DPP, and RELIEF-RBF) when

classifying time-series data sets with varying degrees of between-class separation.

It is interesting to note that while RELIEF-RBF provided worse classification per time step, it still provided perfect classification for the complete segments. This indicates that the RELIEF-RBF approach may make incorrect estimates for some data, but those estimates contribute only low confidence weights to the overall estimate sum. This effect can be seen in Figure 6.21 where only low weights are given in the overlapping regions where misclassification occurs. The incorrect classifications made at each time step were made with relatively low confidence (Figures 6.2, 6.9, 6.21). This implies that for simplistic systems these algorithms are aware when poor classifications are made.

It is also interesting to note that the λ term in the DLS algorithm is ineffectual for linear data sets. Values of $0 < \lambda < 0.4$ have identical classification accuracies. This may be attributable to the Φ_* parameters deviating at equal rates from their respective data sets in the linear case.

The results agreed fairly well with the expected outcomes (Table 5.7). DLS, DPP and RELIEF-RBF all achieved very high classification accuracies.

7.1.2 Non-Linear Simulated Data

The non-linear simulated data set served as a secondary baseline validation for the proposed algorithms. The DLS, DPP, and RELIEF-RBF algorithms were used to classify both non-linear data set separation values ($\kappa = 1.1$ and $\kappa = 1.2$). As indicated in Tables 6.2, 6.6, and 6.13, the proposed algorithms again provided high classification accuracies for the per time-step classification under leave-one-out cross validation. Both the DPP and RELIEF-RBF approaches provided accuracies of 75%- 83% for each data separation value (κ) respectively. The DLS approach yielded slightly better results ($81,87\%$) for per time-step classification. Additionally, as indicated in Tables 6.23, 6.24 these algorithms achieved similar results to the Neural Network (NN) and Random Forest (RF) approaches. The NN and RF approaches yielded accuracies between 74% - 85%.

For the full trajectory classification, all algorithms again yielded an accuracy of 100% indicating that all algorithms were successful at classifying complete time series data sets. This provides further validity for the use of the 3 proposed algorithms (DLS, DPP, and RELIEF-RBF).

Again it is noted that while DPP and RELIEF-RBF provided worse classification per time step, these approaches still resulted in 100% accuracy for the complete segments. This indicates

that the DPP approach can also make incorrect estimates for some data, but those estimates contribute only low confidence weights to the overall estimate sum. This effect can be seen in Figure 6.11 where only low weights are given in the overlapping regions where misclassification occurs. The incorrect classifications made at each time step were made with relatively low confidence (Figures 6.4, 6.12, 6.25). This implies that for simplistic systems these algorithms are aware when poor classifications are made.

Here it is noted that the λ term in the DLS algorithm provides a crucial benefit for non-linear systems. When $\lambda = 0$ classification accuracy is around 70%. however accuracy increased above 81% for $\lambda = 0.4$. This may be attributable to the discriminant Φ_* parameters identifying a more optimal fit to each non-linear trajectory relative to data from other classes.

In this case the results agreed fairly well with the expected outcomes (Table 5.7). However the DLS did not have significant performance issues with the non-linear data, instead in this case the λ term simply became more relevant. The DPP and RELIEF-RBF approaches did perform as expected.

7.1.3 Tissue Identification

The grasping data for the cadaveric tissue served as a more complicated, highly overlapped, data set for the proposed algorithms. The DLS, DPP, and RELIEF-RBF algorithms were used to classify tissue types via back end sensing for leave-one-donor-out (LODO) and leave-one-location-out (LOLO) cross validations. As indicated in Tables 6.7 and 6.14, the DPP and RELIEF-RBF algorithms did not provide high classification accuracies for the tissue identification data. However as seen in Table 6.3 the DLS algorithm provides relatively high classification accuracy given the overwhelming similarity in the systems. The DLS approach yielded $(74,79\%)$ accuracy for the per time-step classification. This translated to 90% and 86% accuracy for the complete grasp classification. In comparison as indicated in Tables 6.25- 6.28 the NN and RF approaches achieved per time-step classification accuracies below 70%. The NN and RF approaches yielded complete grasp accuracies between 68% - 74%.

In contrast to the expected results (Table 5.7 - 5.8), the DLS algorithm performed better than DPP or RELIEF-RBF for the tissue identification. The DPP and RELIEF-RBF approaches both yielded an overall grasp accuracy between 70% - 80%. However the DPP and RELIEF-RBF approaches still performed better than NN or RF when classifying entire segments, even though their per time step classifications were worse. The tissue grasp indicates that the first three

proposed algorithms provide clear improvements when classifying non-linear, overlapping data. This provides validity for the use of the 3 proposed algorithms (DLS, DPP, and RELIEF-RBF) in other data sets where time-series data is seemingly inseparable.

While the DPP and RELIEF-RBF approaches do provide somewhat worse accuracies than DLS, is apparent from the confidence values that these classification estimates reflect the uncertainty. As seen in Figure 6.17 and 6.32, the confidence weights for both correct and incorrect classifications are on average 0.12 and $2e{-}2$ for DPP and RELIEF-RBF, respectively. When compared with the confidence weights for the linear and non-linear data sets, the confidences are considerably lower. This indicates the DPP and RELIEF-RBF algorithms know they have poor discriminating ability for this data. In other words, the classifications while incorrect could be dismissed based on a threshold of confidence. This confidence is directly related to the information gain present in the data.

For the DLS algorithm it is evident that the λ scaling parameter again constitutes a core feature for classification accuracy (Fig. 6.5). Specifically for the LODO cross validation, accuracy significant improves for $\lambda = 0.48$ when compared with $\lambda = 0$ (the OLS solution). This indicates that for systems with significant overlap or inconsistencies, the discriminant least squares formulation has the potential to isolate parameters which maximize the separation between classes. As shown for both the non-linear simulated data and the tissue data, this effect is most pronounced for non-linear systems.

In this case the results differed from the expected outcomes (Table 5.7). The DPP and RELIEF-RBF approaches did provide acceptable classification results. However the DLS did not have performance issues with the tissue grasp data, instead in this case the DLS algorithm performed the best. This is again likely attributable to the λ term correctly identifying key discriminant parameters.

7.1.4 Surgical Skill Level Classification

The most difficult of the data sets in terms of classification was the surgical skill level from laparoscopic surgical tool motion (EDGE data set). The DPP, RELIEF-RBF, and Intent Vector approaches were all employed to classify expert from novice surgeons for three Fundamentals of Laparoscopic Surgery tasks. As indicated in Table 6.36, neither the DPP and RELIEF-RBF approaches, nor the NN and RF comparisons achieved the MAC classification criterion for the full task classification. Using only the raw tool motion, the full task classification achieved

around 50% accuracy. This is likely due to the novice and expert surgeons having minimal separable regions in the raw motion data. This can further be seen by the fact that the per time-step accuracy for experts is very low, meaning they inherently behave like novices periodically.

The per time-step classification was better than the full task classification using the raw tool motion. The RELIEF-RBF, NN, and RF approaches all achieved around 80% classification accuracy per time-step. This can be attributed to the algorithms all biasing their predictors towards the novice data (Tables 6.16, 6.29, and 6.32). More novice data exists per task because their task attempts tend to take longer. Therefore, the overwhelming amount of novice data in the training set causes these algorithms to effectively classify all data as novice since their is no region of separability

It appears that DPP while providing lower overall accuracy for the per-time step classification, does isolate regions of expert probability that other algorithms do not. For example all other algorithms achieve nearly 100% accuracy for novice and 0% for the expert time-step classification. This indicates that these algorithms cannot find regions of adequate separability (analogous to a separating hyper plane). However, as seen in Tables 6.9 - 6.11 the DPP algorithm has a higher percentage of correct classifications per time-step for the expert skill level (30%). This potentially indicates that the DPP approach is capable of finding small pockets of separable data where other algorithms cannot.

The major conclusion provided by the RELIEF-RBF approach is that the two classes (Obvious Novice and Obvious Expert) are effectively inseparable in terms of raw tool motion. This is further evidenced by the poor performance of the Neural Network and Random Forest benchmark algorithms. This indicates that no current machine learning algorithm will work for such a data set. Furthermore, it is impressive that the DPP approach was able to identify partial regions where expert classification was possible. Additionally, this result indicates that research in this field *needs* to move beyond raw data for classification and instead focus on derived features specific to skill evaluation, such as the Intent Vector approach.

The clearly superior approach for the surgical skill level classification is the Intent Vectors framework. Since this approach was designed specifically for surgical tool motion, it is not surprising that it achieved an macro classification accuracy of 96.9% and for certain FLS tasks it achieved the MAC criterion. Since the Intent Vectors approach is based on a demerit count of deviation from the region of high probability in the IVP-IVA space, the per time-step classification estimate is significantly lower (44%). This indicates that the demerit classification

system is necessary in order to identify and penalize motions that differ from expert motions. Classification is achieved for each segment and subsequently each completed task. The overall classification rate rivals or surpasses prior literature especially under LOUOpG cross validation. We note that this approach fails to achieve the MAC criterion for all three FLS tasks. However, our Intent Vector classifier does partially succeed under the MAC criterion for two special cases: the Cutting task and identifying obvious Novices in the the Suturing task. Closer inspection of Fig. 6.35 reveals that the Intent Vector can fully separate the Suturing task (and hence classify with 100% accuracy to achieve the MAC criterion) given an ideal threshold. Furthermore, for the Cutting and Suturing tasks, the Intent Vector provides additional value beyond summary metrics like task time. Notably, it returns classification results upon completion of each motion segment. This permits use cases such as 1) identifying only the worst portions of a surgical video for streamlined targeted review or 2) providing skill feedback in near real-time at the completion of every motion. The segmentation approach used has the additional benefits of not requiring manual segmentation and being task agnostic (e.g. it does not need to know a priori the structure or steps of, say, a suturing task but can directly operate on tool motion data).

In this case the results differed from the expected outcomes (Table 5.7). Neither the DPP and RELIEF-RBF approaches, nor the comparison approaches (NN and RF) were able to classify skill level from the raw motion data. It is clear that the similarity between expert and novice in raw motion data is overwhelming. Despite this, the Intent Vector feature and framework did perform as expected and provided high classification accuracy for surgical skill evaluation.

7.2 Limitations and Possible Extensions

There are a number of limitations related to the proposed algorithms. For the first three algorithms (DLS, DPP, and RELIEF-RBF), the training requires a data set with a large number of data points. The smallest data set used had a total of 14,000 data points for both classes. For data sets with very sparse data, the probability estimates would likely be sensitive to noise and scale. Of the data sets used, the highest dimensionality tested was 6 dimensions. There is not inherent concern related to higher dimensional data sets, however it is likely (especially for the DPP approach) that high dimensional data sets will require a lengthy training time.

While the DPP and RELIEF-RBF approaches can operate on data from a variety of distributions, the DLS approach inherently requires a parametric model for training of the discriminant

parameters. Therefore for linear data a known parametric model is required to train the parameters. Similarly for non-linear data, an approximated model (linear in terms) is required. This has the potential to provide poor classification results if the non-linearity cannot be sufficiently linearized or expressed as linear in parameters.

DPP and RELIEF-RBF assume an underlying Gaussian distribution, it is possible that other distributions would return sub-optimal results. For example the Radial Basis Function kernel used in RELIEF-RBF would potentially ignore data from the tail of a Poisson distribution when assessing data point weights. Similarly it is important that the data for RELIEF-RBF be mean variance scaled otherwise the squared exponential function will be scale dependent.

For the DPP approach the effect of the grid coarseness parameter (*ns*) has the potential to negatively impact results. As discussed previously, a value of $ns = 1$ would result in a naive Bayes classifier. Conversely for some data sets a high value of *ns* could result in over fitting of the training data. For any value of *ns* there is the possibility that the dividing lines for the grid regions will give non-uniform preference to one class while dividing the other class into more, low-density regions. For this reason it is recommended to utilize at least two different *ns* parameters while testing this algorithm or further investigate a method to compute an optimal *ns* value based on the given data.

For the RELIEF-RBF algorithm, the effect of the r_w hyperparameter was not fully assessed. It is clear that a value of $r_w = 1$ would simply result in the full data set being used for training the GPR model. Similarly very small values of r_w would results in a very small region of model data. Therefore the optimal value lies between $0.01 < r_w < 1$. In majority of testing, values of $r_w = 0.3, 0.5$ were used with successful outcomes. However for very large data sets, this value may need to be smaller simply to make the GPR training tractable.

The Intent Vector approach was inherently formulated to deal with surgical tool motion and so in the proposed state is limited to that application. One obvious extension would include applying the Intent Vectors approach to robotic surgical motion data given its similarity to laparoscopic data. It is possible that this approach could also be augmented to perform skill classification for other motion analysis applications where smooth, directed motion is indicative of experienced users. Examples of such applications include remote flight controllers, dentistry, or open surgery.

For all proposed algorithms the training data applications were limited to binary classification. In the case of the DLS approach, the extension to ternary classification is straightforward.

In the case of DPP and RELIEF-RBF, the extension to multi-class classification would require the use of a one-versus-rest or one-versus-all scheme [83]. These transformations are well known in the literature and no inherent limitation in the proposed algorithm would prevent this extension.

Logical extensions of the more general algorithms (DLS, DPP, and RELIEF-RBF) include applications with non-linear, non-parametric time series data sets with significant overlap between systems. Example potential applications include classifying skill level in athletes based on motions such as swings (tennis, golf, pool etc). In the field of gait analysis, these algorithms could be used to analyze healthy gaits from disabled gaits. Another possible application area is in gesture recognition from non-contact hand gestures. For example the Leap Motion (Leap Motion, Inc. San Francisco, CA) records hand motion, the time series of hand position data could be used to classify various motion gestures. In the field of non-rigid registration research has shown the ability to semantically label various organs [110]. The proposed algorithms could further be used to identify regions of high probability in point clouds for features to track.

7.3 Overview of Presented Work

This thesis has presented four approaches designed to address the need for a dynamic discriminant algorithm within computational surgery. Computational surgery consists of several ongoing research themes in which data sets are comprised of time based, non-linear and non-parametric models. This field of computational surgery is concerned with deploying technologies to make the operating room safer and provide more intelligent surgical tools. Thus the goal of these algorithms is to define a framework which provides high accuracy classification and identification for complex, time-series systems. The goal of this work has been to explore these new algorithms within the context of two applications, the first involves tissue grasping for the purpose of tissue identification using a minimally invasive surgical tool. The second application was the classification of surgical skill level using surgical motion data from a laparoscopic simulator.

Chapter 2 provided an overview of prior art in the fields of tissue identification via surgical graspers, surgical skill level classification, and an overview of existing machine learning methods. Chapter 3 outlined the proposed algorithms to be tested. Chapter 4 presented the current generation of the sensorized surgical grasper “Smart Tool”. Chapter 5 outlined the data sets

used for analysis of the proposed algorithms. Finally Chapter 6 detailed the implementation and results of the various algorithms on the sample applications and data sets.

We have shown validation for the three general algorithms (DLS, DPP, and RELIEF-RBF) through the use of simulated linear and non-linear data sets. In these data sets the proposed algorithms achieved equivalent (and sometimes improved) classification results when compared with common algorithms in the field; notably Neural Networks and Random Forests. While not an exhaustive comparison between the proposed and benchmark algorithms, these comparisons have served as a baseline for evaluating these new algorithms. These new algorithms provide additional value through the use of the confidence weights given for each classification estimate. This enables additional use cases such as varying the arbitration weight used in a shared control algorithm where two agents are providing system inputs with potential disagreement.

Section 6.6 provides a brief overview of the relative time required to train and classify the proposed algorithms. It is clear that the DLS algorithm is the fastest approach in terms of training and online estimates. This is not surprising given that the Moore-Penrose pseudo inverse is well understood and efficiently implemented on modern computers. The RELIEF-RBF and DPP approaches had similar training times to the Neural Network approach, indicating that they have similar computational complexity to well understood algorithms in the field. The Random Forest approach had the longest training time. For online classification, the DPP and Neural Network approaches had similar timing results. The RELIEF-RBF approach had the slowest online classification timing. This is primarily due to the Kernel matrix multiplication required in Gaussian Process approaches. The online classification for RELIEF-RBF can be improved for lower training ratios (r_w). While this analysis does not fully asses the computational complexity of the proposed algorithms, it does give a relative time scale for the training and online classification routines.

For the tissue identification via back end sensing application, it was found that the proposed algorithms provide equivalent if not better classification than the comparison algorithms. The DLS approach in particular provides complete grasp classification of 90% for the LODO classification. This result exceeds both the NN and RF approaches, as well as the Ordinary Least Squares (OLS) solution ($\lambda = 0$). In this application the DLS approach provides a surprisingly good classification, especially considering the degree of similarity between the two tissues. Additionally the DPP and RELIEF-RBF approaches also exceed the results of the comparison algorithms. This application has provided further validation for the proposed algorithms.

Finally in the surgical skill evaluation from surgical tool motion application, it was found that surgical tool motion alone does not provide adequate classification accuracy (i.e. $< 90\%$). While the DPP, RELIEF-RBF, NN, and RF approaches do provide per time-step accuracies around 80%, these results are skewed by the preponderance of novice data; expert data is classified very poorly. As a result the per task accuracy is very low (50%). In contrast the Intent Vector feature and framework provides near ideal classification results for all FLS tasks (97%) and achieves the MAC criterion for some tasks. This indicates that raw tool motion alone is not sufficient for skill classification. Instead derived features which represent higher level motion measures are required to achieve high accuracy skill level estimates. It was also shown that the Intent Vector approach improves upon existing surgical skill level metrics such as motion economy and path length (Table 6.19).

For each of the sample applications it was shown that the proposed algorithms meet or exceed the classification provided by existing algorithms in field and furthermore provide honest confidence estimates. These approaches should be applicable to a variety of other applications where classification of time-series data is required. A copy of all source code for these functions is available via Git repository (`https://github.umn.edu/labmrd/Dockter_Thesis_2017_Code`) or by request.

Bibliography

[1] Linda Kohn. To Err Is Human: Building a Safer Health System. National Academy Press, 2000.

[2] Sherry L Murphy, Jiaquan Xu, Kenneth D Kochanek, et al. National vital statistics reports. National vital statistics reports, 60(4):1–116, 2012.

[3] Chunliu Zhan and Marlene R Miller. Excess length of stay, charges, and mortality attributable to medical injuries during hospitalization. Jama, 290(14):1868–1874, 2003.

[4] Gary Null, Carolyn Dean, Martin Feldman, and Debora Rasio. Death by medicine. Journal of Orthomolecular Medicine, 20(1):21–34, 2005.

[5] Randy P Fiorentino, Marc A Zepeda, Bram H Goldstein, Cameron R John, and Mark A Rettenmaier. Pilot study assessing robotic laparoscopic hysterectomy and patient outcomes. Journal of minimally invasive gynecology, 13(1):60–63, 2006.

[6] Jason D Wright, Cande V Ananth, Sharyn N Lewin, William M Burke, Yu-Shiang Lu, Alfred I Neugut, Thomas J Herzog, and Dawn L Hershman. Robotically assisted vs laparoscopic hysterectomy among women with benign gynecologic disease. Jama, 309(7):689–698, 2013.

[7] Atul Gawande. Two hundred years of surgery. The New England Journal of Medicine, 366:1716–1723, 2012.

[8] Marc Garbey, Barbara Lee Bass, Christophe Collet, Michel De Mathelin, and Roger Tran-Son-Tay. Computational Surgery and Dual Training. Springer, 2010.

[9] Gregory Tholey, Jaydev P Desai, and Andres E Castellanos. Force feedback plays a significant role in minimally invasive surgery: results and analysis. Annals of surgery, 241(1):102–109, 2005.

[10] Christopher R Wagner, Nicholas Stylopoulos, Patrick G Jackson, and Robert D Howe. The benefit of force feedback in surgery: Examination of blunt dissection. Presence: teleoperators and virtual environments, 16(3):252–262, 2007.

[11] Anna Mases, Antonio Montes, Rocio Ramos, Lourdes Trillo, and Margarita M Puig. Injury to the abdominal aorta during laparoscopic surgery: an unusual presentation. Anesthesia & Analgesia, 91(3):561–562, 2000.

[12] Antonio Bicchi, Gaetano Canepa, Danilo De Rossi, Pietro Iacconi, and Enzo Scilingo. A sensorized minimally invasive surgery tool for detecting tissutal elastic properties. In IEEE Internation Conference on Robotics and Automation, 1996.

[13] Mark MacFarlane, Jacob Rosen, Blake Hannaford, Carlos Pellegrini, and Mika Sinanan. Force-feedback grasper helps restore sense of touch in minimally invasive surgery. Journal of Gastrointestinal Surgery, 3(3):278–285, 1999.

[14] Astrini Sie, Michael Winek, and Timothy M Kowalewski. Online identification of abdominal tissues in vivo for tissue-aware and injury-avoiding surgical robots. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pages 2036–2042. IEEE, 2014.

[15] Timothy M Kowalewski. Real-Time Quantitative Assessment of Surgical Skill. PhD thesis, University of Washington, 2012.

[16] Zhuohua Lin, Munenori Uemura, Massimiliano Zecca, Salvatore Sessa, Hiroyuki Ishii, Morimasa Tomikawa, Makoto Hashizume, and Atsuo Takanishi. Objective skill evaluation for laparoscopic training based on motion analysis. Biomedical Engineering, IEEE Transactions on, 60(4):977–985, 2013.

[17] Jeffrey H Peters, Gerald M Fried, Lee L Swanstrom, Nathaniel J Soper, Lelan F Sillin, Bruce Schirmer, Kaaren Hoffman, Sages FLS Committee, et al. Development and validation of a comprehensive program of education and assessment of the basic fundamentals of laparoscopic surgery. Surgery, 135(1):21–27, 2004.

[18] John D Birkmeyer, Jonathan F Finks, Amanda O'Reilly, Mary Oerline, Arthur M Carlin, Andre R Nunn, Justin Dimick, Mousumi Banerjee, and Nancy JO Birkmeyer. Surgical skill and complication rates after bariatric surgery. New England Journal of Medicine, 369(15):1434–1442, 2013.

[19] Magdalena K Chmarra, Stefan Klein, Joost CF de Winter, Frank-Willem Jansen, and Jenny Dankelman. Objective classification of residents based on their psychomotor laparoscopic skills. Surgical endoscopy, 24(5):1031–1039, 2010.

[20] Lawton Verner, Dmitry Oleynikov, Stephen Holtmann, H Haider, and L Zhukov. Measurements of the level of surgical expertise using flight path analysis from da vinci robotic surgical system. Medicine Meets Virtual Reality 11: NextMed: Health Horizon, 94:373–378, 2003.

[21] R Vecchio, BV MacFayden, and F Palazzo. History of laparoscopic surgery. Panminerva medica, 42(1):87–90, 2000.

[22] Esther Kuhry. Laparoscopic surgery versus open surgery for colon cancer: short-term outcomes of a randomised trial. The lancet oncology, 6(7):477–484, 2005.

[23] Anthony R Lanfranco, Andres E Castellanos, Jaydev P Desai, and William C Meyers. Robotic surgery: a current perspective. Annals of surgery, 239(1):14–21, 2004.

[24] Marc Garbey, David Thanoon, R Salmon, and B Bass. Multiscale modeling and computational surgery: application to breast conservative therapy. JSSCM, 5:81–89, 2011.

[25] Nikhil V Navkar, Zhigang Deng, Dipan J Shah, Kostas E Bekris, and Nikolaos V Tsekos. Visual and force-feedback guidance for robot-assisted interventions in the beating heart with real-time mri. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 689–694. IEEE, 2012.

[26] Marc Garbey, C Picard, Victoria Hilford, and S Vadakuttu. Toward an intelligent data and visualization desk for endovascular surgery. In Computers and Their Applications, pages 70–76, 2008.

[27] Don Risucci, Alan Geiss, Larry Gellman, Brian Pinard, and James Rosser. Surgeon-specific factors in the acquisition of laparoscopic surgical skills. The American Journal of Surgery, 181(4):289–293, 2001.

[28] NE Seymour, AG Gallagher, SA Roman, MK OBrien, DK Andersen, and RM Satava. Analysis of errors in laparoscopic surgical procedures. Surgical Endoscopy And Other Interventional Techniques, 18(4):592–595, 2004.

[29] Reza Ghavamian. Complications of laparoscopic and robotic urologic surgery. Springer, 2010.

[30] Alexander J Doud, Rodney Dockter, Sanket Chauhan, Robert Sweet, and Timothy Kowalewski. Automated electro-mechanical assessment of psychomotor skill for high-stakes certification in surgical robotics. Journal of Medical Devices, 7(3):030931–030932, 2013.

[31] MS Wilson, A Middlebrook, C Sutton, R Stone, and RF McCloy. Mist vr: a virtual reality trainer for laparoscopic surgery assesses performance. Annals of the Royal College of Surgeons of England, 79(6):403, 1997.

[32] JA Martin, G Regehr, R Reznick, H MacRae, J Murnaghan, C Hutchison, and M Brown. Objective structured assessment of technical skill (osats) for surgical residents. British Journal of Surgery, 84(2):273–278, 1997.

[33] Melina C Vassiliou, Liane S Feldman, Christopher G Andrew, Simon Bergman, Karen Leffondré, Donna Stanbridge, and Gerald M Fried. A global assessment tool for evaluation of intraoperative laparoscopic skills. The American Journal of Surgery, 190(1):107–113, 2005.

[34] Vivek Datta, Simon Bann, Mirren Mandalia, and Ara Darzi. The surgical efficiency score: a feasible, reliable, and valid method of skills assessment. The American journal of surgery, 192(3):372–378, 2006.

[35] Liane S Feldman, Sarah E Hagarty, Gabriela Ghitulescu, Donna Stanbridge, and Gerald M Fried. Relationship between objective assessment of technical skills and subjective in-training evaluations in surgical residents. Journal of the American College of Surgeons, 198(1):105–110, 2004.

[36] Vivek Datta, Sean Mackay, Mirren Mandalia, and Ara Darzi. The use of electromagnetic motion tracking analysis to objectively measure open surgical skill in the laboratory-based model. Journal of the American College of Surgeons, 193(5):479–485, 2001.

[37] Magdalena K Chmarra, Niels H Bakker, Cornelis A Grimbergen, and Jenny Dankelman. Trendo, a device for tracking minimally invasive surgical instruments in training setups. Sensors and Actuators A: Physical, 126(2):328–334, 2006.

[38] John D Mason, James Ansell, Neil Warren, and Jared Torkington. Is motion analysis a valid tool for assessing laparoscopic skill? Surgical endoscopy, 27(5):1468–1477, 2013.

[39] Timothy N Judkins, Dmitry Oleynikov, and Nick Stergiou. Objective evaluation of expert and novice performance during robotic surgical training tasks. Surgical endoscopy, 23(3):590, 2009.

[40] Jacob Rosen, Massimiliano Solazzo, Blake Hannaford, and Mika Sinanan. Objective laparoscopic skills assessments of surgical residents using hidden markov models based on haptic information and tool/tissue interactions. Studies in health technology and informatics, pages 417–423, 2001.

[41] Amod Jog, Brandon Itkowitz, May Liu, Simon DiMaio, Greg Hager, Myriam Curet, and Rajesh Kumar. Towards integrating task information in skills assessment for dexterous tasks in surgery and simulation. In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pages 5273–5278. IEEE, 2011.

[42] Henry C Lin, Izhak Shafran, David Yuh, and Gregory D Hager. Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions. Computer Aided Surgery, 11(5):220–230, 2006.

[43] Carol E Reiley, Henry C Lin, David D Yuh, and Gregory D Hager. Review of methods for objective surgical skill evaluation. Surgical endoscopy, 25(2):356–366, 2011.

[44] Lingling Tao, Ehsan Elhamifar, Sanjeev Khudanpur, Gregory D Hager, and René Vidal. Sparse hidden markov models for surgical gesture classification and skill evaluation. In Information Processing in Computer-Assisted Interventions, pages 167–177. Springer, 2012.

[45] Narges Ahmidi, Yixin Gao, Benjamín Béjar, S Swaroop Vedula, Sanjeev Khudanpur, René Vidal, and Gregory D Hager. String motif-based description of tool motion for detecting skill and gestures in robotic surgery. In International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 26–33. Springer, 2013.

[46] F Frenet. Sur les courbes à double courbure. Journal des mathématiques pures et appliquées, 17:437–447, 1852.

[47] Narges Ahmidi, Piyush Poddar, Jonathan D Jones, S Swaroop Vedula, Lisa Ishii, Gregory D Hager, and Masaru Ishii. Automated objective surgical skill assessment in the operating room from unstructured tool motion in septoplasty. International journal of computer assisted radiology and surgery, 10(6):981–991, 2015.

[48] Alasdair IM Macmillan and Alfred Cuschieri. Assessment of innate ability and skills for endoscopic manipulations by the advanced dundee endoscopic psychomotor tester: predictive and concurrent validity. The American journal of surgery, 177(3):274–277, 1999.

[49] K Moorthy, Y Munz, A Dosis, F Bello, A Chang, and A Darzi. Bimodal assessment of laparoscopic suturing skills. Surgical Endoscopy And Other Interventional Techniques, 18(11):1608–1612, 2004.

[50] Smita De, Jacob Rosen, Aylon Dagan, Blake Hannaford, Paul Swanson, and Mika Sinanan. Assessment of tissue damage due to mechanical stresses. International Journal of Robotics Research, 26(11-12):1159–1171, 2007.

[51] Damian D Marucci, Anthony J Shakeshaft, John A Cartmill, Michael R Cox, Stuart G Adams, and Christopher J Martin. Grasper trauma during laparoscopic cholecystectomy. Australian and New Zealand Journal of Surgery, 70(8):578–581, 2000.

[52] Jeffrey H Peters, GD Gibbons, JT Innes, KE Nichols, SR Roby, EC Ellison, et al. Complications of laparoscopic cholecystectomy. Surgery, 110(4):769–77, 1991.

[53] Panos Sakellariou, Athanasios G Protopapas, Zannis Voulgaris, Nikolaos Kyritsis, Alexandros Rodolakis, Georgios Vlachos, Emmanuel Diakomanolis, and Stylianos Michalas. Management of ureteric injuries during gynecological operations: 10 years experience. European Journal of Obstetrics & Gynecology and Reproductive Biology, 101(2):179–184, 2002.

[54] Charles Chapron, Denis Querleu, Maurice-Antoine Bruhat, Patrick Madelenat, Hervé Fernandez, Fabrice Pierre, and Jean-Bernard Dubuisson. Surgical complications of diagnostic and operative gynaecological laparoscopy: a series of 29,966 cases. Human Reproduction, 13(4):867–872, 1998.

[55] Scott R Steele, Justin A Maykel, Bradley J Champagne, and Guy R Orangio. Complexities in Colorectal Surgery. Springer, 2014.

[56] Allison M Okamura. Methods for haptic feedback in teleoperated robot-assisted surgery. Industrial Robot: An International Journal, 31(6):499–508, 2004.

[57] Y.C. Fung. Biomechanics. Mechanical Properties of Living Tissues. Springer Verlag, New York, 1981.

[58] Xiaolong Yu, Howard Jay Chizeck, and Blake Hannaford. Comparison of transient performance in the control of soft tissue grasping. In Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, pages 1809–1814. IEEE, 2007.

[59] Jacob Rosen, Blake Hannaford, Mark MacFarlane, and Mika Sinanan. Force controlled and teloperated endoscopic grasper for minimally invasive surgery - experimental performance evaluation. IEEE Transactions on BioMedical Engineering, 46:1212–1221, 1999.

[60] Jeffrey Brown, Jacob Rosen, Manuel Moreyra, Mika Sinanan, and Blake Hannaford. Computer-Controlled Motorized Endoscopic Grasper for In Vivo Measuremenet of Soft Tiissue Biomechanical Characteristics. IOS Press, 2002.

[61] Jacob Rosen, Jeffrey D Brown, Smita De, Mika Sinanan, and Blake Hannaford. Biomechanical properties of abdominal organs in vivo and postmortem under compression loads. Journal of biomechanical engineering, 130(2):021020, 2008.

[62] Marc Hollenstein, Alessandro Nava, Davide Valtorta, Jess G Snedeker, and Edo ardo Mazza. Mechanical characterization of the liver capsule and parenchyma. In Biomedical Simulation, pages 150–158. Springer, 2006.

[63] Tomonori Yamamoto, Balazs Vagvolgyi, Kamini Balaji, Louis L Whitcomb, and Allison M Okamura. Tissue property estimation and graphical display for teleoperated robot-assisted surgery. In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, pages 4239–4245. IEEE, 2009.

[64] Yangming Li and Blake Hannaford. Gaussian process regression for sensorless grip force estimation of cable-driven elongated surgical instruments. IEEE Robotics and Automation Letters, 2(3):1312–1319, 2017.

[65] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning. Springer, 2001.

[66] Jieping Ye, Ravi Janardan, and Qi Li. Two-dimensional linear discriminant analysis. In Advances in Neural Information Processing Systems, pages 1569–1576, 2004.

[67] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: A general framework for dimenstionality reduction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29:40–51, 2007.

[68] Jerome H Friedman. Regularized discriminant analysis. Journal of the American statistical association, 84(405):165–175, 1989.

[69] Santosh Srivastava, Maya Gupta, and Bela Frigyik. Bayesian quadratic discriminant analysis. Journal of Machine Learning Research, 8:1277–1305, 2007.

[70] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. The Annals of Mathematical Statistics, 27(3):832–837, 1956.

[71] Emanuel Parzen. On estimation of a probability density function and mode. The annals of mathematical statistics, pages 1065–1076, 1962.

[72] Simon J Sheather and Michael C Jones. A reliable data-based bandwidth selection method for kernel density estimation. Journal of the Royal Statistical Society, pages 683–690, 1991.

[73] David J Hand. Kernel discriminant analysis. Research studies press New York, 1982.

[74] S. Mika, G. Ratsch, J. Weston, B. Scholkopft, and KR Muller. Fisher discriminant analysis with kernels. In Proceedings of the 1999 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing IX, Madison, WI, USA, pages 23–25, 1999.

[75] Gaston Baudat and Fatiha Anouar. Generalized discriminant analysis using a kernel approach. Neural computation, 12(10):2385–2404, 2000.

[76] Deng Cai, Xiaofei He, and Jiawei Han. Efficient kernel discriminant analysis via spectral regression. In Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on, pages 427–432. IEEE, 2007.

[77] Hassan Amoud, Hichem Snoussi, David Hewson, Michel Dousset, and Jacques Duchene. Intrinsic mode entropy for nonlinear discriminant analysis. IEEE Signal Processing Letters, 14(5):297–300, 2007.

[78] Madalena Costa, Ary Goldberger, and C.K. Peng. Multiscale entropy analysis of complex physiologic time series. Physical Review Letters, 89(6):1–4, 2002.

[79] Ran He, Bao-Gang Hu, and Xiao-Tong Yuan. Robust discriminant analysis based on nonparametric maximum entropy. Asian Conference on Machine Learning, pages 120–134, 2009.

[80] Steven M Pincus. Approximate entropy as a measure of system complexity. Proceedings of the National Academy of Sciences, 88(6):2297–2301, 1991.

[81] Joshua S Richman and J. Randall Moorman. Physiological time-series analysis using approximate entropy and sample entropy. American Journal of Physiology-Heart and Circulatory Physiology, 278(6):H2039–H2049, 2000.

[82] Igor Kononenko, Edvard Šimec, and Marko Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with RELIEFF. Applied Intelligence, 7(1):39–55, 1997.

[83] Christopher M Bishop. Pattern recognition and machine learning, volume 4. Springer New York, 2006.

[84] Charles R Henderson. Best linear unbiased estimation and prediction under a selection model. Biometrics, pages 423–447, 1975.

[85] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. The annals of mathematical statistics, pages 1554–1563, 1966.

[86] Carol E Reiley and Gregory D Hager. Task versus subtask surgical skill evaluation of robotic minimally invasive surgery. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009, pages 435–442. Springer, 2009.

[87] Jacob Rosen, Blake Hannaford, Christina G Richards, and Mika N Sinanan. Markov modeling of minimally invasive surgery based on tool/tissue interaction and force/torque signatures for evaluating surgical skills. IEEE Transactions on Biomedical Engineering, 48(5):579–591, 2001.

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

[89] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. Journal of the royal statistical society. Series B (methodological), pages 1–38, 1977.

[90] Herve Bourlard and Nelson Morgan. Hybrid hmm/ann systems for speech recognition: Overview and new research directions. In Adaptive Processing of Sequences and Data Structures, pages 389–417. Springer, 1998.

[91] Zhong-Hua Quan and Kun-Hong Liu. Online signature verification based on the hybrid hmm/ann model. International Journal of Computer Science and Network Security, 7(3):313–322, 2007.

[92] Lalit Bahl, Peter Brown, Peter V de Souza, and Robert Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86., volume 11, pages 49–52. IEEE, 1986.

[93] Biing-Hwang Juang and Shigeru Katagiri. Discriminative learning for minimum error classification [pattern recognition]. Signal Processing, IEEE Transactions on, 40(12):3043–3054, 1992.

[94] Nicholas Metropolis and S Ulam. The monte carlo method. Journal of the American Statistical Association, 44:335–341, 1949.

[95] N.J. Gordon, D.J. Salmond, and A.F.M Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In IEE Proceeding of Radar and Signal Processing, 1993.

[96] Arnaud Doucet, Neil Gordon, and Vikram Krishnamurthy. Particle filters for state estimation of jump markov linear systems. IEEE Transactions on Signal Processing, 49(3):613–624, 2001.

[97] Christophe Andrieu, Arnaud Doucet, Sumeetpal Singh, and Vladislav Tadic. Particle method for change detection, system identification, and control. Proceedings of the IEEE, 92(3):423–438, 2004.

[98] Fredrik Gustafsson and Paul Hriljac. Particle filters for system identification with application to chaos prediction. Proceedings of SYSID03, 2003.

[99] George Poyiadjis, Arnaud Doucet, and Sumeetpal S Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. Biometrika, 98(1):65–80, 2011.

[100] T. Schon, Thomas B., Adrian Wills, and Brett Ninness. System identification of nonlinear state-space models. Automatica, 47(1):39–49, 2011.

[101] Benson Limketkai, Dieter Fox, and Lin Liao. Crf-filters: Discriminative particle filters for sequential state estimation. In 2007 IEEE International Conference on Robotics and Automation, pages 3142–3147. IEEE, 2007.

[102] Leo Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

[103] J. R. Quinlan. Induction of decision trees. Mach. Learn., 1(1):81–106, 1986.

[104] Wayne Iba and Pat Langley. Induction of one-level decision trees. In Proceedings of the Ninth International Conference on Machine Learning, pages 233–240, 1992.

[105] Martin Fodslette Moller. A scaled conjugate gradient algorithm for fast supervised learning. Neural networks, 6(4):525–533, 1993.

[106] Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the marquardt algorithm. Neural Networks, IEEE Transactions on, 5(6):989–993, 1994.

[107] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. Neural networks, 6(6):801–806, 1993.

[108] Carl Edward Rasmussen and Christopher KI Williams. Gaussian processes for machine learning. 2006. The MIT Press, Cambridge, MA, USA, 38:715–719, 2006.

[109] Timothy M Kowalewski, Lee W White, Thomas S Lendvay, Iris S Jiang, Robert Sweet, Andrew Wright, Blake Hannaford, and Mika N Sinanan. Beyond task time: automated measurement augments fundamentals of laparoscopic skills methodology. Journal of Surgical Research, 192(2):329–338, 2014.

[110] John J ONeill, Timothy M Kowalewski, and Robert M Sweet. Feasibility of online semantic labeling of deformable tissues for minimally invasive surgery. Journal of Medical Devices, 9(3):030938, 2015.

[111] Carolyn Chen, Lee White, Timothy Kowalewski, Rajesh Aggarwal, Chris Lintott, Bryan Comstock, Katie Kuksenok, Cecilia Aragon, Daniel Holst, and Thomas Lendvay. Crowd sourced assessment of technical skills: a novel method to evaluate surgical performance. Journal of Surgical Research, 187(1):65–71, 2014.

Appendix A

Glossary and Acronyms

Care has been taken in this thesis to minimize the use of jargon and acronyms, but this cannot always be achieved. This appendix defines jargon terms in a glossary, and contains a table of acronyms and their meaning.

A.1 Glossary

- **Supervised Learning:** Machine learning algorithms in which the training data set is class labeled.
- **Unsupervised Learning:** Machine learning algorithms in which the training data set class is unlabeled.
- **Features:** An informative or combined representation of data within a machine learning application (e.g. x and y are typical features extracted from z).
- **Prior:** The likely probability distribution about an uncertain quantity before an observation is made.
- **Posterior:** The likely probability distribution about an uncertain quantity after an observation is made.
- **Generative Model:** A classification approach in which training data is used to generate a system model by explicitly modeling the probability distribution of the inputs and outputs. This is the inference step of machine learning. In other words, given a class what is the data. This comes directly from Bayes theorem: $p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$.

- **Discriminative Model:** A classification approach in which training data is learned by modeling the conditional probability. This is the decision step of machine learning. In other words, given some data, what is the class.

- **Discriminant Model:** A classification approach in which the inference and decision steps are lumped together. Therefore classification is performed directly by mapping some input x directly to a class label.

- **Entropy:** A measure of the uncertainty about a particular data set. This value is larger for more random data sets.

- **NN:** Artificial Neural Networks.

- **RF:** Random Forests.

- **FLS:** Fundamentals of Laparoscopic Surgery.

- **DLS:** Discriminant Least Squares, one of the proposed algorithms.

- **DPP:** Discriminant Phase Portrait, one of the proposed algorithms.

- **RELIEF-RBF:** RELIEFF w/ Radial Basis Functions, one of the proposed algorithms.

- **Intent Vector:** A proposed feature derived from the overall trajectory of a surgical tool.

Appendix B

Biosketch

Rod received his Ph.D. in Mechanical Engineering from the University of Minnesota in May 2017. He received his B.A. in Physics and Mathematics from St. Olaf College in 2011 and while there completed an undergraduate research project on an inverted pendulum robot. During the summer of 2011, he had an internship at Crossing Automation, a small automation company in Fremont, CA. While at Crossing, he was exposed to the design and manufacturing of custom fabrication devices for use in silicon wafer sorting and clean room inspection facilities. He had the opportunity to develop a custom wafer sorting algorithm which obviated the need for more expensive hardware and instead utilized a circular hough transform and optical sensors. He also performed testing and analysis of this algorithm in order to compare the accuracy with the current modality. This testing involved using a high speed camera to gauge the accuracy of the wafer placement.

Rod entered graduate school at the University of Minnesota in the fall of 2011. He initially worked in the HumanFirst lab under Dr. Max Donath where he developed custom driving simulations for use in collaborative research with a human factors research group. These driving simulations were then utilized in studies with a specially designed vehicle simulator and virtual reality set-up.

In his second year of graduate school, Rod joined the Medical Robotics and Devices lab as a founding student member under Dr. Tim Kowalewski. He initially began work on a research project involving surgical skill evaluation in conjunction with the new product design and business development course. This project was charged with the development of the Fundamentals of Robotic Surgery (FRS) psycho-motor skill evaluation. During this project Rod lead the teams development of circuit design, sensing and embedded programming in order to develop an objective evaluation platform for robotic surgery. This platform was ultimately presented to a national consortium of robotic surgeons.

For his masters plan A project, Rod developed a computer vision algorithm for the 3D tracking of surgical robotic tools using endoscopic stereo cameras. This algorithm utilized relatively simple computer vision techniques in order to greatly improve the framerate compared with prior art (26 hz). This work was presented at the 2014 IEEE Conference on Intelligent Robots and Systems. Rod graduated with his M.S.M.E in the summer of 2013.

During 2013 Rod continued his ongoing work at a Minnesota based startup; Tarsier Inc.

At Tarsier, he was responsible for developing computer vision algorithms for use in a consumer electronics product centered on intuitive human-computer interaction. This device allowed users to manipulate objects on the screen using only their hands, non-relative to their position in the room. Rod was also responsible for designing custom graphical user interfaces centered on a new 3D paradigm. This product was debuted at the Consumer Electronics Show and has since been presented to several large consumer electronics companies.

In 2015 Rod received the University of Minnesota Informatics Institute (UMII) Graduate Fellowship under Minnesota's Discovery, Research, and InnoVation Economy (MnDRIVE) facility. In 2016 Rod received the University of Minnesota Interdisciplinary Doctoral Fellowship (IDF) with the Medical Devices Center serving as the host interdisciplinary research center. Rod has co-authored multiple conference and journal publications appearing in the ASME Journal of Medical Devices, the Journal of Endourology, IEEE International Conference on Robotics and Automation (ICRA), IEEE Conference on Intelligent Robots and Systems (IROS), and the International Journal of Computer Assisted Radiology and Surgery.