Human-robot teams are invaluable for mapping unknown environments, exploring difficult-to-reach areas, and manipulating inaccessible equipment. However, guiding autonomous robots requires dealing with these dynamic domains while synthesizing a significant amount of data and balancing competing objectives. Current mission planning methods often involve manually specifying low-level parameters of the mission, such as exact waypoints or control inputs. These methods cannot perfectly cope with the changing surroundings and limited communications that come with operating in these complex conditions. To address this and reduce the burden on human operators, the field has trended towards ever-increasing levels of autonomy. Providing this long-term autonomy requires more usable, robust collaborative mission planning solutions that leverage the strengths of both the robot and the human operator.

In this thesis, we propose two novel methods for improving the collaboration of human-robot teams by enabling the robot to learn an operator’s preferences for mission planning. These techniques provide the robot with a rich representation of the human’s goals while utilizing familiar techniques to speed learning. The first method is trained by making small-scale, iterative improvements to candidate mission plans generated by the robot, similar to
the small improvements an operator would make while planning an actual mission. Using a novel coactive learning algorithm, the method learns the operator’s preferences from the feature differences between the original and improved mission plans while remaining robust to errors and noise in the operator’s corrections.

The second proposed method simplifies the queries by asking survey-style rating and ranking questions about candidate plans. These queries are generated by a Gaussian process (GP) active learner that uses the responses to learn the most preferred region of the mission preference space. The ranking query responses provide the GP with general relational information about several points in the preference space, while the rating query responses provide a specific preference about a single point. A custom probit allows the GP to incorporate the different strengths of each query type into a single preference model.

Tests in simulated lake monitoring missions show that these methods can efficiently and accurately learn an operator’s preferences. Additionally, a field trial in which an EcoMapper autonomous underwater vehicle monitors the ecology of a lake validates the use of the coactive learning method. These results demonstrate that these techniques can enable a robot to accurately learn a human operator’s preferences, then autonomously plan and perform missions that apply those preferences without relying on regular intervention by the operator.
Efficiently Learning Human Preferences for Robot Autonomy

by
Thane Somers

A THESIS
submitted to
Oregon State University

in partial fulfillment of
the requirements for the
degree of
Master of Science

Presented June 1, 2022
Commencement June 2022
Master of Science thesis of Thane Somers presented on June 1, 2022.

APPROVED:

________________________________________________________________________
Major Professor, representing Robotics

________________________________________________________________________
Head of the School of Mechanical, Industrial, and Manufacturing Engineering

________________________________________________________________________
Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

________________________________________________________________________
Thane Somers, Author
I would like to extend my gratitude to the many people who have helped me throughout graduate school and during the preparation of this thesis. In particular, I’m grateful to my advisor Dr. Geoff Hollinger for his unwavering patience, thoughtful advice, and encouragement. Thanks for sticking with me.

Graduate school would not have been the same without Seth McCammon, Nick Lawrance, Jen Jen Chung, Jeff Caley, Dylan Jones, Lucas Hill, and all of my friends in the Robotics Decision Making Laboratory and Robotics Graduate School. You are all kind, brilliant, and very much appreciated.

I would also like to thank all of the members of my committee—Dr. Bill Smart, Dr. Julie Adams, Dr. Alan Fern, and Dr. Bogdan Strimbu—for their support and guidance throughout the years. Thanks also go to Robby Goetschalckx and Prasad Tadepalli for their insightful comments on the technical details of this work and to Gaurav Sukhatme from the University of Southern California for providing access to the EcoMapper vehicle used in the field experiments.

A big thank-you to my family, Doug, Sharon, and Lydia, for all they’ve done to get me to and through this process, for their passion for learning, and for always believing in me.

Lastly, this would not have been possible without the support and care of my wife, Rachel. Thanks for going along with me on this adventure!

Of course, this research would not have happened without the funding provided by NSF grant IIS-1317815 and Office of Naval Research grant N00014-14-1-0905.
CONTRIBUTION OF AUTHORS

Dr. Nicholas Lawrance collaborated on the design, implementation, and testing of the model for the combined rating and ranking Gaussian process in Chapter 4.
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Background</td>
</tr>
<tr>
<td>2.1</td>
<td>Aquatic Robotics</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Autonomous Underwater Vehicles (AUVs)</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Remotely Operated Underwater Vehicles (ROVs)</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Autonomous Surface Vehicles (ASVs)</td>
</tr>
<tr>
<td>2.2</td>
<td>Related Work</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Planning Under Uncertainty</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Learning from Demonstration</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Preference Modeling</td>
</tr>
<tr>
<td>3</td>
<td>Human-Robot Planning and Learning for Marine Data Collection</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>3.2</td>
<td>Human-Robot Architecture</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Trajectory Refinement Algorithms</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Coactive Learning Algorithm</td>
</tr>
<tr>
<td>3.3</td>
<td>Simulations and Results</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Comparative Learning Algorithm Simulations</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Data-Driven Trajectory Refinement Simulations</td>
</tr>
<tr>
<td>3.4</td>
<td>Field Trials</td>
</tr>
<tr>
<td>3.5</td>
<td>Chapter Summary</td>
</tr>
<tr>
<td>4</td>
<td>Efficient Learning of Trajectory Preferences Using Combined Ratings and Rankings</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>4.2</td>
<td>Related Work</td>
</tr>
<tr>
<td>4.3</td>
<td>Methods</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Problem formulation</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Objective Space Gaussian Process Learning</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Active Selection of Ratings and Rankings</td>
</tr>
<tr>
<td>4.4</td>
<td>Experiments and Results</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Randomized trials for active learning</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Simulated Lake Monitoring Trials</td>
</tr>
</tbody>
</table>
4.5 Chapter Summary ......................................................... 62

5 Conclusions ................................................................. 63
5.1 Future Directions ....................................................... 64

Bibliography ................................................................. 67
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Commanding aquatic robots, such as the pictured Platypus Lutra autonomous</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>surface vehicle, is currently a slow, hands-on process where operators</td>
<td></td>
</tr>
<tr>
<td></td>
<td>manually set individual waypoints at the outset of the mission.</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>Mean GP estimation error and weighted RMS error across 20 trials for the</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>simulated information-gathering user.</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Proposed framework for human-robot collaboration to generate safe and</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>informative paths for autonomous vehicles to gather scientific data.</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>Three maps that must be combined to perform efficient information gathering</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>in an ocean monitoring scenario where an AUV is tracking a harmful algal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>bloom.</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>An example path and utility field generated using the coactive learning</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>algorithm simulations.</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>The regret (sum of deviation between target weight and estimated weight)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>for each coactive algorithm averaged over 20 trials.</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>Example plots of how the estimated weight changes over a trial in simulations</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>of the perceptron and histogram learning algorithms.</td>
<td></td>
</tr>
<tr>
<td>3.6</td>
<td>Example of improving an initial trajectory of waypoints using the proposed</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>learning and planning framework for data from April 24, 2013.</td>
<td></td>
</tr>
<tr>
<td>3.7</td>
<td>Initial trajectory and set of suggested trajectories using different</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>weightings between deviation and risk.</td>
<td></td>
</tr>
<tr>
<td>3.8</td>
<td>Deviation and risk for varying weighting parameters for trials of the</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>trajectory optimizer using data from April 24, 2013.</td>
<td></td>
</tr>
<tr>
<td>3.9</td>
<td>An aerial view of the test area at Puddingstone Reservoir with the</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>corresponding depth map.</td>
<td></td>
</tr>
<tr>
<td>3.10</td>
<td>A comparison of framework and human planned trajectories over the</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>corresponding utility maps of Puddingstone Reservoir.</td>
<td></td>
</tr>
<tr>
<td>3.11</td>
<td>A contour-following path generated using our coactive learning algorithm.</td>
<td>40</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>A trajectory $T$ is mapped to a corresponding objective-space feature vector $\mathbf{x}_T$ (with feature dimensions of path length and accumulated risk).</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>A graphical model illustration of the Gaussian process problem formulation.</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>True latent function, associated likelihoods and samples.</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>Posterior estimate with five absolute samples and five pairwise relative samples.</td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>Results for simulated randomized wave function problem.</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>Two example lake monitoring trajectories from the trajectory pool are shown superimposed over a simulated objective field representing the information gained by traveling over each part of the lake.</td>
<td></td>
</tr>
<tr>
<td>4.7</td>
<td>Trajectory ratings for the simulated user plotted over the lake monitoring mission objective space.</td>
<td></td>
</tr>
<tr>
<td>4.8</td>
<td>Mean GP estimation error and weighted RMS error across 20 trials for the simulated information-gathering user.</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

Collaborative human-robot teams are becoming increasingly prevalent in unstructured environments. Such teams combine the ability of robots to collect and synthesize large amounts of data with the specialized knowledge and depth of experience of human operators. Significant applications of this approach include tracking oil spills [1], mapping archaeological sites [2], and providing situational awareness during natural disasters [3]. However, improving the autonomy of the robots and decreasing the burden on the humans in these teams remain key challenges. Robots operating in complex domains such as urban environments, aquatic ecosystems, and disaster areas require constant monitoring and regular human intervention. Improving the long-term autonomy of these systems requires developing faster and more robust collaborative methods.

In this thesis, we describe our approach to learning human preferences for robotic mission planning. This research is motivated by the challenges robots face when operating in extreme environments; missions involving aquatic robots, for instance, face dynamic conditions, limited communications, and long mission durations. Operating in and adapting to these environments requires the robot have a rich understanding of the human operator’s mission planning preferences. In particular, when planning missions as part of a human-robot team, the robot’s goals must align with those of the human in order to reach true autonomy. In order to plan, the robot needs to first understand the trade-offs between its possible actions and then understand how best to balance those trade-offs in order to meet the goals of the team. Without this understanding, the robot will be unable to autonomously plan and replan without human intervention.
Figure 1.1: Commanding aquatic robots, such as the pictured Platypus Lutra autonomous surface vehicle, is currently a slow, hands-on process where operators manually set individual waypoints at the outset of the mission. This burden motivates the need for algorithms that can quickly learn and meet a human operator’s goals while adapting to the dynamic aquatic environment.

We attempt to address these challenges by providing a set of preference-learning algorithms that are capable of efficiently learning human preferences across a wide range of domains. By leveraging simple, quick queries to learn non-linear, distribution-based preference models, our proposed algorithms allow a robot to gain an understanding of a human’s goals for a mission. In simulations, user studies, and field trials, we demonstrate that this preference model allows the robot to autonomously plan and execute missions that meet the goals of the human without the need for continuous human involvement. In doing so, we hope to improve the ease-of-use, capabilities, and longevity of robots operating in challenging environments.

This thesis is organized as a collection of two published contributions with accompanying
background information: Chapter 2 presents relevant background and related work on aquatic robots, learning from demonstration, and planning under uncertainty. The first method is presented in Chapter 3, which describes our histogram-based coactive learning algorithm (originally published in Somers and Hollinger [4]). This algorithm learns from an expert’s iterative improvements to example mission trajectories. These small improvements are easier to make than complete demonstrations, yet still provide a good understanding of the expert’s preferences. By forming a distribution over the improvements, our algorithm is robust to noise and sub-optimal updates made by the human.

Chapter 4 details our Gaussian process (GP) preference learning algorithm, originally published in Somers, Lawrance, and Hollinger [5]. This algorithm builds upon the previous by learning from a combination of actively-selected, survey-style rating and ranking queries that are faster and easier for the human to answer. Additionally, by modeling the human’s preferences with a GP, the algorithm is resistant to noise and allows for non-linear preference models, improving our approach’s flexibility and learning speed. Finally, Chapter 5 summarizes the major themes of the work and discusses directions for future study.
Chapter 2: Background

This chapter presents relevant background information and related prior work. It first describes the methods and challenges associated with operating robots in aquatic environments. Addressing these challenges through increased autonomy motivates the research presented in this thesis. Then, it discusses prior work in the areas of planning under uncertainty, learning from demonstration, and preference modelling.

2.1 Aquatic Robotics

Aquatic robots are used in lakes, rivers, and marine environments for purposes such as inspecting deep-sea installations, performing maintenance on equipment, and tracking underwater life for oceanographic research. They can undertake long-duration missions and access areas and depths that would be difficult or impossible for human divers. They can be equipped with a variety of sensors, including sonar imagers, cameras, and chemical sensors, and with manipulators to characterize these environments, collect samples, and manipulate equipment. However, the robots operating in these environments face numerous challenges that prevent the use of sensors and algorithms that are commonly used in other domains. Water currents are highly variable and difficult to predict, complicating navigation and obstacle avoidance. Additionally, water blocks all but extra-low frequency radio waves, obviating the use of localization methods like GPS and requiring slow, expensive methods for point-to-point communication [6]. Finally, water is corrosive, damaging to electronics, and creates large hydrostatic pressures that increase rapidly with depth.
Aquatic robots are usually divided into three broad categories based on their design and intended use. The following sections briefly describe the features of each, with examples shown in Figure 2.1.

2.1.1 Autonomous Underwater Vehicles (AUVs)

All untethered, underwater robots fall under the category of AUVs (Figure 2.1a). AUVs are best suited to long-term (>1 hour) surveying and monitoring missions over large bodies of water but are able to monitor underwater phenomena over the entire water column, from the surface to the sea floor. They are driven either by propellers or by changing their density.
to glide through the water on wings. These buoyancy-driven AUVs can have endurances of more than a year. Because of communications restrictions and long deployment durations, these vehicles spend much of their missions acting without human intervention.

2.1.2 Remotely Operated Underwater Vehicles (ROVs)

ROVs encompass all tethered underwater vehicles (Figure 2.1b) and are primarily used to perform short range inspection and intervention missions. Connecting to a base station via a long copper or fiber-optic tether gives ROVs a source of power and high-bandwidth communications but introduces extra mechanical complexity, limits their range of motion, and increases the chance of snagging on debris. Most are equipped with electric thrusters for movement, cameras and sonars for sensing, manipulators, and a variety of acoustic positioning systems to allow them to localize and station-keep in currents. This makes them well-suited to manipulating and inspecting underwater objects in real time.

2.1.3 Autonomous Surface Vehicles (ASVs)

ASVs take the form of boats or other vehicles that float on the surface of the water (Figure 2.1c). They are used in a variety of missions, from short deployments supporting divers or other underwater robots to extended duration cruises that collect environmental data over a large area. While they are not subject to the same power or communications restrictions as other aquatic robots, they must still deal with changing winds, currents, and unexpected obstacles. Their use is also more limited, as they can only sense what is within range near the water’s surface.
2.2 Related Work

2.2.1 Planning Under Uncertainty

Motion and path planning are fundamental problems in robotics and have been studied extensively in the past two decades [7, 8]. Increasing the robot’s degrees of freedom or the dimensionality of the environment typically causes an exponential increase in the computation required to solve the planning problem optimally. Thus, motion and path planning problems are generally computationally hard (NP-hard or PSPACE-hard) [9]. Modern planning methods have focused on the generation of approximate plans with limited computation (e.g., RRT* algorithms [10]). Our work extends these ideas to domains where human-robot collaboration is beneficial for informing the generation of high-quality plans.

This research also relates to work in adaptive sampling algorithms. These algorithms attempt to choose path goals that maximize information gain, minimize prediction uncertainty, or minimize risk [11, 12] while minimizing the cost of the tour. Methods for doing this adaptively include the maximum entropy and maximum mutual information measures, used for planning in environments modeled as Gaussian Processes [13]. In [14], Hoang et al. propose a non-myopic active learning framework using Gaussian Processes that jointly optimizes the exploration-exploitation trade-off. In these algorithms, there is an implicit trade-off between the various goals. Our work allows the robot to easily learn those trade-offs from human users, particularly with respect to the importance of each environmental feature. These adaptive algorithms then closely match the performance expected by the human operator while operating autonomously in the field.
2.2.2 Learning from Demonstration

Learning from Demonstration (LfD) techniques were developed based on the premise that a human’s reward function over the robot’s state and the features of the environment often contain all of the knowledge needed to successfully complete a task. However, it is often difficult for a human to fully enumerate their reward structure in a useful manner. Thus, techniques have been developed to learn these reward preference from human-controlled demonstrations of intended behavior.

Much of the previous work on solving the problem of enabling robots to learn from human demonstration has focused on finding ways for the robot to effectively mimic the human. Researchers have studied a variety of problems such as planning driving trajectories [15] and autonomous helicopter flight [16]. However, most learning from demonstration problems assume that the expert is providing optimal feedback, which is often impossible to achieve. For example, when solving informative path planning problems [17, 18], humans cannot easily find the optimal path, but they can quickly choose which paths they prefer. In our work, we account for this limitation on human performance by allowing the human to merely present a preference for a solution. The optimal solution is never needed.

Learning from demonstration (LfD) methods have been used to learn an operator’s preferences from a set of human-driven demonstrations of near-optimal robot behavior [19]. The most well known LfD algorithm is inverse reinforcement learning (IRL) [20]. IRL assumes that the human is acting as a Markov decision process and is providing optimal demonstrations, then attempts to find the reward function that matches the policy presented in the user’s demonstrations. These reward and policy functions are parameterized by features of the robot’s environment.

A second common LfD method is called coactive learning. In coactive learning, the
algorithm presents its best guess at a solution to the expert, who then makes improvements to the solution. The algorithm then updates its estimation of the expert’s reward function based on the changes in the features of the improved solution. Several forms of coactive learning algorithms with theoretical bounds have been studied. These include regret bounds on the perceptron coactive learning algorithm [21] and cost bounds on the cost-sensitive perceptron and passive-aggressive algorithms [22]. However, these bounds still assume optimal or locally optimal feedback and have not been tested directly with human experts.

Much of the work done on coactive learning algorithms has studied problems where both the expert and the learner are computer programs which solve and improve the solution using different methods [22, 21]. A few studies of using the coactive learning algorithm online with humans have been conducted, most notably in learning trajectories for robotic arms [23]. They show that the robot can successfully learn from the human’s iterative, suboptimal improvements and that the coactive algorithm performs better than other learning algorithms. Our algorithm builds upon these ideas by providing increased resistance to human error and shorter learning times.

Our work also builds on prior studies in imitation learning that construct cost maps from human-operator example paths [15, 24]. These prior studies utilize maximum-margin planning techniques to learn cost maps from user input. Prior work has focused on surveillance problems where the goal is to generate a safe path through a hazardous environment. Our proposed framework expands on these ideas by incorporating the human scientist into the planning loop.
2.2.3 Preference Modeling

In our proposed methods, we model the expert’s preference function as either a weighting of mission objectives or as a Gaussian process in the mission objective space. Much of the prior work on objective spaces has studied multi-objective optimization methods in which an algorithm attempts to learn the optimal trade-offs, known as a Pareto front, between a set of objectives. Several methods for estimating the Pareto front have been studied. These variations include assuming that the optimal trade-offs are a linear weighting of the objectives [25], using evolutionary algorithms to find non-linear fronts [26, 27], and linearizing the objective space in order to learn non-convex fronts [28]. While our method also attempts to find a user’s trade-offs between multiple mission objectives, by learning a Gaussian process over the objective space we learn a more general representation of the user’s preferences as compared to a single Pareto front. This generalized representation allows a robot to continue to operate optimally, even in dynamic conditions where the user’s original Pareto front may no longer be attainable.
Chapter 3: Human-Robot Planning and Learning for Marine Data Collection

In this chapter, we propose an integrated learning and planning framework that leverages knowledge from a human user, along with prior information about the environment, to generate trajectories for scientific data collection in marine environments. The proposed framework combines principles from probabilistic planning with nonparametric uncertainty modeling to refine trajectories for execution by autonomous vehicles. These trajectories are informed by a utility function learned from the human operator’s implicit preferences using a modified coactive learning algorithm. The resulting techniques allow for user-specified trajectories to be modified for reduced risk of collision and increased reliability. We test our approach in two marine monitoring domains and show that the proposed framework mimics human-planned trajectories while reducing the risk of operation. This work provides insight into the tools necessary for combining human input with vehicle navigation to facilitate persistent autonomy. The material in this chapter was originally published in Somers and Hollinger [4].

3.1 Introduction

We envision the future of scientific data collection as a collaborative endeavor between human scientists and autonomous robotic systems. High-impact examples include autonomous underwater vehicles assisting oceanographers to track biological phenomena [29], aerial
vehicles providing imagery of changing ecosystems [30], and ground vehicles monitoring volcanic activity [31]. In these example domains, a key challenge lies in combining the expert knowledge of the scientist with the optimization capabilities of the autonomous system. The scientist brings specialized knowledge and experience to the table, while the autonomous system is capable of processing and evaluating large quantities of data. Leveraging these complementary strengths requires the development of collaborative systems capable of guiding long-term scientific data collection.

When robotic vehicles collaborate with humans, true autonomy relies on the robot having a clear understanding of the goals it uses and the trade-offs it faces when making decisions. When a robot is assisting a human, the robot’s goals must often mimic those of the human. This is particularly true in planning trajectories for underwater robots performing scientific monitoring. The robot must autonomously navigate the environment while maintaining the same goals as a human scientist. While propeller- and buoyancy-driven autonomous underwater vehicles (AUVs) are able to operate in aquatic environments for long time scales, true persistent autonomy requires them to be able to plan and replan their trajectories to fulfill the mission needs of the scientist without human intervention.

When planning trajectories for marine robotic missions, a scientist implicitly balances several environmental variables such as the risk of collision, the uncertainty in ocean currents, and the locations of points of interest. While existing planning algorithms can account for all of these variables, it is difficult to learn the correct trade-offs among them [24]. In this work, we create a framework that allows an AUV to learn a human path planner’s weighting of the variables involved in choosing a trajectory. The AUV then uses that weighting to plan paths mimicking those planned by the human. In this way, we can create an autonomous system that generalizes to different problems while still capturing the scientist’s expert knowledge and experience.
The main novelty of this chapter lies in its unified planning and learning framework that generates reliable and safe trajectories for autonomous marine vehicles based on human input. To our knowledge, this is the first work to combine Bayesian learning, probabilistic path planning, and user-generated trajectories in a unified approach. We present a novel algorithm that accounts for the variability in the quality of input provided by the human. In addition to simulations in ocean monitoring domains, we perform field trials on an EcoMapper AUV operating in an inland lake to demonstrate that our framework is robust and efficient enough for a human expert to easily use in real time.

These results demonstrate the advantage of combining uncertainty models with human preference learning for long-term marine monitoring missions. Our methods allow the robot to autonomously plan safe trajectories that meet a human operator’s personal goals without further human intervention. With these techniques, a robot and scientist are able to function as a team through a shared autonomy framework. Using shared autonomy reduces the burden on the human operators, allowing complex, long-term monitoring missions without the need for continuous human involvement. We believe that solving this challenge is key in making long-term autonomy achievable.

3.2 Human-Robot Architecture

The workflow of the proposed architecture is as follows: (1) a human operator specifies a series of waypoints for a vehicle to gather scientific data, (2) the waypoints are refined by the system to suggest alternative trajectories that have lower risk of collision, and (3) the human operator chooses the desired path. Figure 3.1 gives a visualization of the proposed architecture.

The trajectory refinements that the system suggests are based on an estimate of the
operator’s importance weighting of the environmental features that are used to determine the operator’s planned trajectory. This importance weighting allows the system to suggest trajectories that match the operator’s desired goals more closely. Using a coactive learning algorithm, our system learns these operator preferences based on a set of improvements to simulated environmental maps. These learned preferences are then used to inform the trajectory refinement algorithm of the type of trajectories desired.

The problems that we will examine under this framework comprise of the following components: a trajectory of scientist-provided waypoints that indirectly specify the quality of information gathered, a “risk” map that gives the expected safety of operating in a particular area, and a model of the environment that determines how reliably the autonomous vehicle can move between points. Figure 3.2 gives an example of the necessary maps in an oceanographic monitoring domain where an autonomous underwater vehicle is surveying a
Figure 3.2: Three maps that must be combined to perform efficient information gathering in an ocean monitoring scenario where an AUV is tracking a harmful algal bloom. The ocean currents affect the planned path of the vehicle, the risk map determines the safety of operation, and the pre-specified waypoints that the scientist provides gives the benefit of the information gathered. Our proposed system integrates these maps to improve scientific data collection.

number of ecological hotspots (e.g., harmful algal blooms [32]). The waypoints specified by the scientist provide areas of high algal bloom density, the risk map provides the probability of colliding with a ship or running into land, and the prevailing ocean currents provide the reliability of operation. We note that all of these maps are uncertain in the sense that the quantities are not known exactly at the time of planning. For reliable operation, it is necessary to predict the values of the information, risk, and reliability maps, as well as the uncertainty in those values.

3.2.1 Trajectory Refinement Algorithms

We first discuss our proposed approach for incorporating input from a human scientist into an optimization framework. We will build on ideas originally presented in prior work to learn cost maps that guide autonomous ground reconnaissance vehicles [15]. Unlike prior work,
our methods will incorporate a measure of risk into the planning and learning framework. We will also incorporate measures of uncertainty into our predictions.

The formal problem is to plan a path $\xi$ that is a solution to the following optimization problem:

$$
\xi^* = \arg\min_{\xi \in \Psi} R(\xi) + \alpha D(\xi, \xi_0),
$$

where $D(\xi, \xi_0)$ is the deviation from the scientist’s initial trajectory of waypoints $\xi_0$, $R(\xi)$ is an expected risk of executing $\xi$, $\Psi$ the space of all possible paths, and $\alpha$ is a weighting parameter. We assume that we are given an example trajectory of waypoints $\xi_0$ from the human operator and that an explicit cost function is not provided as part of the trajectory. The $\alpha$ parameter that matches the operator’s preferences is determined using the coactive learning procedure described in Section 3.2.2. After finding approximate solutions to the above optimization problem, the resulting trajectory is presented to the operator for final evaluation. The operator then has the option to adjust the values of $\alpha$ to make the trajectories deviate more or less from the initial plan.

There are several properties of the above problem that make it difficult to solve optimally. If the risk function is non-convex, optimizing it will typically be NP-hard for any rich space of paths [8]. In addition to the inherent complexity of the path planning problem, the functions $D$ and $R$ may be computationally intensive to compute. Furthermore, the deviation and risk functions may not be definitively known in advance (e.g., risk is only known with some certainty), and it may become necessary to estimate their expected values based on a distribution of possibilities. Similarly, for a given path, it may not be certain that the vehicle can execute the path exactly, which adds an additional level of uncertainty to the optimization.

Successfully addressing these challenges and optimizing the vehicle’s paths requires the
development of both uncertainty modeling and planning solutions. We will now describe how the proposed architecture addresses each of these subproblems.

3.2.1.1 Modeling Uncertainty

A key component of our proposed work is to provide a principled estimate of uncertainty for predictions of the vehicle’s actions. These uncertainty estimates will be incorporated through probabilistic planning to provide the final suggested paths for the vehicle. We propose using non-parametric Bayesian Regression in the form of Gaussian Processes (GPs) to provide measures of uncertainty [33]. We will now discuss background in GPs and show how we can use similar ideas to develop novel representations of uncertainty. This formulation closely follows our prior work in uncertainty modeling for ocean currents [34]. For this uncertainty modeling approach, we assume that estimates of the disturbances (such as ocean currents) are available (e.g., through regional ocean modeling systems and satellite data [35]).

The disturbances at a given latitude \(\text{lat}\), longitude \(\text{lon}\), and time \(t\) can be written as a tuple \(c(\text{lat}, \text{lon}, t) = (u, v)\), where \(u\) and \(v\) are scalar values representing components of the disturbance vector along the cardinal axes. At a given time \(T\), we assume we have access to some historical data of the disturbances for times \(t = \{T - 1, T - 2, \ldots\}\). Given these data, we want to provide predictions for future points of time as well as confidence bounds for these predictions.

A GP models a noisy process \(z_i = f(x_i) + \varepsilon\), where \(z_i \in \mathbb{R}\), \(x_i \in \mathbb{R}^d\), and \(\varepsilon\) is Gaussian noise. Since the standard GP models a one-dimensional value \(z_i\), we can model the full 2D or 3D space using separate GPs or as a coupled process (e.g., using the techniques in [36]).
We are given some data of the form

\[ D = [(x_1, z_1), (x_2, z_2), \ldots, (x_n, z_n)], \]

where \( x_i \) represents a vector of latitude, longitude, and time values for a data point \( i \), and \( z_i \) represents a component of the disturbance vector at that point and time. We refer to the \( d \times n \) matrix of \( x_i \) vectors as \( X \) and the vector of \( z_i \) values as \( z \).

To fully define a GP, we must choose a kernel function \( k(x_i, x_j) \) that relates the points in \( X \) to each other. As in our prior work, we utilize a space/time squared exponential kernel to model correlations among the data [34]. Having defined the kernel, combining the covariance values for all points into an \( n \times n \) matrix \( K \) and adding a Gaussian observation noise hyperparameter \( \sigma_n^2 \) yields \( K_z = K + \sigma_n^2 I \). The following equation predicts the mean function value (e.g., a disturbance value along the predicted trajectory) \( \mu(x_*) \) and variance \( V_{gp}(x_*) \) at a selected point \( x_* \) given the historical and prediction training data:

\[
\begin{align*}
\mu(x_*) &= k_*^T(K + \sigma_n^2 I)^{-1}z, \\
V_{gp}(x_*) &= k(x_*, x_*) - k_*^T(K + \sigma_n^2 I)^{-1}k_*,
\end{align*}
\]

where \( k_* \) is the covariance vector between the selected point \( x_* \) and the training inputs \( X \). This model gives a mean and variance for a particular latitude, longitude, and future time point.

The Gaussian Process variance described above gives an estimate of the uncertainty of a prediction based on the estimated hyperparameters and the sparsity of the data around that point. While the GP variance provides some useful insight into the uncertainty in predictions, it has been shown in prior work that it fails to correlate with the error in
complex disturbances (e.g., ocean currents) [37, 38]. Based on this work, we instead utilize a method based on the interpolation variance, providing a more informed uncertainty measure. Once a GP has been learned, the interpolation variance can be estimated as

\[
V_{iv}(x_*) = k_{*}^T(K + \sigma_n^2 I)^{-1}(z - \mu)^T(z - \mu),
\]  

(3.4)

where \( \mu \) is a vector of all \( \mu(x_i) \) values.

This measure of variance provides a richer representation that accounts for both data sparsity and data variability and while providing improved prediction for the trajectories of AUVs [34].

3.2.1.2 Probabilistic Planning

The learned uncertainty predictions described above can be incorporated into probabilistic path planners to refine human-provided trajectories. We propose utilizing Monte Carlo Sampling methods to estimate the transition function in a probabilistic model. The planner assumes that the stochasticity in the predictions uses the spatio-temporal variance estimates from the Gaussian Process (either the GP variance or the interpolation variance). These variances are used to generate a distribution of surfacing locations from a set of prior simulations.

This distribution of surfacing locations is obtained by performing a set of Monte Carlo simulations of a glider traveling through the ocean. For each simulation, starting at an initial state \( s \), we choose a waypoint to move towards, which represents taking action \( a \). The ocean currents for each point \( x_* \) are then drawn from the normal distribution centered at \( \mu(x_*) \) with variance \( V_{iv}(x_*) \) or \( V_{gp}(x_*) \). The simulation then determines the surfacing
location $s'$ based on these ocean current values. Aggregating these trials together, let $M_{s',s,a}$ be the number of samples ending at $s'$, starting at $s$, and taking action $a$. Also let $M_{s,a}$ be the total number of samples starting at $s$, taking action $a$, and ending in any state. We can generate an estimate of the transition function as $T(s'|s,a) = M_{s',s,a}/M_{s,a}$, which describes the probability of moving to state $s'$ given the choice of taking action $a$ from state $s$.

The proposed algorithm uses the transition function described above to evaluate a number of candidate plans. The costs of the plans are calculated using a weighting of the risk obtained from the risk map and the deviation from the operator’s initial trajectory of waypoints $\xi_0$ as described in Equation 3.1. The algorithm sequentially examines each operator-provided waypoint, then checks all possible alternative waypoints. From each initial waypoint $s$, the cost values $C$ are calculated for each possible action $a$ that could be taken using the following rule:

$$C(s,a) \leftarrow \sum_{s'} T(s'|s,a)(\Delta D(s',\xi_0) + \alpha \Delta R(s')),$$  \hspace{1cm} (3.5)

where $\Delta D(s',\xi_0)$ is the distance deviation between waypoints caused by adding state $s'$ to the trajectory and $\Delta R(s')$ is the difference in risk incurred by adding state $s'$ to the trajectory. In the domains of interest, the actions in the above equation represent target waypoints. Note that the actual waypoint reached will be different from the target waypoint due to the modeled disturbances. We discretize the possible target waypoints in the environment and then select the action with the lowest expected cost value. This process is then repeated for the remaining waypoints until the entire trajectory has been modified. This modified trajectory is spatially similar to the input trajectory, but it is optimized based on the weighting between the expected distance deviation and risk reduction in the surfacing locations.
Given the appropriate uncertainty measures and the planning methods described above, we now have a framework to modify the waypoints that are provided by the user. This combines the user’s intuition with the computer’s ability to optimize over large datasets. We would expect the transition models and risk maps to provide improvements in the reliability and safety of the resulting plan. The data-driven simulations in Section 3.3 will confirm this trend. However, in order for the computer to provide these modified trajectories, it must know $\alpha$, the user’s balance between risk and reward.

### 3.2.2 Coactive Learning Algorithm

The trajectory refinement algorithm presented in the preceding section balances risk and deviation from a provided trajectory using a weighting parameter $\alpha$. This weighting represents the human operator’s willingness to trade between the risks an AUV faces and the value of the information it gathers during a mission. In order to present relevant trajectories to the operator, our framework must have an estimate of the operator’s implicit weighting. To provide this, we propose an algorithm where the human iteratively refines trajectories given by a computer, which allows us to learn a generalized weighting of risk and reward.

Measuring deviation requires the operator to supply an initial trajectory. In this section, we relax this assumption and consider a more general “reward” map to model the human’s intent. We adapt the coactive learning algorithm [21] to learning a human expert’s preferences when planning paths for underwater scientific data collection. The algorithm attempts to learn the expert’s judgment of the utility of a set of paths. This learned utility function, described by $\alpha$, can then be used to create a refined trajectory mimicking that which the human would have planned.

Unlike the trajectory refinement algorithm, which only considers user preferences based on
a single human-provided path, the coactive learning algorithm utilizes multiple, incremental updates to a set of paths. This allows the algorithm to learn a much more generally applicable representation of human preferences based on risk and reward maps. In addition, it does not require the user to input a new trajectory whenever the algorithm runs.

We first present the basic perceptron coactive learning algorithm from prior work. We build upon previous work by adapting the coactive algorithm to noisy environments [39] and present a novel approach to dealing with suboptimal updates made by the human expert.

### 3.2.2.1 Perceptron Coactive Learning Algorithm

The perceptron coactive learning algorithm attempts to learn an expert’s utility function, $U(⟨x, y⟩) \to \mathbb{R}$, for judging a candidate solution $y$ for a given problem $x$ (as in [22]). We assume that the expert’s utility function can be approximated as a weighted linear function of features of the candidate solution: $\hat{U}(⟨x, y⟩) = \vec{w}^\top \vec{ϕ}(⟨x, y⟩)$. These features are simple numerical descriptions of a solution to the task, either concrete or abstracted (e.g., path length, distance to obstacles, probability of failure). However, the set of features used must be rich enough to describe the utility function the human uses to evaluate the task [20].

The ultimate goal of the algorithm is to learn the parameters $\vec{w}$ that match the expert’s method for judging the utility of a solution. This is equivalent to learning the operator’s preference weighing, $α$, from the previous section, as $α = \frac{w_1}{w_2}$, where $w_1$ and $w_2$ are the weights of the path deviation and risk features, respectively.

On each update of the coactive learning algorithm, the algorithm creates a candidate solution $y_t$ that maximizes $\vec{w}_t^\top \vec{ϕ}(⟨x_t, y_t⟩)$, based on its current estimate $\hat{U}$ of the expert’s utility function. This solution is presented to the expert. The expert has a set of operators, $\mathcal{O}$, that can be applied to the solution to improve it: $\mathcal{O}_i \in \mathcal{O} : ⟨x, y⟩ \to ⟨x, y'⟩$. These operators
are specific to the problem domain. In path planning, for instance, these operators could involve altering the trajectory. The cost for the update $C_t$ is equal to the number of operators the expert applies to improve the solution. The learning algorithm then adjusts $\hat{U}$ based on the difference in parameters between $y_t$ and $y'$.

Algorithm 1: CoactiveLearningUpdate (problem $x_t$, learning algorithm’s solution $y_t$, improved solution $y'$, cost $C_t$)

\[
\text{if } C_t > 0 \text{ then} \\
\quad \delta_t := \phi(x_t, y') - \phi(x_t, y_t) \\
\quad \bar{w}_{t+1}^\top = \bar{w}_t^\top + \lambda_t \delta_t \\
\text{end}
\]

Algorithm 1 shows how the weights $w$ are updated. If the expert has modified the proposed solution, the difference in parameters $\delta$ between the proposed and modified solution is calculated. This difference is then scaled by the learning rate and added to the previous estimated weights to find the new estimated weights.

In this work, as in most previous research into coactive learning, $C_t$ is simply the number of operators applied. However, there are several ways to make the cost more expressive, particularly for complex domains. For instance, different operators can have different costs, or the cost could vary based on how the operator is used (e.g., proportional to the distance a point is moved). Ultimately, the cost should reflect how much effort the human spends in improving a candidate solution, measuring how close the proposed solution is to the human’s ideal solution. In our work, this effort decreases as the proposed solutions converge on the target ratios.

Several variations of the coactive learning algorithm have been proposed. In [22], Goetschalckx and Tadepalli examine adjusting the learning rate $\lambda$. In addition to the above perceptron (PER) algorithm with with a constant learning rate, they also study a
passive-aggressive (PA) algorithm, which adjusts lambda to ensure the solver’s most recent mistake is corrected and a cost-sensitive perceptron (CSPER) algorithm where the learning rate is proportional to the number of operators applied.

Assuming that the expert provides a locally optimal solution, they prove an upper bound on the effort required by the expert. With $T$ being the number of update steps, the upper bound is $O(1/\sqrt{T})$ for the PER and PA algorithms and a bound of $O(1/T)$ for the CSPER algorithm. In [21], a lower bound of $O(1/\sqrt{T})$ on the algorithm’s regret is shown, assuming the expert provides an optimal solution.

However, using the algorithm with a human expert breaks these assumptions. The solutions provided by the human are unlikely to be locally optimal and could even be unintentionally misleading. Furthermore, only the human’s incorrect updates to a near-optimal solution change the learned weights. This causes the weights to oscillate as update steps are performed. One way to mitigate this issue is to present suboptimal candidate solutions half of the time, allowing the learned weights to be reinforced [39]. We incorporate the essence of this solution, as our learning algorithm does not create perfect solutions. We further extend it to specifically remove the effect of incorrect or erroneous updates.

3.2.2.2 Proposed Histogram Algorithm

The baseline perceptron algorithm is sensitive to suboptimal updates made by the human expert. The coactive learning algorithm assumes that all changes made to the candidate solutions are improvements in the eyes of the human. However, particularly in complex and noisy domains, it is easy for the human to make sub-optimal updates that they believe are improvements. The perceptron algorithm weights all updates equally, and does not attempt to distinguish good updates from poor ones. To overcome this limitation, we developed an
alternate algorithm to identify and reduce the effect of these suboptimal updates.

Our algorithm takes all previous improved weights into account when determining the new estimated weights. A histogram of the new and previously improved weights, \( \vec{w}_t \), for each feature is created. A normal distribution is fitted to the histogram. The center of the normal distribution is taken as the new estimated weight for each respective feature. This histogram identifies the weighting that is mostly likely to be correct based on the distribution of previous weights. Weights that are significantly different than the majority are averaged out, greatly reducing their effect. Thus, this method excludes outliers and prevents new updates from completely changing the estimated weights. In this way, the algorithm is able to continuously converge on the human expert’s weightings, even when a number of incorrect estimated weights are included.

For a small number of updates it can be difficult to reliably fit a reasonable distribution to the data. In this case the median \( (\vec{w}^\top) \) provides a good estimate of the new weights.

### 3.3 Simulations and Results

We evaluated the components of our learning and planning framework in the context of several different data-driven simulations. We begin by comparing the performance of our histogram-based coactive learning algorithm in estimating a human’s planning preferences to the performance of the baseline coactive learning algorithm. This demonstrates the algorithm’s success in learning a human’s weighting of several environmental variables. We then apply our trajectory optimization framework to a simulated Slocum glider AUV collecting environmental data in the Southern California Bight. The human’s preference weighting is used to inform the type of modified trajectories presented to the user by the framework. Finally, we present the results of field trials using the framework on a propeller-
driven EcoMapper AUV for monitoring lake ecology.

3.3.1 Comparative Learning Algorithm Simulations

First, we evaluate the ability of our coactive learning algorithm to estimate a human operator’s trajectory planning preferences. The problem we examine consists of several components: a planned trajectory of waypoints, a target vector $\vec{w}^T$ of feature weights for the learning algorithm to learn, and maps representing the value of those features in a region. These feature maps could represent real world variables, such as temperature or pH, or abstract features, such as a “risk” feature representing the cost of traveling in a given region or a “reward” feature that represents the quality and value of information gained by traveling in a given area [40, 41].

For our simulation, as in the trajectory optimization algorithm, we assume that the expert’s utility function is linearly composed of two features: the risk the robot incurs and the information it gains during its tour. The total risk and total information for a path are found calculating the line integral of each respective feature map along the path.

To test the algorithm’s ability to learn a human expert’s weighting, we have a human plan paths over a map created using a predetermined utility function. The expert is presented with a path overlaid on a map of the utility at each location in a region, as shown in Figure 3.3c. Maps of risk and reward are generated as a random sum of Gaussians, shown in Figures 3.3a and 3.3b. The utility map is generated by weighting these risk and reward maps by their respective target weights and summing them. For all tests, we use target weights of -10 and 30, for risk and reward respectively. This represents a stronger preference for gathering information than avoiding risk. The human is shown only the utility map. Since they are optimizing the path based solely on a map of utility calculated using the
(a) Reward map representing the value of traveling in a particular area.

(b) Risk map showing the risk of traveling in a region.

(c) Utility map generated from a weighted sum of the risk and reward maps. Here, the target weights of risk and reward are -10 and 30, respectively.

Figure 3.3: An example path and utility field generated using the coactive learning algorithm simulations. Only the utility map shown in (3.3c) is presented to the expert during trials. The black line represents the robot’s path through the environment. Here, risk and reward integrated along the path are the features used in the utility function. The computer proposes a trajectory to the expert, who then improves it. The framework learns the expert’s underlying weighting between gathering information and risk using coactive learning.

Predetermined target weights, we can test how effectively the coactive algorithm learns these weights without changing the human’s propensity for misjudging the utility of a given path.

In our tests, the algorithm used a simple greedy information-gathering path planner to generate candidate paths. First, it finds the peaks of a utility map made from the feature maps weighted by the learned weights. Then, using a locally optimal traveling salesman problem solver [42], it connects the peaks using a path that minimizes the inverse of the utility along the path. Thus, the planner finds a short path while still maximizing the utility of that path.

At each update, the expert improves the path by moving one of the points of the path. Shown only the utility map, they modify the path, attempting to maximize the line integral of the utility along the path. Thus, the expert’s utility function in planning matches the target utility function. After each modification, the change in information and risk are
calculated from the hidden risk and information maps, and used in the coactive learning update to update the learning algorithm’s estimate of the expert’s utility function. A new map and path are generated for each coactive update.

We conducted 20 trials each for the baseline perceptron and histogram algorithms. Each trial consisted of performing 16 updates based solely on the provided utility maps. Each update used a different map, with the expert moving one point on each map.

Figure 3.4: The regret (sum of deviation between target weight and estimated weight) for each coactive algorithm averaged over 20 trials. The standard error of the mean for the averages is shown. The histogram coactive learning algorithm accumulates regret more slowly.

As shown in Figure 3.4, the histogram algorithm accumulates regret more slowly than the perceptron coactive learning algorithm. Additionally, it also smoothly converged towards a set of estimated weights, as each update shifts the histogram only slightly. As seen in Figure 3.5, the perceptron algorithm is still highly susceptible to suboptimal updates made by the human, even after many iterations. This is because each update is valued equally
Figure 3.5: Example plots of how the estimated weight changes over a trial in simulations of the perceptron and histogram learning algorithms. Note that the perceptron algorithm initially tracks the target weights relatively well until a suboptimal update from the human expert throws it off. Comparatively, the histogram method converges on the target ratio, discounting an initial suboptimal update.
and the algorithm cannot compare the current update to previous feedback. However, the histogram algorithm learns what the optimal weighting is and is able to ignore or reduce the effects of suboptimal updates.

In our tests, we noted that, while the operator generally made good modifications to the path, suboptimal updates were still common and the resulting ratios of risk and reward were noisy, as seen in Figure 3.5a. While further study is necessary, we hypothesize that this is because humans concentrate on the features at the vertices of the trajectory, instead of examining the complete path.

The coactive learning algorithm provides a fast and effective method for learning an expert’s weighting, $\alpha$, between information gathered and risk incurred. There is no need for the human to tune $\alpha$ through trial and error. This weighting is then used as a parameter in the trajectory refinement algorithm to provide path suggestions that are relevant to the expert’s goals.

### 3.3.2 Data-Driven Trajectory Refinement Simulations

We now present a validation of the proposed trajectory refinement framework in the underwater monitoring domain in the Southern California Bight region where an autonomous underwater vehicle (AUV) is monitoring an oceanographic phenomenon with the help of a scientist. The simulations model a Slocum Glider [43], which is a buoyancy-controlled AUV that moves at a speed of approximately 0.3 $m/s$. The scientist provides the glider with a series of waypoints, and the vehicle dives between the waypoints while using dead reckoning to determine when to surface. Due to its slow speed, the glider is highly susceptible to ocean currents, and it is often difficult to predict where exactly the glider will surface relative to the specified waypoint. The glider is in danger of running aground if it comes too close to
land, and in addition, if the glider surfaces within a shipping lane, it becomes susceptible to collision with passing boat traffic.

The goal of these simulations is to determine the extent to which we can improve the safety of operation using the proposed learning and planning methods. In this domain, safety is measured by the probability of the underwater vehicle successfully completing its mission without coming too close to land or encountering a passing ship. The scientist provides the initial series of waypoints, and the proposed framework is then used to modify these waypoints to decrease the probability of collision while also minimizing the necessary deviation from the initial plan.

The simulations were performed on a single desktop PC with a 3.2 GHz Intel i7 processor and 9 GB of RAM. The simulations incorporate data from ocean currents provided by the JPL Regional Ocean Modeling System (ROMS) [35]. The JPL ROMS system provides estimates of the ocean currents but not uncertainty in those estimates. The uncertainty in the ocean currents was determined using the interpolation variance as discussed in Section 3.2.1. The uncertainty learning portion of the proposed method took approximately 5 minutes to complete, and the planning portion completed in less than a second using a 40 × 40 discretized grid of possible waypoint locations. We note that the uncertainty learning portion only needs to be run once per day, and many trajectories can then be refined using those uncertainty estimates.

Risk maps were generated for the simulation using historical Automatic Identification Systems (AIS) shipping data. AIS is a tracking system that mandates a large number of vessels in the United States (and other countries) to broadcast their location information via VHF transceivers (see [43] for more details). We used historical AIS data collected over a period of five months (between January and May, 2010) in the region 33.25 degrees N to 34.13 degrees N and 117.7 degrees W and 118.8 degrees W. Using these data, we calculated
an aggregate risk value, $R(s)$, at all possible discretized waypoints, $s$, in the region, which correlates with the chance of hitting a passing ship.

Figure 3.6 shows an example of how an initial trajectory was modified to reduce risk in the ocean monitoring domain. In this example, the human operator chooses to move the AUV into a risky harbor region to gather data. Using the previously learned $\alpha$, the algorithm then modifies this trajectory. The incorporation of the learned weighting allows the refined trajectories to balance the operator’s level of risk aversion with their desire of gathering data in the harbor. Figure 3.7 gives a visualization of how changing the weighting value ($\alpha$) affects the resulting trajectories.

These modified trajectories collect data in the same general areas while simultaneously avoiding the high-risk shipping lanes and the most dangerous areas of the harbor. We also see that the estimates of the ocean currents are more certain in the areas traversed by the modified trajectories (i.e., the modified trajectories move away from the red areas in Figure 3.6(b)). Effectively combining these two types of information would require an expert operator capable of processing data in real-time. With the proposed system, the operator can choose the desired trajectory without any underlying knowledge of the risk and then have the system refine it for increased safety and reliability.

Finally, we examine the effect changing the magnitude of the learned $\alpha$ parameter has on the modified trajectory. We provide quantitative evaluations of the deviation and risk trade-off between a high ($\alpha = 1000$) and low ($\alpha = 100$) value of the weighting parameter. In Figure 3.8, we see that the proposed method provides trajectories that range from closely tracking the initial trajectory with high risk to loosely following the initial trajectory with lower risk. In some cases we are able to achieve up to a 51% reduction in risk while deviating from the path by less than 0.8 km.
(a) Initial and refined waypoints overlaid on risk map. Lighter areas have higher risk of collision with land or passing ships.

(b) Initial and refined waypoints overlaid on ocean current uncertainty map. Redder areas denote higher areas of normalized uncertainty.

(c) Initial and final trajectory overlaid on ocean current predictions. Vectors denote direction and magnitude of ocean currents.

Figure 3.6: Example of improving an initial trajectory of waypoints using the proposed learning and planning framework for data from April 24, 2013. The refined path avoids the riskier (lighter) areas and also remains in areas where the uncertainty of the ocean currents is low. The resulting path is safer and more reliable without deviating much from the initial waypoints given by the scientist.
3.4 Field Trials

Finally, we demonstrate the framework’s capabilities in a lake monitoring environment. While our simulations show that the framework is able to learn and converge on a target weight and that it can refine trajectories to improve their safety, the goal in these experiments is to demonstrate that the framework was robust enough for use in an integrated field environment. They also show that the resulting feature weights of paths planned by the human and those by the framework were the same in a real world scenario. This demonstrates that the coactive learning algorithm is able effectively learn the human’s planning preferences and that these preferences can be incorporated into a path planner to effectively improve...
Figure 3.8: Deviation and risk for varying weighting parameters for trials of the trajectory optimizer using data from April 24, 2013. The proposed method allows the scientist to trade between the initially selected waypoints and safer paths that deviate from them. Each data point is averaged over 20 user-input trajectories, and error bars are one SEM.
upon a human’s planning abilities. By combining the human’s preferences and goals with a computer’s ability to quickly analyze data and plan trajectories, safer, more informative missions can be planned and run with less effort expended by the human operator.

We performed a series of trials with our framework using a YSI EcoMapper autonomous underwater vehicle in a lake ecology monitoring scenario. These propeller-driven AUVs are able to maintain speeds of $2 \, m/s$ for up to 10 hours. As such, they are often used in ecological monitoring and oceanographic research missions [44]. They have a wide range of sensors. These include water conductivity, temperature, and depth sensors for ecological monitoring and a Doppler Velocity Log and GPS unit for vehicle localization. Missions of waypoints for the AUV to follow are uploaded wirelessly using the standard 802.11 wireless protocol. Our field trials were conducted in an inlet of Puddingstone Reservoir in San Dimas, California (Lat. 34.08°, Lon. -117.81°), shown in Figure 3.9.

We trained our algorithm to plan paths based on the water temperatures and water depths along the path. These act as an analog to the risk and information maps used in the simulations. These also closely match a true ecological monitoring mission, where an oceanographer might target a certain combination of environmental features, such as temperature, depth, conductivity, and salinity in order to study a certain organism or ecological phenomenon. For each trial, we began by teaching our preferences to an information-gathering planner using our coactive learning algorithm. Unlike in the simulations, the human was shown both temperature and depth maps. Their true preference weighting between the features was learned. As in Section 3.2.2.1, this preference was represented as a linear utility function comprised of weighted temperature and depth features.

We then ran a dense lawnmower pattern over the inlet to establish a map of water temperature and lake depth for the planning framework to use. Using the learned utility function, the planner than ran the AUV on a path attempting to maximize the utility of
Figure 3.9: An aerial view of the test area at Puddingstone Reservoir (3.9a) with the corresponding depth map (3.9b). A simple baseline lawnmower path is shown over the depth map. The lawnmower path is a commonly used path for guiding AUVs in ecological monitoring missions [45]. The depth map was created by interpolating between depth points measured on a dense survey of the area.
the sensed information. Due to the depth of the inlet and to simplify the experiments, the AUV was used on the surface using 2D trajectories.

One limitation of these experiments is that depth and temperature are not independent features. Deeper locations are often colder because it takes longer for solar heating to warm them. As such, it is not always possible to find a path matching a given ratio of depth and temperature. For example, if maximizing depth is weighted highly with minimizing temperature having a smaller weight, it is likely the planned path will appear to satisfy both features equally. Additionally, since historical data for the lake were unavailable, the uncertainty estimation portion of the framework was not used. Even with these limitations, these trials provide a demonstration of the value of incorporating preference learning into a planning framework.

We began by running a loose lawnmower pattern, as shown in Figure 3.9. These types of trajectories are often used on ecological monitoring missions as they are easy to set up, so they formed a relevant baseline for our tests. The resulting ratio of depth to temperature integrated along the path was 1.157.

We then taught the algorithm to strongly maximize the depth while minimizing the temperature and ran the mission shown in Figure 3.10a. The measured ratio of 2.48 closely matched the learned ratio of 2.47. It also closely matched the measured ratio from the human-planned mission in Figure 3.10b of 2.45.

We also taught the algorithm to target specific depths and temperatures. We targeted lake depths close to 6 meters and temperatures very close to 27 degrees Celsius, weighting the depth more strongly. We again ran a computer-planned and a human-planned mission, shown in Figures 3.10c and 3.10d. The algorithm learned a weight ratio of 2.56. The measured utility ratios were 1.564 for the computer’s path and 1.495 for the human’s path. While these are not exactly the same, they are still quite close. Additionally, the achievable
A trajectory planned by our proposed algorithm maximizing the depth while minimizing the measured temperature.

A human-planned trajectory maximizing the depth while minimizing the measured temperature.

A trajectory planned by our proposed algorithm targeting a depth of 6 meters and a temperature of 27 degrees Celsius.

A human-planned trajectory targeting a depth of 6 meters and a temperature of 27 degrees Celsius.

Figure 3.10: A comparison of framework and human planned trajectories over the corresponding utility maps of Puddingstone Reservoir.

The ratio was limited by the correlation of the temperature and depth in the lake.

Finally, we tried to train the algorithm to follow the 6-meter depth contour by strongly preferring paths at that depth while ignoring all other features. The algorithm learned a weight of 19.34 for the utility of sampling a 6-meter depth and a weight of only 1.97 for the utility gained from sampling a 27.7 degrees Celsius temperature, giving a learned ratio of 9.81. The measured ratio of depth to temperature was 1.078, similar to the human-planned ratio of 1.205. Again, the learned ratio was not achievable due to the correlation of depth and temperature. Due to the high importance weight placed on measuring points at a 6-meter depth, the algorithm chose points with a depth of 6 meters, successfully planning a
For each trial, we compared the ratio of the temperature and depth features sensed along the path for the human- and framework-planned paths to the learned weights. The results are summarized in Table 3.1. While there was a small amount of variability due to inaccuracies in following the planned path and changing water temperatures, the ratios matched well. This shows that the framework is able to autonomously plan paths that follow the same preferences as a human’s.

These results show the framework’s benefit in marine data gathering scenarios. The robot is able to quickly learn a human operator’s goals and preferences, then autonomously plan trajectories that match these goals and preferences without further human intervention.
Table 3.1: Results of the lake ecology monitoring field trials. For each trial, a human operator taught our algorithm a preference between temperature and depth and planned a path based on that preference. The resulting measured depth to temperature ratios for the human-planned and framework-planned trajectories are shown. In each trial, the ratios measured match closely.

<table>
<thead>
<tr>
<th>Experimental Trial</th>
<th>Learned Ratio (Human-Planned)</th>
<th>Actual Ratio (Coactive-Planned)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximize Depth, Minimize Temp</td>
<td>2.47</td>
<td>2.48</td>
</tr>
<tr>
<td>Target 6m Depth and 27 deg C</td>
<td>2.56</td>
<td>1.495</td>
</tr>
<tr>
<td>Target 6m Depth Contour</td>
<td>9.81</td>
<td>1.205</td>
</tr>
</tbody>
</table>

3.5 Chapter Summary

The results in this chapter have shown that it is possible to combine waypoints provided by a human operator with historical data to improve the operation of autonomous vehicles in scientific monitoring scenarios. We have proposed Bayesian learning techniques that allow for uncertainty in predictions to be incorporated into the final trajectory, and we have integrated these uncertainty estimates into a probabilistic planning framework. We have also successfully integrated coactive learning algorithms into the trajectory optimization framework, allowing it to learn and mimic the priorities of a human expert. Using probabilistic techniques, our modified coactive learning algorithm gracefully handles imperfect updates made by the human.

The resulting framework allows for reduced risk of collision for an autonomous glider performing an ocean monitoring task with input from a human operator. By integrating feedback from the user into an algorithmic planning framework, we have effectively improved the safety and reliability of autonomous vehicle operation. The effectiveness of the framework has been shown in two simulations. In the first, we found that the algorithm’s estimated weights converge on a set of target weights in a reasonable amount of time for use with a
human expert. In field trials, we showed that the framework is able to use these learned weights to plan paths with a performance matching that of those planned by a human operator. The second data-driven simulation showed the effectiveness of the framework in making slight modifications to a trajectory to greatly increase the safety of the planned route.

Ultimately, we posit that techniques like the one proposed here will improve the efficiency of scientific data collection and allow human-robot teams to gather data safely and persistently in challenging environments. In the next chapter, we build upon our coactive learning approach and trajectory optimization framework to develop a new preference learning method that maintains this method’s resistance to noisy updates while using simplified, survey-style queries and a Gaussian process to allow for learning more complex, non-linear preferences.
Chapter 4: Efficient Learning of Trajectory Preferences Using Combined Ratings and Rankings

Building upon the framework developed in the previous chapter, we propose an approach for modeling and learning human preferences using a combination of absolute and relative queries in order to reduce the cognitive burden of analyzing and performing updates to candidate trajectories. Absolute queries require an expert to respond with a numerical value, while relative queries ask them to select the highest-value option from a set. Instead of weighting mission objectives, we model the user’s preferences as a numerical preference function over the mission objective space. This model uses a Gaussian process regression model with an associated likelihood function that can take into account both pairwise preferences and numerical ratings to approximate the user’s latent value function from a set of noisy queries. We show that a combination of relative and absolute queries performs better than either query type alone and propose a simple active learning approach to sequentially select informative queries that speed up the learning process when searching for high-value regions of the user’s latent preference space. We demonstrate the effectiveness of our method on a 1-D function approximation task and on a simulated autonomous surface vehicle performing a lake monitoring mission. These experiments show that our algorithm is able to efficiently learn an operator’s mission preferences and use those mission preferences to autonomously plan trajectories that fulfill the operator’s goals. The material in this chapter was originally published in Somers, Lawrance, and Hollinger [5].
4.1 Introduction

Current methods for performing robotic environmental monitoring place a high mental, physical, and time burden on human operators. Reducing this burden requires increasing levels of autonomy. In addition to adapting to and withstanding dangerous, dynamic, and unstructured environments, attaining a greater level of autonomy requires that a robot have a complete picture of the operator’s preferences and goals.

Due to environmental complexities, it can be difficult and time-consuming for operators to build controllers for autonomous vehicles performing complex tasks. Furthermore, many tasks do not have a single goal, but rather involve a trade-off between multiple objectives. For instance, an autonomous robot monitoring an ocean environment is required to observe multiple ecological variables while avoiding strong currents and obstacles, all with limited endurance and communication.

With current levels of autonomy, a team of trained experts is required to deploy and operate these robots for the duration of each of these missions. By improving the robot’s understanding of the operator’s goals, we can reduce this high mental (and often physical) burden on the operators while simultaneously increasing the robot’s ability to adapt to unexpected environmental conditions.

Several methods have been proposed to allow the robot to learn and model the expert’s goals [4]. With an accurate model of the expert’s preferences, the robot can plan actions for itself, leveraging the large amount of research into planning and optimization [46]. By automatically building a controller based on a reward function learned from an expert, the time-consuming complexity of manually programming a controller is removed. Ultimately, these methods combine the expert’s domain knowledge with the robot’s ability to gather and adapt to new data. However, current approaches, such as learning from demonstration,
have several disadvantages. They require the expert to manually provide demonstrations of robot trajectories and are susceptible to noise.

We propose a novel method that combines absolute and relative queries to learn a Gaussian process (GP) representation of the user’s latent reward function over the mission objective space. This method has several advantages over existing frameworks. First, combining the specificity of absolute trajectory ratings with the exploration value of a relative ranking of several trajectories speeds the learning process. Additionally, actively selecting between absolute and relative queries maximizes the information gained from each question. Furthermore, since our approach represents the learned preference function as a GP in the high-level objective space, it is able to incorporate domain knowledge and is resistant to noisy, non-linear inputs.

With experiments in non-linear function approximation and on simulated lake monitoring trials, we show that our proposed method efficiently learns an expert’s preferences and can use those preferences to plan trajectories that meet the expert’s goals. Furthermore, we demonstrate the value of adaptively selecting rating and ranking queries in reducing the effort required of the robot operator.

4.2 Related Work

The method developed in this chapter is motivated by the results presented in the previous chapter and draws upon previous research in ranking preference learning and active learning. In this section we discuss the relevant related work from each of these fields.

A key component of this work is the combined use of simple survey-style rating and ranking questions. Learning and modeling user ratings is a well-studied problem across many domains, such as for the well-known Netflix challenge [47]. However, learning ranking
models is still an open problem. Several methods have been proposed, including the ELO chess rating system commonly used to rank competitors in multiple games and sports[48]. Rankings have also been incorporated into the latent factor models used in recommender systems [49]. We base our methods for combining rating and ranking queries on previously studied methods for training GPs on ranking inputs [50, 51, 52]. In a robotic handover task, Kupcsik et al. [53] used a similar GP to estimate a human’s reward function during policy search. However, they only briefly mention combining query types and do not further discuss the generalizability or limitations of the method.

Active learning methods are used to further improve the convergence rate of supervised learning algorithms. They’ve been applied across a wide range of domains, including performing IRL in a simple grid world [54], informing reinforcement learning rewards for grasping tasks [55], and selecting poses for underwater inspection [56]. Many of these methods use a form of Bayesian optimization, such as an Upper Confidence Bound (UCB) metric [57]. However, finding the best heuristic has proven challenging, as estimating the value of any given query is highly domain dependent [57]. We demonstrate that actively selecting query types and trajectories provides significant benefit in the user preference domain.

4.3 Methods

In this section, we outline our methods for representing a user’s preferences as a GP in the mission objective space and for actively learning that representation using a combination of absolute and relative queries. Our goal is to create a system that efficiently learns a rich representation of the user’s preferences by utilizing faster and simpler queries than previous methods while also incorporating domain knowledge to speed up the learning process. Ultimately, this increases the autonomy of the robotic system while simultaneously
Figure 4.1: A trajectory $T$ is mapped to a corresponding objective-space feature vector $x_T$ (with feature dimensions of path length and accumulated risk). An expert is assumed to have an internal function $f$ that associates an objective-space vector with a scalar utility value $f(x_T)$.

reducing the human effort required to program and control the robot.

4.3.1 Problem formulation

We assume that the expert has an internal utility model that can associate a trajectory $T$ with a scalar utility value. We represent this model using a fixed objective feature space mapping that maps a trajectory $T$ to a set of $k$ objective values in the form of a vector $x_T \in \mathbb{R}^k$ in the objective space of the problem. A simple illustration is shown in Fig. 4.1. The dimensions of the objective space represent features of the trajectory that relate to mission success. For an autonomous underwater vehicle traversing an ocean these could include distance traveled, number of informative samples, and risk measures such as the strength of ocean currents. An example of a set of ratings in the lake monitoring objective space is shown in Fig. 4.7.

We assume that the user’s utility can be modeled as a function $f : \mathbb{R}^k \to \mathbb{R}$ that maps
from a $k$-dimensional feature space to a scalar utility. The user can be queried about their objective space utility through either absolute or relative queries. For an absolute query, the user is presented with a trajectory and asked to provide a numerical rating $u \in [u_{\text{min}}, u_{\text{max}}]$, representing a scalar utility measure on a bounded scale.

For relative queries, the user is presented with a set of $m$ trajectories, $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ and asked to identify the best (highest utility) trajectory $T_j \in \mathcal{T}$. This provides a set of $m - 1$ pairwise relationships where the selected trajectory is preferred over each of the other members of the query set, such that $T_j \succ T_i \forall T_i \in \mathcal{T} \setminus \{T_j\}$, where the trajectory preference relationship is related to the associated utility values, such that $T_j \succ T_i \leftrightarrow f(x_j) > f(x_i)$.

The goal is to identify regions of the objective space with high utility using a limited number of queries to the expert. In this work we explore a number of measures to quantify performance, as discussed in the results section. Overall, we are interested in showing that a combination of absolute and relative queries performs better than either query-type alone, and we explore approaches to actively selecting queries.

### 4.3.2 Objective Space Gaussian Process Learning

We represent the user’s preference function $f$ as a GP over the mission objective space of the robot [33]. The dimensions of the objective space represent features of the trajectory that relate to mission success. For an autonomous underwater vehicle traversing an ocean these could include distance traveled, energy used, and risk measures such as the strength of ocean currents, distance to nearest obstacle or probability of collision with shipping traffic. For a given trajectory, its $k$ objective values form a tuple in the objective space of the problem. The set of achievable objective tuples in the objective space form the feasible objective region $\Lambda \subset \mathbb{R}^k$, given the constraints of the environment and the operator’s preferences.
Our work draws on a formulation for GP regression that combines absolute and pairwise relative observations into a single modeling framework [52], adapting it to the preference learning domain. The GP predicts the unbounded, unobserved latent function $f$, from which we use a likelihood function to estimate the probability of observing the input training data. Figure 4.2 illustrates our approach as a graphical model. We assume that $f$ is smooth in the objective space, such that a GP is capable of interpolating to unknown points in the feasible region. This flexibility allows the GP to model user preferences for many different environments and objectives. Importantly, the GP is capable of both modeling complex preference functions as well as providing uncertainty estimates for predictions in a single model, allowing regions with high uncertainty to be targeted in active learning.

4.3.2.1 Gaussian process latent function

We want to estimate $f \sim \mathcal{GP}(0, k(x, x'))$, the user’s (unobserved) latent reward function from a set of observations collected from the user. For ease of notation, we group both absolute and relative queries into a single observation set consisting of input locations $X = \{x_1, x_2, \ldots, x_n\}$ and output observations $Y = \{d_1, \ldots, d_p, u_1, \ldots, u_q\}$. We assume that each $x \in X$ is unique but can be referenced by multiple observations (fig. 4.2). The GP is conditioned on the input observation locations and the GP covariance function hyperparameters $\theta_{GP}$ (fig. 4.2). For this work, we use the common squared exponential covariance function

$$k(x, x') = \sigma_f^2 \exp \left( -\frac{|x - x'|^2}{2l^2} \right), \quad (4.1)$$

with process variance $\sigma_f^2$ and length scale $l$, so $\theta_{GP} = \{\sigma_f, l\}$. The GP regression is similar to a standard GP regression problem, except that there is no analytic solution for solving
Figure 4.2: A graphical model illustration of the Gaussian process problem formulation. Circles represent random variables, boxes are deterministic. Filled shapes are observed. Observations consist of input feature vectors (filled squares $x$), output absolute utility observations (filled blue circles $u$) and output pairwise relative relationships (filled orange circles $d$). The GP (illustrated as the black horizontal bar) generates latent function values (unfilled circles $f$) conditioned on the hyperparameters of the covariance function $\theta_k$ and input locations. The relative and absolute likelihoods, $L_{rel}$ and $L_{abs}$, also conditioned on their respective parameters, provide the likelihood of the observations. Solving the model consists of finding values of $f$ and the hyperparameters that maximize the marginal likelihood of the observations. Predictions can also be made by querying the GP at an input location $x^*$ to generate the output distribution of the latent function $f^*$, and calculating the observation likelihood (a relative query between $x^*$ and $x_4$ is illustrated).
for the maximum likelihood posterior. Instead, we use the Laplace approximation that approximates the posterior as a normal distribution, and the system is solved by iteratively searching for the mode of the distribution that maximizes the posterior likelihood,

\[
\hat{f} = \arg\max_{f(X)} p(f(X)|Y)
\]

\[
= \arg\max_{f(X)} p(Y|f(X), \theta_L)p(f(X)|\theta_{GP}).
\]

Solving for the hyperparameters requires an additional step, repeatedly solving the maximum likelihood \(\hat{f}\) for given hyperparameters, then varying the hyperparameters to maximize the posterior likelihood until both converge. Chu and Ghahramani [50] show that this problem is convex and can be solved using gradient descent. Jensen and Nielsen [52] provide analytic derivatives for the likelihood functions listed below with respect to their hyperparameters.

### 4.3.2.2 Relative observation likelihood

The formulation for a pairwise preference likelihood function was originally formulated in [50]. It is closely related to the probit formulation for binary GP classification, where a likelihood function is used to determine a posterior estimate of the GP latent function that maximizes the (log) marginal likelihood of observing the binary training labels [33]. We use a preference relationship for ranked points, where an input point \(x_i\) is said to be preferred over \(x_j\) (written \(x_i \succ x_j\)), if \(f(x_i) \geq f(x_j)\).

Thus, a relative training point consists of a pair of input points \((x_i, x_j)\) and an associated binary observation \(d \in \{-1, 1\}\), with \(d = -1\) signifying to \(x_i \succ x_j\) and \(d = 1\) the opposite. To incorporate noise, we assume that the observations of \(f\) are drawn from i.i.d. Gaussian distributions with fixed variance \(\sigma^2_R\) around the true function \(f\), and the label \(d\) represents
which sample is larger. The likelihood $L_{rel}$ of observing a label can be written

$$L_{rel}(d|f(x_i), f(x_j), \sigma_R) = \Phi \left( \frac{d \cdot f(x_j) - f(x_i)}{\sigma_R \sqrt{2}} \right)$$

(4.3)

where $\Phi : \mathbb{R} \rightarrow (0, 1)$ is the cumulative distribution function for the normal distribution, $\Phi(z) = \int_{-\infty}^{z} N(\gamma; 0, 1) d\gamma$. There is one hyperparameter of equation (4.3), $\theta_{L_{rel}} = \{\sigma_R\}$.

4.3.2.3 Absolute observation likelihood

To incorporate absolute observations, where the expert is queried about a single trajectory with associated feature vector $x$ and provides a scalar utility value $u \in [0, 1]$, we use a formulation from [51] where the likelihood is represented by a beta distribution,

$$L_{abs}(u|f(x), \theta_{L_{abs}}) = \text{Beta}(\alpha(x), \beta(x)).$$

(4.4)

The beta distribution provides a probability density over a bounded interval $(0, 1)$, and is parameterized by two shape parameters $\alpha$ and $\beta$:

$$\alpha(x) = \nu \mu_B(x),$$

(4.5)

$$\beta(x) = (1 - \nu) \mu_B(x).$$

(4.6)

Since the GP itself maps onto an unbounded scale ($\mathbb{R}$), we also need a function that links the prediction from the latent function $f$ to the observed utility value $u$. We adopt an approach proposed in [52] that links the mean of the beta distribution $\mu_B$ with the mean of the GP
prediction $\hat{f}(x)$ using the common probit mean link function,

$$
\mu_B(x) = \Phi \left( \frac{\hat{f}(x)}{\sigma_B \sqrt{2}} \right). \quad (4.7)
$$

The hyperparameters for the absolute likelihood are $\theta_{L_{abs}} = \{\sigma_B, \nu\}$. $\sigma_B$ scales how $f$ is mapped to the output range $(0, 1)$, and $\nu$ is a precision variable that determines how ‘peaky’ the beta distribution is around the mean.

4.3.2.4 Prediction

For the active learning process, and to identify high-utility paths, we need to be able to generate predictions of both absolute and relative likelihoods from the GP model for unobserved locations $X^*$. Generating predictions from the GP latent function requires the maximum likelihood solution $\hat{f}$, and the negative Hessian of the log likelihood $W$, where

$$
W_{i,j} = \sum_k \frac{\partial^2 - \log \mathcal{L}(y_i \mid f(x_i), f(x_j), \theta_L)}{\partial f(x_i) \partial f(x_j)} \cdot \quad (4.8)
$$

The latent posterior distribution is $f^* \sim \mathcal{N}(\hat{f}^*, K^*)$, where

$$
\hat{f}^* = K^T_{X,X \cdot} K^{-1}_{X,X \cdot} f,
K^* = K_{X \cdot, X \cdot} - K^T_{X,X \cdot} (I + WK_{X,X})^{-1} WK_{X,X \cdot} \quad (4.9)
$$

These results can then be used to calculate the output likelihood distributions my marginalizing out $f^*(x)$:

$$
p(y^* \mid x^*, X, Y) = \int \mathcal{L}(y \mid f(x^*), \theta_L)p(f(x^*) \mid X, Y) df(x^*). \quad (4.11)
$$
Note that in the relative case, this integral (over $x_i, x_j$) can be solved analytically to yield
\[
p(x_i > x_j | X, Y) = \Phi \left( \frac{\hat{f}^*(x_i) - \hat{f}^*(x_j)}{2\sigma^2 + K_{i,i}^* + K_{j,j}^* - 2K_{i,j}^*} \right).
\] (4.12)

In the absolute case, solving for the full distribution $p(u | X, Y)$ requires a sampling approach. However, [52] provides a fast solution for solving for the mean of the beta distribution,
\[
\mathbb{E}_{p(u | X, Y)}[u] = \Phi \left( \frac{\hat{f}^*(x)}{\sqrt{2\sigma^2 + K_{x,x}^*}} \right).
\] (4.13)

Figure 4.3 shows an example of a ‘true’ one-dimensional latent function, and the resulting sampling likelihood distributions. Figure 4.4 shows the posterior estimate of the latent function from the GP and resulting posterior likelihood estimates given the training samples from Fig. 4.3. It is interesting to note the effect of the relative observations which provide general shape information versus the absolute measurements which provide strong estimates of the value of the latent function but only in a small region.

4.3.3 Active Selection of Ratings and Rankings

Absolute and relative queries provide different coverage of the objective space. An absolute query learns an accurate utility for a single point while a relative query provides general pairwise comparisons of several points, thus exploring a larger area of the space. Additionally, rankings are intuitively easier to make and users are more confident about their responses [49]. By combining rating and ranking queries, our method is able to make use of the benefits of each.

The trajectories $T_i$ for each query $q_i$ are selected based on the upper confidence bound
Figure 4.3: True latent function, associated likelihoods and samples. The absolute likelihood (center subfigure) illustrates the probability density of $u$ over the domain of $x$ values, so that each vertical slice of the image is a density $p(u|f(x))$, with $\sigma_B = 1.0$ and $\nu = 80.0$. The right subfigure shows the likelihood of sampling the class label $d = -1 (x_0 > x_1)$ for all pairings of $x_0, x_1$ in the domain, with $\sigma_R = 0.1$. The blue lines with circles show pairwise relative samples, where the circle is at the end of the preferred input, and black circles indicate $d = -1$ and white circles $d = 1$. Samples from the absolute function $u$ are shown as white crosses.

(UCB) of the GP’s estimate of the user’s rating for $T_i$:

$$x_{t+1} = \arg\max_x \hat{f}^*(x) + \gamma \sqrt{K^*(x)}$$  \hspace{1cm} (4.14)

UCB is well suited to our method as it selects trajectories that have both a high level of uncertainty and are also likely to be highly rated [57]. In learning a user’s preferences, it is most important for the robot to be confident that it understands which trajectories have high utility. These trajectories are often difficult to learn as they comprise a small, localized portion of the objective space. Thus, once a region of the objective space has been found to be poorly rated, there is little value in continuing to explore it.
4.4 Experiments and Results

4.4.1 Randomized trials for active learning

To demonstrate the advantage of combining ratings and rankings in a single framework we compare the performance of different query selection algorithms on a learning task. We generate ‘truth’ functions as a sum of three random sinusoidal wave features:

\[ f(x) = \sum_{k=1}^{3} a_k \cos(\pi f_k(x - o_k)) \exp(-d_k(x - o_k)^2) \]  

where the frequency, amplitude, offset and damping are uniformly randomly selected from the intervals \( f_k \in [10, 30], a_k \in [0.6, 1.2], o_k \in [0.1, 0.9] \) and \( d_k \in [250.0, 350.0] \) respectively. The input space is limited to \( x \in [0, 1] \). In each trial instance, we sample a wave function that can be (noisily) queried, and provide one randomly placed absolute training sample as initial data. Each tested method sequentially selects query locations and samples the true function until the maximum number of queries is reached. All methods use the same GP formulation and hyperparameters, and differ only in their active selection algorithm.

Figure 4.4: Posterior estimate with five absolute samples and five pairwise relative samples. The GP estimate of the latent function is shown left with \( 1\sigma \) bounds, and the resulting posterior likelihoods are shown in the center and right subfigures respectively.
We compare our proposed UCB Combined method (labeled UCBC in plots) to a random absolute-only sampler that randomly selects a single rating query at each step, a random relative-only sampler that randomly selects five points for a ranking query and a pure UCB method that greedily selects a rating query based on the upper confidence bound. Our experiments used $\gamma = 3$ based on hand tuning for best performance.

To measure performance we use a weighted RMSE (similar to [58]) over a uniformly distributed set of $n = 100$ test points which calculates the RMS error between the predicted absolute rating $u_{\text{est}}$ and the true absolute rating $u_{\text{true}}$ weighted by the magnitude of the rating, such that high valued points are weighted higher than low-valued points. We modify the method used in [58] by weighting the squared error by the maximum of the predicted and the actual ratings. This ensures that if the method predicts a high value where the truth is low, or vice-versa, this will adversely affect the performance score:

$$WRMS = \sqrt{\frac{\sum((u_{\text{est}} - u_{\text{true}})^2 \cdot \max(u_{\text{est}}, u_{\text{true}}))}{n}}.$$  \hspace{1cm} (4.16)

Figure 4.5a shows the WRMS for each method over 100 trials.

We are also interested in how well each method would select high-value points given a fixed number of observations. We identified the 15 points with the highest ratings from the 100 uniform samples of each true function, and after each observation selection, we queried each method for their 15 highest rated points to compare to. Figure 4.5b shows the number of true positive selections. This metric shows methods that correctly identify the high-value areas, but don’t necessarily correctly estimate the rating magnitude.
4.4.2 Simulated Lake Monitoring Trials

In these experiments, we study the use of our method on a simulated autonomous surface vehicle (ASV) monitoring a lake environment. The ASV must travel from a start location to a goal location while planning a trajectory that balances the distance traveled with the amount of information sampled along the trajectory. The goal is to allow the ASV to autonomously plan its mission trajectories while maintaining the same balance of objectives, distance and information gathered, that the operator would.

The environment consists of a simulated information field over a lake, a diverse set of trajectories across it (Fig. 4.6), and their associated objective scores (Fig. 4.7). The information field is generated using a sum of 2D Gaussians with added Perlin noise [59]. The information objective score is calculated as a path integral of the information field along the trajectory. Two motion planners, STOMP [46] and RRT [60], are used to provide path diversity. In order to cover the objective space, the paths are planned using a weighted
Figure 4.6: Two example lake monitoring trajectories from the trajectory pool are shown superimposed over a simulated objective field representing the information gained by traveling over each part of the lake. Red areas represent locations with higher information. The dashed trajectory is longer but gathers more information.

linear combination of the objectives as a cost function $cost = v_1 + w \cdot v_2$. By varying $w$, trajectories in different parts of the objective space can be created. 200 paths, 100 from each planner, were generated for each training and test set.

We designed a simulated user that represents a human operator performing an environmental sampling mission. As shown in fig. 4.7, the user wants to score at least 150 on the information gathering objective. Above that, it attempts to minimize the distance traversed. Given these non-linear user preferences, the utility $u$ of a trajectory is encoded by the following equations:

$$u = \begin{cases} 
1, & \text{if information } < 125 \\
2, & \text{if } 125 < \text{information } < 150 \\
[5 - (\text{distance} - 450)/50], & \text{otherwise.}
\end{cases}$$
Figure 4.7: Trajectory ratings for the simulated user plotted over the lake monitoring mission objective space. Markers are colored by rating: red = 1 (a poor trajectory), yellow = 2, green = 3, cyan = 4, and blue = 5 (an excellent trajectory). The simulated user wants to gather 150 information samples. Above that threshold, lower distances are preferred.

For an absolute rating query, the user rated the presented trajectory on a five-point Likert scale, with 1 being an unacceptable trajectory and 5 being an excellent trajectory [61]. In a relative ranking query, the user is asked to select the best trajectory from a set of five. These scales and set sizes were selected as they represent a good balance of gaining specific information without requiring lengthy consideration by a human user. The ratings are linearly scaled to the range [0, 1] for use in the GP.

We compare our proposed combined method of choosing between absolute and relative queries (labeled UCBC in plots) to randomly selecting only absolute queries and to actively selecting only absolute queries based on the UCB in eqn. 4.14. 20 learning trials were run
To measure the performance of these methods, after each query we calculate the rating prediction error of the learned GP on the trajectories in the test set that would be rated five by the simulated user. Additionally, we calculate the WRMS error, as in equation 4.16. These error metrics measure a method’s ability to learn the user’s preferred region.

The mean error and WRMS of each method are given in Figures 4.8a and 4.8b. These results show that our method learns and identifies high-utility trajectories with fewer queries than methods using ratings alone.

We made several qualitative observations of the algorithm’s performance during the trials which help explain these results. As compared with random queries, active learning reduces the number of queries about uninformative portions of the objective space. However, in some trials, the learner would fail to query about a trajectory with a user rating of five, having estimated that four was the highest possible rating. Incorporating relative queries alleviates this issue by allowing a much larger number of trajectories to be examined. Overall,
these results show that our method can successfully generate highly-rated trajectories while improving learning times, which can lead to reduced operator burden and more efficient human-robot teaming.

4.5 Chapter Summary

In this chapter, we proposed a novel preference-learning algorithm for learning a robot operator’s preferences about mission objectives. We showed that a robot can efficiently obtain a rich representation of these preferences by actively combining simple rating and ranking queries with a Gaussian process. Our experiments on non-linear one-dimensional preference functions and in a simulated lake-monitoring environment showed that multiple query types and principled active learning can significantly improve the convergence rate of preference learning. The learned preference models are applicable to planning and trajectory optimization in a wide variety of domains, such as aquatic robotics, where the dynamic environment and limited communication necessitate a complete understanding of mission goals, and in disaster scenarios, where strong autonomy is necessary for reducing the burden on response teams.

Though further development work is needed to validate the use of this Gaussian process preference learner in an end-to-end scenario like the field trials presented in Chapter 3, the learned model could be used to directly replace the coactively-learned weighting in the planning framework. Ultimately, this framework and the associated preference-learning methods have the potential to reduce the cost, time, and operator burden of deploying and controlling autonomous robots.
Chapter 5: Conclusions

This thesis describes two methods for a robot to learn the preferences of a human. These learned preferences enable the robot to autonomously perform as a member of a cohesive human-robot team by planning its actions using a detailed, accurate representation of the human operator’s goals for the mission. This eases the burden on the operator by reducing the amount of data they must process when planning, helping to transfer their domain knowledge and intuition to the robot, and decreasing the number of decisions they need to make in real time during the mission.

The novel algorithmic contributions presented in this thesis include:

- A noise-resistant, histogram-based, coactive learning algorithm that learns human preferences for trade-offs between mission objectives from repeated, incremental improvements to example mission plans.

- An algorithm that learns a Gaussian-process-based preference model from survey-style rating and ranking queries. By actively selecting rating and ranking queries, the algorithm leverages the strengths of each to quickly learn the most-preferred regions of the preference space.

Additional contributions of this thesis include:

- Tests on simulated lake monitoring environments which compare the histogram-based coactive learning algorithm to the baseline coactive learning algorithm. These tests show that the histogram-based algorithm accurately learns a human’s preferences while simultaneously being resistant to erroneous updates made by the human.
• Field trials using an EcoMapper AUV to demonstrate the use of the coactive learning method in ecological monitoring. In these trials, the AUV learned a trade-off between two environmental variables, then used that learned trade-off to autonomously plan and carry out a monitoring trajectory. The sensor data from these runs showed that the autonomous mission performance matched the operator’s preferences.

• Experiments comparing our Gaussian process with actively-selected relative and absolute queries to Gaussian processes using either relative or absolute queries alone. On simulated functions and lake-monitoring trajectories, these tests demonstrate the value of combining both query types and actively selecting the trajectories in those queries for reducing the iterations needed to learn the user’s preferred mission parameters.

5.1 Future Directions

The tests and demonstrations in this paper show the value of using preference learning to inform robot autonomy. They also point to several ways to improve upon these results, and they suggest other avenues for study that would further enhance the ease of use and autonomy of robots in human-robot teams.

First, actively selecting between rating and ranking queries shows great promise, but this method would benefit from a more theoretically grounded algorithm for actively selecting between the two query types. While UCB works, it is not based on the strengths of each query type. In particular, relative queries provide general information that can inform the algorithm about the best preference-space locations for future queries, but they do not provide concrete information until referenced against an absolute query. This suggests that an information measure that considers multiple future queries may have better performance
than the current UCB metric. Additionally, methods for incorporating other query types into the GP, such as complete demonstrations and fully-ordered rankings, should be explored to further broaden the method’s capabilities.

Next, large-scale user studies are needed to fully define the capabilities of the presented learning methods and to study how well they function when faced with differing levels of user expertise. These studies are necessary to measure the reduction in time and cognitive burdens placed on the human operators. Additional mission objectives should be added to these studies in order to allow the learned model to more closely match the human’s intention. To complement this, techniques for identifying these relevant objective features should be examined.

Another area for future work lies in combining the preference-learning and risk-modeling techniques presented in this thesis with modern adaptive sampling methods. Our learning and refinement methods allow the human and robot to work together to meet a common goal through shared autonomy. Combining this capability with adaptive sampling and planning methods would allow marine vehicles to continuously monitor and adapt to their environments on a long-term mission while pursuing the same goals as a human. Thus, marine robots would have a level of persistent autonomy, allowing them to safely complete long, complex marine data collection missions.

There are also several broader directions for future study which align with current trends in robotics research. The techniques described in this thesis could be combined with transfer learning methods to bootstrap these learning algorithms with a previously learned set of preferences that are close to, but do not exactly match, an operator’s preferences. The space coverage of relative queries could allow for efficient finding and re-learning of regions of the preference space that do not easily transfer to the new operator.

Further work could include leveraging the contrasting strengths of absolute and relative
queries as part of an explainable AI algorithm. By explaining the algorithm’s goal for a particular query, the algorithm may increase the operator’s trust that their preferences are being learned accurately. The value of such an approach became clear while conducting informal tests of the coactive learning algorithm, during which users would regularly pause and question why the algorithm chose to present them with a particular path to improve.

While human-robot teams are used in an increasing variety of tasks, there is still a need for a deeper understanding of the best methods for coordinating humans and robots. The research described in this thesis develops general methods for improving this coordination through an increased understanding of a human’s preferences. Ultimately, this work brings together machine learning and human-robot interaction with the goal of simplifying the use of robots so they can better assist with a variety of tasks, from scientific data collection to personal assistance.
Bibliography


