© 2013 Abdullah Akce

HUMAN CONTROL OF ROBOTS OVER DISCRETE NOISY CHANNELS WITH HIGH LATENCY: TOWARD EFFICIENT EEG-BASED BRAIN-ROBOT INTERFACES

BY

ABDULLAH AKCE

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Doctoral Committee:

Assistant Professor Timothy Bretl, Chair
Professor David Forsyth
Professor Seth Hutchinson
Professor Steven LaValle
Associate Professor Andrew Bagnell, Carnegie Mellon University

Abstract

This thesis presents a framework for the design of interfaces that can only obtain noisy and discrete inputs at high latency from a human user (e.g., with an electroencephalograph) to control a robotic system (e.g., a robotic wheelchair) that can provide visual feedback. In this framework, the human user communicates their intent by providing inputs in response to queries posed by the robot. The underlying problem is then to construct a policy that determines the next query to be posed in order for the robot to infer the user's intent as quickly and as reliably as possible. The approach is to maximize the expected amount of information to be obtained per unit of time from the user's response given a Bayesian estimate of the user's intent and an estimate of how quickly and accurately the robot can obtain the user's response. Under certain conditions, this policy reduces to the optimal feedback policy for transmitting a message between two computational agents over discrete noisy channels. Remarkably, for an interesting class of user intents (e.g., desired paths for robotic navigation), the queries synthesized by the optimal feedback policy can be easily understood and used by humans to convey their intent to the robot.

As a case study in the application of this framework, this thesis focuses on the design of EEG-based brain-robot interfaces, which allow human users to control robotic systems with an electroencephalograph (EEG). It presents two interfaces for robotic navigation, where the user's intent was a desired path to be followed by the robot, and one interface for text entry, where the user's intent was a desired character to be spelled. The first interface enabled users to navigate a simulated aircraft flying at a fixed speed and altitude over smooth paths that corresponded to a sequence of path primitives. The second interface enabled users to navigate a mobile robot in a virtual indoor environment over paths that (locally) minimized a cost function recovered from human-demonstrated data. These two interfaces provided a new strategy, i.e., navigation based on querying desired paths, which was shown to be advantageous over existing EEG-based interfaces for robotic navigation. The third interface enabled users to specify text commands with inputs obtained from steady-state visually-evoked potentials in EEG at a rate twice as fast as they would using prior state-of-the-art text entry interfaces working with the same input mechanism. This interface showed the importance of querying human intent adaptively based on the prior knowledge on human intent (e.g., likelihoods of characters) and the expected accuracy and latency of inputs.

To my lovely wife Sara, and our unborn child.

Acknowledgments

First and foremost, all praise belongs to Allah for granting me the opportunity to write this dissertation.

I would like to express my sincere appreciation to my advisor, Timothy Bretl, for his endless guidance, support, and encouragement. It has been a privilege to do research on challenging problems under his supervision, which motivated me to constantly look for better approaches. I have benefited a lot from his vision, and especially from his interdisciplinary efforts. Without these efforts, and his persistent help, this thesis would not have been possible.

I have been very honored to have David Forsyth, Seth Hutchinson, Steven LaValle, and Andrew Bagnell in my dissertation committee. I would like to thank them all for their invaluable feedback and assistance. Thanks to David Forsyth, who has been extremely helpful with his inspiration and encouragement. Thanks to Seth Hutchison, who has provided useful suggestions and directions for research. Thanks to Steven LaValle, who has always been eager to help since my first days as a graduate student, and has encouraged me to do interdisciplinary research. Thanks to Andrew Bagnell, who has invited me to his lab at Carnegie Mellon University, and has provided new perspectives on my research.

In addition to the members of my dissertation committee, I have interacted with several faculty members at University of Illinois. I would like to thank them all for their time and support. In particular, I owe many thanks to Dan Roth, Jeff Erickson, and Todd Coleman.

I would like to extend my appreciation to members of my research groups, the Robotics & Neuro-Mechanical Systems lab, and the Brain-Machine Interface group at University of Illinois, for their support, encouragement and contributions. In particular, I am indebted to James Norton, Miles Johnson, Rui Ma, Or Dantsker, Aadeel Akhtar, Navid Aghasadeghi and Aaron Becker for helping me in many different ways.

I am also grateful to Onur Pekcan and Baris Aktemur for helping me adapt to the environment at University of Illinois, and for offering me their help at every opportunity.

Finally, I would like to thank my family for their support, encouragement, and reassurance. My deepest appreciation goes to my wife, Sara Olgun, for her love and patience.

Table of Contents

Part I

INTRODUCTION

Chapter 1

Introduction

1.1 Motivation

We are motivated by the design of non-invasive brain-machine interfaces (BMIs) for the control of robotic devices, such as a prosthetic or a powered wheelchair. These interfaces translate measurements of brain activity into input commands for the robotic device, effectively enabling a human user to control the robotic device with thoughts instead of with physical movements. In particular, BMIs can be used by individuals with impaired sensory-motor function, such as the patients suffering from severe spinal cord injuries, to improve their quality of life. For instance, a tetraplegic human patient has successfully controlled a wheelchair in a virtual environment with input only from an electroencephalograph (EEG) [1].

The challenge in the design of EEG-based brain-machine interfaces is that the input commands decoded from EEG signals are discrete, noisy and have high latency. For example, in the motor-imagery paradigm, human users can provide binary input commands by imagining movement of their left- versus right-hand. The correlates of motor imagery in EEG signals come at a low rate and with a low signal-to-noise ratio. Decoding of EEG signals yields a single binary input at about every second with usually more than 10% chance of error. Mapping these inputs directly to a control command for navigating a robot, as would be done with a joystick, is no longer a good idea.

Most of the work on brain-machine interfaces has originated within the neuroscience community and has focused on enriching the measurement and decoding of brain activity. Much less work focused on improving the human control of the robotic device under the constraints in the human user's ability to provide inputs. What is the appropriate level of task specification? How should the user choose input commands? How should the robot provide

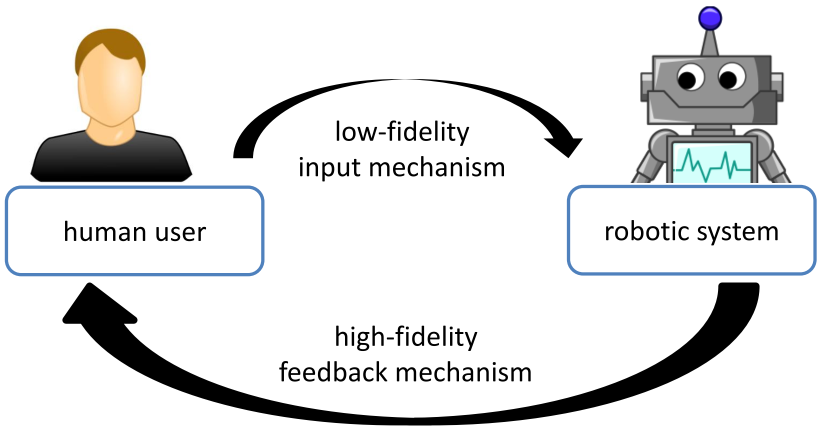


Figure 1.1: This thesis considers human control of robots over a low-fidelity input mechanism by leveraging the high-fidelity feedback mechanism. See Section 1.2 for details.

feedback? These are the kind of questions we are interested in answering because they have a significant impact on the overall performance of a brain-machine interface.

1.2 Scope

We restrict our scope to the human control of robots with interfaces that have a low-fidelity input mechanism and a high-fidelity feedback mechanism (Figure 1.1). In particular, we consider low-fidelity input mechanisms that provide noisy discrete input commands from the user to the robot with high latency, such as the EEG-based brain-machine interfaces, and high-fidelity feedback mechanisms that provide visual stimuli generated by the robot to the user, such as a graphical display showing possible routes for robotic navigation. Our objective in human control of robots with such interfaces is to allow humans communicate their intent to the robot as quickly and as reliably as possible.

In this section, we describe the assumptions we make about the input and feedback mechanisms, and the capabilities of human users and the robotic systems. We emphasize that these assumptions are satisfied by the three EEG-based brain-machine interfaces we developed, which will be described in Part III of this thesis.

1.2.1 Assumptions about the input mechanism

The input mechanism is used by the human user to provide discrete inputs, i.e., an input command from a low-cardinality set of possible input commands. The provided input command is observed with noise by the robot, meaning that with some error probability the robot will not observe the command that the user provided. Furthermore, the observation occurs with latency, meaning that the robot will receive an input command a period of time after the user provides an input command. We also assume that the noise and the latency in the input mechanisms follow a predictable structure. In other words, the robot can estimate how much quickly and accurately input commands can be provided with the input mechanism.

1.2.2 Assumptions about the feedback mechanism

The feedback mechanism is used to provide visual stimuli generated by the robot to the human user. We assume that the visual stimuli are perceived by the human user without any noise, and without any latency. In other words, the feedback becomes immediately available to the human user without any corruption.

1.2.3 Assumptions about the human user

We assume that the human user can interpret the visual stimuli generated by the robot accurately, and follow a protocol that tells exactly which input command to provide in order to communicate a particular intent. We refer such a protocol as a "query", and providing an input command by following the protocol as responding to the query. For example, if the human user is presented with a target speed for the robot, and the protocol is to provide the input 0 for specifying a slower speed, and to provide the input 1 for specifying a higher speed, the human user can accurately provide the correct input bit correlated with their desired speed.

We further assume that the user determines their intent prior to communication, and their intent remains fixed until the communication is over. However, the methods and applications that will be described in this thesis are still applicable when the user changes their intent during the communication, as long as the new intent remains consistent with the prior input commands (i.e., the responses to the previous queries remain the same).

1.2.4 Assumptions about the robotic system

We assume that the robot can present a visual stimulus, with which the human user can follow a protocol to determine the input command for conveying a particular intent. We refer presenting such a stimulus as posing a query or querying the human intent. For example, in order to determine the desired place the human user wants the robot to go to, the robot can pose a query with a visual stimulus that highlights a particular place in a menu of target places. The protocol to be followed, described by the query, might be to provide the input 1 if the highlighted place is the desired place; and the input 0 otherwise.

We further assume that the robot is capable of autonomously performing the desired task correlated with the user's intent. For example, if the user's intent is a desired path for the robot to follow, we assume that the robot can autonomously follow the paths that might be specified by the user.

1.3 Take-Away Message and Contributions

This thesis will show that human control of robots (as described in Section 1.2) can be done efficiently by having humans provide input commands in response to queries that are posed by the robot to maximize the expected amount of information to be obtained per unit time about human intent. The contributions of this thesis are as follows:

- *A framework to design interfaces for human control of robots.* We introduced an information-theoretic framework that modeled human control of robots over discrete noisy channels with high latency as the problem of communicating the human user's intent to the robot as quickly and as reliably as possible. Under this model, we developed two approaches to the interface design. In the first approach, we derived an optimal feedback policy to querying human intent [2, 3, 4, 5]. This policy was based on the theory of *optimal feedback communication*, which aims to design codes for transmission of messages between two computational agents [6, 7]. We showed that, for an interesting class of human intents (e.g., desired paths for robotic navigation), the queries (codes) synthesized by the optimal feedback policy can be easily understood and used by humans to convey their intent (message) to the robot. In the second approach, which generalized the first approach, we derived an active inference policy to querying human intent. This policy was based on the theory of *Bayesian active inference (or learning)*, which aims to adaptively select events that provide observations about an object to be inferred by maximizing their expected value. [8, 9, 10]. We proposed a new measure to evaluate the value of a query (event), called *information gain rate*, which computed the expected amount of information to be obtained per unit of time about human intent.

- *Compact representations of human intent for robotic navigation.* For robotic navigation tasks, we modeled the human intent as a desired path to be followed by the robot. To allow desired paths to be communicated quickly and reliably using low-fidelity input mechanisms, we constructed two different representations of desired paths. First, we represented desired paths as strings of symbols from an ordered symbolic language [2, 4]. This representation depended on a probabilistic language model recovered from human-demonstrated paths, which used a new approach we developed to represent arbitrary paths as strings of symbols [11]. Second, we represented desired paths as local geodesics, which (locally) minimized a cost function recovered from human-demonstrated paths [12, 3]. We showed that the topology (i.e., the path-homotopy class) of a local geodesic originating from a fixed point in a polygonal domain (with obstacles) can be specified using only its length, and a real-valued angle [12]. This representation allowed humans to specify desired paths by communicating only a real-valued angle [3].

- *Efficient EEG-based BMIs for robotic navigation using optimal feedback policy.* We developed two EEG-based brain-machine interfaces that allowed human users to specify desired paths for robots as quickly and as reliably as possible using queries synthesized by the optimal feedback policy. See Figure 1.2. The first interface enabled users to navigate

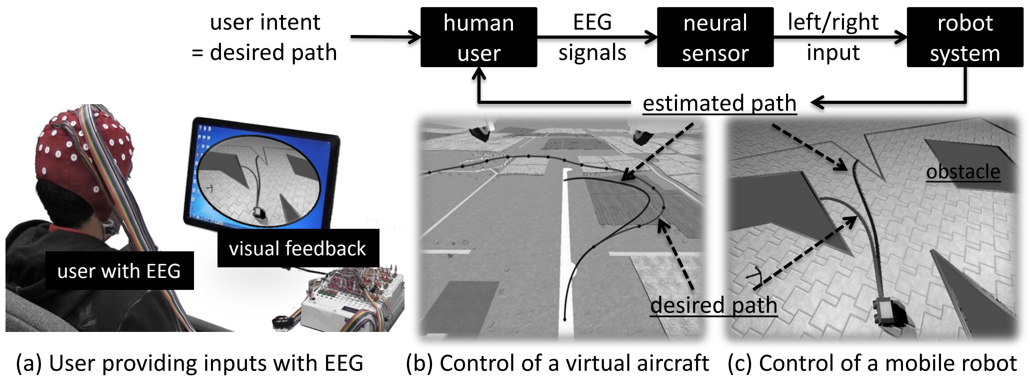


Figure 1.2: Illustration of the two EEG-based brain-robot interfaces developed as part of this thesis. Users of these interfaces provide input commands with EEG [see (a)] in response queries presented via visual feedback in order to control a simulated aircraft [see (b)], or to control a virtual mobile robot indoors [see (c)].

a simulated aircraft flying at a fixed speed and altitude over smooth paths that were represented as strings of symbols [4]. To our knowledge, reliable EEG-based control of a simulated aircraft was not possible with the existing interfaces. The second interface enabled users to navigate a mobile robot in a virtual indoor environment over paths that were represented as local geodesics [3]. To our knowledge, this interface provided superior performance over existing EEG-based interfaces for indoor navigation in terms of time to complete the navigation tasks and the quality of paths followed by the robot.

- *An efficient EEG-based BMI for text entry using active inference policy.* We developed a brain-machine interface that allowed human users to specify text commands with inputs obtained from steady-state visually-evoked potentials in EEG at a rate twice as fast as they would using prior state-of-the-art text entry interfaces working with the same input mechanism. This performance improvement was achieved using the active inference policy that took into account a language model, and expected accuracy and latency of inputs. To our knowledge, across EEG-based text entry interfaces, our interface was first to take into account expected accuracy and latency of inputs in adaptive selection of queries for identifying desired characters.

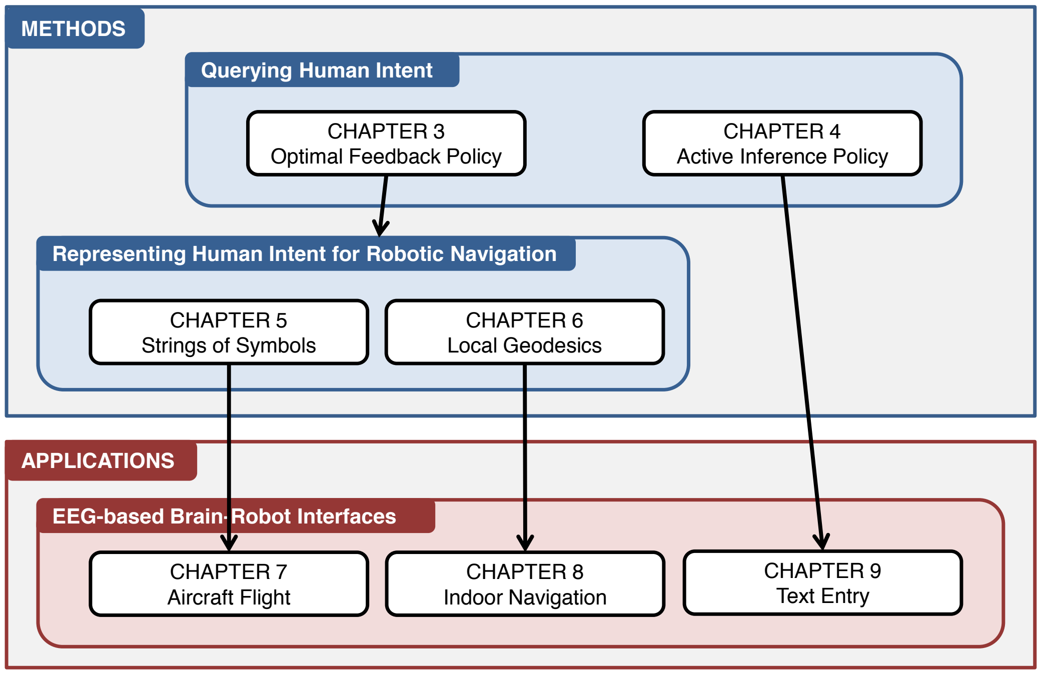


Figure 1.3: The organization of the thesis, showing the dependencies between the chapters.

1.4 Thesis Organization

The remainder of this thesis is organized as illustrated in Figure 1.3, and as described below.

- **Chapter 2** provides an overview of EEG-based brain-machine interfaces. In particular, it describes existing methods for obtaining noisy discrete input commands from EEG, and existing interfaces for human control of robots and for text entry.

- **Chapter 3** describes the use of optimal feedback communication to construct *the optimal feedback policy* for querying human intent. In particular, it explains the optimal feedback communication scheme called "posterior matching", designed for transmission of a message point between two computational agents, and shows its applicability to human control of robots over discrete noisy channels with high latency.

- **Chapter 4** describes the use of Bayesian active inference to construct *the active inference policy* for querying human intent. In particular, it defines a measure called “information gain rate” to evaluate the value of a query.

- **Chapter 5** describes our representation of desired paths as strings of symbols from an ordered symbolic language, where symbols correspond to path primitives. In particular, it provides an algorithm for representing arbitrary paths as strings of symbols, which facilitates the learning of a probabilistic language model from human-demonstrated data, and an approach for encoding strings as message points.

- **Chapter 6** describes our representation of desired paths as local geodesics with respect to a cost function that takes into account proximity to obstacles. In particular, it provides an algorithm for encoding local geodesics as message points.

- **Chapter 7** describes our EEG-based brain-robot interface that allows human users to fly a simulated aircraft moving at a fixed speed and altitude with binary input commands obtained from EEG. This interface uses the optimal feedback policy (Chapter 3) to query desired paths, and represents desired paths as strings of symbols (Chapter 5).

- **Chapter 8** describes our EEG-based brain-robot interface that allows human users to navigate a mobile robot in a virtual indoor environment with binary input commands obtained from EEG. The interface uses the optimal feedback policy (Chapter 3) to query desired paths, and represents desired paths as local geodesics (Chapter 6).

- **Chapter 9** presents our EEG-based interface for text entry that allows human users to specify text commands efficiently with discrete input commands obtained from EEG. This interface uses the active inference policy (Chapter 4) to query desired characters.

- **Chapter 10** concludes the thesis with a discussion of the methods and applications presented here, and with directions for future work.

Chapter 2

Survey on Brain-Machine Interfaces

2.1 Overview

A brain-machine interface (BMI) is a direct communication pathway between a human user and an external device [13, 14, 15]. This pathway allows human users to control external devices with their thoughts.

The primary reason brain-machine interfaces are developed is that these interfaces might improve the quality of life of individuals with spinal cord injuries. In United States alone, there are approximately 300,000 people with such injuries, and about ten percent of these people do not have voluntary motor function [16]. Brain-machine interfaces developed over the past decade have led to promising results like enabling tetraplegic human patients to control prosthetic devices and wheelchairs [17, 1].

A brain-machine interface has four principle components: (1) a neural sensor that measures the user's brain activity, (2) a decoding algorithm that maps these measurements to input commands for the device, (3) an external device that executes the user's intent, and (4) a sensory mechanism that provides feedback to the user [18, 15, 19]. These components are illustrated in Figure 2.1 and described below.

The neural sensor. The measurements of the brain activity may come from non-invasive sensors like an electroencephalograph (EEG) that observes the gross electrical activity of many neurons [20], or a functional near-infrared spectroscope (fNIRs) that observes blood oxygen concentration levels in the brain [21]. These measurements may also come from invasive sensors like an electrocorticograph (ECoG) that measures electrical activity from cortical surface [22], or a circuit that can observe ensemble spiking of individual neurons [23, 24, 25]. Invasive sensors may provide higher signal to noise ratio

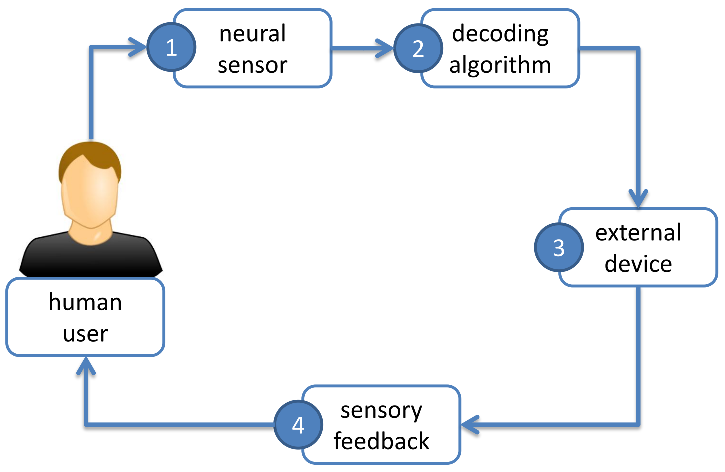


Figure 2.1: The components of a brain-machine interface.

than non-invasive sensors [18]. But, their use in practice is very limited due to the clinical procedures required to place or maintain the sensory circuit inside the scalp [18]. EEG is the most widely used non-invasive sensor for measuring brain activity in BMIs [26].

The decoding algorithm. The signals acquired by the neural sensor are translated by the decoding algorithm to input commands for the device. The first step in decoding often involves extraction of features that reflect the user's intent [18]. For example, features might be amplitudes of evoked potentials in EEG. The second step is then to map these features into commands that are used to operate the external device. The input commands might be either discrete, indicating a choice from a set of possible choices, or continuous, indicating a real value in a particular real interval. When the input commands are discrete, the decoding algorithm is also called a classifier. The decoding can be synchronous, i.e. occurring in fixed time intervals, or asynchronous, i.e., occurring after a variable amount of time, for ex., when a feature value exceeds a certain threshold [15]. Asynchronous interfaces are also called self-paced because the user may have the control over when to initiate a mental task that corresponds to an input command, such as imagination of left-hand motor imagery to provide a "turn left" command.

The external device. The devices operated in brain-machine interfaces infer the user's intent from input commands produced by the decoding algorithm. For example, the device can be a graphical computer interface that displays sentences typed by the user to communicate with others, or a cursor positioned by the user to select a desired item from a set of items on the screen. The device can be a virtual system such as an avatar in a computer game, or a mobile robot simulated in a graphical environment. The device can be a physical system such as a powered wheelchair that the user can use to navigate, or a robotic manipulator that the user can use to pick up objects.

The sensory feedback. The sensory feedback may provide the state of the external device as well as the stimuli necessary for the human user to perform particular mental tasks. For example, the state of the device may be the current position of a mobile robot in an interface for robotic navigation, or the characters spelled so far in an interface for text entry. The stimuli may be a set of screen objects flashing steadily at different frequencies. Such stimuli, used in SSVEP-based BCIs, allow users to provide input commands by selectively attending to one of the objects [15]. The sensory feedback often take one or more of the following forms: visual, auditory feedback, or tactile. Visual feedback is by far the widely used form. It can be provided by a graphical display or by observation of the external device through physical coupling. Auditory feedback can inform the user about the state of the device or stimuli by playing auditory cues. In tactile feedback, the user observes a force applied to their body, which might encode the state of the device or stimuli.

2.2 Our Scope in Brain-Machine Interface Design

The framework presented in this thesis provides a systematic approach to the design brain-machine interfaces that decode brain activity into discrete input commands and that use visual feedback to inform the user about the state of the device or stimuli.

In this chapter, we restrict our scope to EEG-based BMIs because EEG is the most widely used neural sensor and recorded EEG activity might be decoded into discrete input commands using many of the established paradigms for EEG. These paradigms determine how the user communicates with the device such as which mental tasks are performed by the user, and how recorded EEG activity can be mapped to input commands. We further restrict our scope to BMIs that use visual feedback because visual feedback is by far the most widely used form of feedback, and humans can interpret most visual stimuli correctly.

The rest of this chapter provides a survey of existing BMIs that fall under our scope. Section 2.3 describes the major paradigms used to obtain discrete input commands in EEG-based BMIs. Section 2.4 describes existing BMIs for human control of robotic systems. Section 2.5 describes existing BMIs for text entry. Although interfaces for text entry have not been considered for human control of robots, they might be used by human users to specify a text which might then be interpreted as a command for the robot to execute a desired task.

2.3 Obtaining Discrete Noisy Inputs in EEG-based Interfaces

Brain-machine interfaces can be operated with many paradigms, such as by having the user perform left- or right-hand motor imagery to provide a binary input to the external device. These paradigms determine how the user communicates with the device through EEG. For example, they might specify the mental tasks that must be performed by the user to generate decodable brain activity, or they might specify the stimuli that must be presented to the user to generate specific brain potentials.

In this section, we describe four widely-used paradigms for obtaining discrete inputs through EEG: motor-imagery, P300, SSVEP, and ErrP. These paradigms have been used to enable a wide range of tasks that include control of robotic systems (see Section 2.4), and communication with text entry (see Section 2.5).

The performance of these paradigms is often measured using one or more of the following metrics:

- Overall accuracy that determines the average probability that the classifier will produce the correct input command, i.e., the input that corresponds to the mental task being performed by the user. It can be computed from a set of trials as the fraction of correct input commands to the number of trials.

- Overall latency that determines the average time it takes to obtain an input command after the onset of a mental task.
- Information transfer rate (ITR) that determines the number of reliable bits communicated per trial or per second. Let p_a be the overall accuracy, d be the overall latency, and N be the number of possible input commands. Then, ITR per trial is

$$ITR^{trial} = \log N + p_a \log p_a + (1 - p_a) \log \frac{(1 - p_a)}{N - 1} \tag{2.1}$$

bits, and ITR per second is

$$ITR^{\text{sec}} = \frac{ITR^{\text{trial}}}{d}. \tag{2.2}$$

Motor-imagery paradigm. In this paradigm, the user performs mental tasks by imagining a motor action such as left-hand, right-hand, tongue or foot movement [27, 28, 29]. A visual stimulus might be presented to help the user determine the motor-imagery correlated with their intent. In order to classify the motor-imagery performed by the user from recorded EEG signals, a decoding algorithm is trained using EEG data labeled with ground-truth inputs. The performance of decoding may vary significantly among subjects, and decoding may not be possible for about 20% of potential users [30]. This paradigm can be used to obtain a few number of distinct inputs (usually 2,3 or 4). The average performance of this paradigm has been reported in [31] as 0.38 bits of ITR per second, and about 1.5 seconds of overall latency.

P300 paradigm. In this paradigm, the interface presents a set of choices to the user and highlights each choice successively in a random or infrequent order [32, 33, 34]. The user is instructed to focus their attention to their desired choice. When the user's desired choice is highlighted, the brain elicits a strong potential, called P300, with a peak occurring after about 300ms. The decoding algorithm can detect this potential from recorded EEG signals, and use the observed timing of the peak to determine the user's desired choice. This paradigm can be used to obtain a large number of distinct inputs, but the overall latency grows linearly with the number of possible inputs. The average performance of this paradigm has been reported in [31] as 0.47 bits of ITR per second.

SSVEP paradigm. In this paradigm, EEG signals are used to extract steady-state visually-evoked potentials (SSVEP). These potentials are observed over the visual cortex as natural responses to steadily flashing stimuli [35, 36]. In this paradigm, the interface presents a set of choices, where each choice is associated with a visual stimulus flashing at a unique frequency. The user's desired choice can be obtained with little or no training by analyzing the characteristics of the observed SSVEP responses in EEG. This paradigm can be used to obtain a large number of distinct inputs, but increasing the number of possible inputs, i.e., using more stimulation frequencies, might reduce the information transfer rates. The number of different stimulation frequencies are limited if stimuli are presented on an off-the-shelf monitor, but using custom-built LED-based devices several dozens of distinct inputs might be obtained [37, 38]. The average performance of this paradigm has been reported in [31] as 0.44 bits of ITR per second, and about 2.10 seconds of overall latency.

ErrP paradigm. This paradigm uses error-related potentials (ErrP) that are elicited when a subject observes an incorrect feedback. A BMI can utilize these potentials to detect if the user has made an incorrect choice [39], or to detect if the external device took an incorrect action [40]. In [40], this paradigm provided a binary input with overall accuracy of 0.80, and overall latency of 0.5 seconds. Despite its limitation to binary inputs, this paradigm can be used to enhance the throughput of other paradigms [39].

2.4 EEG-based Interfaces for Robotic Control

BMIs for human control of robots typically function in one of two different ways: process control and goal selection [14, 41]. In process control, measurements of brain activity are used to specify an immediate action to be taken, such as moving a cursor to the left or to the right. In goal selection, measurements of brain activity are used to specify the desired output after a sequence of actions, such as the location at which the cursor should end up. Many existing EEG-based BMIs use one of these two approaches for human control of robots.

2.4.1 Interfaces based on process control

Many existing EEG-based BMIs use process control for movement tasks, such as for navigating a wheelchair [42, 43], or a mobile robot [44, 45, 46]. A problem with this strategy is that it tends to produce erratic motion due to the direct mapping from noisy measurements of brain activity to control inputs. Methods of shared control have been proposed as a way to reduce this problem [47, 42, 43, 45]. With shared control, movement is determined by "averaging" inputs produced by the BMI with inputs that might have been expected given a prior model.

The works in [44, 42, 43] used three mental tasks to drive a mobile robot, a physical wheelchair, and a virtual wheelchair, respectively. These mental tasks corresponded to three input commands: "left", "right", and "forward". These inputs were filtered based on the context (e.g., proximity to obstacles) before being mapped to motor commands for the wheelchair. The actual speed and direction of the robot were determined by taking into account the decoding algorithm's belief on each command, and the prior probabilities that penalized the commands that move the robot close to the obstacles. Although shared control improved performance, erratic motion still occurred when user inputs were erroneous or in conflict with the prior model used in the filter.

In [46], a mobile robot was navigated with ErrP paradigm. Based on the robot's state, the interface proposed the user one of the following five possible actions: "stop", "left", "right", "forward", "u-turn". The ErrP paradigm was used to detect whether or not the proposed action matched the user's desired one.

In [45], a helicopter was navigated in a 3D virtual environment using 4-class motor-imagery paradigm, where the inputs mapped to left/right turns or up/down movements of the virtual helicopter. This mapping used a cone of guidance to avoid collisions with obstacles.

In [48], a simulated humanoid was navigated with SSVEP using three steering actions (“left”, “right”, and “stop”). Gaussian processes were used to learn a probabilistic model over actions from the demonstrated trajectories. This learnt model was used to auto-complete trajectories based on the uncertainty in the probabilistic model.

The works in [1], and [49] controlled an avatar in a virtual environment with 2-class motor imagery, and 3-class SSVEP paradigm, respectively.

2.4.2 Interfaces based on goal selection

EEG-based BMIs based on goal selection allow human users to choose from a set of destinations for a wheelchair [50, 51], or objects to be picked up by a humanoid robot [52]. A problem with this strategy is that the set users can choose from is restricted to the goals determined by the designer, and the user has no control over how the robot achieves the selected goal.

For instance, Rebsamen et al. [51] enabled a human user to drive a powered wheelchair to a destination selected by the user from a list of locations. These destinations were restricted to a given list (bath, bed, office, etc.) and users did not have any control over paths followed by the wheelchair to reach the selected destination.

In [52], the interface showed pictures of the objects that could be picked up by a humanoid robot, and the user manipulated the robot by selecting one of these objects.

2.4.3 Interfaces based on hybrid strategies

In an effort to address the problems with process control and goal selection, the BMI proposed by Iturrate et al. [53] used a hybrid strategy for robotic wheelchair navigation. In this strategy, measurements of brain activity were used to specify a subgoal that indicated the next location to be moved by the robot. The user selected from a finite set of locations that were visible to the robot, and the robot then moved to the selected location autonomously. By repeating this process, in effect, the user specified a desired path step by step.

2.5 EEG-based Interfaces for Text Entry

Table 2.1: Example state-of-the-art spellers across several input paradigms.

Input Paradigm	Text Speller	Alphabet Size	Language Model	Spelling Rate
P300 (offline)	Speier et al. 2011	36	char-based	7.3 cpm
P300 (online)	Townsend et al. 2010	72	none	3.66 cpm
Motor-imagery	Blankertz et al. 2007	30	char-based	up to 6.7 cpm
SSVEP	Volosyak et al. 2011	32	word-based	7 to 10 cpm
Gaze (discrete)	Majaranta et al. 2009	36	none	20 wpm
Gaze (continuous)	Ward et al. 2002	27	char-based	up to 25 wpm

In this section, we describe existing EEG-based spellers that use one of the paradigms described in Section 2.3 to obtain noisy discrete inputs from the user. The performance of a text speller is often measured by its spelling rate, which can be defined as the average number of characters spelled per minute (cpm) or the average number of words spelled per minute (wpm). Table 2.1 shows the spelling rates of example state-of-the-art spellers for several input paradigms. Here, we see that EEG-based input paradigms (P300, motor-imagery, and SSVEP) allow spelling rates of up to 10 cpm. On the other end, spellers based on gaze, which can be measured by an eye tracker, can achieve spelling rates of up to 25 wpm.

2.5.1 Spellers using P300

P300 paradigm is the most widely used paradigm for text spelling across all EEG-based interfaces [54, 55, 56]. Successful uses of P300 spellers by disabled subjects, especially those that were diagnosed with amyotrophic lateral sclerosis (ALS), have been reported in several studies [57, 58, 59, 55].

In the original P300 speller design [32, 33], characters are placed into a 6 by 6 matrix like the one shown in Figure 2.2. The interface illuminates row and columns of the matrix successively for a fixed duration in random order, while the user is gazing or focusing attention at a desired character. The illumination of the gazed character, which corresponds to a rare event in the "oddball" paradigm, elicits the P300 potential after about 300ms. After all rows and columns are illuminated once, the interface can determine the gazed character with some chance of error. Illumination of the rows and columns

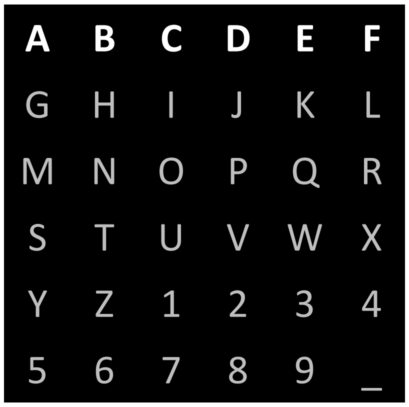


Figure 2.2: The classical layout of characters in a P300 speller. The first row is shown illuminated.

multiple times decreases the error but increases the time to spell a character.

Illumination of the rows and columns, called the RC paradigm, is the most common stimulus presentation paradigm in P300 spellers [55]. Two other paradigms are the single cell (SC) paradigm, and the checkerboard (CB) paradigm. In the SC paradigm, the characters are illuminated one by one in a random order. In the CB paradigm, characters in the original matrix are divided into two halves, one half containing the characters on the "white" cells, and the other half containing the characters on the "black" cells [60]. White and black cells are determined using a checkerboard style coloring of the original matrix. The characters to be illuminated are chosen from either the first half or the second half. This avoids the "adjacency-distraction problem", since characters that are adjacent in the original matrix will not be illuminated at the same time. Townsend et al. [60] compared these three stimulus presentation paradigms, and reported that CB performs better than RC, which performs better than SC.

In addition to the stimulus presentation paradigm, many other factors affect the performance of a P300 speller. These factors include stimulus duration, inter-stimulus interval, matrix size, stimulus intensity, and many others [55]. One important factor is the signal processing and the decoding process that are used to detect P300 responses. There has been a significant amount of work in the past decade to improve the detection accuracy of P300 responses [55]. Recent works focused on improving the spelling rates by taking into account the online performance in P300 detection [61, 62, 63], probabilistic language models [64, 65, 66, 67, 63, 68], and by using adaptive protocols that decide which characters get illuminated in which order [66, 67].

Recent works also aimed to reduce or eliminate the off-line training for P300 detection process by using semi-supervised or unsupervised algorithms that took into account unlabeled online data [69], offline knowledge available from other subjects [70], as well as probabilistic language models [71].

2.5.2 Spellers using motor-imagery

In this section, we describe three motor-imagery based spellers developed by different research groups in the last decade.

The interface by Scherer et al. [72] uses three motor-imagery tasks: imagination of left-hand, right-hand, and foot movement. Users perform one of these tasks continuously until a desired event occurs. 26 characters are separated into two sets and a few characters from each set are shown on the left and right side of the screen in alphabetical order. The user imagines foot movement to scroll the character bottom up so that characters at the top disappear and new characters appear at the bottom. Whenever the user's desired character moves to the top-most position on the screen, the user imagines left-hand motor imagery, if the desired character is on the left side, or right-hand motor imagery, if the desired character is on the right side, respectively. The desired character is spelled whenever a 1D cursor controlled by continuous left- and right-hand motor imagery classification exceeds a user-specific threshold. With this interface, a mean spelling rate of about 2 characters per minute have been obtained in experiments [72].

The Hex-o-Spell interface by Blankertz et al. [73] allows users to spell in two steps by correctly timing two distinct motor-imagery events such as right-hand imagery and foot imagery. In the first step, 30 characters are placed into six clockwise-ordered hexagons, each containing five characters. A cursor points to the top-most hexagon in the beginning. Users rotate this cursor clockwise until it points to the hexagon containing their desired character by providing continuous right-hand motor imagery. Users then provide continuous foot motor imagery until a certain amount of time to select the target hexagon. In the second step, the characters in the target hexagon are distributed into five hexagons, each containing a single character. This distribution is based on a probabilistic language model that assigns likelihoods to characters based on the history of characters spelled so far. In particular, in the second step, the characters that are more likely are assigned to the hexagons that are closer to the cursor's initial position. Similar to the first step, users provide the motor-imagery events to choose the target hexagon that contains the character to be spelled. With this interface, spelling rates of up to 7.6 characters per minute have been reported in [73].

The interface by D'Albis et al. [74] uses four motor-imagery tasks: imagination of left-hand, right-hand, both-hands, and feet movement. Users perform one of these tasks to select one of the four targets displayed on the screen. Three of these targets contain subsets of characters, and the other target contains auxiliary commands such as an "undo" command that returns the interface to its previous state. Users can specify a character in three steps by selecting the target that contains the desired character in each step. At first, 27 characters are distributed across the three targets, each containing 9 characters. In the second step, 9 characters in the selected target are distributed across the three targets, each containing 3 characters. In the third step, each target contains a single character. These three steps allow the interface to identify and spell the desired character. A probabilistic language model is used to disable the characters that are unlikely to be spelled in the current context. This might improve spelling rates, because disabling characters might result in having only one enabled character in the target selected in the first or second step. Whenever this occurs, the enabled character might be spelled without any further steps. The interface also has a word prediction mode, where the interface shows the three words that are most likely in the current context. The user can select a word among these words by selecting an auxiliary command first, and then the target associated with the desired word. With this interface, three subjects obtained a spelling rate of between 2 and 3 characters per minute in [74].

2.5.3 Spellers using SSVEP

In this section, we describe three state-of-the-art SSVEP spellers. The speller by Cecotti [75] and the speller by [76] use an off-the-shelf graphics monitor to display 5 different stimulation frequencies. The speller by Hwang et al. [38]

uses a custom-built LED-based hardware to display 30 different stimulation frequencies.

The interface by Cecotti [75] allows users to specify a desired character using a decision tree with three levels. Initially, 27 characters are presented in three groups which consist of nine characters. These groups are displayed at the left, middle, and right of the screen and are associated with stimuli steadily flashing at different frequencies. Following the user's initial choice of a group, the nine characters are divided into three subgroups, each with three characters apiece. This process repeat one more time, allowing for the selection of a specific character. In addition to three stimuli used to select the groups of characters, two other stimuli are used to issue a "delete character" command that erases the previous character spelled, and a "previous action" command that returns to the previous selection or moves up in the tree. With this speller, they achieved a spelling rate of 5.51 characters per minute with an overall accuracy of 0.92 on average in experiments [75].

The Bremen speller [76] uses a custom-designed grid containing 32 characters, and five stimuli steadily flashing at different frequencies. Users specify a desired character by navigating a cursor starting at the center of the grid to the grid location of their desired character. Four input commands are used to navigate the cursor left, right, up, down in the grid, and a fifth input command is used to spell the character at the cursor location. The number of inputs required to spell a character depends on the distance of the character from the center of the grid. To have user specify more frequent characters with less number of inputs, the characters are placed in the grid at the design time based on their frequencies in English texts. A demographics study on the Bremen speller has shown that it could be used by a large population of able-bodied users [77]. In a more recent version of the interface, users are given an option to select a word from a short list of candidate words, which are chosen based on their frequencies in the prior use of the interface [78]. Although this word prediction mechanism has provided marginal improvements in spelling rates, some subjects reported that they would prefer not to have this mechanism. Experiments with the Bremen speller show that it can provide a spelling rate of between 7 to 10 characters [76, 78].

The interface by Hwang et al. [38] uses a custom-built hardware that generates 30 different stimulation frequencies. Users specify a desired character by attending to the unique stimulus associated with their desired character. Although the one-to-one mapping between stimuli and characters allows users to spell in one step, the use of many frequencies cause a decrease in overall accuracy. Therefore, users need to correct for errors more frequently than the spellers achieving higher overall accuracy with the use of less number of frequencies. Despite this, they achieved a spelling rate of 9.4 characters per minute with an overall accuracy of 0.88 on average in online spelling experiments [38].

Part II

METHODS

Chapter 3

Querying Human Intent using Optimal Feedback Policy

3.1 Introduction

This chapter presents a feedback information-theoretic approach, based on *optimal feedback communication* [6, 7], to querying human intent for interfaces that can only obtain noisy and discrete inputs at high latency from a human user, and that can provide the human user visual feedback. In this approach, we model the interface as a communication channel with feedback, with which the human user can communicate their intent. Its appeal is that, there is a probably optimal communication scheme for transmission of information over such a channel. We refer to the policy that determines how the human user provides inputs to the communication channel with feedback as the *optimal feedback policy*.

In Section 3.2, we describe the optimal communication scheme, called "posterior matching". This scheme is designed for transmission of information represented as message points between two computational agents. In Section 3.3, we show that this scheme can be used by the robot to construct optimal policies for querying human intent. It is important to emphasize that not all communication schemes might be implemented efficiently by human users to transmit their intent, such as the schemes based on error-correcting codes. However, we will see that the policies generated under the posterior matching scheme for transmission over discrete noisy channels can be used to query human users under certain conditions.

One limitation of this approach is that it assumes a particular structure on the information to be transmitted. In particular, it represents this information as a message point, i.e., real number uniformly distributed in a closed interval. In order to use this policy for querying human intent, we need to encode human intent as a message point. For robotic navigation tasks, this

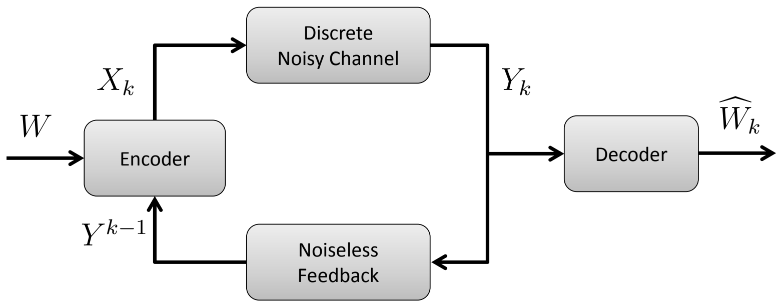


Figure 3.1: Communication over a discrete noisy channel with noiseless feedback

encoding problem will be addressed in Chapters 5 and 6.

3.2 Optimal Feedback Communication of Message Points

3.2.1 The feedback communication problem

We model the information to be transmitted between two computational agents as a *message point*, which is defined by a random variable W uniformly distributed in the unit interval $[0, 1)$. We call the agent sending the information as the *channel encoder*, and the agent receiving the information as the *channel decoder*. A *discrete noisy channel* is a system using which the channel encoder sends symbols from an input alphabet $\mathcal{X}$, and the channel decoder receives symbols from an output alphabet $\mathcal{Y}$. Such a channel consists of a *probability transition matrix* $P_{Y|X}$, where $P_{Y|X}(y|x)$ defines the probability of receiving the output symbol $y \in \mathcal{Y}$ given that the input symbol $x \in \mathcal{X}$ was sent. The channel is called *memoryless* if the channel output depends only on the current channel input and conditionally independent of previous channel inputs. Here, we consider only the channels that are memoryless. In a *channel with feedback*, the channel outputs become immediately available without noise to the channel encoder.

Communication over a discrete noisy channel with feedback is illustrated in Figure 3.1. The channel's $k-$th input is represented by a random variable $X_k \in \mathcal{X}$, and its realization is denoted x_k. The channel's $k-$th output is represented by a random variable $Y_k \in \mathcal{Y}$, and its realization is denoted y_k. We denote the past output variables $Y_1, Y_2, \ldots, Y_k$ by Y^k, and similarly the past output instances $y_1, y_2, \ldots, y_k$ by y^k. Without loss of generality, if $\mathcal{X}$ contains N symbols, we set $\mathcal{X} = \{0, 1, \ldots, N-1\}$.

A *communication policy*, also called a *channel code*, consists of a sequence of encoding functions and a sequence of decoding functions. The encoding function f_k determines how the encoder chooses the input x_k given the output values received via feedback, i.e.,

$$X_k = f_k(W, Y^{k-1}). \tag{3.1}$$

The decoding function g_k determines how the decoder computes an estimate of the message point, denoted $\widehat{W}_k$, from k channel outputs, i.e.,

$$\widehat{W}_k = g_k(Y^k). \tag{3.2}$$

The decoding error e_k after k outputs is computed as

$$e_k = \left| W - \widehat{W}_k \right|. \tag{3.3}$$

In addition to generating a point estimate $\widehat{W}_k \in [0, 1]$, the decoder can generate an interval estimate $\Delta_k \in \xi$, where ξ is the set of all subintervals $[a, b], a \leq b$ in $[0, 1]$. Δ_k can be computed from the posterior distribution over W after receiving k channel outputs. We denote the probability density function (PDF) of this distribution as $P_{W|Y^k}(w|y^k)$. Let $P_k\{[a, b]\}$ be the probability concentrated on the interval $[a, b]$ of the posterior $P_{W|Y^k}(w|y^k)$, i.e.,

$$P_k\{[a, b)\} = \int_{w=a}^{w=b} P_{W|Y^k}(w|y^k)dw. \tag{3.4}$$

Given a target error probability δ_k, the interval estimate Δ_k is the smallest subinterval that has a posterior probability concentration of at least $1 - \delta_k$, i.e.,

$$\Delta_k = \arg\min_{\Delta \in \xi, P_k\{\Delta\} \geq 1-\delta_k} |\Delta|, \tag{3.5}$$

where $|\cdot|$ denotes the width of an interval, i.e., $b - a$ for $[a, b]$.

3.2.2 Conditions for optimality

A transmission rate R is achievable if the resulting estimate $\widehat{W}_k$ computed after k channel outputs satisfies

$$\lim_{k\to\infty} P\left(\left|W-\widehat{W}_k\right| > 2^{-kR}\right) = 0, \tag{3.6}$$

which states that the decoding error decades exponentially fast with the number of channel outputs. The capacity of the channel gives us an upper bound on the achievable rates. The capacity, denoted C, is defined as

$$C = \max_{P_X} I(X;Y), \tag{3.7}$$

where the maximum is taken over all possible input distributions P_X. The input distribution that maximizes the capacity is called the capacity-achieving distribution and denoted P_X^*. A communication policy is optimal if any rate $R \leq C$ is achievable.

A necessary condition for optimality is to maximize $I(W;Y^k)$, i.e., the mutual information between the message point W and the channel outputs observed till time k, Y^k [79, 80, 7]. This mutual information can be bound as follows

$$\begin{aligned}
I(W;Y^k) &= H(Y^k) - H(Y^k|W) && (3.8)\\
&= H(Y^k) - \sum_{i=1}^{k} H(Y_i|Y^{i-1},W) && (3.9)\\
&= H(Y^k) - \sum_{i=1}^{k} H(Y_i|X_i) && (3.10)\\
&\leq \sum_{i=1}^{k} H(Y_i) - \sum_{i=1}^{k} H(Y_i|X_i) && (3.11)\\
&= \sum_{i=1}^{k} I(X_i;Y_i) && (3.12)\\
&\leq kC, && (3.13)
\end{aligned}$$

where (3.10) follows because X_i is a function of Y^{i-1} and W, (3.11) follows because conditioning reduces entropy, (3.12) follows from the definition of mutual information, and (3.13) follows from the definition of capacity. The equality $I(W;Y^k) = kC$ is established when

- Y_i's are independent, i.e., $I(Y_i|Y^{i-1}) = 0$ [then (3.11) becomes equality],
- X_i's are distributed according to the capacity-achieving distribution P_X^* [then (3.13) becomes equality].

We will see that these two properties are satisfied by the optimal communication scheme that will be described in Section 3.2.3.

3.2.3 Posterior matching scheme for optimal feedback communication

For constructing optimal communication policies for general memoryless channels with noiseless feedback, Shayevitz et al. [7] recently introduced a scheme termed "posterior matching". Posterior matching (PM) specifies the channel input at time k according to

$$X_k = f_k(W, Y^{k-1}) = F_X^{-1}(F_{W|Y^{k-1}}(W|Y^{k-1})), \tag{3.14}$$

where $F_X(x)$ is the input cumulative distribution function (CDF) for the channel (known a priori) and $F_{W|Y^{k-1}}(w|y^{k-1})$ is the posterior conditional CDF of W, and F_X^{-1} is the inverse CDF given by $F_X^{-1}(t) = \inf\{x : F_X(x) > t\}$. Applying F_X^{-1} shapes $F_{W|Y^{k-1}}(W|Y^{k-1})$ into the input distribution P_X, which is taken to be the capacity-achieving input distribution P_X^* for optimality. X_k is statistically independent of past channel outputs Y^{k-1}, and captures the information that is still missing at the decoder. Since the channel is memoryless, Y_k is independent of Y^{k-1} under the PM scheme.

For discrete memoryless channels, the encoding function f_k in (3.14), given by the PM scheme, can be remarkably simplified. This simplification will be very useful in Section 3.3, since a human user will apply the encoding function. The encoding function in (3.14) uses $F_{W|Y^{k-1}}(w|y^{k-1})$, the posterior conditional CDF of W, which is also computed at the decoder. In the PM scheme, the encoder only needs a sufficient statistics of this CDF to determine the next channel input. This suggests the use of the sufficient statistic, denoted Z_k after k channel outputs, as the noiseless feedback to the encoder (see Figure 3.2).

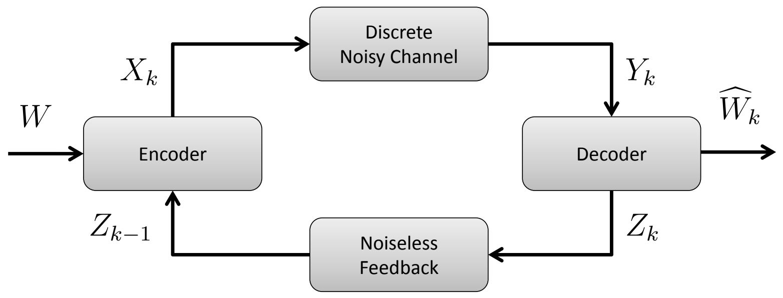


Figure 3.2: Communication over a discrete noisy channel with noiseless feedback using the posterior matching scheme. After receiving the $k-$th channel output, the decoder generates the feedback Z_k. Instead of Y^{k-1}, a sufficient statistic Z_{k-1} can be used as feedback by the encoder, without loss of optimality.

If the input alphabet contains N symbols, i.e., $\mathcal{X} = \{0, 1, , \dots, N-1\}$, then $Z_k = (Z_k^{(0)}, Z_k^{(1)}, \dots, Z_k^{(N-2)})$ is a random vector with $N-1$ random variables satisfying the constraint $0 \leq Z_k^{(0)} \leq Z_k^{(1)} \leq \dots \leq Z_k^{(N-2)} \leq 1$. Given the CDF of the input distribution, F_X, and the CDF of posterior of W, $F_{W|Y^{k-1}}(w|y^{k-1})$, Z_{k-1} is the random vector obtained by

$$Z_{k-1}^{(i)} = F^{-1}_{W|Y^{k-1}}(F_X(i)), \quad \forall i \in \{0, 1, \dots, N-2\}, \tag{3.15}$$

where $F^{-1}_{W|Y^{k-1}}(t)$ is the inverse CDF of the posterior. Intuitively, Z_{k-1} partitions the unit interval into N non-overlapping contiguous subintervals

$$[0, Z_{k-1}^{(0)}),\ [Z_{k-1}^{(0)}, Z_{k-1}^{(1)}),\ \dots,\ [Z_{k-1}^{(N-2)}, 1),$$

such that the probability concentrated on the $i-$th interval, $i = 0, \dots, N-1$, from left to right, is equal to $P_X^*(X = i)$.

Given Z_{k-1} as feedback, the encoder can determine the next input X_k using

$$\phi(W, Z_{k-1}) = \begin{cases} 0 & \text{if} \quad W \in [0, Z_{k-1}^{(0)}), \\ 1 & \text{if} \quad W \in [Z_{k-1}^{(0)}, Z_{k-1}^{(1)}), \\ \dots & \\ N-1 & \text{if} \quad W \in [Z_{k-1}^{(N-2)}, 1). \end{cases} \tag{3.16}$$

Intuitively, $\phi(W, Z_{k-1})$ returns the index of the subinterval that contains W. This can be used in place of the original encoding function f_k, at any time step k, i.e.

$$X_k = f_k(W, Y^{k-1}) = \phi(W, Z_{k-1}), \quad k \in \{1, 2, \dots\}. \tag{3.17}$$

Given the channel output y_k, the posterior distribution is updated using the Bayes' rule as

$$\begin{aligned} P_{W|Y^k}(w|y^k) &= \eta \cdot P_{Y|W,Y^{k-1}}(y_k|w, y^{k-1}) \cdot P_{W|Y^{k-1}}(w|y^{k-1}) && (3.18)\\ &= \eta \cdot P_{Y|X}(y_k|f_k(w, y^{k-1})) \cdot P_{W|Y^{k-1}}(w|y^{k-1}), && (3.19)\end{aligned}$$

where η is a normalizing constant.

3.2.4 Example: Optimal feedback communication over a BSC

Let us demonstrate the optimal feedback communication of a message point W over a binary symmetric channel (BSC). A BSC consists of the input and output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$, and a probability transition matrix $P_{Y|X}$ such that

$$P_{Y|X}(y|x) = \begin{cases} 1-\epsilon & \text{if} \quad y = x \\ \epsilon & \text{if} \quad y \neq x, \end{cases} \tag{3.20}$$

where ϵ is the crossover probability. The capacity achieving distribution for a BSC is $P_X^*(0) = P_X^*(1) = 0.5$. The CDF of this distribution is $F_X(0) = 0.5, F_X(1) = 1.0$. The feedback Z_{k-1} provided to the encoder under the PM scheme consists of a single random variable $Z_{k-1}^{(0)}$. We omit the superscript and use Z_{k-1} to denote $Z_{k-1}^{(0)}$. Using (3.15), we see that the optimal feedback is

$$Z_{k-1} = F^{-1}_{W|Y^{k-1}}(0.5), \tag{3.21}$$

which is simply the *median* of the posterior PDF $P_{W|Y^{k-1}}$. The $k-th$ channel input X_k is chosen as

$$\phi(W, Z_{k-1}) = \begin{cases} 0 & \text{if} \quad W < Z_{k-1} \\ 1 & \text{otherwise.} \end{cases} \tag{3.22}$$

After receiving the $k-$th channel output as $y_k = 0$, the decoder updates the posterior distribution over W as

$$P_{W|Y^k}(w|y^k) = \quad \eta \cdot \begin{cases} (1-\epsilon) \cdot P_{W|Y^{k-1}}(w|y^{k-1}) & \text{if } w < Z_{k-1} \\ \epsilon \cdot P_{W|Y^{k-1}}(w|y^{k-1}) & \text{otherwise,} \end{cases} \tag{3.23}$$

where η is a normalizing constant. This update increases the probability density of all points that are smaller than the median Z_{k-1}. The update for $y_k = 1$ is analogous.

An example application of the PM scheme for transmitting a message point $W = w^*$ over a BSC with crossover probability $\epsilon = 0.1$ is shown in Figure 3.3. At first, the posterior is uniform, and the feedback z_0 is the median at 0.5. Because w^* is smaller than Z_0, the encoder provides $x_1 = 0$. After observing a channel output $y_1 = 0$, the decoder updates the posterior, which increases the probability density of all points to the left of Z_0 and decreases the probability density of all points to the right of Z_0. Then, the decoder computes Z_1 as the median of $P_{W|Y_1}$. Upon receiving Z_1 as feedback, the encoder determines the next channel input and the process repeats. As more channel outputs are observed, the posterior PDF, roughly, concentrates on a smaller interval, and the decoding error decreases.

The results obtained with Monte Carlo simulations of the PM scheme for message point transmission over a BSC are shown in Figure 3.4. For the decoding function, we estimated the message point as the median of the posterior PDF, i.e., $\widehat{W}_k = Z_k$. As expected, the decoding error $e_k = \left|W - \widehat{W}_k\right|$ decayed exponentially with the number of channel outputs. The figure also plots the probability that the point estimate after k channel outputs, $\widehat{W}_k$, is within 5% of the randomly chosen message point W, i.e., $P_k\{[W - 0.025, W + 0.025)\}$.

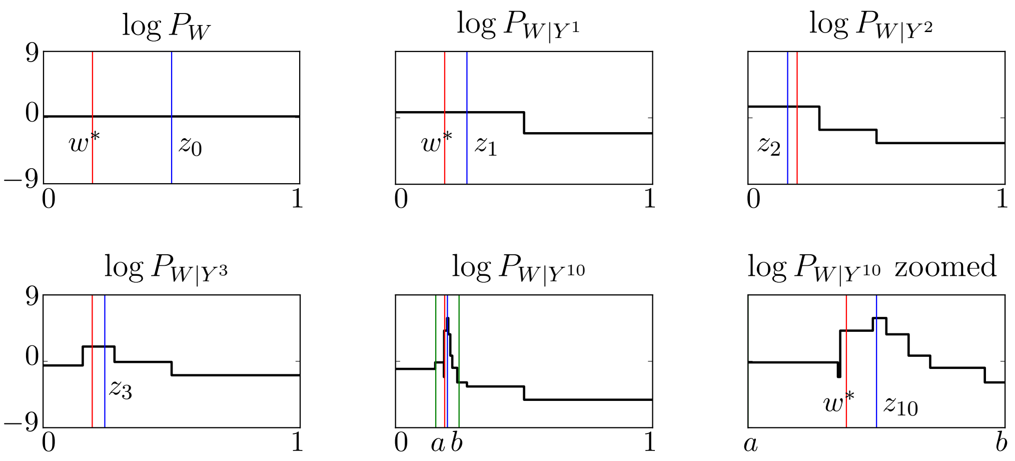


Figure 3.3: The evolution of the posterior distribution of W during a transmission of the message point w^*. All plots show the base-2 logarithm of the posterior PDF, i.e., $\log P_{W|Y^k}$. At first, the posterior P_W was uniform (hence $\log P_W$ was zero), and the feedback z_0 was the median at 0.5. Since, $w^* < z_0$, the encoder chose the next input to be $x_1 = 0$. Upon observing $y_1 = 0$, the posterior was updated using Bayes' rule to get the PDF $P_{W|Y_1}(\cdot|y_1)$ and the feedback z_1. This process was repeated until ten channel outputs were obtained. In this process, only the 7th channel output did not match the 7th channel input, i.e, $y_7 \neq x_7$. The 5th frame shows the posterior PDF at the end, the decoded interval Δ_{10} with a target error probability of 0.10 (marked by green lines, denoted $[a, b)$). Finally, the 6th frame shows the posterior PDF zoomed to this decoded interval.

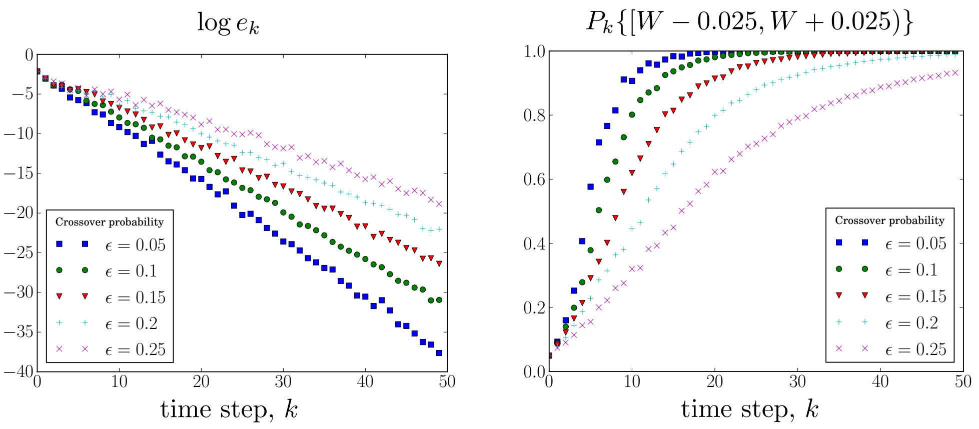


Figure 3.4: The results of Monte Carlo simulations of the PM scheme for transmission of a random message point $W \in [0, 1)$ over a BSC with five different crossover probabilities (see legend). For each time step $k = 0, \ldots, 49$, the mean decoding error (in $\log_2$), and the mean probability that the point estimate $\widehat{W}_k = Z_k$ is within 5% of the message point W are plotted in the left and right frame, respectively.

3.3 Querying Human Intent using Optimal Feedback Communication

In this section, first we cast the problem of designing interfaces for human control of robots as a communication problem for reliable transmission of human intent over a discrete noisy channel with noiseless feedback. Then, we show that the PM scheme for transmission of a message point, described in Section 3.2, can be used for transmission of human intent, provided that there is an invertible mapping from human intents to message points.

Interface design as a communication problem for transmitting human intent. We view the goal of the interface is to facilitate quick and reliable transmission of human intent, which we represent as a random object V, taking values from a set $\mathcal{V}$. We assume that human intent is distributed with a probability distribution P_V. We model the user's input mechanism as a noisy discrete channel, and we assume that this channel can provide noiseless feedback through the feedback mechanism. In this model, the human user acts as an encoder (essentially mapping the feedback to the channel input), and the robot acts as a decoder (essentially mapping the channel outputs to an estimate of the human intent V). The goal is to design a communication policy over the communication channel, with which the human intent can be transmitted to the robot with vanishing error probability. Let $\widehat{v}_k$ be the estimate generated by the robot after k channel outputs, and $V = v^*$ be the chosen human intent. We call a communication policy optimal if it achieves capacity and if $P(\widehat{v}_k \neq v^*) \to 0$ as $k \to \infty$.

Solving the communication problem using the optimal communication policy for transmitting a message point. The optimal communication policy described in Section 3.2 assumes that the message to be transmitted is a message point W distributed uniformly in the unit interval $[0, 1)$. The source-channel separation theorem tells us that the design of an optimal communication policy for transmitting an arbitrary message can be done in two stages [79]. The first stage produces a *source code*, which is an invertible mapping $\psi : \mathcal{V} \to [0, 1)$ from the space of possible human intents to points in the unit interval. Given the human intent is V, the message to be transmitted becomes $W = \psi(V)$. A source code is optimal if the average

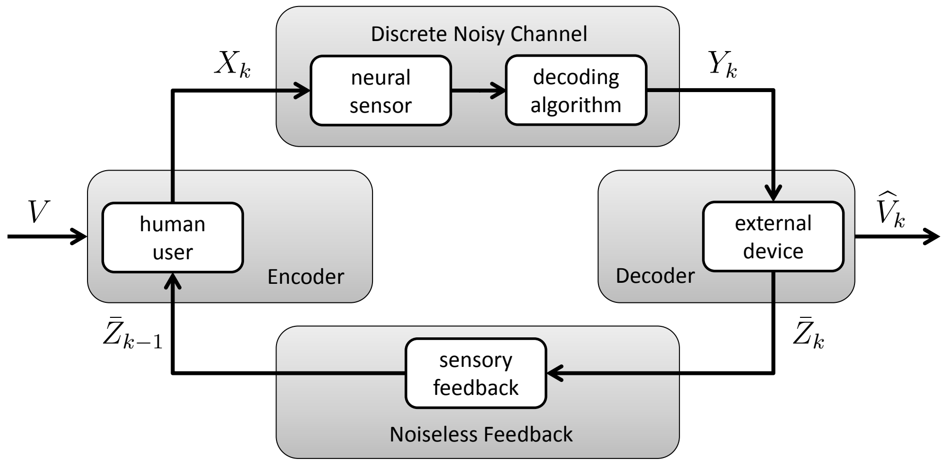


Figure 3.5: Our abstraction of brain-machine interface design as a feedback communication problem. The human users convey their intent $V = v^*$ to the external device through a discrete noisy channel with noiseless feedback.

number of bits used to represent W matches the entropy of V as given by the distribution P_V. The second stage produces a *channel code*, or a communication policy, for transmission of W over the underlying communication channel. In this stage, we can use the optimal channel code given by the PM scheme described in Section 3.2. If both the source code and the channel code are optimal, the resulting communication protocol will be optimal [79].

The resulting communication protocol for transmitting human intent under the PM scheme. Figure 3.5 illustrates the resulting communication protocol, in particular for the design of brain-machine interfaces. Let $\psi : \mathcal{V} \to [0, 1)$ be the invertible mapping described by an optimal source code. Recall that the encoder determines the next input X_k using the message point W and the feedback Z_{k-1}, i.e., $X_k = \phi(W, Z_{k-1})$. We show that the encoder can use a function $\bar{\phi}$ that depends on V instead of W to determine the next input, i.e.,

$$X_k = \phi(W, Z_{k-1}) = \bar{\phi}(V, \bar{Z}_{k-1}), \tag{3.24}$$

where $V = \psi^{(-1)}(W)$, and $\bar{Z}_{k-1}$ is the feedback consisting of items from $\mathcal{V}$, computed by applying the inverse mapping ψ^{-1} to the items of Z_{k-1}. In particular, recall that $Z_{k-1} = (Z_{k-1}^{(0)}, Z_{k-1}^{(1)}, \dots, Z_{k-1}^{(N-2)})$, where $0 \leq Z_{k-1}^{(0)} \leq$

$Z_{k-1}^{(1)} \leq \ldots \leq Z_{k-1}^{(N-2)} \leq 1$. Then, we define

$$\bar{Z}_{k-1}^{(i)} = \psi^{-1}(Z_{k-1}^{(i)}), \tag{3.25}$$

and require that

$$0 \leq \psi(\bar{Z}_{k-1}^{(0)}) \leq \psi(\bar{Z}_{k-1}^{(1)}) \leq \ldots \leq \psi(\bar{Z}_{k-1}^{(N-2)}) \leq 1. \tag{3.26}$$

Therefore, if we let $\bar{V}_{k-1}^{(i)} = \psi(\bar{Z}_{k-1}^{(i)})$, the encoding function $\bar{\phi}$ becomes

$$\bar{\phi}(V, \bar{Z}_{k-1}) = \begin{cases} 0 & \text{if} \quad V < \bar{V}_{k-1}^{(0)}, \\ 1 & \text{if} \quad \bar{V}_{k-1}^{(0)} \leq V < V_{k-1}^{(1)}, \\ \cdots & \\ N-1 & \text{if} \quad V \geq \bar{V}_{k-1}^{(N-2)}, \end{cases} \tag{3.27}$$

where "$<$" defines an ordering between the elements of $\mathcal{V}$, and the mapping $\psi : \mathcal{V} \to [0, 1)$ preserves this ordering. Since the encoder is the human user, we also require that human users can determine the pairwise ordering of any two elements in $\mathcal{V}$.

For robotic navigation tasks, we model the human intent as a desired path to be followed by the robot. In Chapters 5 and 6, we will consider two different representations of desired paths, and show that these representations can be encoded as message points and human users can determine the ordering of desired paths under these representations.

Chapter 4

Querying Human Intent using Active Inference Policy

4.1 Introduction

This chapter presents an approach based on the framework of *Bayesian active inference* [8, 9, 10] to querying human intent for interfaces that can only obtain noisy and discrete inputs at high latency from a human user, and that can provide the human user visual feedback. In this approach, we construct a policy that finds the query with the maximum value (e.g., information content) given a Bayesian estimate of the human intent after the responses to previous queries and an estimate of how quickly and accurately the robot can obtain the human response. We propose a measure to evaluate the value of a query, called *information gain rate*, which computes the expected amount of information to be obtained per unit of time about human intent. We refer to the policy that selects the query with the maximum information gain rate from a given pool of queries as the *active inference policy.* This policy generalizes the optimal feedback policy (Chapter 3), which was applicable for querying human intent under certain conditions, i.e., when the synthesized queries could be interpreted without error and without significant delay by human users. Active inference policy can select queries from a given pool of queries designed beforehand to ensure that humans can respond to each query quickly and accurately. Therefore, active inference policy does not have the limitations of the optimal feedback policy.

In Section 4.2, we describe the framework of Bayesian active inference, which is used to infer an unknown object with a target probability of error by adaptively selecting events that provide observations about the unknown object. This framework can be viewed as (sequential) *Bayesian experimental design*, where the goal of experimentation is to infer the unknown value of the random object, which is assumed to be distributed according to a prior probability known by the designer. The experiment to design is chosen to be the one with the maximum value given the observations from previous experiments.

In Section 4.3, we show how this framework can be used for inferring human intent with input commands that are discrete, noisy and that have high latency. Here, experiments are queries that are posed by the robot to extract information about the human intent. We compute the value of a query as the expected amount of information to be extracted about human intent per unit of time by posing the query. We show that, under certain conditions, active inference policy reduces to the optimal feedback policy described in Chapter 3 for transmitting a message point over discrete noisy channels with noiseless feedback.

4.2 Bayesian Active Inference

Bayesian active inference aims to infer an unknown object (e.g., an unknown parameter in a model) by adaptively selecting events (e.g., experiments, tests or queries) that provide observations about the unknown object. For example, consider the problem of inferring a patient's disease by asking the patient a sequence of questions about their symptoms. Here, the unknown object is the patient's disease, and the events are the queries that the doctor asks the patient to understand whether the patient has a particular symptom or not. The problem is to infer the unknown object with a target probability of error using a minimum number of events, or more generally, by incurring a minimum amount of cost, where each event has an associated cost (e.g., the expected time to describe the patient the kind of conditions that represent the particular symptom queried).

In the absence of noise in the observations, this problem is known as object identification, or optimal decision tree problem, which has been shown to be NP-complete [81]. A greedy algorithm known as *generalized binary search* (GBS) provides a $O(\log n)$ approximation ratio, where n is the number of possible objects [82]. The idea is to select the query that eliminates the maximum number of objects on average. This is achieved by finding the most even split between the remaining set of possible objects, so that the true response will eliminate one side of the split.

When the object to be inferred is a hypothesis for labeling data, this problem is known as active learning [83, 10]. In active learning, the events are queries that are used to obtain the true labels for selected unlabeled data points. In pool-based active learning, we are given a pool of unlabeled data points $\mathcal{X}$, and we are interested in learning an hypothesis h that maps data points $\mathcal{X}$ to labels chosen from a finite set $\mathcal{L}$. Often, the labels are binary, i.e., $\mathcal{L} = \{+1, -1\}$, and a query tests whether a particular data point $x \in \mathcal{X}$ has a positive or negative label. Such queries are called membership queries. In general, each query has an associated cost, and the goal is to find a consistent hypothesis by minimizing the expected amount of cost. It has been shown that the generalized binary search achieves a near-optimal expected cost when the labels obtained by the queries are noiseless [82, 84, 85]. Recently, for the case of learning hypotheses using membership queries with noisy observations, modified versions of the GBS algorithm have been shown to be near-optimal in both the setting where true responses might be observed by repeating a particular query [86], and the setting where repeating a particular query produces the same noisy response [87]. A different algorithm for the latter setting has also been shown to be near-optimal [88].

One way to formalize the Bayesian active inference problem is to use the framework of *Bayesian experimental design* [89]. Here, a query is in the form of an experiment that is conducted to obtain information about an unknown object, or to make a decision. Assume that the goal of experimentation is to infer the unknown value of a random object V taking values from a set $\mathcal{V}$. Before the experimentation, we have a prior probability distribution over the random objects, given by $P(V)$. An experiment ε for attaining information on V is a tuple $\varepsilon = (\mathcal{Y}, P(Y|V))$, where $\mathcal{Y}$ is the set of possible observations, and $P(Y|V)$ is a collection of conditional probability densities $p(y|v)$ for each $y \in \mathcal{Y}, v \in \mathcal{V}$. The approach is to select the experiment to be the one with the highest value, i.e.,

$$\arg\max_{\varepsilon \in \mathcal{E}} R(\varepsilon), \tag{4.1}$$

where $R(\varepsilon)$ is a function that measures the value of the experiment. Lindley [8] suggested selecting the experiment for which the expected information gain is the greatest, i.e., selecting

$$\arg\max_{\varepsilon \in \mathcal{E}} I(V; Y|\varepsilon), \tag{4.2}$$

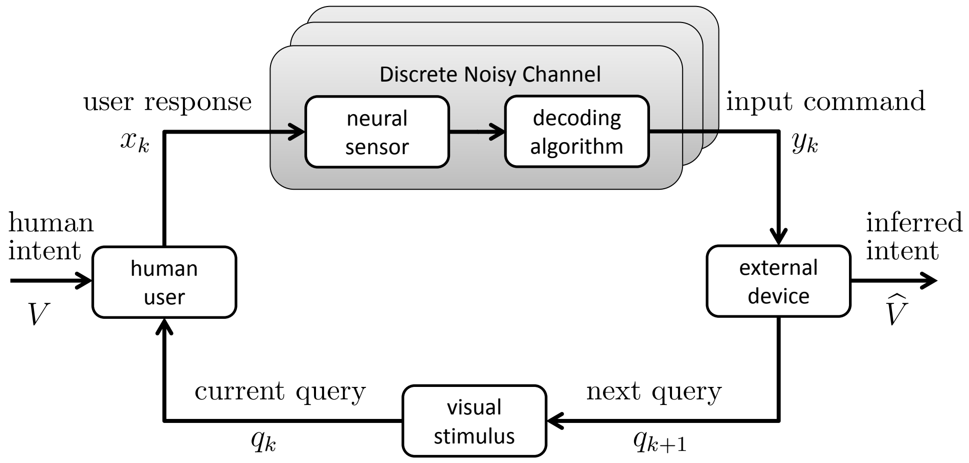


Figure 4.1: The components of an EEG-based brain-machine interface and how they interact with each other in closed loop to obtain input commands from the user.

where $I(V;Y|\varepsilon)$ is the expected information gain of the experiment ε measured by the mutual information between the random observation Y and the random object V. Several other measures have been proposed for Bayesian experiment design [9, 89].

In the decision-theoretic setting, the goal of the experimentation is to make a terminal decision d from a set of possible decisions $\mathcal{D}$ [89]. Let $U(d, v, \varepsilon, y)$ be the utility of choosing the terminal decision $d \in \mathcal{D}$ when the unknown object is $v \in \mathcal{V}$, the experiment designed is $\varepsilon \in \mathcal{E}$, and y is the observation from ε. The experiment to design is then

$$\arg\max_{\varepsilon\in\mathcal{E}} \int_{\mathcal{Y}} \max_{d\in\mathcal{D}} \int_{\mathcal{V}} U(d, v, \varepsilon, y) p(v|y, \varepsilon) p(y|\varepsilon) dv dy, \tag{4.3}$$

which maximizes the expected utility. Closed form solutions can be obtained in simple models (e.g., the normal linear model) for some utility functions [89].

4.3 Querying Human Intent using Bayesian Active Inference

In this section, we describe our method of using Bayesian active inference (Section 4.2) in the design of interfaces for human control of robots with inputs that are discrete, noisy and that have high latency.

4.3.1 The model for querying human intent

We make the following choices that allow us to use the framework of Bayesian active inference for querying human intent. These choices are consistent with the scope defined in Section 1.2.

Modeling an input mechanism as a discrete noisy channel with non-uniform latency structure. The human user communicates their intent V to the robot by providing discrete noisy input commands. We model this input mechanism as a discrete noisy memoryless channel with non-uniform latency structure. We denote such a channel c as a 4-tuple $c = (\mathcal{X}, \mathcal{Y}, P_{Y|X}, L_{X,Y})$, where $\mathcal{X}$ is the channel input alphabet, $\mathcal{Y}$ is the channel output alphabet, $P_{Y|X}$ is the probability transition matrix, and $L_{X,Y}$ is the latency matrix. $P_{Y|X}(y|x)$ is the probability that the channel output $y \in \mathcal{Y}$ is observed when the channel input is $x \in \mathcal{X}$. $L_{X,Y}(x, y)$ is the expected time it takes to observe y when the channel input is $x \in \mathcal{X}$. We refer to the elements of a particular channel c as $\mathcal{X}^{(c)}$, $\mathcal{Y}^{(c)}$, $P^{(c)}_{Y|X}$, and $L^{(c)}_{Y|X}$.

Modeling a query as a deterministic mapping from V to channel inputs. Each query q defines a mapping f from the human user's intent V to the set of possible channel inputs, so that the user knows how to respond to the query. In particular, we define a query q as a tuple $q = (f, c)$, where $f : \mathcal{V} \to \mathcal{X}^{(c)}$, $c \in \mathcal{C}$ is the associated input mechanism, and $\mathcal{X}^{(c)}$ is the channel input alphabet of the input mechanism c.

Modeling the human user as the agent responding to queries without making mistakes. The human user provides information about their intent V by responding to queries. When the user's intent is $V = v$, given a query $q = (f, c)$, the user provides the channel input $x = f(v)$. We call this channel input as the user response, and represent it by a random variable $X^{(q)} \in \mathcal{X}^{(c)}$. We assume that there is a pool of queries, denoted $\mathcal{Q}$, where the human user can respond each query $q \in \mathcal{Q}$ without making mistakes. This means that the human user selects the channel input according to the function f associated with the query q.

Modeling the robot as the agent selecting queries from a pool of queries. The robot uses queries to extract information about the human user's intent V. Each query is selected from the pool of queries $\mathcal{Q}$. After a query $q = (f, c)$ is selected, it is posed to the user by presenting a visual stimulus. Then a channel output $y \in \mathcal{Y}^{(c)}$ is received. This channel output, also called input command, serves as a noisy response to the selected query, and is represented by a random variable $Y^{(q)} \in \mathcal{Y}^{(c)}$.

Illustration. Querying human intent under these models is illustrated in Figure 4.1. The human user communicates their intent $V = v$ to the robot by providing input commands successively through the selected input mechanisms. At time step k, the human user is presented with a visual stimulus, which is a graphical representation of a query $q_k \in \mathcal{Q}$. We denote the mapping and the input mechanism specified by q_k as f_k, and c_k, respectively. As a response to the query q_k, the user provides the channel input $x_k = f_k(v)$. Then, the robot receives the channel output $y_k \in \mathcal{Y}^{(c_k)}$, which is distributed according to $P^{(c_k)}_{Y|X}(y_k|x_k)$, after an expected amount of time given by $L^{(c_k)}_{X,Y}(x_k, y_k)$. We note that we refer the channel output y_k as the input command for the robot. The robot uses this command to increase its information about the user's desired character V. If a stopping criterion is achieved, it generates an estimate $\widehat{V}$, which might then be executed autonomously by the robot.

4.3.2 The approach for adaptively selecting queries

The approach is to adaptively select the next query to be the one with the highest value for inferring V after the responses to previous queries, and an estimate of how quickly and accurately the robot can obtain the user's response. In particular, we assume that there is a finite set $\mathcal{C}$ of input mechanisms modeled as discrete noisy channels with latency, and for each input mechanism we know the characteristics of the underlying channel, i.e., the channel input alphabet $\mathcal{X}$, the channel output alphabet $\mathcal{Y}$, and probability transitions matrix $P_{Y|X}$ (estimating how accurate the channel outputs are), and the latency matrix $L_{X,Y}$ (estimating how fast the channel outputs will be observed). We further assume that there is a pool $\mathcal{Q}$ of possible queries, for which the human user can respond without making mistakes.

Following the approach in Bayesian active inference (Section 4.2), the robot selects each query to be the one with the highest value. At time step k, the robot has observed the noisy responses to previous queries as input commands $y_1, \ldots, y_k$. Denote these input commands as y^k. Therefore, the next query q_{k+1} is selected to be

$$q_{k+1} = \arg\max_{q \in \mathcal{Q}} R(q|y^k), \tag{4.4}$$

where $R(q|y^k)$ is the value of a query q. We will describe our choice of the value function in Section 4.3.3.

The robot maintains a posterior probability over $\mathcal{V}$, denoted $P(V|y^k)$, which represents the robot's belief of V after the noisy responses to queries posed so far. Assume that the robot posed the query $q_{k+1} = (f_{k+1}, c_{k+1})$, and observed y_{k+1}, then this belief is updated as

$$P(V = v|y^k, y_{k+1}) = \eta P^{(c_{k+1})}_{Y|X}(y_{k+1}|f_{k+1}(v))P(V = v|y^k), \tag{4.5}$$

where η is a normalizer. This is obtained by applying the Bayes' rule

$$P(V = v|y^k, y_{k+1}) = \eta P(y_{k+1}|V = v, y^k)P(V = v|y^k), \tag{4.6}$$

and noting that

$$P(y_{k+1}|V = v, y^k) = P^{(c_{k+1})}_{Y|X}(y_{k+1}|f_{k+1}(v)) \tag{4.7}$$

holds because the posterior probability of observing y_{k+1} is independent of V given the channel input $x_{k+1} = f_{k+1}(v)$.

4.3.3 Computing the value of a query: information gain rate

We compute the value of a query by taking into account both the expected information gain from the noisy observation of the user's response to the query, and the expected time it takes to receive the noisy observation. The expected information gain for a query $q = (f, c)$ after k observations is computed as

$$I(V; Y^{(q)} | y^k), \tag{4.8}$$

where $Y^{(q)}$ is the noisy observation to $X^{(q)} = f(V)$, governed by $P^{(c)}_{Y|X}$. The expected time it takes to receive the noisy observation, referred as the expected latency of a query $q = (f, c)$, is computed as

$$E_{X,Y}[L^{(c)}_{X,Y}] = \sum_{x \in \mathcal{X}^{(c)}} \sum_{y \in \mathcal{Y}^{(c)}} P(x, y) L^{(c)}_{X,Y}(x, y), \tag{4.9}$$

where $P(x, y) = P^{(q)}_X(x) P_{Y|X}(y|x)$ is the joint distribution over $X^{(q)}$, and $Y^{(q)}$, and $P^{(q)}_X$ is the probability distribution of user responses. Our value function is the expected information gain per unit time, defined by the ratio of the expected information gain to the expected latency, i.e.,

$$R(q|y^k) = \frac{I(V; Y^{(q)} | y^k)}{E_{X,Y}[L^{(c)}_{X,Y}]}. \tag{4.10}$$

We refer to this function as the *information gain rate*.

The information gain rate can be simplified to increase time efficiency and to observe the link between this approach and the optimal feedback policy described in Chapter 3, as will be discussed in Section 4.3.4. First, note that after k observations, $X^{(q)}$ is a random variable distributed with

$$P^{(q)}_X(x) = \sum_{\forall v \in \mathcal{V}, f(v) = x} P(v|y^k). \tag{4.11}$$

Intuitively, $P_X(x)$ is the total posterior probability of all intents for which the user must select $x = f(v)$. Second, note that the conditional $P(y|v)$ is given by

$$P(y|v) = P(y|x = f(v)) = P^{(c)}_{Y|X}(y|x), \forall v \in \mathcal{V}, \tag{4.12}$$

using the deterministic mapping f from intents to user responses. Using these two observations, we see that

$$\begin{aligned} I(V;Y^{(q)}|y^k) &= H(Y^{(q)}|y^k) - H(Y^{(q)}|V,y^k) \\ &= H(Y^{(q)}|y^k) - H(Y^{(q)}|X^{(q)},y^k) \\ &= I(X^{(q)};Y^{(q)}|y^k). \end{aligned} \tag{4.13}$$

Therefore, we can represent the information gain rate (IGR) for a given query $q = (f, c)$ as

$$R(q|y^k) = IGR(P_X^{(q)}, P_{Y|X}^{(c)}, L_{X,Y}^{(c)}) = \frac{I(X^{(q)};Y^{(q)}|y^k)}{E_{X,Y}[L_{X,Y}^{(c)}]}, \tag{4.14}$$

which depends on the likelihood of responses to the query, given by $P_X^{(q)}$, and the estimate of how accurate and fast the response can be obtained, given by the characteristics $P_{Y|X}^{(c)}, L_{X,Y}^{(c)}$ of the input mechanism c associated with the query.

4.3.4 Conditions under which the active inference policy reduces to the optimal feedback policy

The active inference policy, which adaptively selects queries that maximize information gain rates, reduces to the optimal feedback communication policy described in Chapter 3 for reliable transmission of a message point W over a discrete noisy communication channel under the following assumptions.

- The human intent can be modeled as a random variable V distributed uniformly in the unit interval $[0, 1)$. In other words, the human intent can be represented as a message point.
- There is only one input mechanism c, and c has unit latency, i.e., the latency matrix $L_{X,Y}^{(c)}$ is a constant matrix.
- The query pool is $\mathcal{Q} = \mathcal{F} \times \{c\}$, where $\mathcal{F}$ is the space of all encoding

functions ϕ of the form in (3.16), i.e., functions

$$f(V) = \begin{cases} 0 & \text{if} \quad V \in [0, z^{(0)}), \\ 1 & \text{if} \quad V \in [z^{(0)}, z^{(1)}), \\ \cdots & \\ N-1 & \text{if} \quad V \in [z^{(N-2)}, 1), \end{cases} \tag{4.15}$$

for all $z^{(0)}, z^{(1)}, \ldots, z^{(N-2)} \in [0, 1)$ satisfying $0 \leq z^{(0)} \leq z^{(1)} \leq \ldots \leq z^{(N-2)} \leq 1$, are in $\mathcal{F}$, where N is the size of the channel input alphabet.

Under these assumptions, maximizing IGR is equivalent to maximizing the mutual information between the channel input X, and the channel input Y. By selecting the queries in this way, the channel operates at the capacity. In other words, the likelihood of responses to the selected query is distributed with the capacity-achieving input distribution P_X^*.

Chapter 5

Representing Human Intent as Strings of Symbols for Robotic Navigation

5.1 Introduction

In robotic navigation tasks, we model the human intent as a desired path to be followed by the robot. This chapter presents an approach based on a symbolic language to represent desired paths.

Our choice of a symbolic language to specify desired paths is motivated by the symbolic structure of human language, in particular, the fact that in text form it can be expressed as a sequence of characters and that in speech form it can be expressed as a sequence of phonemes. We will see that a planar path expressed as a string of symbols from a symbolic language, admits useful properties, which allow the use the optimal feedback policy for querying human intent, see Chapter 3, in interfaces for robotic navigation.

Several studies conducted on humans suggest that purposeful human movements are composed of simple and highly stereotyped entities that are usually called motor primitives, motion primitives, or submovements [90, 91, 92, 93]. These primitives have a characteristic shape or speed profile [94, 95] and have been variously explained as the result of optimal control applied to objective functions like minimum jerk, movement time, energy, or acceleration [96, 97, 98, 99]. A motion primitive can be represented in many different ways [100, 101, 102] such as a stochastic linear dynamical system with primitive-specific set of parameters [103, 104], as a hidden Markov model with primitive-specific output and transition probabilities [105], or as a curve segment with primitive-specific velocity or shape profile [95, 106].

In our approach, we define a set of path primitives and use these primitives as symbols to construct an alphabet. Strings of symbols from this alphabet correspond to paths that result from concatenation of path primitives identified by the symbols. Section 5.2 describes our choice of path primitives and the properties of the resulting paths. In order to represent desired paths with as few bits as possible in expectation, we learn a prior model that assigns likelihood to each string from existing data. This prior model allows us to represent paths that are more likely with less number of bits. Section 5.3 describes our algorithm for learning a prior model.

In Chapter 3, we concluded that encoding of human intent as a message point in the unit interval admits an optimal feedback policy for querying human intent. In Section 5.4, we describe an algorithm that encodes strings as message points such that the expected number of bits necessary to specify a message point is as small as possible.

5.2 Representing Desired Paths as Strings of Symbols

Our goal is to represent desired paths for a robot as strings of symbols chosen from a fixed low-cardinality alphabet. Let us assume that the robot is in a state $q(t) \in \mathcal{Q}$ in a configuration space $\mathcal{Q}$ at time t, and it has a deterministic state transition function $\dot{q} = f(q, u)$, where $u(t) \in \mathcal{U}$ is the control input at time t. We define a path primitive as a tuple (a, d), where a is a quantized input restricted to values from a finite set $\mathcal{A} \subset \mathcal{U}$, and $d \in \mathbb{R}^+$ is the duration for which the input a is applied to the robot. In other words, if a primitive (a, d) is applied to the robot at state q at $t = 0$, we require that

$$u(t) = a \quad \text{for} \quad t \in [0, d]. \tag{5.1}$$

We refer to the path generated by integrating f, from a starting state q, using the fixed input $a \in \mathcal{A}$ for d units of time as a *symbol*. We associate a unique symbol to each possible primitive, and construct an *alphabet* $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_M\}$, where each symbol $\sigma_i \in \Sigma$ corresponds to a path primitive $(a, d), a \in \mathcal{A}, d \in \mathbb{R}^+$. We can now represent paths concisely as strings of symbols from Σ. Given a starting state q, a path γ in $\mathbb{R}^2$ or $\mathbb{R}^3$ can be represented as a *string* $\mathbf{v} = (v_1 \cdots v_N)$ with symbols $v_i \in \Sigma$, $i \in \{1, \ldots, n\}$. We denote the set of all strings by Σ^*, and the set of all strings of length N by Σ^N. An example alphabet for constructing planar paths is presented in Section 5.2.1.

5.2.1 Alphabet of Circular Arcs for Constructing Planar Paths

We consider representing piecewise-smooth planar curves as sequences of fixed-length circular arcs with curvatures chosen from a finite set. From differential geometry of curves [107], we know that any smooth planar curve $\gamma\colon [0, L] \to \mathbb{R}^2$ of arbitrary length L can be described by

$$\dot{x} = \cos\theta \qquad \dot{y} = \sin\theta \qquad \dot{\theta} = -u, \tag{5.2}$$

for an initial condition

$$x(0) = x_0 \qquad y(0) = y_0 \qquad \theta(0) = \theta_0, \tag{5.3}$$

where $\theta(t)$ is the angle of the tangent to the curve at $(x(t), y(t))$, and $u(t)$ is the curvature. We assume $x_0 = y_0 = \theta_0 = 0$ without loss of generality. The curve is straight when $u = 0$, turns left when $u < 0$, and turns right when $u > 0$. These ordinary differential equations also define the state transition function f of a unit-speed unicycle [108]. We consider a subset of curves for which u is piecewise-constant on intervals of length d and takes values from a finite set $\mathcal{A} = \{a_1, a_2, \dots, a_M\}$, where elements of $\mathcal{A}$ are ordered so that $a_i < a_j$ for all $i, j \in \{1, \dots, M\}$ satisfying $i < j$. In particular, on each interval $k \in \{1, 2, \dots\}$, we require that

$$u(t) = \sigma_{\mathcal{I}(k)} \qquad (k-1)d \leq t \leq kd \tag{5.4}$$

for some $\mathcal{I}\colon \mathbb{N} \to \mathcal{A}$. We define an alphabet $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_M\}$, where each symbol σ_i represents the circular arc generated by applying the primitive (a_i, d), for $i \in \{0, 1, \dots, M\}$.

Figure 5.1 demonstrates an example alphabet consisting of seven fixed-length circular arcs with central angles evenly distributed in $[-\pi/2, \pi/2]$. Three sample paths composed form this alphabet are shown in Figure 5.2.

Ordering strings of symbols

A key property of a symbolic language using an alphabet of circular arcs is that it admits an intuitive lexicographic ordering. In our alphabet of circular arcs, we have $\sigma_1 < \sigma_2 < \cdots < \sigma_M$ so that $\sigma_i = (a_i, d) < \sigma_j = (a_j, d)$ if and

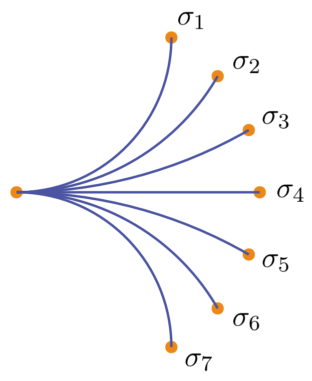


Figure 5.1: An example alphabet consisting of a set of fixed-length circular arcs.

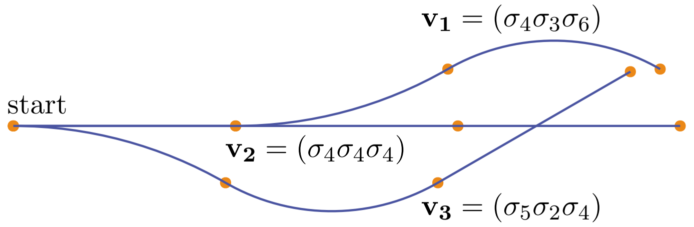

$\mathbf{v_1} = (\sigma_4\sigma_3\sigma_6)$
start
$\mathbf{v_2} = (\sigma_4\sigma_4\sigma_4)$
$\mathbf{v_3} = (\sigma_5\sigma_2\sigma_4)$

Figure 5.2: Three sample paths composed from the example alphabet in Figure 5.1. They exhibit the ordering $\mathbf{v_1} < \mathbf{v_2} < \mathbf{v_3}$.

only if $a_i < a_j$. This corresponds to the notion that σ_i turns more left than σ_j. For two curves $\mathbf{v} = (v_1 \cdots v_N)$ and $\mathbf{v}' = (v_1' \cdots v_N')$, we say that $\mathbf{v} < \mathbf{v}'$ if and only if $v_k < v_k'$, where k is the minimum index for which $v_k \neq v_k'$. This ordering corresponds to the notion that $\mathbf{v}$ "turns left" the first time it differs from $\mathbf{v}'$, and allows humans to "alphabetize" curves just like they would alphabetize strings of text. Figure 5.2 shows three ordered paths starting from the same configuration.

5.3 Learning Prior Model for Strings of Symbols

We think of a string $\mathbf{v} = (v_1 \cdots v_N)$ as a realization of a random vector $\mathbf{V} = (V_1 \cdots V_N)$, where each $V_i \in \Sigma$, $i \in \{1, \dots, N\}$ is a random variable identifying the $i-$th symbol of a desired path. We associate a prior statistical model with our symbolic language by assuming that the user's desired path is governed by a Markov process.

Building statistical models for human motion has been proven to be useful in applications like motion synthesis for computer animation [101, 103], gesture or activity recognition from motion data [109, 110], and motion planning for robots [111, 112, 113]. The most common approach is to learn a statistical model from a dataset of human demonstrated behavior by first representing data using a class of motion primitives, and then by computing conditional probabilities among these primitives using a Markov assumption. In our model of human motion based on a symbolic language, this approach consists of solving the following two problems:

- *the segmentation problem*: represent a human-demonstrated path γ as a sequence of symbols from a given alphabet Σ. In particular, find $\mathbf{v} = (v_1, \dots, v_n) \in \Sigma^*$ that best approximates γ.

- *the modeling problem*: construct a probabilistic language model that assigns conditional probability to a symbol given the past symbols,i.e., $P(V_k | V_1, \dots, V_{k-1})$.

5.3.1 Our approach to the segmentation problem

In the segmentation problem, our goal is to represent a human-demonstrated desired path γ as a string $\mathbf{v} \in \Sigma$ in a given alphabet Σ. In order to solve the segmentation problem, we minimize the approximation error

$$E\left(\mathbf{v} = (v_1, \ldots, v_N), \gamma\right) = \sum_{v_i, i \in \{1,\ldots,N\}} e\left(v_i, \gamma^{(v_i)}\right) \tag{5.5}$$

where $e(\gamma^{(1)}, \gamma^{(2)})$ is a dissimilarity measure between two curve segments γ_1, γ_2, and $\gamma^{(v_i)}$ is the portion of the curve γ that the symbol $v_i \in \Sigma$ approximates under a given correspondence between the symbols of $\mathbf{v}$ and the segments of the curve γ. Unfortunately, finding $\mathbf{v}^* = \arg\min_{\mathbf{v} \in \Sigma^*} E(\mathbf{v}, \gamma)$ exactly is not computationally efficient in most practical cases, and requires a partial enumeration of all possible strings in Σ^*. However, we can get an approximate answer by considering a discretization of the curve γ parameterized by the time-steps in a finite set $\mathcal{T}$, and a cluster of quantized configurations, denoted $\widehat{\mathcal{Q}}(t) \subset \mathcal{Q}$, for each time set $t \in \mathcal{T}$. We assume that the correspondence between the string $\mathbf{v}$ and the curve γ is defined so that each curve segment corresponding to a symbol starts and ends at a time step $t \in \mathcal{T}$. In addition to minimizing the approximation error in (5.5), we minimize the correspondence error

$$D\left(\mathbf{v} = (v_1, \ldots, v_N), \gamma\right) = \sum_{v_i, i \in \{1,\ldots,N\}} d\left(q_{\text{end}}(v_i), \widehat{q}_{\text{end}}(\gamma^{(v_i)})\right), \tag{5.6}$$

where $d(q_1, q_2)$ is a distance measure between two configurations, $q_{\text{end}}(v_i)$ is the endpoint configuration of the symbol $v_i \in \mathbf{v}$, and $\widehat{q}_{\text{end}}(\gamma^{(v_i)})$ is the quantized endpoint-configuration of the curve segment corresponding to v_i. Given a curve γ, we reduced our problem to minimizing the cost

$$J(\mathbf{v}, \gamma) = E(\mathbf{v}, \gamma) + \tau D(\mathbf{v}, \gamma), \tag{5.7}$$

where τ is a parameter that determines the trade-off between the approximation error E and the correspondence error D.

This problem can be solved by dynamic programming. Let $C(t, \widehat{q}), t \in \mathcal{T}, \widehat{q} \in \widehat{\mathcal{Q}}(t)$ be the minimum cost of approximating the portion of the curve γ till time step t, using the quantized configuration $\widehat{q}$ at its endpoint, with a string $\mathbf{v} \in \Sigma^*$. Let $V(t, \widehat{q})$ be the string attaining the minimum cost. Then, we have the recurrence

$$C(t, \widehat{q}) = \min_{t' \in \mathcal{T}, t' < t, \widehat{q}' \in \mathcal{Q}(t')} \left\{ \min_{\sigma \in \Sigma, \text{ after } V(t', \widehat{q}')} \left\{ e\left(\sigma, \gamma_{t' \to t}\right) + \tau d\left(q_{\text{end}}\left(\sigma\right)\right), \widehat{q}\right) \right\} + C(t', \widehat{q}') \right\}, \tag{5.8}$$

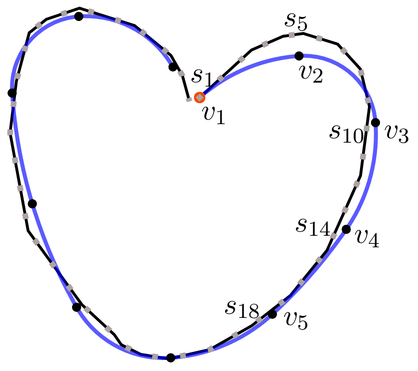
where $\gamma_{t' \to t}$ denotes the portion of the curve γ from t' to t. Assuming that the size of each cluster $\widehat{\mathcal{Q}}(t)$ is limited by $|\widehat{Q}|$, the complexity of this approach is $O(|\Sigma||\mathcal{T}||\widehat{\mathcal{Q}}|)$.

5.3.2 Our approach to the modeling problem

In the modeling problem, our goal is to derive conditional probabilities $P(V_k|V_1, \ldots, V_k)$ from a set of given strings $\mathbf{v_1}, \mathbf{v_2}, \ldots$, that represent human-demonstrated desired paths. Given $\mathbf{v} = (v_1 v_2 \cdots v_N)$, denote its prefix $(v_1 v_2 \cdots v_b)$ as v^b, and its substring $(v_a v_{a+1} \cdots v_b)$ as v_a^b. An n-order Markov model takes into account only the n previous symbols. In other words, $P(v_k|v^{k-1})$ is given by $P(v_k|v_{k-n}^{k-1})$, which can be computed by counting the frequency of v_{k-n}^{k} relative to the frequency of v_{k-n}^{k-1}. In order to handle the zero frequency problem, in practice, a small probability is assigned to each unobserved sequence. Our approach is to use a variable-order Markov model in which the size of the context used for prediction is adjusted based on the available statistics. This is achieved by an algorithm known as prediction by partial matching (PPM) [114], which has shown to be very effective in compression of text sources, and in various other domains for predicting likely discrete sequences [115].

5.3.3 Example: Learning a prior model for hand drawings

As an example, we demonstrate a statistical model for human hand drawings. Such a model can be used in a robotic drawing interface for human specification of paths for a robotic pen moving on a planar surface. From a database of existing artwork, we obtained a set of strokes where each stroke is a planar curve $\gamma : [0, 1] \to \mathbb{R}^2$ formed by applying a drawing tool such as a pen to a



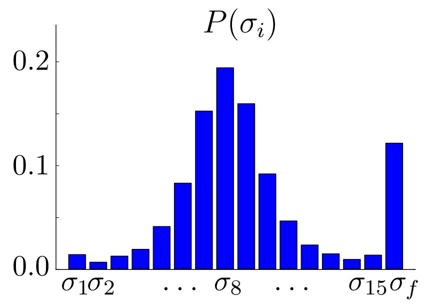


Figure 5.3: *Left:* Fitting an arbitrary planar curve γ (shown in black) to a string $\mathbf{v}$ (shown in blue) in Σ^*. The alphabet Σ consisted of 15 fixed-length circular arcs $\sigma_1, \cdots, \sigma_{15}$, and a special symbol σ_f to indicate the end of the curve. Points $s_1, \dots, s_M$ describe the curve γ, and arcs $v_1, \dots, v_n$ describe the string $\mathbf{v}$. *Right:* The unigram model obtained from a dataset of hand drawings. See [11].

drawing surface. Using our alphabet of circular arcs for constructing planar paths, we solved the segmentation problem first to represent each stroke as a string $\mathbf{v} \in \Sigma^*$. Then, using these strings as input, we built a zero-order Markov model, which is also called a unigram model. The results are shown in Figure 5.3. The details of this work can be found in [11].

5.4 Encoding Strings of Symbols as Message Points

Our goal is to construct an invertible mapping $\psi : \Sigma^* \to [0, 1)$ from random strings $\mathbf{V} \in \Sigma^*$ distributed according to a given statistical model $P(\mathbf{V})$ to message points $W \in [0, 1)$ distributed uniformly in the unit interval. Such a mapping is called a *source code*. A source code is optimal if the average number of bits used to represent the message point $W = \psi(\mathbf{V})$ matches the entropy of the statistical model $P(\mathbf{V})$.

Our approach is to use a method of lossless data compression called arithmetic coding [116] as our source code. Arithmetic coding maps a random string $\mathbf{V} = (V_1, V_2, \dots, V_N)$ with random variables $V_i \in \Sigma$ into a subinterval $[l_{\mathbf{V}}, r_{\mathbf{V}}) \in [0, 1)$. Any point $W \in [l_{\mathbf{V}}, r_{\mathbf{V}})$ can be used to represent V, here we simply choose $\psi(\mathbf{V})$ as the midpoint $(l_{\mathbf{V}} + r_{\mathbf{V}})/2$. A set of paths and their mappings under a uniform prior over symbols are illustrated in Fig. 5.4.

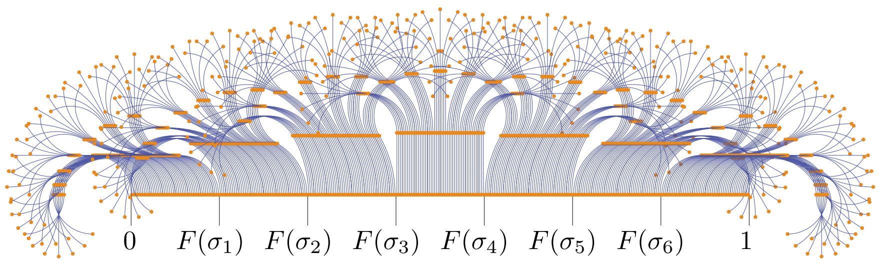


Figure 5.4: All paths composed of three symbols mapped to a real point in $[0, 1)$ using arithmetic coding. $F(\sigma_i)$ is the probability that the first symbol is σ_i or comes before σ_i, which is $i/7$ in this example.

Algorithm 1 Source Decoding: $\psi^{-1} : [0, 1) \to \Sigma^N$

```
Input: $W \in [0, 1)$
Output: $\mathbf{V} = (V_1, V_2, \ldots, V_M), V_i \in \Sigma, \mathbf{V} \in \Sigma^N$
 1: $u_{\min} \leftarrow 0$
 2: $u_{\max} \leftarrow 1$
 3: for $j = 1$ to $N$ do
 4:     $\delta \leftarrow u_{\max} - u_{\min}$
 5:     $i = 0$
 6:     repeat
 7:         $i \leftarrow i + 1$
 8:     until $P(V_j \leq \sigma_i | v_1 \ldots v_{j-1}) \geq (W - u_{\min})/\delta$
 9:     $u_{\max} \leftarrow u_{\min} + \delta P(V_j \leq \sigma_i | v_1 \ldots v_{j-1})$
10:     $u_{\min} \leftarrow u_{\min} + \delta P(V_j < \sigma_i | v_1 \ldots v_{j-1})$
11:     $V_j = \sigma_i$
12: end for
```

Given a desired length M, the inverse mapping $\psi^{-1} : [0, 1) \to \Sigma^N$, from a real number $W \in [0, 1)$ to a random string $\mathbf{V} = (V_1, V_2, \ldots, V_N)$, is described in Algorithm 1.

This method is both optimal and has the useful property that it preserves lexicographic ordering, so that $\mathbf{v} < \mathbf{v}'$ if and only if $\psi(\mathbf{v}) < \psi(\mathbf{v}')$.

Chapter 6

Representing Human Intent as Local Geodesics for Robotic Navigation

6.1 Introduction

In robotic navigation tasks, we model the human intent as a desired path to be followed by the robot. It is necessary to restrict the space of all possible paths the user can specify because arbitrary paths cannot be described with a finite number of inputs. In Chapter 5, we made a heuristic choice, and used an ordered symbolic language to represent paths of piecewise-constant curvature. In this chapter, we present an approach based on learning a principle of optimality to make the choice of representation more systematic.

Although the space of all possible paths through a finite-dimensional space is infinite-dimensional, paths taken by "real" robotic systems often cluster on a finite-dimensional manifold that is embedded in this infinite-dimensional space and that is governed by a principle of optimality. We take advantage of this property to generate a compact representation of all paths likely to be seen in the context of a particular application. In particular, we represent desired paths as local geodesics with respect to a cost function that takes into account environment-driven features such as proximity to obstacles (Section 6.2). We show that an important subset of all local geodesics can be encoded as a message point in the unit circle, so that each angle in $[0, 2\pi)$ corresponds to a unique local geodesic (Section 6.3). Given a set of human-demonstrated paths, we can apply an algorithm based on structure learning to recover a cost function so that local geodesics resemble human-demonstrated paths (Section 6.4).

6.2 Representing Desired Paths as Local Geodesics

We represent desired paths as geodesics that are locally-optimal solutions to

$$\begin{aligned} \text{minimize} \quad & \int_{t=0}^{t=T} g(\gamma(t))dt \\ \text{s.t.} \quad & \gamma(t) \in Q_{\text{free}}, \forall t \in [0,T] \\ & \gamma(0) = q_0 \text{ and } \gamma(T) = q_1, \end{aligned} \tag{6.1}$$

where $\gamma : [0,T] \to Q_{\text{free}}$ is a continuous function, Q_{free} is the free configuration space, $g : Q_{\text{free}} \to \mathbb{R}^+$ is a given cost function, q_0 and q_1 are given start and end configurations, respectively, and T is the free final time. We call a configuration free if the robot at this configuration is not in contact with an obstacle, and call it semi-free if the robot touches an obstacle. We define Q_{free} as the set of free and semi-free configurations.

To make things concrete, we consider a point robot moving through a bounded planar workspace with polygonal obstacles—also called a polygonal domain— which has $Q_{\text{free}} \in \mathbb{R}^2$. A set of sample geodesics between two configurations are shown in Figure 6.1.

Obstacles induce a topological structure to paths in Q_{free}. Two paths γ and γ' are called path-homotopic if they share the same endpoints in Q_{free} and if γ can be continuously deformed to γ'. The path-homotopy defines an equivalence relation and this relation divides paths into path-homotopy classes. We assume that a cost function $g : Q_{\text{free}} \to \mathbb{R}^+$ is given so that there is a unique local geodesic for each path-homotopy class. This allows us to represent a local geodesic only by a path-homotopy class which may be defined using a (reference) path $\pi : [0,1] \to Q_{\text{free}}$ with $\pi(0) = q_0$ and $\pi(1) = q_1$, and denoted $[\pi]$.

We refer a local geodesic as a locally-shorted path if the cost function g is a constant, i.e., $g(q) = c, \forall q \in Q_{\text{free}}$. We say that a geodesic from q_0 to q_1 is globally-shortest if it has the shortest length among all locally-shortest paths from q_0 to q_1. For each path-homotopy class $[\pi]$, there is a unique locally-shortest path that is path-homotopic to π.

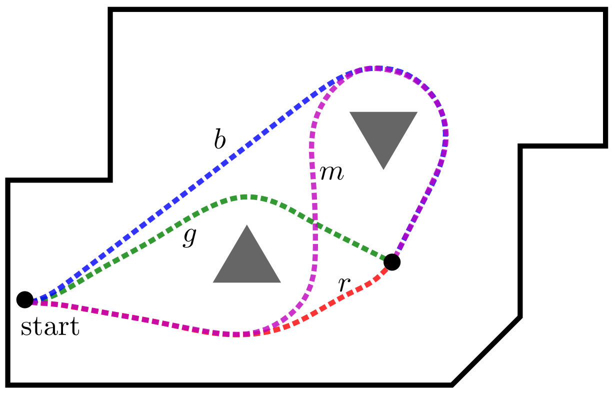


Figure 6.1: Four sample local geodesics between a start and an end position in a polygonal domain.

6.3 Encoding Local Geodesics as Message Points

In this section, we show that an important subset of all local geodesics originating from a fixed starting point can be encoded as a message point in the unit circle, so that each angle in $[0, 2\pi)$ corresponds to a unique local geodesic.

6.3.1 The homeomorphism between locally-shortest paths and the unit disk

In this section, we show by direct construction that the set of all paths $\gamma : [0, T] \to Q_{\text{free}}$ having length that is bounded and locally minimal is homeomorphic to the unit disk, denoted $D^1 = \{x | x \in \mathbb{R}^2, \|x\| \leq 1\}$.

Visibility and the structure of shortest paths

First, we review the structure of shortest paths in a planar workspace with polygonal obstacles and describe efficient ways to compute this structure [108, 117, 118]. Let $\mathcal{P}$ represent a polygonal domain with a free configuration space Q_{free}. We say that two points p, q in Q_{free} are visible if there exist a line segment between p and q in Q_{free}. A vertex v of $\mathcal{P}$ is reflex if the interior angle between its two incident edges is greater than π. Let $\mathcal{S}_{\mathcal{R}}$ be the set of reflex vertices in P. The shortest path from q_0 to q_1 is always a polygonal line in Q_{free} with vertices chosen from the set $\mathcal{S} = \mathcal{S}_{\mathcal{R}} \cup \{q_0, q_1\}$. In particular, its first edge is a line segment from q_0 to a reflex vertex and its last edge is a line segment from a reflex vertex to q_1. All intermediate edges are line segments between the visible reflex vertices, and they either lie in the boundary of $\mathcal{P}$ (a boundary edge) or they are tangents to $\mathcal{P}$ at both endpoints (a bitangent edge). The graph containing the boundary and bitangent edges of $\mathcal{P}$ between the visible reflex vertices is called a shortest-path roadmap $\mathcal{G}$, or a reduced visibility graph. This graph representation allows us to obtain globally-shortest paths from q_0 to q_1 efficiently. We extend $\mathcal{G}$ to contain edges from $\{q_0, q_1\}$ to the reflex vertices that are visible, and then we do a graph search from q_0 to q_1 in the resulting graph. If multiple globally-shortest path queries from a fixed q_0 are going to be performed, we can construct a shortest-path map using the continuous Dijkstra method. This map is a planar decomposition of Q_{free} into cells such that all globally-shortest paths ending in the same cell are identical except at their last polygonal segment. Once constructed, this map representation allows us to obtain the globally-shortest paths from q_0 without searching a graph. Another version of the shortest path problem is to find the locally-shortest path in a given path-homotopy class. The path-homotopy class can be expressed as a sequence of triangles in a triangulation of $\mathcal{P}$. In this case, the locally-shortest path can be computed efficiently using the funnel algorithm. See [108, 117, 118] for details.

For a polygonal domain $\mathcal{P}$, the region of $\mathcal{P}$ visible from a source point x is called the *visibility polygon* of x, and denoted by V_x, see Figure 6.2. The visibility polygon can be computed in time $O(n \log n)$ if $\mathcal{P}$ has n edges. We refer a line segment in V_x that crosses the interior of $\mathcal{P}$ as an *extension edge*. For each extension edge, we refer its vertex that is closest to x as a *way point* and its other vertex as an *extension point*. The way points are vertices through which shortest paths originated from x can cross. We refer to a polygonal chain between two extension edges in the boundary of V_x as a *boundary edge*. This allows us to represent the boundary of V_x as a sequence of extension and boundary edges.

Our representation for locally-shortest paths is most closely related to gap navigation trees [119]. It is constructed from the sequence of critical events occurring in the robot's visibility region as the robot moves in the environment. This tree structure can be used to solve several visibility-based tasks including locally-optimal navigation.

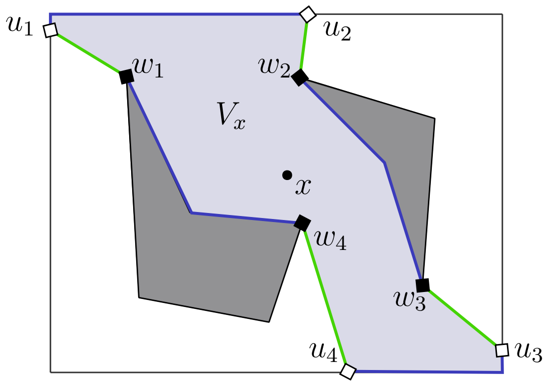


Figure 6.2: The visibility polygon of a point x, denoted by V_x is shown in light gray. The way points from x are marked with black diamonds, and the extension points are marked with empty diamonds. The extension edges are shown in green, and the boundary edges are shown in blue.

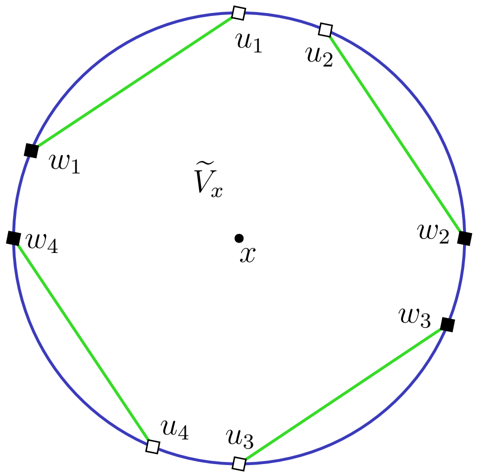


Figure 6.3: Embedding the visibility polygon V_x of Figure 6.2 to a region (cell) of the unit disk, denoted by $\widetilde{V}_x$. The boundary edges of V_x become circular arcs along the disk boundary and the extension edges of V_x become the chords of the unit disk.

Locally-shortest paths as points in the unit disk

We define $\Pi(x)$ to be the set of all locally-shortest paths (with bounded length) originating from $x \in Q_{\text{free}}$, and $\Pi_b(x) \subset \Pi(x)$ to be the set of paths in $\Pi(x)$ that terminate at a semi-free configuration (a boundary point). We denote the (closed) unit disk by D^1. We construct a homeomorphism $\psi : \Pi(x) \to D^1$ such that the restriction of ψ to $\Pi_b(x)$ is the boundary of the unit disk, i.e. the unit circle S^1.

First, we map the locally-shortest paths terminating in V_x to a closed planar region (cell) of D^1, denoted by $\widetilde{V}_x$. The cell $\widetilde{V}_x$ is bounded by a set of chords in D^1 and a set of circular arcs along the disk boundary. We have a chord for each extension edge in V_x, and we have a circular arc for each boundary edge in V_x. The cyclic ordering of the chords and circular arcs in $\widetilde{V}_x$ matches to the cyclic ordering of the edges of V_x. An embedding of V_x of Figure 6.2 to $\widetilde{V}_x$ is shown in Figure 6.3. The shortest paths ending at the boundary edges of V_x map to the points along the disk boundary, and the shortest paths ending at the extension edges of V_x map to the points on the chords.

Then, we iterate over the way points of V_x. Let $V_{x \to w_i}$ be the set of points that become newly visible by going from x to a way point w_i. All shortest paths terminating at a configuration $q \in V_{x \to w_i}$ contain the straight segment from x to w_i in their prefix. $V_{x \to w_i}$ can be computed by cutting the visibility polygon of w_i, V_{w_i}, by the extension edge from w_i in V_x. The embedding of $V_{x \to w_i}$ is the cell adjacent to the chord in $\widetilde{V}_x$ that corresponds to the extension edge from w_i. We denote this cell by $\widetilde{V}_{x \to w_i}$. Its boundary, like $\widetilde{V}_x$, consists of a circular arc for each boundary edge, and a chord for each extension edge. As the algorithm moves to a new way point, a new set of points become visible and we map them to the cells that are adjacent to the chords corresponding to extension edges of the previous visibility region. This process is illustrated in Figure 6.4.

Finally, we map the interior of $V_{p \to q}$ to the interior of $\widetilde{V}_{p \to q}$ by doing a triangulation of $V_{p \to q}$ and matching triangles in $V_{p \to q}$ to the cells of $\widetilde{V}_{p \to q}$ that are either bounded by three chords or bounded by two chords and a circular arc. Any point in the interior of $\widetilde{V}_{p \to q}$ represent a path $\pi \in \Pi(x)$ that terminate in the interior of $V_{p \to q}$. Likewise, any point in the boundary of $\widetilde{V}_{p \to q}$ represents a path $\pi \in \Pi_b(x)$ that terminates in the boundary of $V_{p \to q}$. Figure 6.5 shows a set of paths in $\Pi_b(x)$ and their mapping to points in S^1.

6.3.2 Using the homeomorphism to encode local geodesics as message points

There is a one-to-one correspondence between the elements of the following three sets for a given starting point $q_0 \in Q_{\text{free}}$:

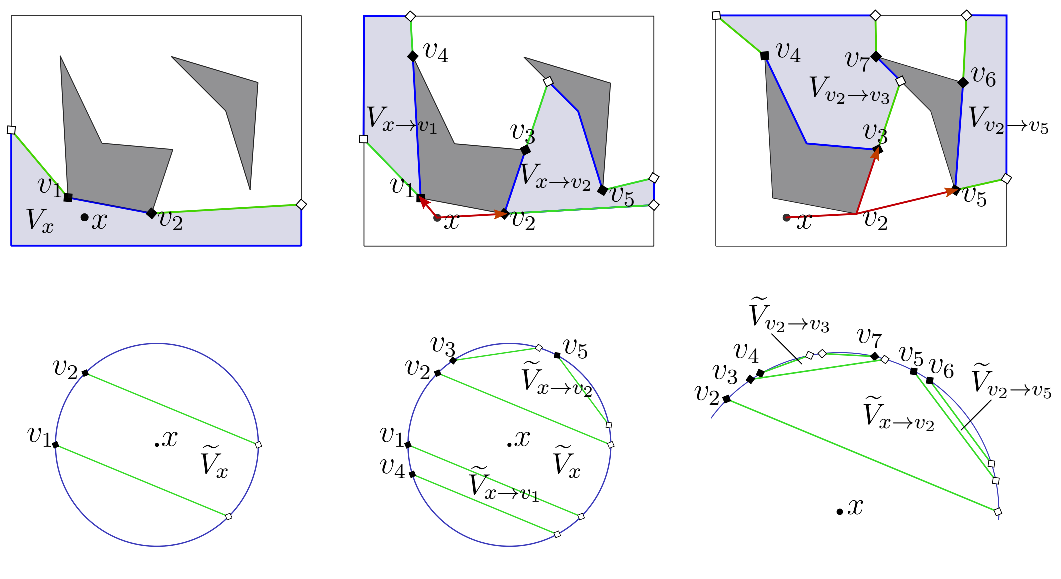


Figure 6.4: Three iterations of constructing a unit disk representation of locally-shortest paths. The top row shows the set of points that become newly visible as the algorithm iterates over the way points. The bottom row shows the embeddings of these visibility regions in the unit disk. V_x is the visibility polygon of x, and $V_{p\to q}$ is the set of points that become newly visible by moving from p to q. Their embeddings are $\widetilde{V}_x$ and $\widetilde{V}_{p\to q}$, respectively.

- The set of all locally-shortest paths from $q_0 \in Q_{\text{free}}$, denoted $\Pi(q_0)$.
- The set of all path-homotopy classes in Q_{free} with a starting point q_0. This set is known as the universal covering space, denoted $\widehat{Q}$, and defined as $\widehat{Q}(q_0) = \{[\gamma] | \gamma : [0,1] \to Q_{\text{free}} \text{ and } \gamma(0) = q_0\}$. We can think of $\widehat{Q}$ as the space formed by concatenating the end points of all path-homotopy classes from q_0.
- The set of all local geodesics from q_0, under a cost function g that is defined so that each path-homotopy class contains a unique local geodesic. We denote this set as $\Gamma^g(q_0)$.

This correspondence allows us to use the homeomorphism $\psi : \Pi(q_0) \to D^1$ derived in Section 6.3.1, to encode path-homotopy classes in $\widehat{Q}(q_0)$ or local geodesics in $\Gamma^g(q_0)$ as points in the unit disk. Moreover, the restriction of ψ to locally-shortest paths in $\Pi_b(q_0)$ allows us to encode path-homotopy classes in $\widehat{Q}(q_0)$ that end at the boundary (denote them $\widehat{Q}_b(q_0)$) or local geodesics in

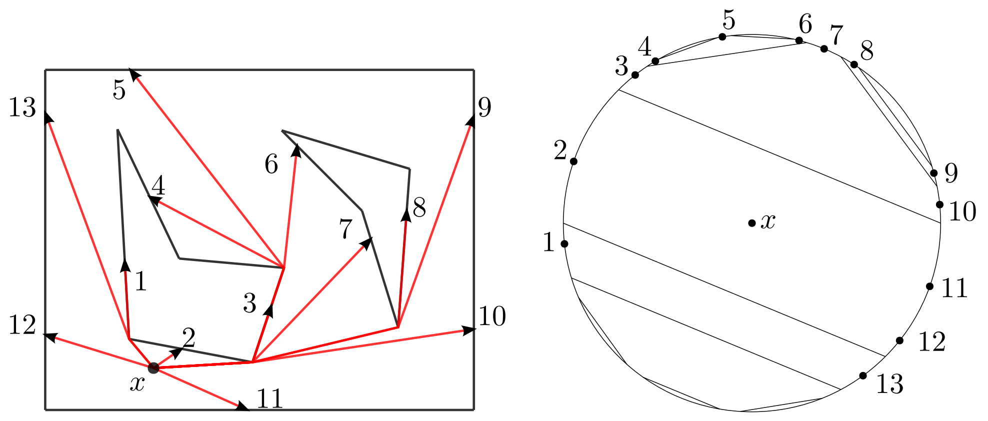


Figure 6.5: A set of locally-shortest paths starting at x and terminating at the boundary is shown on the left, and their corresponding location in the boundary S^1 of the unit disk is shown on the right.

$\Gamma^g(q_0)$ that end at the boundary (denote them $\Gamma^g_b(q_0)$) as points in the unit circle S^1. We denote the homeomorphism between the geodesics in $\Gamma^g_b(q_0)$ and S^1 as $\psi_b : \Gamma^g_b(q_0) \to S^1$.

For robotic navigation tasks, we can model the user's desired path as a local geodesic γ_b in $\Gamma^g_b(q_0)$, where $q_0 \in Q_{\text{free}}$ is the robot's current configuration. We note that any geodesic in $\Gamma^g(q_0)$ is a prefix of some other geodesic in $\Gamma^g_b(q_0)$. In particular, if $\gamma : [0,T] \to Q_{\text{free}}$ is in $\Gamma^g(q_0)$, then there is a unique $\gamma_b : [0,T'] \to Q_{\text{free}}$, where $T' \geq T$, $\gamma_b(T')$ is a boundary point, and the restriction of γ_b to the interval $[0,T]$ is identical to γ. In order to navigate the robot under this model, the user specifies both γ_b and time T. Specification of γ_b can be done efficiently using the optimal feedback policy derived in Chapter 3. Specification of T can be deferred to navigation time. For example, the user can issue a "stop" command to indicate the point at which they want to terminate the navigation.

6.3.3 Ordering local geodesics

A key property of the space of local geodesics $\Gamma^g_b(q_0)$ is that it admits an intuitive cyclic ordering. In other words, $\Gamma^g_b(q_0)$ is a cyclically ordered set, with the ordering defined by the homeomorphism $\psi_b : \Gamma^g_b(q_0) \to S^1$, which maps the local geodesics to the unit circle. Given two geodesics γ_1 and γ_2 in $\Gamma^g_b(q_0)$, we say that γ_1 is ordered to the "left" of γ_2, denoted $\gamma_1 < \gamma_2$, if and only if the clockwise angle from $\psi_b(\gamma_1)$ to $\psi_b(\gamma_2)$ is smaller than the

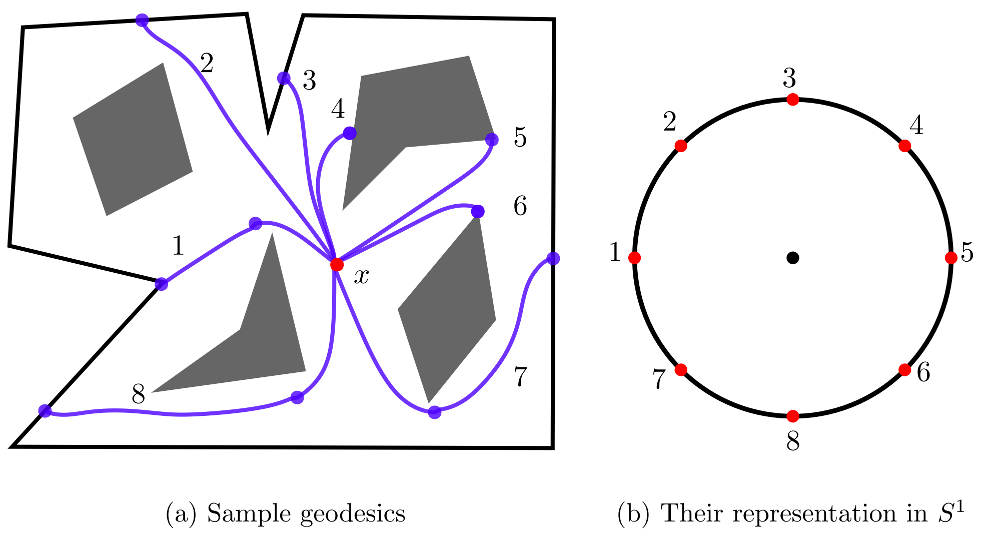


Figure 6.6: A set of geodesics starting at x and ending at the boundary are shown in (a), their corresponding points in the unit circle S^1 are shown in (b). The intermediate points along the geodesics in (a) indicate the points at which the geodesics cross an extension edge.

counterclockwise angle from $\psi_b(\gamma_1)$ to $\psi_b(\gamma_2)$. Figure 6.6 demonstrates a set of geodesics in a $\Gamma_b^g(q_0)$ and their cyclic ordering.

6.4 Learning Prior Model for Local Geodesics

We modeled a desired path as a local geodesic in $\Gamma_b^g(q_0)$, under a cost function $g : Q_{\text{free}} \to \mathbb{R}^+$. In this section, we describe a method to recover a cost function g from existing data so that the geodesics in $\Gamma_b^g(q_0)$ will resemble paths that humans prefer in navigating a mobile robot (e.g., a wheelchair) amidst obstacles in a target path-homotopy class. We note that it is also possible to learn a prior probability distribution over path-homotopy classes corresponding to the local geodesics in $\Gamma_b^g(q_0)$ from existing data, but we do not consider this problem here.

Our approach is to use maximum-margin structured learning (MMSL) [120, 121]. We assume that we are given a training set $D = \{\gamma^{(i)}\}_{i=1}^N$ where $\gamma^{(i)} : [0, T] \to Q_{\text{free}}$ is a human-demonstrated path. We further assume and the cost function $g : Q_{\text{free}} \to \mathbb{R}^+$ takes the form of $g(q) = w^T c(q)$ where w is a non-negative weight vector in a convex parameter space W, and $c = [c_1, \ldots, c_n]$ is a given vector-valued feature function, with each

$c_i : Q_{\text{free}} \to \mathbb{R}^+$. The total cost of a path $\gamma : [0, T] \to \mathbb{R}^2$ in Γ can be expressed as

$$J(\gamma) = \int w^T c(\gamma(t))dt = w^T f(\gamma), \tag{6.2}$$

where $f : \Gamma \to \mathbb{R}^+$ is the feature sum along the path γ. In MMSL, we introduce a margin that scales with the loss of choosing an alternative $\gamma \in \Gamma$ in place of the demonstrated example $\gamma^{(i)}$, and denote it by the function $L_i : \Gamma \to \mathbb{R}^+$, where $L_i(\gamma) > 0$ for all $\gamma \neq \gamma^{(i)}$, and $L_i(\gamma^{(i)}) = 0$. The underlying problem is to learn weights $w \in W$ so that

$$w^T f(\gamma^{(i)}) \leq w^T f(\gamma) - L_i(\gamma), \quad \forall \gamma \neq \gamma^{(i)}, \, \forall i,$$

which requires the cost of an alternative γ to be larger than the cost of the example $\gamma^{(i)}$. Maximizing margin subject to these constraints is equivalent to the convex program

$$\begin{aligned} &\min_{w \in W, \zeta_i \in \mathbb{R}} \frac{1}{N} \sum_{i=1}^{N} \zeta_i + \frac{\lambda}{2} \|w\|^2 \\ &\quad \text{s.t. } w^T f(\gamma^i) \leq \min_{\gamma \in \Gamma} \left\{ w^T f(\gamma) - L_i(\gamma) \right\} + \zeta_i, \quad \forall i, \end{aligned}$$

where $\{ \zeta_i \}_{i=1}^{N}$ are the slack variables, and $\lambda \geq 0$ is a constant that trades off between a penalty on constraint violations and margin maximization. This program can be solved using a subgradient descent algorithm [120, 121].

Part III

APPLICATIONS

Chapter 7

Enabling Humans to Fly Simulated Aircraft with EEG

7.1 Introduction

This chapter presents an interface for navigating a simulated aircraft that moves at a fixed speed in a planar workspace, with noisy binary inputs that are obtained asynchronously at low bit-rates from a human user through an electroencephalograph. In this interface, we use the alphabet of circular arcs described in Chapter 5 to represent desired planar paths for navigation as strings of an ordered symbolic language. The underlying problem is then to design a communication policy by which the user can, with vanishing error probability, specify a string in this language using a sequence of inputs. The approach is to use the optimal feedback policy described in Chapter 3. We show that this policy, which relies on a human user's ability to compare smooth curves, can be easily implemented by a human user. We demonstrate our interface by performing experiments in which twenty subjects fly a simulated aircraft at a fixed speed and altitude with input only from EEG. Experimental results show that the majority of subjects have been able to specify desired paths reliably despite a wide range of errors made in decoding EEG signals.

As described in Chapter 2, existing brain-machine interfaces typically function in one of two different ways: process control and goal selection [14, 41]. In process control, measurements of brain activity are used to specify an immediate action to be taken, such as moving a cursor to the left or to the right. In goal selection, measurements of brain activity are used to specify the desired output after a sequence of actions, such as the location at which the cursor should end up.

Many existing BMIs use process control for movement tasks, including

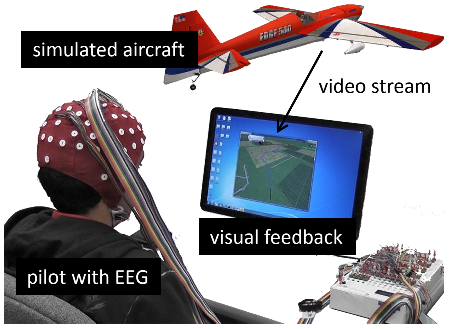
The material in this chapter has been published in [4].

both invasive BMIs for control of a robotic arm by primates [23], and non-invasive BMIs for control of a cursor [122], a wheelchair [42, 43] or a mobile robot [44, 45, 46]. A problem with this strategy, particularly for non-invasive BMIs, is that it tends to produce erratic motion due to the direct mapping from noisy measurements of brain activity to control inputs. Methods of shared control have been proposed as a way to reduce this problem [47, 42, 43, 45]. With shared control, movement is determined by "averaging" inputs produced by the BMI with inputs that might have been expected given a prior model. For instance, Vanacker et al. [42] enabled a human user to drive a powered wheelchair with three inputs—"left","right", and "forward"—obtained from EEG. These inputs were filtered based on the context (e.g., proximity to obstacles) before being mapped to motor commands for the wheelchair. Although shared control improved performance, erratic motion still occurred when user inputs were erroneous or in conflict with the prior model used in the filter.

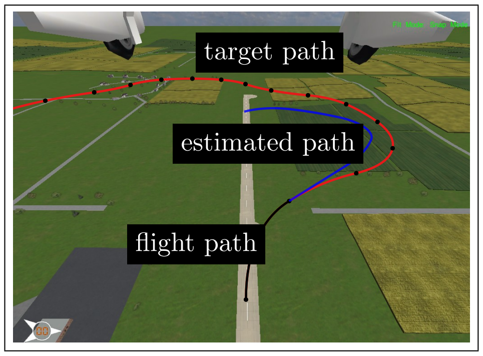
More BMIs are now starting to use goal selection for movement tasks, including an invasive BMI that allowed primates to choose from a set of reaching targets [123], and non-invasive BMIs that allowed human users to choose from a set of destinations for a wheelchair [51], or objects to be picked up by a humanoid robot [52]. A problem with this strategy is that the set users can choose from is restricted to the goals determined by the designer, and the user has no control over how the external device achieves a selected goal. For instance, Rebsamen et al. [51] enabled a human user to drive a powered wheelchair to a destination selected by the user from a list of locations. These destinations were restricted to a given list (bath, bed, office, etc.) and users did not have any control over paths followed by the wheelchair to reach the selected destination.

In an effort to address the problems with process control and goal selection, the BMI proposed by Iturrate et al. [53] uses a hybrid strategy for robotic wheelchair navigation. In this strategy, measurements of brain activity are used to specify a subgoal that indicates the next location to be moved by the robot. The user selects from a finite set of locations that are visible to the robot, and the robot then moves to the selected location autonomously. By repeating this process, in effect, the user specifies a desired path step by step.

Our BMI uses a hybrid strategy that is similar to the one proposed by



(a) Illustration of our interface



(b) A snapshot of the display in the tracking task

Figure 7.1: Our interface for flying a simulated aircraft at a fixed altitude and speed with input only from electroencephalograph (EEG), see (a). The pilot flies the aircraft by imagining either left- or right- hand movement, the choice between which is based on visual feedback provided by a graphical display. In the tracking task, the display shows the target path (red curve), the estimated path (blue curve), and the flight path (black curve) in the video obtained from the aircraft's on-board camera, see (b).

Iturrate et al. [53]. However, rather than use measurements of brain activity to specify the desired path step by step, we use measurements of brain activity as evidence to reduce uncertainty about the entire path all at once. The "uncertainty" here refers to the uncertainty the robot has about the desired path (which is known only to the human). Note that our approach does not require the robot to wait until the entire path is specified before beginning to move. Instead, the robot may begin to move immediately along its best estimate of the desired path, using all evidence obtained so far. In order to enable this strategy, we make system design choices that allow us to cast interface design as a communication problem. Our first choice is to model desired paths as strings in an ordered symbolic language for representing smooth planar curves. Our second choice is to model the neural sensor as a communication channel—in particular, to model EEG with left- and right-hand motor imagery as a binary symmetric channel. Our third choice is to use a graphical display for providing feedback to the user. With these choices, the underlying problem is to design a provably optimal communication protocol that says how the user should provide inputs, and how the interface should generate feedback. We solve this problem by using the optimal feedback policy derived in Chapter 3.

We compare our approach to the hybrid strategy of Iturrate et al. [53] in simulation, and show that our approach performs better for navigating a robot moving at a fixed speed. We emphasize that the goal of this study is not to provide a comparison between process control, goal selection, or hybrid approaches. Such a comparison has been performed in [41]. Instead, our goal is to propose and evaluate a hybrid approach that outperforms previous hybrid approaches in navigating a robot moving at a fixed speed.

Our BMI allows a human pilot to fly a simulated aircraft at a fixed speed and altitude using EEG (Fig. 7.1). Using a preliminary version of the interface, we demonstrated EEG-based teleoperation of a physical model-aircraft in a perimeter surveillance task [2]. Although our pilot achieved the task by successfully flying the aircraft with EEG over a 3km perimeter in 5 minutes, this preliminary version had two drawbacks. First, it required access to an overhead map of the environment to inform the pilot about the aircraft's possible routes. Second, it required the pilot to be very good at providing input commands to the interface (e.g., 90% accuracy at decoding EEG signals). Here, we address the first issue by displaying possible routes directly on video streamed from the aircraft's onboard camera. We address the second issue by establishing a systematic way to choose interface parameters based on measurements of the pilot's ability to provide input commands. We forego hardware experiments in order to perform a focused analysis of our interface using a high fidelity flight simulation environment with many subjects—which makes a hardware demonstration impractical. We also emphasize that we use an existing algorithm for decoding input commands from EEG signals, and the goal of this work is not to advance the state-of-the-art in EEG signal processing.

7.2 Method for Navigating a Robot in a Planar Workspace

Our goal is to design a brain-machine interface that allows a human user to navigate a mobile robot in a planar workspace with EEG signals. First, we consider the problem of BMI design for specifying a desired path for a stationary robot, and cast it as a communication problem that consists of designing an optimal communication protocol (Section 7.2.1). Then, we

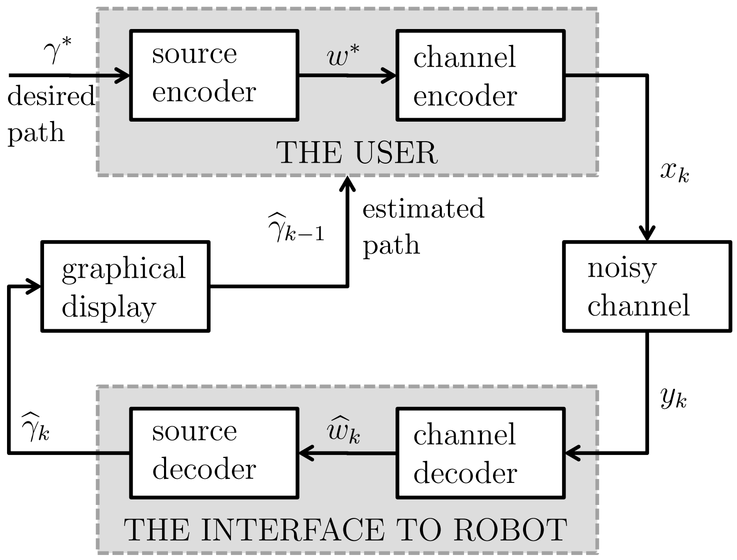


Figure 7.2: Our abstraction of brain-robot interface design as a communication problem. The human user conveys their desired path γ^* to the robot through a binary symmetric channel with causal feedback provided by a graphical display.

describe how the optimal feedback policy described in Chapter 3 can be used to allow a human user specify a desired path with vanishing error probability (Section 7.2.2). Finally, we apply this policy in our BMI design for specifying a desired path for a robot moving at a fixed speed (Section 7.2.3).

7.2.1 Interface design as a communication problem

In this section, we make three choices about system architecture that allow us to cast brain-machine interface design as a communication problem.

Structure of Desired Paths Our first choice is to represent desired paths as strings in a symbolic language (Chapter 5). In particular, we use the alphabet of circular arcs for constructing planar paths (Section 5.2.1). This alphabet consists of symbols $\Sigma = \{\sigma_1, \sigma_2, \dots, M\}$ that correspond to fixed-length circular arcs. Strings of symbols $\mathbf{v} = (v_1 \cdots v_M)$ correspond to paths that are piecewise smooth.

Measurement and Interpretation of Brain Activity Our second choice is to use a binary classifier to distinguish between left- and right-hand motor imagery in the brain based on EEG signals. We model it as a binary symmetric channel with some crossover probability ϵ [79]. The input to this channel is $x_k \in \{0,1\}$, where we associate $x_k = 0$ with "left" and $x_k = 1$ with "right." The output of this channel is $y_k \in \{0,1\}$, where $P(y_k|x_k) = 1-\epsilon$ if $y_k = x_k$ and $P(y_k|x_k) = \epsilon$ otherwise.

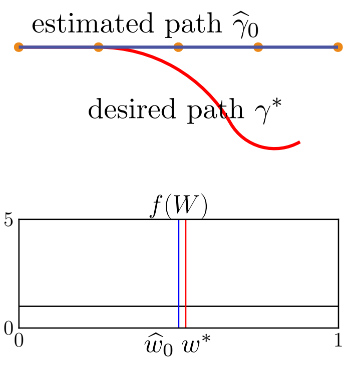
Mechanism for Sensory Feedback Our third choice is to use a graphical display to show candidate paths $\widehat{\gamma}$ to the human user. As we will see in Section 7.2.2, the best choice of $\widehat{\gamma}$ to show at time step k is an estimate of the user's desired path γ^* given the statistical model associated with our symbolic language and the outputs $y_1, \ldots, y_{k-1}$ of the binary symmetric channel we defined above. This visual feedback is "causal" in the sense that it depends only on prior outputs of the channel (i.e., only on the past history of EEG signals). We also model this feedback as "noiseless" in the sense that we assume the human user can decide with perfect accuracy whether or not $\gamma^* < \widehat{\gamma}$, i.e., whether or not their desired path is to the left of the candidate path.

With these choices, our goal has now become the design of a protocol to communicate a string γ^* in an ordered symbolic language across a binary symmetric channel with causal noiseless feedback. Such a protocol consists of an encoder (essentially mapping the estimate $\widehat{\gamma}_{k-1}$ to the input x_k) and a decoder (essentially mapping the outputs $y_1, \ldots, y_k$ to the estimate $\widehat{\gamma}_k$), as shown in Fig. 7.2.

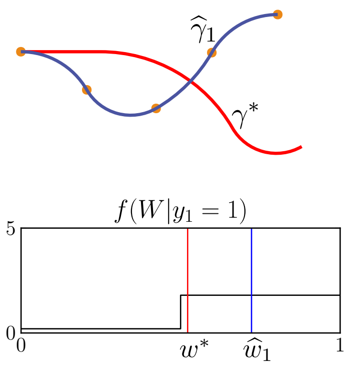
7.2.2 Optimal feedback policy for specifying desired paths

The choices we made in the previous section allows us to use the optimal feedback policy for querying the human user's desired path. In Chapter 5, we constructed an invertable mapping $\psi : \Sigma^* \to [0,1)$ from random strings $\mathbf{V} \in \Sigma^*$ distributed according to a given statistical model $P(\mathbf{V})$ to message points $W \in [0,1)$ distributed uniformly in the unit interval. We apply the optimal feedback policy described in Chapter 3 to allow a human user specify a desired path $\gamma^* \in \Sigma^*$, or equivalently, the message point $w^* = \psi(\gamma^*)$ in the unit interval $[0,1)$, using a sequence of noisy binary inputs.

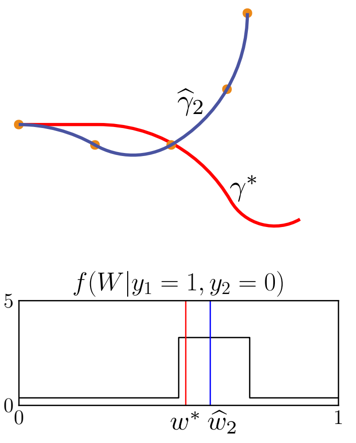
This policy is not only optimal but also easy for a human user—the "encoder"—to implement. Assume a graphical display shows the user the



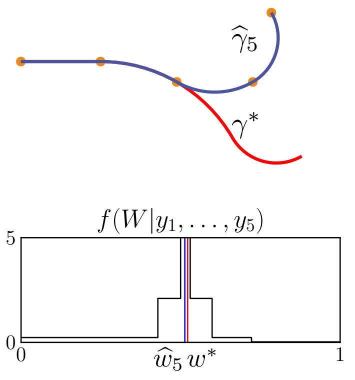
(a) Step 0



(b) Step 1



(c) Step 2



(d) Step 5

Figure 7.3: Four iterations of the communication protocol between the user and the interface to the robot. The interface maintains a posterior distribution over the unit interval and displays the path that corresponds to the median of this distribution. The user responds by comparing the desired path to the estimated path using lexicographic ordering. See text for details.

path $\widehat{\gamma}_{k-1} = \psi^{-1}(\widehat{w}_{k-1})$ that corresponds to the estimate $\widehat{w}_{k-1}$. Then, the user only has to decide if their desired path γ^* appears lexicographically to the left (hence $x_k = 0$) or to the right (hence $x_k = 1$) of $\widehat{\gamma}_{k-1}$. Figure 7.3 shows an example. Initially, the posterior is uniform and the median $\widehat{w}_0$ corresponds to the straight path given by $\widehat{\gamma}_0$. Because γ^* turns more right than $\widehat{\gamma}_0$, the user provides $x_1 = 1$. After a true observation $y_1 = 1$, the interface updates the posterior—increasing the probability of all paths to the right of $\widehat{\gamma}_0$ and decreasing the probability of all paths to the left of $\widehat{\gamma}_0$— and generates a new estimate $\widehat{w}_1$. In this case, the estimated path $\widehat{\gamma}_1$ moves to the right of γ^*, so the user provides $x_2 = 0$. As the interface receives more inputs, the posterior concentrates on smaller intervals around the desired path and a longer prefix of the estimated path matches with the desired path. For instance, see the posterior and the estimated path after five "correct" inputs in Fig. 7.3d.

7.2.3 Application to navigation of a moving robot

In this section, we apply the optimal communication protocol derived in the previous section to enable navigation of a mobile robot that moves at a fixed speed. We make use of the following definitions:

- π_{desired}: user's desired path γ^* (not known to the robot).
- π_{estimate}: the interface's current estimate of π_{desired}.
- π_{track}: the path the robot is following at a fixed speed.
- π_{initial}: the path the robot follows before π_{estimate} starts.

The procedure implemented by the human user to specify π_{desired} is described in Algorithm 2. The user provides either a "left" (left-hand motor imagery) or a "right" (right-hand motor imagery) input to indicate whether π_{desired} turns "more left" or "more right" than π_{estimate}, which is displayed by the interface as part of feedback. This is easy to implement for a trained human eye since it only requires a visual search in the local neighborhood of π_{estimate}.

The procedure implemented by the interface to move the robot along the path being specified by the user is described in Algorithm 3. At first, π_{track} begins with a fixed π_{initial}, a straight path of some length, so that the interface can obtain several user inputs till the robot approaches the end of π_{initial}. The estimated path π_{estimate} is initialized to $\widehat{\gamma}_0$, and updated after the k-th user input to $\widehat{\gamma}_k$ according to the policy (Section 7.2.2). In particular, after observing y_k, the message point $\widehat{w}_k \in [0, 1)$ is computed using the optimal feedback policy and then decoded as a path $\widehat{\gamma}_k = \psi^{-1}(\widehat{w}_k)$ using Algorithm 1. The robot moves along π_{track} at a fixed speed, and whenever it approaches the end of π_{track}, the interface appends the first symbol of π_{estimate} to π_{track}.

Algorithm 2 Human User's Algorithm for Providing Inputs.

```
1: $k \leftarrow 1$
2: loop
3:     Observe $\pi_{\text{estimate}}$ and robot state
4:     if $\pi_{\text{desired}} < \pi_{\text{estimate}}$ then
5:         Input "left" $(x_k = 0)$ by left-hand motor imagery
6:     else
7:         Input "right" $(x_k = 1)$ by right-hand motor imagery
8:     end if
9:     $k \leftarrow k + 1$
10: end loop
```

Algorithm 3 Robot Navigation Algorithm.

```
1: $\pi_{\text{track}} \leftarrow \pi_{\text{initial}}$
2: $\pi_{\text{estimate}} \leftarrow \widehat{\gamma}_0$
3: $k \leftarrow 1$
4: loop
5:     Display $\pi_{\text{estimate}}$ and robot state
6:     Move robot along $\pi_{\text{track}}$ at fixed speed
7:     if User's $k$th input is observed then
8:         Compute $\widehat{w}_k$ from $y_k$ using the optimal feedback policy
9:         Compute $\widehat{\gamma}_k$ from $\widehat{w}_k$ using Algorithm 1
10:        $\pi_{\text{estimate}} \leftarrow \widehat{\gamma}_k$
11:        $k \leftarrow k + 1$
12:    end if
13:    if Robot moved till the end of $\pi_{\text{track}}$ then
14:        Append the first symbol of $\pi_{\text{estimate}}$ to $\pi_{\text{track}}$
15:        Shift $\pi_{\text{estimate}}$ one symbol ahead
16:    end if
17: end loop
```

π_{estimate} is then shifted one symbol ahead, which in the protocol corresponds to assigning zero probability to paths that do not begin with the appended symbol, and normalizing the posterior so that $\widehat{w}_k$ remains as the median.

7.3 Interface for Flying a Simulated Aircraft with EEG

This section describes the implementation of our brain-machine interface for flying a simulated aircraft at a fixed speed and altitude with input only from EEG.

7.3.1 The simulated aircraft

We used an high-fidelity flight simulation environment for model aircrafts based on Fs One®RC flight simulator [124]. The simulated aircraft was controlled by an autopilot that implemented a receding-horizon linear quadratic regulator to fly over paths specified by the user [125]. We fixed the aircraft's speed to 25m/s, and the aircraft's altitude to 200m. The aircraft's camera faced towards the direction of the flight, was inclined 45 degrees with respect to the ground, and was roll-angle stabilized by the autopilot to ensure that the human pilot had a good field of view of the ground ahead of the aircraft.

7.3.2 The feedback stimuli

The interface provided visual feedback to the human user by showing real-time state and video obtained from the aircraft and the paths π_{estimate}, denoted estimated path, and π_{track} (without π_{initial}), denoted flight path, in a graphical display (see Fig. 7.1b). These paths were augmented into the video frames by placing them in a horizontal plane just above the ground-level in the 3D workspace, and then projecting them onto the 2D image plane.

7.3.3 Configuration of the interface

In this section, we describe the important parameters that affected the performance of our interface.

Crossover probability

Crossover probability was the fraction of input commands that were expected to be corrupted due to noise in decoding EEG signals (Section 7.2.1). This probability was estimated by comparing observed input commands with ground-truth input commands at the beginning of each experimental session (see Section 7.4.4).

Symbol-length

Symbol-length affected the trade-off between how "expressive" (the degree in which our space paths approximated the space of paths the pilot might desire to fly over) and how “compact” (the expected number of input commands that were necessary to infer the pilot’s desired path correctly) our space of paths was. In order to balance this trade-off, symbol-length was configured adaptively with respect to the user’s performance in providing input commands at the beginning of each experimental session (see Section 7.4.4). We restricted symbol-length to be between 100m and 500m, because we assumed that values smaller than 100m might put an excessive cognitive load on users, and values larger than 500m might not provide a space of paths expressive enough to accomplish our experimental tasks (see Section 7.4.1).

Alphabet of symbols

We used the alphabet shown in Fig. 5.2. It consisted of seven circular arcs with central angles evenly distributed in $[-\pi/2, \pi/2]$. We empirically found this alphabet to provide a good balance between expressiveness and compactness of the generated paths.

Statistical model

We assumed a zeroth-order Markov model given by a discrete Gaussian kernel centered on the straight arc, which was denoted “model1” and illustrated in Fig. 7.5. In this statistical model, the straight arc had the highest probability and taking a wider turn was more likely than taking a sharper turn. This corresponded to our prior belief on the structure of paths that pilots could prefer to fly the aircraft over.

Startup delay

In the beginning of flight, the aircraft flew over a straight path (π_{initial}) of two symbol-lengths before flying over the path specified by the user. We empirically found that using a π_{initial} of two symbol-lengths provided a significant improvement over using a π_{initial} of one symbol-length in Monte Carlo experiments (see Section 7.4.3).

Estimated-path length

Estimated paths were decoded up to four symbols, i.e., $M = 4$ in Algorithm 1. A large number of symbols was not desired because all symbols except the first few ones might have an almost zero probability of being part of the user's desired path and showing many symbols might make the display cluttered and more difficult for human users to provide their inputs. In contrast, a small number of symbols might lead to a situation where the estimated path overlaps with the pilot's desired path, causing the user's algorithm for providing inputs to fail. We empirically found that this situation was mostly avoided when $M \geq 4$.

7.3.4 Decoding input commands from EEG signals

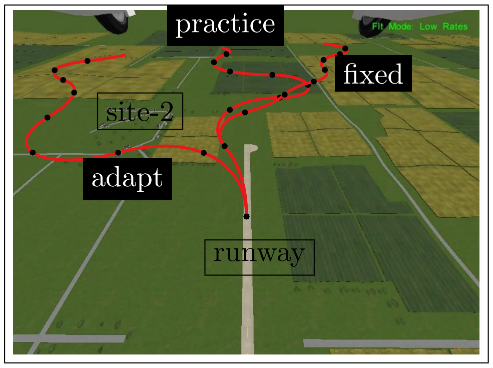
The interface decoded the pilot's input commands from EEG signals using an asynchronous classifier implemented in C. EEG signals were collected by eight electrodes positioned on the scalp at $F3, F4, C3, C4, T7, T8, P3, P4$ with ground measured at Fpz, and reference measured at Cz [126]. Measured signals were amplified (James-Long Co.), low-pass filtered, and synchronously sampled at 400Hz by an IOtech Personal Daq 3000 A/D converter. The classifier was trained using the algorithm in [127], which used common spatial analytic pattern (CSAP) to extract discriminative signals that capture large disparities for each input command by viewing it as a blind-source separation problem. Incoming signals were processed at 15Hz, using a hidden Markov model (HMM), to perform classification by belief propagation. The decoding occurred asynchronously, i.e., when the belief probability exceeded a threshold.

7.4 Experimental Procedure

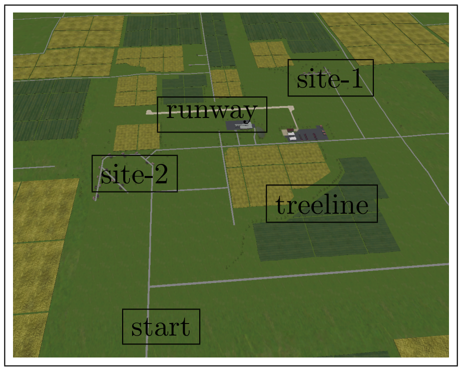
This section describes the experimental procedure we used to evaluate the method described in Section 7.2, and the BMI described in Section 7.3.

7.4.1 Experimental tasks

We used the following two tasks to evaluate our interface.



(a) The target paths selected for the tracking task



(b) The target terrain structures selected for the high-level task

Figure 7.4: Illustration of the experimental tasks used in the EEG experiments. The target paths selected for the tracking task in the practice, adapt, and fixed phases were shown in (a), assuming a symbol-length of 330m. In the high-level task, illustrated in (b), there was no specific target path to track, instead the objective was to fly the aircraft first over the treeline, and then over the house site-1 and site-2.

Tracking task

The goal was to fly the aircraft over a given target path, which was displayed to the user during flight (see Fig. 7.1b). This task measured our interface's ability in allowing users to fly the aircraft over their desired paths. In order to succeed in the task, the human user was required to specify a path that matched the target path symbol by symbol. The interface terminated a run of a tracking task as soon as the flight path deviated from the target path in order to provide a fair comparison of performance across different runs. We generated target paths randomly according to a given statistical model and symbol-length. The target paths consisted of 3km/symbol-length (rounded to nearest integer) symbols, hence they were approximately 3km long, which was chosen to restrict the duration of an experimental run to about two minutes—the time it took the aircraft to fly a 3km distance.

High-level task

The goal was to fly the aircraft over a given list of target terrain structures, which were indicated on an overhead map of the environment and were shown to the user before flight. Unlike the tracking task, the user was not required to fly the aircraft over a specific target path, instead the actual path specification was completely left up to the user. This task measured the efficacy of our interface when the desired path was not shown as reference in the display. In order to succeed in the task, the human user was required to specify a path that hit all targets in the given order. Hitting targets in an arbitrary order was not allowed to provide a fair comparison of performance across subjects. We identified three targets on the map: a treeline, and two house sites (see Fig. 7.4b). We said that a house site was hit by a path if the site center was within 250m of some point along the path, and a treeline was hit by a path if all points along the treeline were within 250m of some point along the path.

7.4.2 Evaluation criteria

We used the following measures to evaluate the performance of our approach in Monte Carlo experiments:

- **success rate**: the fraction of trials that were successful in Monte Carlo simulations of a thousand trials for a given symbol-length.
- **safe symbol-length**: the smallest symbol-length less than or equal to 500m, at which a success rate of 90% or higher was achieved.

We used the following measures to evaluate the input performance observed during a tracking task:

- **input error (E):** fraction of input commands that were incorrect across all input commands.
- **input rate (R):** average number of input commands (bits) received per second.
- **information transfer rate (ITR):** number of reliable bits per second given by
$$R\left(\log n + E\log E + (1-E)\log\frac{(1-E)}{n-1}\right),$$
where n is the number of input classes.

We used the following measures to evaluate the subject's performance in flying the aircraft using our interface:

- **run success:** a run was said to be successful if the task performed in the run was successful.
- **symbol-count (SC):** number of symbols in the flight path that matched the target path in the tracking task.
- **hit-count (HC):** number of targets that were hit by the flight path in the high-level task.

7.4.3 Monte Carlo experiments

We evaluated the performance of our approach (Section 7.2) and the hybrid approach of [53] under different configurations by performing Monte Carlo experiments. In these experiments, the performance was measured by running several trials of the tracking task with randomly sampled inputs under each configuration. The compared configurations are described below.

Sequential-select

In this configuration, we implemented the hybrid approach of [53] as follows. We allowed the user to specify a desired path symbol by symbol using three inputs {“left”,“right”,“rest”}. Recall that we denoted the alphabet of symbols by $\Sigma = \{\sigma_1, \ldots, \sigma_7\}$, and assumed that the symbols were ordered so that $\sigma_i < \sigma_j$ if and only if $i < j$. Let $v_m \in \Sigma$ be the m-th symbol in π_{desired}, and v_0 be a pre-determined symbol that the robot followed at start. The user provided inputs to specify v_m until the robot approached the end of v_{m-1}. Let $\widehat{v}_m \in \Sigma$ be the m-th symbol chosen so far by the user. At first $\widehat{v}_m$ was σ_4. The user provided “left” if $v_m < \widehat{v}_m$, “right” if $v_m > \widehat{v}_m$, and “rest” if $v_m = \widehat{v}_m$. Upon receiving a “left” or “right” input, the interface updated $\widehat{v}_m = \sigma_i$ to be σ_{i-1} (if $i > 1$) or σ_{i+1} (if $i < 7$), respectively. Note that “rest” inputs were discarded by the interface, and v_m was set to $\widehat{v}_m$ when the robot approached the end of v_{m-1}, at which point specification of v_{m+1} started.

Robust-sequential-select

In this configuration, we attempted to increase the robustness of the hybrid approach of [53] by making use of the “rest” inputs in sequential-select as follows. After receiving a “rest” input, the interface entered into a “lock” mode where “left” or “right” inputs did not immediately change the current selection $\widehat{v}_m$. In “lock” mode, the interface maintained a posterior probability P over the correct input X. Let Y_k be the random variable denoting the k-th observation in the “lock” mode, with Y_0 = “rest”. The interface assumed a uniform prior P_X, and computed the posterior after each Y_k, $k = 0, 1, \ldots$ using Bayes’ rule as $P(x|y_0, \ldots, y_k) = \eta P_{Y|X}(y_k|x)P(x|y_0, \ldots, y_{k-1})$, where η was a normalizing constant. Upon receiving an observation y_i = “left” (or “right”, analogously), if the posterior probability of “left” after $y_0, \ldots, y_i$ was greater than that of “rest”, the interface terminated the “lock” mode and updated $\widehat{v}_m$ accordingly. The “lock” mode was also terminated when the specification of the next symbol started.

Our approach + uniform

In this configuration, our approach used a statistical model with a uniform prior over symbols, denoted “uniform”, and the robot followed an initial path of one symbol-length at start. This configuration provided a fair comparison against sequential-select and robust-sequential-select, because all configurations shared the same statistical model and the startup condition.

Our approach + delay + uniform

In this configuration, the robot followed an initial path of two symbol-lengths at start. Note that following such a path introduced a delay that was longer than the delay in the previous configurations, and this might lead to a better performance because the user had more time to specify π_{desired}.

Our approach + delay + model1

In this configuration, which was the same as the configuration of our interface, the statistical model used by our approach was “model1”, as described in Section 7.3.3 and illustrated in Fig. 7.5.

Our approach + delay + model2

In this configuration, our approach used a statistical model, denoted “model2” and illustrated in Fig. 7.5, that was a zeroth-order Markov model given by a discrete Gaussian kernel centered on the straight arc, with a variance smaller than the variance used in "model1" and with the probabilities of σ_1, σ_7 set to 0.05. The purpose of evaluating this configuration was to see how much performance could be gained if desired paths were generated according to a model with less uncertainty such as "model2".

In Monte Carlo experiments, we randomly sampled inputs at every second ($R = 1$) to yield the same ITR for each configuration. In particular, to yield ITRs of $0.75, 0.50, 0.25$ bits/second, the simulated inputs satisfied an input error E of $0.04, 0.11, 0.21$ for $n = 2$, and $0.17, 0.26, 0.38$ for $n = 3$, respectively from lowest to highest ITR. The probability transition matrix $P_{Y|X}$, which specified the conditional probability of generating Y when the correct input was X, was such that $P_{Y|X}(y|x) = 1-E$ if $y = x$, and $P_{Y|X}(y|x) = E/(n-1)$ otherwise. We computed success rate as a function of symbol-length for each ITR in $\{0.75, 0.50, 0.25\}$, by running a thousand trials for each symbol-length from 100m to 500m with increments of 10m. We computed safe symbol-length from the resulting success rates. We also note that the target paths were randomly sampled according to the statistical model used by each configuration.

7.4.4 EEG experiments

We evaluated the performance of our brain-machine interface (Section 7.3) with 20 able-bodied subjects that were right-handed, between the ages of 20 and 30, and had normal or corrected-to-normal vision. Only two subjects had prior experience in EEG motor-imagery based BMI studies. In total, 57 experimental sessions were performed by these subjects. An experimental session consisted of the following five phases in the given order. This study was approved by the Institutional Review Board of the University of Illinois.

Training phase

We collected labeled EEG data to train the decoding algorithm by displaying visual prompts, either a left or a right arrow, for a duration of four minutes. The prompts were chosen according to a randomly generated sequence containing 30 "left" and 30 "right" input commands. Each prompt lasted four seconds with no break period in between the prompts. The decoding algorithm was trained only once using the EEG data collected from these 60 prompts.

Practice phase

Subjects performed several runs of the tracking task, denoted as practice-runs, using a keyboard instead of EEG to provide "left" and "right" inputs until they succeeded in the task. The goal was to make the subject well acquainted with the use of the interface. We set the crossover probability to 4% and the symbol-length to 100m so that the subjects could succeed in the task only by choosing their input commands accurately to yield a low input error, and by providing these inputs quickly to yield a high input rate. This phase ended after the subject succeeded in the tracking task using keyboard.

Adapt phase

Several runs of the tracking task, denoted as adapt-runs, were performed to configure the two important parameters, crossover probability and symbol-length (Section 7.3.3), with respect to the subject's performance in providing input commands through EEG motor imagery. After each adapt-run, we measured the input rate, the input error and the run success (Section 7.4.2). In the first run, we set the parameters to their pre-determined values (crossover probability was 15% and symbol-length was 337m). In all future runs, the interface chose crossover probability to be the input error of the previous run, and symbol-length to be the safe symbol-length computed from Monte Carlo simulations of the tracking task using random inputs yielding the input error and input rate of the previous run. If the safe symbol-length did not exist, symbol-length was set to 500m. The subject performed a new run until the chosen parameter values satisfied the following convergence criteria: the subject succeeded in the last run, the chosen symbol-length was within some tolerance (100m) of the symbol-length used in the last run, the subject performed at least 3 runs, and the subject performed at least 5 runs if the chosen symbol-length was longer than 400m. If the convergence criteria were met, the interface, future phases, and future runs were said to be properly configured. This phase ended after the interface was properly configured, in which case it was considered a successful adapt-phase, or after 10 adapt-runs.

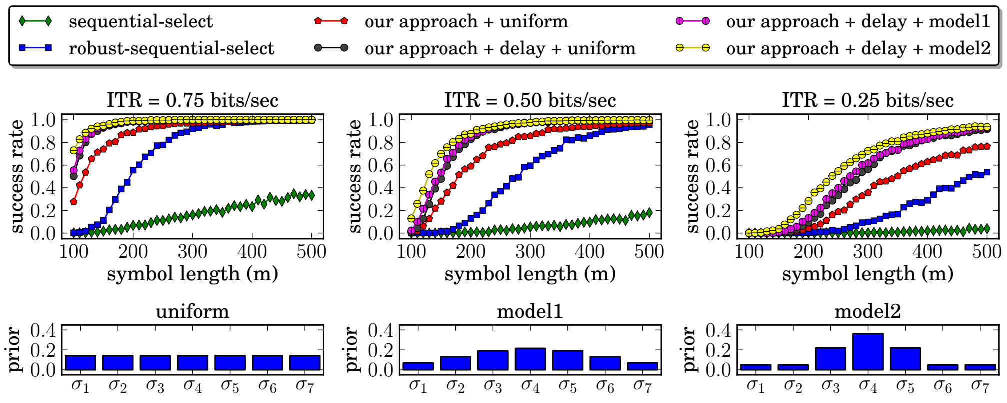


Figure 7.5: Results of Monte Carlo experiments comparing different configurations of our approach and the hybrid approach of [53]. The figures at top show the success rate of each method in our navigation task as a function of symbol-length. The figures at bottom show the priors for each statistical model.

Fixed phase

The subjects performed several runs of the tracking task, denoted as fixed-runs, with the parameters chosen in the adapt phase. This phase ended after a successful fixed-run, in which case it was considered a successful fixed-phase or after 10 fixed-runs.

Free phase

The qualified subjects performed severals runs of the high-level task, denoted as free-runs, with the parameters chosen in the adapt phase. A subject was qualified for free phase if the preceding fixed phase was successful. The number of free-runs performed in this phase was determined by the pilot and the operator of the experiment. A free-phase was said to be successful if at least one of the free-runs was successful.

We note that in all runs of the tracking task in the practice, adapt, and fixed phases, the aircraft started flight from the same point, but the sequence of symbols in the target path were unique to each phase (see Fig. 7.4a). In the free phase, the aircraft started flight from a different point to make the terrain flown over in free-runs different than the terrain flown over in runs of the tracking task.

7.5 Experimental Results

This section describes the results obtained from the experiments described in the previous section.

7.5.1 Results from Monte Carlo experiments

Results show that our approach outperformed our implementations of the hybrid approach of [53] under all settings considered in Monte Carlo experiments, and the performance of our approach could be improved by increasing the startup delay or by using non-uniform statistical models. Figure 7.5 shows the success rates as a function of symbol-length under three different ITRs and Table 7.1 shows the safe symbol-lengths for each configuration. We summarize the results as follows:

- Sequential-select was not robust to input errors, and did not have a safe symbol-length. Failures in sequential-select trials might be due to false observations of "rest" inputs that occurred shortly before the end of the time window for symbol selection.

- Robust-sequential-select provided a significant improvement over sequential-select, and had safe symbol-lengths except for the lowest ITR.

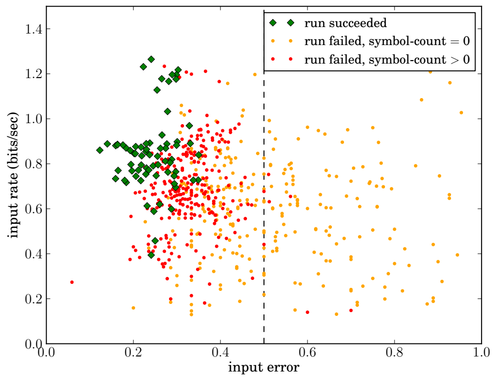
- The baseline configuration of our approach, "our approach + uniform" outperformed sequential-select and robust-sequential-select under all considered values of symbol-length and ITR.

- Increasing startup delay improved the performance significantly. It shortened the safe symbol-length more than 70m for the two higher ITRs, and unlike the baseline configuration of our approach, it had a safe symbol-length for the lowest ITR.

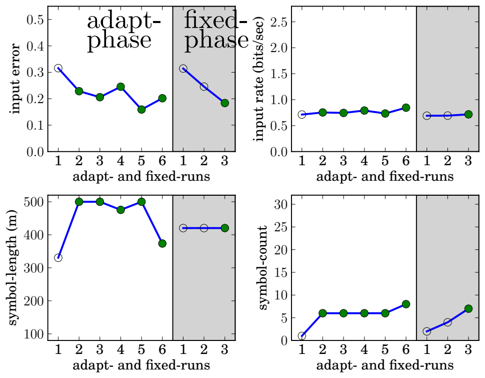
- The performance obtained with the three statistical models were comparable, with "model2" yielding marginally better results especially for the lowest ITR. The performance differences between the statistical models could be explained by the entropy of the model priors, which were 2.81, 2.70, and 2.35 for "uniform", "model1", and "model2", respectively.

Table 7.1: Safe symbol-lengths

Method	ITR=0.75	ITR=0.50	ITR=0.25
sequential-select	N/A	N/A	N/A
robust-sequential-select	300	430	N/A
our approach + uniform	210	330	N/A
our approach + delay + uniform	140	220	480
our approach + delay + model1	140	220	470
our approach + delay + model2	130	210	420



(a) Results in all adapt- and fixed-runs



(b) Results in adapt- and fixed-phases of a sample session

Figure 7.6: Results obtained in adapt- and fixed-runs of all runs, shown in (a), and of a sample session, shown in (b). Each marker in (a) denotes the input error, the input rate and the success of a run. In the sample session, the adapt-phase consisted of 6 runs (unshaded region) and the fixed-phase consisted of 3 runs (shaded region). Successful runs were denoted with a green solid circle in (b).

7.5.2 Results from EEG experiments

Results show that our interface allowed subjects to specify desired paths accurately for our simulated aircraft even under very low ITRs. Observed input performances with EEG were very low, which led to the failure of many runs (Section 7.5.2), and a low number of successful adapt phases (Section 7.5.2). Despite these low input performances, half of the subjects succeeded in the fixed phase (Section 7.5.2), and most of the qualified subjects succeeded in the free phase (Section 7.5.2).

Results with tracking task

Subjects performed several runs of the tracking task by providing inputs through a keyboard in the practice phase, and through EEG in the adapt and fixed phases. Results show that input performances with EEG were significantly lower than the performances with a keyboard due to errors in decoding EEG signals, and only the runs with a sufficient level of input performance were successful. In the practice phase, all subjects succeeded in the tracking task after a few trials and became well acquainted with the use of the interface. In successful practice-runs, subjects yielded on average an input error of 0.03, an input rate of 1.02, and an ITR of 0.86. In the adapt and fixed phases, in total, subjects performed 622 runs, of which only 78% had an input error less than 0.5, only 50% had an ITR greater than 0.05, only 11% were successful (Fig. 7.6a). In successful adapt- and fixed-runs, subjects yielded on average an input error of 0.24, an input rate of 0.85, and an ITR of 0.18. Table 7.2 shows results obtained in adapt- and fixed- runs for each subject. Out of 20 subjects, 15 of them (subjects A-O) succeeded in at least one of the runs, and 3 of them (subjects A,B,C) succeeded in most of the runs by yielding on average an ITR greater than 0.15 and a symbol-count larger than 5 symbols. The best symbol-count, 13 symbols (each with length $223m$), was achieved by subject A in one of the adapt runs.

Results in adapt phases

The interface was properly configured in only 30% of the sessions (17 out of 57). The failure in adapt-phases could be explained by the observation that the ITRs (0.05 bits on average) in failed adapt-phases were significantly lower ($p < 0.01$) than the ITRs (0.12 bits on average) in successful adapt phases. Fig. 7.6b shows the results obtained in the adapt phase of a sample session. Here, the interface was properly configured after 6 adapt-runs, with crossover probability set to 0.20 (input error observed in the last adapt-run), and symbol-length set to 420m. On average, in successful adapt-phases, 7 adapt-runs were performed, and crossover probability was set to 0.25. In the majority of these phases, the chosen symbol-length was 500m, the largest value allowed, and in the rest of them, it was $345, 420, 466, 484, 490$m from smallest to largest.

Table 7.2: Experimental Results for Each Subject

subject	input error[1]	input rate[1]	mean ITR[1]	best ITR[1]	mean SC[1]	best SC[1]	adapt-phases[2]	fixed-phases[3]	fixed-runs[4]	free-runs[5]
A	0.23	0.84	0.21	0.40	6.44	13	3/3	2/2	2/2	3, 3
B	0.23	0.82	0.19	0.32	5.85	10	2/2	2/2	2/2	3, 3, 3, 1
C	0.28	1.03	0.15	0.21	5.50	8	1/1	1/1	1/2	
D	0.29	0.55	0.08	0.22	2.50	6	1/2	1/1	1/3	3
E	0.27	0.69	0.12	0.27	2.32	8	1/3	1/1	1/3	3, 1, 1
F	0.34	0.88	0.07	0.14	2.15	6	2/3	2/2	2/5	3, 1, 0, 0
G	0.34	0.91	0.08	0.29	1.71	7	2/5	2/2	2/3	3, 3, 2, 2, 0, 0
H	0.34	0.62	0.06	0.13	1.26	6	1/3	1/1	1/5	2, 1, 0
I	0.34	0.61	0.05	0.14	1.26	6	2/5	2/2	2/3	2, 2, 2, 2
J	0.30	0.48	0.06	0.12	0.90	6	1/3	1/1	1/7	2
K	0.35	0.62	0.04	0.12	0.93	6	1/3	0/1	0/10	
L	0.36	0.72	0.04	0.11	1.42	6	0/4	0/0[7]		3, 3
M	0.32	0.71	0.07	0.16	2.32	6	0/1[6]	0/0[7]		
N	0.26	0.64	0.12	0.18	3.78	6	0/1[6]	0/0		
O	0.35	0.7	0.05	0.11	0.77	6	0/3[6]	0/0		
PQRST[8]	0.36	0.57	0.04	0.08	0.74	3.6	0/15	0/0		

(1) average and best values computed across all adapt- and fixed- runs with input error less than 0.5
(2) number of successful adapt-phases / number of adapt-phases the subject participated
(3) number of successful properly-configured fixed-phases / number of properly-configured fixed-phases the subject participated
(4) number of successful properly-configured fixed-runs / number of properly-configured fixed-runs
(5) list of hit-counts (sorted from highest to lowest) obtained in all free-runs
(6) at least one of the adapt-runs was successful
(7) subject succeeded in a fixed-phase that was not properly configured (allowed to participate in the free-phase)
(8) collective results of 5 subjects that did not succeed in any adapt- or fixed-runs

Results in properly-configured fixed phases

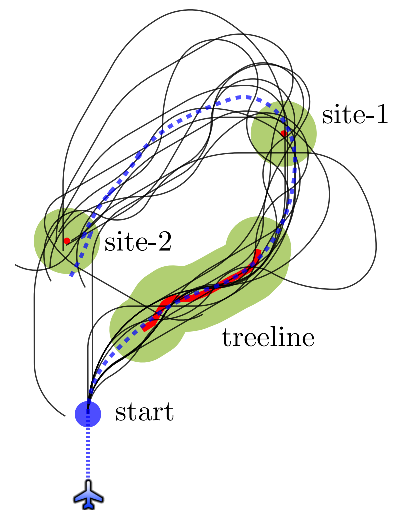
Results show that subjects succeeded in the tracking task after the interface was properly configured, if their input performances were comparable to the input performances that had been used to configure the interface. In total, 16 properly-configured phases were performed by 11 subjects, and all these phases except one were successful (Table 7.2). Fig. 7.6b shows the results obtained in the fixed phase of a sample session. Here, the subject succeeded in the fixed-phase after failing in the first two fixed-runs. This failure could be explained by observing that the input errors in these runs were greater than the input error in the last adapt-run, which was used to configure the interface, while the input rates were similar. In fact, 89% of the failures in properly-configured fixed-runs could be explained by the same observation.

Results in free phases with high-level task

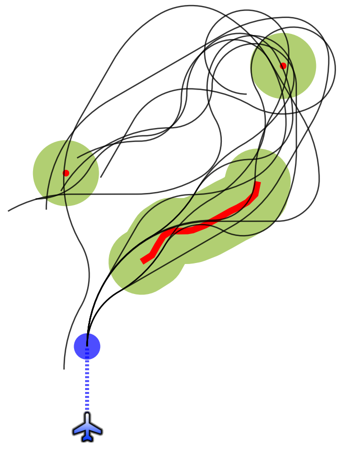
Results show that qualified subjects could fly the aircraft over their desired path in the absence of a specific target path shown as part of the feedback stimuli. Ten subjects were qualified for the free phase by succeeding in the preceding fixed phase. Seven of these subjects succeeded in the high-level task in at least one of their free-runs (Table 7.2). Out of 30 free-runs performed in total, 12 runs were successful (hit-count was 3), and in 8 runs hit-count was 2. Figure 7.7 shows the flight paths obtained in all free-runs. Across successful free-runs, the length of the flight path from start to the first point that hit the third target (site-2) was between 4.8 and 6.2km with a mean of 5.4 km. The mean flight path computed by averaging all paths from successful free-runs closely matched our expectation of the path subjects would fly the aircraft over.

7.6 Conclusion

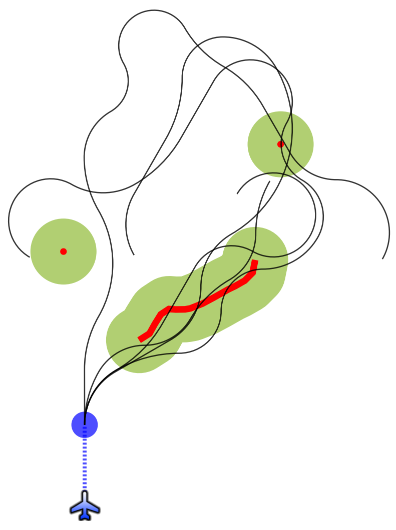
In this study, we presented an EEG-based brain-machine interface for flying a simulated aircraft at a fixed speed and altitude with noisy binary inputs that were provided by imagining either left- or right-hand movements in the brain. Our approach was to construct an optimal communication protocol that said how user inputs and sensory feedback must be generated in order



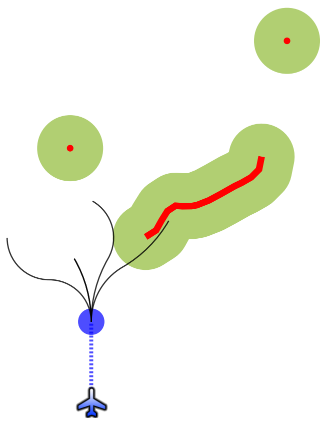
(a) hit-count = 3



(b) hit-count = 2



(c) hit-count = 1



(d) hit-count = 0

Figure 7.7: The flight paths (thin black curves) specified by the subjects in the high-level task, where the objective was to fly the aircraft over three targets: treeline, site-1, and site-2. The frames (a), (b), (c), (d) show the flight paths with hit-counts $= 3, 2, 1, 0$, respectively. A target was hit by a path if the target center (the red dots for site-1, site-2, and the red curve for treeline) was within 250m of some point along the path. The frames also show the points within 250m of the target centers (light-green regions), and the frame (a) shows the mean flight path (dashed blue curve) computed by averaging all paths with hit-count = 3 in the time domain.

to convey the user's desired path to the aircraft as quickly and as robustly as possible. Experimental results showed that our approach outperformed an existing state-of-the-art hybrid approach in navigating a robot moving at a fixed speed and our BMI based on this approach allowed human users to fly a simulated aircraft successfully despite very low ITRs with EEG. Poor input performances might be due to training our EEG decoding algorithm with a small amount of data collected at the start of each experimental session. Better input performances might be obtained by retraining during a session or by using a larger dataset that might include signals from a bigger set of electrodes. The success in our experiments depended critically on the adaptation of the interface to the user's input performance. We enabled users with higher input performances to navigate the robot along more expressive paths by increasing (roughly) the precision at which desired paths were specified. With this adaptation, our interface provided a comfortable experience to our subjects. A discomfort might have been experienced if we were to use symbol-lengths smaller than 100m, but in practice such discomforts might be avoided by adjusting the robot speed accordingly. In the rest of this section, we present the limitations of our approach and how these limitations might be resolved in future work.

7.6.1 Limitations and future work

Choosing structure of desired paths systematically

Our interface used a heuristic alphabet associated with a set of circular arcs and a heuristic statistical model given by a zeroth-order Markov model. In future work, we intend to make the choice of the alphabet for representing paths and the choice of Markov model more systematic by learning them from human-demonstrated data.

Enabling navigation amidst obstacles

In this study, we did not consider obstacle avoidance. It might be possible to incorporate this into our approach by choosing structure of desired paths systematically (see previous paragraph) using a dataset of human-demonstrated paths for navigation amidst obstacles, similar to the work in [128]. Note that this would assign zero probability to paths that collide with obstacles.

Extending our approach to more than binary inputs

In our interface, users specified desired paths only with binary inputs. One way of extending our approach to make use of discrete inputs with more than two choices might be as follows. Recall that in the case with binary inputs, the estimated path after k user inputs, $\hat{\gamma}_k$, partitioned the set of all possible paths Σ^* into two subsets $\{\gamma | \gamma < \hat{\gamma}_k\}$, and $\{\gamma | \gamma \geq \hat{\gamma}_k\}$ with equal posterior probability. Similarly, in the case with m discrete inputs, the interface might choose $m - 1$ paths $\gamma^1, \gamma^2, \ldots, \gamma^{m-1}$ to partition Σ^* into m disjoint subsets such that each subset will have equal posterior probability and the i-th subset will contain only the paths that are ordered between γ^{i-1} and γ^i, where γ^0, γ^m are left-most and right-most paths in Σ^*, respectively. Then, the user might provide an input to indicate the subset that contains their desired path.

Enabling navigation in 3D space

In this study, we assumed that the robot was moving in a 2D space. One way of extending our approach to enable 3D navigation might be as follows. A space curve γ can be defined by its curvature κ, determining how much γ turns left or right, and its torsion τ, determining how much γ bends up or down, at each point along the curve using Frenet-Serret frame [129]. In order to model desired paths, our approach might use an alphabet consisting of (κ, τ) pairs corresponding to curves of fixed length with constant curvature and torsion. Then, it might be possible to design a communication protocol that would rely on user's ability to compare space curves. This comparison might be based on finding the first point at which two space curves differ, and then observing if one of the curves has smaller curvature (i.e., turns more left), or torsion (i.e., bends more down) than the other.

In order for our approach to find a real-world use case, we need further developments that BCI community as a whole needs to address. First, our interface must have a mechanism for starting or stopping the navigation, which might be achieved by using more input classes or different input paradigms. Second, our interface demanded high cognitive load on users. In the future, we might train our EEG decoding algorithm using additional data corresponding to a "rest" class and adjust the robot's speed based on the uncertainty the robot has about the desired path so that users might take a break from providing inputs. Third, our interface used a “locked-down” graphical display to show feedback stimuli. In future work, we might consider presenting feedback using an augmented reality display such as a head mounted or virtual retinal display.

Chapter 8

Enabling Humans to Navigate Simulated Robots Indoors with EEG

8.1 Introduction

This chapter presents an interface that allows a human user to specify a desired path for a mobile robot in a planar workspace with noisy binary inputs that are obtained at low bit-rates through EEG. We represent desired paths as local geodesics with respect to a cost function that is defined so that each path-homotopy class contains exactly one local geodesic (Chapter 6). We apply max-margin structured learning to recover a cost function that is consistent with observations of human walking paths. Our approach is to use the optimal feedback policy described in Chapter 3 to allow users select a local geodesic—equivalently, a path-homotopy class—using a sequence of noisy bits. We validate this approach with experiments that quantify both how well our learned cost function characterizes human walking data and how well human subjects perform with the resulting interface in navigating a robot in a virtual indoor environment with EEG.

It is necessary to restrict the space of all possible paths the user can specify because we cannot describe arbitrary paths with a finite number of inputs. In Chapter 5, we made a heuristic choice, and used an ordered symbolic language to represent paths of piecewise-constant curvature. However, this decision made it hard to incorporate certain types of statistical information. For example, how does path likelihood vary in the presence of obstacles? In Chapter 6, we suggested a more systematic approach, observing that the principle of optimality could be used to generate a compact representation of all paths likely to be seen in the context of a particular application. In particular, we represented a desired path as a local geodesic with respect to a cost function that is defined so that each path-homotopy class contains

A preliminary version of the material in this chapter has been published in [3].

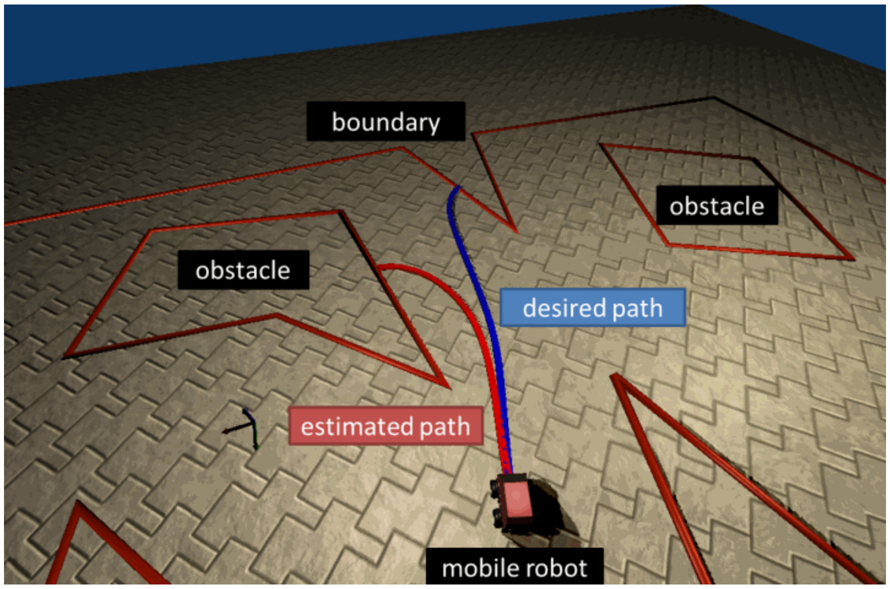


Figure 8.1: Our interface for navigating a mobile robot in a planar workspace with polygonal obstacles. The interface allows a human user to specify a desired path, which corresponds to a local geodesic from the robot's current location to a boundary point. The interface provides feedback by showing its estimate of the user's desired path (called estimated path). The user provides binary inputs by determining the cyclic ordering of their desired path with respect to the estimated path. See text for details.

exactly one geodesic.

In this chapter, we use the representation of desired paths described in Chapter 6, and the optimal feedback policy described in Chapter 3 to build an efficient brain-robot interface. This brain-robot interface is designed to enable a human user navigate a mobile robot indoors along human-like paths (Section 8.2). Section 8.3 describes the implementation of our interface for controlling a simulated robot with EEG. We then describe the experiments used to learn the cost function from a dataset of human-walking paths, and the experiments performed to evaluate the performance of the resulting interface in navigating a simulated robot using EEG (Section 8.4).

8.2 Method for Navigating a Robot along Local Geodesics

8.2.1 Encoding local geodesics as message points in the unit circle

In Chapter 6, we constructed a homeomorphism $\psi : \Gamma^g(q_0) \to D^1$, where $\Gamma^g(q_0)$ is the space of all local geodesics from q_0 in a free configuration space Q_{free}, and D^1 is the unit disk. In this homeomorphism, paths in $\Gamma^g(q_0)$ terminating at a boundary point, denoted $\Gamma^g_b(q_0)$, mapped to the unit circle S^1. This mapping, denoted $\psi_b : \Gamma^g_b(q_0) \to S^1$, allowed us to encode local geodesics in $\Gamma^g_b(q_0)$ as real-valued angles in S^1.

There are three properties of this encoding that facilitates the design of an efficient interface for navigating a mobile robot indoors.

- We can easily compute the probability that a geodesic crosses a particular extension edge. Extension edges were defined during the construction of the homeomorphism $\psi : \Gamma^g(q_0) \to D^1$ (Section 6.3.1). Let γ_l, γ_r be the two geodesics that cross $e_1, \dots, e_{m-1}$, and terminate at an endpoint of the extension edge e_m such that $\psi_b(\gamma_l) < \psi_b(\gamma_r)$. Then, any geodesic γ in $\Gamma^g_b(q_0)$ that crosses $e_1, \dots, e_{m-1}, e_m$ is mapped to an angle $\theta = \psi_b(\gamma)$ such that $\psi_b(\gamma_l) < \theta < \psi_b(\gamma_r)$. This means that we can compute the probability that a geodesic crosses a sequence of extension edges by calculating the probability concentrated on a contiguous interval.

- Any geodesic in $\Gamma^g(q_0)$ is a prefix of some other geodesic in $\Gamma^g_b(q_0)$. With this property, navigation along any path $\gamma \in \Gamma(q_0)$ can be accomplished by carrying out the navigation along the path $\gamma_b \in \Gamma^g_b(q_0)$ with prefix γ, and then by stopping the navigation when the desired endpoint is reached.

- A human user might easily determine the cyclic ordering of two geodesics in $\Gamma^g_b(q_0)$. Recall that, we induced a cyclic ordering between the geodesics in $\Gamma^g_b(q_0)$ using the mapping ψ_b in Section 6.3.3.

8.2.2 Optimal feedback policy for selecting local geodesics

We apply the optimal feedback policy described in Chapter 3 to allow a human user specify a local geodesic $\gamma^* \in \Gamma_b^g(q_0)$, or equivalently, the angle $\theta^* = \psi_b(\gamma^*)$, with vanishing error probability, using a sequence of noisy binary inputs. Although this policy was designed for transmission of a message point in the unit interval $[0, 1)$, we extend it here for transmission of a message point in the unit circle S^1, which is assumed to be distributed uniformly. We model the noisy input source as a BSC with crossover probability ϵ. At time step k, the input to this channel is $x_k \in \{0, 1\}$, where we associate $x_k = 0$ with the input "left" and $x_k = 1$ with the input "right". The output of the channel is $y_k \in \{0, 1\}$ with $P(Y_k \neq X_k) = \epsilon$, where Y_k, X_k are random variables corresponding to channel input and output, respectively. We assume that the BSC can provide noiseless feedback to the user, which in our case corresponds to providing an estimate $\widehat{\gamma}$ of the user's desired path γ^* to the user and assuming that the user can determine with perfect accuracy whether or not $\gamma^* < \widehat{\gamma}$ according to the ordering defined in Section 6.3.3.

The optimal feedback policy used by the interface is as follows. Assume that at time step k, the interface computed the posterior distribution $P_{\Theta|Y^k}(\theta|y_1 \dots y_k)$, where $\Theta \in S^1$ is the random variable indicating the desired angle, and Y^k is the random vector $(Y_1, \dots, Y_k)$. First, the interface finds a pair of angles $(\mu_k, \bar{\mu}_k)$ that are opposite to each other in S^1, i.e., $\bar{\mu}_k = (\mu_k + \pi)$ mod (2π), and that the probability concentrated on the half circle from μ_k to $\bar{\mu}_k$ is equal to the probability concentrated on the opposite half circle from $\bar{\mu}_k$ to μ_k. The interface selects the angle in $(\mu_k, \bar{\mu}_k)$ with higher posterior density as the estimate $\widehat{\theta}_k$, and provides it as feedback by showing the geodesic $\widehat{\gamma}_k = \psi_b^{-1}(\widehat{\theta}_k)$. Then, the user selects the next input x_{k+1} as 1 if $\gamma^* \geq \widehat{\gamma}_k$ or as 0 otherwise. Finally, if $y_{k+1} = 1$ (the case $y_{k+1} = 0$ is analogous), the interface applies Bayes' rule to update the posterior distribution as

$$
\begin{aligned}
&P_{\Theta|Y^{k+1}}\left(\theta \middle| y_1 \dots y_{k+1}\right) = \\
&\eta \cdot \begin{cases} (1-\epsilon) \cdot P_{\Theta|Y^k}\left(\theta \middle| y_1 \dots y_k\right) & \text{if } \psi_b^{-1}(\theta) \geq \widehat{\gamma}_k \\ \epsilon \cdot P_{\Theta|Y^k}\left(\theta \middle| y_1 \dots y_k\right) & \text{otherwise,} \end{cases}
\end{aligned}
\tag{8.1}
$$

where η is a normalizing constant. Then, the process repeats.

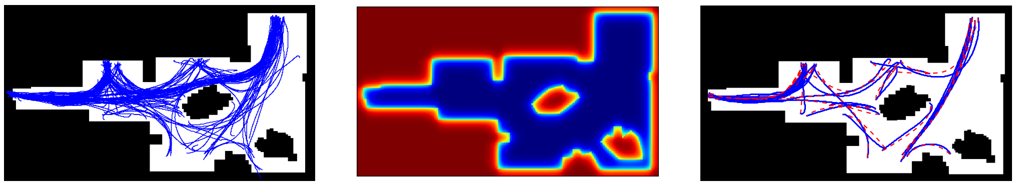

(a) Intel pedestrian path prediction dataset [128]
(b) The recovered cost function
(c) Test paths and corresponding geodesics

Figure 8.2: The dataset consisting of human walking paths is shown on a 525 by 325 map in (a), where black pixels denote obstacles, and white pixels denote free space. The recovered cost function is shown in (b), where red pixels denote a high cost and blue pixels denote a low cost. The paths in the test dataset (dashed blue curves) and the geodesics (solid blue curves) generated with the recovered cost function for each test path is shown in (c).

8.3 Interface for Navigating a Simulated Robot with EEG

8.3.1 Our approach for navigation with binary inputs

In this section, we describe our algorithm for enabling the navigation of the robot while the user is specifying their desired path using the method described in Section 8.2. Let $q_0 \in Q_{\text{free}}$ be the robot's starting position, and $\widehat{Q}(q_0)$ be the universal covering space consisting of all path-homotopy classes with a fixed starting point q_0. At time step k, the interface obtains the noisy input y_k from the user and computes the posterior distribution $P_{\Theta|Y^k}(\theta|y_1, \ldots, y_k)$. From this distribution, it generates the estimate $\widehat{\theta}_k \in S^1$ and the geodesic $\widehat{\gamma}_k = \psi_b^{-1}(\widehat{\theta}_k)$, which starts at q_0 and ends at a boundary point. The estimate $\widehat{\theta}_k$ also describes a path-homotopy class with an endpoint q_f in $\widehat{Q}(q_0)$. Instead of displaying $\widehat{\gamma}_k$ as feedback, the interface displays the geodesic $\widehat{\pi}_k$ from the robot's current position $q \in \widehat{Q}(q_0)$ to $q_f \in \widehat{Q}(q_0)$. Let e be the first extension or boundary edge that $\widehat{\pi}$ crosses in $\widehat{Q}(q_0)$ (see Section 8.2). The robot moves along $\widehat{\pi}$ until it crosses e when the posterior probability of the event that the user's desired path crosses e is at least 0.90. This procedure is outlined in Algorithm 4.

Algorithm 4 Robot Navigation Algorithm.

```
1: let $\widehat{Q}(q_0)$ be the universal covering space
2: while robot is not at a boundary point do
3:     wait until noisy input $y_k$ is observed
4:     evaluate $P_{\Theta|Y^k}$ and obtain $\widehat{\theta}_k \in S^1, \widehat{\gamma}_k \in \Gamma_b^g(q_0)$
5:     let $q$ be the robot's current position in $\widehat{Q}(q_0)$
6:     let $q_f$ be the point in $\widehat{Q}(q_0)$ corresponding to $\widehat{\theta}_k$
7:     display geodesic $\widehat{\pi}_k$ from $q$ to $q_f$ in $\widehat{Q}(q_0)$ as feedback
8:     let $e$ be the first edge that $\widehat{\pi}_k$ crosses in $\widehat{Q}(q_0)$
9:     if posterior probability of crossing $e \geq 0.90$ then
10:        command robot to "move along" $\widehat{\pi}_k$ until it crosses $e$
11:    else
12:        command robot to "stop"
13:    end if
14: end while
```

8.3.2 Our approach for obtaining binary inputs through EEG

Our BMI was based on steady-state visually-evoked potentials (SSVEP), a natural neural response to repetitive flickering stimuli in the environment [130]. By providing stimuli of known frequency patterns, it is possible to determine which stimulus the user is attending to. Our interface displayed two stimuli that steadily flashed on a CRT monitor at 8.67Hz and 12Hz. These frequencies were chosen because they yield high signal-to-noise ratio and lie outside of the range known to induce seizures [131]. EEG signals were extracted from seven electrode sites across the occipital region of the scalp, in particular PO7, PO3, PO4, PO8, O1, OZ, O2, at impedances not exceeding 10kΩ, with a reference measured at PZA [132]. These signals were acquired using a 128-channel bioamplifier at 256Hz, bandpass-filtered from 1Hz to 30Hz, and analyzed with BCI2000 [133] and MATLAB using a three second sliding window. The signals from each channel were filtered into four different spatial representations using bipolar and Laplacian configurations [134]. For each spatial filter, we computed a signal to noise ratio (SNR) of each frequency of interest together with its second harmonic. For each frequency, after discarding the highest and lowest SNR values, the average of the remaining two SNR values was obtained. A classification was made when this average exceeded a threshold of 8. After each classification, auditory feedback was provided to the user by playing a unique sound to indicate observation of a “left” or “right” input.

8.4 Experiments

8.4.1 Learning cost function from data

We performed experiments to learn the cost function in (6.1) using the Intel pedestrian path prediction dataset [128] that consists of human walking paths recorded in an office environment (Fig. 8.2a). Similar to the procedure in [128], a black and white pixel map of the environment was generated, with a pixel corresponding to a distance of 0.04 meters. Ten cost features were defined for each pixel based on blurred images of the pixel map with different levels of blurring. The loss function, a measure of dissimilarity of a given path γ from an example path γ_i, was computed by first assigning a zero loss to all pixels that are within 7 pixels (0.28 meters) of distance from γ_i, and constant loss to all other pixels, and then by summing the loss values assigned to the pixels that γ crossed over.

We observed that some paths in the dataset do not resemble any optimal behavior that can be explained by our cost features. We labeled such paths as outliers and removed them from the dataset. In order to detect whether a path γ_i was an outlier, we applied MMSL using γ_i as the only training example to obtain a cost function g_i, and computed the loss of the geodesic from the start point $\gamma_i(0)$ to the end point $\gamma_i(1)$ under the cost function g_i. If this loss was non-zero, we labeled the path γ_i as an outlier. Out of 166 paths, 100 paths were found to be outliers, and the remaining paths were split into a training set and a test set, each containing 33 paths.

The recovered cost function with MMSL using all paths in the training set is illustrated in Fig. 8.2b. We empirically verified that geodesics generated with respect to this cost function in each path-homotopy class were unique. For evaluation, we computed the loss of the geodesics corresponding to each training and test path. In the training set, 45% of the geodesics had zero-loss and in the test set, 55% of the geodesics had zero-loss. The results (Fig. 8.2c) suggest that we can approximate human walking paths with zero-loss by a

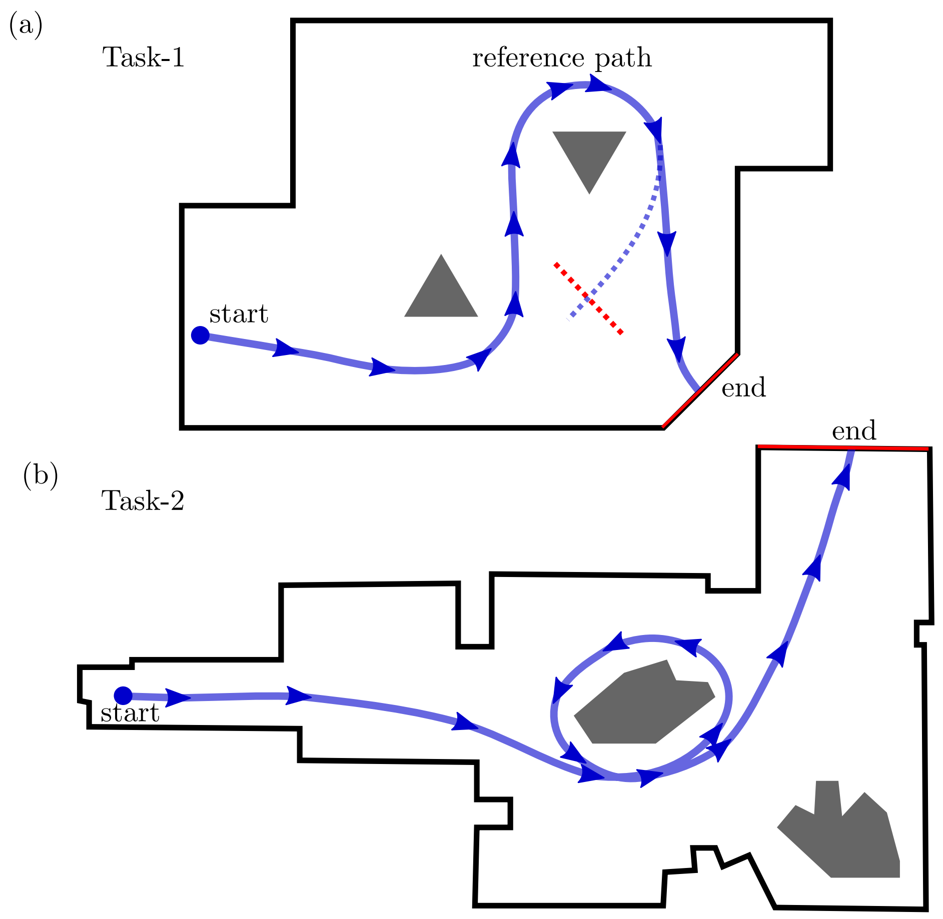


Figure 8.3: Description of the two experimental tasks. The goal was to navigate the robot from start (blue dot) to the midpoint of the finish segment (red boundary segment) by specifying a desired path homotopic to the reference path (blue curve). Task-1 was very similar to the task used in [53], where the reference path followed the dashed blue curve towards the end.

geodesic curve in about half of the cases, and the recovered cost function generalizes well to new cases.

8.4.2 Using the interface to navigate a simulated robot

In order to quantify the performance of our interface for navigating a simulated robot with EEG, we performed experiments with two able-bodied subjects that had no prior experience in SSVEP-based BMIs. Each subject completed two trials of two unique navigation tasks (Fig. 8.3). The goal in each task was to navigate the robot from its start position to a goal position

Table 8.1: Experiment Results for Task 1

	Subject A		Subject B	
	Trial 1	Trial 2	Trial 1	Trial 2
Task success	YES	YES	YES	YES
Time-to-navigate (s)	177	188	142	138
Time-opt-ratio	1.69	1.79	1.35	1.31
Time-to-specify (s)	143	102	117	99
Input accuracy	0.83	0.82	1.0	0.95
Input latency (s)	3.4	3.7	6.2	5.0
ITR (bits/min)	6.04	5.19	9.68	8.56

Table 8.2: Experiment Results for Task 2

	Subject A		Subject B	
	Trial 1	Trial 2	Trial 1	Trial 2
Task success	YES	NO	YES	YES
Time-to-navigate	253	-	257	314
Time-opt-ratio	1.62	-	1.65	2.01
Time-to-specify	138	-	165	264
Input accuracy	0.91	0.82	1.0	0.88
Input latency	4.3	8.1	6.9	8.0
ITR (bits/min)	7.86	2.37	8.70	3.53

in the boundary by specifying a desired path homotopic to a target reference path. In task-1, we simulated the environment used in [53] to evaluate their EEG-based BMI for navigating a physical wheelchair. Our reference path was very similar to theirs except that our path terminated in the boundary rather than in the free space. In task-2, we simulated the environment in which the dataset of human walking paths used in Section 8.4.1 was produced.

We used the following metrics to evaluate the performance of our brain-machine interface:

- **Task success**: whether the robot successfully navigated along a path that was homotopic to the reference path.

- **Time-to-navigate**: the duration of the task in seconds, if the task was successful.

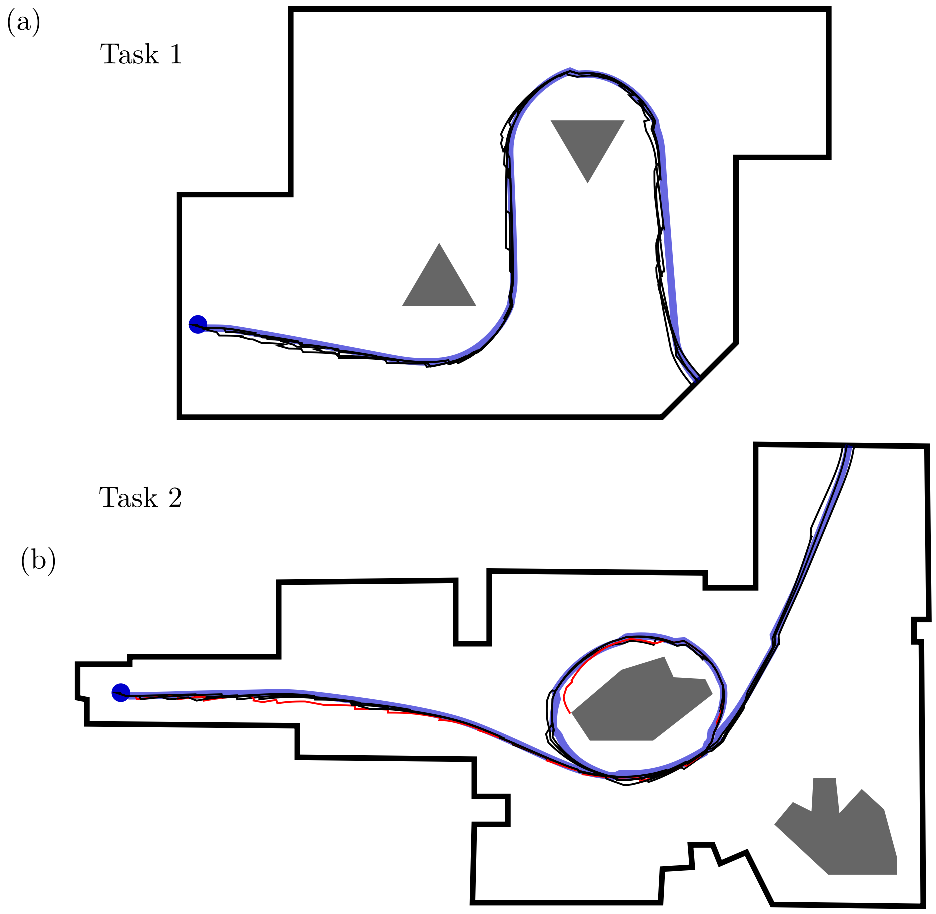


Figure 8.4: The actual paths (thin curves) followed by the mobile robot compared to the geodesic homotopic to the reference path (thick blue curve). The black curves correspond to the resulting paths from the successful trials. The red curve corresponds to the resulting path obtained in the failed trial.

- **Time-opt-ratio**: the ratio of time-to-navigate to the time it would take for the robot to continuously move along the geodesic homotopic to the reference path, which was 105 seconds for task-1, and 156 seconds for task-2.

- **Time-to-specify**: the time it took for the posterior probability of choosing a geodesic with an end point in the finish segment to exceed 90% probability.

- **Input accuracy**: the fraction of inputs that were correctly classified before time-to-specify.

- **Input latency**: the average time in seconds between two consecutive inputs.

- **ITR**: the information transfer rate that gives the number of reliable bits obtained from the user per minute.

Results show that our interface enabled subjects to successfully complete the navigation tasks in 7 of the 8 trials. The actual paths followed by the mobile robot are shown in Fig. 8.4, and the performances obtained in task-1 and task-2 are reported in Table 8.1, and 8.2, respectively. Remarkably, in all successful trials, the actual paths were very close to the geodesics homotopic to the reference paths. Subjects obtained between 5 and 10 bits/minute ITR in all trials except in trial-2 of task-2, where both subjects reported being tired and unable to focus their attention during the entire time. This might explain why subject A failed in trial-2 of task-2.

In task-1, the average time-to-navigate was 2.7 minutes, which is significantly less than the average time-to-navigate of 9.5 minutes reported in [53]. However, the fact that they used a different paradigm (P300) for obtaining inputs from EEG than ours (SSVEP), the fact that they used a physical wheelchair rather than a simulated mobile robot, and the fact that they did not rely on a given map of the environment prevents us from making a true comparison. For both tasks, time-opt-ratio was between 1.3 and 2.0 (time-opt-ratio of [53] was 5.4), which means that the navigation time with our interface was less than twice the time it would have taken the robot to move continuously along the geodesic homotopic to the reference path. It is important to note that the average gap between time-to-specify and time-to-navigate was 1.0 minutes, meaning that the subject specified a geodesic with an end point in the finish segment of the task about one minute before the robot actually moved to the midpoint of the finish segment. This gap was partly because between two consecutive user inputs, the robot was only allowed to move until it crossed an extension edge (see Algorithm 4). In future work, the robot might be allowed to move along a longer prefix of the specified geodesic and the speed of the robot might be adjusted to reduce this gap.

8.5 Conclusion

This study demonstrated an interface that allowed users to navigate a robot through a planar space containing obstacles using only the inputs from an SSVEP-based BMI. By representing desired paths as geodesics under a cost function recovered from human-demonstrated paths, we enabled users to navigate this space in a smooth human-like manner with only binary inputs. Our results suggest that not only can users navigate in this manner, but that they can do so with a very high success rate. Our subjects navigated the robot along two experimental paths in less than twice the time it would have taken the robot if informed of the path explicitly before the task. In the future, we would like to use a 3-class SSVEP-based BMI to enable the user to start or stop the robot at their will. Optimization of the robot's control behaviors and our signal classification algorithm may allow us to further improve the performance.

Chapter 9

Enabling Humans to Enter Text Commands with EEG

9.1 Introduction

This chapter presents an interface for text entry with discrete input commands obtained from EEG using steady-state visually evoked potentials (SSVEP). In existing SSVEP spellers, users specify a desired character by providing input commands in response to a sequence of queries, which are designed beforehand and selected at run time based on a fixed decision tree. Our approach is to use the active inference policy described in Chapter 4 to select queries adaptively based on a probabilistic language model, an estimate of how fast the interface can observe input commands, and an estimate of how accurate the input commands are. In particular, the next query is chosen to be the one that yields the highest information gain rate (IGR), which is defined as the expected amount of information to be received per unit of time from a query. We performed experiments to compare our interface against two state-of-the-art SSVEP spellers that used queries designed beforehand. Results show that our interface allows users to spell sentences twice as fast as they would with the compared spellers using the same input mechanism.

In the last decade, there has been a significant progress in the development of EEG-based brain-computer interfaces (BCIs) for text spelling [54]. Users of these interfaces specify a desired character using established paradigms such as those based on motor-imagery, P300, or steady-state visually-evoked potentials (Chapter 2). For example, in SSVEP-based BCIs, users allocate their attention to a visual target across a set of targets steadily-flashing at different frequencies. Users' allocation of their attention to a particular target generates an EEG activity that reflects the frequency of the attended target [135]. Effectively, this allows users to provide discrete input commands, where each input command corresponds to allocation of attention to a particular target. Although the operation of motor-imagery and P300 based BCIs differ from SSVEP-based BCIs, in principle, users provide discrete input commands by selecting targets.

The typical process of identifying the user's desired character from input commands is illustrated in Figure 9.1. First, the user is presented with a visual stimulus. We view this stimulus as a graphical representation of a "query", which is designed to extract information about the desired character. Each query defines a mapping from characters to targets, so that the user knows which target must be selected to communicate information about a particular character. For example, consider using an EEG paradigm with two targets, labeled 1 and 2, to specify an English letter from "A" to "Z". A query might ask users to select the target 1 to communicate that the desired character is a vowel, and to select the target 2 to communicate that the desired character is a consonant. Second, the user selects the target that is correlated with the desired character, as defined by the query posed with the visual stimulus. We refer to this target as the user response. In our example, the response would be 1 if the desired character is "E". Third, the user communicates the response using an EEG paradigm, and a classifier processes recorded EEG activity to predict the response. We refer to the predicted response as an input command, and view the translation of the user response to an input command as the communication of information over a discrete noisy channel. Fourth, the interface uses the input command to increase its information about the desired character. If the information obtained so far is not enough to identify the desired character, the interface designs a new query to extract more information. This query is posed with the presentation of a new stimulus, and the process repeats.

The queries designed by the interface play a significant role in determining how quickly and reliably the interface can identify the user's desired character. In particular, some queries are better than others, where "goodness" of a query depends on the following three models:

- A language model that determines the likelihood of a character given the characters spelled previously,
- An accuracy model that determines the conditional probability of an input command given a user response,

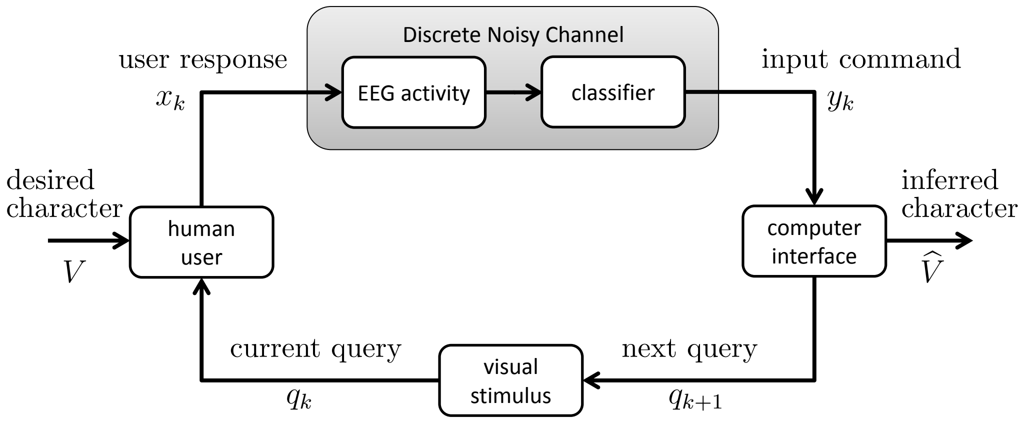


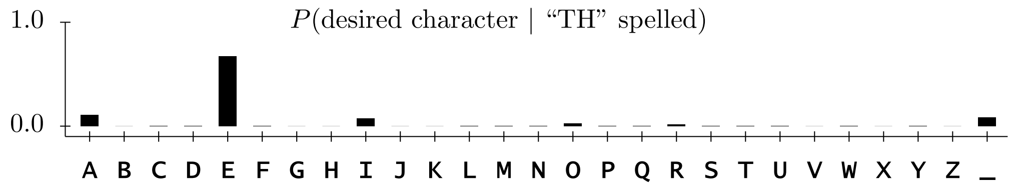
Figure 9.1: The components of an EEG-based brain-computer interface and how they interact with each other in closed loop to obtain inputs from the user.

- A latency model that determines the time it takes to obtain an input command given a user response.

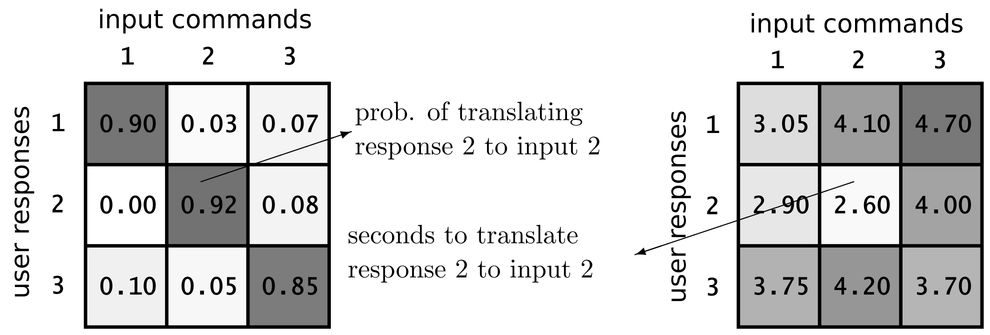
In order to illustrate these models and their importance in query design, let us consider spelling with an interface that has 27 characters consisting of letters from "A" to "Z", and the space character "_". Assume that the interface has three targets, labeled $1, 2, 3$, each of which can be selected as the response to a query. From now on, we use "response" to mean "user response", and use "input" to mean "input command".

To demonstrate the importance of the language model, consider the case where the user is trying to spell a word, and has already spelled "TH". In this case, the likelihoods of characters assigned by the language model, shown in Figure 9.2a, tell us that "E" is the most likely, and "A" is the second most likely character to be spelled after "TH". A query that allows users to specify "E" or "A" with a single response (see Figure 9.3b) is more valuable than a query that allows users to specify "X" or "Y" with a single response. This is because when the former query is used; the desired character will be identified using a single input with high probability. But, when the latter query is used; additional inputs will be required to identify the desired character with high probability.

To demonstrate the importance of the accuracy and latency models, consider the example models shown in Figure 9.2b, and Figure 9.2c, respectively. We represent these models as matrices, where the value at the x-th row and



(a) Language model



(b) Accuracy model

(c) Latency model

Figure 9.2: The example models that affect the value of query. Frame (a) shows the likelihood of characters after spelling "TH"; (b) shows the accuracy model as a matrix, where higher values (shown darker) mean higher conditional probability of observing the corresponding input given the corresponding response; (c) shows the latency model as a matrix, where lower values (shown brighter) mean a shorter time of classification in seconds.

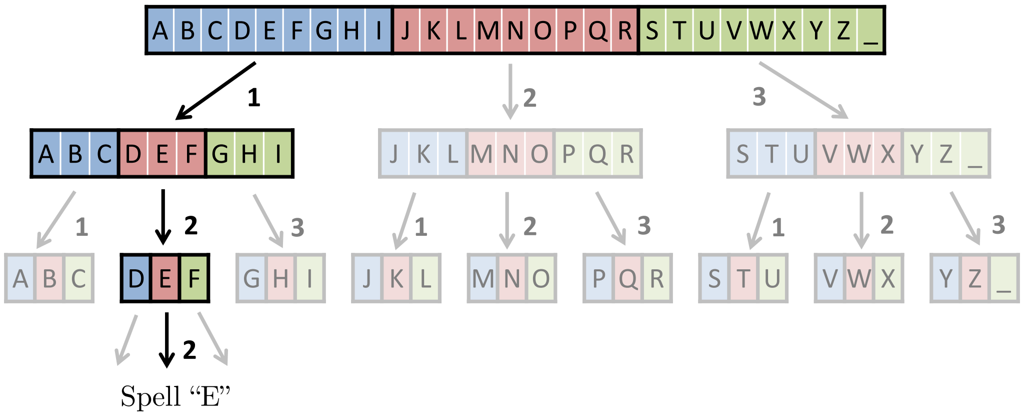
y-th column of the accuracy matrix, denoted $A(x, y)$, corresponds to the conditional probability of observing y when the response is x, and the value at the x-th row and y-th column of the latency matrix, denoted $L(x, y)$ corresponds to the time it takes to observe y when the response is x. One observation that can be made from the example models is that observing $y = 2$ when $x = 2$ is more accurate and faster than observing $y = 3$ when $x = 3$, i.e., $A(2, 2) > A(3, 3)$ and $L(2, 2) < L(3, 3)$. Using this observation, we argue that in the case of spelling after "TH", a query that asks users to provide $x = 2$ to specify "E" is more valuable than a query that asks users to provide $x = 3$ to specify "E". This is because associating characters that are more likely with responses that can be communicated more accurately and more quickly might reduce both the probability of spelling an incorrect character, and the expected time to spell a character.

Most existing EEG-based spellers do not take into account any of these models for selecting queries. In these spellers, queries are often designed a priori at the time of design and selected at run time based on a decision tree and the inputs obtained so far. We refer such queries as "static" queries. As an example, Fig. 9.3a demonstrates how users might specify the letter "E" using a speller with static queries. In this speller, the user can specify a character by responding to three successive queries, where each response indicates the character group that contains the desired character. The number of possibilities for the desired character is reduced from 27 to 9 after the first query, from 9 to 3 after the second query, and from 3 to 1 after the third query. In our example, the interface identifies "E" after observing the inputs $1, 2, 2$. The same sequence of inputs is used to identify "E" regardless of what has been spelled before "E" or how fast and accurate each input can be obtained.

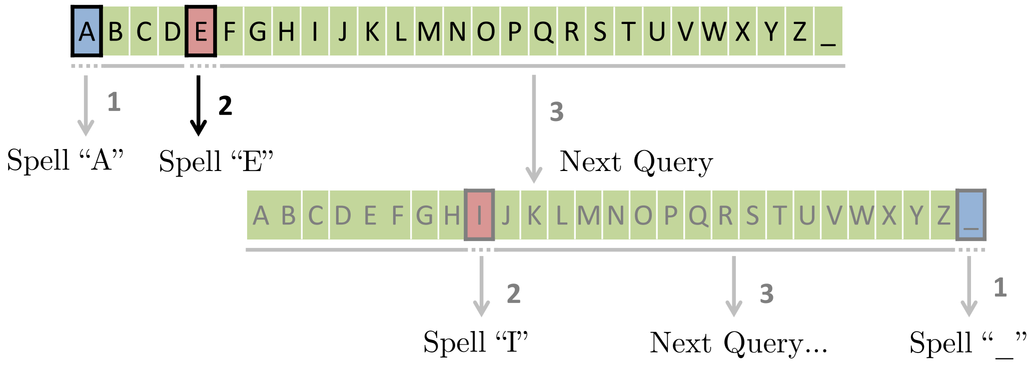
Our goal is to develop an EEG-based text-entry interface that adaptively queries desired characters by taking into account a probabilistic language model, an accuracy model, and a latency model, as illustrated in Figure 9.2. As an example, Figure9.3b shows an adaptive query posed after spelling "TH". In this query, the user is asked to provide 1, 2, or 3 to indicate whether the desired character is "A", "E", or any other character, respectively. The user provides $x = 2$ to specify "E", and the interface spells "E" when $y = 2$ is observed. Based on the example models provided, this query assigns the most likely character to the input that can be obtained faster and more accurate than other inputs. Although adaptive queries might give better spelling performance than static queries, they may cause the visual stimuli to be difficult to understand. Therefore, adaptive queries and associated visual stimuli must be carefully designed to ensure ease of use.

In order to select adaptive queries effectively, our approach is to choose the next query to be the one that maximizes the information gain rate (IGR), as defined in Chapter 4. IGR measures the expected amount of information received per unit of time, and is a more general version of the information transfer rate (ITR), which is commonly used in the BCI literature as a performance measure [18, 19, 136]. The assumptions used to calculate ITR, such as assuming equal likelihood of responses, make it inappropriate as a measure for selecting queries. Unlike ITR, IGR depends on the likelihoods of responses to a query, and the characteristics of the input channel as defined by the accuracy and latency models. The likelihoods of responses can be computed from the posterior likelihoods of characters given by the language model and the inputs obtained to the queries posed so far. To ensure ease of use, we restrict the set of possible queries from which the next query is chosen to queries for which we assume that users can determine the respond quickly and accurately.

To our knowledge, adaptive queries have not been considered in SSVEP-based spellers before, e.g., see [75, 76]. Although adaptive queries have been considered in spellers based on P300 and motor-imagery brain potentials, the models they assumed and the approach they took are quite different than ours. The Hex-o-Spell interface by Blankertz et al. [73] and the interface by D'Albis et al. [74] allowed users to specify a character with motor-imagery tasks. These interfaces used a probabilistic langauge model to change parts of a pre-determined decision tree based on the characters specified so far. Unlike ours, they did not take into account either an accuracy or a latency model. In P300-based interfaces, the number of targets the user can choose is often the same as the number of characters. One exception is the interface by Ma et al. [67], where sets of available targets were adaptively chosen with respect to a probabilistic language model and a latency model that scales linearly with the number of available targets. Their approach was to minimize the expected time to spell a character by constructing an optimal decision tree to identify the next character in the current context. However, their approach did not take into account an accuracy model. An intermediate



(a) An example speller with static queries



(b) An example speller with adaptive queries

Figure 9.3: Illustrating the process of spelling "E" after "TH" using static queries that are chosen a priori at the design time, see (a), and using adaptive queries that are chosen at run time based on the example models shown in Fig. 9.2, see (b). Users provide the response 1, 2, or 3, to indicate that their desired character is in the group colored with blue, red, or green, respectively. "E" is spelled after 3 static queries in (a), and after a single adaptive query in (b).

step in the design on P300 spellers is to decide which sets of characters are illuminated in which order. For example, in the original P300 speller [32, 33], the characters are placed into a 6x6 matrix, and rows and columns of this matrix are illuminated in a random order. The interface by Park et al. [66], determines the order of illumination of rows or columns of the matrix adaptively based on a probabilistic language model and the online confidence of the P300 detection algorithm produced after the previous illuminations. However, their approach may not be used to select queries from a large pool of queries due to its computational complexity.

As a case study in the application of our approach, we present an SSVEP-based speller with adaptive queries. Section 9.2 describes our method of identifying a desired character by adaptively selecting queries to maximize information gain rates. Section 9.3 describes the implementation of our speller, and two state-of-the-art SSVEP spellers with static queries, which are used for performance comparison. We performed experiments to evaluate our speller, and the performance gained by the use of adaptive queries in contrast to the spellers implemented for comparison (Section 9.4). Results show that our interface allows users to spell sentences twice as fast as they would with the compared spellers using the same input mechanism.

9.2 Method for Identifying a Desired Character

In this section, we apply our active inference policy (Chapter 4) that adaptively selects queries to maximize information gain rates for identifying the user's desired character. Recall that, Chapter 4 represented human intent as a random variable V taking values from a set $\mathcal{V}$. Here, human intent is a desired character chosen from a given alphabet of characters. In order to avoid confusion between the random variable V and the character "V", we use $\Sigma = \{\sigma_1, \ldots, \sigma_M\}$ to denote the alphabet, and $\Psi \in \Sigma$ to denote the random variable specifying the desired character.

In Chapter 4, we modeled input mechanisms as discrete noisy channels with latency. Here, we consider only a single input mechanism with which users respond to queries by choosing a target from a finite set of targets denoted $\mathcal{X}$. The chosen target, denoted as a random variable $X \in \mathcal{X}$, is observed by the interface as an input command $Y \in \mathcal{Y}$, where $\mathcal{Y} = \mathcal{X}$. The

Algorithm 5 The process of identifying a single character

```
 1: $\mathcal{Q} \leftarrow$ query pool
 2: $A \leftarrow$ accuracy matrix
 3: $L \leftarrow$ latency matrix
 4: $s \leftarrow$ characters spelled in the past
 5: $P_\Psi \leftarrow$ likelihood of characters after $s$, given by the lang. model
 6: loop
 7:     $q \leftarrow SelectQuery(\mathcal{Q}, P_\Psi, A, L)$
 8:     POSE $q$ by updating visual stimulus
 9:     WAIT until an input command is observed
10:     $y \leftarrow$ observed input command
11:     $P_\Psi \leftarrow UpdateBelief(P_\Psi, A, q, y)$
12:     if $IsStoppingCriteriaMet(P_\Psi, q, y)$ then
13:         SPELL $\arg\max_{\sigma \in f_q^{-1}(y)} P_\Psi(\sigma)$
14:     end if
15: end loop
```

accuracy model A is given by the probability transition matrix $P_{Y|X}$ of the input mechanism, i.e., $A(x, y) = P_{Y|X}(y|x), \forall x \in \mathcal{X}, \forall y \in \mathcal{Y}$. Recall that $A(x, y)$ is the probability that the input $Y = y$ will be observed when the response is $X = x$. The latency matrix of the input mechanism $L_{X,Y}$ is denoted as L, i.e., $L(x, y) = L_{X,Y}(x, y)$. Recall that $L(x, y)$ is the time it takes to observe $Y = y$ after the presentation of a query, when the response is $X = x$.

We compute the likelihood of characters in a spelling context using a probabilistic language model. The spelling context, denoted $s \in \Sigma^*$, is defined as the sequence of characters spelled so far. We use the distribution P_Ψ to represent the likelihood of characters in the current spelling context, and after the responses to queries posed in this context. The queries are selected from a query pool, denoted $\mathcal{Q}$. Associated with each query $q \in \mathcal{Q}$ is a function $f_q : \Sigma \to \mathcal{X}$ that defines the response the user must provide to communicate their desired character. The preimage $f_q^{-1} : \mathcal{X} \to \Sigma^*$ for a response $x \in \mathcal{X}$ is defined as $\{\sigma \in \Sigma | f_q(\sigma) = x\}$. Intuitively, $f_q^{-1}(x)$ is the set of characters in Σ that are mapped to x by the function f_q.

The process used to specify a desired character is outlined in Algorithm 5. First, the interface is given the query pool, the accuracy matrix, the latency matrix, and the probabilistic language model. The approach for constructing these components in our interface for text entry will be described in Sec-

Algorithm 6 *SelectQuery*$(\mathcal{Q}, P_\Psi, A, L)$

```
1: maxScore ← 0
2: bestQuery ← None
3: for $q \in \mathcal{Q}$ do
4:     $P_X = [\dots]$
5:     for $x \in \mathcal{X}$ do
6:         $P_X(x) = \sum_{\sigma \in f_q^{-1}(x)} P_\Psi(\sigma)$
7:     end for
8:     score = $IGR(P_X, A, L)$
9:     if score > maxScore then
10:        score ← maxScore
11:        bestQuery ← $q$
12:    end if
13: end for
14: return bestQuery
```

tion 9.3. The interface's belief of the desired character P_Ψ is initialized by the probabilistic language model taking into account the spelling context s. Second, the interface selects the query from $\mathcal{Q}$ that maximizes the information gain rate (Chapter 4), i.e.,

$$\arg\max_{q\in\mathcal{Q}} IGR(P_X^{(q)}, A, L), \tag{9.1}$$

where $P_X^{(q)}$ is the likelihood of responses to the query q, computed from

$$P_X^{(q)}(x) = \sum_{\sigma \in f_q^{-1}(x)} P_\Psi(\sigma), \qquad \forall x \in \mathcal{X}. \tag{9.2}$$

Algorithm 6 describes this selection process. Then, the selected query is posed to the human user by updating the visual stimulus. Third, the interface updates its belief P_Ψ over the desired character Ψ using the input command y observed as a noisy response to the query. Algorithm 7 describes this belief update. Intuitively, this update scales the posterior probability of a character σ by the probability that y will be observed when the user response is $f_q(\sigma) \in \mathcal{X}$. As a result, this increases the posterior probability of all characters in the preimage of y by $A(y, y)$. Fourth, the interface tests whether it received enough information to identify the user's desired character using the stopping criteria described in Algorithm 8. If the algorithm cannot identify the character, the process continues with the selection of a

Algorithm 7 *UpdateBelief*(P_Ψ, A, q, y)

```
1: for $\sigma \in \Sigma$ do
2:     $P_\Psi(\sigma) = A(f_q(\sigma), y)P_\Psi(\sigma)$
3: end for
4: NORMALIZE $P_\Psi$
5: return  $P_\Psi$
```

Algorithm 8 *IsStoppingCriteriaMet*(P_Ψ, q, y)

```
1: $\alpha, \beta \leftarrow$ thresholds chosen empirically
2: $\psi_1 \leftarrow \arg\max_{\sigma \in f_q^{-1}(y)} P_\Psi(\sigma)$
3: $\psi_2 \leftarrow \arg\max_{\sigma \in \Sigma - f_q^{-1}(y)} P_\Psi(\sigma)$
4: if $|f_q^{-1}(y)| > 1$ then
5:     $\psi_3 \leftarrow \arg\max_{\sigma \in f_q^{-1}(y) - \{\psi_1\}} P_\Psi(\sigma)$
6:     return  $P_\Psi(\psi_1) > \alpha P_\Psi(\psi_2)$ and $P_\Psi(\psi_1) > \beta P_\Psi(\psi_3)$
7: else
8:     return  $P_\Psi(\psi_1) > \alpha P_\Psi(\psi_2)$
9: end if
```

new query and lasts till the stopping criteria are met.

The stopping criteria described in Algorithm 8 depend on the likelihood of characters given by the current belief P_Ψ, and the preimage $f_q^{-1}(y)$ which consists of the characters correlated with the input command by the mapping used in the last query. Let ψ_1 be the most likely character in the preimage, and ψ_2 be the most likely character outside of the preimage. If the preimage contains a single character, the algorithm tests whether the likelihood ratio $\frac{P_\Psi(\psi_1)}{P_\Psi(\psi_2)}$ is greater than a certain threshold α. If the preimage contains multiple characters, in addition to this test, the algorithm checks if the likelihood ratio $\frac{P_\Psi(\psi_1)}{P_\Psi(\psi_3)}$ is greater than a certain threshold β, where ψ_3 is the second-most likely character in the preimage. The intuition behind this heuristic stopping criteria is that if the observation of the user's response points to multiple characters we want the spelling to occur with (roughly) more evidence (assuming $\beta > \alpha$) than if the observation of the user's response points to a single character.

9.3 Interface for Entering Text Commands

Our interface allowed users to specify a character from an alphabet Σ that consisted of the 26 English letters, the space character denoted "_", and the backspace character denoted "<". We defined an ordering between the characters in Σ by assigning "<" as the first character, the characters "A-Z" as the second to twenty-seventh character, and "_" as the twenty-eight character.

9.3.1 Obtaining input commands through EEG

The interface obtained discrete input commands using the SSVEP paradigm. Five steady-state targets flickering at frequencies $f_1 = 7.50, f_2 = 10.0, f_3 = 6.67, f_4 = 12.0, f_5 = 8.57$, from left to right respectively, were presented on an LCD monitor (Figure 9.5). The user responded to queries by allocating their attention to one of the five targets. In particular, the set of responses and inputs were $\mathcal{X} = \mathcal{Y} = \{1, 2, 3, 4, 5\}$, where the index i corresponded to the target flickering at the frequency f_i. EEG signals were extracted from seven electrode sites across the occipital region of the scalp, in particular PO7, PO3, PO4, PO8, O1, OZ, O2, at impedances not exceeding 10kΩ, with a reference measured at PZA [132]. These signals were acquired using a 128-channel bioamplifier at 256Hz, bandpass-filtered from 1Hz to 30Hz, and analyzed using a 1.5 seconds sliding window. The signals from each channel were filtered into four different spatial representations using bipolar and Laplacian configurations [134]. For each spatial filter, we computed a signal to noise ratio (SNR) of each frequency of interest together with its second harmonic. For each frequency, after discarding the highest and lowest SNR values, the average of the remaining two SNR values was obtained. A classification was made when this average exceeded a pre-determined threshold.

9.3.2 Constructing the accuracy and latency models

We constructed the accuracy and latency models by having users perform a set of training trials before their use of the text entry interface. In these trials, we asked users to attend to a particular target $X \in \mathcal{X}$ chosen randomly.

After each trial, we recorded the observed input command $Y \in \mathcal{Y}$, and the latency given by the elapsed time since the onset of the prompt for selecting X. To construct the accuracy matrix A, we estimated $A(x, y), \forall x \in \mathcal{X}, \forall y \in \mathcal{Y}$, as the fraction of the number of trials in which $Y = y$ observed when $X = x$ to the number of trials in which $X = x$. We smoothed A by adding 0.01 to all $A(x, y)$ for which $x \neq y$, and then by scaling appropriately to get a probability measure. To construct the latency matrix L, we estimate $L(x, y), \forall x \in \mathcal{X}, \forall y \in \mathcal{Y}$ as the mean latency across the latencies of all trials in which $Y = y$ observed when $X = x$. We filled in the empty $L(x, y)$ values, i.e., (x,y) values for which y was not observed when the response was x, with the mean latency across all trials.

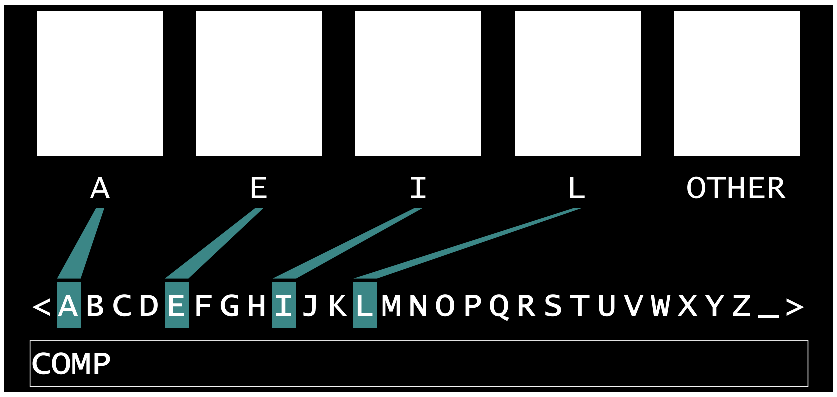
9.3.3 Constructing the probabilistic language model

We constructed a probabilistic language model using prediction by partial matching (PPM) [114]—a method based on variable-order Markov models. We trained PPM using the English text provided with the Dasher text-entry interface [137] with context lengths of up to 5 characters. Given the spelling context $s \in \Sigma^*$, this model provided the conditional probabilities $P_\Psi(\Psi|s)$, where $P_\Psi(\Psi = \sigma|s)$ is the probability that the desired character is $\sigma \in \Sigma$ after spelling s. Figure 9.4c illustrates the conditional probabilities obtained with our probabilistic language model after spelling "COMP".

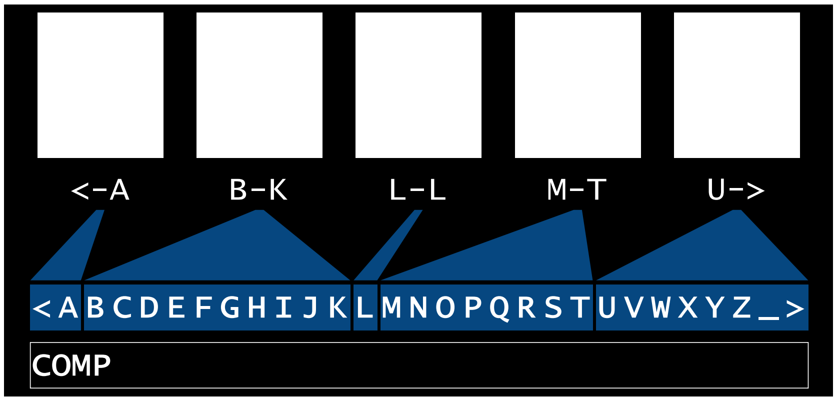
9.3.4 Constructing the query pool

The interface poses queries to extract information about the user's desired character. In principle, each surjective function $f : \Sigma \to \mathcal{X}$ can represent a query. We require f to be surjective (onto) because observing an input command y that has an empty preimage $f^{-1}(y)$ does not provide any information about the desired character.

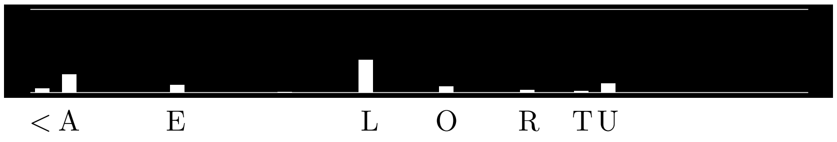
In our interface, we only considered a subset of all possible queries that could be constructed by surjective functions. The reason behind this was to restrict our queries to ones for which we believed the associated stimuli would allow humans to understand the query with ease and would allow them to provide their responses without making mistakes. In particular, our query



(a) A char-select query posed after spelling “COMP”



(b) A range-select query posed after spelling “COMP”



(c) Conditional probabilities after spelling “COMP”

Figure 9.4: Illustration of the queries used by our interface.

pool $\mathcal{Q}$ consisted of two types of queries called *char-select* and *range-select*.

In a char-select query q, the interface assigns four characters from Σ to responses $1, 2, 3, 4$ with respect to their order in Σ, and assigns the remaining characters to the response 5. This corresponds to the mapping

$$f_q(\Psi) = \begin{cases} 1 & \text{if } \; \Psi = \psi_1 \\ 2 & \text{if } \; \Psi = \psi_2 \\ 3 & \text{if } \; \Psi = \psi_3 \\ 4 & \text{if } \; \Psi = \psi_4 \\ 5 & \text{if } \; \Psi \in \Sigma - \{\psi_1, \psi_2, \psi_3, \psi_4\}, \end{cases} \tag{9.3}$$

where $\psi_1 < \psi_2 < \psi_3 < \psi_4$ and $\psi_1, \psi_2, \psi_3, \psi_4 \in \Sigma$. Figure 9.4a shows the stimulus used by our interface to pose the char-select query with the maximum IGR after spelling "COMP". For instance if the desired character is "L", the user must provide $X = 4$ as the response to this query.

In a range-select query q, the interface partitions the ordered set of characters into five contiguous non-empty groups (ranges), and assigns these groups to targets $1, 2, 3, 4, 5$ with respect to their order. This corresponds to the mapping

$$f_q(\Psi) = \begin{cases} 1 & \text{if } \Psi < \psi_1 \\ 2 & \text{if } \psi_1 \leq \Psi < \psi_2 \\ 3 & \text{if } \psi_2 \leq \Psi < \psi_3 \\ 4 & \text{if } \psi_3 \leq \Psi < \psi_4 \\ 5 & \text{if } \psi_4 \leq \Psi, \end{cases} \tag{9.4}$$

where $\psi_1 < \psi_2 < \psi_3 < \psi_4$ and $\psi_1, \psi_2, \psi_3, \psi_4 \in \Sigma$. Figure 9.4b shows the stimulus used by our interface to pose the range-select query with the maximum IGR after spelling "COMP". For instance if the desired character is "L", the user must provide $X = 3$ as the response to this query.

9.4 Experiments

9.4.1 The interfaces implemented for comparison

Bremen Interface

In our implementation of the Bremen interface [76], the user was presented with a grid containing the characters in Σ (Figure 9.5b). In order to specify a desired character, the user first navigated a cursor within the grid to the cell with their desired character by providing responses $1, 2, 4, 5$ to move left, up, down, and right in the grid, respectively. The cursor always started at the center cell with "E", and moved onto the corresponding adjacent cell after receiving a navigation command $1, 2, 4$ or 5. When the cursor was at the desired character, the user provided the response 3 to specify that character. The placement of characters in the grid was determined *a priori* based on the frequencies of characters in English texts [76]. For example, "B" was spelled

Table 9.1: Target texts and their negative log-likelihoods

Target	Text	NLL (bits/char)
T1	BCI	7.19
T2	BRAIN	3.04
T3	SIREN	5.31
T4	BRAIN_ COMPUTER_INTERFACE	1.98
T5	PLEASE_GET_ME_A_BLANKET	2.64

after receiving the navigation commands $5, 5, 4$ that moved the cursor right, right, and down, respectively, and after the select command 3 that triggered spelling.

Cecotti Interface

In our implementation of the Cecotti interface [75], the 3-level decision tree in Figure 9.3a was used. Initially, all possible characters were presented in three subgroups which consisted of nine characters. The user provided responses $1, 3, 5$ to choose the subgroup displayed at left, middle, or right, respectively. Following this initial group choice, the nine characters were again subdivided into three subgroups, each with three characters apiece. This process repeated one more time, allowing for the selection of the desired character. Additional responses allowed the user to return to the previous decision or move up the tree (by providing response 4), or delete the previous character (by providing response 2). For example, "B" was spelled after receiving the input commands $1, 1, 3$.

9.4.2 Participants and procedure

We performed experiments with 4 able-bodied participants that were between the ages of 20 and 30, and had normal or corrected-to-normal vision. In the beginning, participants completed a training phase consisting of 100 trials, where they were asked to attend to a randomly selected frequency from five frequencies. The purpose of the training phase was to estimate the accuracy and the latency matrix. After the training, the participants spelled the five target texts given in Table 9.1 using first our interface, then Bremen interface, and finally Cecotti interface.

In order to evaluate the performance of the SSVEP classification algorithm, we used the following measures

- overall accuracy (denoted $\bar{A}$): the fraction of trials where the observed input command matched the user response across all trials.
- overall latency (denoted $\bar{L}$): the average time it took to obtain an input command across all trials.
- information transfer rate (ITR): the average number of reliable bits received per minute using 5 targets, computed using

$$\bar{L}^{-1}\left(\log 5+\bar{A}\log\bar{A}+(1-\bar{A})\log\left((1-\bar{A})/4\right)\right). \tag{9.5}$$

In order to evaluate the performance of a text entry interface, we used the following measures:

- spelling rate (denoted R), the number of characters spelled per minute (cpm), without counting any backspaces.
- input-char rate(denoted ICR), the average number of trials used to spell one character.

9.4.3 Results

The spelling rate achieved by our interface in spelling a target text T depended upon the likelihood of T under our probabilistic language model. We computed the negative log-likelihood per character (denoted NLL) under our model for each target text, and reported the results in Table 9.1. NLL specified the average number of bits required to represent per character of a target text. NLL for target words (T1,T2,T3), ranged from 3.0 to 7.2, were higher than NLL for target sentences (T4,T5), ranged from 2.0 to 2.6. To account for this difference, we reported the results from spelling words (Table 9.3) separately than the results from spelling sentences (Table 9.4). We emphasize that NLL of English text was predicted to be about 2 bits per character [115, 138]. Therefore, the results obtained from the spelling of the

Table 9.2: Results obtained with our interface.

	Spelling time (sec)					ITR	R
	T1	T2	T3	T4	T5	(bits/min)	(cpm)
P1	27	30	70	131	124	34.2	8.6
P2	23	30	27	96	81	52.2	12.2
P3	78	29	34	118	103	39.6	9.4
P4	46	30	63	138	154	33.6	7.6

Table 9.3: Performance comparisons across target words (T1,T2,T3).

	$\bar{A}$	$\bar{L}$ (sec)	ICR	R (cpm)
Beremen interface	0.96	3.67	2.78	6.45
Cecotti interface	0.99	3.18	3.15	6.68
Our interface	0.97	3.49	2.81	7.53

target sentences were a better estimate of the actual performance that might be obtained with the long-term use of the interface.

In spelling target words, our adaptive interface provided a marginal improvement over the compared interfaces. In particular, the mean spelling rate with our interface (7.53 cpm) was about 15% better than the rate with Bremen and Cecotti interfaces. In spelling target sentences, the improvement in spelling rate jumped to over 100%, and participants achieved a mean spelling rate of 12.41 cpm. The difference in performance gain between spelling target words and sentences were due to the difference in their likelihoods under our language model. Individual performances using our interface varied between participants, as shown in Table 9.2, where they obtained ITRs ranging from 33.6 to 52.2 bits/min and spelling rates ranging from 7.6 to 12.2 cpm.

The classification performances observed across all three interfaces were similar, with overall accuracies ranging from 0.93 to 0.99, and with overall latencies ranging from 2.9 to 3.67 seconds. The overall latency with Cecotti interface was slightly shorter than the other interfaces. This might be because in Cecotti interface, users may determine the sequence of responses leading to the spelling of their desired character much faster than they may do in the other interfaces.

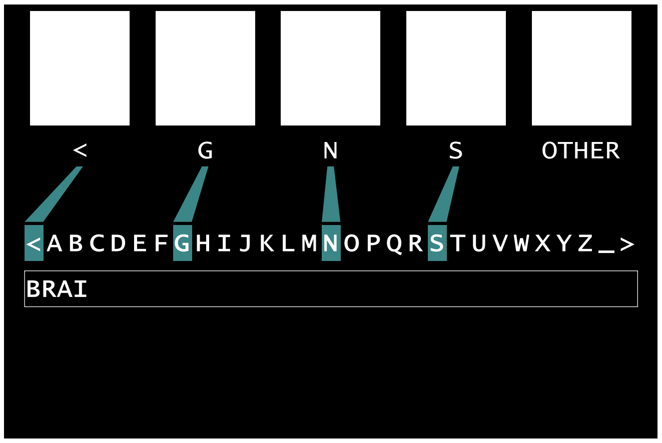
Table 9.4: Performance comparisons across target sentences (T4, T5).

	$\bar{A}$	$\bar{L}$ (sec)	ICR	R **(cpm)**
Bremen interface	0.97	3.20	3.25	**6.05**
Cecotti interface	0.93	2.90	3.67	**6.20**
Our interface	0.98	3.26	1.54	**12.41**

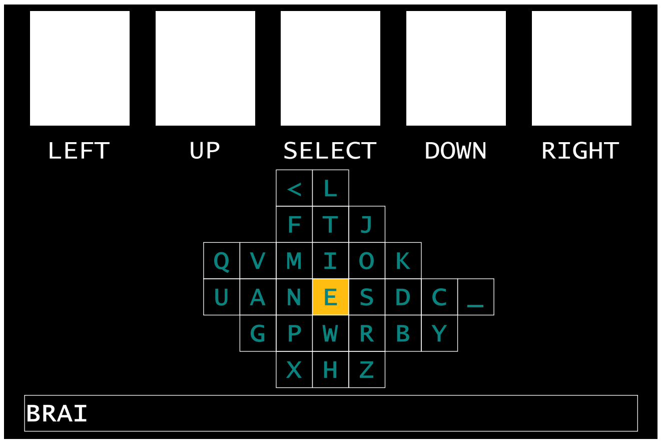
9.5 Conclusion

We presented an EEG-based interface that allowed users to spell English sentences by responding to a sequence of queries that were chosen adaptively to maximize information gain rates. Although we experimented with only four able-bodied participants, the results suggest that the performance of EEG-based text entry interfaces might be improved significantly with our approach. In particular, in our experiments, the spelling rates with our interface were twice the spelling rates obtained with our implementations of the two existing state-of-the-art SSVEP-based spellers.

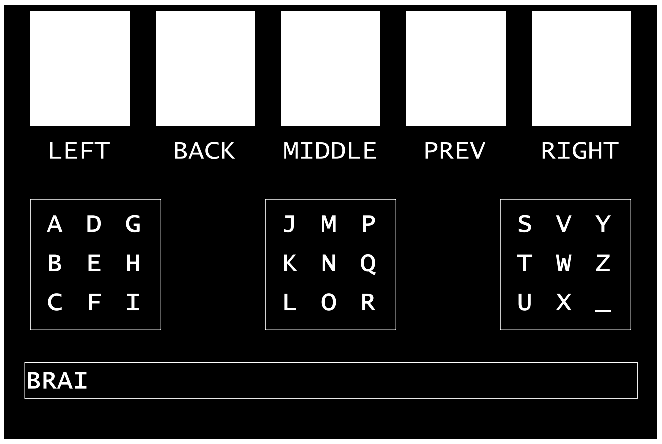
We emphasize that we used a relatively simple algorithm to decode EEG signals. Despite this limitation, one of our participants achieved an ITR of 52.2 bits/min, and spelled target sentences at a rate of 16 cpm. By using an advanced decoding algorithm such as the one in [76], spelling rates over 20 cpm might be achieved.



(a) Our Interface



(b) Bremen Interface



(c) Cecotti Interface

Figure 9.5: Sample screenshots of the interfaces evaluated in this study. Our SSVEP-based interface for text entry [shown in (a)], and the interfaces implemented for comparison: Bremen speller [shown in (b)] and Cecotti speller [shown in (c)]. The flickering objects are the white rectangles that appear on the top of each interface.

Part IV

CONCLUSION

Chapter 10

Discussion and Future Work

10.1 Discussion

In this thesis, we presented a framework for human control of robots with low-fidelity input mechanisms, such as the mechanisms used in EEG-based brain-machine interfaces, which produce input commands that are noisy, discrete and that have high latency. We considered that although the input mechanism is low-fidelity, there is often a high-fidelity feedback mechanism that the robot can use to provide stimuli to the human user. Our objective was to enhance the performance of the interfaces for human control of robots by having humans provide input commands in response queries that are posed by the robot to maximize information gain rates for quick and reliable estimation of the human intent. We computed information gain rate of a particular query as the expected amount of information to be received per unit of time about the human intent by posing that query.

10.1.1 Discussion of the optimal feedback policy

Our first approach to querying human intent was to leverage an optimal feedback communication policy (Chapter 3), which was designed for reliable transmission of a message point between two computational agents. We presented the use of this policy in the context of human control of robots where the agent transmitting the message is a human user, and the message represents the human intent. This policy assumed a particular structure on the human intent, specifically that it can be mapped to a message point distributed uniformly in a closed real interval, and that there is an intuitive ordering between all possible human intents. We showed that such a structure exists when we represent human intent as a desired path to be followed by a robot, based on the following two models. In the first model, desired paths were represented as strings of symbols chosen from an ordered alphabet, where each symbol identified a path primitive, such as a fixed-length circular arc with a particular curvature (Chapter 5). In the second model, desired paths were represented as local geodesics with respect to a cost function that took into account environment-driven features such as proximity to obstacles (Chapter 6). We demonstrated the feasibility of using the optimal feedback policy in robotic navigation tasks by querying the desired paths the humans want the robot to follow. In particular, we developed interfaces for flying a simulated aircraft and for navigating a mobile robot indoors with only noisy binary inputs obtained from EEG.

10.1.2 Discussion of the EEG-based interfaces for robotic navigation

The interface for flying a simulated aircraft represented desired paths as strings of circular arcs, which corresponded to paths of piecewise-constant curvature, and allowed users to navigate the aircraft successfully over these strings at a fixed speed and altitude (Chapter 7). Experimental results with this interface showed that our approach outperformed an existing state-of-the-art approach in navigating a robot moving at a fixed speed and enabled successful specification of desired paths despite very low information transfer rates obtained with binary motor imagery decoding from EEG signals.

The interface for navigating a mobile robot in a virtual indoor environment represented desired paths as local geodesics with respect to a cost function that was recovered from human-demonstrated paths using structured prediction (Chapter 8). Effectively, this interface allowed a human user to navigate a robot by specifying the topological structure of their desired path, and letting the robot determine the geometric structure using the cost function learnt from existing data. In experiments, the robot was navigated along two target paths successfully in less than twice the time it would have taken to robot if informed of the path explicitly before the task.

Prior to these two interfaces, the existing interfaces for human control of robots with EEG heuristically used inputs commands, which were inherently noisy and low-bandwidth, to identify moment-to-moment steering commands (i.e., in the interfaces based on process control such as [43]), waypoints (i.e., in the interface by Iturrate et al. [53]) or final destinations (i.e., in the interfaces based on goal selection such as [51]). In contrast, our interfaces used input commands as evidence to infer the user's desired path over a compact space of desired paths. The experimental results showed that our approach is not only optimal for reliable communication of intent but also generates a protocol that is easy for humans to implement for navigating robots.

10.1.3 Discussion of the active inference policy

Our second approach to querying human intent was to leverage the Bayesian active inference framework (Chapter 4). In particular, we presented the use of a (sequential) Bayesian experimental design, which suggested the selection of the experiment with the maximum information gain about the unknown parameter to be learnt, in the context of designing queries for inferring human intent, where each query specified the input command the human must provide to communicate information about their intent. To evaluate the value of a query, we introduced a measure we called information gain rate (IGR), which was defined as the ratio of the information gain to the expected latency of observing an input command after a query was posed. However, other measures could be used to evaluate the value of a query depending on the objectives of the interface design.

We also established a link between the active inference policy that maximizes information gain rates and the optimal feedback policy presented in Chapter 3. In particular, the queries selected by these two policies are the same if the human intent can be mapped to a particular structure (i.e., to a message point uniformly distributed over a closed real interval), the human responds to each query using the same input mechanism, and the expected time to obtain an input command after posing a query is the same for all queries. If these conditions are satisfied, the active inference policy reduces to the optimal feedback policy.

10.1.4 Discussion of the EEG-based interface for text-entry

We demonstrated the feasibility of using the active inference policy by developing an EEG-based interface for text entry (Chapter 9). Users of this interface specified desired texts by responding to a sequence of queries that were chosen adaptively to maximize information gain rates. To our knowledge, our interface was the first to design queries to make the input commands received from human users have the maximum value for identifying the user's desired character based on the characteristics of the input mechanism. The experimental results showed that our interface allowed human users to specify text twice as fast as they would with the compared state-of-the-art interfaces using the same input mechanism.

10.2 Future Work

10.2.1 Representing human intent as executions of MDP policies for robotic navigation

We modeled human intent as a desired path to be followed by the robot, and considered two representations: one represented paths as strings of symbols, and the other represented paths as local geodesics. The former representation allowed users to specify paths with a resolution that was (roughly) a function of the alphabet size, and the length of each symbol. The latter representation allowed users to specify only the topology of the paths (including the endpoint), where the geometric structure was provided by a prior model taking into account environment-driven features such as proximity to obstacles. In the future, to make desired paths more expressive and more compact by considering a hybrid of these two representations, we can represent a desired path as a path taken under a stochastic optimality policy of a deterministic Markov Decision Process (MDP).

An MDP is a 4-tuple (S, Σ, T, R), where S is the set of states, Σ is the set of actions, $T : S \times \Sigma \to S$ is the deterministic state-transition function, and $R : S \times \Sigma \to \mathbb{R}$ is the reward assigned to each state-action pair. A stochastic policy assigns a likelihood $\pi(\sigma|s)$ to each action $\sigma \in \Sigma$ taken in each state $s \in S$. A path γ is a sequence of state-action pairs visited under a policy, i.e., $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_N)$, $\gamma_i \in S \times \Sigma$, and the reward of a path is $J(\gamma) = \sum_{i=0}^{|\gamma|} R(\gamma_i)$. We view an action $\sigma \in \Sigma$ as describing the path generated by applying a motion primitive to the current state, similar to our representation of paths as compact strings (Chapter 5). For example, we can represent the paths generated using our alphabet of circular arcs (Section 5.2.1) as paths generated by execution of a stochastic MDP policy. In this MDP, the states are points on a 3D grid defined over the x, y positions and the orientation θ. Each state-symbol pair is assigned a reward, and each path is assigned a likelihood based on the accumulated rewards. Given a set of human-demonstrated paths, we can learn a stochastic optimality principle using the method of [139].

10.2.2 Adapting representations of human intent to the capacity of the input mechanism

In this thesis, we considered low-fidelity input mechanisms that have limited bandwidth such as the mechanisms used in EEG-based brain-machine interfaces. In our model of such input mechanisms as discrete noisy channels, the capacity of a channel provides a fundamental limit on the bandwidth of communication. It is important to express human intent with an uncertainty that admits reliable communication within a desired amount of time. We emphasize that the higher the uncertainty of the human intent, the more input commands will be required to communicate the human intent to the robot. In future work, we can use tools of rate-distortion theory [79] to adapt representations of human intent to enable reliable communication under the constraints of the input mechanism.

In our interface for flying a simulated aircraft, we considered adapting the representation of desired paths by changing the length of the symbols that the paths are composed of (Chapter 7). This enabled human users with higher-bandwidth input mechanisms to fly the aircraft along more expressive paths. However, our approach for adaptation was based on performing Monte Carlo simulations of the interface under a simulated input mechanism for a wide range of symbol lengths. In future work, we plan to use a more systematic approach.

One way to adapt representations of desired paths based on a symbolic language is to choose the symbols of the language, i.e., the alphabet Σ, systematically from a large pool of symbols $\mathcal{A}$. We can formalize this as the problem of selecting a subset Σ in A that maximizes a measure of expressivity of the paths generated using the symbols in Σ subject to the constraint that the reliable communication of paths of desired length can be achieved within a desired amount of time with the capabilities of the input mechanism. This problem might be addressed by using a submodular set function to measure the expressivity of paths, and by approximating the communication constraint using a modular function (e.g., the number of symbols in Σ must be less than a particular amount). The solution to this problem might be obtained using near-optimal greedy algorithms for submodular maximization [140].

10.2.3 Querying human intent without knowing the characteristics of the input mechanism beforehand

We restricted our scope to the setting where the robot knows the characteristics of the input mechanisms, such as how much accurately and quickly input commands can be provided using a particular mechanism (Section 1.2). In our interface for text-entry, we learnt an accuracy and latency model of the user's input mechanism by performing experimental trials in which the ground-truth user responses were known. This was a lengthy process and many trials needed to be performed to obtain a good estimate of the accuracy and latency models before allowing users to enter text. A similar process was used to estimate the crossover probability of input commands in our interface for flying a simulated aircraft with binary commands. Results with this interface showed that this crossover probability changed across successive runs of the experimental tasks and most of the failures with the interface could be explained by the change in the crossover probability.

In order to eliminate or shorten the process of estimating the characteristics of the input mechanisms, and to enable re-estimation of these characteristic during the use of the interface for inferring human intent, in future work, we plan to cast the problem of adaptively selecting queries as a *multi-armed bandit problem*. Formalized by Robbins [141], in the multi-arm bandit problem, pulling each arm yields a reward with a probability that is fixed but unknown to the puller and that is independent across each arm. The goal is to determine which arm to pull in which order in order to maximize the total reward. The solution strategies trade-off between exploration (estimating the fixed probabilities used by each arm) and exploitation (pulling the arm that is expected to give maximum expected reward).

References

[1] R. Leeb, D. Friedman, G. R. Müller-Putz, R. Scherer, M. Slater, and G. Pfurtscheller, "Self-paced (asynchronous) BCI control of a wheelchair in virtual environments: a case study with a tetraplegic," *Computational Intelligence and Neuroscience*, vol. 2007, pp. 7:1–7:12, April 2007. 2, 10, 17

[2] A. Akce, M. Johnson, and T. Bretl, "Remote teleoperation of an unmanned aircraft with a brain-machine interface: Theory and preliminary results," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 5322 –5327. 5, 6, 70

[3] A. Akce, J. Norton, and T. Bretl, "A brain-machine interface to navigate mobile robots along human-like paths amidst obstacles," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct., pp. 4084–4089. 5, 6, 7, 95

[4] A. Akce, M. Johnson, O. Dantsker, and T. Bretl, "A brain-machine interface to navigate a mobile robot in a planar workspace: Enabling humans to fly simulated aircraft with EEG," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 21, no. 2, pp. 306–318, March. 5, 6, 7, 67

[5] C. Omar, A. Akce, M. Johnson, T. Bretl, R. Ma, E. Maclin, M. McCormick, and T. P. Coleman, "A feedback information-theoretic approach to the design of brain-computer interfaces," *International Journal of Human-Computer Interaction*, vol. 27, no. 1, pp. 5–23, 2011. 5

[6] M. Horstein, "Sequential transmission using noiseless feedback," *Information Theory, IEEE Transactions on*, vol. 9, no. 3, pp. 136–143, 1963. 6, 25

[7] O. Shayevitz and M. Feder, "Optimal feedback communication via posterior matching," *Information Theory, IEEE Transactions on*, vol. 57, no. 3, pp. 1186–1222, March 2011. 6, 25, 28, 29

[8] D. V. Lindley, "On a measure of the information provided by an experiment," *The Annals of Mathematical Statistics*, vol. 27, no. 4, pp. pp. 986–1005, 1956. 6, 37, 39

[9] D. J. MacKay, "Information-based objective functions for active data selection," *Neural computation*, vol. 4, no. 4, pp. 590–604, 1992. 6, 37, 40

[10] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active Learning with Statistical Models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1996. 6, 37, 39

[11] A. Akce and T. Bretl, "A probabilistic language model for hand drawings," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, Aug. 2010, pp. 109 –112. 6, 54

[12] A. Akce and T. Bretl, "A compact representation of locally-shortest paths and its application to a human-robot interface," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May, pp. 2713–2718. 6

[13] N. G. Hatsopoulos and J. P. Donoghue, "The science of neural interface systems," *Annual Review of Neuroscience*, vol. 32, no. 1, 2009. 10

[14] J. R. Wolpaw, "Brain-computer interfaces as new brain output pathways," *The Journal of Physiology*, vol. 579, no. 3, pp. 613–619, 2007. 10, 15, 67

[15] D. Tan and A. Nijholt, Eds., *Brain-Computer Interfaces and Human-Computer Interaction.* Springer, 2010. 10, 11, 12

[16] "Spinal cord disability fact sheet," 2012, published online by United Spinal Association at http://unitedspinal.org/pdf/scdfactsheet.pdf. 10

[17] L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. A. Mukand, M. Saleh, A. H. Caplan, A. Branner, D. Chen, R. D. Penn, and J. P. Donoghue, "Neuronal ensemble control of prosthetic devices by a human with tetraplegia," *Nature*, vol. 442, pp. 164–171, July 2006. 10

[18] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain-computer interfaces for communication and control," *Clinical Neurophysiology*, vol. 113, no. 6, pp. 767–791, 2002. 10, 11, 112

[19] B. Z. Allison, S. Dunne, and R. Leeb, Eds., *Towards Practical Brain-Computer Interfaces: Bridging the Gap from Research to Real-World Applications.* Springer, 2012. 10, 112

[20] G. E. Fabiani, D. J. McFarland, J. R. Wolpaw, and G. Pfurtscheller, "Conversion of EEG activity into cursor movement by a brain-computer interface (BCI)," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 12, no. 3, pp. 331–338, Sep. 2004. 10

[21] S. Coyle, T. Ward, C. Markham, and G. McDarby, "On the suitability of near-infrared (NIR) systems for next-generation brain–computer interfaces," *Physiological Measurement*, vol. 25, p. 815, 2004. 10

[22] G. Schalk and E. Leuthardt, "Brain-computer interfaces using electrocorticographic signals," *Biomedical Engineering, IEEE Reviews in*, vol. 4, pp. 140 –154, 2011. 10

[23] J. M. Carmena, M. A. Lebedev, R. E. Crist, J. E. O'Doherty, D. M. Santucci, D. F. Dimitrov, P. G. Patil, C. S. Henriquez, and M. A. L. Nicolelis, "Learning to control a brain-machine interface for reaching and grasping by primates," *PLoS Biol*, vol. 1, no. 2, 10 2003. 10, 68

[24] C. Kemere, K. V. Shenoy, and T. H. Meng, "Model-based neural decoding of reaching movements: a maximum likelihood approach," *Biomedical Engineering, IEEE Transactions on*, vol. 51, no. 6, pp. 925–932, 2004. 10

[25] A. Schwartz, "Cortical neural prosthetics," *Annual Review of Neuroscience*, vol. 27, no. 1, pp. 487–507, 2004. 10

[26] E. W. W. Brendan Z Allison and J. R. Wolpaw, "Brain–computer interface systems: progress and prospects," *Expert Review of Medical Devices*, vol. 4, no. 4, 2007. 11

[27] G. Pfurtscheller and C. Neuper, "Motor imagery and direct brain-computer communication," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1123 –1134, jul 2001. 14

[28] T. Wang, J. Deng, and B. He, "Classifying EEG-based motor imagery tasks by means of time-frequency synthesized spatial patterns," *Clinical Neurophysiology*, vol. 115, no. 12, pp. 2744 – 2753, 2004. 14

[29] C. Neuper, G. R. Muller-Putz, R. Scherer, and G. Pfurtscheller, "Motor imagery and EEG-based control of spelling devices and neuroprostheses," in *Event-Related Dynamics of Brain Oscillations*, ser. Progress in Brain Research, C. Neuper and W. Klimesch, Eds. Elsevier, 2006, vol. 159, pp. 393 – 409. 14

[30] B. Blankertz, C. Sannelli, S. Halder, E. Hammer, A. Kübler, K. Müller, G. Curio, and T. Dickhaus, "Neurophysiological predictor of SMR-based BCI performance," *NeuroImage*, vol. 51, no. 4, pp. 1303–1309, 2010. 14

[31] O. Tonet, M. Marinelli, L. Citi, P. Rossini, L. Rossini, G. Megali, and P. Dario, "Defining brain-machine interface applications by matching interface performance with device requirements," *Journal of Neuroscience Methods*, vol. 167, no. 1, pp. 91–104, 2008. 14, 15

[32] L. Farwell and E. Donchin, "Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials," *Electroencephalography and Clinical Neurophysiology*, vol. 70, pp. 510–523, 1988. 14, 18, 114

[33] E. Donchin, K. Spencer, and R. Wijesinghe, "The mental prosthesis: assessing the speed of a P300-based brain-computer interface," *Rehabilitation Engineering, IEEE Transactions on*, vol. 8, no. 2, pp. 174 –179, jun 2000. 14, 18, 114

[34] E. W. Sellers, A. Kubler, and E. Donchin, "Brain-computer interface research at the University of South Florida Cognitive Psychophysiology Laboratory: the P300 speller," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 14, no. 2, pp. 221–224, 2006. 14

[35] M. Middendorf, G. McMillan, G. Calhoun, and K. Jones, "Brain-computer interfaces based on the steady-state visual-evoked response," *Rehabilitation Engineering, IEEE Transactions on*, vol. 8, no. 2, pp. 211 –214, jun 2000. 15

[36] Y. Wang, X. Gao, B. Hong, C. Jia, and S. Gao, "Brain-computer interfaces based on visual evoked potentials," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 27, no. 5, pp. 64 –71, september-october 2008. 15

[37] X. Gao, D. Xu, M. Cheng, and S. Gao, "A BCI-based environmental controller for the motion-disabled," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 11, no. 2, pp. 137 –140, june 2003. 15

[38] H.-J. Hwang, J.-H. Lim, Y.-J. Jung, H. Choi, S. W. Lee, and C.-H. Im, "Development of an SSVEP-based BCI spelling system adopting a QWERTY-style LED keyboard," *Journal of Neuroscience Methods*, vol. 208, no. 1, pp. 59 – 65, 2012. 15, 21, 22, 23

[39] G. Schalk, J. Wolpaw, D. McFarland, and G. Pfurtscheller, "EEG-based communication: presence of an error potential," *Clinical Neurophysiology*, vol. 111, no. 12, pp. 2138–2144, 2000. 15

[40] P. Ferrez and J. del R Millan, "Error-related EEG potentials generated during simulated brain–computer interaction," *Biomedical Engineering, IEEE Transactions on*, vol. 55, no. 3, pp. 923–929, 2008. 15

[41] A. Royer and B. He, "Goal selection versus process control in a brain–computer interface based on sensorimotor rhythms," *Journal of Neural Engineering*, vol. 6, p. 016005, 2009. 15, 67, 70

[42] G. Vanacker, J. del R Millán, E. Lew, P. Ferrez, F. Moles, J. Philips, H. Van Brussel, and M. Nuttin, "Context-based filtering for assisted brain-actuated wheelchair driving," *Computational Intelligence and Neuroscience*, vol. 2007, pp. 3–3, 2007. 16, 68

[43] F. Galan, M. Nuttin, E. Lew, P. Ferrez, G. Vanacker, J. Philips, and J. del R. Millan, "A brain-actuated wheelchair: Asynchronous and non-invasive brain-computer interfaces for continuous control of robots," *Clinical Neurophysiology*, vol. 119, no. 9, pp. 2159 – 2169, 2008. 16, 68, 130

[44] J. Millan, F. Renkens, J. Mouriño, and W. Gerstner, "Noninvasive brain-actuated control of a mobile robot by human EEG," *Biomedical Engineering, IEEE Transactions on*, vol. 51, no. 6, pp. 1026–1033, 2004. 16, 68

[45] A. Royer, A. Doud, M. Rose, and B. He, "EEG control of a virtual helicopter in 3-dimensional space using intelligent control strategies," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, no. 6, pp. 581 –589, dec. 2010. 16, 68

[46] X. Perrin, R. Chavarriaga, F. Colas, R. Siegwart, and J. d. R. Millán, "Brain-coupled interaction for semi-autonomous navigation of an assistive robot," *Robotics and Autonomous Systems*, vol. 58, pp. 1246–1255, December 2010. 16, 68

[47] L. Srinivasan, U. T. Eden, A. S. Willsky, and E. N. Brown, "A state-space analysis for reconstruction of goal-directed movements using neural signals," *Neural Computation*, vol. 18, no. 10, pp. 2465–2494, 2006. 16, 68

[48] M. Chung, W. Cheung, R. Scherer, and R. Rao, "A hierarchical architecture for adaptive brain-computer interfacing," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011. 16

[49] J. Faller, G. Müller-Putz, D. Schmalstieg, and G. Pfurtscheller, "An application framework for controlling an avatar in a desktop-based virtual environment via a software SSVEP brain-computer interface," *Presence: Teleoperators and Virtual Environments*, vol. 19, no. 1, pp. 25–34, 2010. 17

[50] B. Rebsamen, E. Burdet, C. Guan, H. Zhang, C. L. Teo, Q. Zeng, C. Laugier, and M. H. Ang Jr., "Controlling a wheelchair indoors using thought," *Intelligent Systems, IEEE*, vol. 22, pp. 18–24, 2007. 17

[51] B. Rebsamen, C. Guan, H. Zhang, C. Wang, C. Teo, M. Ang, and E. Burdet, "A brain controlled wheelchair to navigate in familiar environments," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, no. 6, pp. 590–598, 2010. 17, 68, 130

[52] C. J. Bell, P. Shenoy, R. Chalodhorn, and R. P. N. Rao, "Control of a humanoid robot by a noninvasive brain-computer interface in humans," *Journal of Neural Engineering*, vol. 5, no. 2, pp. 214–220, 2008. 17, 68

[53] I. Iturrate, J. Antelis, A. Kubler, and J. Minguez, "A noninvasive brain-actuated wheelchair based on a P300 neurophysiological protocol and automated navigation," *Robotics, IEEE Transactions on*, vol. 25, no. 3, pp. 614–627, June 2009. 17, 68, 69, 70, 81, 85, 86, 102, 103, 105, 130

[54] H. Cecotti, "Spelling with non-invasive brain-computer interfaces - current and future trends," *Journal of Physiology-Paris*, vol. 105, no. 1–3, pp. 106 – 114, 2011. 18, 107

[55] J. Mak, Y. Arbel, J. Minett, L. McCane, B. Yuksel, D. Ryan, D. Thompson, L. Bianchi, and D. Erdogmus, "Optimizing the P300-based brain–computer interface: current status, limitations and future directions," *Journal of Neural Engineering*, vol. 8, no. 2, p. 025003, 2011. 18, 19

[56] R. Fazel-Rezai, B. Z. Allison, C. Guger, E. W. Sellers, S. C. Kleih, and A. Kbler, "P300 brain computer interface: current challenges and emerging trends," *Frontiers in Neuroengineering*, vol. 5, no. 14, 2012. 18

[57] T. Vaughan, D. McFarland, G. Schalk, W. Sarnacki, D. Krusienski, E. Sellers, and J. Wolpaw, "The wadsworth BCI research and development program: at home with BCI," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 14, no. 2, pp. 229–233, 2006. 18

[58] F. Nijboer, E. Sellers, J. Mellinger, M. Jordan, T. Matuz, A. Furdea, S. Halder, U. Mochty, D. Krusienski, T. Vaughan et al., "A P300-based brain–computer interface for people with amyotrophic lateral sclerosis," *Clinical neurophysiology*, vol. 119, no. 8, pp. 1909–1916, 2008. 18

[59] E. W. Sellers, T. M. Vaughan, and J. R. Wolpaw, "A brain-computer interface for long-term independent home use," *Amyotrophic Lateral Sclerosis*, vol. 11, no. 5, pp. 449–455, 2010. 18

[60] G. Townsend, B. LaPallo, C. Boulay, D. Krusienski, G. Frye, C. Hauser, N. Schwartz, T. Vaughan, J. Wolpaw, and E. Sellers, "A novel P300-based brain-computer interface stimulus presentation paradigm: moving beyond rows and columns," *Clinical Neurophysiology*, vol. 121, no. 7, p. 1109, 2010. 19

[61] H. Serby, E. Yom-Tov, and G. Inbar, "An improved P300-based brain-computer interface," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 13, no. 1, pp. 89–98, March. 19

[62] A. Lenhardt, M. Kaper, and H. Ritter, "An adaptive P300-based online brain–computer interface," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 16, no. 2, pp. 121–130, 2008. 19

[63] W. Speier, C. Arnold, J. Lu, R. Taira, and N. Pouratian, "Natural language processing with dynamic classification improves P300 speller accuracy and bit rate," *Journal of Neural Engineering*, vol. 9, no. 1, 2011. 19, 20

[64] Y. Li, C. S. Nam, B. B. Shadden, and S. L. Johnson, "A P300-based brain–computer interface: Effects of interface type and screen size," *Intl. Journal of Human–Computer Interaction*, vol. 27, no. 1, pp. 52–68, 2011. 20

[65] D. B. Ryan, G. E. Frye, G. Townsend, D. R. Berry, S. Mesa-G, N. A. Gates, and E. W. Sellers, "Predictive spelling with a P300-based brain–computer interface: Increasing the rate of communication," *International Journal of Human-Computer Interaction*, vol. 27, no. 1, pp. 69–84, 2011. 20

[66] J. Park and K. Kim, "A POMDP approach to optimizing P300 speller BCI paradigm," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 20, no. 4, pp. 584–594, 2012. 20, 114

[67] R. Ma, N. Aghasadeghi, J. Jarzebowski, T. Bretl, and T. Coleman, "A stochastic control approach to optimally designing variable-sized menus in P300 communication prostheses," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 2012. 20, 112

[68] S. Ahi, H. Kambara, and Y. Koike, "A dictionary-driven P300 speller with a modified interface," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, no. 99, pp. 1–1, 2011. 20

[69] S. Lu, C. Guan, and H. Zhang, "Unsupervised brain computer interface based on intersubject information and online adaptation," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, no. 2, pp. 135 –145, april 2009. 20

[70] R. Panicker, S. Puthusserypady, and Y. Sun, "Adaptation in P300 brain–computer interfaces: A two-classifier cotraining approach," *Biomedical Engineering, IEEE Transactions on*, vol. 57, no. 12, pp. 2927–2935, 2010. 20

[71] P. Kindermans, H. Verschore, D. Verstraeten, and B. Schrauwen, "A P300 BCI for the masses: Prior information enables instant unsupervised spelling," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 719–727. 20

[72] R. Scherer, G. Muller, C. Neuper, B. Graimann, and G. Pfurtscheller, "An asynchronously controlled eeg-based virtual keyboard: improvement of the spelling rate," *Biomedical Engineering, IEEE Transactions on*, vol. 51, no. 6, pp. 979 –984, june 2004. 20

[73] B. Blankertz, M. Krauledat, G. Dornhege, J. Williamson, R. Murray-Smith, and K.-R. Muller, "A note on brain actuated spelling with the berlin brain-computer interface," in *Universal Access in Human-Computer Interaction. Ambient Interaction*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 759–768. 20, 21, 112

[74] T. D'albis, R. Blatt, R. Tedesco, L. Sbattella, and M. Matteucci, "A predictive speller controlled by a brain-computer interface based on motor imagery," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 19, no. 3, p. 20, 2012. 21, 112

[75] H. Cecotti, "A self-paced and calibration-less SSVEP-based brain–computer interface speller," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, no. 2, pp. 127–133, 2010. 21, 22, 112, 122

[76] I. Volosyak, "SSVEP-based Bremen–BCI interface—boosting information transfer rates," *Journal of Neural Engineering*, vol. 8, no. 3, p. 036020, 2011. 21, 22, 112, 121, 125

[77] B. Allison, T. Luth, D. Valbuena, A. Teymourian, I. Volosyak, and A. Graser, "Bci demographics: How many (and what kinds of) people can use an SSVEP BCI?" *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, no. 2, pp. 107 –116, april 2010. 22

[78] I. Volosyak, A. Moor, and A. Gräser, "A dictionary-driven SSVEP speller with a modified graphical user interface," *Advances in Computational Intelligence*, pp. 353–361, 2011. 22

[79] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, July 2006. 28, 34, 35, 72, 132

[80] T. P. Coleman, "A stochastic control approach to 'posterior matching'-style feedback communication schemes," in *IEEE International Symposium on Information Theory*, 2009. 28

[81] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is NP-complete," *Information Processing Letters*, vol. 5, pp. 15–17, 1976. 38

[82] S. Kosaraju, T. Przytycka, and R. Borgstrom, "On an optimal split tree problem," in *Algorithms and Data Structures*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1999, vol. 1663, pp. 157–168. 38, 39

[83] B. Settles, "Active learning," in *Synthesis Lectures on Artificial Intelligence and Machine Learning.* Morgan & Claypool Publishers, June 2012, p. 114. 39

[84] S. Dasgupta, "Analysis of a greedy active learning strategy," *Advances in neural information processing systems*, vol. 17, pp. 337–344, 2005. 39

[85] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research*, vol. 42, no. 1, pp. 427–486, 2011. 39

[86] R. Nowak, "The geometry of generalized binary search," *Information Theory, IEEE Transactions on*, vol. 57, no. 12, pp. 7893–7906, Dec 2011. 39

[87] G. Bellala, S. Bhavnani, and C. Scott, "Group-based active query selection for rapid diagnosis in time-critical situations," *Information Theory, IEEE Transactions on*, vol. 58, no. 1, pp. 459–478, Jan 2012. 39

[88] D. Golovin, A. Krause, and D. Ray, "Near-optimal bayesian active learning with noisy observations," in *NIPS'10*, 2010, pp. 766–774. 39

[89] K. Chaloner and I. Verdinelli, "Bayesian experimental design: A review," *Statistical Science*, pp. 273–304, 1995. 39, 40

[90] R. S. Woodworth, "The accuracy of voluntary movement," Ph.D. dissertation, Columbia University, New York, July 1899. 47

[91] N. Hogan, J. A. Doeringer, and H. I. Krebs, "Arm movement control is both continuous and discrete," *Cognitive Studies*, vol. 6, no. 3, pp. 254–273, 1999. 47

[92] T. Flash and B. Hochner, "Motor primitives in vertebrates and invertebrates," *Current Opinion in Neurobiology*, vol. 15, no. 6, pp. 660–666, 2005. 47

[93] B. Rohrer, S. Fasoli, H. I. Krebs, R. Hughes, B. Volpe, W. R. Frontera, J. Stein, and N. Hogan, "Movement smoothness changes during stroke recovery," *J Neurosci*, vol. 22, no. 18, pp. 8297–304, Sep 2002. 47

[94] H. I. Krebs, M. L. Aisen, B. T. Volpe, and N. Hogan, "Quantization of continuous arm movements in humans with brain injury," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 96, no. 8, pp. 4645–4649, 04 1999. 47

[95] F. Polyakov, R. Drori, Y. Ben-Shaul, M. Abeles, and T. Flash, "A compact representation of drawing movements with sequences of parabolic primitives," *PLoS Computational Biology*, vol. 5, no. 7, July 2009. 47

[96] W. L. Nelson, "Physical principles for economies of skilled movements," *Biological Cybernetics*, vol. 46, no. 2, pp. 135–147, 02 1983. 47

[97] E. Todorov and M. I. Jordan, "Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements," *Journal of Neurophysiology*, vol. 80, no. 2, pp. 696–714, 8 1998. 47

[98] H. Hicheur, Q. Pham, G. Arechavaleta, J. Laumond, and A. Berthoz, "The formation of trajectories during goal-oriented locomotion in humans. I. A stereotyped behaviour," *European Journal of Neuroscience*, vol. 26, no. 8, pp. 2376–2390, 2007. 47

[99] Q. Pham, H. Hicheur, G. Arechavaleta, J. Laumond, and A. Berthoz, "The formation of trajectories during goal-oriented locomotion in humans. II. A maximum smoothness model," *European Journal of Neuroscience*, vol. 26, no. 8, pp. 2391–2403, 2007. 47

[100] O. Jenkins and M. Mataric, "Performance-derived behavior vocabularies: Data-driven acqusition of skills from motion," *International Journal of Humanoid Robotics*, vol. 1, no. 2, pp. 237–288, 2004. 47

[101] D. Forsyth, O. Arikan, and L. Ikemoto, *Computational Studies of Human Motion: Tracking and Motion Synthesis.* Now Pub, 2006. 47, 51

[102] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *Robotics & Automation Magazine, IEEE*, vol. 14, no. 1, pp. 61–70, 2007. 47

[103] Y. Li, T. Wang, and H. Y. Shum, "Motion texture: a two-level statistical model for character motion synthesis," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '02. New York, NY, USA: ACM, July 2002, pp. 465–472. 47, 51

[104] D. Del Vecchio, R. Murray, and P. Perona, "Decomposition of human motion into dynamics-based primitives with application to drawing tasks," *Automatica*, vol. 39, no. 12, pp. 2085–2098, dec 2003. 47

[105] T. Inamura, I. Toshima, H. Tanie, and Y. Nakamura, "Embodied symbol emergence based on mimesis theory," *The International Journal of Robotics Research*, vol. 23, no. 4-5, p. 363, 2004. 47

[106] A. Fishbach, S. Roy, C. Bastianen, L. Miller, and J. Houk, "Deciding when and how to correct a movement: discrete submovements as a decision making process," *Experimental Brain Research*, vol. 177, pp. 45–63, 2007. 47

[107] A. Gray, E. Abbena, and S. Salamon, *Modern differential geometry of curves and surfaces with Mathematica.* Chapman & Hall/CRC, 2006. 49

[108] S. M. LaValle, *Planning algorithms.* New York, NY: Cambridge University Press, 2006. 49, 58, 59

[109] A. Bobick and Y. Ivanov, "Action recognition using probabilistic parsing," in *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, June 1998, pp. 196 –202. 51

[110] N. Kojo, T. Inamura, K. Okada, and M. Inaba, "Gesture recognition for humanoids using proto-symbol space," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, Dec. 2006, pp. 76 –81. 51

[111] O. Jenkins and M. Mataric, "Automated derivation of behavior vocabularies for autonomous humanoid motion," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems.* ACM, 2003, pp. 225–232. 51

[112] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," in *Algorithmic Foundation of Robotics VII*, ser. Springer Tracts in Advanced Robotics, S. Akella, N. Amato, W. Huang, and B. Mishra, Eds. Springer Berlin / Heidelberg, 2008, vol. 47, pp. 507–522. 51

[113] F. Stulp, E. Oztop, P. Pastor, M. Beetz, and S. Schaal, "Compact models of motor primitive variations for predictable reaching and obstacle avoidance," in *IEEE-RAS/RSJ International Conference on Humanoid Robots*, 2009. 51

[114] J. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *Communications, IEEE Transactions on*, vol. 32, no. 4, pp. 396–402, Apr 1984. 53, 119

[115] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order Markov models," *Journal of Artificial Intelligence Research (JAIR)*, vol. 22, pp. 385–421, 2004. 53, 123

[116] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987. 54

[117] J. S. B. Mitchell, "Shortest paths and networks," in *Handbook of Discrete and Computational Geometry, 2nd Ed.*, J. E. Goodman and J. O'Rourke, Eds. New York: Chapman and Hall/CRC Press, 2004, pp. 607–641. 58, 59

[118] J. O'Rourke, "Visibility," in *Handbook of Discrete and Computational Geometry, 2nd Ed.*, J. E. Goodman and J. O'Rourke, Eds. New York: Chapman and Hall/CRC Press, 2004, pp. 643–663. 58, 59

[119] B. Tovar, L. Guilamo, and S. M. LaValle, "Gap navigation trees: Minimal representation for visibility-based tasks," in *Algorithmic Foundations of Robotics VI*, ser. Springer Tracts in Advanced Robotics, M. Erdmann, M. Overmars, D. Hsu, and F. der Stappen, Eds. Springer Berlin Heidelberg, 2005, vol. 17, pp. 425–440. 59

[120] N. D. Ratliff, A. J. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *ICML '06: Proceedings of the 23rd international conference on Machine learning.* New York, NY, USA: ACM, 2006, pp. 729–736. 64, 65

[121] N. Ratliff, J. A. Bagnell, and M. Zinkevich, "Subgradient methods for maximum margin structured learning," in *ICML Workshop on Learning in Structured Output Spaces*, 2006. 64, 65

[122] J. R. Wolpaw and D. J. McFarland, "Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 51, pp. 17 849–17 854, 2004. 68

[123] S. Musallam, B. D. Corneil, B. Greger, H. Scherberger, and R. A. Andersen, "Cognitive control signals for neural prosthetics," *Science*, vol. 305, no. 5681, pp. 258–262, 2004. 68

[124] "FS One V2, Precision RC Flight Simulator," Software Developed by InertiaSoft, Champaign, IL, http://www.fsone.com, Accessed April 2012. 76

[125] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods.* Prentice-Hall, Inc., 1990. 76

[126] P. Nunez and R. Srinivasan, *Electric fields of the brain: the neurophysics of EEG.* Oxford University Press New York, 2006. 78

[127] M. McCormick, R. Ma, and T. Coleman, "An analytic spatial filter and a hidden markov model for enhanced information transfer rate in EEG-based brain computer interfaces," in *ICASSP*, Dallas, TX, March 2010. 78

[128] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on.* IEEE, 2009, pp. 3931–3936. 92, 99, 101

[129] B. O'Neill, *Elementary Differential Geometry*, 2nd ed. Academic Press, 1997. 93

[130] G. Muller-Putz, R. Scherer, C. Brauneis, and G. Pfurtscheller, "Steady-state visual evoked potential (SSVEP)-based communication: impact of harmonic frequency components," *Journal of neural engineering*, vol. 2, p. 123, 2005. 100

[131] R. Fisher, G. Harding, G. Erba, G. Barkley, and A. Wilkins, "Photic- and pattern-induced seizures: A review for the epilepsy foundation of america working group," *Epilepsia*, vol. 46, no. 9, pp. 1426–1441, 2005. 100

[132] V. Jurcak, D. Tsuzuki, and I. Dan, "10/20, 10/10, and 10/5 systems revisited: their validity as relative head-surface-based positioning systems," *Neuroimage*, vol. 34, no. 4, pp. 1600–1611, 2007. 100, 118

[133] G. Schalk, D. McFarland, T. Hinterberger, N. Birbaumer, and J. Wolpaw, "BCI2000: a general-purpose brain-computer interface (BCI) system," *Biomedical Engineering, IEEE Transactions on*, vol. 51, no. 6, pp. 1034–1043, 2004. 100

[134] O. Friman, T. Luth, I. Volosyak, and A. Graser, "Spelling with steady-state visual evoked potentials," in *Neural Engineering, 2007. CNE'07. 3rd International IEEE/EMBS Conference on.* IEEE, 2007, pp. 354–357. 100, 118

[135] D. Zhu, J. Bieger, G. G. Molina, and R. M. Aarts, "A survey of stimulation methods used in SSVEP-based BCIs," *Computational intelligence and neuroscience*, vol. 2010, p. 1, 2010. 107

[136] M. Billinger, I. Daly, V. Kaiser, J. Jin, B. Z. Allison, G. R. Müller-Putz, and C. Brunner, "Is it significant? guidelines for reporting BCI performance," in *Towards Practical Brain-Computer Interfaces*, ser. Biological and Medical Physics, Biomedical Engineering. Springer Berlin Heidelberg, 2013, pp. 333–354. 112

[137] D. J. Ward, A. F. Blackwell, and D. J. C. MacKay, "Dasher: A gesture-driven data entry interface for mobile computing." *Human-Computer Interaction*, vol. 17, no. 2/3, pp. 199–228, 2002. 119

[138] C. Manning and H. Schütze, *Foundations of statistical natural language processing.* MIT Press, 2002. 123

[139] B. Ziebart, A. Maas, J. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. AAAI*, 2008, pp. 1433–1438. 132

[140] A. Krause, "Optimizing sensing: Theory and applications," Ph.D. dissertation, School of Computer Science Carnegie Mellon University, 2008. 133

[141] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952. 133