A Brain–Machine Interface to Navigate a Mobile Robot in a Planar Workspace: Enabling Humans to Fly Simulated Aircraft With EEG

Abdullah Akce, *Student Member, IEEE*, Miles Johnson, *Student Member, IEEE*, Or Dantsker, and Timothy Bretl, *Member, IEEE*

***Abstract*—This paper presents an interface for navigating a mobile robot 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 (EEG). The approach is to construct an ordered symbolic language for smooth planar curves and to use these curves as desired paths for a mobile robot. The underlying problem is then to design a communication protocol by which the user can, with vanishing error probability, specify a string in this language using a sequence of inputs. Such a protocol, provided by tools from information theory, relies on a human user's ability to compare smooth curves, just like they can compare strings of text. 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 are able to specify desired paths despite a wide range of errors made in decoding EEG signals.**

***Index Terms*—Brain–machine interface (BMI), information theory, robotics, semi-autonomous navigation.**

I. Introduction

Brain–machine interfaces (BMIs) provide new output pathways for the brain by translating measurements of brain activity into inputs for an external device [1]. These output pathways typically function in one of two different ways: process control and goal selection [1], [2]. 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 both invasive BMIs for control of a robotic arm by primates [3], and noninvasive BMIs for control of a cursor [4], a wheelchair [5], [6], or a mobile robot [7]–[9]. A problem with this strategy, particularly for noninvasive 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 [5], [6], [8], [10]. 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.* [5] enabled a human user to drive a powered wheelchair with three inputs—"left," "right," and "forward"—obtained from electroencephalograph (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 [11], and noninvasive BMIs that allowed human users to choose from a set of destinations for a wheelchair [12], or objects to be picked up by a humanoid robot [13]. 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.* [12] 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.* [14] 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.

The BMI we present in this paper uses a hybrid strategy that is similar to the one proposed by Iturrate *et al.* [14]. 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

Manuscript received April 22, 2012; revised August 31, 2012, November 20, 2012; accepted November 25, 2012. Date of publication December 13, 2012; date of current version March 07, 2013. This work was supported by the National Science Foundation under the Award CNS-0931871 and Award CMMI-0955088.

A. Akce is with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: aakce2@illinois.edu) .

M. Johnson and T. Bretl are with the Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: mjohns50@illinois.edu; tbretl@illinois.edu).

O. Dantsker is with the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: dantske1@illinois.edu).

Digital Object Identifier 10.1109/TNSRE.2012.2233757

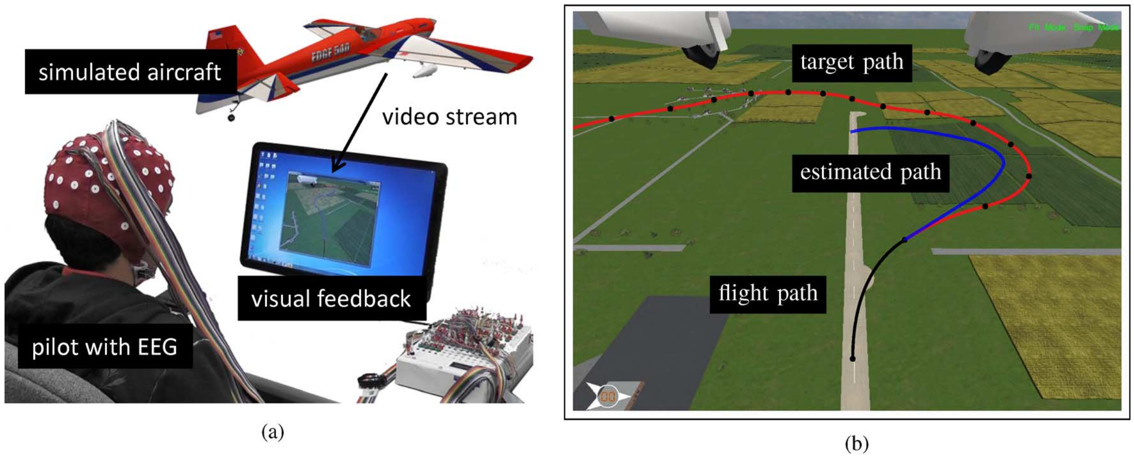


Fig. 1. Our interface for flying a simulated aircraft at a fixed altitude and speed with input only from EEG, see (a). The pilot flies the aircraft by imagining either left- or right- hand movement, the choice between which is based on visual feedback provided by a graphical display. In the tracking 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). (a) Illustration of our interface. (b) A snapshot of the display in the tracking task.

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 derive such a protocol using tools from information theory.

We compare our approach to the hybrid strategy of Iturrate *et al.* [14] 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 paper is not to provide a comparison between process control, goal selection, or hybrid approaches. Such a comparison has been performed in [2]. 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.

As a case study in the application of our approach, we present a BMI in this paper that allows a human pilot to fly a simulated aircraft at a fixed speed and altitude using EEG (Fig. 1). In previous work, with a preliminary version of the interface, we demonstrated EEG-based teleoperation of a physical model-aircraft in a perimeter surveillance task [15]. Although our pilot achieved the task by successfully flying the aircraft with EEG over a 3 km perimeter in 5 min, 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). In this paper, 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. In this paper, 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 paper is not to advance the state-of-the-art in EEG signal processing.

The rest of this paper is organized as follows. First, we describe our approach that allows a human user to navigate a mobile robot moving at a fixed speed in a planar workspace with EEG signals (Section II). We implemented this approach in our interface for flying a simulated aircraft with EEG (Section III). We performed Monte Carlo experiments to compare our approach the hybrid approach of [14], and performed EEG experiments to evaluate our BMI in a high-fidelity flight simulation environment. (Section IV). Results showed that our approach outperformed the hybrid approach of [14] and our BMI allowed the majority of subjects in EEG experiments to succeed in flying the aircraft over desired paths (Section V). Finally, we conclude by presenting the limitations of our approach and how these limitations might be resolved in future work (Section VI).

II. Method for Navigating a Mobile Robot in a Planar Workspace

Our goal is to design a BMI 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 II-A). Such a protocol is provided by the combination of arithmetic coding as a method of lossless data

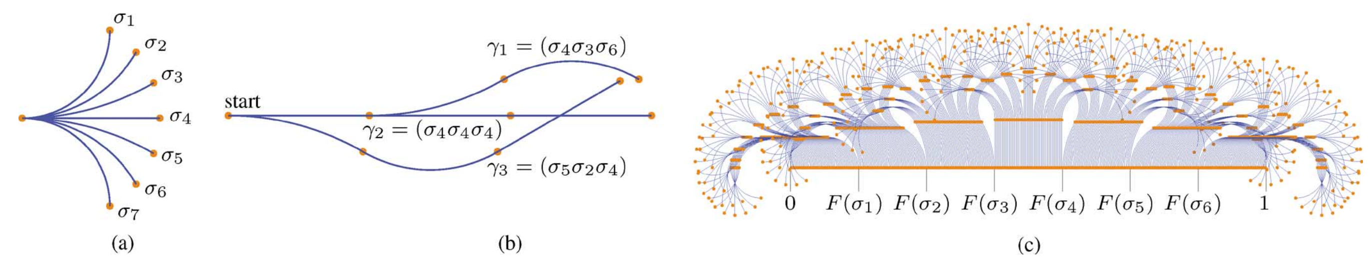


Fig. 2. An ordered symbolic language for describing smooth planar paths. (a) An example alphabet with a set of fixed-length circular arcs. (b) Three sample paths composed from this alphabet. They exhibit the ordering $\gamma_1 < \gamma_2 < \gamma_3$. (c) All paths composed of three symbols mapped to 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. (a) Our alphabet. (b) Sample curves. (c) Arithmetic coding.

compression with posterior matching as a capacity-achieving channel code (Section II-B). Then, we apply this protocol in our BMI design for specifying a desired path for a robot moving at a fixed speed (Section II-C).

A. Problem Formulation

In this section, we make three choices about system architecture that allow us to cast BMI design as a communication problem. Other choices could have been made—the appeal of our approach is that it admits a provably optimal communication protocol (see Section II-B).

1) Choices About System Architecture:

a) Structure of Desired Paths: Our first choice is to use piecewise-smooth planar curves from an ordered symbolic language as desired paths for the robot, which are tracked with an onboard control system. Any piecewise-smooth planar curve of arbitrary length L is the projection $(x, y) : [0, L] \to \mathbb{R}^2$ of the solution to

$$\dot{x} = \cos\theta \quad \dot{y} = \sin\theta \quad \dot{\theta} = -u \tag{1}$$

for an initial condition

$$x(0) = x_0 \quad y(0) = y_0 \quad \theta(0) = \theta_0 \tag{2}$$

and piecewise-smooth curvature $u : [0, L] \to \mathbb{R}$. We assume $x_0 = y_0 = \theta_0 = 0$ without loss of generality. We consider a subset of curves for which u is piecewise-constant on intervals of length d and takes values in a finite set $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$, where elements of Σ are ordered so that $\sigma_i < \sigma_j$ for all $i, j \in \{1, \ldots, m\}$ satisfying $i < j$. In particular, on each interval $k \in \{1, 2, \ldots\}$, we require that

$$u(t) = \sigma_{\mathcal{I}(k)} \quad (k-1)d \le t \le kd \tag{3}$$

for some $\mathcal{I} : \mathbb{N} \to \Sigma$. We refer to the finite set Σ as an *alphabet*, to elements $\sigma_i \in \Sigma$ of the alphabet as *symbols*, and to the sequence $\gamma = (\sigma_{\mathcal{I}(1)}, \sigma_{\mathcal{I}(2)}, \ldots)$ as a *string*. We denote the set of all strings γ by Σ^*, and the set of all strings of length M by Σ^M. There is a one-to-one correspondence between Σ^* and the set of all curves (x, y) that satisfy (1)–(3), so we often refer to $\gamma \in \Sigma^*$ itself as a curve. For any two curves $\gamma = (\sigma_{\mathcal{I}(1)}, \sigma_{\mathcal{I}(2)}, \ldots)$ and $\gamma' = (\sigma_{\mathcal{J}(1)}, \sigma_{\mathcal{J}(2)}, \ldots)$, we say that $\gamma < \gamma'$ if and only if $\sigma_{\mathcal{I}(k)} < \sigma_{\mathcal{J}(k)}$, where k is the minimum index at which $\mathcal{I}(k) \ne \mathcal{J}(k)$. This ordering corresponds to the notion that γ "turns left" the first time it differs from γ', and allows us to "alphabetize" curves just like we would alphabetize strings of text (see Fig. 2). We associate a statistical model to our symbolic language by assuming that the user's desired path is generated by a Markov process. From offline or online data, we can compute a Markov model that assigns a conditional probability to each symbol given a string of prior symbols.

b) 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. Motor imagery is a well-established paradigm for non-invasive measurement and interpretation of brain activity that has recently found application to BMI design [16]. With this paradigm, the human user imagines moving either their left or right hand, and the classifier attempts to recover this binary decision based on characteristic patterns of EEG signals in particular frequency bands. Although the details of this process are complex (and a topic of active research), in this paper we treat it as a black box and model it simply as a binary symmetric channel with some crossover probability ϵ [17]. 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 $\mathrm{P}(y_k \,|\, x_k) = 1 - \epsilon$ if $y_k = x_k$ and $\mathrm{P}(y_k \,|\, x_k) = \epsilon$ otherwise.

c) Mechanism for Sensory Feedback: Our third choice is to use a graphical display to show candidate paths $\hat{\gamma}$ to the human user. As we will see in Section II-B, the best choice of $\hat{\gamma}$ 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 $\gamma^* < \hat{\gamma}$, i.e., whether or not their desired path is to the left of the candidate path (according to the lexicographic ordering we defined above). It will turn out that this decision is exactly the one required by the optimal communication protocol that we derive in Section II-B. The availability of causal noiseless feedback cannot increase the capacity of a binary symmetric channel, which is an upper bound on the achievable transmission rate. However, this feedback can dramatically simplify the communication protocol [17], which is helpful because this protocol must be implemented in part by a human user.

2) Interface Design as a Communication Problem: With the choices we made in the previous section, our goal has now become the design of a protocol to communicate a string γ^* in an

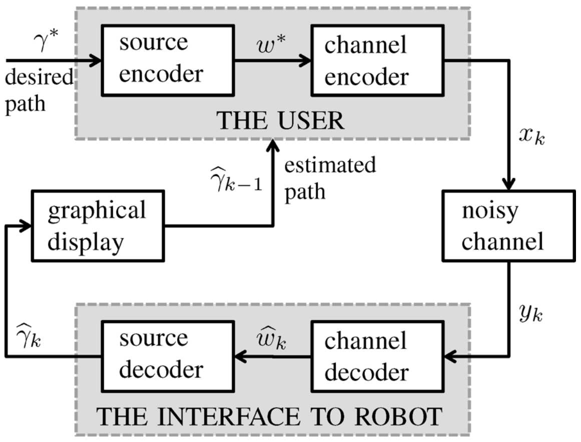


Fig. 3. Our abstraction of BMI design as a communication problem. The human user conveys their desired path γ^* to the robot through a binary symmetric channel with causal feedback provided by a graphical display. The communication protocol is designed in two parts: a source code and a channel code. See text for details.

ordered symbolic language across a binary symmetric channel with causal noiseless feedback. Such a protocol consists of an encoder (essentially mapping the estimate $\hat{\gamma}_{k-1}$ to the input x_k) and a decoder (essentially mapping the outputs $y_1, \ldots, y_k$ to the estimate $\hat{\gamma}_k$), as shown in Fig. 3. We call this protocol optimal if it achieves capacity and if $P(\hat{\gamma}_k \neq \gamma^*) \to 0$ as $k \to \infty$.

The source-channel separation theorem tells us that the design of an optimal communication protocol can be done in two stages [17]. The first stage produces a *source code*, which is an invertible mapping $\phi : \Sigma^* \to [0,1)$ from strings to points in the unit interval. Given a desired path γ^*, we follow standard convention and call $w^* = \phi(\gamma^*)$ the *message point*. The source code is optimal if the average number of bits used to represent w^* matches the entropy of γ^* as given by the statistical model defined in Section II-A1a. The second stage produces a *channel code*, which is a sequence of functions $x_k = f_k(w^*, \hat{w}_{k-1})$ that determine how the encoder chooses the input x_k, and a sequence of functions $\hat{w}_k = g_k(y_1, \ldots, y_k)$ that determine how the decoder computes the estimate $\hat{w}_k$. The channel code is optimal if it achieves the capacity C and if $P(|w^* - \hat{w}_k| > 2^{-kC}) \to 0$ as $k \to \infty$. If both the source code and the channel code are optimal, the resulting communication protocol will be optimal. We emphasize that "source code" and "channel code" are *not* synonymous with "encoder" and "decoder" (see Fig. 3).

Good choices for both a source code and a channel code will be derived in the following section. Keep in mind that "good" in this context means not only that codes are optimal but also that they are implementable in part by a human user.

B. Solution Approach

In the previous section, we reformulated BMI design as an optimal communication problem, which consisted of designing an optimal source code and an optimal channel code. In this section, we derive such codes using arithmetic coding (as the source code) and posterior matching (as the channel code).

1) Source Code: Our choice of source code is provided by a method of lossless data compression called arithmetic coding [18]. Arithmetic coding maps a random string $\gamma = (V_1, V_2, \ldots, V_M)$ with random variables $V_i \in \Sigma$ into a subinterval $[l_\gamma, r_\gamma) \in [0,1)$. Any point $W \in [l_\gamma, r_\gamma)$ can be used to represent γ, here we simply choose $\phi(\gamma)$ as the midpoint $(l_\gamma + r_\gamma)/2$. A set of paths and their mappings under a uniform prior over symbols are illustrated in Fig. 2(c). Given a desired length M, the inverse mapping $\phi^{-1} : [0,1) \to \Sigma^M$, from a real number $W \in [0,1)$ to a random string $\gamma = (V_1, V_2, \ldots, V_M)$, is described in Algorithm 1. This method is both optimal and has the useful property that it preserves lexicographic ordering, so that $\gamma < \gamma'$ if and only if $\phi(\gamma) < \phi(\gamma')$.

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

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

2) Channel Code: Our choice of channel code is provided by a principle for constructing optimal communication protocols in the presence of noiseless feedback that is called posterior matching [19]. This principle requires only the estimate $\hat{w}_{k-1}$ to be provided as noiseless feedback after obtaining $k-1$ channel outputs. For our communication channel (Section II-A1b)—a binary symmetric channel with noiseless feedback—posterior matching has a simple interpretation. Let X and Y be the random variables that specify the inputs x and the outputs y of the channel, respectively. Let W be the random variable that specify w^*. Denote the random sequence $(Y_1, \ldots, Y_k)$ as Y^k. Assume that after receiving $k-1$ channel outputs, the decoder has computed the posterior distribution $f_{W \mid Y^{k-1}}(W \mid y_1 \ldots y_{k-1})$, has chosen the estimate $\hat{w}_{k-1}$ and has provided $\hat{w}_{k-1}$ as feedback to the encoder. Then, the encoder selects the next input x_k as

$$f_k(w^*, \hat{w}_{k-1}) = \begin{cases} 0, & w^* < \hat{w}_{k-1} \\ 1, & w^* \geq \hat{w}_{k-1}. \end{cases} \tag{4}$$

After receiving the channel output y_k, the decoder uses the Bayes' rule to obtain $f_{W \mid Y^k}(W \mid y_1 \ldots y_k)$. If $y_k = 1$ (the case $y_k = 0$ is analogous), the decoder computes

$$\begin{aligned} &f_{W \mid Y^k}(W \mid y_1 \ldots y_k) \\ &= \eta \cdot \begin{cases} (1-\epsilon) \cdot f_{W \mid Y^{k-1}}(W \mid y_1 \ldots y_{k-1}), & \text{if } W \in [\hat{w}_{k-1}, 1) \\ \epsilon \cdot f_{W \mid Y^{k-1}}(W \mid y_1 \ldots y_{k-1}), & \text{otherwise} \end{cases} \end{aligned} \tag{5}$$

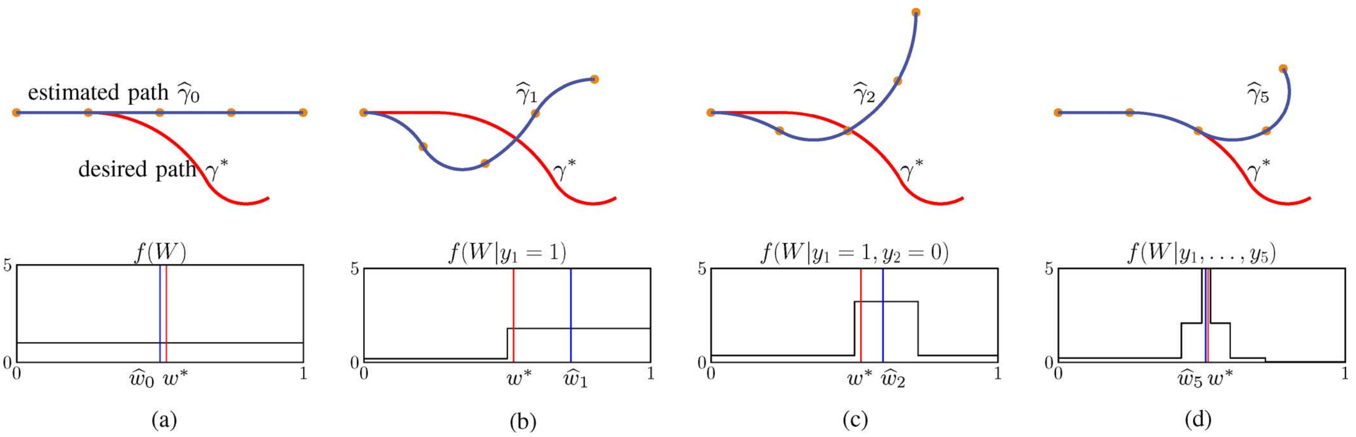


Fig. 4. Four iterations of the communication protocol between the user and the interface to the robot. The interface maintains a posterior distribution over the unit interval and displays the path that corresponds to the median of this distribution. The user responds by comparing the desired path to the estimated path using lexicographic ordering. See text for details. (a) Step 0. (b) Step 1. (c) Step 2. (d) Step 5.

where ϵ is the crossover probability and η is a normalizing constant. Then, the decoder chooses the estimate $\hat{w}_k$ as

$$g_k(y_1, \ldots, y_k) = \text{median}\left(f_{W\,|\,Y^k}(W\,|\,y_1 \ldots y_k)\right) \quad (6)$$

the median of this posterior distribution, and provides $\hat{w}_k$ as feedback to the encoder. The sequences of functions f_k and g_k define an optimal communication protocol for transmission of w^* over a binary symmetric channel with noiseless feedback.

This communication protocol is not only optimal but also easy for a human user—the "encoder"—to implement. Assume a graphical display shows the user the path $\hat{\gamma}_{k-1} = \phi^{-1}(\hat{w}_{k-1})$ that corresponds to the estimate $\hat{w}_{k-1}$. 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 $\hat{\gamma}_{k-1}$. Fig. 4 shows an example. Initially, the posterior is uniform and the median $\hat{w}_0$ corresponds to the straight path given by $\hat{\gamma}_0$. Because γ^* turns more right than $\hat{\gamma}_0$, 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 $\hat{\gamma}_0$ and decreasing the probability of all paths to the left of $\hat{\gamma}_0$—and generates a new estimate $\hat{w}_1$. In this case, the estimated path $\hat{\gamma}_1$ 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. 4(d).

C. 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 $\hat{\gamma}_0$, and updated after the kth user input to $\hat{\gamma}_k$ according to the protocol (Section II-B). In particular, after observing y_k, the message point $\hat{w}_k \in [0, 1)$ is computed using (5), (6) and then decoded as a path $\hat{\gamma}_k = \phi^{-1}(\hat{w}_k)$ 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}. 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 $\hat{w}_k$ remains as the median.

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:  $\pi_{\text{estimate}} \leftarrow \hat{\gamma}_0$
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 $\hat{w}_k$ from $y_k$ using (5), and (6)
9:          Compute $\hat{\gamma}_k$ from $\hat{w}_k$ using Algorithm 1
10:         $\pi_{\text{estimate}} \leftarrow \hat{\gamma}_k$
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
```

III. 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.

A. The Simulated Aircraft

We used an high-fidelity flight simulation environment for model aircrafts based on Fs One RC flight simulator [20]. The simulated aircraft was controlled by an autopilot that implemented a receding-horizon linear quadratic regulator to fly over paths specified by the user [21]. We fixed the aircraft's speed to 25 m/s, and the aircraft's altitude to 200 m. The aircraft's camera faced towards the direction of the flight, was inclined 45° 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.

B. 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. 1(b)]. These paths were augmented into the video frames by placing them in a horizontal plane just above the ground-level in the 3-D workspace, and then projecting them onto the 2-D image plane.

C. Configuration of the Interface

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

1) Crossover Probability: Crossover probability was the fraction of input commands that were expected to be corrupted due to noise in decoding EEG signals (Section II-A1b). This probability was estimated by comparing observed input commands with ground-truth input commands at the beginning of each experimental session (see Section IV-D3).

2) Symbol-Length: Symbol-length affected the trade-off between how "expressive" (the degree in which our space of 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 IV-D3). We restricted symbol-length to be between 100 and 500 m, because we assumed that values smaller than 100 m might put an excessive cognitive load on users, and values larger than 500 m might not provide a space of paths expressive enough to accomplish our experimental tasks (see Section IV-A).

3) Alphabet of Symbols: We used the alphabet shown in Fig. 2(a). 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.

4) 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. 6. 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.

5) 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 IV-C).

6) 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 $M \geq 4$.

D. 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 [22]. Measured signals were amplified (James-Long Co.), low-pass filtered, and synchronously sampled at 400 Hz by an IOtech Personal Daq 3000 A/D converter. The classifier was trained using the algorithm in [23], 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 15 Hz, using a hidden Markov model (HMM), to perform classification

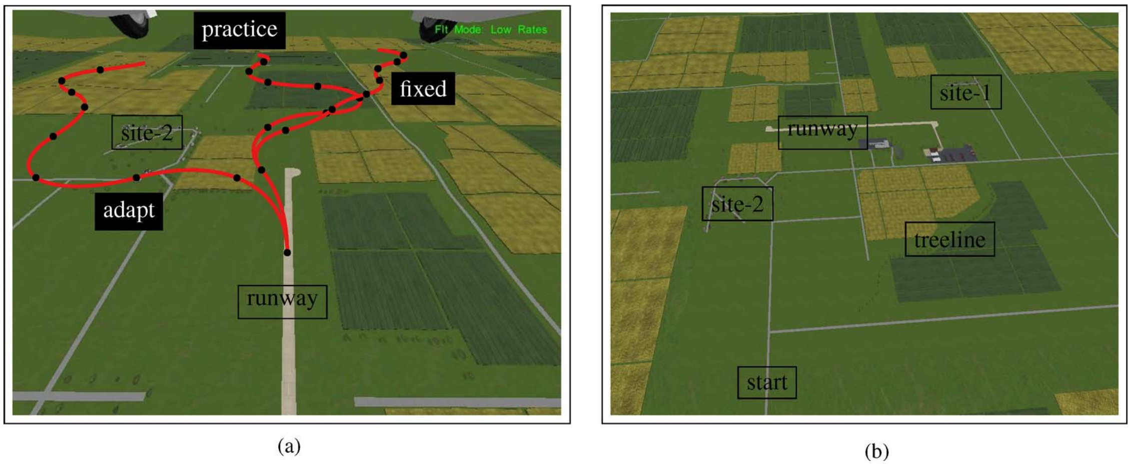


Fig. 5. Illustration of the experimental tasks used in the EEG experiments. The target paths selected for the tracking task in the practice, adapt, and fixed phases were shown in (a), assuming a symbol-length of 330 m. In the high-level task, illustrated in (b), there was no specific target path to track, instead the objective was to fly the aircraft first over the treeline, and then over the house site-1 and site-2. (a) The target paths selected for the tracking task, (b) The target terrain structures selected for the high-level task.

by belief propagation. The decoding occurred asynchronously, i.e., when the belief probability exceeded a threshold.

IV. Experimental Procedure

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

A. Experimental Tasks

We used the following two tasks to evaluate our interface.

1) Tracking Task: The goal was to fly the aircraft over a given target path, which was displayed to the user during flight [see Fig. 1(b)]. 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 3 km/symbol-length (rounded to nearest integer) symbols, hence they were approximately 3 km long, which was chosen to restrict the duration of an experimental run to about 2 min—the time it took the aircraft to fly a 3 km distance.

2) 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. 5(b)]. We said that a house site was hit by a path if the site center was within 250 m of some point along the path, and a treeline was hit by a path if all points along the treeline were within 250 m of some point along the path.

B. 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 500 m, 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.

C. Monte Carlo Experiments

We evaluated the performance of our approach (Section II) and the hybrid approach of [14] under different configurations

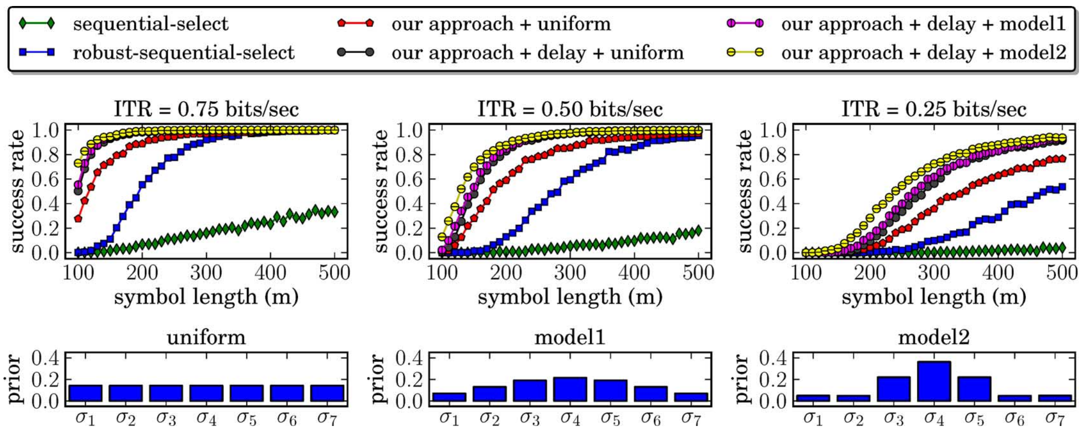


Fig. 6. Results of Monte Carlo experiments comparing different configurations of our approach and the hybrid approach of [14]. The figures at top show the success rate of each method in our navigation task as a function of symbol-length. The figures at bottom show the priors for each statistical model.

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.

1) Sequential-Select: In this configuration, we implemented the hybrid approach of [14] as follows. We allowed the user to specify a desired path symbol by symbol using three inputs $\{"\mathrm{left}," "\mathrm{right}," "\mathrm{rest}"\}$. 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 mth symbol in π_{desired}, and v_0 be a predetermined 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 $\hat{v}_m \in \Sigma$ be the mth symbol chosen so far by the user. At first $\hat{v}_m$ was σ_4. The user provided "left" if $v_m < \hat{v}_m$, "right" if $v_m > \hat{v}_m$, and "rest" if $v_m = \hat{v}_m$. Upon receiving a "left" or "right" input, the interface updated $\hat{v}_m = \sigma_i$ 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 $\hat{v}_m$ when the robot approached the end of v_{m-1}, at which point specification of v_{m+1} started.

2) Robust-Sequential-Select: In this configuration, we attempted to increase the robustness of the hybrid approach of [14] 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 $\hat{v}_m$. In "lock" mode, the interface maintained a posterior probability P over the correct input X. Let Y_k be the random variable denoting the kth observation in the "lock" mode, with $Y_0 = "\mathrm{rest}"$. The interface assumed a uniform prior P_X, and computed the posterior after each Y_k, $k = 0, 1, \ldots$ using Bayes' rule as $P(x \mid y_0, \ldots, y_k) = \eta P_{Y \mid X}(y_k \mid x) P(x \mid y_0, \ldots, y_{k-1})$, where η was a normalizing constant. Upon receiving an observation $y_i = "\mathrm{left}"$ (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 $\hat{v}_m$ accordingly. The "lock" mode was also terminated when the specification of the next symbol started.

3) 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.

4) 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}.

5) 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 III-C and illustrated in Fig. 6.

6) Our Approach + Delay + Model2: In this configuration, our approach used a statistical model, denoted "model2" and illustrated in Fig. 6, 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/s, 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 $P_{Y \mid X}$, which specified the conditional probability of generating Y when the correct input was X, was such that $P_{Y \mid X}(y \mid x) = 1 - E$ if $y = x$, and $P_{Y \mid X}(y \mid x) = E/(n-1)$ 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 100 to 500 m with increments of 10 m. 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.

D. EEG Experiments

We evaluated the performance of our BMI (Section III) 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.

1) 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.

2) 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 100 m 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.

3) 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 III-C), 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 IV-B). In the first run, we set the parameters to their predetermined values (crossover probability was 15% and symbol-length was 337 m). 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 500 m. 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 (100 m) of the symbol-length used in the last run, the subject performed at least three runs, and the subject performed at least five runs if the chosen symbol-length was longer than 400 m. 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.

TABLE I
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

4) 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.

5) 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. 5(a)]. 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.

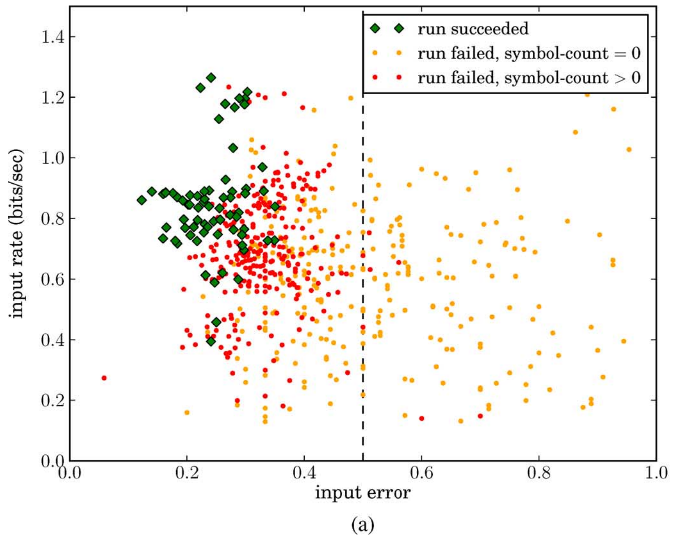
V. Experimental Results

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

A. Results From Monte Carlo Experiments

Results show that our approach outperformed our implementations of the hybrid approach of [14] 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 nonuniform statistical models. Fig. 6 shows the success rates as a function of symbol-length under three different ITRs and Table I 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 70 m



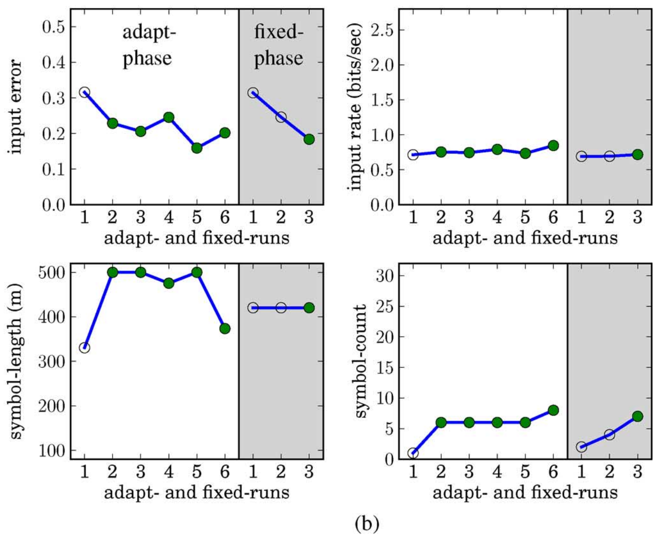


Fig. 7. Results obtained in adapt- and fixed-runs of all runs, shown in (a), and of a sample session, shown in (b). Each marker in (a) denotes the input error, the input rate and the success of a run. In the sample session, the adapt-phase consisted of six runs (unshaded region) and the fixed-phase consisted of three runs (shaded region). Successful runs were denoted with a green solid circle in (b). (a) Results in all adapt- and fixed-runs. (b) Results in adapt- and fixed-phases of a sample session.

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.

B. *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 V-B1), and a low number of successful adapt phases (Section V-B2). Despite these low input performances, half of the subjects succeeded in the fixed phase (Section V-B3), and most of the qualified subjects succeeded in the free phase (Section V-B4).

1) 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(a)]. 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 II 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 three 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 five symbols. The best symbol-count, 13 symbols (each with length 223 m), was achieved by subject A in one of the adapt runs.

2) 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(b) shows the results obtained in the adapt phase of a sample session. Here, the interface was properly configured after six adapt-runs, with crossover probability set to 0.20 (input error observed in the last adapt-run), and symbol-length set to 420 m. On average, in successful adapt-phases, seven adapt-runs were performed, and crossover probability was set to 0.25. In the majority of these phases, the chosen symbol-length was 500 m, the largest value allowed, and in the rest of them, it was 345, 420, 466, 484, 490 m from smallest to largest.

3) 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 II). Fig. 7(b) 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

TABLE II
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		
PQRST8	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

failures in properly-configured fixed-runs could be explained by the same observation.

4) 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 II). Out of 30 free-runs performed in total, 12 runs were successful (hit-count was 3), and in 8 runs hit-count was 2. Fig. 8 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 6.2 km 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.

VI. Conclusion

In this paper, 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 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 100 m, 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.

A. Limitations and Future Work

1) 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.

2) Enabling Navigation Amidst Obstacles: In this paper, 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 [24]. Note that this would assign zero probability to paths that collide with obstacles.

3) 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

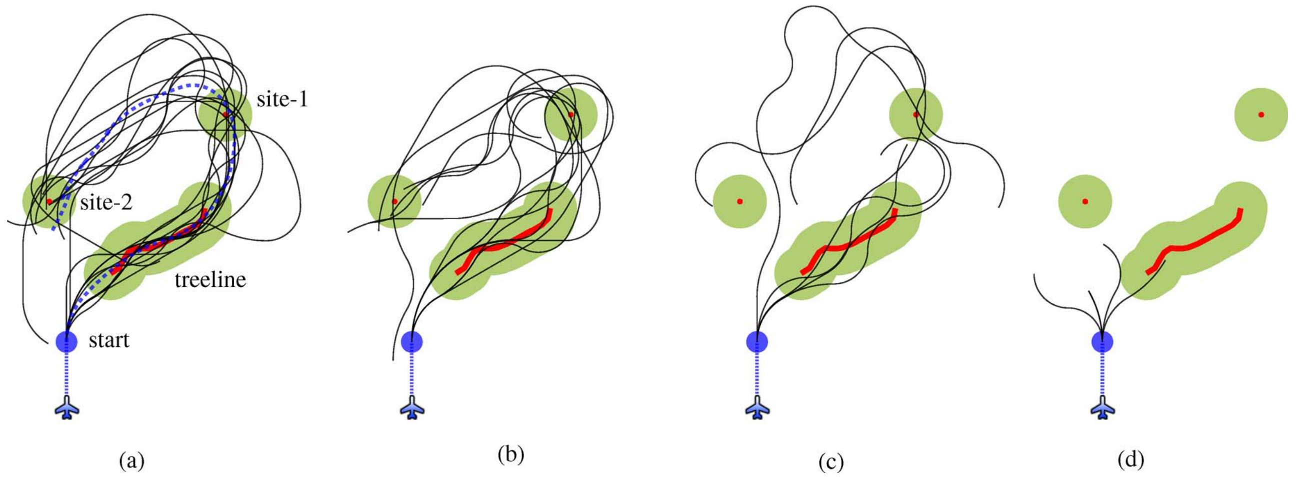


Fig. 8. The flight paths (thin black curves) specified by the subjects in the high-level task, where the objective was to fly the aircraft over three targets: treeline, site-1, and site-2. The frames (a), (b), (c), (d) show the flight paths with hit-counts = 3,2,1,0, respectively. A target was hit by a path if the target center (the red dots for site-1, site-2, and the red curve for treeline) was within 250 m of some point along the path. The frames also show the points within 250 m 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.

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 ith 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.

4) Enabling Navigation in 3-D Space: In this paper, we assumed that the robot was moving in a 2-D space. One way of extending our approach to enable 3-D 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 [25]. 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.

REFERENCES

[1] J. R. Wolpaw, "Brain-computer interfaces as new brain output pathways," *J. Physiol.*, vol. 579, no. 3, pp. 613–619, 2007.

[2] A. Royer and B. He, "Goal selection versus process control in a brain-computer interface based on sensorimotor rhythms," *J. Neural Eng.*, vol. 6, p. 016005, 2009.

[3] 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, p. e42, Oct. 2003.

[4] J. R. Wolpaw and D. J. McFarland, "Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans," in *Proc. Nat. Acad. Sci. USA*, 2004, vol. 101, no. 51, pp. 17 849–17 854.

[5] 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," *Comput. Intell. Neurosci.*, vol. 2007, pp. 3–3, 2007.

[6] 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," *Clin. Neurophysiol.*, vol. 119, no. 9, pp. 2159–2169, 2008.

[7] J. Millan, F. Renkens, J. Mouriño, and W. Gerstner, "Noninvasive brain-actuated control of a mobile robot by human EEG," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 1026–1033, Jun. 2004.

[8] A. Royer, A. Doud, M. Rose, and B. He, "EEG control of a virtual helicopter in 3-dimensional space using intelligent control strategies," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, no. 6, pp. 581–589, Dec. 2010.

[9] X. Perrin, R. Chavarriaga, F. Colas, R. Siegwart, and J. D. R. Millán, "Brain-coupled interaction for semi-autonomous navigation of an assistive robot," *Robot. Auton. Syst.*, vol. 58, pp. 1246–1255, Dec. 2010.

[10] 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 Comput.*, vol. 18, no. 10, pp. 2465–2494, 2006.

[11] 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.

[12] B. Rebsamen, C. Guan, H. Zhang, C. Wang, C. Teo, M. Ang, and E. Burdet, "A brain controlled wheelchair to navigate in familiar environments," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, no. 6, pp. 590–598, Dec. 2010.

[13] 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," *J. Neural Eng.*, vol. 5, no. 2, pp. 214–220, 2008.

[14] I. Iturrate, J. Antelis, A. Kubler, and J. Minguez, "A noninvasive brain-actuated wheelchair based on a P300 neurophysiological protocol and automated navigation," *IEEE Trans. Robot.*, vol. 25, no. 3, pp. 614–627, Jun. 2009.

[15] A. Akce, M. Johnson, and T. Bretl, “Remote teleoperation of an unmanned aircraft with a brain-machine interface: Theory and preliminary results,” in *Proc. 2010 IEEE Int. Conf. Robot. Automat.*, May 2010, pp. 5322–5327.

[16] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, “Brain-computer interfaces for communication and control,” *Clin. Neurophysiol.*, vol. 113, no. 6, pp. 767–791, 2002.

[17] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New York, NY, USA: Wiley-Interscience, Jul. 2006.

[18] I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression,” *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[19] O. Shayevitz and M. Feder, “Optimal feedback communication via posterior matching,” *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1186–1222, Mar. 2011.

[20] FS One V2, Precision RC Flight Simulator Software Developed by InertiaSoft. Champaign, IL [Online]. Available: http://www.fsone.com, Apr. 2012

[21] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, 1990.

[22] P. Nunez and R. Srinivasan, *Electric Fields of the Brain: The Neurophysics of EEG*. New York, NY, USA: Oxford Univ. Press, 2006.

[23] 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 *Proc. 2010 IEEE Int. Conf. Acoust. Speech Signal Process.*, 2010, pp. 602–605.

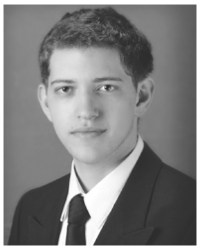
[24] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, “Planning-based prediction for pedestrians,” in *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 3931–3936.

[25] B. O’Neill, *Elementary Differential Geometry*, 2nd ed. New York, NY, USA: Academic, 1997.

Abdullah Akce (S’10) received the B.S. degree in computer engineering from Bogazici University, Istanbul, Turkey, in 2007. He is currently working toward the Ph.D. degree in computer science at the University of Illinois at Urbana-Champaign, Urbana, IL, USA.

His research interests lie at the intersection of robotics, machine learning, and human–computer interaction. As part of his dissertation, he is integrating intent inference and optimal feedback communication into the design of brain–machine interfaces.

Miles Johnson (S’08) received the B.S. degree in mechanical and aerospace engineering from Cornell University, Ithaca, NY, USA, in 2002. Subsequently, he was a member of the Mars Exploration Rover panoramic camera team at Cornell University, performing flight instrument calibration, landed operations, and data analysis. He received the M.S. degree in aerospace engineering, in 2007, from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, where he is currently working toward the Ph.D. degree.



Or Dantsker received the B.S. degree in engineering mechanics from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2012. During his time as an undergraduate student he performed research in the Robotics and Neuro-Mechanical Systems Laboratory under Prof. Timothy Bretl. He is currently a graduate student in the Department of Aerospace Engineering at the University of Illinois at Urbana-Champaign. He is a member of Prof. Michael Selig’s Applied Aerodynamics Group and is a Stillwell fellow. He performs experimental aerodynamics research using unmanned aircraft.

Timothy Bretl (S’02–M’05) received the B.S. degree in engineering and the B.A. degree in mathematics from Swarthmore College, Swarthmore, PA, USA, in 1999, and the M.S. degree in 2000 and the Ph.D. degree in 2005 both in aeronautics and astronautics from Stanford University, Stanford, CA, USA. Subsequently, he was a Postdoctoral Fellow in the Department of Computer Science, also at Stanford University.

Since 2006, he has been with the University of Illinois at Urbana-Champaign, Urbana, IL, USA, where he is an Assistant Professor of Aerospace Engineering with an affiliate appointment in the Coordinated Science Laboratory.

Dr. Bretl was a recipient of the National Science Foundation Faculty Early Career Development Award, in 2010.