A POMDP Approach to P300 Brain-Computer Interfaces*

Jaeyoung Park, Kee-Eung Kim, and Sungho Jo
Department of Computer Science
Korea Advanced Institute of Science and Technology
Daejeon, Korea
jypark@ai.kaist.ac.kr, kekim@cs.kaist.ac.kr, shjo@cs.kaist.ac.kr

Abstract

Most of the previous work on brain-computer interfaces (BCIs) using P300 has been focused on feature extraction and classification algorithms to achieve high performance for the communication between the brain and the computer. While significant progress has been made in such lower layer of the BCI system, the issues in the higher layer have not been addressed sufficiently. Existing P300-based BCI systems use a random order of stimulus sequence for eliciting P300 signal for identifying users' intentions. This paper is about computing an optimal sequence of stimuli in order to minimize the number of stimuli, hence improving the performance. To accomplish this objective, we model the problem as a partially observable Markov decision process (POMDP) with observation delays. Through simulation and human subject experiments, we show that our approach achieves a significant performance improvement in terms of the success rate and the bit rate.

Introduction

A brain-computer interface (BCI) aims to provide a communication channel for conveying messages and commands from the brain to the external system by interpreting brain activities (Wolpaw et al. 2002). There are a variety of devices and methods for BCI, including non-invasive techniques such as encephalography (EEG). The EEG-based BCI is perhaps the most popular method to date, because it is relatively easy and inexpensive to set up the system (Mason et al. 2007). One of the most reliable signal features of EEG is the P300 evoked potential (Krusienski et al. 2008), which is a positive peak in the signal amplitude at about 300ms after a stimulus is given to the user's attention (Farwell and Donchin 1988).

A number of BCI systems using P300 have been proposed in the literature, including the P300 speller (Farwell and Donchin 1988). In the P300 speller system, the user faces a 6×6 matrix of letters and gazes at one of the 36 letters that one desires to select. The 6 letters in a row or a column are flashed (stimulated) together. If the row or the column containing the gazed letter is flashed, P300 is generated at about 300ms later. We can thus train a classifier to detect this P300 to identify the desired letter. Some BCI systems with a smaller stimulus matrix may flash one letter at a time (Bell et al. 2008).

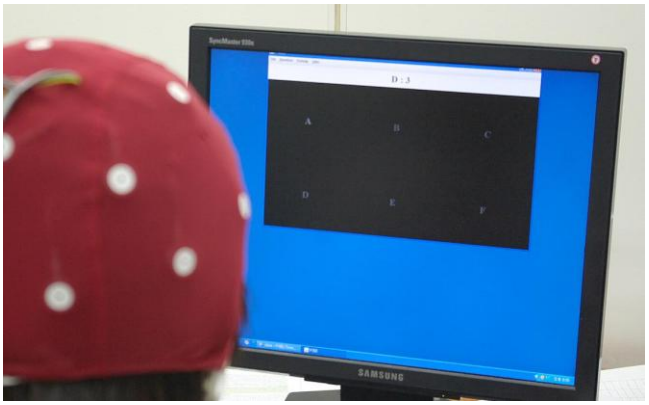

Figure 1. Human sitting in front of P300-based BCI with a [2x3] stimulus matrix.

Figure 1 shows a typical setup of P300-based BCIs. These conventional systems generate flashes in a random order. The P300 speller, for example, generates the same number of flashes for every row and column in a random order. We note that however, by determining the optimal sequence of flashes, we can identify the desired selection using a smaller number of flashes. For example, based on the history of flashes and the P300 detection results, if the probability that the first row contains the desired selection is very low compared to other rows, there is no reason to flash the first row. In contrast, if the probability is high for both the second and the third row, it is desirable to flash the second or the third row in order to resolve the uncertainty. Bell et al. (2008) suggest a high-level idea of maintaining a running sum of P300 classifier output scores and using it for identifying the most likely selection during flashes for reducing the number of flashes.

This paper presents a systematic approach to finding an optimal sequence of flashes in order to identify the desired selection using the fewest number of flashes in P300-based BCIs. Determining the best flash sequence based on the

* This paper has appeared in the Proceedings of ACM International Conference on Intelligent User Interfaces (IUI), 2010.
ICAPS 2010 POMDP Practitioners Workshop, May 12, 2010, Toronto, Canada.

accumulated results of P300 detection is viewed as a sequential decision making problem, and we adopt the partially observable Markov decision process (POMDP) model (Kaelbling et al. 1998) for the representation of the problem. The POMDP model provides a rigorous framework for representing sequential decision making problems under limited sensory capabilities. It perfectly fits our purpose since we have to deal with P300 classifier errors. Although it is computationally infeasible to find an optimal solution from POMDPs, the recent development of fast approximate algorithms such as PBVI (Pineau et al. 2006) and HSVI (Smith and Simmons 2005) has made the POMDP approach practical for a wide variety of real-world applications such as spoken dialogue management (Williams and Young 2007) and assisted daily living (Hoey et al. 2009).

While most of the previous works on EEG-based BCIs have focused on the lower level of the interface such as better feature extraction or classification methods for detecting P300 from EEG, our focus is on finding an optimal sequence of flashes given a P300 classifier, addressing the higher level of the interface, which is an important but currently missing part for an effective BCI.

Electroencephalography (EEG) and P300

In this section, we briefly review some of the knowledge on P300 in EEG and the common settings used in BCIs using EEG.

EEG signals are the electrical signals recoded from the scalp and produced by the electrical activity of neurons in the brain. The electric potentials reflect the summation of the synchronous electrical activity of thousands or millions of neurons that are near the electrode for recording the EEG signals.

Event related potential (ERP) is elicited by an infrequent or particularly significant somatosensory stimulus. P300 is the positive peak component of ERP at about 300ms after the stimulus (Farwell and Dochin 1988, Wolpaw et al. 2002). P300 is known as one of the most reliable signals for composing BCI systems. However, the P300 elicited by the stimulus cannot be obtained easily, because EEG captures the brain activities from numerous sources and the P300 may be buried deeply. Fortunately, there exist several feature extraction and classification methods for detecting the P300 component of ERP. The feature extraction methods include the averaging of EEG signals, the Mexican hat wavelet, and the spatial filter algorithm (Fazel-Rezai and Peters 2005, Ramanna and Fazel-Rezai 2006, Hoffman et al. 2006). Once the relevant features are extracted, classification methods such as Fisher's linear discriminant, stepwise linear discriminant analysis (SWLDA), and support vector machine (SVM) (Krusienski et al. 2006) are used to detect the existence or absence of P300 in the EEG.

BCI systems based on P300 have a typical architecture as shown in Figure 2. There are components for stimulus generation, signal acquisition, preprocessing, and translation. The stimulus generator component gives stimuli to a user to elicit P300 on the desired situation. The signal acquisition component records the EEG signal for the given stimulus. The preprocessing component carries out feature extraction for detecting P300 from the given EEG signal. The translation component classifies the existence of P300, and sends appropriate commands to the external devices.

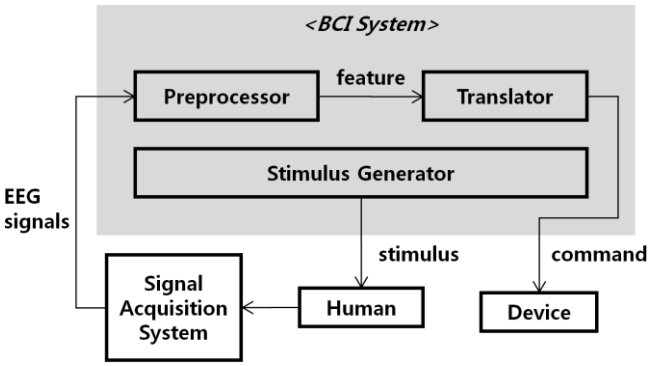

Figure 2. A typical architecture of the P300-based BCIs.

A POMDP Modeling of the P300-Based BCI

Our problem in hand is to find the target letter as accurately as possible while using as the smallest number of flashes as possible. We model this problem using POMDP assuming N letter stimulus matrix with single letter flashing scheme, rather than flashing rows or columns. Conceptually, this problem can be considered as an extension of the tiger problem in the POMDP literature (Kaelbling et al. 1998), where the number of doors matches the number of letters in the stimulus matrix.

A POMDP is defined as 8-tuple $\langle S, A, Z, b_0, T, O, R, \gamma \rangle$ where S is the set of environment states; A is the set of actions available to the agent; Z is the set of all possible observations; b_0 is the initial belief where $b_0(s)$ denotes the probability that the environment starts in state s; T is the transition probability where $T(s, a, s')$ denotes the probability that the environment changes from state s to state s' when executing action a; O is the observation probability where $O(s, a, z)$ denotes the probability that the agent makes observation z when executing action a and arriving at state s; R is the reward function where $R(s, a)$ denotes the reward received by the agent when executing action a in state s; γ is the discount factor such that $0 \leq \gamma \leq 1$.

The states in the POMDP correspond to the target letters, hence a total of N states. For each letter in the matrix, we can either flash it in the hope of detecting P300 (N flash actions) or claim that it's the target letter (N select actions), hence a total of $2N$ actions. The output value from the P300 classifier serves as the observation, where the real value between 0 and 1 is discretized into intervals of size 0.1 (e.g., z_1 for the output value in [0.0, 0.1), z_2 for the output value in [0.1, 0.2), etc.), hence a total of 10 observations.

To make the system identify the target letter as soon as possible, we give -1 reward for the flash actions, +10 re-

ward for the select actions that make a correct claim of the target letter, and -100 rewards for the select actions that incorrectly make a claim on a non-target letter.

We define the transition probabilities for each flash action as identity matrices, assuming that the target does not change to some other letter within a test. Hence, we assign the transition probability of 1 if the state at the current time step is the same as the state at the next time-step, and 0 if the state at the current time step is different from the state at the next time step. The transition probabilities for each select action are defined to be uniform, assuming that the target letter will reset with the same probability between consecutive tests.

The observation probabilities model the errors in the P300 classification results. Specifically, we assume that the classifier output follows the discretized beta distribution when flashing the target letter. The parameters α and β of the beta distribution will be obtained from the training data. We also assume that the distribution of output values when flashing a non-target letter is symmetric to the case when flashing the target letter. Hence, if α_{target} and β_{target} are the parameter values of the beta distribution for flashing the target letter, we set $\alpha_{\text{non-target}} = \beta_{\text{target}}$ and $\beta_{\text{non-target}} = \alpha_{\text{target}}$ for the beta distribution for flashing a non-target letter. Since these parameter values are radically different among the subjects and POMDP algorithms take a significant amount of time to find an optimal policy, we prepared a set of 9 (and 11) POMDP models, each with observation probabilities from different beta distribution parameters for [2x2] (and [2x3]) stimulus matrix. Hence, we pre-compute optimal policies for each POMDP models, determine the most similar model using a short pilot experiment, and select the associated policy for execution. The discount rate is set to 0.99, and the initial belief is set to the uniform distribution.

We briefly explain the behavior of an optimal policy. Each state corresponds to the target letter in the current test, of which the BCI system does not have the direct knowledge. Hence, the system has to infer the target letter by some sequence of flash actions and the corresponding classification output values. When the system flashes a letter and it happens to be the target letter, then the probability is high for the classification output value close to 1. If the letter happens to be a non-target, then the probability is high for the classification output value close to 0. Thus, from a flash action and the corresponding classification output value (i.e., observation), the system can infer the probability distribution on the target letters using the belief state of the POMDP. If a letter has a significant probability to be the target letter, the system repeatedly flashes the letter in an attempt to increase the certainty of the letter being the target. If the probability gets higher than some threshold, the system selects the letter as the target to maximize the expected return.

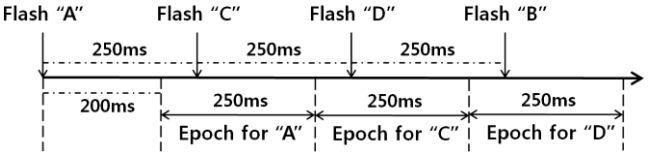

Figure 3. Time course of flash events and the corresponding epochs in EEG signals.

Solving the POMDP model

When we use the POMDP model for P300-based BCI, we have to address two constraints that come from the nature of the P300, namely the delay in P300 and the repetition blindness. We describe how these constraints are handled while implementing the POMDP algorithm.

Delay in P300

The standard definition of POMDPs assumes that the relevant observations are obtained before the execution of the next action. This assumption does not hold in our BCI system. As shown in Figure 3, the relevant P300 epoch ends at 450ms after the flash. Since a small amount of additional delay is incurred by the data acquisition system, the preprocessor and the classifier, the observation is available at almost 510ms after the flash. Hence, the relevant observation is not available throughout the next two actions.

We can handle this constraint by using POMDPs with delayed observations (Bander and White 1999). Solving the model essentially reduces to finding the best action given past action sequence of size equal to the delay in time steps (in our case, the sequences of length 2) during each dynamic programming backup, in contrast to finding the best single action in standard POMDPs without delayed observations.

Repetition blindness

The repetition blindness refers to the situation where P300 may not be elicited when two flashes on the target letter are given within 500ms (Fazel-Rezai 2007, Kanwisher 1987). For example, when the target letter "A" is flashed and the "A" is flashed again within 500ms, the EEG signal corresponding to the second flash may not contain P300. A simple way to avoid this phenomenon is to make sure that the flash is not given on the same letter within the 500ms interval. Since our flash scheme makes two flashes in 500ms, the flash at the current time-step should be different from the previous two flashes. In terms of our POMDP model, the action at the current time step should be different from the previous two actions.

This constraint can be handled by modifying the standard dynamic programming backup operation in POMDP algorithms: when we compute the best action that yields the best value, we only consider the actions that were not executed in the previous two time steps.

Algorithm BCI-PBVI(B, K, D, ϵ)
INPUTS: belief set B; repetition blindness length K; observation delay D; required precision ϵ
for all action sequence $\langle a_1, \ldots, a_D \rangle$ of length D **do**
Initialize $\Gamma_{a_1,\ldots,a_D} = \{\vec{0}\}$
end for
repeat
for all action sequence $\langle a_1, \ldots, a_D \rangle$ of length D **do**
$\Gamma'_{a_1,\ldots,a_D} = \text{backup}(B, K, D, \Gamma, a_1, \ldots, a_D)$
end for
$\delta = \text{difference}(B, \Gamma, \Gamma')$
$\Gamma = \Gamma'$
until $\delta < \epsilon$
return Γ

Figure 4. Top-level pseudocode of BCI-PBVI.

POMDP algorithm for BCI

We now present our implementation of POMDP algorithm that addresses the P300 delay and the repetition blindness. Basically our implementation is a modified version of the PBVI algorithm (Pineau et al. 2006), which we refer to as BCI-PBVI. Figure 4 shows the main loop of the algorithm. As in PBVI, this algorithm requires the set B of randomly sampled belief states (i.e., belief set) for constraining the dynamic programming backup to those belief states (i.e., point-based backup). Whereas the standard PBVI maintains the best α-vector and the corresponding best action for each sampled belief, the BCI-PBVI maintains α-vector for all action sequences of length-D for each sampled belief state. It is similar computing the action-value function rather than the state-value function. This maintenance is necessary in order to prevent the same action being executed within K time steps to handle repetition blindness.

For every possible action sequence of length-D, we update the corresponding set of α-vectors using the point-based backup procedure. Figure 5 shows the pseudocode of the procedure, and it is the central part of our algorithm. Assuming that the current time step is T, we first compute the set A_{allow} of actions that are allowed to execute at time step $T + D$, i.e., the set of actions except those appearing in the last K steps in the sequence. Among the set of α-vectors that are computed in the previous iteration, only those with action sequences ending with allowed actions are valid α-vectors for the backup. The first loop in the pseudocode carries out this task. The second loop performs the actual point-based backup on the belief set B. Given a belief state b, we choose α_b among $\alpha_1, \ldots, \alpha_D$ such that $\sum_s b(s)\alpha_b(s)$ is the maximum expected value gathered *after D-steps in the future*, if, starting from the current belief state b, we execute the action sequence $a_1,\ldots,a_D$. Cautious readers may question that α_b should also consider the rewards gathered *within D-steps in the future*, but as we will see shortly, it is not necessary to do so.

Belief update. Assume that, at time-step t, we have executed an action sequence $a_{t-D},\ldots,a_{t-1}$, and deciding which action a_t to execute. Since α-vectors in BCI-PBVI do not reflect the rewards gathered during the last D-steps, we cannot use the current belief state b_t. Instead, we need the belief state b_{t-D} of D-steps in the past, and this is the belief state we maintain while executing the policy.

Once we execute a_t and observe z_t, the belief state b_{t-D} is updated by

$$b_{t-D+1}(s') = \frac{O(s', a_{t-D}, z_t) \sum_s T(s, a_{t-D}, s') b_{t-D}(s)}{P(z_t | b_{t-D}, a_{t-D})}$$

Algorithm backup($B, K, D, \Gamma^{t-1}, a_1, \ldots, a_D$)
INPUTS: belief set B; repetition blindness length K; observation delay D; set of α-vectors for (t-1)-step value function Γ^{t-1}; the action sequence of interest $\langle a_1, \ldots, a_D \rangle$
$A_{\text{allow}} = A - \{a_{D-K+1}, \ldots, a_D\}$
for all action $a \in A_{\text{allow}}$ **do**
for all observation $z_D \in Z$ **do**
for all α-vector $\alpha_i \in \Gamma^{t-1}_{a_2,\ldots,a_D,a}$ **do**
$a_i^{a,z_D}(s) = \gamma \sum_{s'} T(s, a_1, s') O(s', a_1, z_D) \alpha_i(s'), \quad \forall s \in S$
end for
$\Gamma^{t,a,z_D}_{a_1,\ldots,a_D} = \bigcup_i \{\alpha_i^{a,z_D}\}$
end for
end for
for all belief state $b \in B$ **do**
for all action $a \in A_{\text{allow}}$ **do**
$\alpha_b^a = \sum_{z \in Z} \text{argmax}_{\alpha \in \Gamma^{t,a,z}_{a_1,\ldots,a_D}} \sum_{s' \in S} b(s')\alpha(s')$
$\alpha_b^a(s) = \alpha_b^a(s) + \sum_{s_1 \in S, \ldots, s_D \in S} T(s, a_1, s_1) T(s_1, a_2, s_2) \cdots T(s_{D-1}, a_D, s_D) R(s_D, a), \forall s$
end for
$\alpha_b = \text{argmax}_{\alpha_b^a, a \in A_{\text{allow}}} \sum_{s \in S} b(s) \alpha_b^a(s)$
$\Gamma^t_{a_1,\ldots,a_D} = \Gamma^t_{a_1,\ldots,a_D} \cup \{\alpha_b\}$
end for
return $\Gamma^t_{a_1,\ldots,a_D}$

Figure 5. The point-based backup operator for BCI-PBVI.

where $P(z_t|b_{t-D}, a_{t-D})$ is the normalizing constant. Hence, in order to perform appropriate belief update, we need to remember the past belief state b_{t-D} as well as the action sequence $a_{t-D},\ldots,a_{t-1}$.

Optimal action selection. At this point, selecting the optimal action for execution is quite straightforward. Assume once again that, at time-step t, we have executed an action sequence $a_{t-D},\ldots,a_{t-1}$, and deciding on which action a_t to execute. Since we have the belief state b_{t-D} at hand, we compute $\sum_s b_{t-D}(s)\alpha(s)$ for $\forall\alpha \in \Gamma_{a_{t-D},\ldots,a_{t-1}}$, and execute the best action associated with the α-vector that yields the best value.

Experiments

We conducted experiments comparing the performances of the conventional BCI with random flash sequence and the proposed BCI with optimal flash sequence. In this section, we describe the experimental setup, the metrics for performance measurement, and the results from both simulations and human subjects.

Methods

Baseline. The baseline method follows a flash sequence decided by random order and hence it is equivalent to the method used in traditional BCI systems including Bell et al. (2008): the flash sequence is randomized uniformly among the letters that are not flashed within the current trial, where the *trial* refers to a subsequence of length equal to the letters in the matrix, hence each letter is flashed once per trial. However, our baseline method additionally takes repetition blindness into account: if a letter was flashed within 500ms, it will not be considered as a candidate for the current flash. The decision is made by the total score from the classifier: the output value of the classifier is regarded as the posterior probability, and the letter with the largest sum throughout trials is selected as the target letter. Hence, the method can be stopped at the end of any trial during a test, and determine the most likely target letter. The total score of a letter can be regarded as the posterior probability of being the target letter given the history of flashes and classifier output values. Note that the baseline method has no explicit "stop and select" decision making mechanism.

POMDP with select actions (PWSA). The PWSA method uses the optimal flash policy computed from POMDP. The select actions represent the explicit decision making mechanism.

POMDP without select actions (PWOSA). The PWOSA method uses the same set of actions as PWSA except the select actions. When stopped, we determine the target letter with the highest belief state probability. We prepared this method for the sake of performance comparison with the baseline method when the same number of flashes is used. Ideally, this method will be more efficient than the baseline method in terms of the number of flashes because the flash sequence is determined by the POMDP policy, rather than by some random distribution. Conceptually, this method can be considered as an optimal policy from a POMDP model with negative infinite rewards for selecting an incorrect target letter.

Measurements

Success rate. The success rate is defined by the percentage of tests in the simulation with correct identification of the target letters. We stopped the methods at the end of each trial (4 flashes for [2x2] and 6 flashes for [2x3]), and measured the success rate. Note that the PWSA method can terminate before the specified number of flashes. In this case, we extended the results until the end of tests. For example, if the PWSA method terminated during the 5th trial with an incorrect target letter, the method is regarded as selecting an incorrect target letter for all subsequent trials, and vice versa. For all three methods, if the method has the same total score or the same maximum belief value for K letters, we gave a partial success of $1/K$.

Bit rate. The bit rate represents the quantity of transferred information per unit time during communication (Serby et al. 2005, Wolpaw et al. 2000). The bit rate is defined as $B \cdot D$, where B is the number of bits per decision and D is the number of decisions per unit time. Let N be the total number of letters in the matrix and P be the success rate. Then we have

$$B = \log_2 N + P\log_2 P + (1-P)\log_2 \frac{1-P}{N-1}.$$

In order to measure D, we used the following scheme: For the baseline method, since the decision has to be delayed (260ms) until the result of the last flash is available, we add the delay to the time spent on flashes. For example, if we have flashed 40 times, then the total time spent for the decision is calculated as 250ms * 40 flashes + 260ms. For the PWSA and PWOSA methods, since it takes an additional 115ms delay for updating the belief state, we add a delay of 375ms to the time spent on flashes. For example, if we have flashed 40 times, the total time for the decision is 250ms * 40 flashes + 375ms. D is calculated as the reciprocal of the total time.

Since the bit rate changes depending on the success rate, we calculated the bit rates for different success rates. The tradeoff here is that we can improve the success rate by increasing the number of flashes, but more time is accordingly spent per decision. For the PWSA method, since it has its own explicit termination mechanism, we will report only one value for the bit rate measurement.

Simulation Experiments

We performed 20 simulations on [2x2] and [2x3] matrices, each simulation consisting of 10000 tests. Output values from the classifier (i.e. observations) are sampled from the beta distribution with parameters $\alpha_{\text{target}} = \beta_{\text{non-target}} = 1.228$ and $\beta_{\text{target}} = \alpha_{\text{non-target}} = 0.625$, which were obtained from the pilot experiment involving one of the human subjects.

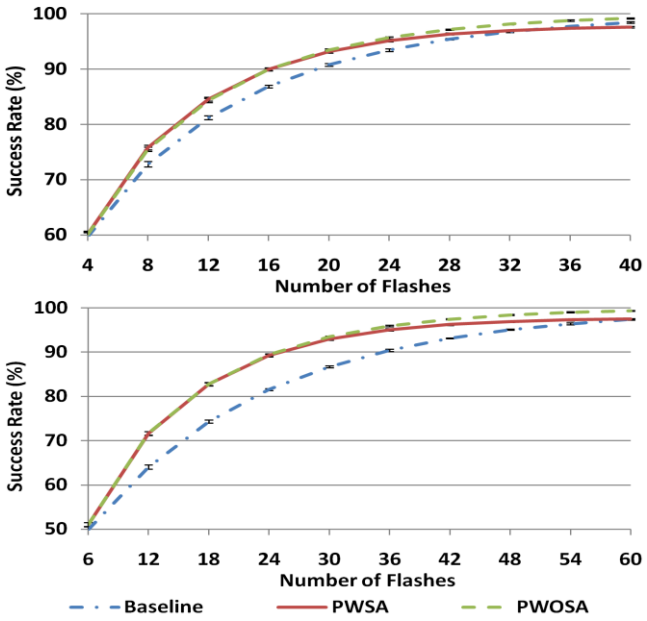

Figure 6. The success rate results of the simulation experiments. The top graph shows the result on [2x2] matrix and the bottom graph shows the result on [2x3] matrix.

Each test consists of 40 flashes for the [2x2] matrix and 60 for the [2x3] matrix. For the baseline method, these numbers correspond to 10 observations for each letter in the matrix, whereas the PWSA and PWOSA methods may have different number of observations for each letter. Note also that the baseline and PWOSA methods run until the test ends, whereas the PWSA method can terminate early when the final select action is executed.

Figure 6 shows the success rate results for the three methods. The performance gap between the baseline method and the POMDP methods is larger in the [2x3] matrix than in the [2x2] matrix. We conjecture that the performance gap will become even larger when we experiment on larger stimulus matrices. The baseline and the PWOSA methods will converge to a 100% success rate as the number of flashes goes to infinity. In contrast, the success rate of the PWSA method doesn't due to the bias inherent in the reward function. However, having an infinite number of flashes is not a practical assumption, and this kind of bias is necessary if we ever want the method to have some explicit termination mechanism. In our experiments, the PWSA method converges to a success rate close to 100% very quickly when forced not to terminate before the specified number of flashes, which is sufficient to demonstrate the validity of our approach.

Figure 7 shows the bit rate results. The PWOSA method is always significantly better than the base line method. The PWSA method yields the success rates of 0.980 for [2x2] and 0.977 for [2x3], which correspond to 23.840 bits/min and 23.080 bits/min respectively. Table 1 summarizes the bit rate results of the three methods. Comparing the bit rates at the best achievable success rates, the PWSA method improves the bit rates by 220%~249%. Comparing

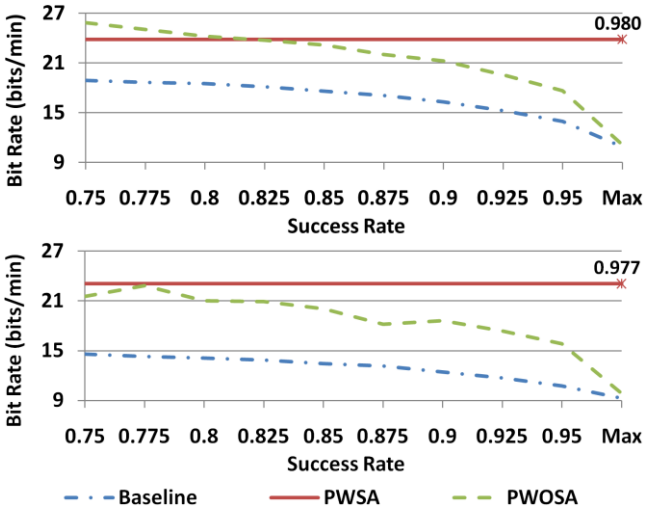

Figure 7. The bit rate results of the simulation experiments. The top graph shows the results on [2x2] matrix and the bottom graph shows the results on [2x3] matrix.

	[2x2] Matrix	[2x3] Matrix
Baseline†	18.879 (75.0%)	14.612 (75.0%)
Baseline‡	10.867 (98.4%)	9.257 (98.4%)
PWSA	23.840 (98.0%)	23.080 (97.7%)
PWOSA†	25.830 (75.0%)	22.859 (77.5%)
PWOSA‡	11.102 (99.2%)	9.820 (99.2%)

Table 1. Bit rate results of simulation experiment for each system. '†' denotes for maximum bit rate and '‡' denotes the bit rate on maximum success rate. The percentage in parenthesis is the corresponding success rate.

to the best bit rates achievable by the baseline method[1], the PWSA method achieves improvements of 126% to 158%.

Human Subject Experiments

We compared the performances of the baseline and PWSA methods on the [2x2] and [2x3] matrices for the human subjects. Each test randomly assigned a target letter, while making sure that each letter was selected as the target letter exactly 10 times for the [2x2] matrix and 5 times for the [2x3] matrix. Hence, we performed a total of 40 tests for the [2x2] matrix and 30 tests for the [2x3] matrix. As in the simulation experiments, each test consisted of 40 flashes for the [2x2] matrix and 60 for the [2x3] matrix.

[1] The baseline and PWOSA methods can achieve higher bit rates by lowering the success rate, but we set the minimum to 75% since we also want a sufficiently high success rate.

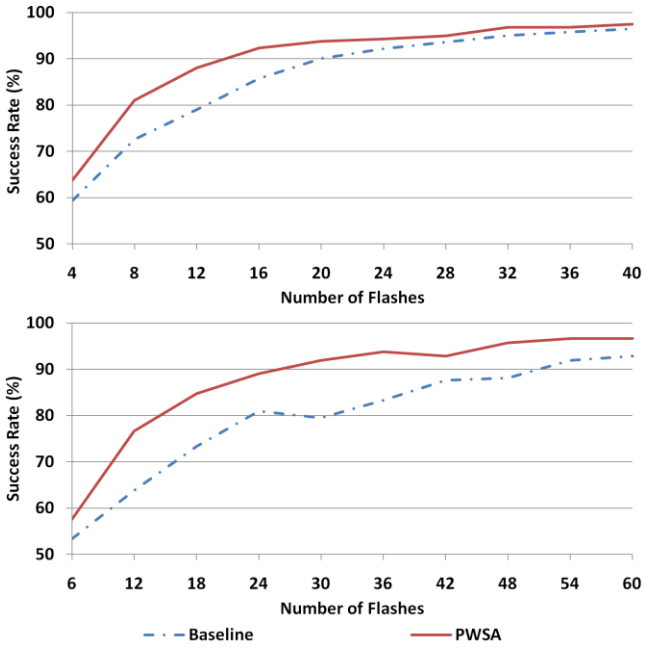

Figure 8. The success rates from the human subject experiments. The top graph shows the result on [2x2] matrix, and the bottom graph shows the result on [2x3] matrix.

We first prepared a number of different POMDP models with varying observation probabilities, since the classifier showed different error rates depending on the human subject. By varying the parameters of the beta distribution, we obtained 9 different models of the [2x2] matrix, and 11 for the [2x3] matrix. We pre-computed the optimal policy for each model, since our implementation of the POMDP algorithm currently takes hours to finish. This is due to the fact that the point-based backup requires enumerating all possible action sequences of length D. Further optimization via pruning useless action sequences is left as a future work. We used 1028 randomly selected belief states for the [2x2] matrix, and 1030 for the [2x3] matrix.

At the onset of the experiment for each subject, we carried out a short pilot experiment where we gathered the training data for the preprocessor and the classifier. Once they were trained, we performed cross-validation evaluation, chose the POMDP model with the minimum KL-divergence, and used the corresponding optimal POMDP policy.

We originally involved 9 human subjects, but 2 of them had beta distributions very far from any of the pre-computed models. Hence we use the data from 7 human subjects.

Figure 8 shows the success rate results of the human experiments. The success rate for the PWSA method is higher than the baseline method on any number of available flashes and the performance gap becomes larger as the matrix gets larger, which is consistent with the results from the simulation experiments.

Figure 9 shows the bit rate results. The PWSA method yielded an average success rate of 98.2% for the [2x2] matrix and 97.6% for the [2x3] matrix. The corresponding bit rate is 24.368 bits/min for the [2x2] matrix, and 21.367 bits/min for the [2x3] matrix. In the case of the baseline method, we can control the success rate by changing the number of flashes, and the maximum average success rates are 96.4% for the [2x2] matrix and 92.9% for the [2x3] matrix. Regardless of how we set the success rate for the baseline method, the PWSA yielded higher bit rates. The summary of the bit rate results is shown in Table 2. Compared to the bit rates at the best achievable success rates, the PWSA method improves the bit rates by 242% to 265%. Compared to the best bit rates achievable by the baseline method, the PWSA method achieves improvements of 135%~151%. These results are consistent to those from the simulation experiments.

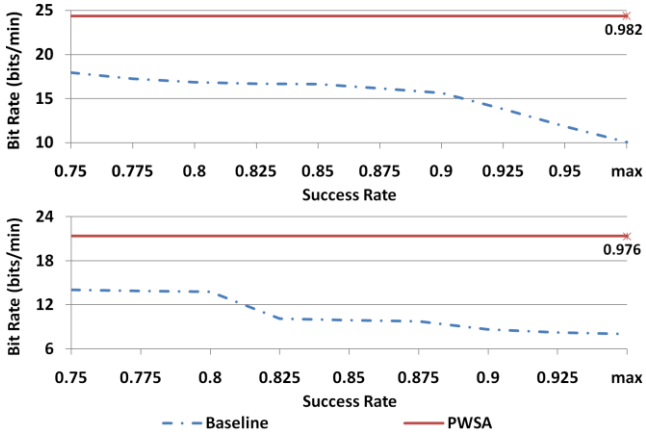

Figure 9. The bit rate results from the human subject experiments. The top graph shows the result on [2x2], and the bottom graph shows the result on [2x3].

	[2x2] Matrix	[2x3] Matrix
Baseline†	17.951 (75.0%)	14.070 (75.0%)
Baseline‡	10.065 (96.4%)	8.052 (92.9%)
PWSA	24.368 (98.2%)	21.367 (97.6%)

Table 2. Bit rate results on human experiment for each system. '†' denotes for maximum bit rate and '‡' denotes the bit rate on maximum success rate. The percentage in parenthesis is the corresponding success rate.

Discussion

In this paper, we have presented a P300 BCI system that uses POMDP for calculating the optimal flash sequence. In contrast to the previous body of research that concentrate on obtaining better feature extraction and classification algorithm from the raw EEG signals, our work provides a unified framework for building the BCI system. Bell et al. (2008) have roughly suggested the idea of using the confidence values from the P300 classifier for optimizing the

flash sequence, but to the best of our knowledge, our work is the first to address the problem in a principled way.

The contributions of this paper are as follows: First, we provided a formal decision-making model for P300-based BCI. Specifically, we showed how the POMDP model with observation delays can be adapted to BCI. Although we explained in the context of P300-based BCI systems, we believe that our approach is general enough to be applied to other BCI paradigms. Second, we presented a novel point-based algorithm for solving POMDPs with observation delays. The algorithm extends the standard point-based backup operator to handle observation delays. Third, we report experimental results using simulation as well as human subjects. Our POMDP-based system achieves significant performance improvement over the baseline method currently used in other BCI systems.

Currently, we are working on improving the speed of the algorithm for POMDPs with observation delays. One of the most time-consuming aspects of our algorithm is in the enumeration of all possible action sequence of length equal to the delay. Since some action sequences may be inferior to others, a combination of forward search and dynamic programming may yield substantial improvement in the speed. We are also working on applying the technique to P300 speller, where the user intentions exhibit more regularity. We strongly believe that we can achieve a magnitude of order improvement in performance (bit rate) if we embed the bigram/trigram model of alphabets into the intention-level transition probability of the POMDP. Finally, we are investigating into the methods (Doshi et al. 2008) that enable the adaption of model to individual subjects without explicit pilot trial experiments or pre-computing optimal policies by enumerating candidate models.

References

Bander, J.L., and White Ⅲ, C.C. 1999. Markov decision processes with noise-corrupted and delayed state observations. *J. Operational Research Society 50*.

Bell, C.J.; Shenoy, P.; Chalodhorn, R.; and Rao, R.P.N. 2008. Control of a humanoid robot by a non-invasive brain-computer interface in humans. *J. Neural Eng. 5*.

Doshi, F.; Pineau, J.; and Roy, N. 2008. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. *Int. Conf. on Machine Learning*.

Farwell, L. A., and Donchin, E. 1988. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalogr. Clin. Neurophysiol 70*.

Fazel-Rezai, R. 2007. Human error in P300 speller paradigm for brain-computer interface. *Proc. 29th Annual Int. Conf. of the IEEE EMBS*.

Fazel-Rezai, R., and Peters, J. F. 2005. P300 wave feature extraction: preliminary results. *Proc. 18th Annual Canadian Conf. on Electrical and Computer Eng*.

Hoey J.; Poupart, P.; von Bertoldi, A.; Boutilier, C.; and Mihailidis, A. 2009. Automated handwashing assistance for persons with dementia using video and a partially observable Markov decision process. *Computer Vision and Image Understanding*.

Hoffmann, U.; Vesin, J. M.; and Ebrahimi, T. 2006. Spatial filters for the classification of event-related potentials. *Proc. 14th ESANN*.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence 101*.

Kanwisher N. G. 1987. Repetition blindness: Type recognition without token individuation. *Cognition 27*.

Krusienski, D. J.; Sellers, E. W.; Cabestaing, F.; Bayoudh, S.; McFaland, D. J.; Vaughan, T. M. and Wolpaw, J. R. 2006. A comparison of classification techniques for the P300 Speller. *J. Neural Eng. 3*.

Krusienski, D. J.; Sellers, E. W.; McFaland, D. J.; Vaughan, T. M.; and Wolpaw, J. R. 2008. Toward enhanced P300 speller performance. *J. of Neuroscience Methods 167*.

Mason, S. G.; Bashashati, A.; Fatourechi, M.; Navarro, K. F.; and Birch, G. E. 2007. A comprehensive survey of brain interface technology designs. *Annals of Biomedical Engineering 35*, 2.

Pineau, J.; Gordon, G.; and Thrun, S. 2006. Anytime point-based approximations for large POMDPs. *J. Artificial Intelligence Research 27*.

Ramanna, S., and Fazel-Rezai, R. 2006. P300 wave detection based on rough sets. *Lecture Notes in Computer Science 4100*, Springer.

Serby, H.; Yom-Tov, E.; and Inbar, G. F. 2005. An improved P300-based brain-computer interface. *IEEE Trans. Neural. Syst. Rehabil. Eng. 13*.

Smith, T., and Simmons, R. 2005. Point-based POMDP algorithms: Improved analysis and implementations. *Proc. 21st Conf. on Uncertainty in Artificial Intelligence*.

Williams, J. D., and Young, S. 2007. Partially observable Markov decision processes for spoken dialog systems. *J. Computer Speech and Language 21*.

Wolpaw, J. R.; Birbaumer, N.; Heetderks, W. J.; McFarland, D. J.; Peckham, P. H.; Schalk, G.; Donchin E.; Quatrano, L. A.; Robinson C. J.; and Vaughan, T. M. 2000. Brain-computer interface technology: A review of the first international meeting, *IEEE Trans. Rehabil. Eng. 8*.

Wolpaw, J. R.; Birbaumer, N.; McFarland, D. J.; Pfurtscheller, G.; and Vaughan, T. M. 2002. Brain-computer interfaces for communication and control. *Clin. Neurophysiol. 113*.