An Adaptive Human Brain to Computer Interface System for Robotic or Wheel-Chair based Navigational Tasks

DULLAL GHOSH

KTH VETENSKAP OCH KONST

KTH Industrial Engineering and Management

Master of Science Thesis
Stockholm, Sweden 2012

An Adaptive Human Brain to Computer Interface System for Robotic or Wheel-Chair based Navigational Tasks

Dullal Ghosh

Master of Science Thesis MMK 2012:75 MDA 445
KTH Industrial Engineering and Management
Machine Design
SE-100 44 STOCKHOLM

Examensarbete MMK 2012:75 MDA 445

Ett adaptivt gränssnitt mellan hjärna och datorsystem för navigering av robot eller rullstol

Dullal Ghosh

Godkänt	Examinator	Handledare
2012-11-05	Jan Wikander	Mats Hanson
	Uppdragsgivare	Kontaktperson
	Technical University of Munich, Germany	Mohammad Abu-Alqumsan / Angelika Peer

Sammanfattning

Ett av flera huvudsyften med ett gränssnitt mellan hjärna och datorer är att förbättra livskvaliteten för förlamade personer. I tidigare studier har betydande ansträngningar gjorts för att känna igen en användares avsikter. Dock finns inga tidigare studier som visar varför detta är viktigt. De system som idag ligger i framkant gällande gränssnittet mellan hjärna och dator har dessvärre låg bandbredd vilket leder till en hög kognitiv belastning på användaren. Ett gränssnitt mellan hjärna och dator som är designat med dynamisk anpassning kan minska ansträngningen som krävs av användaren för att använda systemet samt ge denna en förkroppsligande känsla. Utgående för detta försöker denna rapport att utveckla ett intelligent system för gränssnittet mellan hjärna och användare. Detta system baseras på lärandet av en användares beteende samt att förutspå användarens intention när det kommer till att navigera en semiautonom robot. Vidare har arbete lagts på att modellera en användare samt att bygga en robot för att träna den utvecklade algoritmen. Simulerade tester av den utvecklade algoritmen bekräftar möjligheterna att använda ett sådant användare anpassat system för interaktionen mellan hjärna och dator.

Även med en adaptiv gränssnitt mellan hjärna och dator så är systemet utsatt för fel på grund av felaktig klassificering av användarens avsikt. Detta fel kan leda till felaktigt beteende hos en robot eller felaktig förflyttning av en rullstol för en funktionshindrad person. Flera fysiologiska studier har visat att fel-relaterade potentialer framkallas i hjärnan när ett fel begås av ett system som agerar som gränssnitt mellan människa och maskin. Erkännande av fel-relaterade potentialer i de inspelade elektroencefalografi data kan utnyttjas för att förhindra felfortplantning till genomförande och därmed fungera som en kontrollmekanism för att känna igen användarens intention. Således i den sista delen fokuserar denna uppsats på att utveckla en klassificerare för detektering av fel-relaterade potentialer genom användning av Support Vector Machine och resultatet visar jämförbar klassificerings prestanda för Support Vector Machine som för de bästa Gauss klassificerare.

KTH Industrial Engineering and Management

Master of Science Thesis MMK 2012:75 MDA 445

An Adaptive Human Brain to Computer Interface System for Robotic or Wheel-Chair based Navigational Tasks

Dullal Ghosh

Approved 2012-11-05	Examiner Jan Wikander	Supervisor Mats Hanson
	Commissioner Technical University of Munich, Germany	Contact person Mohammad Abu-Alqumsan / Angelika Peer

Abstract

Among several, one of the main goals of Brain-computer interface (BCI) system is to improve the quality of life for paralyzed persons. While significant effort has been made to recognize user intention, the necessity of predicting user intention in the context of BCI for navigational tasks to design a dynamic interface, has not been addressed yet. State-of-the-art BCI system has low bandwidth because of which the user is subjected to much cognitive or interaction load. However a BCI system designed with dynamic customization feature to adapt as per individual user, would indeed reduce the interaction load and provide embodiment feeling to the user. Therefore this thesis attempts to develop an intelligent BCI system based on reinforcement learning approach to learn the user behaviour and predict the intentions in the context of a semi-autonomous robotic navigational task.

In addition, an effort has also been made towards modeling a user or to build an agent, in order to train the developed algorithm. Simulated testing of the developed algorithm confirms the potential of using such a user adapted brain-computer interface system.

Even with an adaptive brain-computer interface, the system is prone to errors due to misclassification of the user's intention. Propagation of this classification error could lead to wrong execution of a robotic avatar or movement of a wheel-chair in use by a paralyzed person. Several physiological studies have shown that error-related potentials are elicited in the brain when an error is committed by the BCI system. Recognition of error-related potentials in the Electroencephalography recorded data could be exploited to prevent error propagation till execution and hence act as a verification mechanism for the classifier devoted for recognizing user intention in the context of BCIs. Thus in the last part, this thesis focuses on developing a classifier for detection of error-related potentials using Support Vector Machine and the result shows comparable classification performance of Support Vector Machine to that of state-of-the-art classifiers such as Gaussian classifier and Bayesian filter method.

FOREWORD

Here I would like to thank all those who have contributed with their help, guidance, motivation and assistance during the thesis work.

This master thesis work has been carried out on behalf of KTH Royal Institute of Technology, Stockholm, Sweden at Technical University of Munich, Germany as an Erasmus Mundus Exchange program and is a part of VERE (Virtual Embodiment and Robotic Re-Embodiment) European Program.

First, I would first like to express my appreciation to my direct supervisor M.Sc. Mohammad Abu-Alqumsan at TU Munich, for his support, guidance and help in achieving the goals of this thesis. I would also like to thank Professor Mats Hanson for supervising me on behalf of KTH and for his time to guide me during this thesis report writing. Thanks to Dr. Angelika Peer at TU Munich for reviewing my work and suggesting improvements during several presentations. I am really grateful to KTH Erasmus Coordinator Anna Hellberg Gustafsson for nominating and providing me the opportunity to pursue the thesis at TU Munich as an Erasmus Mundus scholar.

Thanks to my fellow colleagues Rigas-Georgios Zapounidis, Akhter Jamil, Abhilash Babu, Rickard Nilsson, Martin Andersson and Ertan Kayan for their company and motivation throughout the thesis period. Thank you Aditya Ghantasala and Áron Cserkaszky for being there with me during the thesis.

Last but not the least I would like to thank my Parents for their love, continuous motivation and moral support throughout the Master program. Thank you Mom and Dad…

Dullal Ghosh

Stockholm, November 2012

NOMENCLATURE

Abbreviations

AUI	Adaptive User Interface
BCI	Brain-Computer Interface
BMI	Brain-Machine Interface
EEG	Electroencephalography
ERP	Event-related Potentials
ErrP	Error-related Potentials
IErrP	Interaction Error-related Potentials
KKT	Karush-Kuhn-Tucker
LS	Least Square
MDP	Markov Decision Process
noErrP	no Error-related Potentials
POMDP	Partially Observable Markov Decision Process
QP	Quadratic Programming
RBF	Radial Basis Function
RL	Reinforcement Learning
ROS	Robot Operating System
SMO	Sequential Minimal Optimization
SNR	Signal-to-Noise Ratio
SSVEP	Steady State Visually Evoked Potential
SVM	Support Vector Machine

TABLE OF CONTENTS

SAMMANFATTNING (SWEDISH) 1
ABSTRACT 3
FOREWORD 5
NOMENCLATURE 7
TABLE OF CONTENTS 9
1 INTRODUCTION 11
1.1 Brain-Computer Interface 11
1.2 Brain-Computer Interface Paradigms 12
1.3 Problem Description 13
1.4 Report Structure 15
2 PART ONE (ADAPTIVE BCI) 16
2.1 State-of-the-Art Adaptive Interface 16
2.2 Brain-Computer Interface Manager 17
2.2.1 Adaptive Interface with Dynamic Policy 18
2.3 Markov Decision Process 18
2.3.1 Learning User Behaviour and Prediction of User Intention 20
2.3.2 Learning Behaviour 20
2.3.3 Change in Behaviour 22
2.3.4 Outlier in Behaviour 23
2.3.5 Prediction of User Intention 24
2.4 Training of the Algorithm Parameters and Testing 25
2.4.1 Fixed Policy Interface 27
2.4.2 Adaptive Policy Interface 27
2.4.3 Results of Fixed and Adaptive Policy Interfaces 28
2.4.4 Extension of the Learning Scheme 31
2.5 Future work on Adaptive Brain-Computer Interface 31
3 PART TWO (CLASSIFICATION OF ERROR-RELATED POTENTIALS) 32
3.1 State-of-the-Art Classification of Error-related Potentials 32
3.2 Support Vector Machine 33
3.3 Data Set Used for Classifier 34
3.4 Feature Extraction and Training of Classifier 35
3.4.1 Method of Feature Extraction 35
3.5 Support Vector Machine Classifier Training 37
3.6 Classification Performance of Support Vector Machine 38
3.6.1 Timing Performance of Support Vector Machine Classifier 39
3.7 Future Work on Support Vector Machine Classification Method 39
4 SUMMARY 40
5 LIST OF FIGURES 41
6 REFERENCES 42
APPENDIX A: Markov Agent for Training and Testing of the Algorithm 46
B: Parameters for Support Vector Machine Classification 47

1 INTRODUCTION

In this chapter a brief introduction of brain-computer interface, brain-computer interface paradigms or techniques, problem description and report structure have been presented.

1.1 Brain-Computer Interface

Communication with patients suffering from paralysis is an unsolved challenge. As per statics, the number of paralyzed persons goes up to 6 million in United States alone, which means 1 in every 50 people living with paralysis[1]. "*Movement restoration for patients with chronic stroke or other brain damage also remains a therapeutic problem and available treatments do not offer significant improvements*" [1]. The quality of life of paralyzed persons could be improved by virtue of a technique known as Brain-Computer Interface (BCI), "*often called a mind-machine interface, or sometimes called a direct neural interface or a brain–machine interface, which is a direct communication pathway between the brain and an external device*" [2]. For many decades, instrument control by just thinking, using brain waves has been accepted in science fiction. However, it is only in the last few years that these systems have been shown to be feasible in laboratories [3]. An example of a person operating a robotic arm through brain waves is shown in Figure 1. In general there could be invasive or non-invasive BCIs. Invasive BCI needs electrodes to be inserted into the brain tissue [3]. In contrary to above BCI category, non-invasive BCI avoids surgery and rather uses an Electroencephalographic (EEG) cap by placing electrodes on the scalp as in Figure 2. EEG measures voltage fluctuations resulting from ionic current flows within the neurons of the brain. Among several applications of non-invasive BCI, one is where paralyzed patients can spell out words and form sentences using their thoughts alone. Some other applications may include the ability of a user to control a video game by thought [4], virtual key board control, cursor control or for neuroprosthetics applications. In the context of robotic embodiment, a paralyzed person could use BCI to navigate a robot or wheel-chair in a known environment (with known topological map), as if the robot is an avatar of the person's own body.

[2]

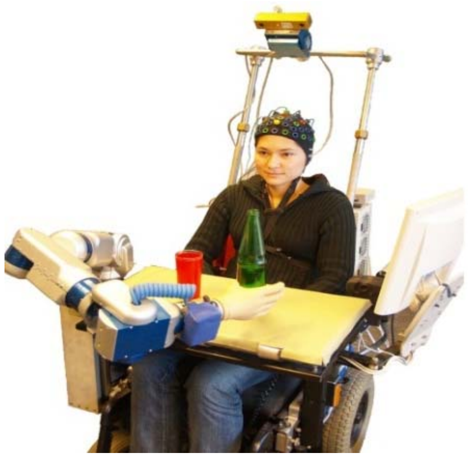


Figure 1: Example Task

(FRIEND -II System – University of Bremen)

[3]

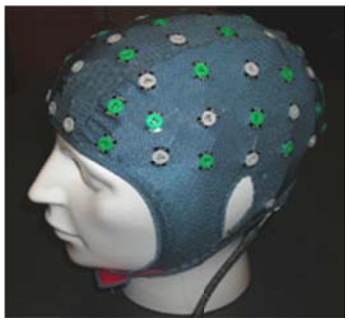


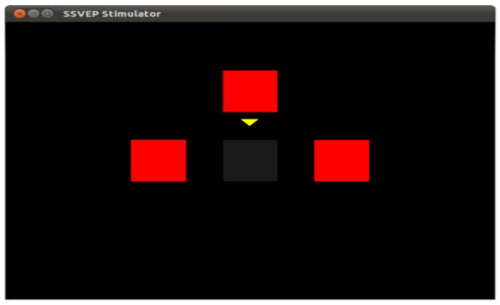
Figure 2: EEG Cap (NeXus)

[1] christopher and dana reeve foundation
[2] http://current.com/1jidc4c
[3] Stens corporation – Biofeedback Equipment and Training

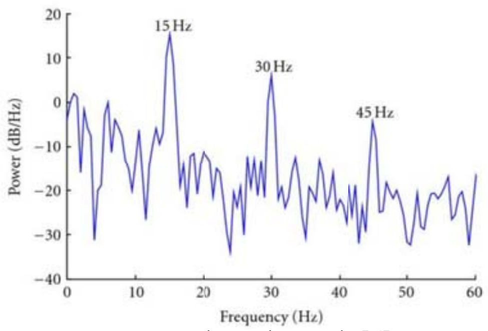
1.2 Brain-Computer Interface Paradigms

Under the framework of Brain-Computer Interface (BCI), selective attention is used to extract user intentions and preferences by use of Steady State Visually Evoked Potential (SSVEP) or P300 wave. SSVEP is the oscillatory wave appearing in the occipital leads of the EEG in response to a visual stimulus modulated at a certain frequency. The frequency of the SSVEP matches that of the stimulus and its harmonics. SSVEPs can be elicited by repetitive visual stimuli at frequencies in the range of approximately 3.5 Hz to 75 Hz [5]. In Figure 3 shown below is an example of SSVEP. In SSVEP-based BCI simultaneous flashing or flickering squares on screen are used. SSVEP has excellent signal-to-noise ratio but has limitations in the number of options that could be displayed on the screen.

4



OpenViBE (INRIA, France)



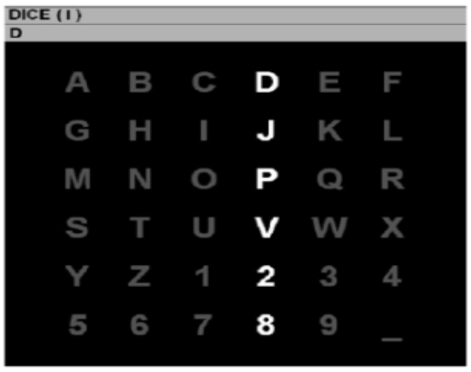
Danhua Zhu et al. [6]

Figure 3: SSVEP-based BCI

P300 based BCI uses Event-related Potentials (ERP) to record user intention in the BCI experiment. An ERP is the measured brain response that is the direct result of a specific sensory, cognitive, or motor event. More formally, it is any stereotyped electrophysiological response to a stimulus. P300 is a late appearing feature of an ERP that is elicited in the process of decision making. It is observed as a positive deflection in voltage with a latency of about 300ms [7] and hence the name P300. The signal is typically measured most strongly by the electrodes covering the parietal lobe [8]. The presence, magnitude, topography and timing of this signal are often used as metrics of cognitive function in decision making processes. It is usually elicited using a typical oddball paradigm in which low-probability target items are inter-mixed with high-probability non-target items using flashing of a matrix introduced by Farwell and Donchin [9]. Figure 4 shows the matrix at the flashing time of the items that lie in the fourth column from left. P300-based BCI has higher number of options that could be displayed on the BCI screen simultaneously, compared to SSVEP-based BCI.

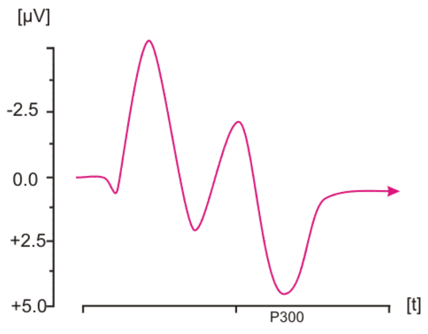
[4] OpenViBE software for Brain Computer Interfaces

[5]



Standard 6x6 P300 speller matrix
(ETSU BCI Lab)

[6]



(Birbaumer & Schmidt, 2006, S. 481)

Figure 4: P300-based BCI

During the choice task in both of these paradigms, if the interface interprets user intention wrongly and the interface recognized option is displayed on the screen to the user, then an error-related potentials (ErrP) is elicited in the brain. ErrP is a component of an event-related potential and time-locked to commission of error.

1.3 Problem Description

Semi-autonomous navigation is a flexible task sharing system and the user could assist the robot with low level navigation whenever needed and the system allows the robot to specify goals autonomously and maintain high level path planning. The communication between user and robot, when BCI is used in this context is accomplished by EEG signals. With the help of sensory information and a topological map of the environment, the robot is able to recognize situations when a decision should be made. Decisions are made either by the robot autonomously or by the user, depending on the level of sharing in use. In case it is recognized by the robot that a decision need to be made by the user, the interface should provide with navigational options from which the user selects. These options could be intended goal locations or low level commands such as turn left or move forward.

BCIs have low bandwidth or bit-rate with maximum information transfer rates of 5–25 bits/min at best, which depend both on speed and accuracy [10]. Present BCI interfaces are based on fixed policy where the user, environment and the interface start always at a specific state, and based on the user input; the interface deterministically updates its contents and presents them to the user. This method of presentation to the user is often time consuming and involves much interaction load to the user and thereby diminishing the embodiment feeling of the user in commanding a robot avatar or wheel-chair in use. Therefore the first objective is to build a brain-computer interface based on adaptive policy with automatic customization feature as per individual user for a navigational task, which is needed to make the best use of the scarce resources, i.e. with available options or bit-rate. The robot should be able to learn routines in user behaviour and propose the navigational actions to the user in a way to optimize user interaction. The developed adaptive interface needs to be tested with the help of a Gazebo simulated environment where the

[5] http://www.etsu.edu/cas/bcilab/ETSU
[6] http://de.wikipedia.org/wiki/Ereigniskorrelierte_Potentiale

robot would receive commands from the user through Robot Operating System (ROS) [7]. The Gazebo simulator is a primary tool used for robots and robotic applications in ROS community to simulate indoor and outdoor environments [8]. Generating a user model is also required to train and test the algorithm. The BCI system in this case is simulated by a Graphical User Interface.

In both fixed and adaptive policy based BCI system, the user intention recognition rates of classifier is subjected to error due to misclassification [11] and flows down till execution as a command unless prevented. The detection of ErrP in the EEG signal could be exploited to prevent possible error propagation as shown in Figure 5 below. Error-related potentials raise many challenges in terms of classification due to low signal-to-noise ratio, especially in the case of single trials [12]. In Figure 6 the average of Error-related Potentials and no Error-related Potentials (noErrP) signals over number of trials are visible to the naked eyes with clear distinction. Averaging the signal over number of trials is not a feasible method to detect ErrP as it is generated only once within a certain duration after seeing the erroneous response of the interface. For the case of single trial, the difference between the two signals is not clear to the naked eyes as shown in Figure 7. This results in big challenge to classify and categorize the scalp-recorded EEG data for detection of ErrP.

Therefore the second objective is to recognize user's cognition state that reflects user awareness to errors committed by the interface or in other words, is to develop a classifier for detection of possible ErrP in the EEG data and hence prevent misclassified user intention or command from being executed.

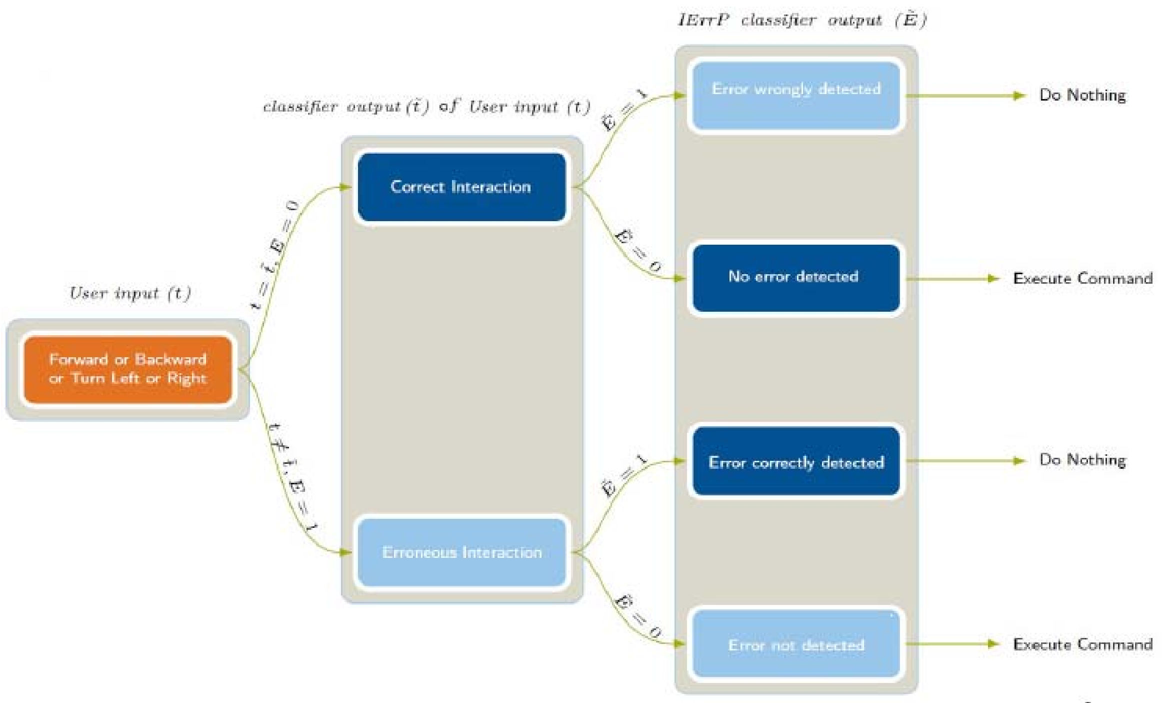


Figure 5: EEG Data Classifiers

[7] Willow Garage, Robot Operating System (ROS)
[8] http://gazebosim.org/about.html

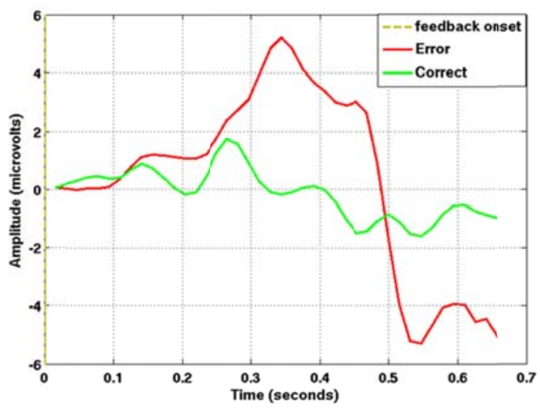


Figure 6: Average ErrP (Error trial) and noErrP (Correct trial) – [Cz electrode]

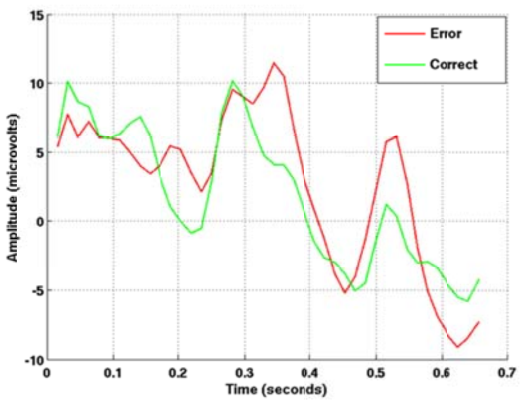


Figure 7: Single trial ErrP (Error trial) and noErrP (Correct trial) - [Cz electrode]

1.4 Report Structure

The entire report has been divided into two parts. The first part (Adaptive BCI) addresses the problem of developing an algorithm for intelligent adaptation of a BCI system as per different users. I present later in this part, the proposed algorithm and possible future works as well. The last part (Classification of Error-related Potentials), mentions the state-of-the-art classifiers for detecting ErrP in the EEG data and the methodology to use Support Vector Machine (SVM) as a classifier for detection of ErrP. Further, the results of SVM with other state-of-the-art classifiers such as Gaussian classifier and Bayesian filter method are compared.

2 PART ONE (ADAPTIVE BCI)

In this chapter the state-of-the-art adaptive user interface, brain-computer interface manager, the algorithms for learning and markov decision process for prediction of user intention along with the training and testing results of the algorithm have been presented. At last the future works related to adaptive brain-computer interface have been described.

2.1 State-of-the-Art Adaptive User Interface

An adaptive user interface also known as AUI, is a user interface which adapts or changes, its layout and elements to the needs of the user or context and is similarly alterable by each user. Alternatively an adaptive user interface is a software artefact that improves its ability to interact with a user by constructing a user model based on partial experience with that user [13]. An adaptive interface should have a property to only show relevant information to a user as per the context. The area of intelligent and adaptive user interfaces has been of interest to the research community for a long time [14]. "*To date, research in this field has not led to a coherent view of problems, let alone solutions*" [14]. Several studies including the work by Gajos et al. [15] which generate different interface renditions in response to different usage patterns, mention benefits of adaptive systems. Machine learning is an emerging field which could be exploited for their possible uses in designing adaptive interfaces, as has been examined in Pat Langley's work [16]. To mention, machine learning has been successfully incorporated in cell phone devices [17]. Reinforcement Learning (RL) [18], a machine learning paradigm, stands as a promising approach under the situation where the exact dynamics of the environment are not known. The learning problem is to find an optimal policy that maps states to actions, through a trial-and-error process of repeated interaction with the user. It has been successfully utilized in many applications including the problem of dialogue management [19]. RL has also been used to permit the robot to learn and optimize appropriate control policies from its interaction with the user [20, 21]. An interesting piece of work by Liu et al. [22] mentions how a mobile phone could learn context and user preferences via RL to adapt and set its alarm type automatically based on context information obtained from a variety of sensors. At another instance, the power of RL algorithm has been demonstrated [23] with application to a real-world problem as complex as controlling an autonomous helicopter. Partially Observable Markov Decision Process (POMDP) is another engineering framework that integrates Reinforcement Learning and Bayesian belief tracking and the benefits of this approach are demonstrated by the example of a simple gesture-driven interface to an iPhone application [13].

With regard to application of machine learning techniques in Brain-Machine Interface (BMI), a group of researchers have exploited RL algorithm to illustrate how BMI learns to complete a reaching task using a prosthetic arm based on the user's neuronal activity [24]. For the case of brain-computer interface system, Chavarriaga et al. have mentioned how the performance of user intention classifier could be improved with the help of RL [25]. Also POMDP model has been utilized to compute an optimal sequence of stimuli in P300 based BCI task [26]. In the case of a navigational task, it is interesting to see the research work by Perrin et al. [27] that demonstrates a method for the robot to propose low level actions to the user, like turn left or move forward, at the decision making points where the user could either accept or reject the proposition. In an unknown environment, the robotic system first extracts features so as to recognize places of interest where a human-robot interaction should take place (e.g. crossings). Based on the local topology, relevant actions are then proposed. It is to be noted that with this above methodology, only low level actions as analysed from the environment, are proposed one by one and Error-related potentials in case of disapproval were used for this purpose. The dialogue management strategy continued as long as the user chooses no. Even for a single user, utilizing ErrP to record user choice poses the problem that ErrP classifier is often subjected to recognition error due to single trial classification and therefore multiple recordings (until the probability difference between the two most probable actions exceeds a given threshold) need to be performed to confirm the classification result. If the cardinality of goal space is large, this method seems to be time consuming and involves much interaction workload to the BCI user and thus poses the risk of diminishing the embodiment feeling of the user in commanding a robot avatar or wheel-chair in use. Further, the work doesn't take care of change in user behaviour and possible outlier, or in other words, it doesn't take care of uncertainties in user behaviour to predict the appropriate action and propose them to the user, which often is the case for a navigational task [28].

State-of-the-art-BCI interface is based on fixed policy which would be explained in the following section. Virtually all existing user interfaces, not only in BCI, follow the finite-state machine model [13]. The area of designing an adaptive brain computer interface with dynamic customization for navigational tasks is still not explored and is being addressed through this thesis work. Several success stories of RL algorithms to many domains, motivates us to design an adaptive BCI system for navigational task with the help of RL, by treating the interface adaptation as an optimization problem. Further, the uncertainties in user behaviour such as change in behaviour and outlier could also be tackled as described in the following sections of our work. The key contribution of this thesis is to design a reinforcement learning algorithm to learn user behaviour including possible uncertainties in behaviour and predict the user intentions in a BCI based navigational task which could provide robotic embodiment feeling to the user.

2.2 Brain-Computer Interface Manager

As mentioned earlier, a paralysed person could command a robot avatar or wheel-chair using brain-computer interface system. A BCI system designed on fixed-policy interface is a finite-state automaton that mediates between the user and the robot avatar of the user. For a limited user goal-space, every goal can be mapped into one of the available BCI controls. In the case of large goal-spaces, though it is still possible to use the fixed policy approach but it is not convenient for the user due to several of the following factors. A user always starts from a starting interface and then goes down in a well-designed hierarchy which is defined by task experts till reaching the intended command, which might be a long and tedious trajectory, leading to a negative impact on the embodiment feeling. Another drawback of this approach is the requirement of a detailed task description and the fact that it imposes strong assumptions about the user and the environment by ignoring dynamics of both. Figure 8 depicts an interface designed as a finite-state automaton with three masks. Using this approach, in case the user intended option lies in the third mask, then the user needs to select 'Next' buttons in the first mask and second masks and in the third mask the user needs to select the intended option. So in this case, three interactions are required to decode a single user intention.

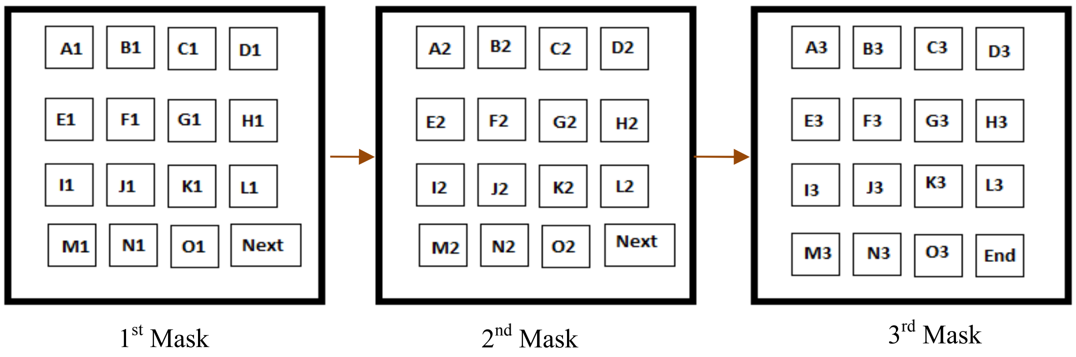


Figure 8: Example of a Finite State Automaton

2.2.1 Adaptive Interface with Dynamic Policy

As a fixed-policy interface only depends on the task description and model, the time needed to select a goal with a fixed-policy interface is user, situation and context independent. Therefore customization of the interface according to user preferences is necessary to reduce the interaction workload and consequently increase the robotic embodiment feeling. This originates the necessity of designing a dynamic interface where the options proposed to the user are no longer fixed but are dynamically customized for individual user automatically. Therefore the policy based on which the interface would be designed and managed is desired to be an optimized policy. This could be formulated as an optimization problem where decision needs to be made by the interface manager to propose the optimized set of options to the user at every decision making situation.

2.3 Markov Decision Process and Reinforcement Learning

Although humans have the potential for relatively random patterns of behaviour, there are easily identifiable routines in every person's life [28]. Based on this, the proposition of best set of actions to the user could be modelled as a Markov Decision Process (MDP). MDP provide a mathematical framework for modelling decision making in situations where outcomes are partly random and partly under the control of a decision maker. More precisely MDP is "*a decision-theoretic model capable of taking into account both uncertainty in the effects of its actions and trade-offs between competing short-term and long-term objectives when making decisions*" [29]. $\boldsymbol{S}$ is the set of all states, $\boldsymbol{A}$ is the set of all possible actions and $\boldsymbol{P}$ is the set of all state to action selection probabilities. At each time step, the process is in a state $\in \boldsymbol{S_t}$, and the decision maker may choose any action $a \in \boldsymbol{A_t}$, that is available in state s with probability $p \in \boldsymbol{P_t}$ from that state. The process responds at the next time step by randomly moving into a new state $s' \in \boldsymbol{S}$, and giving the decision maker a reward $\boldsymbol{R}$. The next state s' depends on the current state s and the decision maker's action a. But given s and a, it is conditionally independent of all previous states and actions for first order Markov model; in other words, the state transitions of a MDP possess the Markov property. MDP is modelled with $\{\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R}, \boldsymbol{\gamma}\}, where\ \boldsymbol{\gamma}$ is the future discount factor for rewards, with a value typically close to 1. In case the dynamics of the environment or state transition probabilities are not known in prior, the system should learn these transitions explicitly by interacting with the environment. Therefore this learning mechanism which gains knowledge of environment by virtue of interacting with its environment falls under the category of Reinforcement Learning (RL) scheme. The learning of the algorithm should lead to a policy that would map a state to an action for all possible states.

In our present work a P300 interface based on fixed policy has been implemented that presents, in a predefined order, to the BCI user a sequence of command options sets to choose from. It is worth mentioning that the total number of possible command options exceeds the maximum number of goal locations that can be hosted by a P300-based BCI mask. The user is modelled as a MDP and the interface should learn the user behaviours or actions carried out from every state and propose a set of best possible actions to the user as per the capacity of P300-based BCI mask. The in other words, it should capture the user intentions. Here action $a \in \boldsymbol{A}$ is the proposition of an action to the user. It is to be noted that actions here are propositions with respect to the interface manager, i.e. what set of commands should be presented to the user at each decision making situations. The scenario used is a pure navigation scenario, where all user goal-oriented commands are of the form "move to location x" or low level commands such as turn "left" or move "forward" as shown is Figure 9 for few states and action pairs.

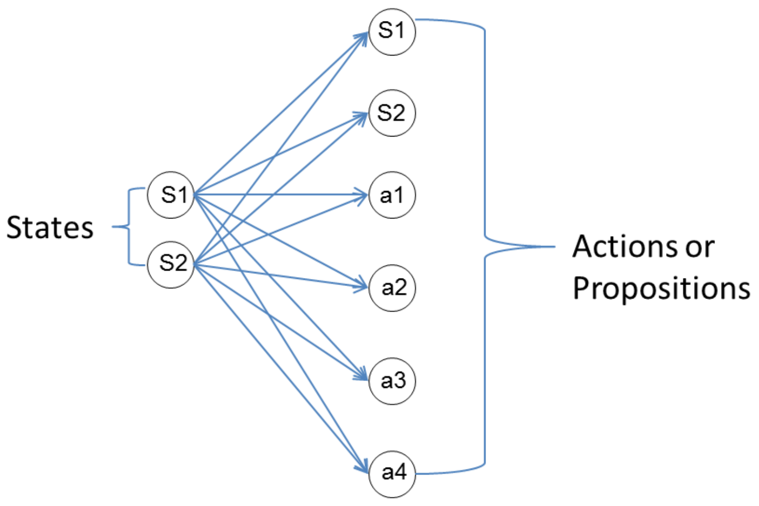


Figure 9: State-action Mapping

Additionally, RL-based algorithm has been developed thereby aiming at finding an optimal policy for the interface adapted as per individual user. The results of both approaches have been compared in terms of user interaction workload which is measured by the time needed to get user intention decoded by the system. In both cases, i.e. adaptive and fixed-policy interface, however, user input is mapped into user commands that trigger the transition to new states and interface actions without considering classification errors. For tasks with available detailed description and model or where the adaptive policy assigns same priorities to multiple actions, fixed policy will be used for those tasks or actions as it is intuitive to go for fixed policy if no concrete decision could be made regarding the possible user intended actions. This results in a system with mixed policy considering both fixed and adaptive policies. Therefore the interface proposes actions to the user as per the mixed policy. Figure 10 relates the adaptive interface manager to other system components such as EEG decoder to recognize user intention $\boldsymbol{I_u}$ and present the decoded intention to the user by the help of interface manager. The noise $\boldsymbol{N_i}$ is due to the disturbances in the EEG signal as it is captured from the electrodes placed at the scalp of the user and also includes the user intention classification error or noise by the classifier. The recognised user intention in P300-based or SSVEP-based BCI, is sent as a robotic command to the robot avatar or wheel-chair in use.

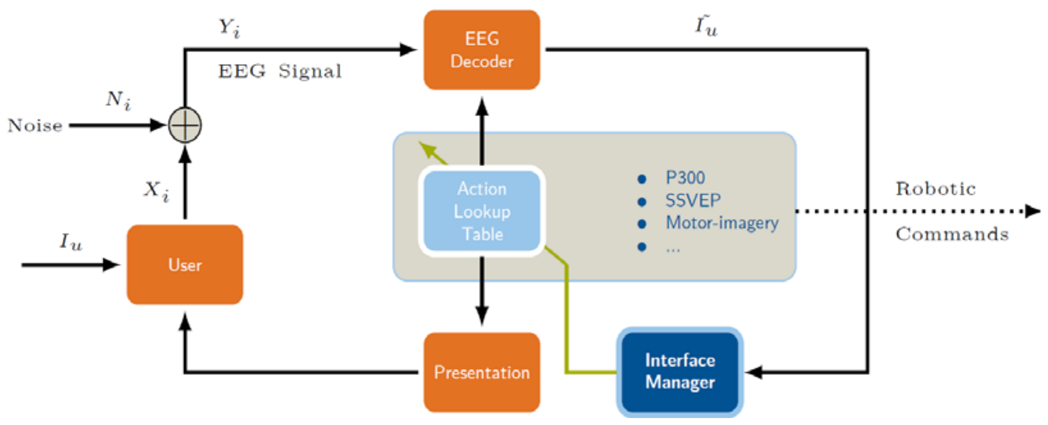


Figure 10: Adaptive Interface Manager and its Relation to Other Components of the Embodiment System

2.3.1 Learning User Behaviour and Prediction of User Intention

Decision-making model is used here to first observe and learn the user behaviour and then predict the user intention in the context of a navigational task. The underlying pattern in user behaviour could be of first, second or higher orders, though no assumption is made with respect to the order of the dynamics. Modeling user behaviour with second or higher order Markov Decision Process, pose the problem of increased memory consumption. Therefore a first order MDP is considered to solve the navigation problem. The knowledge of the user model is incomplete which means that the action selection probabilities from a state are not known in advance and they need to be explicitly learned from the user behaviour using the strategy of reinforcement learning.

States are defined as every possible goal locations in the map. Actions are propositions of next states or low level commands to the user. These low level commands include commands like moving forward or turning left etc. Thus the actions are more in numbers than the states. Both the number of states and the number of possible actions are known in prior. The number of state-action pairs thus rises to number of states times' number of actions. The objective is to find a policy which could map each state to a best possible action. As P300 mask could contain not just one but a set of possible actions or propositions, therefore for each state the policy should rank all user intended actions and show a set of most probable actions in the first mask depending upon the capacity of the designed P300 mask which is available in different variants.

2.3.2 Learning Behaviour

The learning mechanism associated with navigation task is a continuous process having finitely large goal state space $S \in \{s1, s2, s3, \dots, sn\}$ of the robot and interface action space $A \in \{a1, a2, a3, \dots, an\}$. Every goal state is considered as a location in the map which is fully observable. The user action is fully observable but the action probability P from a state is not known in prior. The algorithm has the opportunity to learn regarding user behaviour after every interaction with the user. It is interesting to note, as every state here is a goal location and therefore the problem of predicting with finitely large goal spaces could be divided into sub-problems each consisting of only two goal locations or states, one as a starting and the other as ending state. In other words each sub-problem could be considered as an episodic task where the episode ends after every transition. Based on the interface action, the interface receives an immediate reward r, which is considered here as the needed time or number of interactions with the user. Though there are several possible actions from a state, the user only performs one action every time and thus reaching from a starting state to a goal state in one step. The user behaviour is stochastic with actions having a probability distribution from each state. For the purpose of learning a stochastic user model, sampling is done from the probability distributions of actions in a state and a trajectory generated is made available to the algorithm. The algorithm is expected to learn the user model from this generated trajectory. As the sampled trajectory is made available to the algorithm, bootstrapping method is not used in this case, where a guess is made depending upon another guess as done in Temporal Difference prediction method [18]. It is to be observed that here every transition in the trajectory involves a learning step for the algorithm.

The state of the robot (location) at time t is defined as $s \in \boldsymbol{S_t}$. Taking action $a \in \boldsymbol{A}$ from state $s \in \boldsymbol{S}$ will either make the robot move to a state s$'$if a $\in \boldsymbol{A_h}$, where $\boldsymbol{A_h}$is the set of all high level commands or, in other words, the set of all goal locations or will make no change to s if a $\in \boldsymbol{A_l}$, where $\boldsymbol{A_l}$ is the set of all low level action commands. More formally, $\boldsymbol{S_t}$is defined to be a random variable referring to current state and $\boldsymbol{S_{t+1}}$ refers to next state. $\boldsymbol{A_t}$ is a random variable that refers to an action at time t and hence;

$$\boldsymbol{S_{t+1}} = \boldsymbol{S_t} \text{ if } a \in \boldsymbol{A_l}$$
$$\boldsymbol{S_{t+1}} \neq \boldsymbol{S_t} \text{ if } a \in \boldsymbol{A_h} \text{ or } a \in \boldsymbol{S_{t+1}} \tag{1}$$

I assume that that robot executes all command deterministically, which means that if I know a then I know s'deterministically. Therefore the problem of predicting user next state is exactly the same as predicting the next user action. Based on this transition, the state-action pair receives a positive reward of $r_f = 1$. A term defined as frequency of selecting an action $a$from a state s is the accumulated rewards over time and updated for the occurred state-action pair as follows:

$$F(a/s) = F(a/s) + r_f \tag{2}$$

As the robot has no initial knowledge of the dynamics of the environment, the value of each state-action pair is termed as:

$$Q(s,a) = 0 \;\; where \; \forall \, s \; \in \boldsymbol{S} \; and \; \forall a \in \boldsymbol{A} \tag{3}$$

It is worth mentioning that frequency of a state-action pair $F(a/s)$ represents the $Q(s,a)$ value of that state-action pair and therefore the below assignment holds true for our case:

$$Q(s,a) = F(a/s) \tag{4}$$

A policy is defined as a mapping from every state s to an action a. Policy improvement is done by making the policy greedy with respect to the current value function. For any state-action value function Q, the corresponding greedy policy is the one that, for each $s \in \boldsymbol{S}$, deterministically chooses an action with maximal Q value:

$$\pi(s) = argmax_a Q(s,a) \tag{5}$$

The reward r is a form of motivation for the occurred state-action pair and is likely to increase the chances of occurrence of the action a while the robot visits the state s next time. As an example, after first interaction with the user, the Q value of the occurred state-action pair at time $= 1$, is updated to:

$$Q(s,a) = 1 \tag{6}$$

Matrix of dimension equal to number of states times number of actions, is memorized which contains the $Q(s,a)$ values $\forall s \in \boldsymbol{S}$ and $\forall a \in \boldsymbol{A}$, and the values are updated after every interaction with the user.

2.3.3 Change in Behaviour

Humans have potential for relatively random patterns of behaviour and hence the behaviour could be termed as stochastic [28]. The amount of randomness regarding behaviour of a user corresponds to its entropy. Entropy (in bits) of a discrete random variable $\boldsymbol{X}$ is defined by Claude Shannon in the equation below.

$$\boldsymbol{H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i)} \tag{7}$$

People who live less entropic lives are easier to predict and but who live entropic lives tend to be more variable and harder to predict. One such case of user behaviour could be related to change in behaviour after certain number of interactions with the interface where the user jumps from a policy π to another policy π' and continues following policy π' thereafter. In order to handle this situation of changed behaviour by increasing the convergence rate of the algorithm or learn the altered behaviour faster, another form of reward is incorporated. Separate interaction counter or local timer for each state is defined and an action executed in a state increments the timer associated with that state by one. This local timer value is rewarded to the occurred action of the corresponding state. Therefore this timer associated reward function $T(s,a)$ for a state-action pair is intuitively a function of the time the action a happened last in the corresponding state s. It is worth noting that the timer associated reward function $T(s,a)$ is equal in magnitude to the marginal count $F_{t_a}(s)$ of the state s at the time of last occurrence of the action a in the same state s. Hence each state-action pair was time stamped with a reward value equal to the marginal count of the corresponding state when the respective action was seen last occurring in that same state and hence defined mathematically as below:

$$T(s,a) = F_{t_a}(s) \quad where\ \forall s \in \boldsymbol{S} \text{ and } \forall a \in \boldsymbol{A} \tag{8}$$

Therefore the modified $Q(s,a)$, when the possibility of changed behaviour pattern of a user is included, takes the form:

$$Q(s,a) = w_f * F(a/s) + w_t * F_{t_a}(s) \tag{9}$$

$where\ w_f$ is the weight of Frequency based reward, w_t is the weight of Timer based reward

While the first term on the right hand side of the above equation favors frequent actions as it is related to frequency of taking an action in a state, the second term favors recent actions in a state. The intuition behind the second term on the right hand side of the equation is that in a state the action that occurred recently is assigned higher timer value than the action occurred earlier in the same state. Therefore the recent action or user behaviour is given higher priority compared to previous action from the same state.

It could be noted that use of a global timer or interaction counter as a reward function is avoided here though it seems easier with regard to implementation. It is due to the fact that using a global timer would assign a large value to a recent action that occurred from a less visited state in which case the reward associated with the global timer would be more dominant than the reward associated with the frequency term in spite of having the weighting terms for each of timer and reward. Hence for the case of a global timer, finding a weight that would compensate for both less visited as well as frequent states would be difficult and may lead to wrong prediction of user intentions. Also if there is no change in user behaviour observed then also a global timer based reward function would perform poor in terms of prediction, compared to a local timer based reward.

2.3.4 Outlier in Behaviour

Outlier is defined as completely random pattern in user behaviour or action for certain short period of interactions such as visiting a 'Friend' once in a month etc. In other words, the user follows a policy π and after certain number of interactions the user started following or switched to another policy π', with having completely random state-transition probabilities for few number of interactions and then again goes back to the original policy π. The difference between change in behaviour and outlier is that in the case of first one the user never goes back to the original policy π after changing the behaviour to follow another policy π', while for the latter case the user goes back to the original policy π after certain number of interactions. The weight of timer w_t introduced in equation (9) above helps the algorithm to learn outlier as it assigns higher timer associated reward to recent transition but in order to increase the rate of learning during outlier period and also after completion of it, another reward function is defined which assigns a reward $r_i \in \{0,1,2\}$, based on the number of interactions needed to select user intended action a available in a state s. Here the number of interactions mean the number of masks that are needed to be shown to the user for selection of an action from a given state. Hence $r_i$is a measure of the immediate reward from user to the interface policy. The cumulative interaction reward function for a state-action pair $I(s,a)$ is memorized in a table, which is simply the accumulated immediate interaction rewards for an action a that occurs in state s and is updated after every interaction with the user as shown below:

$$I(s,a) \leftarrow I(s,a) + r_i, \ \ where\ \forall s \in \boldsymbol{S} \text{ and } a \in \boldsymbol{A} \qquad (10)$$

In our experimental setup if the user intended option appears in the first mask then immediate interaction reward r_i for that state-action pair is assigned a value 0 and similarly if it happens to be in second or third mask, it is rewarded with values 1 and 2 respectively. The intuition behind introduction of this interaction based reward function is to allow the algorithm to converge faster, mainly after outlier period is over, by motivating the user intended actions that need greater number of masks to be shown to the user.

Combining above described three kinds of situations, i.e. considering a stochastic user behaviour with change in pattern as well as presence of outlier, the $Q(s,a)$ could be reformulated as a combination of reward based on state-action frequency, local timer and number of needed interactions, as follows:

$$Q(s,a) = w_f * F(a/s) + w_t * F_{t_a}(s) + w_r * I(s,a) \qquad (11)$$

$where\ w_f$ is the weight of Frequency based reward, w_t is the weight of Timer based reward and w_r is the weight of needed Interaction based reward

2.3.5 Prediction of User Intention

Decision needs to be made as per the optimal policy of interface that imitates the user hidden policy. The action based on optimal policy π^*, is searched in the look-up table of $Q(s,a)$. As in the case of P300-based BCI, there is possibility of displaying a set of actions in the mask, therefore the optimal policy π^* in this case ranks all the actions with higher Q valued actions at the top for every state and proposes the actions to the user as per the capacity of the mask. Mathematically:

$$\pi^*(s) = \arg max_{set\ a\in A} Q(s,a) \tag{12}$$

As the actions are ranked as per the relative values of $Q(s,a)$, therefore the expression:

$$Q(s,a) = w_f * F(a/s) + w_t * F_{t_a}(s) + \ w_r * I(s,a),$$

could be transformed into $Q'(s,a) = F(a/s) + w'_t * F_{t_a}(s) + w'_r * I(s,a)$ (13)

In the following sections, $Q'(s,a)$ is mentioned as $Q(s,a)$ for simplification.

The process of learning and prediction steps could be viewed together as in the Figure 11 and the algorithm is also mentioned below.

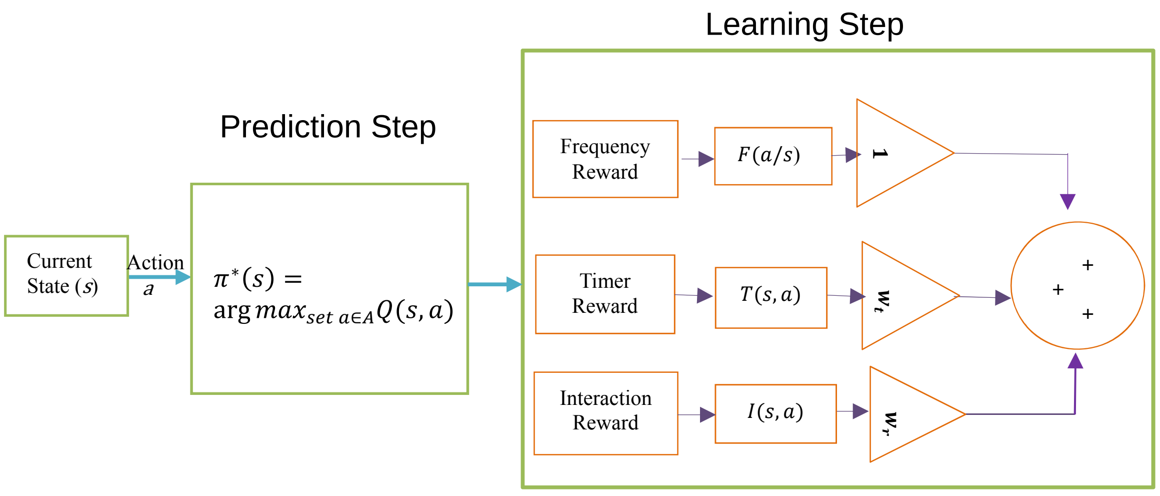


Figure 11: Process of Learning and Prediction

```
$begin$
$Initialize\ F(a/s),\ F_{t_a}(s),\ I(s,a) \leftarrow 0 \quad \forall\ s,a$
$repeat\ for\ each\ iteration$

$Initialize\ r \leftarrow 0,$
$A \leftarrow set\ of\ all\ actions$                    $//\ r\ is\ immediate\ interaction\ reward$

            $Q(s,a) = F(a/s) + w_t * F_{t_a}(s) + w_r * I(s,a)$     $//\ prediction\ step$
      $for\ i = 1{:}\,3$
            $A_I = \pi^*(s) =$
           $arg\ max_{A_I \in A}\ \sum_{a \in A_I} Q(s,a)\ \ s.t.\ \ A_I\ contains\ non-$
                                              $repeated\ elements\ and\ |A_I| =$
                                              $P300\ mask\ capacity$
            $if\ a_u \subset A_I$                          $//\ user\ intended\ action$
                $break$
            $else\ if\ a_u \leftarrow next\ (IErrP = 1)$
                $increment\ r\ by\ 1$
                $A \leftarrow A - A_I$
            $end\ if\ loop$
      $end\ for\ loop$
            $F(a/s) \leftarrow F(a/s) + 1$
            $update\ F_{t_a}(s)$
           $I(s,a) \leftarrow I(s,a) + r$
      $continue$
$end$
```

Table 1: Learning and Prediction Algorithm

2.4 Training of the Algorithm Parameters and Testing

A map of the environment is built in Gazebo with several goal locations. Each of the locations like ‘Office’, ‘Hostel’ etc., are assumed to be goal locations or states of the PR2 willow garage robot[9], in this case. In total, thirty nine states are defined in the map. Actions are thirty nine high level goal based and four low level commands with total of forty three actions to choose among. Robot Operating System is used to publish the desired goal location as a ROS topic to PR2 navigational stack or to command the robot through low level navigation. The path planning algorithm is made available with PR2 navigational stack. A top-view snap shot of the simulated Gazebo world with PR2 robot is shown below in Figure 12.

[9] http://www.willowgarage.com/pages/pr2/overview

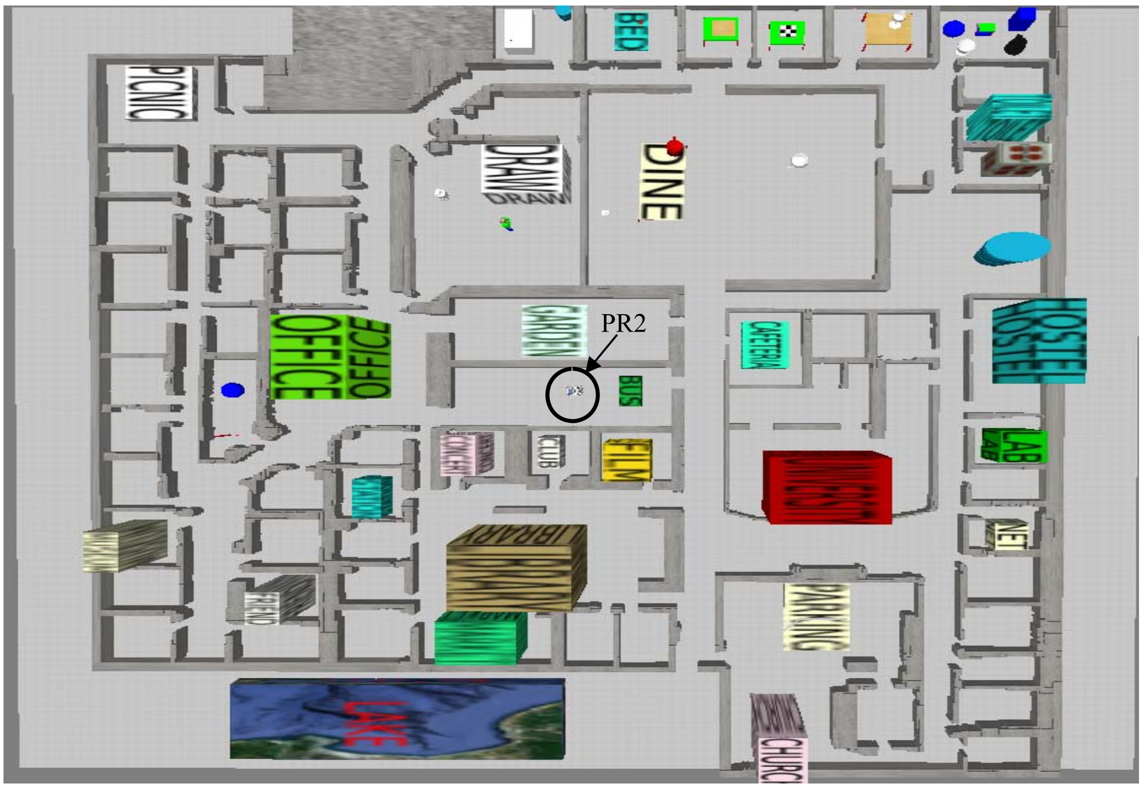
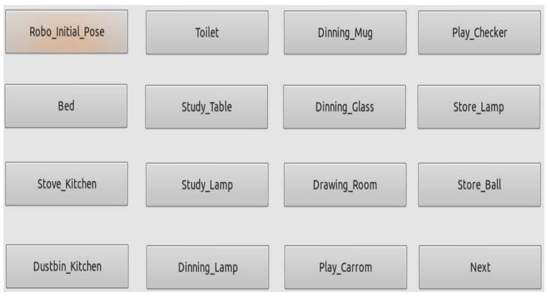


Figure 12: Simulated Robot World in Gazebo Environment

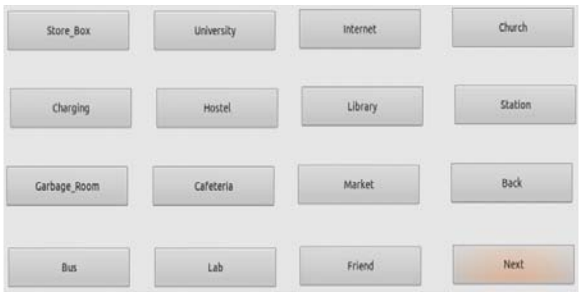
As described earlier the user model includes stochastic behaviour. The probability of each action in a state are defined for the purpose of developing a user model. The user was modeled as first order MDP and implemented as shown in Appendix A. Three different user models are developed in order to test the performances of the algorithm with different kinds of models. The first model which is a high entropic model, has uniform distribution with average of maximum probabilities around 0.04 while in the second user model with medium entropy, it is assigned a maximum state-action probability as 0.5 on an average. The last user model had 80% deterministic behaviour and thus lower entropic model, to prefer a particular action from a state. In order to introduce changed behaviour mode, the user model had the provision to define number of training instances after which the behaviour pattern is changed. Two modes for changed behaviour pattern are considered. During strong change in behaviour mode, all the state-action probabilities are redefined with high entropy while during weak change in behaviour mode; only the action with maximum probability from each state is swapped with another action available in the same state. In the user model, there is also provision of defining outlier period during which the user follows a completely random policy. All together, the three user models are equipped with different maximum state-action probability distributions and simulate change in behaviour and outlier. For fixed policy interface, the user model was used to evaluate the testing performance while for adaptive policy it was used both for training and testing.

2.4.1 Fixed Policy Interface

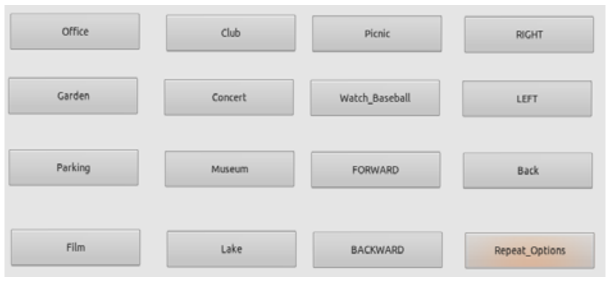
For our test case, I have defined an environment with 39 states and 4 low level commands and this sum up to 43 actions in total to simulate a P300-based BCI with 3 masks. Each mask could contain at most 16 options including options such as 'next', 'back' or 'repeat' buttons. As it is expected, there was no 'back' button for the first mask which means it had 15 possible actions including the initial robot pose which is displayed as 'Robo_initial_Pose'. The 'Repeat_Options' is dedicated only for the mask that comes in the last. In the case of fixed policy interface, the BCI masks have predefined options in a specified sequence. I have used the map of the environment to define a fixed policy. The masks designed in QT platform used for fixed policy P300-based BCI is shown in Figure 13 below.



(a) 1st Mask



(b) 2nd Mask



(c) 3rd Mask

Figure 13: (a), (b) and (c) Simulated BCI with 3 Masks Used for Testing of Fixed Policy P300-based BCI

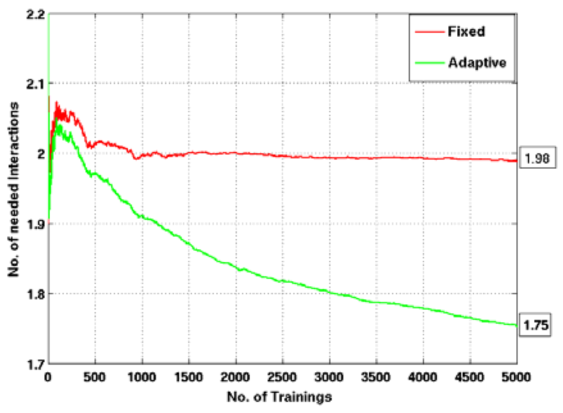
2.4.2 Adaptive Policy Interface

In order to train the algorithm, the trajectory developed with the help of a user model as described earlier, is used. The transitions in the trajectory are used as training instances, sequentially starting from an initial state till the length of the trajectory. After each instance of training, the algorithm is tested against the number of interactions needed in order to capture the immediate next action while following the trajectory.

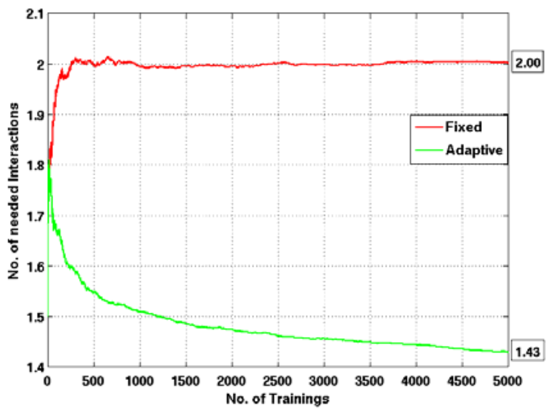
2.4.3 Results of Fixed and Adaptive Policy Interfaces

For the user model without incorporating change in behaviour and outlier, result could be seen below in Figure 14.

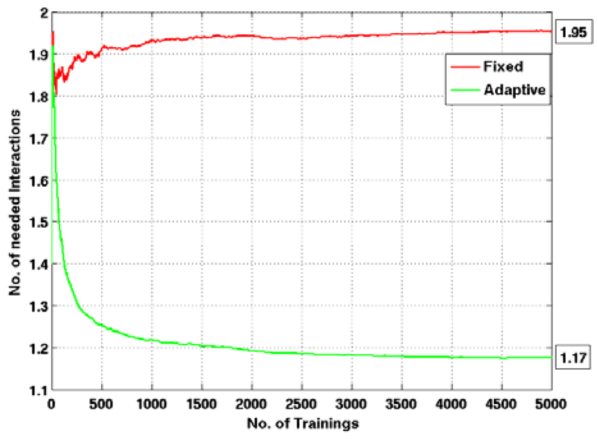
(Note: The abscissas of the following plots have different ranges)



(a)



(b)



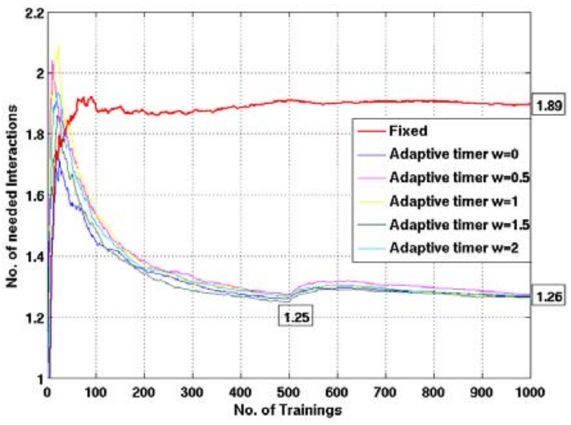
(c)

Figure 14: User Behaviour: (a) Highly Entropic Behaviour, (b) Medium Entropic Behaviour and (c) Low Entropic Behaviour

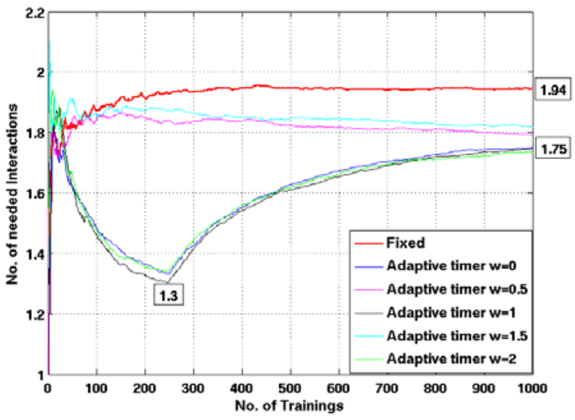
For the fixed policy case, the average number of needed interactions is higher than the interactions needed for adaptive policy in any of the three kinds of user models described above.

It could be seen for the model with higher entropy which is a measure of randomness, the learning rate is slow and it approaches a higher value after certain number of trainings, while in the case of medium entropic behaviour it attains a lower steady state value. As expected, in that third case with low entropic behaviour, the steady state value reaches near to the ideal interface which requires only one interaction to capture user intention. So from the results above, it could be stated that a higher entropy in user behaviour leads to a lower convergence rate of the learning algorithm or higher average per time per interaction. It means that if there is no routine in user behaviour, there is no learning of the algorithm.

Different values of timer weights have been analyzed and the results are displayed both for weak and strong change in behaviour in Figure 15 separately.



(a)



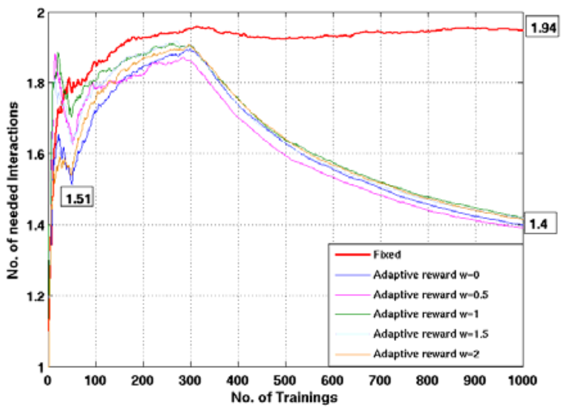
(b)

Figure 15: Changed Behaviour Mode: (a) Weak Change at 500 number of trainings, (b) Strong Change at 250 number of trainings

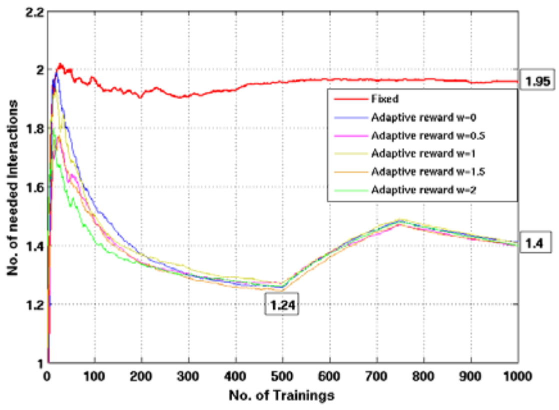
As was in the previous case, fixed policy performs poorer than the adaptive policy even in the case of change in behaviour. It is worth noticing that in the case of weak change in behaviour, the algorithm quickly converges even after the changed behaviour pattern is introduced and approaches a value close to the steady state value before there is change in behaviour. But for a strong behaviour change, the learning curve approaches towards the case of completely random behaviour as shown in Figure 14 (a). Different timer weights affect the learning rate which is visible in Figure 15, above.

For the situation where there are outliers in user behaviour pattern, the results are as follows in Figure 16.

(Note: The abscissas of the following plots have different ranges)



(a)



(b)

Figure 16: User Behaviour with Outliers (a) from 50 to 300 number of trainings and (b) from 500 to 750 number of trainings

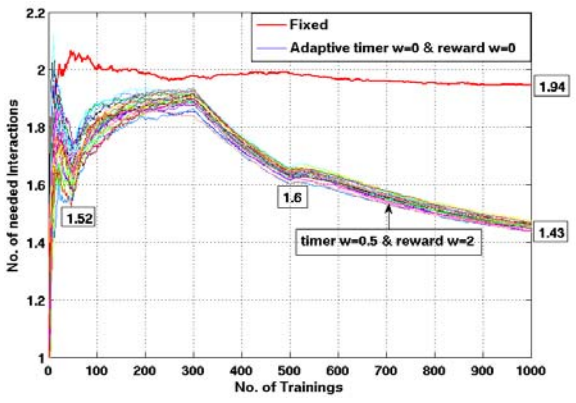
As expected, there is loss of information for the cases of early occurrence of outlier and also for later occurrence of outlier in user behaviour. After the outlier period is over, learning curves in both cases approach towards the same value. Again, different weights for cumulative interaction reward contribute to different convergence rates and steady state values. An attempt was made to analyze the performance of the algorithm where there are both change in user behaviour and outlier and the results for two completely different user models, are as follows:

Outlier (50-300 number of trainings) and weak behaviour change at 500 number of trainings

(a)

Outlier (250-750 number of trainings) and weak behaviour change at 900 number of trainings

(b)



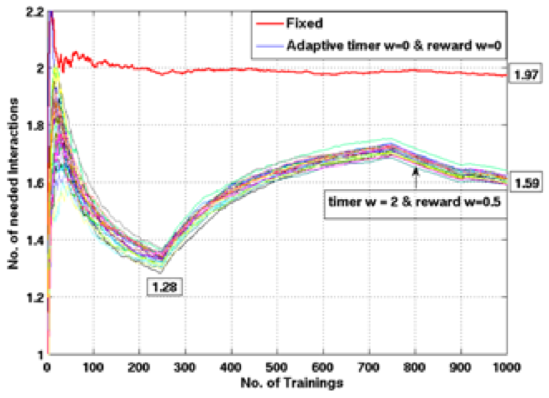


Figure 17: Both Outlier and Change in Behaviour: (a) and (b)

Even with these combination or outlier and change in behaviour cases, fixed policy was observed to be performing poor compared to adaptive policies evaluated based on different combinations of weight for timer and interaction reward. It could be stated that if the outlier period is longer, the loss of information is more. However different weights could be set with regard to timer and interaction reward, which affect the convergence rate and steady state value or steady state error. It is worth noting that setting positive values for w_t and w_r helps the algorithm to converge faster both during changed behaviour mode and also during outlier period but they need to be set individually in order to lower the steady state error. Therefore designing individual weights is a trade-off between convergence rate and steady state error of the learning algorithm. For particular user behaviour, these weights are set manually after observing the steady state value of the algorithm for different set of weights. The algorithm could further be enhanced by automatic adaptation of the weights for different users. In order to provide this enhanced capability, number of iterations need to be carried out and the weight values need to be selected which satisfy the convergence criteria, i.e. to select those weight values which help the algorithm to converge within specified number of iterations and within a defined steady state value.

The training and testing method mentioned above could be viewed as an on policy training and testing, because prediction is done after every interaction with the user model. However, testing was also carried out after the learning period is over or the algorithm was exposed to a specified number of instances. Thereafter, it was tested for a trajectory of length twenty and the observed average number of interactions closely matched the interaction values seen from the above figures. Adaptive policy again performed better with a needed average interaction value of 24 compared to the fixed policy which was 38 on average for lower entropic user behaviour of Figure 14 (c) above.

2.4.4 Extension of the Learning Scheme

The first order markov decision model is extended to second order markov decision model, as it is believed that implementing a higher order markov model may better predict the user behaviour which could be of any order in reality. However a higher order markov model increases the memory consumption rapidly. As an example, in a second order markov model, the next state of the user not only depends on the present state but also on the immediate state prior to the present state. It consumes memory of approximately $(|\boldsymbol{S}|^{3})$, where $|\boldsymbol{S}|$ is the cardinality of the state space. Therefore due to increased memory consumption and also due to unavailability of second order MDP of user model, this learning algorithm has not been tested. There is an attempt made to extend the definition of state to include time schedule, in order to have better prediction of user behaviour which is dependent not only on current state but also on time of the day, e.g. a finite number of temporal values, twenty four in our case considering an entire day with twenty four hours, is used to discretize each state which is originally a goal location into finite number of states in order to include both location and time of the day in the definition of a state. Separate track of locations along with hour of the day is memorized. However testing of the algorithm with time schedule would need a more complex user model than implemented here and increased memory usage due to state discretization.

2.5 Future Work on Adaptive Brain-Computer Interface

As part of the future work, this algorithm could also be enhanced with the ability to distinguish a goal location to a sub-goal location where a sub-goal might be triggered due to external events such as ringing of a bell etc. This could be implemented by ignoring the look-up table Q value update rule when a sub-goal is recognized as an external event. The Markov Decision Model could be extended to include partially observability of user state which might arise due to misclassification of user intention in the BCI task. Another improvement in the algorithm might be related to current memory requirement of the algorithm as it is evident that the memory requirement increases rapidly depending upon the number of states because of the fact that the look-up table memorizes values for each state-action pair. In order to reduce the memory requirement, sparse representation might be implemented. Also the definition of state could further be enhanced by including additional sensory information that might be input to the learning algorithm such as environmental temperature condition, light intensity etc. in order to better predict user behaviour as these parameters as well affect user behaviour and hence user intention. But it would require additional memory for implementation. Further to enhance the optimality of this BCI, the reward in terms of cognitive load of the user could also be considered.

3 PART TWO (CLASSIFICATION OF ERROR-RELATED POTENTIALS)

In this chapter state-of-the-art classification of error-related potentials, introduction to support vector machine, feature extraction and classification method using support vector machine, training data set used for classifier, and the performance results have been presented. This chapter ends describing the potential future work related to support vector machine classifier for detection of error-related potentials in electroencephalographic data.

3.1 State-of-the-Art Classification of Error-related Potentials

Interaction Error-related potentials (IErrPs) are special features that can be detected in the EEG, after a wrong action selection by the BCI system or the user. After the onset of the feedback indicating the selected action, these features can be distinguished by first, ‘‘*a sharp negative peak after 250ms followed by a positive peak after 320ms and a second broader negative peak after 450ms*” [30]. Figure 18 and 19 shows both ErrP and noErrP signals for average and single trail cases respectively. In order to detect the presence of ErrP in the EEG data, a classifier needs to be developed.

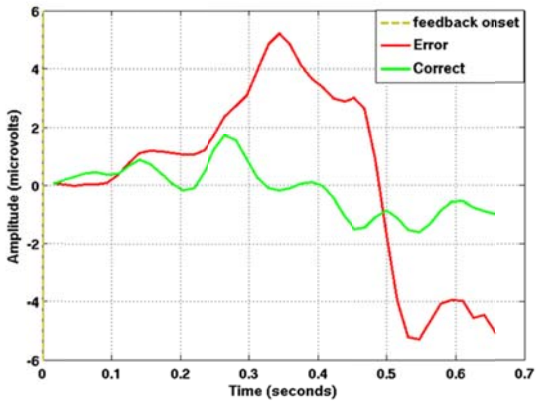


Figure 18: Average ErrP (Error trial) and noErrP (Correct trial) – [Cz electrode]

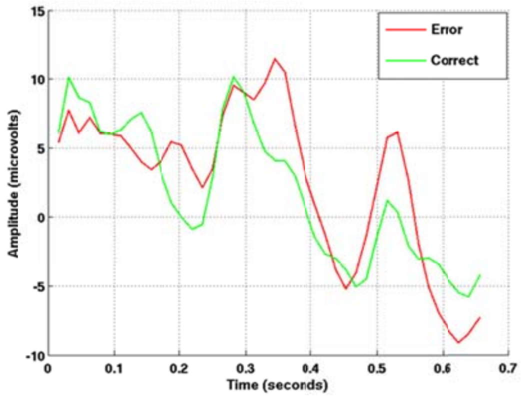


Figure 19: Single trial ErrP (Error trial) and noErrP (Correct trial) - [Cz electrode]

Out of several methods of data classification, as there is no optimal generic method, the method should be chosen depending on the task. If the training data is labelled, then supervised learning methods are used for classification and if the training data is unlabelled then it falls under unsupervised learning category while finding pattern in data. A supervised learning algorithm analyses the training data and produces an inferred function, which is called a classifier. From training samples a decision rule can be constructed that accurately assigns labels for feature vectors in new sets of data, e.g. Support vector machine (SVM) or Gaussian classifier. Literature survey revealed that that previous work on EEG data classification for detection of ErrP carried out by Bollon et al. [31] was able to recognize 71% of ErrP and 85% for noErrP data using a Bayesian filter. The study by Chavarriage et al. [25] had reported classification accuracy for three different subjects with maximum recognition rate around 73% and 92% for ErrP and true positive respectively, using Gaussian classifier. Similarly the average of the results obtained from three different subjects, reported by Ferrez et al. [32] were around 79% and 82% with standard deviation of 6.6 and 7.0 percentages for true negative and true positive respectively, as shown below in Table 2. In their statistical classifier, every Gaussian unit represents a prototype of one of the classes to be recognized.

Chavarriage et al.			**Ferrez et al.**		
Subject	**ErrP (%)**	**noErrP (%)**	**Subject**	**ErrP (%)**	**noErrP (%)**
1	73.5	92.01	1	87.3 ± 11.3	82.8 ± 7.2
2	58.91	83.82	2	74.4 ± 12.4	75.3 ± 10.0
3	66.29	86.86	3	78.1 ± 14.8	89.2 ± 4.9
Avg	66.23	87.56	Avg	79.9 ± 6.6	82.4 ± 7.0

Table 2: State-of-the-art results of Error related Potentials classification

3.2 Support Vector Machine

Support vector machine is a method that uses supervised learning for analysing and recognizing patterns in data both for classification and regression. SVM emerged in mid-1990 from the area of statistical learning theory developed by Vapnik in the late 1970's [33]. SVM has several benefits compared to other classification techniques [34]. Today SVM are widely used in many areas, for handwritten digit recognition, object recognition and many others. "*Support Vector Machines are among the best (and many believe is indeed the best) "off-the-shelf" supervised learning algorithm*" [35] and therefore among various available methods of classification, SVM was chosen to classify EEG signal for detection of error-related potentials elicited in the brain after observing erroneous response. SVM is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible as shown in Figure 20. It constructs a hyperplane or set of hyperplanes in a high or infinite-dimensional space, which is used for classification. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class, also called as functional margin; since in general the larger the margin the lower the generalization error of the classifier. A functional margin is defined as follows:

$$\boldsymbol{y^i}(\boldsymbol{w^t}\boldsymbol{x^i} + \boldsymbol{b}) \tag{14}$$

$where$ $\boldsymbol{x^i}$ is the training instance, $\boldsymbol{y^i}$ is the corresponding class, $\boldsymbol{w^t}$ is the weight vector with b as the intercept term.

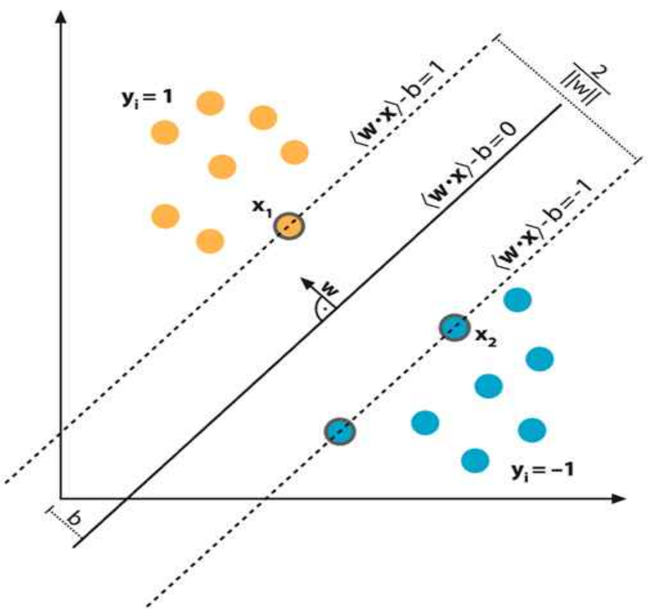

$y_i = 1$
x_1
$\langle w \cdot x \rangle - b = 1$
$\langle w \cdot x \rangle - b = 0$
$\langle w \cdot x \rangle - b = -1$
$\frac{2}{\|w\|}$
w
x_2
$y_i = -1$
b

Figure 20: Support Vector Machine Example [36]

New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. For linearly separable data, a Linear SVM classifier works well. In addition to performing linear classification, SVMs can efficiently perform non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $\boldsymbol{K(x, y)}$ selected to suit the problem. Therefore selection of kernel function is an important step in the process of classification. This problem of kernel and kernel parameter selection could be simplified because simple kernels have proved to be sufficient enough and appropriate parameters can be found using grid or pattern search. Among various, one of the advantages of SVM is that it could find a boundary between the classes even for data which are not linearly separable, by projecting them into higher dimensions. Also SVM provide a good out-of-sample generalization, if the kernel parameters are appropriately chosen. For the sake of these above mentioned advantages, SVM classifier was chosen for classification of EEG data to detect presence of error-related potentials.

3.3 Data Set Used for Classifier

In this simulated BCI experiment similar to the one described in [32], it has been tried to explore interaction ErrPs in case of erroneous keyboard interactions. Thereby the user tries to push a ball into a hole which is located on the same horizontal line as the ball using keyboard left and right arrow keys only. The user input is translated by the interface into movements of the ball; thereby it moves the ball into the wrong direction with a probability of error P as shown in Figure 21. The recognition of the ErrPs is challenging due to the low signal-to-noise ratio (SNR) inherent in single trials, as opposed to averaging number of trials in the case of P300.

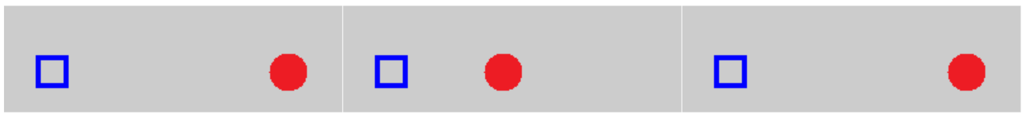

(a) t=T (b) t=T+1 (c) t=T+2

Figure 21: Experimental setup for interaction ErrPs. Using left and right arrow keys, the user should bring the ball (red circle) into the hole (blue rectangle). As an example, the initial positions of the ball and the hole, are shown in (a). The user hits the left arrow button to move the ball closer to the hole, and the result is shown in (b). Interaction ErrPs are evoked when the user hits the left arrow button and the ball goes to the right as shown in (c). Erroneous interactions reduce the information transfer rate ITR, e.g. ITR=0 for this example.

Ignoring user interactions, which are followed by IErrPs, leads to higher information transfer rates [30, 32, 37]. As a pilot study, data was recorded from one subject using a 32-channel acquisition system (from g.tec company [10]). Based on the extended 10-20 system, 32 active electrodes were placed at the following positions: FP1, FP2, F7, F8, F3, F4, T7, T8, C3, C4, P7, P8, P3, P4, O1, O2, AF3, AF4, FC5, FC6, FC1, FC2, CP5, CP6, CP1, CP2, Fz, FCz, Cz, CP, Pz, Oz. One additional passive ground electrode was placed at Fpz. Signals were referenced to the right earlobe. The signal was sampled with frequency of 256 Hz. EEG data is often subjected to noise which may arise due to blinking of eyes or movement of body parts. EEG data therefore need to undergo pre-processing, before it could be used for classification. Pre-processing includes operations like artefact removal which is a noise with high magnitude. Then the signal is re-sampled which applies an anti-aliasing filter or low pass finite impulse response filter to the signal and changes the sampling rate to one fourth of original sampling value, i.e. to a sampling frequency of 64 Hz. Among several available channels, data from channel 'Cz' which is located near the parietal lobe that has strong signal content is used to train the classifier. Over 5 sessions total of 74 and 260 instances of ErrP and noErrP were captured respectively for training, validation and testing of classifier. A parameter 'time of interest' was used to extract each signal portion of interest which was set between 0 and 650ms here. Implementation of SVM was carried out the following way.

3.4 Feature Extraction and Training of Classifier

3.4.1 Method of Feature Extraction

It is one of the most important steps in the recognition process [38]. Mentioned below, several categories of iterations that were performed in order to estimate a suitable way for feature selection to distinguish ErrP from noErrP instances, in the EEG data.

a) Each Data Point as an Instance

In this category, single trial ErrP and noErrP signal portion of interest have forty two data points as shown in Figure 22 (a) Each of these data points are considered as independent instances that either belong to ErrP or noErrP class without considering the temporal dimension of the signal. For example each ErrP and noErrP signal portion of interest has in total forty two instances belonging to ErrP and noErrP classes respectively. Therefore with this approach, each signal is transformed into forty two instances of one dimensional feature vectors. After the classification result is obtained, a voting scheme is used to have the result for entire test signal, e.g. if more

[10] http://www.gtec.at/Products/Complete-Solutions/g.BCIsys-Specs-Features

than 50% data points of a signal are classified as ErrP, then the test signal is classified as ErrP signal else it is recognized as noErrP signal.

b) Dimensionality Reduction of Each Signal

In this method, dimensions or features in signal portion of interest are reduced depending upon the following scheme. For each individual signal, the data points are compared against each other and if they fall in neighbourhood of each other i.e. lie within a pre-set threshold value, then those points are considered as redundant. Redundant points are discarded keeping only one of those points or in other words they are merged to a single value. Hence the remaining data points of each signal are considered as important features and each of these points are treated as independent instances neglecting the temporal dimension. Similarly this dimensionality reduction scheme is carried out for testing data as well, i.e. it reduces data points which fall in neighbourhood of each other considering all of them as a single point. The threshold is set here as 0.01µv. Similar to the previous category mentioned above, the voting procedure is again carried out here to classify an entire test signal depending upon the classification result of all of its important constituting features.

c) Each Data Point in the Time Series Signal as a Feature

As mentioned earlier, every signal portion of interest contains forty two data points here. Each of these data points is considered as a feature of the signal and therefore each instance of ErrP and noErrP has a feature vector of forty two dimensions.

d) Addition of Temporal Feature

It is observed that the time difference between maximum and minimum of average ErrP signals are different from that of average noErrP which could be seen in Figure 22 (b). This temporal difference is considered as a feature of the signal and proportional to the difference in index of maximum and minimum value of the respective signal. Therefore, this index difference is estimated for each signal and considered as a feature along with the other forty two features mentioned earlier. Therefore these sums up to a feature vector of forty three dimensions in each instances of both ErrP and noErrP.

e) Important Features Only

Among the earlier mentioned forty three features extraction method, a technique is implemented to retain only important features of a signal and use only those selected important features during classification. Classifiers in sequence are developed based on only one feature of all signals followed by another until all the features are used in separate classifiers. Each classifier is validated using the validation data set. If a classifier contributes to at least a pre-set percentage which is set as 30%, for correct classification of both true negative and true positive, then the feature associated with that classifier is considered as biased to correct class feature. Feature might also exist that are biased to opposite class which means feature that helps in classifying the opposite class for both categories of signals instead of predicting the class they belong to. Therefore to detect possibilities of these opposite class biased features, the classifier should have an opposite class classification result above a defined value which is set as 70% for both true negative and true positive signals. Among total of forty three features including the temporal dimension, those which are biased to correct and opposite classes need to be determined. For the sake of it, a set of forty three linear classifiers are trained separately using only one feature followed by another feature of the signal. All of those classifiers are checked individually against their performances with the validation set. A classifier is considered as poor classifier if it doesn't contribute to either above 30% or above 70% for correct and opposite class classification respectively, for both ErrP and noErrP signals in validation set. The corresponding feature associated with the poor classifier is regarded not important and hence is discarded before training the final classifier to be verified with test data. Important features of average ErrP and noErrP signals are shown in Figure 22 (b).

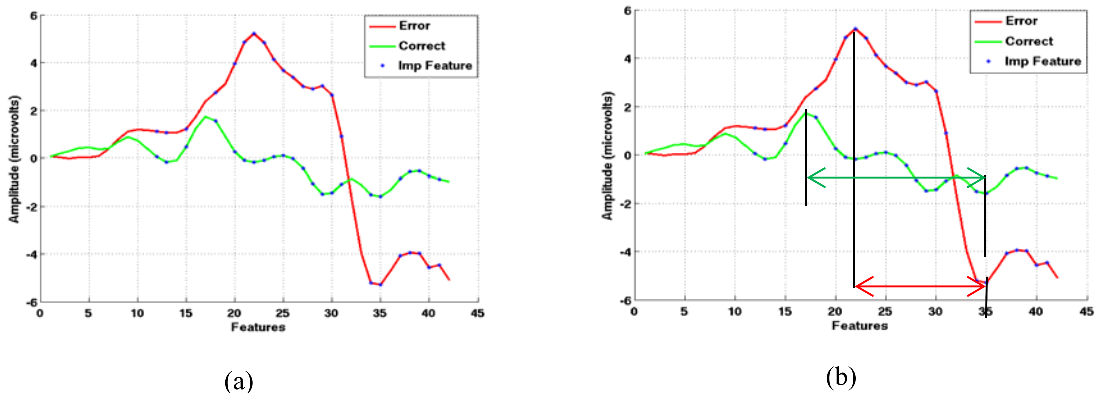


Figure 22: ErrP (Error trial) and noErrP (Correct trial) signals (a) Avg ErrP & noErrP with Important Features, and (b) Average ErrP & noErrP with Temporal Difference Feature

3.5 Support Vector Machine Classifier Training

a) Randomization of Data

True negative and true positive data are randomized separately, 10 times, in order to test the result for different set of training, validation and testing sets. The final test result is calculated as the average outcome of each randomized set of data.

b) Normalization of Data

Each feature needs to be normalized in order to keep the feature values bounded within unity, for the sake of assigning equal weight or importance to all of the features. Also keeping the values within a small range allows the kernel to occupy less memory during kernel function operations which essentially uses calculations like dot product of vectors.

c) Labelling of Data

All of the available data are labelled as -1 if it belongs to ErrP and as 1 if it belongs to noErrP.

d) Splitting of Data

The ErrP and noErrP data are split into 80, 10 & 10 ratio for training, validation and testing respectively.

e) Kernel and Kernel Parameter Selection

Among several varieties of classifiers, Linear classifier tries to linearly separate data while a Radial Basis Function (RBF) kernel or Gaussian kernel maps the limited set of features to infinite number of features in order to obtain a hyperplane separating both classes of data when the data is not linearly separable. By doing so, there are better chances of finding a separating hyper plane than finding it with limited available features. RBF classifier is generally considered having better performance due to this infinite dimension mapping, compared to other classifiers like Polynomial kernels [39]. Among various available techniques like Quadratic Programming (QP), Sequential minimal optimization (SMO) and Least Square (LS), SMO method is selected to find out the separating hyperplane. The SMO algorithm, gives an efficient way of solving the dual problem arising from the derivation of the SVM. It was invented by John Platt in 1998 at Microsoft Research [40]. SMO is widely used for training support vector machines. The publication of the SMO algorithm in 1998 has generated a lot of excitement in the SVM community, as previously available methods for SVM training were much more complex and required expensive third-party QP solvers [41].

f) Training

During the step of important features selection, only linear classifier was used as it was quicker and also had comparable results with RBF kernel. After selection step of important features, both Linear and RBF classifiers are trained using the features. In general, the performance of classifier is very much dependant on the choice of parameters [42]. In the case of RBF classifier these crucial parameters used in the kernel function are box constraint and sigma value. Initially a coarser and later a finer grid search are generally performed on these parameters. These parameters are chosen depending upon the cross-validation accuracy, after iterating on different values of the parameters. Cross-validation, sometimes called rotation estimation [43], is a technique for assessing how the results of a statistical analysis generalize to an independent data set. It is mainly used when determining the tuning parameters and also the method can estimate the bias of the excess error in prediction [44]. In k-fold cross-validation, the original sample is randomly partitioned into k subsamples. Of the k subsamples, a single sub-sample is retained as the validation data for testing the model, and the remaining (k – 1) subsamples are used as training data. The cross-validation process is then repeated k times or k folds, with each of the k subsamples used exactly once as the validation data. The k results from the folds then are averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. Classification parameters used during training of classifier are listed in Appendix B.

3.6 Classification Performance of Support Vector Machine

Test instances are recognized using both Linear and RBF classifiers separately and the performances are compared against each other. Test results for each data set generated because of randomization, are produced and the average performance is estimated. Also calculated are the standard deviations for both ErrP and noErrP classification accuracy. Mentioned below is the average result of SVM classification for different categories of iterations that is developed in this thesis work. Among the five different categories used for feature extraction mentioned earlier, the first “Each Data Point as an Instance” and the second “Dimensionality Reduction” categories have similar performances with average classification rate around 70% for both ErrP and noErrP. “Each Data Point in the time series signal as a Feature” which is third in the category list and the next “Addition of Temporal Feature” both have better performances but average classifications are below 75% for both ErrP and noErrP. The result obtained from the last category of iteration “Important Features Only” defined above, is superior to all other categories mentioned. Twenty three features are found to be important after following the procedure mentioned under “Important Features Only” category, among the forty three available features for each signal portion of interest and there is no such feature found which is biased to opposite class. The average classification results of this category are 71% with standard deviation of 12% for ErrP and 73% with standard deviation of 9% for noErrP with a linear classifier. As expected, performance seems to increase for RBF classifier with average ErrP classification of 88% with 8% standard deviation and 75% with 7% standard deviation for noErrP. The RBF classifier while considering only important features from data is the best performer among other types of feature extraction methods and choice of classifiers mentioned. The result of SVM classifier and Gaussian classifier for the same data produces ErrP and noErrP average classification accuracies as mentioned below in Table 3, for 10 fold cross-validations.

SVM				**Gaussian**	
Linear		**RBF**			
ErrP (%)	noErrP (%)	ErrP (%)	noErrP (%)	ErrP (%)	noErrP (%)
71 ± 12	73 ± 9	88 ± 8	75 ± 7	95 ± 5	72 ± 13

Table 3: ErrP Classification Rate for SVM and Gaussian Classifiers

3.6.1 Timing Performance of Support Vector Machine Classifier

The time consumed during training and validation of the linear classifier is around 70 seconds while that of RBF classifier is approximately 110 seconds, for around 300 instances of ErrP and noErrP in total. Both linear and RBF classifier produced test results within 0.1 second for a single test instance.

3.7 Future Work on Support Vector Machine Classification Method

The work could be extended to perform a BCI experiment using either P300 or SSVEP paradigm to record user intention with more number of participants and thereafter training SVM with large data set. A finer grid search could be performed in order to estimate better values of box constraint and sigma which would produce higher recognition rate of ErrP and no ErrP in the EEG data, when radial basis function (RBF) kernel is used for SVM classification. Advanced feature extraction methods from ErrP and noErrP signals in the EEG data, may lead to even better performance of the SVM classifier.

4 SUMMARY

This is the last chapter of this thesis and it concludes with the summary of the entire thesis work of part 1 (Adaptive BCI) and part 2 (Classification of Error-related Potentials).

The future generations of user interfaces need to consider uncertainties in user behaviour and should be able to adapt as per the user, in order to provide increased user satisfaction while using the interface. This study shows a novel use of machine learning technique, to be specific reinforcement learning to develop an algorithm for adaptive Brain-Computer Interface system in application to navigational tasks. The learning algorithm used here also takes care of change in behaviour of the user and outliers in user behaviour. As a future work of this learning algorithm, the reward function defined in the algorithm could be enhanced by considering cognitive load of the user. I have shown that such an adaptive BCI is indeed possible and could provide embodiment feeling to the user as if the robot body is an avatar of the user, compared to conventional fixed policy based BCI systems.

Further this thesis tackles the detection of single trial error-relate potentials in the EEG data using Support Vector Machine. As expected, it is observed that feature selection plays a decisive role for classification accuracy. SVM is found to work well, in parallel with other classifiers such as Gaussian classifier or Bayesian filter method and could have higher ErrP recognition rate for RBF kernel with tuned kernel parameters. The timing performance shows that SVM could be considered for real-time BCI applications. Successful detection of error-related potentials indicates the possibility of increasing the reliability and also improving the information transfer rates of BCI systems.

5 LIST OF FIGURES

Figure 1: Example Task 11
Figure 2: EEG Cap 11
Figure 3: SSVEP-based BCI 12
Figure 4: P300-based BCI 13
Figure 5: EEG Data Classifiers 14
Figure 6: Average ErrP (Error trial) and no ErrP (Correct trial) 15
Figure 7: Single Trial ErrP (Error trial) and noErrP (Correct trial) 15
Figure 8: Example of a Finite State Automaton 18
Figure 9: State-action Mapping 19
Figure 10: Adaptive Interface Manager and its Relation to Other Components of the Embodiment System 20
Figure 11: Process of Learning and Prediction 24
Figure 12: Simulated Robot World in Gazebo 26
Figure 13: Simulated BCI with 3 Masks Used for Testing of Fixed Policy P300-based BCI 27
Figure 14: User Behaviour 28
Figure 15: Changed Behaviour Mode 29
Figure 16: User Behaviour with Outliers 29
Figure 17: Both Outlier and Change in Behaviour 30
Figure 18: Average ErrP (Error trial) and no ErrP (Correct trial) 32
Figure 19: Single Trial ErrP (Error trial) and noErrP (Correct trial) 32
Figure 20: Support Vector Machine Example 34
Figure 21: Experimental Setup for Interaction ErrPs 35
Figure 22: ErrP (Error trial) and noErrP (Correct trial) signals 37

6 REFERENCES

[1] Niels Birbaumer, Ander Ramos Murguialday and and Leonardo Cohen, Brain–computer interface in paralysis, Current Opinion in Neurology, 21:634–638, (2008).

[2] Brain-computer interface Wiki - http://en.wikipedia.org/wiki/Brain-computer_interface .

[3] Dennis J. McFarland and Jonathan R. Wolpaw, Brain-Computer Interfaces for Communication and Control, Communications of the ACM, Vol. 54 No. 5, Pages 60-66, (2011).

[4] Pour P.A., Gulrez T., AlZoubi O., Gargiulo G and Calvo, R.A., Brain-Computer Interface: Next Generation Thought Controlled Distributed Video Game Development Platform, IEEE Symposium on Computational Intelligence and Games, (2008).

[5] Beverina F, Palmas G, Silvoni S, Piccione F and Giove S, User adaptive BCIs: SSVEP and P300 based interfaces, PsychNology Journal, Volume 1, Number 4, 331 – 354 (2003).

[6] D. Zhu, J. Bieger, G. G. Molina, and R.M. Aarts, A Survey of Stimulation Methods Used in SSVEP-Based BCIs. Computational Intelligence and Neuroscience Volume, Article ID 702357, (2007).

[7] R.M. Chapman and H.R. Bragdon, Evoked responses to numerical and non-numerical visual stimuli while problem solving, Nature, volume 203, 1155-1157, (1964).

[8] J. Polich. Updating P300: An integrative theory of P3a and P3b. Clinical Neurophysiology, 118(10), 2128-2148, (2007).

[9] L. Farwell and E. Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. Electroencephalography and Clinical Neurophysiology, 70(6):510–523, (1988).

[10] J. Wolpaw, N. Birbaumer, W. Heetderks, D. McFarland, P. Peckham, G. Schalk, E. Donchin, L. Quatrano, C. Robinson and T. Vaughan. Brain–Computer Interface Technology: A Review of the First International Meeting. IEEE Transactions on Rehabilitation Engineering, Volume 8, (2000).

[11] B. Blankertz, G. Dornhege, C. Schäfer, R. Krepki, J. Kohlmorgen, K. Müller, V. Kunzmann, F. Losch, and G. Curio. Boosting bit rates and error detection for the classification of fast-paced motor commands based on single-trial EEG analysis. IEEE Transactions on Neural Systems and Rehabilitation Engineering, Vol. 11, No.2, (2003).

[12] J. Bollon, R. Chavarriaga, J. del R. Millan, and P. Bessiere. EEG Error-RelatedPotentials Detection With A Bayesian Filter. Proceedings of the 4th International SaD1.38 IEEE EMBS Conference on Neural Engineering Antalya, Turkey, (2009).

[13] Steve Young, Cognitive User Interfaces, IEEE Signal Processing Magazine [128], (2010).

[14] Matthias Schneider-Hufschmidt, Thomas Kühme and Uwe Malinowski. Adaptive User Interfaces: Principles and Practice, (Book - 2007).

[15] Gajos K., and Weld D. S., Supple: automatically generating user interfaces. In Proceedings of the 9th international conference on Intelligent user interface, 93–100. Funchal, Madeira, Portugal: ACM Press, (2004).

[16] Pat Langley, User Modeling in Adaptive Interfaces, Seventh International Conference on User Modeling, Alberta, (1999).

[17] Alf Inge Wang and Qadeer Khan Ahmad, CAMF – Context-aware Machine Learning framework for Android, Iasted International Conference on Software Engineering and Applications (2010).

[18] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, (1998).

[19] Satinder P. Singh, Michael J. Kearns, Diane J. Litman, and Marilyn A.Walker., Reinforcement learning for spoken dialogue systems - NIPS (1999).

[20] Wang Y, Huber M., Papudesi V.N. and Cook D.J., User - Guided Reinforcement Learning for an Intelligent Environment, IEEE/RJS International Conference on Intelligent Robots and Systems , Las Vegas, NV, (2003).

[21] William D. Smart and Leslie Pack Kaelbling. Effective Reinforcement Learning for Mobile Robots, IEEE International Conference on Robotics 8 Automation Washington, DC, (2002).

[22] Jing Michelle Liu and Raphael Hoffmann. Zhiphone: A Phone that Learns Context and User Preferences, CSE567 class project, University of Washington, (2005).

[23] J. Andrew Bagnell and Jeff Schneider , Autonomous Helicopter Control Using Reinforcement Learning Policy Search Methods, International Conference on Robotics and Automation, (2001).

[24] DiGiovanna J., Mahmoudi B., Fortes J., Principe J.C., Sanchez J.C., Coadaptive brain-machine interface via reinforcement learning, IEEE Transactions on Biomedical Engineering, Vol. 56, No. 1, (2009).

[25] R. Chavarriaga, P.W. Ferrez, and J.del R. Millán. To err is human: learning from error potentials in brain-computer interfaces, 1st International Conference on Cognitive Neurodynamics, Shanghai, China, (2007).

[26] Jaeyoung Park, Kee-Eung Kim, and Sungho Jo. A POMDP Approach to P300 Brain-Computer Interfaces, International conference on Intelligent user interfaces, New York, USA, (2010).

[27] Xavier Perrin, Ricardo Chavarriaga, Francis Colas, Roland Siegwart and José del R. Millán. Brain-coupled interaction for semi-autonomous navigation of an assistive robot - Robotics and Autonomous Systems, Journal of Robotics and Autonomous Systems archive Volume 58 Issue 12, Pages 1246-1255, (2010).

[28] Nathan Eagle, Alex (Sandy) Pentland., Reality mining: sensing complex social systems, Journal of Personal and Ubiquitous Computing, Volume 10 Issue 4, Pages 255 - 268 (2006).

[29] Jennifer Boger, Jesse Hoey, Pascal Poupart, Craig Boutilier, Geoff Fernie, and Alex Mihailidis. A Planning System Based on Markov Decision Processes to Guide People with Dementia Through Activities of Daily Living, IEEE Transactions on Information Technology in Biomedicine, Vol. 10, No. 2, (2006).

[30] P. W. Ferrez and J. d. R. Millán, "You are wrong! - Automatic detection of interaction errors from brain waves," In Proceedings of the 19th International Joint Conference on Artificial Intelligence, Edinburgh, UK, (2005).

[31] Jean-Marc Bollon, Ricardo Chavarriaga, Jose del R. Millan† and Pierre Bessiere. EEG Error-Related Potentials Detection With A Bayesian Filter, IEEE EMBS Conference on Neural Engineering Antalya, Turkey, (2009).

[32] P. W. Ferrez and J. d. R. Millán, "Error-related EEG potentials generated during simulated brain-computer interaction", IEEE Transactions on Biomedical Engineering 55, 923–929, (2008).

[33] Staffan Bengtsson, Detection and prediction of lane-changes: A study to infer driver intent using support vector machine (A study to infer driver intent using support vector machine), Master of Science Thesis, KTH Stockholm, Sweden (2012).

[34] Laura Auria and R.A. Moro, Support Vector Machines (SVM) as a Technique for Solvency Analysis - Discussion Papers, German Institute for Economic Research, DIW Berlin Discussion Paper No. 811, (2008).

[35] Andrew Ng, Stanford Machine Learning, CS229 Lecture notes, (2012) - http://cs229.stanford.edu/notes/cs229-notes3.pdf

[36] Early detection of plant diseases and weeds with Support Vector Machines, DFG Research Training Group (Graduiertenkolleg) 722, International Conference Precision Agriculture, Denver, USA, (2010).

[37] Schalk, G., Wolpaw, J. R., McFarland, D. J., and Pfurtscheller, G., "EEG-based communication: presence of an error potential". Clinical neurophysiology□: official journal of the International Federation of Clinical Neurophysiology, 111(12), 2138-44, (2000).

[38] X. Wang and K.K. Paliwal. Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition. The Journal of the Pattern Recognition Society 36, 2429 – 2439, (2003).

[39] Andrew K. Chan and Cheng Peng, Wavelets For Sensing Technologies, Page 184, (Book – 2003).

[40] John C. Platt, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, Microsoft Research, Advances in Kernel Methods – Support Vector Learning (1998).

[41] Ryan Michael Rifkin, "Everything Old is New Again: a Fresh Look at Historical Approaches in Machine Learning", MIT, Doctoral Dissertation, Sloan School of Management, (2002).

[42] O. Chapelle, V. Vapnik, O. Bousquet and S. Mukherjee. Choosing Multiple Parameters for Support Vector Machines, Journal of Machine Learning, 46, 131–159, Kluwer Academic Publishers, MA, USA, (2002).

[43] P.A. Devijver and J. Kittler. Pattern Recognition: A Statistical Approach, Prentice- Hall, London, GB, (Book - 1982).

[44] M. Tsujitani, Y. Tanaka, Cross-Validation, Bootstrap, and Support Vector Machines, Advances in Artificial Neural Systems Volume, Article ID 302572, (2011).

Appendix A: Markov Agent for Training and Testing of the Algorithm

```
begin
   Define number of actions, trajectory length          // Number of training instances
   Initialize current state, state→action probabilities
   Calculate cumulative state→action probabilities

    while   training index < trajectory length   do
              Generate random number                    // Normalize the random number

         while   action index < number of actions   do
            if   random number >  cumulative probability (current state → action)   then

                 if      random number = 1   then   next action  ← action index
                 else   next action ← action index + 1
                 end if

            else
                          next action ← action index
            end if
         end while
                          trajectory [training index] ← next action
         if      action = low level   then   do nothing
         else   current state ← action
         end if

    end while
end
```

Appendix B: Parameters for Support Vector Machine Classification

Karush-Kuhn-Tucker (KKT) violation level	0.01
KKT Tolerance Limit	0.01
Maximum Number of Iterations	20000
Kernel Catch Limit	10000
RBF Sigma	29
RBF Box Constraint	2
Matlab Version (svmtrain)	R2011b
Operating System	Ubuntu 10.10