isvr Institute of Sound and Vibration Research

Mouse Control and Arm Movement Coordination

F. Dohnal, E. Rustighi and B.R. Mace

ISVR Technical Memorandum No 980

December 2008

University of Southampton

SCIENTIFIC PUBLICATIONS BY THE ISVR

Technical Reports are published to promote timely dissemination of research results by ISVR personnel. This medium permits more detailed presentation than is usually acceptable for scientific journals. Responsibility for both the content and any opinions expressed rests entirely with the author(s).

Technical Memoranda are produced to enable the early or preliminary release of information by ISVR personnel where such release is deemed to the appropriate. Information contained in these memoranda may be incomplete, or form part of a continuing programme; this should be borne in mind when using or quoting from these documents.

Contract Reports are produced to record the results of scientific work carried out for sponsors, under contract. The ISVR treats these reports as confidential to sponsors and does not make them available for general circulation. Individual sponsors may, however, authorize subsequent release of the material.

UNIVERSITY OF SOUTHAMPTON

INSTITUTE OF SOUND AND VIBRATION RESEARCH

DYNAMICS GROUP

Mouse Control and Arm Movement Coordination

by

F. Dohnal, E. Rustighi and B.R. Mace

ISVR Technical Memorandum No: 980

December 2008

Authorised for issue by
Professor M. J. Brennan
Group Chairman

Contents

Acknowledgement

No accomplishment of any significance is achieved solely by the efforts of one person.

(*author unknown*)

First of all I would like to thank Prof. Brian Mace for having this interesting idea of introducing a bias angle in the characteristics of a computer mouse. Most of us who use a Windows operating system pressed accidently (probably during a copy-paste task) the key combination , Ctrl+Alt+Arrow Key, rotated the screen display by a multiple 90° and struggled to reset the screen rotation. A straightforward rescue is to navigate the mouse pointer counterintuitively towards the restart button. While many of us try to forget this incident, it was then when Brian had the idea to introduce a rotation of a pointer device deliberately and to investigate its effect on the performance which led to this exciting project. Further I would like to thank Dr Emiliano Rustighi for discussions and support throughout the work.

Early in this project, it became clear that besides the effect on the performance, the brain's activity is an important measure for possible future applications. Here, I would like to thank Dr Cristopher James for lending me equipment for a single channel EEG measurement and his former PhD students Suogang Wang and Disha Gupta for giving me advice on the proper use of the equipment and the electrodes. Finally, I would like to thank Scott Notley for giving me advice on how to attach electrodes properly by determining their impedance, their dynamic resistance.

The financial support of the EPSCR Platform Grant in Structural Acoustics (EP/E006450/1) is gratefully acknowledged.

1 Motivation

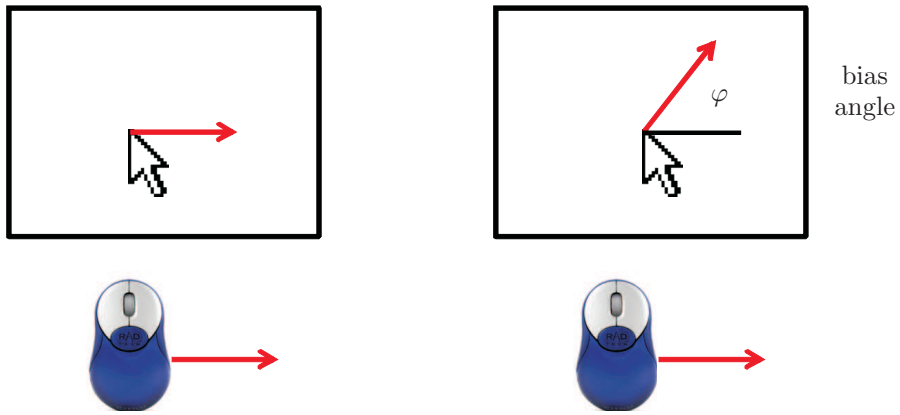
The general motivations behind this project are twofold. First, there is interest and demand to develop engineering models of neuro-muscular activity, the interplay between muscles and their control. As a consequence of such models or as a second motivation, the development of concepts and the investigation of applications to rehabilitation are of interest. This work tries to contribute first steps in both of these major aims. By altering the characteristic of a conventional PC-mouse, performing an unfamiliar task, the motion dynamics (trajectories) and the consequence on the activity in the brain's motor centre are analysed.

2 Introduction

A pointer device is a popular device in human sciences. It is used to analyse goal-directed movements under environmental changes, to reveal the correlation between eye and hand coordination, to indicate disorders in children by analysing tracking abilities, stating risk factors for musculoskeletal disorders [1] or even in security applications [2]. Here we consider control of a computer mouse.

A motivation for the chosen setup of this work are studies on goal-directed arm movements and on visual feedback [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 15] in which disturbances in the eye-hand coordination in e.g. children or rehabilitation patients were investigated. Alterations of the visual feedback have been investigated in many studies. In the current study, the visual feedback is altered before each task and not during the task as in [9].

The main idea of this short research project is to investigate the consequence of altering the characteristic of a computer mouse pointer. The trials designed and performed consist of sitting in front of a PC and operating a conventional optical PC-mouse for different coordination tasks. Normally, this device copies the hand motion to the pointer symbol displayed on the screen, see sketch in Fig. 1a. The range of motion depends on the sensitivity settings of the mouse driver but the direction is copied accordingly. In the present trial, a rotation in the pointer symbol's motion is included. This means that if the mouse device is moved, for instance, to the right, then the pointer symbol is moved in a rotated direction by introducing a bias angle φ, see Fig. 1b. The aim of this trial is to analyse the influence of this rotation on the motion dynamics and the brain activity.



(a) Normal (conventional) pointer movement

(b) Rotation in pointer movement

Figure 1: Alteration of the mouse pointer characteristic.

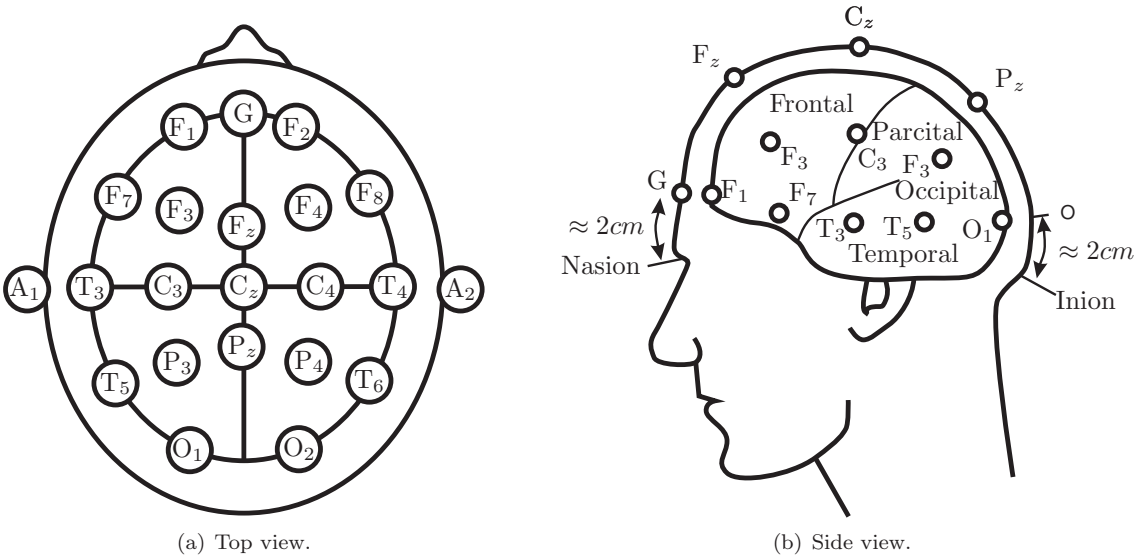
The brain's activity in the motor centre is measured in addition to the motion trajectories of the pointer. This activity is identified by measurements of the brain's electrical activity, the electroencephalograph (EEG). The first measurements of EEG on humans was demonstrated by Hans Berger, who named this electrical activity the 'Elektrenkephalogramm'. Berger was able to determine that EEG was related to activity within the brain and to rule out other physiological activity such as cerebral pulsations, cerebral blood flow, blood flow through scalp vessels, heart rate activity, muscle activity, eye movements and electrical properties of the skin. He was one of the first to suggest that periodic fluctuations of the EEG might be related in humans to cognitive processes such as arousal, memory and consciousness. An exciting feature of our brain is that the EEG changes qualitatively rather than quantitatively. For example, if a test person moves from a relaxed state to a state of increased activity, Berger noted that the EEG did not increase in amplitude but rather in its wave forms; in its frequency content. At a time where signal amplifier were not very efficient, Berger could identify two different EEG wave forms, the α- and β-activity, with α being associated with cortical inactivity and β with cortical activity. Today, the α-waves are also called Berger-waves.

An electroencephalogram (EEG) is basically a recording of the brain's electrical activity. The analysis of electroencephalogram signals is widely recognized for providing insights into brain activity level assessment. In this pilot study, a single channel EEG is recorded at the centre point Cz, see Fig. 2b (the letter z stands for *Zentrum* the German word for centre). Epochs between 1 and 30 seconds were recorded, depending on the difficulty of the task. Most of the relevant information is contained in the frequency domain. The brain's activity accessible via EEG measurements is classified in several frequency bands, typically, in a range of up to 40 Hz. The frequency limits of commonly used bands are listed in Table 1. However, these limits should be seen as guidance and may vary from person to person. Increased levels of the bands α and ϑ are an indication of sleepiness, drowsiness and onset of sleep [17].

An EEG signal is recorded with electrodes attached conductively to the scalp and represents the summation of the activity of thousands of neurons in the brain. After passing through layers of fat, bone and cerebrospinal fluid, these currents are summed and contribute to the generation of EEG voltages. Its analysis gives a quantitative measure of the electrical activity of the brain area at the electrode's location. The pattern of activity changes with the level of a person's arousal - if a person is relaxed then the EEG has many slow waves; if a person is excited then the EEG has many fast waves. The EEG is used to record brain activity for many purposes including sleep research and to help in the diagnosis of brain disorders, such as epilepsy. The most commonly used system for placement of electrodes is the `10-20 system` [18], which is based on the brain regions and the anatomical landmarks on the scalp, see Fig. 2. The distance between the nasion, the point between the forehead and the nose, and the inion, the bump at the back of the skull is divided into 10% and 20% parts. The 10-20 system is based on the relationship between the location of an electrode and the underlying area of cerebral cortex. Each point on in Fig. 2a indicates a possible electrode position. The points are labelled according to the adjacent lobes in Fig. 2b (frontal (F), temporal (T), occipital (O) and parcital (P). C stands for central. Even numbers (2,4,6,8) refer to the right hemisphere and odd numbers (1,3,5,7) refer to the left hemisphere. The z refers to an electrode placed on the midline. Also note that the smaller the number, the closer the position is to the midline.

The amplitude of a normal adult EEG is about 10 to 100 μV when measured on the scalp. A differential amplifier, that amplifies the voltage between one active and two reference electrodes, is needed for recording. In the present study a digital amplifier is used with a sampling rate of 250 Hz. In total three electrodes need to be attached to the scalp for a single channel measurement. One electrode is placed at the central electrode placement Cz which is located within the motor activity centre. Often, the placement G is used as a reference placement. This introduces electrooculographic (EOG) artifacts from eye movements. Since operating a PC mouse over the whole monitor range causes frequent eye movements, the decision was made that the remaining two electrodes that serve as reference channels (measuring mainly noise) are fitted behind each ear at the placements A1 and A2.

In general, there are four types of electrodes that can be used: reusable disks, caps with disks, adhesive



(a) Top view.

(b) Side view.

Figure 2: 10-20 system EEG

Table 1: The frequency bands of EEG signals.

name	frequency limits	associated with
δ	0.5 - 4Hz	deep sleep
ϑ	4 - 8 Hz	drowsiness, hypnosis, trance
α	8 - 13 Hz	relaxed, alert state of consciousness states (Berger's wave)
β	13 - 30 Hz	active, busy or anxious thinking, active concentration

gel electrodes and subdermal needles. In this study reusable disks with conductive paste (TEN 20) and abrasive paste are applied. Abrasive paste is used to remove dead (poor-conductive) skin from the scalp. Since a re-useable product was chosen, different measures against cross-infection has to be introduced. First, after a test trial, remaining conductive paste have to be removed thoroughly from the electrodes by using tissues and cotton wool. Secondly, the electrodes are cleaned with alcohol swabs. All cleaning products are disposable. The only re-usable product are the electrodes, which are sterilised before each experiment. Hand washing is obligatory. The abrasion is performed under low pressure, so that under normal circumstances no blood will be drawn, even a red skin is highly unlikely. If, for any unforeseen reason, blood will be drawn, the preparation for the experiment has to be aborted immediataly and no electrodes should be attached. It is important to point out that the abrasive paste is solely used to remove dead skin on the surface, and not to penetrate the upper skin layers.

2.1 Tasks definition

Each subject is given various tasks to perform. These involve moving the mouse is some prescribed manner. A motivation for designing the different tasks was the study in [13] showing that goal-directed and pointing movement can be separated into direction and distance control. Based on this study, three different tasks have been designed that should stimulate a different level of interaction of these two control mechanisms.

Task1 – moving towards target
For this task the test person was asked to move towards a target as fast as possible. A sketch of the screen that a test person would see is shown in Fig. 3. The actual screen has a grey background colour and smaller symbols. The target in the right top corner is fixed and the mouse cursor starting in the left bottom corner should be moved as quickly as possible towards the target. When the target is reached, the target is shifted immediately towards the bottom left corner and the task continues but now the pointer should be moved downwards. To implement this shifting of the target, the subject is asked to click on the target. The target is reached six times in total, which means moving the pointer back and forth six times; three times upwards and three times downwards. This kind of goal-directed movement is coordinated by two *subsequent* types of control mechanisms. During the first part of the task, the fast movement of the cursor, the direction control mechanism is in operation, while distance control is switched on in the final part.

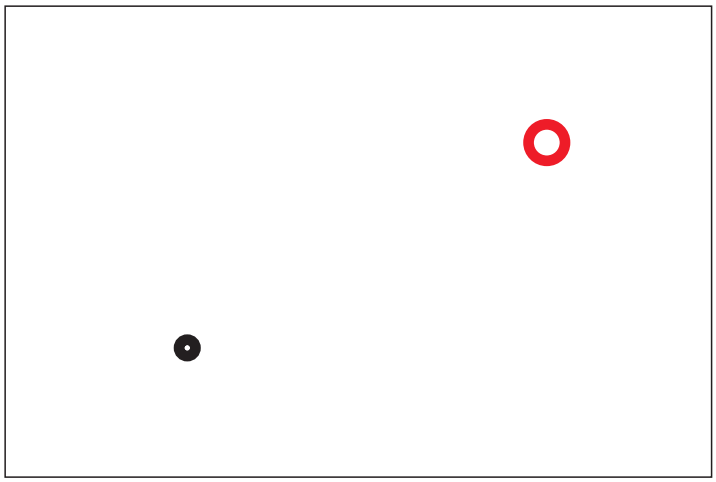


Figure 3: Task1 – moving a pointer (small circle) towards a target (big ring) – subsequent direction and distance control.

Task2 – tracking a linear path
During this task the test person is no longer free to choose an individual path but is asked to follow a given line, see Fig. 4. As in Task 1, the target is shifted back and forth once the target is reached until a total number of six trainings is performed consisting of three upwards and three downwards motions. In this setup, direction and distance control operate *simultaneously*, since the distance to the line is kept minimal at all time.

Task3 – tracking a circular path
The final task is an alteration of Task 2 but now the path becomes circular, see Fig. 5. In addition to simultaneous direction and distance control, this task enforces a continuous motion of the pointer without any back and forth motion. Six circumnavigations need to be completed.

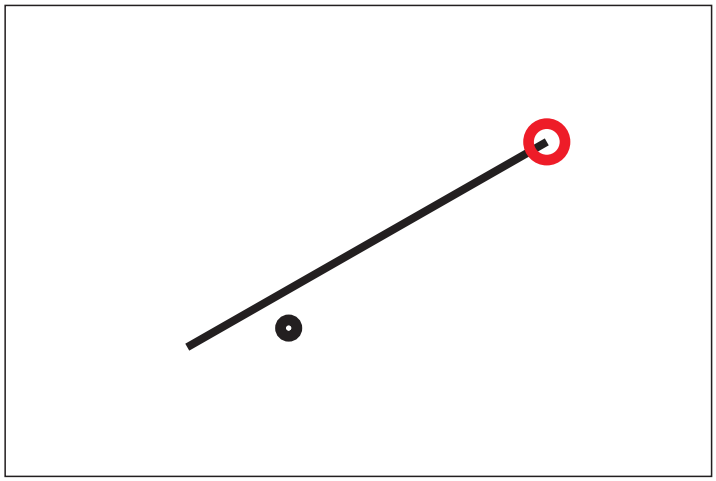


Figure 4: Task2 – tracking a line towards a target – simultaneous direction and distance control.

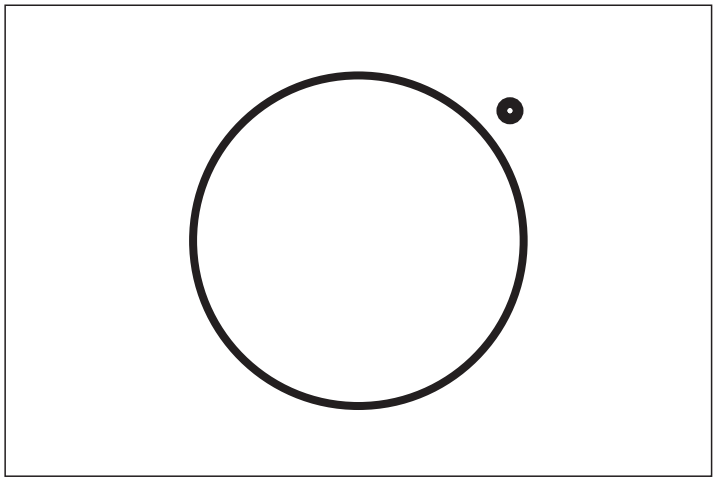


Figure 5: Task3 – tracking a circle – simultaneous control and continuous motion.

3 Measurements

3.1 Measurement preparation

Two setups were tested during the preparation for this trial:

1. Guger Technologies: amplifier Biosignal (model 2000.06.02) powered by Akkupack (model 2000.06.01a),
2. BIOPAC Systems: amplifier MP100 with module ERS 100C powered by a DC 12 V power supply; data acquisition via USB connection and software `AcqKnowledge` (ver 3.9.0).

Additional equipment: conductive EEG paste (Ten 20, 8.0 z), abrasive gel (Green Prep), cotton wool, alcohol swabs, patience.

Setup 1
This setup allows measurements of up to four EEG signals, each of them consisting of three electrodes (green/reference, yellow/ground and red/signal). The unit has one output, a mono audio plug, for each EEG measurement. Two options exist to acquire data with this equipment. The first one is to measure the audio output via the sound card using the `Matlab` command `analoginput('winsound')`. In this case, new scripts need to be created for the analysis of the data and the data are digitalised during processing in the sound card. The second option is to use the analog inputs of the BIOPAC System of Setup 2. This system is connected via a USB lead to the PC through which data is sent to and processed by the software. In this way, some of the analysis procedures common for EEG measurements are readily available through the use of the commercial software `AcqKnowledge`. The drawbacks of both options are:

1. Very long charge time (24h) of the battery compared to the usage time of approximately 1 h (12 V/0.6 A). Some of the tests performed in this trial lasts longer than an hour.
2. Human ears are said to be sensitive to frequencies between 20 Hz and 20 kHz. Therefore many (all?) sound cards are made to filter out signals below 20 Hz and above 20 kHz. Since much of the EEG signal is below 20 Hz, a lot of it would be lost if captured with a sound card directly. The lower cut-off frequency of the filter lies in the middle of the frequency range of the brain's β-activity, see Table 1. Hence, acquiring data via the sound card is technically possible but the frequency content of interest might be lost.
3. The commercial software `AcqKnowledge` can be used for the expense of needing both setups, Setup 1 and 2, simultaneously. No data loss occurs here. The only limiting factor is the short measurement time of 1 hour.

Due to these reasons, Setup 1 was not used within this trial.

Setup 2
This setup uses a DC power supply of 12 V and has no need for a battery. Up to 16 EEG measurements could be performed with this unit but a separate signal amplifier is needed for each set of electrodes. Only one amplifier was accessible for this trial. The unit is connected by a USB lead to the PC. The whole unit (memory, chips) can be accessed via the corresponding commercial software `AcqKnowledge`. The maximum acquire length is 20670 k samples and the maximum sample rate is 70 k samples/sec.

The software `AcqKnowledge` offers a convenient way to acquire a single measurement via a graphical user interface (GUI) by clicking on a start and stop button manually. Within this trial about 250 measurements need to be acquired for each test person which requires an automated way to collect data. Here, it is advantageous if AcqKnowledge is set to append mode. Different ways to acquire data automatically were tested:

1. Using `AcqKnowledge`:
   The data acquisition can be started by an external trigger (a switch pressed by the user, or a generated trigger signal). Nevertheless, the software requires to preset the acquisition time beforehand. If the time is chosen too short the acquisition is stopped before the test subject finishes a task. This drawback can be avoided by setting the fixed acquisition time to a sufficiently high value. Different ways exist to start an acquisition:

   a) A manual switch that is pressed by the user which connects the channels GND D and Trigger.

   b) Generating a trigger signal and feeding this signal as an external trigger signal into the Biopack unit (again channels GND D and Trigger). The sound card in connection with `Matlab` could be used to generate the trigger signal. The limiting factor here is that a sound card can generate a signal with a maximum output of +1V while +5V would be needed for the trigger signal. This could be overcome by designing a small battery supplied circuit of a low voltage amplifier or transformer.

   However, the acquisition is stopped either when the stop button is clicked manually or the preset measurement time is reached. The first option needs an interaction by the test subject or another operator and the second option would require to know the individual time for each task of each test subject. It is a significant limiting factor that there is no option within the software to stop an acquisition automatically by a trigger signal.

2. Using `AcqKnowledge`:
   In this arrangement, the software `AcqKnowledge` is solely used to start measurement, which can be triggered automatically. The data acquisition happens via a mono audio lead that connects the analog output of the Biopac unit and the sound card. The limiting factor here is that the maximum input into the sound card is ± 0.7 V. Again, some transformer would be needed here. Another option would be to use a commercial acquisition card (e.g. from National Instruments).

3. Using `AcqKnowledge`:
   A script software enables manual user inputs, like a mouse click, to be simulated. In this sense, option 1 can be automated by executing one macro to start data acquisition and a second one to stop the acquisition. The crucial point here is that the script should operate in the background while the Matlab GUIs, in which the performance of the test subjects are tested, are operated on top of the screen. This background operation avoids screen flickering. Different script softwares exist to program macros. However, most of them, e.g. the internal `MS Windows` script software, cannot be used since they require the automated window to be in the foreground. Also important to consider is that the pointer position is not altered during the script operation since the test subject should be in full control of the pointer position during the whole task. In order to avoid screen flickering or change in mouse position the free (under the GNU licence) script software `AutoIt` is used to simulate manual user inputs and, thus, to generate and send click events to objects in the `AcqKnowledge` GUI. The important thing here is that the software enables click events to be sent without the need to access the pointer at all.

4. No `AcqKnowledge`:
   Acquire data via the digital output of the Biopac unit, e.g. via the sound card. This does not work since digital output generates only reference signals and not the actually measure signal.

5. No `AcqKnowledge`:
   Develop a `Matlab` CMEX code based on the commercial C++ code `Bhapi` provided by Biopac Systems. This is the most efficient option if the measured data needs to be processed further or synchronised with other calculation. Unfortunately, this commercially available code (libraries and source code) works only for the newer Biopac Systems model MP150 and not for the actual model MP100. So this is not an option.

Since the best option 5 is not available, option 3 was implemented by using the script software `AutoIt`. This software enables a mouse click event to be sent to window objects without the parent window being on top (on focus/activated) or the need to access the actual pointer. Thus, a mouse click can be generated without the mouse being actually moved and without the need to see the window of the software in operation. The script code is simply

```
1   ControlClick("AcqKnowledge - ","","LiteButton13")
```

LiteButton13 is the name of the start/stop button of the `AcqKnowledge` GUI window. A spy software comes with `AutoIt` to gather names of user objects within a GUI. The script is compiled to an executable `MS Windows` software with the file extension "exe". This executable is called within `Matlab` via the syntax "!name" at the start and end of each measurement task. More sophisticated scripts have been developed to synchronise the measurement of the position of the pointing device and the brain activity and to export the brain signals to readable `Matlab` files.

3.2 Measurement setup

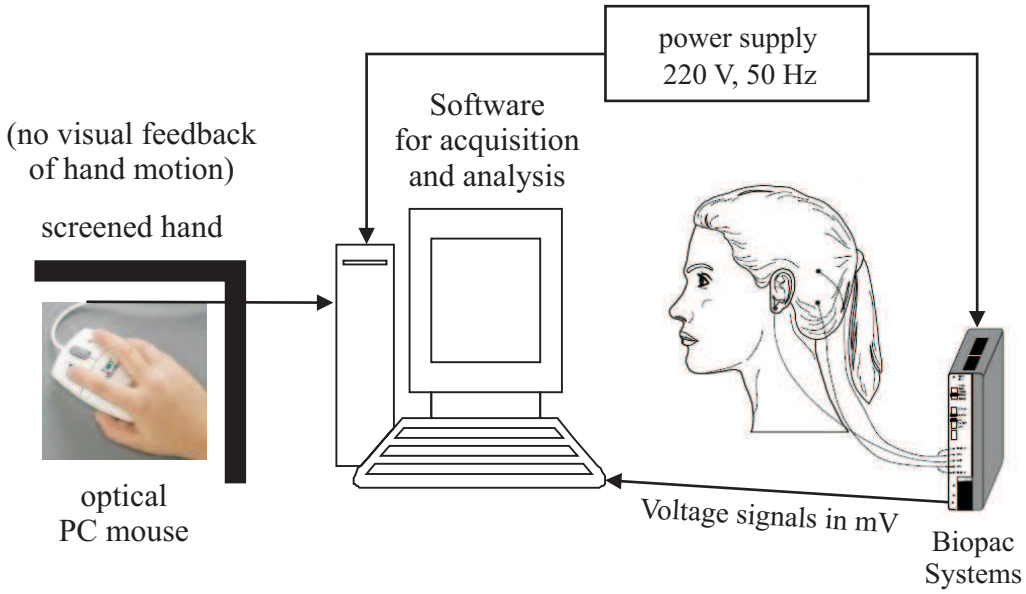
The chosen setup for measuring the pointer position and brain activity is sketched in Fig. 6. A conventional optical computer mouse was used and the trajectory of its pointer symbol was acquired on the PC. In addition to this motion dynamics, the brain activity of the test person was monitored using the commercially available package Biopac Systems. The hand was screened to exclude visual feedback of the hand motion so that the subjects relied entirely on the motion of the pointer symbol on the screen.

3.3 Measurement series

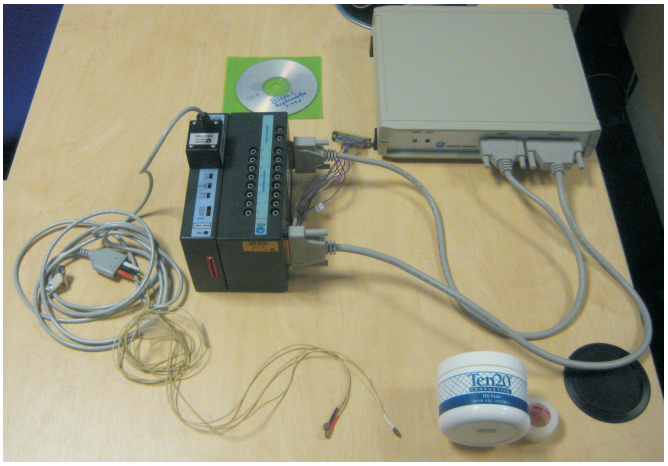
A good contact of the electrodes on the scalp is crucial for a good EEG signal. An impedance meter helps to achieve this by being able to check the contact quality before plugging them into the measurement equipment. For a good contact the impedance lies between 2 and 8 kΩ. Thorough cleaning of the application spot and use of a peeling-gel (Green Prep) helps to achieve a low impedance.

Typical values of the measured EEG signal lie between 10 and 100 μV and need to be amplified by a factor (gain) of up to 50000. The electrodes used in this trial have contact wires of a length of 1.25 m. While the electrodes measure the potential at the attachment points, the wires pick up all sorts of electromagnetic pollution. This induced noise is generally much larger than the actual signal (the signal to noise ratio can vary between 1/1000 and 1/100000). This is the reason why three electrodes are used, one at the location of interest (active) and two reference electrodes. Assuming that all electrodes pick up about the same noise, the amplifier subtracts the two signals measured between the reference electrode and the active electrode so that only the EEG signal is left. The noise level in the electrodes depends highly on their impedance. To assure that the substraction works properly, the contact wires should lie in parallel and their impedance should, ideally, match. In fact, matching both impedances is even more important than their actual values.

Measurements of EEGs were acquired using the Biopac systems unit with the ERS 100C module from the Signal Processing and Control Group, ISVR. Three electrodes were attached at the scalp: one at the motor centre at position `Cz` and one behind each ear as reference. A good contact at the position `Cz` was especially hard to achieve for test persons with dense and strong hair. Placing the reference electrodes behind the ears avoids ocular artifacts but introduces muscular artifacts from head movements (especially at times when the target was shifted) and jaw movements caused by nervous chewing or speaking. Another source of artifacts was lifting an arm in order to reposition the computer mouse if the movement stroke felt too long. The range of hand motion depends on the sensitivity of the computer mouse settings and was set to 10 cm for this trial. The impedance of the attached electrodes was measured at a fixed frequency of 30 Hz using a battery supplied impedance meter from the Audiology Group, ISVR.



(a) Sketch of measurement setup.



(b) Biopac Systems for EEG measurements with electrodes, conductive paste and software CD.

Figure 6: Measurement setup.

The following settings were chosen for the ERS100C module:

- Gain: 10.000
- Upper frequency: 10 kHz
- Notch filter: OFF
- Lower frequency: 1.0 Hz
- Sampling frequency: 500 Hz

The data of twelve subjects were collected within five days:

1. subject (frequent user = daily office computer user): Performing first tests on suitability of scheduled tasks; 6 tasks with 6 repeats, 25min (including attachment of electrodes); visible jaw patterns → no speaking during tests!
2. subject (frequent user): First subject performing full trial; 17 tasks with 6 repeats, 45min → really fast!
3. subject (frequent user): 17 tasks with 6 repeats, aborted trial after 1h 20min → skipping two tasks!

Subsequently 15 tasks with 6 trainings were performed by each subject and the impedances of the attached electrodes at 30 Hz were measured.

4. subject (frequent user): 1h, noise at 10:46, speaking at 11:09, (6kΩ, 5kΩ)
5. subject (heavy user = computer games): subject is not sure about concentration, 10min for attaching electrodes, 40min for tasks, (7kΩ, 8kΩ)
6. subject (heavy user), **data loss**: subject is not sure about concentration, 15min attaching electrodes, 35min for tasks, 13:31 before Task 10 – target disappeared, (3kΩ, 8kΩ) (data set removed permanently by virus protection software due to `AutoIt` background activities! → added `AutoIt` to "good" software list)
7. subject (frequent user): 8min attaching electrodes, 52min for tasks, reattached electrode at task 4 (strong sweating behind right ear), (2kΩ, 3kΩ)
8. subject (heavy user): 8min attaching electrodes, 19min for tasks, (3kΩ, 4kΩ)
9. subject (frequent user): 8min attaching electrodes, 27min for tasks, Subject's details: not sure about concentration, 10:20 speaking (2kΩ, 3kΩ, the best values so far)
10. subject (frequent user), very dedicated: 12min attaching electrodes, 43min for tasks, (10kΩ(very dense hair), 6kΩ)
11. subject (frequent user), very nervous, impatient, stopped: 14min attaching electrodes, stopped after 1h 16min for 13 out of 14 tasks, 14:55 to 15:15 moving around and stretching, calmer afterwards, (9kΩ (due to very dense and strong hair), 6kΩ)
12. subject (heavy user), nervous but dedicated: 18min attaching electrodes, 22min for tasks (the fastest so far), (5kΩ, 6kΩ)

All subjects agreed that the tasks 8 to 10 were especially hard, especially the task following a line (variable `Pathlogic` is 1 within the `Matlab` GUI).

For evaluation the data of subjects 2,4,5,8,9,10 and 12 were used. Within this group of people, subjects 2,4,9 and 10 belong to the group of 'frequent users' while subjects 5,8 and 12 belong to the group of 'heavy users'.

4 Software implementations

The software product `Matlab` was chosen for changing the characteristic of the mouse cursor inline and to implement the three tasks defined in Section 2.1. Several data sets were measured and stored during the trial for each task and bias setting separately: the computer mouse position, the biased mouse position and the EEG. The mouse positions were acquired within `Matlab` while the EEG measurement needed the acquisition software `AcqKnowledge`. For a thorough measurement these acquisitions are *synchronised* by employing the free (under the GNU licence) script language `AutoIt`. These three software products interplay to acquire data during the trial.

4.1 Biased cursor and data acquisition

The biased cursor is realised by hiding the real pointer symbol and generating dynamic axes which hold the biased cursor symbol. This avoids loosing control over the PC if the trial needs to be aborted because of an unforeseen reason. Hence, the real mouse pointer is moved unaltered but is invisible to the subject in the task window that covers the full screen. This real cursor controls *instantaneously* the position of a moveable axes object that holds the symbol of the biased cursor. The test persons gain the impression as if the mouse characteristics change but in fact they operate a new object and not the actual cursor. It is important that the axes are invisible (`'Visible','off'`) and not distorted (`'DataAspectRatio',[1 1 1]`). The handles of the main axes covering the full screen (`hax`) and the local axes holding the biased cursor symbol (`haxcur`) are stored in the figure object in the field `UserData`.

The implementation of the biased cursor is explained in more detail here. Each task is performed in a separate figure window that covers the full screen such that the left bottom corner of the window is placed in the left bottom corner of the screen and the window bar is not visible. Consequently, the figure window is slightly larger than the actual screen display. To achieve this independently of the monitor's screen size, the figure object field `Units` is set to normalized for which the value of 1 corresponds to a full horizontally or vertically displayed screen length. This figure window holds an axes object that overlays the entire visible screen display. A sketch is shown in Fig. 7 with the corresponding Matlab variables listed in Table 2. Note that the centre of the axis object differs from the centre of the figure object since the figure is vertically longer. It is important to distinguish between different coordinate systems. For instance, while $\mathbf{b}$ is measured within the local axes coordinate system, the positions $\mathbf{a}$ and $\mathbf{e}$ are measured in screen coordinates. Four different coordinate systems need to be used: the screen coordinate system whose handle is defined by `Matlab` as 0, the figure coordinates system with its handle `hfig`, the coordinate system of the the full screen axis `hax` and the coordinate system of the moveable axis `haxcur`. The bias rotation is implemented as a rotation of the distance $\mathbf{c}$ between the actual mouse cursor and the centre $\mathbf{b}$ of the axes coordinates `haxcur`, see Fig. 7,

$$\mathbf{c} = \mathbf{e} - \mathbf{a} - \mathbf{b}. \tag{1}$$

This vector is rotated by a rotation angle φ using the rotation matrix $\mathbf{R}$

$$\mathbf{d} = \mathbf{R}(\varphi)\mathbf{c}, \qquad \mathbf{R}(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}. \tag{2}$$

In screen coordinates, the position of the biased cursor axis becomes

$$\mathbf{f} = \mathbf{a} + \mathbf{b} + \mathbf{d} - \frac{\Delta}{2} \cdot \begin{bmatrix} 1 & 1 \end{bmatrix}, \tag{3}$$

where Δ is the size of the moveable axis and biased cursor symbol. For convenience, the definition of $\mathbf{f}$ shifts the hot spot (the point which refers to a mouse click) of the cursor from the centre of the axes to its bottom left corner.

The initial position of the cursor is set to the bottom left of the window which is equal to the target position during the second training (first shifted target) and defined in `StartPos`. The pointer position is set using the screen object field (handle value 0) `PointerLocation`:

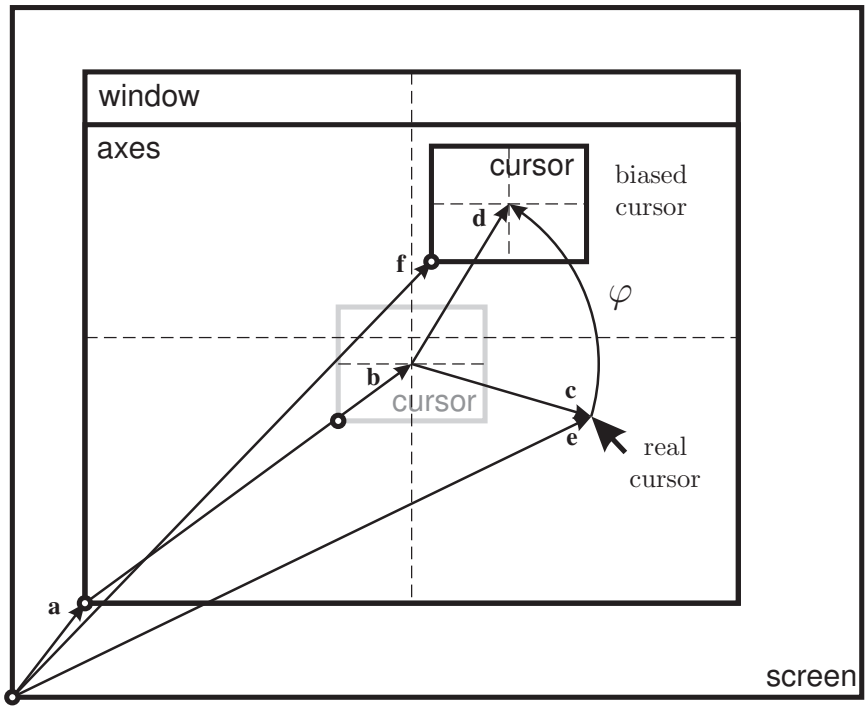


Figure 7: Biased cursor.

```
curPos = figshift+figcenter+StartPos;
set(0,'Units','Pixels','PointerLocation',curPos)
```

The bottom left corner of the figure window, `figshift` and the centre of the figure `figcenter` are determined by the actual figure size which is stored in the vector `get(hfig,'Position')`. An inline adaptation of the biased cursor position is realised by assigning a function handle to the figure object field `WindowButtonMotionFcn`:

```
  set(hfig,'WindowButtonMotionFcn',...
    {@GetCursorPos,RotMatMouse,figshift,figcenter,cur_size,haxcur})
```

This field is executed whenever `Matlab` detects a mouse movement. The developed function `GetCursorPos` reads the actual cursor position in screen coordinates and updates the biased cursor position:

Table 2: Matlab variables for biased cursor.

symbol in Fig. 7	Matlab variable
a	`figshift`
b	`figcenter`
e	`curPos`
R	`RotMat`
Δ	`cur_size`

```
function GetCursorPos(obj,evt,RotMatMouse,figshift,figcenter,cur_size,haxcur)

...
curPos = get(0, 'PointerLocation');
curPosNew = RotMatMouse*(curPos-figcenter-figshift)';
set(haxcur,'Position',[figcenter+curPosNew'-cur_size/2*[1,1],cur_size*[1,1]])
setappdata(haxcur,'CurPosData',[getappdata(haxcur,'CurPosData'),...
                                [curPos'; curPosNew; clock']])
...
```

Remember, the centre point of the cursor in screen coordinates is `figcenter` + `figshift`. The last line in the code above stores the trajectory of the conventional (but invisible) mouse pointer, `curPos` and the trajectory of the biased cursor, `curPosNew`. Note that the data is not acquired at a fixed sampling rate but asynchronously whenever a movement of the cursor is detected. Therefore, a time stamp is added to each data sample using the built-in command `clock`.

Whenever a task needs to be aborted and the task window is closed, however, before closure all acquired data sets should be stored. To avoid data loss, the built-in function `closereq` is adapted to the user function `CloseMouseControlWindow` and passed as a function handle to the figure object field `CloseRequestFcn`:

```
set(hfig,'CloseRequestFcn',{@CloseMouseControlWindow,phiVals,SavedirName,PathLogic})
```

Embracing the handle expression by {} enables arguments to be passed to the function handle. In general, using function handles is an efficient and elegant way to avoid introducing global variables. The function itself is listed here:

```
function CloseMouseControlWindow(obj,evt,phiVals,SavedirName,PathLogic)

% get handle
hfig = gcbf; hax = get(hfig,'UserData');

% load trajectory and time data from cursor object <haxcur> fields
CurPosData = getappdata(hax(2),'CurPosData');
ClickEventData = etappdata(hax(2),'ClickEventData');

% save data (target number is coded in the file name)
dummy = ['_' num2str(phiVals(1)), '_', num2str(phiVals(2))];
dummy(find(dummy == '-')) = 'm'; % account for negative <phiVals>
save([SavedirName '/Run',num2str(PathLogic),'_Phi' dummy],...
                                  'CurPosData','ClickEventData')

% stop & store EEG
!AcquireAcqKnowledge

% move system pointer to screen center
screencenter = get(0,'MonitorPositions');
set(0,'PointerLocation',screencenter(3:4)/2)

% close window
delete(hfig)
```

Before window closure by calling the built-in function `delete` with the appropriate figure handle, all stored trajectory data is loaded by `getappdata` into the function workspace. Additionally the EEG acquisition is stopped using the AutoIt-script `AcquireAcqKnowledge`. Stopping the acquisition of the EEG signal by the software `AcqKnowledge` stores the signal automatically into an initially defined file.

The function `CloseMouseControlWindow` is called by clicking the window button x or by double-clicking on the left corner of the window bar or by pressing the shortcut Alt+F4. In the unlikely event in which the window needs to be closed manually, the command `delete(gcf)` should be used.

4.2 Tasks 1 and 2

The implementations of Tasks 1 and 2 are realised in the script `Mousecontrol.m`. During these tasks a test person moves the biased cursor towards a fixed target and must click on the target once it is reached. Initially, the cursor is positioned in the bottom left part of the window and moved towards a target located in the top right part of the window. Once the target is reached after this mainly upward motion, it is shifted to the bottom left part of the window and the test person has to move downwards to reach the target and click on it, and so on. The back and forth switching of the position is implemented by switching the visibility and activation of objects.

After figure and axes generation, the linear path object, the top and bottom target objects and the cursor object are generated. Initially *all* objects are plotted independently of the task and motion direction. Then, the visibility and activation of the objects is set using object handles.

The target was chosen to be a thicklined red ring. Two concentric circle objects are generated for the top right target position and the bottom left target position, respectively. One object represents the visible red ring and one the invisible smaller ring that fits exactly within the ring and matches the background colour. While the inner ring is visible with matching background colour and clickable (`Visible on`, `HitTest on`), the outer ring is visible only (`Visible on`, (`HitTest off`)). The size of the inner ring is the same as the cursor symbol which guarantees that the test user is allowed to click on the target only if the biased curser is sufficiently close to the centre of the red target. Clicking on the red target outside of the inner ring has no effect. `htarget(1)` and `htarget(2)` are the handles of the inner and outer ring object in the top right window part and `htarget(3)` and `htarget(4)` the handles for the bottom left window part.

Initially the top target position is visible with the inner ring clickable while the bottom target position is invisible and not clickable. Once the test user reaches this target and clicks on the inner ring the top target objects are deactivated and the bottom target objects are visible with the inner ring clickable. This switching is implemented in the function `ChangeTarget`. To detect mouse clicks on the inner ring objects `htarget(1)` or `htarget(3)` their corresponding object field `ButtonDownFcn` is associated with a figure handle to the function `ChangeTarget`:

```
set(htarget(1 or 3),'Visible','on','HitTest','on','ButtonDownFcn',...
  {@ChangeTarget,htarget,hax,haxcur,TrainNum,phiVals,SavedirName,PathLogic});
```

Once the target is clicked, the target positions are switched and the coordinate and time stamp of the click event is stored in the user-defined variable `ClickEventData` in the moveable axes object:

```
setappdata(haxcur,'ClickEventData',[getappdata(haxcur,'ClickEventData'),...
                          [size(getappdata(haxcur,'CurPosData'),2);clock']])
```

This enables a separation of the training sequences during a task into `TrainNum` sequences.

Further comments on `MouseControl.m`

Syntax

```
function hfig = MouseControl(phiVals,TrainNum,SavedirName,PathLogic)
```

Herein, `phiVals` is a vector of length 2 containing the inclination of the linear path and the bias angle, `TrainNum` is the number of trainings for the fixed rotations defined by `phiVals`, `SavedirName` is the name of the directory for data storage and `PathLogic` is a switch between Task 1 (0) and Task 2 (1). The number of trainings defines how often the target is switched back and forth during the task. The function returns the figure `hfig`, a handle to the figure object wherein all axes handles and trajectory data are stored.

Cursor symbol

The switch variable CurVisible was introduced to show either (0) the real cursor symbol or (1) to generate a customised cursor consisting of two parallel lines with inclination angle equal to φ or `phiTaskData(1,?)`. A customised cursor is set by

```
set(hfig,'Pointer','custom','PointerShapeCData',cursorMat)
```

where `cursorMat` is a 16×16 matrix with `NaN`-entries for transparent pixels.

Bias rotation

Two rotations are defined in the matrix `phiTaskData` for each run: A rotation of the linear path defining the rotation matrix `RotMatLine` and the bias rotation defining `RotMat` and the position of the biased cursor.

Background

The background colour is set by the variable `axes_color` in the syntax [red green blue].

Window and axes

The figure window is generated in normalised coordinates in order to cover whole screen (no visual interruptions for user) then the units are set to pixel for data acquisition. The display of the menu bar is hidden (`menubar`, the selection of any figure object, e.g. text fields, line objects, etc., (`selected`) or resizing the figure window (`selected`) are suppressed. Only closing the window is allowed.
It is important to set the field `XLim` and `YLim` to prevent an automatic resize of the axes if the (biased) cursor moves outside of the window.

4.3 Task 3

The implementation of Task 3 is realised in the script `MousecontrolCircle.m`. The main difference to Tasks 1 and 2 is that instead of shifting a target back and forth, now the number of completed circuits needs to be determined. This is achieved by dividing the window into four quadrants that meet in the centre of the circular path which is also the origin of the axes object. The passages through the quadrants' borders are observed by observing a change in the sign of the the product of the x and y-coordinate of the biased position `curPosNew` as well as the x-coordinate only. A detailed listing is given here:

```
% how many rounds completed
if curPosNew(1)*curPosNew(2) > 0
    loground = get(h(3),'UserData');
    if loground+sign(curPosNew(1))==0
        set(h(3),'UserData',sign(curPosNew(1)))

        if loground == -1
            % update number of training runs
            Train = get(h(1),'UserData');

            % store event together with time stamp
            setappdata(h(2),'ClickEventData',[getappdata(h(2),'ClickEventData'),...
                                    [size(getappdata(h(2),'CurPosData'),2);clock']])

            % check whether training completed
            TrainNum = get(h(4),'UserData');
            if Train == TrainNum
                CloseMouseControlWindow(obj,evt,phiVals,SavedirName,PathLogic)

            else
```

```
21                  % increase training number
22                  set(h(1),'UserData',Train+1)
23                  set(h(4),'String',{'round', [num2str(Train) '/' num2str(TrainNum)]})
24              end
25          end
26      end
27  end
```

In the first and third quadrants the relation $x \cdot y > 0$ holds, which refers to the first if clause in line 2. These two quadrants are distinguished by checking the sign of the x-coordinate, sign(x), a value that is stored in the variable `loground` in line 5 and recalled in line 3. In fact line 3 recalls the value of the previous data acquisition point (sample) which is used to check passages from the first into the third quadrant in the if clause in line 4. For the first quadrant `loground` is +1 and for the third quadrant -1. For instance, if the biased cursor is initially in quadrant one and the cursor is moved within this quadrant then the variable `loground` is set to +1. Exiting this quadrant either $x \cdot y = 0$ or $x \cdot y < 0$ holds, depending on how fast a data sample is acquired. In both cases `loground` remains +1. Moving within the second or forth quadrant has no effect on `loground` due to the if clause in line 2. Entering the third quadrant fulfills the if clause in line 2. For the first data sample that fulfills this if clause `loground` is still +1 so that the if clause in line 4 is satisfied and the value of `loground` for the next data sample is set to -1 in line 5. The third if clause in line 7 filters all passages from `loground` +1 to -1, passages from the first into the third quadrant. Hence, only motions coming from the third quadrant and passing into the first quadrant are considered. This guarantees that the test user performs at least a back and forth motion between these two quadrants. For each round the time stamp is stored in line 12 similar to the click event for the Tasks 1 and 2. The number of circuits to be completed is displayed in the centre of the window as visual information for the user. Note that the direction of motion is not checked so that circuits are counted even if the test person follows the circle opposite to the desired direction.

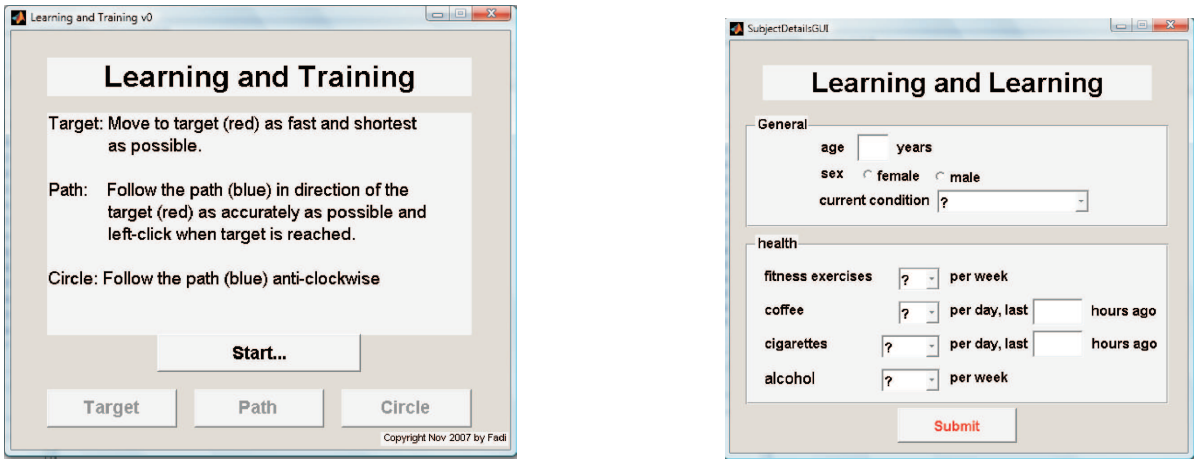
Using only two quadrants instead of four, e.g. observing a change in the sign of the x-coordinates only, then a miscount could easily occur if the cursor motion is jerky when passing through the quadrant border. Every second passage would be detected wrongly as a full round. This can be avoided by using four quadrants incorporation the x- and y-coordinate, respectively. In doing so, the location where a miscount could happen due to a jerky cursor motion is shifted to the origin of the axis. This should be taken into consideration when designing a particular path. For Task 3 a sufficiently big circle was chosen far away from the axis' origin.

4.4 User interface

All scripts are embedded within a graphical user interface (GUI) in `Matlab`. After placement of the electrodes and screening the hand and pointer, the script `MatlabControlGUI.m` is started. This *master* script initiates, coordinates, stores and finishes the whole trial. The script is set to *singleton*, which means than only one instance of the script can be executed at one time. the script calls the user interface window shown in Fig. 8a. The script consists mainly of the *opening* and *callback* functions and a `DoRun` function. The opening function holds a section *definitions* which defines the following master variables:

- `TrainNum` is a row vector of length 3 holding the number of trainings for each task. For six trainings for each task this vector is $[1, 1, 1] * 6$. Due to the implementation of shifting the target the entries in this vector are limited to even number.

- `phiTaskData` is a matrix with two rows. The first row vector hold a sequence of inclination angles of the linear path in Fig. 4 and the target positions in 3 while the second row vector holds the sequence of bias angles φ in Fig. 1b. For each entry in this vector Task 1 is evoked first followed by Tasks 2 and 3. After all three tasks were executed the next bias angle is red and again all three tasks executed. In this trial the bias angles were set to be `phidummy` =



(a) Main window `MouseControlGUI.fig`.

(b) Subject details `SubjectDetails.fig`.

Figure 8: `Matlab` GUI.

$[0, -15, 15, -30, 30, -60, 60, -90, 90, -120, 120, -150, 150, 180]$, and the matrix was defined as [30∗ ones(1, length(`phidummy`)); `phidummy`].

- `TaskNum` is a row vector of length 3 used for labeling the acquired data sets and the push buttons of the main window in Fig. 8a for each task and bias angle. This vector is set to $[1, 1, 1] *$ length(`phidummy`).
- `AcqDir` is a string defining the *absolute* location of the acquisition software `AcqKnowledge` in the syntax '"full path to file `Acq39.exe`"'. Since this string is processed outside of `Matlab` within the `C`-based script software `AutoIt`, a double quotation mark is needed to be able to reference to folder names with spaces, e.g. the folder name *Program Files*.
- `AcqFile` defines the template for acquisition of EEG data within the software `AcqKnowledge` and is defined as '"EEGRaw.gtl"'.

The total number of tasks performed is simply sum(`TrainNum`)*length(`phidummy`) which was 252 in this trial. The opening function creates a new subfolder `Run` with a dynamically generated new suffix number of type integer. All acquired data are stored in this folder and its name is stored in the object property `UserData` of the main GUI window `MouseControlGUI.fig` by set(hObject,'UserData',SavedirName). Then the function calls the `AutoIt` script `OpenAcqKnowledge` that opens the acquisition software `AcqKnowledge` and starts the template defined in `AcqFile`. Finally, the initial text displayed on the three task buttons in Fig. 8a is defined as the number of completed and to be completed tasks which is initially, 0/`TrainNum`(i) or in this trial 0/14. During this trial, the Tasks 1, 2 and 3 should be executed subsequently. To enforce this sequence and avoid any interference by the user, two out of the three task buttons were always deactivated. A *callback* function for each push button in Fig. 8a was developed to achieve the switching in activation/deactivation. Initially, all three push buttons are deactivated. The `DoRun`-function within `MouseControlGUI` updates the displayed text for the push buttons and calls the main scripts `MouseControl.m` and `MouseControlCircle.m` that execute single runs, store results and close sub-windows. The command `waitfor` is used to pause the execution of the main window while a single task is performed for a certain bias angle.

4.5 Starting the main script

The trial starts when a subject presses the `Start...` push button which calls the questionnaire `SubjectDetails.fig` shown in Fig. 8b. These questions are anonymous but should help to identify groups of users with similar performances. The selected data are stored in the local structure variable `subject`. By pressing the `Submit` push button this variable is stored as the file `SubjectData.mat` in the subdirectory `Run` and the push button for Task 1 is activated.

Pressing the active push button opens a new window and starts the first task with the first bias angle defined in the variable `phiTaskData` and sets up and starts the EEG acquisition using the `AutoIt` scripts `SetFileAcqKnowledge.exe` and `AcquireAcqKnowledge.exe`. First, Task 1 is performed by calling the script `MouseControl.m` with the corresponding settings in `phiTaskData` for a number of trainings `TrainNum(1)` and `PathLogic` equal to 0. When the test person reaches and clicks on the target for the `TrainNum(1)`-th time, the EEG acquisition is stopped and stored by the script `AcquireAcqKnowledge.exe`, the real and the biased trajectories are stored and the task window is closed. The trajectories are stored in the file

```
Run$a$_Phi_$b$_$c$.mat
```

where a stands for *PathLogic*, b for `phiTaskData(1,?)` and c for `phiTaskData(2,?)`. The EEG measurement is stored in the file

```
EEG$a$_PHI_$b$_$c$0000.acq
```

The last four zeros are required to meet the standard of `AcqKnowledge`.

After closure of the task window for Task 1, the push button for the subsequent Task 2 is activated while the other two are deactivated in the main window in Fig. 8a. Pressing the active push button calls again the script `MouseControl.m` with the corresponding settings in `phiTaskData` but now for a number of trainings `TrainNum(2)` and `PathLogic` equal to 1. Finally, completing Task 2 and storing all trajectories and EEG data, the push button for Task 3 is activated while the other two are deactivated. Finally, pressing the active push button calls now the script `MouseControlCircle.m` with the corresponding settings in `phiTaskData` for a number of trainings `TrainNum(3)` and `PathLogic` equal to 2. At the end of this task all data is stored, the task window is closed and the subsequent bias angle is red from `phiTaskData` starting with Task 1 and so on. When the last entry in `phiTaskData` is reached, the trial is stopped and displays are *Thank you*-window to indicate that the torture is over.

4.6 Data conversion

All EEG files are acquired via `AcqKnowledge` and need to be translated into the `Matlab` data format. A file is translated into `Matlab` using the export menu in `AcqKnowledge` for each data set. This conversion was automated by the `AutoIt` script TranslateAcqKnowledge.exe:

```
; Syntax: TranslateAcqKnowledge <filename>
;
; Script Function:
;   Translates AcqKnowledge file <filename>.acq to MatLab file <filename>.mat.

; If MP100 is not connected then load AcqKnowledge software only
If WinExists("MP100") Then
    ControlClick("MP100","Cancel","Button2")
EndIf

; open acq file
WinMenuSelectItem("AcqKnowledge","","&File","&Open...")
WinWait("Open")
ControlSetText("Open","","Edit1",$CMDLINE[1])
ControlClick("Open","","\&Open","Button2")
```

```
16
17 ; save as MatLab file
18 WinMenuSelectItem("AcqKnowledge","","\&File","Save \&As...")
19 WinWait("Save As")
20 ControlCommand("Save As","","ComboBox2","SetCurrentSelection",5)
21 ControlClick("Save As","","\&Save","Button2")
22 WinWaitClose("Save As")
23
24 ; wait until Matlab file is created
25 While FileExists(StringTrimRight($CMDLINE[1], 3) \& "mat") <> 1
26 WEnd
27
28 ; Finished!
```

This `AutoIt` script is called within the `Matlab` script `TranslateData`:

```
1  function TranslateData(SavedirName)
2
3  % accomplished tasks (0: moving towards target, 1: path tracking, 2: circle path)
4  PathLogic = [0,1,2];
5  PathName = {'moving towards target', 'path tracking', 'circular path'};
6  % location of EEG software:
7  AcqDir = '"C:\Program Files\BIOPAC Systems, Inc\AcqKnowledge 3.9\Acq39.exe"';
8
9  %--- EEG measurement: open acquisition software AcqKnowledge, if needed
10 eval(['!OpenAcqKnowledge ', AcqDir, ' ""'])
11
12 %--- evaluate data for each task and training number
13 for ip=1:length(PathLogic)
14     % load and plot data have different vector lengths for each test --> use cell array
15     files = dir([SavedirName,'\Run' num2str(PathLogic(ip)) '*.mat']);
16     acqfiles = dir([SavedirName,'\EEG' num2str(PathLogic(ip)) '*.acq']);
17     acqfilesdir = cd;
18
19     for itask = 1:length(files)
20         % EEG measurement: open, if not already open, AcqKnowledge software
21         % and translate acq file to mat file
22         filename = [acqfilesdir,'\',SavedirName,'\',acqfiles(itask).name(1:end-4),'.mat'];
23         if ~exist(filename) %check whether Matlab file was already generated
24                     filename
25             system(['TranslateAcqKnowledge "' ...
26                                 [acqfilesdir,'\',SavedirName,'\',acqfiles(itask).name],'"']);
27             % <system> waits until execution of command finishes, DIFFERENT to 'eval'
28         end
29         EEG = load(filename);
30     end
31 end
32
33 % end translation of acq files to mat files
34 !CloseAcqKnowledge
```

Note the use of the command system instead of eval in line 10 which pauses the execution of `TranslateData` until `TranslateAcqKnowledge` is processed. The commonly used command `eval` would execute `TranlateAcqKnowledge` too but the `Matlab` script would be resumed immediately and would inevitably break down in line 29, due to the not yet converted file defined in `filename`. A more robust conversion could be achieved by using AcqRead available on the `Matlab` file-exchange server.

5 Signal Processing and Data Analysis

In this section the cursor trajectories and the brain activity measurements are processed and analysed. First, the cursor motion is processed and two performance indices representing the total time and a characteristic distance are introduced to characterize two aspects of a test person's performance. Comments on the motion characteristics and dynamics are made. Secondly, the EEG data are filtered and power spectral densities are estimated in order to determine the total power in the frequency bands of α- and β-brain waves. This power corresponds to the subject's brain activity during a training and task.

The following files are needed for evaluation:

- masterfile.m: The main script.
- Evaluation.m: A function that evaluates the data for a single subject.
- EvaluationCursor.m: This subfunction evaluates all cursor motion data and calls the reference paths for Tasks 2 and 3 stored in LinePath.mat and CirclePath.mat.
- EvaluationEEGSlideFFTCut.m: This subfunction evaluates all EEG data and calls the subfunction ArPSDD.m to identify an autoregressive (AR) model.

Since evaluating all data took initially on average 30 minutes for each subject, figures are generated in the background and not displayed on the screen (figure('Visible','off')). This keeps the PC fully functional during evaluation. Consequently, all figures saved during this evaluation are invisible and the command sequence set(gcf,'Visible','on') has to be called to retrieve the figure window. The current evaluation time is between 2 and 5 minutes per subject, of which the identification of the AR model is the most time consuming task of this evaluation. This identification is affected strongly by the number of poles used in the algorithm.

5.1 Processing the cursor motion

The acquired PC mouse movement is stored in the variables CurPosData and ClickEventData, whose structure is shown in Table 3. The trajectory CurPosData is a matrix of size $10\times n$ where n is the number of acquired samples of one task at a fixed bias angle φ, and the click events when a target was reached is stored in the matrix ClickEventData of size $7\times$`TrainNum`, where `TrainNum` is the number of trainings for a single task at a fixed bias angle. For all tasks two performance indices are introduced related to the trajectory of the pointer motion: the total time and a characteristic distance.

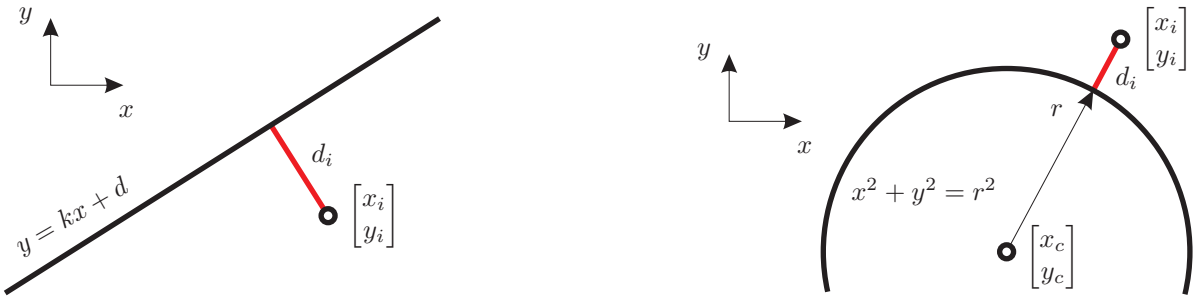
Table 3: Structure of acquired trajectory and click events.

CurPosData	
original	$\begin{bmatrix} x_i \\ y_i \end{bmatrix}$-coordinates
rotated	$\begin{bmatrix} x_i \\ y_i \end{bmatrix}$-coordinates
time stamps: year/month/day/hour/min/sec	

ClickEventData
indices i of coordinate where click occurred
time stamps: year/month/day/hour/min/sec

5.1.1 Performance index total time

For all three tasks the total time is calculated for each of the `TrainNum` trainings and each bias angle. These time values are good measures of how difficult it is to perform a particular task or training within a task. It is arguable, however, whether the total time correlates with the brain's activity, see e.g. [14], so that a proper brain measurement cannot be avoided.



(a) Shortest distance from point to line

(b) Shortest distance from point to circle.

Figure 9: Performance index for Tasks 2 and 3.

5.1.2 Performance index characteristic distance

The second performance index introduced is a characteristic distance for each training at a fixed bias angle within a task. Since different motions are expected from the subjects for Tasks 1, 2 and 3, three different distances are defined.

Total path length

During Task 1 the participants are asked to move the pointer as quick as possible towards a target that is fixed in space. For Task 1 The path towards the target can be chosen freely; the ideal case is a line connecting the initial with the target position. The total path length is introduced as a measure for their performance,

$$\mathsf{Dist}(1) = \sum_i \left| \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right|. \tag{4}$$

Shortest distance to a line

Task 2 is similar to Task 1 but now the participants are asked to follow a linear path towards the target. The sum of the shortest distances to the linear path is introduced as a measure of their performance, see Fig. 9a,

$$\mathsf{Dist}(2) = \sum_i d_i = \sum_i \left| \frac{y_i - kx_i - d}{\sqrt{k^2+1}} \right|. \tag{5}$$

The derivation of Eq. 5 is given in the following.

The shortest distance from a point i to a given line $y = kx + d$, along which the unit vector $\mathbf{a}$ lies, can be determined using vector algebra. If the vector equation of the line is given by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ d \end{bmatrix} + \lambda \begin{bmatrix} 1 \\ k \end{bmatrix}, \qquad \lambda \in \mathbb{R} \tag{6}$$

then a unit vector normal to the direction vector is defined as

$$\mathbf{n} = \frac{1}{\sqrt{k^2+1}} \begin{bmatrix} k \\ -1 \end{bmatrix}, \tag{7}$$

satisfying the scalar product

$$\mathbf{a} \cdot \mathbf{n} = a_1 n_1 + a_2 n_2 = 0. \tag{8}$$

The vector pointing from an arbitrary point p on the line to the point i is given by

$$\mathbf{p} = \begin{bmatrix} x_i - x_p \\ y_i - y_p \end{bmatrix}. \tag{9}$$

The projection of this vector to the normal vector is then

$$|\mathbf{n} \cdot \mathbf{p}| = |n_1 p_1 + n_2 p_2| = d_i, \tag{10}$$

or

$$d_i = \frac{1}{\sqrt{k^2+1}} \Big| \Big(k(x_i - x_p) - (y_i - y_p) \Big) \Big| = \frac{\Big| kx_i - y_i + d \Big|}{\sqrt{k^2+1}}, \tag{11}$$

where the relation $y_p = kx_p + d$ has been used in the last transformation.

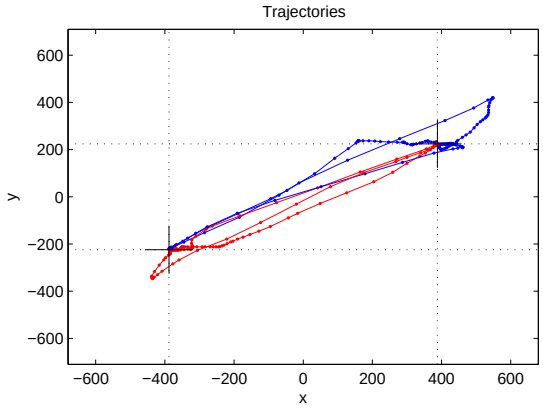
Shortest distance to a circle

During Task 3 the participants are asked to follow a circular path for a certain number of turns. The sum of the shortest distances within one turn to this path is introduced as a measure for their performance, see Fig. 9b,

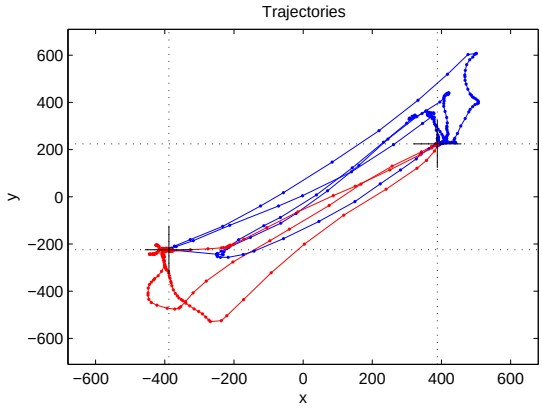
$$\mathsf{Dist(3)} = \sum_i d_i = \sum_i \left| r - \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2} \right|. \tag{12}$$

5.1.3 Motion dynamics

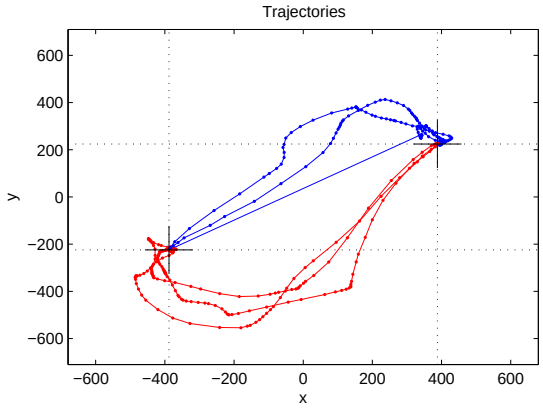
In this section a few comments on the motion dynamics are made. The trajectories of a single subject acquired during Task 1 for different bias angles φ are shown in Fig. 10. For this task the test person had to move towards a target as quickly as possible but the trajectory was chosen freely. For a conventional mouse characteristic, bias angle $\varphi = 0°$, the trajectories are shown in Fig. 10a. Blue coloured lines depicts the three upwards motions towards the target and red lines the downwards motions. Each dot represents an acquired data sample. Data is not acquired at a fixed sampling frequency but is collected whenever the cursor is moved. However, a maximum sampling frequency exists due to hard- and software limitations (CPU clocktime, operating system, other occasional background processes on the PC, etc.). Consequently, moving a cursor does not generate data samples of every single pixel doublet the cursor passes over but only a subgroup of these. Hence, closely spaced points refer to a slow cursor motion and widely spaced points to a quick motion. In Fig. 10a, the cursor is moved quickly towards the target and the trajectory either over- or undershoots the target followed by a slower motion towards the target. The first sequence relates to the direction control mechanism followed by a distance control mechanism; a subsequent direction and distance control as mentioned in Section 2.1. Introducing a small bias in rotation, see Fig. 10b for $\varphi = 10°$, widens the gap between the back and forth movements. Close to the target, the control is switched from direction control to distance control and U-shaped curvatures occur; simply a fast motion in any direction with adjusting the direction towards the end of the U-stroke. For bias rotations of $\varphi = 20°$ and more, see Fig. 10c to e, the gap between upwards and downwards movement becomes significantly wider. For increasing bias angles there comes a point where this subsequent fast movement and direction correction forms a beautiful spiral ending in the target position. These spirals are clockwise for positive rotations and anticlockwise for negative rotations. For an even larger rotation of 90° it may happen that the test person loses control of the mouse device and basically passes every point on the screen before reaching the target, see Fig. 10f. After becoming familiar with the mouse characteristics during the first training, the performance improves significantly for the remaining five trainings (three downwards and two upwards) and the trajectories and spirals become smoother. All trajectories in Fig. 10 support the hypothesis in [13] of two different types of control being in operation during goal-directed cursor motion.



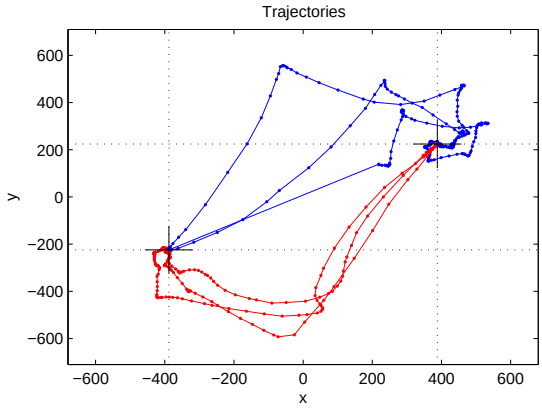
(a) Bias angle $\varphi = 0°$.



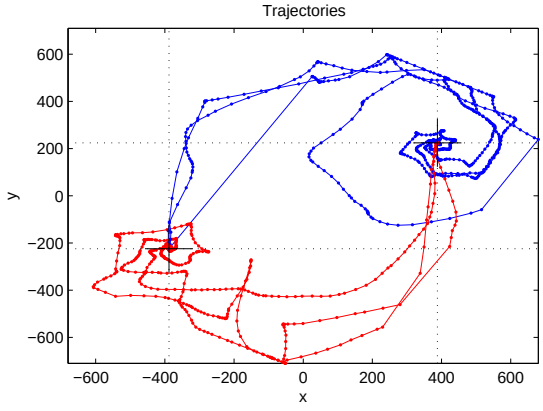
(b) Bias angle $\varphi = 10°$.



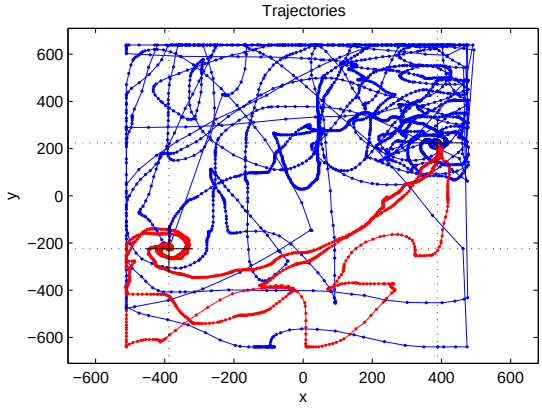
(c) Bias angle $\varphi = 20°$.



(d) Bias angle $\varphi = 30°$.



(e) Bias angle $\varphi = 60°$.



(f) Bias angle $\varphi = 90°$.

Figure 10: Trajectories during Task 1 for subject 2. The targets coordinates are (-388,-224) and (388,224) respectively.

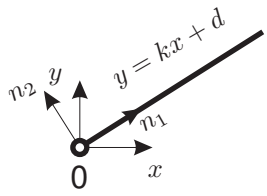


Figure 11: Projection to line coordinates

Path in line coordinates

The projection of the coordinates of point i from its original coordinates x, y to the line-coordinates n_1, n_2 is performed, see Fig. 11. Assuming that the origins of both coordinate systems coincide and using the direction vectors introduced in Section 5.1.2, the following relation holds for an arbitrary point i

$$x_i \begin{bmatrix} 1 \\ 0 \end{bmatrix} + y_i \begin{bmatrix} 0 \\ 1 \end{bmatrix} = a_1 \mathbf{n}_1 + a_2 \mathbf{n}_2 = \frac{a_1}{\sqrt{k^2+1}} \begin{bmatrix} 1 \\ k \end{bmatrix} + \frac{a_2}{\sqrt{k^2+1}} \begin{bmatrix} -k \\ 1 \end{bmatrix}, \tag{13}$$

where $\mathbf{n}_1$ is a vector parallel and $\mathbf{n}_2$ a vector perpendicular to the line $y = kx$. Solving these equations for the coordinates a_1, a_2 in the basis n_1, n_2 yields

$$a_1 = \frac{x_i + k y_i}{\sqrt{k^2+1}} \qquad \text{and} \qquad a_2 = \frac{y_i - k x_i}{\sqrt{k^2+1}}. \tag{14}$$

If the line does not go through the origin of the coordinate systems, $y = kx + d$, then the line-coordinates in Eq. 14 are simply transformed according to $y_i \mapsto y_i - d$.

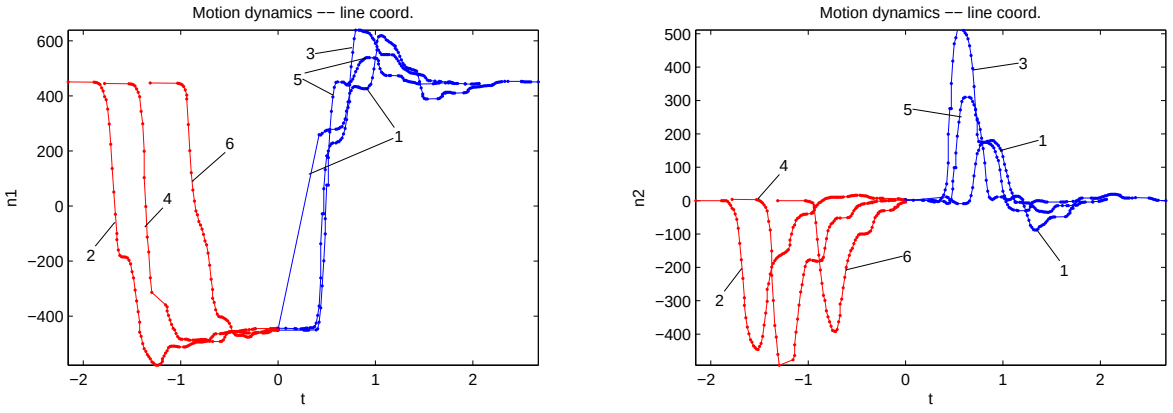
In the projected trajectory plots, all left to right movements are plotted on the left hand side.

Circular coordinates

Now

$$d(t) = \sqrt{(x_i(t) - x_c)^2 + (y_i(t) - y_c)^2} - R, \tag{15}$$

where x_i, y_i are the trajectory coordinates, x_c, y_c are coordinates of the centre and R is the radius.



(a) Projection to $n1$-axis.

(b) Projection to $n2$-axis.

Figure 12: Projected trajectories of the cursor motion of subject 2 for Task 1 at bias angle 30° in Fig. 10d. Left-to-right movements (blue lines 1 3 5) start a $t = 0$ (mouse click) and right-to-left movements (red lines 2 4 6) have been translated in time so to end at $t = 0$.

Example
The projections of the sample trajectory in Fig. 10d projected onto the $n1$- and $n2$-directions are shown in Fig. 12. These normal and tangential components can be used to fit e.g. first or second order model parameters (PT1 or PT2 Laplace transfer functions) including dead time (reaction time of the subject). Doing so enables a qualitative model to be developed, or two neural networks, one for vertical the other one for horizontal motion control, to be trained. The code for generating these projected data sequences is implemented. Based on them the identification of control parameters could be performed in follow-on work.

5.2 Brain activity

The performance analysis to this point was concerned with the motion of the mouse cursor. Apart from the motion dynamics, the electrical activity of the brain at its centre point `Cz` was measured, see Figs. 2 and 6.

As mentioned in the introduction, measuring EEG on the scalp, not only the brain's activity is recorded but also other sources of electrical activity. Any recorded activity that does not originate from the brain is called an artifact. These artifacts can be divided into physiological causes (generated in the body of the subject; eye and body movement, muscles, heart, ...) and extraphysiological causes (generated outside the body of the subject; equipment and environment). These artifacts are often much larger than the original brain activity and may overlay the frequency region of interest.

Artifacts can obscure completely the information of interest and need to be removed prior to any EEG analysis. It is a well known fact that the brain's activity oscillates in certain frequency bands. This fact forms the basis of EEG applications. Certain frequencies correlate better with certain activities. For example, lower frequencies in the 0-8 Hz range (δ- and ϑ-band in Table 1) are not associated with motor movements. By selecting which frequency band is useful can increase the accuracy of the subsequent data processing and analysis. For conscious motor activity the β-band is of interest which ranges from 13 to 30 Hz. For reference, the α-activity, which lies in the range between 8 and 13 Hz, is evaluated here as well, although it gives no new information. It may serve only as an indicator of how much the signal is contaminated by artifacts. Thus, the measured EEG signal was low-pass filtered with a cutoff frequency of 35 Hz and high-pass filtered with a cutoff frequency of 5 Hz. This preprocessing was implemented using a Butterworth filter of order two with subsequent application of the zero phase command filtfilt. Additionally, the signal is usually corrupted by interference from the mains (50 Hz). Thus, the decision was taken to filter the signal. First, a band-stop filter with cutoff frequencies of 48 and 52 Hz was applied to remove the dominant interference from the mains.

Muscle artifacts are much more difficult to remove since their frequency band overlaps the region of interest and, consequently, cannot be removed by filtering. Many methods have been proposed to remove artifacts from EEG recordings. If multiple EEG signals are being recorded then more sophisticated methods such as independent component analysis (ICA) and similar can be applied [24, 25]. A comprehensive introduction into ICA can be found in [22]. For this method to be applicable the number of channels must be at least equal to the number of noise sources. Some other methods proposed in the literature are: inline EEG analysis (during task) [40, 31], artefact removal for a brain computer interface (BCI) [32, 33, 20, 23], artefact removal for multiple channels [35], moving average autoregressive (MAR) model based on Kalman filtering [36], time-varying autoregressive (AR) models with Kalman filter implementation [40, 39, 37] or comprehensive introduction to PCA [41]. In this trial, only a single channel EEG measurement is performed in which case all the above mentioned methods are not applicable and the contaminated sequences need to be removed entirely from the epoch. Rejection of trials inevitably leads to a loss of data. However, in this study we are not interested in a time evolution but only the mean brain activity so that a rejection of parts of a training can be justified.

A good measure of brain activity is the mean power in a frequency band defined in Table 1. To calculate the power in a frequency band for each task and training within a task, a power spectral density (PSD) of an EEG sequence is estimated. Integration of the PSD in the frequency range of

interest gives the mean power of the signal in this band.

Summarising, an EEG analysis consists of three steps: data rejection, data filtering and data processing. An automated script for rejection of data is realised in the script `SlideFFT` which is explained in the following section. Then, the *cleaned* sequence is filtered to remove the 50 Hz component and components above the β-band and below the α-band. Finally, the mean power of a filtered signal is estimated within the α- and β-bands.

5.2.1 Sliding short-time Fourier transform (SFT)

First, low-frequency artifacts are removed from the acquired signals in an automated way. The EEG signal measured during one task which consists of TrainNum trainings is split according to the individual click times stored in ClickEventData. Subsequently, each sequence is filtered using a short-time Fourier transform in a sliding window. In general, the window length is a fraction of the total sequence length.

The discrete Fourier transform $\mathfrak{F}$ of a sequence $x(j)$ of length N is defined as

$$\mathfrak{F}\{x(j)\} = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} x(j) \exp\Big(\frac{-2\pi i}{N}(j-1)(k-1)\Big) = \frac{1}{\sqrt{N}}\mathfrak{F}^M\{x(j)\}. \tag{16}$$

This is implemented in the MATLAB command fft except for a scaling factor, which requires a post-multiplication in the code, see [42] for some details.For a window length of $2\Delta_w + 1$, the discrete short-time Fourier transform of the measured EEG signal is found to be

$$\mathfrak{F}_s\{EEG(k)\}(s) = \frac{\mathfrak{F}_s^M\Big\{EEG(1 + s\Delta_s \le k \le 2\Delta_w + s\Delta_s)\Big\}}{\sqrt{2\Delta_w + 1}}, \qquad s = 0, 1, \ldots \tag{17}$$

Muscle artifacts in the signal add dominant low-frequency components to the short-time spectra. To identify these artifacts, a characteristic value CharTrunc is introduced which is proportional to the mean value of the absolute values of Eq. 17 within a frequency band of up to 8 Hz. Another option might be to use the maximum value, or a weighted sum of maximum and mean value. If this characteristic value exceeds a predefined limit LimitFFT then the measured sequence is truncated. A detailed code listing is here:

```
function indEEG = SlideFFT(tEEG,EEG,SampleFreq,SlideWin,SlideDelta,LimitFFT)
% truncate data according to short time Fourier transform (STFT)
% by applying sliding window on data

% sliding window
NumFrame = floor((length(EEG)-(2*SlideWin+1))/SlideDelta);
for ifft = 1:NumFrame+1
    % extract frame of data
    EEGfft = abs(fft(EEG((ifft-1)*SlideDelta+(1:2*SlideWin+1))))/sqrt(2*SlideWin+1);
    fEEG = SampleFreq*(0:SlideWin)/(2*SlideWin+1);

    % extract data between up to 8 Hz; excluding mean value (0 Hz)
    indf = find(fEEG <= 8);
    indf = indf(2:end);

    % calculate mean value of abs(fft)
    CharTrunc(ifft) = sum(EEGfft(indf))/length(indf);
end
tfft = tEEG(SlideWin+(1:NumFrame)*SlideDelta);

% truncate data if <CharTrunc> exceeds certain limit
% only the last <SlideDelta> values are truncated and NOT <2*SlideWin+1> values
indEEG = SlideWin + (1:NumFrame*SlideDelta); % original data index
indTrunc = SlideWin + SlideDelta * find(CharTrunc > LimitFFT); % indentified anomalies
```

```
% remove data containers of size <SlideDelta>
for ind = 1:length(indTrunc)
    indEEG(indTrunc(ind) - (1:SlideDelta)) = 0;
end
indEEG(indEEG==0) = [];
```

This code is visualised in Fig. 13. If a data sequence of length Δ_w+1 possesses a dominant low frequency content then it is assumed to be contaminated by muscle artifacts and a data container of size Δ_s is removed from the measured signal, see Fig. 13 for details. This procedure is repeated until the last window within the full data sequence is reached. Any remaining part of the signal that is shorter than Δ_s is removed.

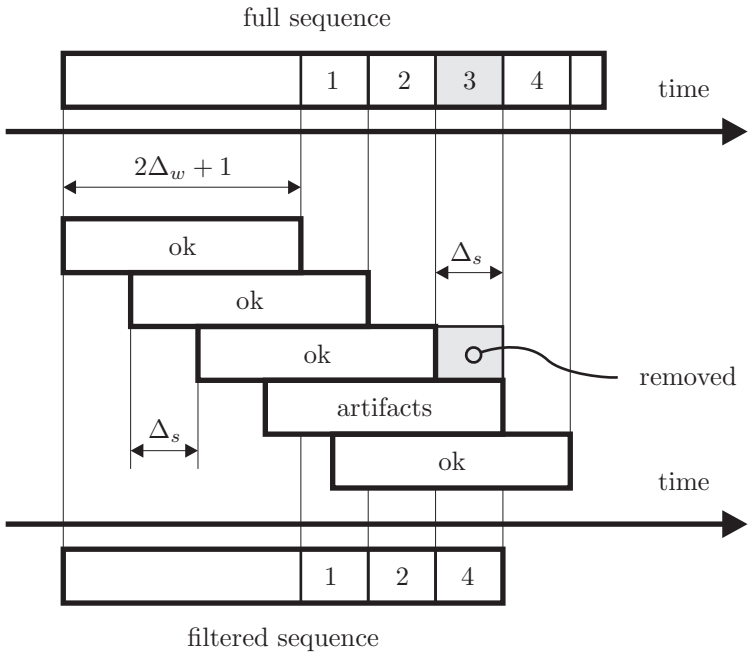


Figure 13: Filtering low-frequency artifacts by using sliding window and SFT.

This method depends strongly on the window size $2\Delta_w + 1$, the shift of the sliding window Δ_s and the threshold value `LimitFFT`. Several trials are needed to identify suitable values for an automated rejection. A sample of this automated sequence rejection is shown in Fig. 14.

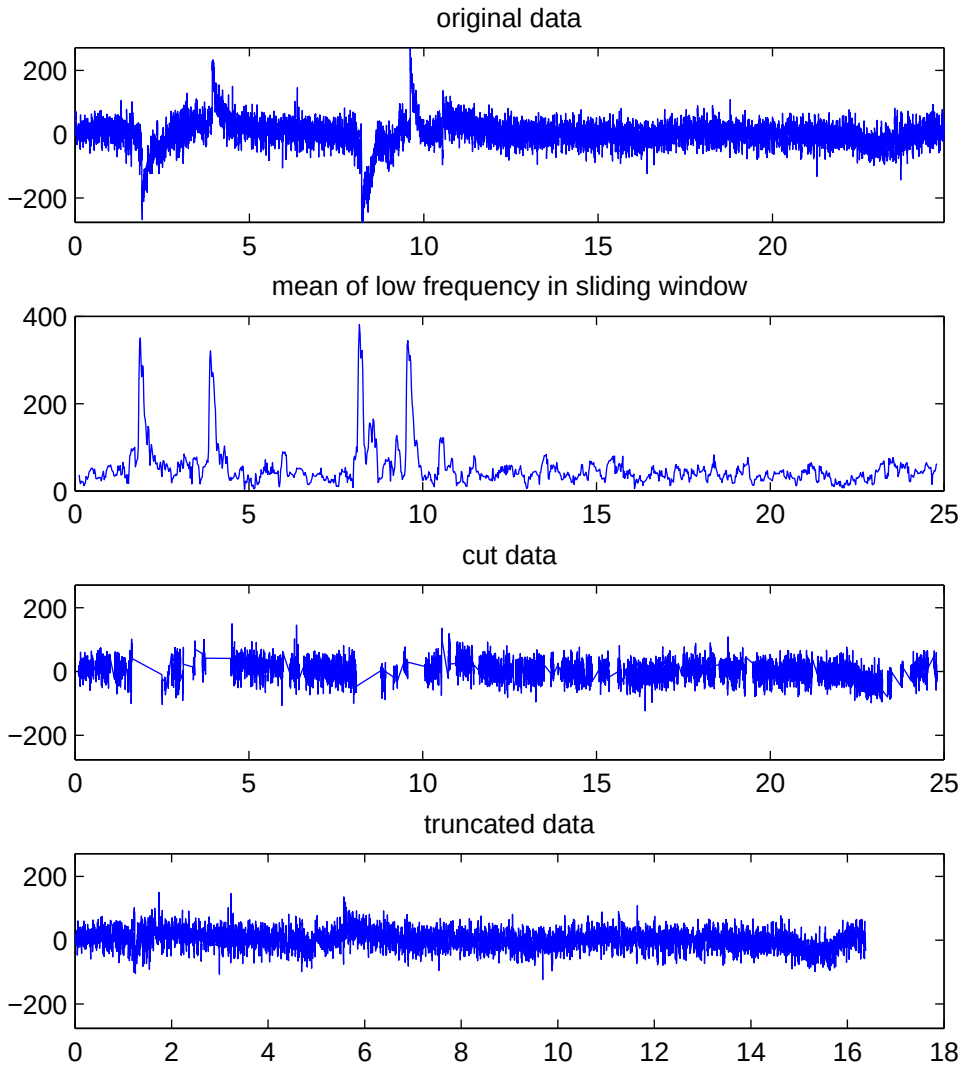


Figure 14: Processing a sample epoch of EEG: (a) Signal contaminated by noise and EMG artifacts, (b) Low frequency content of short-time FT spectra evaluated in a sliding window, (c) Removing muscle artifacts, (d) Concatenated signal for analysis.

5.2.2 PSD estimation

After filtering the measured data from muscle artifacts and other noise sources, the power content in different frequency bands is determined. To do this, the power spectral density needs to be estimated.

Several methods were investigated to estimate a PSD based on the acquired EEG data sets. For a direct calculation based on a discrete FFT the simplest estimation procedure is given here. The artifacts-filtered signal is windowed before estimation. Applying a Blackman window

$$w(n) = 0.42 - 0.5\cos\left(2\pi\frac{n}{N}\right) + 0.08\cos\left(4\pi\frac{n}{N}\right) \tag{18}$$

has been shown to be more efficient than the commonly applied Hamming or rectangular window. The two-sided spectrum is determined similar to Eq. 16 as

$$Y_t = \mathfrak{F}^M\Big\{w \cdot EEG_t\Big\}. \tag{19}$$

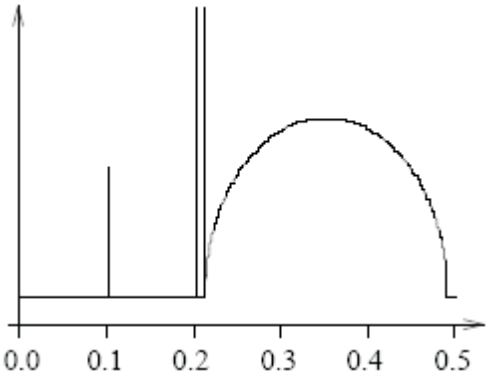
Hence, the power spectral density PSD can be estimated as

$$PSD_t = \frac{2Y_t \cdot \overline{Y_t}}{f_s(w \cdot w)}, \tag{20}$$

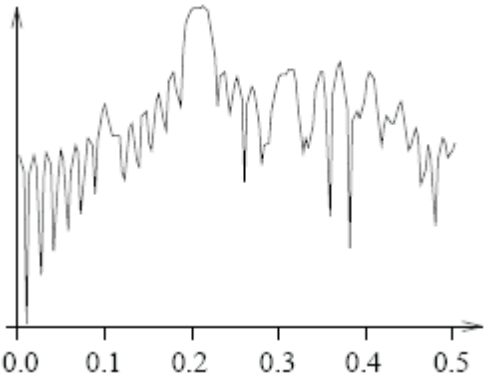
where the index t corresponds to the training number iTrain and f_s is the sampling frequency. A scaling factor as encountered in Eq. 17 cancels in Eq. 20, since the window w and the time-sequence EEG_t are of same length. The factor 2 comes from the transition of a two-sided to a one-sided PSD. An alternative would be to use the built-in command periodogram, which is in fact equivalent to pwelch.

In Eq. 20 the simplest form of PSD estimation was shown. Many other procedures exist to identify a PSD and the question is which method is the correct one to choose for a certain type of signal. For EEG signals a possible answer can be found in [28], see Fig. 15. Therein, an ideal PSD was given with a single sharp peak, two closely space peaks and a broadband component, see Fig. 15a. After transformation into the time domain, a truncated time sequence was calculated on which different methods were applied to estimate its underlying PSD. The results for applying the periodogram method, see the `Matlab` built-in command `periodogram`, with a rectangular and a Hamming window are shown in Figs. 15bc. Ideally these densities should match Fig. 15a but they are different due to sampling frequency and finite length of the time sequence. In both cases the two closely spaced peaks and the sharpness of the small peak cannot be resembled. Applying another method, an autocorrelation method, similar to the `Matlab` built-in command `arcov`, gives the results in Fig. 15de. For the Hamming window the estimated PSD resembles the main features of the ideal PSD well. Finally, applying an autoregressive (AR) model identification yields the result in Fig. 15f. Of all three methods, the AR model resembles the ideal spectrum the best, especially its sharp peaks. AR spectral estimation often gives a significant improvement in frequency resolution compared to the traditional FFT-based methods [28, 29, 30]. An important feature of the last method is its independence from windowing since the model is fitted to the time sequence directly and does not need to be repeated periodically as for the FFT-based methods `periodogram` and `arcov`. This procedure removes the need for windowing since no assumptions are made about data samples outside the data sequence. Detailed explanations of different ways to identify an AR model can be found in [28].

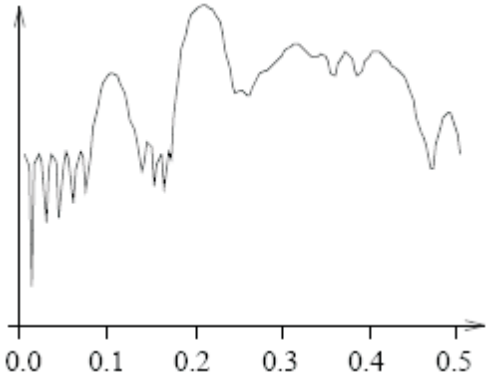
Motivated by [28], in this trial a second PSD estimate was calculated utilising an autoregressive (AR) model which is identified using a two-directional fit according to Burg's equations. To improve the quality of this fitting, the processed signal needs to be filtered. A notch-like filter is applied around the power supply frequency of 50 Hz, a high pass filter was applied with the cut-off frequency equal to 5 Hz and a low pass filter with a cut-off frequency at 35 Hz. The high-pass filter helps to satisfy the main assumption in AR modelling: the stationarity of the signal. All filters are designed using a Butterworth digital filter butter of certain order in combination with the zero-phase filter command filtfilt. The pass filters have an order of 2 and the notch-like filter is of order 5.



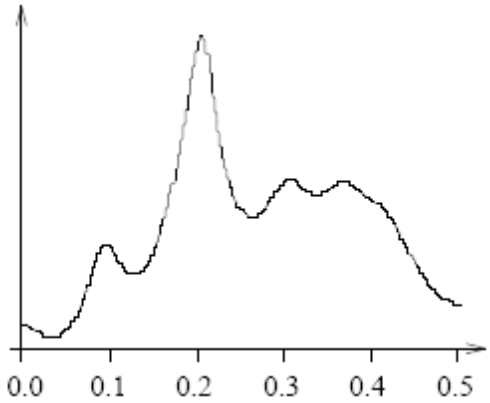
(a) Ideal spectrum.



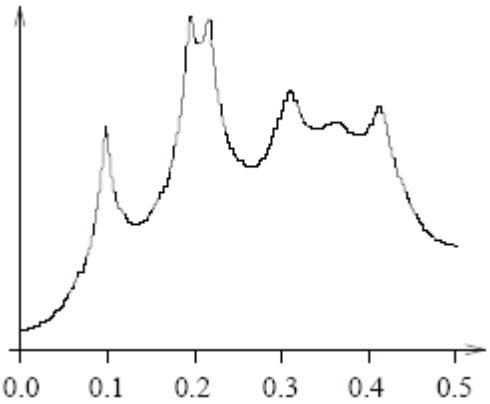
(b) Periodogram method.



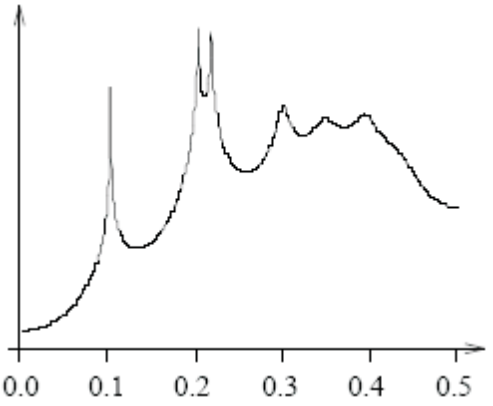
(c) Periodogram method with Hamming window.



(d) Autocorrelation method.



(e) Autocorrelation method with Hamming window.



(f) Maximum entropy method or AR using Burg's equations.

Figure 15: Motivation for AR modelling (copied from [28]).

The MATLAB environment offers built-in commands pburg, pcov, peig or pwelch that estimate an AR-model using Burg's equations for a given number of poles. Besides pburg and pwelch these methods are not applicable for the short time sequences considered. Unfortunately, the number of poles depends highly on the individual signal, which excludes an automated identification of many hours of measurements. Different criteria exist to predict a proper value for the number of poles. Here the approximate Kullback information criterion (AKIC) is implemented which attempts to balance the complexity of the model (number of poles) against how well the model fits the original data. The AKIC has the least bias and best resolution of the available model-selection criteria, see [27] for more details. Note the important fact that no windowing was applied to identify the AR-models. At this point, it should be noted that the main assumption of an AR model is the stationarity of the signal. Clearly, an EEG signal is not stationary during a goal-directed task. In such a case, the acquired signal is usually cut into short sequences (about 1 second) in which the stationarity of the signal is assumed and an AR model is fitted. In doing so (see directory `Evaluation_September2008`), no deviations in the resulting mean power values were noted and, therefore, cutting each signal into short sequences was omitted in the following analysis.

The built-in commands pburg or arburg are an implementation of the Burg's algorithm. The function arburg estimates an autoregressive model based on Burg's equations via the discrete transfer function

$$H(z) = \frac{\sqrt{e}}{1 + a_2 z^{-1} + \ldots + a_{p+1} z^{-p}}. \tag{21}$$

The stability of this *fitting* is important and is determined by the location of the poles of this transfer function defined by [26]

$$z^p + a_2 z^{p-1} + \ldots + a_{p+1} = 0. \tag{22}$$

The model is stable if all poles lie within the unit circle. The output transfer function of arburg can be plotted using freqz. However, this built-in implementation expects the number of poles as an input and does not allow information criteria to be used to estimate the number of poles. Therefore, the `Matlab` functions `ArburgD` and `ArPSDD` were downloaded [43], in which different information criteria were already implemented and tested. The model orders to give the best fitting of the time signals are found using a corrected information criterion (`AKICc`). The number of poles of the AR model to be fitted to a cleaned EEG data sequence is determined by this criterion. In any case, the number of poles is limited to 25% of the total length of the data sequence in order to avoid overfitting in case of a short data sequence.

Integrating the power spectral density over a frequency band gives the total power in a frequency band of interest as

$$P^{tot}_{\alpha \text{or} \beta} = \sum_i \frac{\Big(f_2 - f_1\Big) PSD_i\{f_1 \le f \le f_2\}}{\text{number of elements in band}} \tag{23}$$

The total power in the α- and β-bands during each training is stored in the matrices PowEEGAlpha and PowEEGBeta. The last column corresponds to the mean value over all `TrainNum` trainings at a fixed bias angle, so that their size is equal to (the number of bias angles defined by `phiTaskData`)$\times$(the number of trainings stored in `TrainNum` plus 1 for the mean values). This matrix was stored for all three tasks so that the variables PowEEGAlpha and PowEEGBeta become cell arrays of length 3, where each cell holds the matrix of power values in each frequency band.

Two variables are introduced to express the confidence in measured data. The variable EEGConf is introduced to store how much of the measured signal is contaminated by muscle artifacts. The variable CurConf is introduced to capture cases in which the hidden cursor hits the edge of the monitor. During the experiment a monitor with a resolution of 1280 $\times$ 1024 pixels was used. This property is stored in MATLAB as get(0,'ScreenSize'). Note that the operational range of the cursor is from 1 to 1280 on the horizontal axis and from 1 to 1024 on the vertical axis so that CurConf must be defined as:

```
CurConf(itask,ic) = sum((check(1,:)==XCurConf(1)) | ...
                        (check(2,:)==XCurConf(2)) | (check(1,:)≤1) | (check(2,:)≤1)) == 0;
```

A try-catch sequence embraces the EEG evaluation. If the automated evaluation of the EEG measurement fails, the corresponding entry in the array EEGConf is set to 0 and the entry in PowEEG to NaN.

5.2.3 Example

An example for an FFT-based estimation using a Blackman window and and an AR-based estimation of a PSD is shown in Fig. 16. The time sequence of the acquired x-coordinate of the cursor motion is shown in the top left graph. The acquired EEG signal after filtering with a sliding window as outlined in Section 5.2.1 is plotted in the bottom left graph. Both signals were acquired during a task at a fixed bias angle which consists of `TrainNum` trainings (6 in this trial). A PSD is estimated for each training using the FFT-based method `periodogram`, see centre top graph, and an AR model fitting `ArburgD` and `ArPSDD` without windowing in the top right graph. The number of poles representing the AR model is significantly reduced by band filtering the time sequences in the bottom left graph as explained in the previous section (5 to 35 Hz). To indicate the stability of the fitted AR model, the title in the top right graph holds the absolute value of the critical pole in the discrete and complex z-plane, see Eq. 22. The bottom centre and right graphs show the mean power of the corresponding PSD within the α- (left bars) and β-bands (right bars). The last column represents the mean values over all `TrainNum` trainings. The strong dependence on the PSD estimation method becomes evident. For subsequent analysis, only the AR model based PSD estimation are considered.

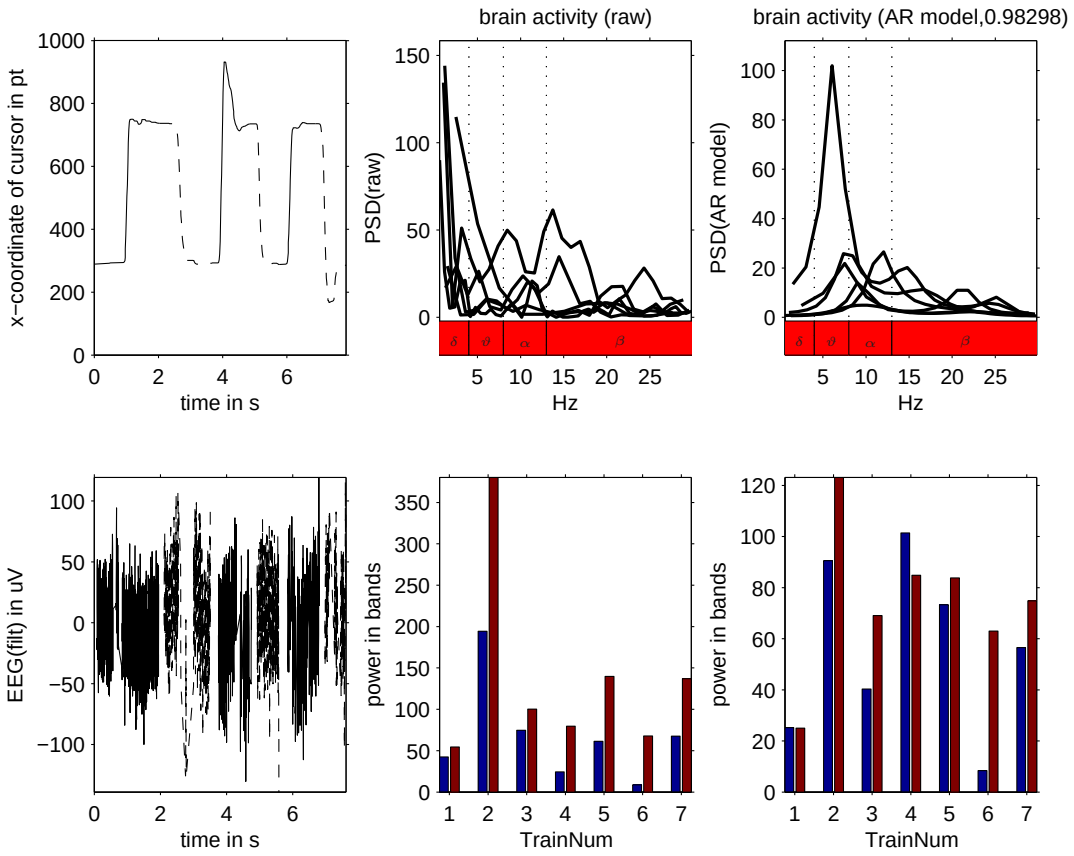


Figure 16: Example for estimating a PSD.

5.3 Analysis for single subject

This trial considers a small number of subjects and can be seen as a first gaining of experience of an unfamiliar topic. Inevitably, some of the EEG signals were strongly contaminated so that the data was rejected, the confidence thresholds `EEGConf` and `CurConf` introduced in Section 5.2.2 being equal to 1. However, for a larger set of subjects and an improved electrode arrangement, these thresholds, which are already implemented within the evaluation script, should be activated.

The sequence of bias angles φ is defined in the matrix `phiTaskData` and the sequence of tasks is implemented in the main code defined in the push button configuration in the GUI `MouseControlGUI.m` and is set to Task 1, 2 followed by 3. The trial starts with the bias settings in the first column of `phiTaskData` and the Tasks 1 to 3 are trained subsequently. Then the mouse characteristic is adapted according to the definitions in the second column of `phiTaskData` and the Tasks 1 to 3 are trained subsequently, and so on. Each task and bias angle is trained six times which is set by the variable `TrainNum`. In this trial the sequence of bias angles was defined as $0°$, $15°$, $-15°$, $30°$, $-30°$, $60°$, $-60°$, $90°$, $-90°$, $120°$, $-120°$, $150°$, $-150°$ and finally $180°$. For the first three subjects, the sequence of bias angles up to $30°$ was $0°$, $10°$, $-10°$, $20°$, $-20°$, $30°$, $-30°$ but was shorter for the following subjects to avoid lack of concentration towards the end of the trial.

In total four characteristic values were introduced for each task, bias angle and training: the total time, a characteristic distance, and the mean power in the α- and β-bands. These characteristic values are plotted for each subject separately in order to evaluate the individual performance of a single trial. The diagrams for subjects 2 (frequent user) and 10 (heavy user) are shown for Task 1 in Figs. 18 and 19 on page 44, for Task 2 in Figs. 20 and 21 on page 45 and for Task 3 in Figs. 22 and 23 on page 45.

For the initial task at $\varphi = 0$ (conventional mouse characteristic), or small bias angles of up to $30°$, in general, an exponential decrease of the characteristic distance is observed during the six tasks while the total time for each training is approximately the same, see e.g. Fig. 22 for an exaggerated case. This improved accuracy in motion is associated with a learning effect during which the subject gets *familiar* with the task. Less steep distributions can be found in e.g. Fig. 18 but are hidden due to high peaks in the vicinity of $\pm 90°$. These high peaks in the total time and the power in the β-bands indicate increased motion or brain activity compared to neighbouring values of the bias angle. The entire set of subjects consists of experienced computer users for whom operating the mouse at small bias angles seems to be straightforward. Similarly, most subjects find it easy to operate the mouse at its maximum bias angle of $\varphi = 180°$, where the mouse pointer moves exactly opposite to the hand motion. Interestingly, a narrow range exists in between the conventional mouse setting and the maximum bias angle in which, for most users, an increase in the performance indices is observed. This is analysed in some detail in the following section.

5.4 Analysis for all subjects

Each subjects trained each task at a certain bias angle for `TrainNum` times. The average value of each performance index over each set of 6 trainings is calculated and plotted for each task in the diagrams in Figs. 24, 25 and 26. The line depicts the mean value for all subjects for a particular bias angle. The maximum and minimum mean values for individuals is indicated by the range (error bars). For Task 1 in Fig. 24, observing the trend of the performance indices time and distance confirms the statement of increased motion activity in the vicinity of the bias angle $\pm 90°$ made in the previous section and generalises it to all subjects. A similar statement results for Tasks 2 and 3 in Figs. 25 and 26 from the performance index distance, however, different to Task 1, nothing can be concluded from the performance index time. For all three tasks, no statement can be concluded from the β-activity due to its large range across all individuals.

Each subject was questioned before the trial on physical activity, average alcohol, cigarette and coffee consumption, and their actual state of concentration, level of computer usage. Subjects with common features were grouped and their performance indices combined graphically. Tighter ranges

in the resulting graphs are a good indicator that a specific feature has a significant influence on the performance. Of all features from above, only the level of computer usage was identified to be important. Dividing the set of subjects into two groups, the results for the group of *frequent* users (daily office user) are plotted in Figs. 27, 28 and 29 and for the group of *heavy* users (e.g. gamers) in Figs. 30, 31 and 32.

For the group of frequent users, this separation decreases the range of the performance indices time and distance significantly, compare for example results for Task 1 in Figs. 24 with 27, and This even enables an interpretation of the corresponding β-activity to be made. Three increased activities can be identified in Fig. 27: one at $\varphi = 0°$ during which most of the subjects get familiar with the tasks (see previous section), one at $\varphi = -120°$ and one at $\varphi = 90°$. Due to the large range around the last two peaks not all subjects experience an increases brain activity. Nevertheless, tighter ranges outside of these three average peaks indicate that all frequent users had low brain activity at bias angles apart from these three.

The performance of the group of heavy users during Task 1 is shown in Fig. 30. Again, the ranges become tighter compared to Fig. 24 which justifies the separation into the proposed two groups of users. A significant increase at $\varphi = -90°$ and $\varphi = +90°/+120°$ becomes clearly evident for the performance indices time and distance, however, the β-activity is still hard to interpret. The 'get-familiar' peak at $\varphi = 0°$ with vanishing range is remarkable. The peak at $\varphi = 60°$ results from a high β-activity of a single subject and was probably generated during a phase of impatience and nervousness. A small increase in the brain activity can be found at $\varphi = +120°$. A similar peak cannot be found for the trial following negative bias angle although the total time and characteristic distance are of similar size. This indicates a learning effect for which the group of heavy users have adapted their performance to the bias angle since their brain activity is smaller for a comparable accuracy in their motion.

The same separation of user groups is performed for Task 2 in Figs. 28 and 31. For the distance performance index a dominant peak is found at the positive bias angle $\varphi = +120°$ and a significantly lower peak at the negative bias angle $\varphi = -120°$. This indicates a significantly improved accuracy in motion for a similar total time for both tasks. Keeping in mind that the trial consists of a sequence of alternating positive and negative bias angles suggest a learning effect for both user groups. A higher number of subjects is needed to interpret the brain activity. Finally, similar graphs are generated for Task 3 in Figs. 29 and 32. Now, the aforementioned learning effect is observed for the group of heavy users only while the characteristic distance for the group of frequent users remains the same.

6 Summary

This first trial showed clearly that a goal-directed movement, described by its motion dynamics and the corresponding brain activity, depends on the mouse bias angle and difficulty of the task. The analysis of the trajectories during Task 1 provides evidence to support existing literature that two separate control mechanisms coordinate the movement: a direction and a distance control. Projecting the trajectories during Tasks 2 and 3 could be used to estimate first and second order controller models including time delays. Regarding the brain activity measured via EEG, a higher number of subjects would be needed to give more confident results but clear effects are still observed in several cases. The brain activity was measured as the power in certain frequency bands and identified by a fitted AR model. AR spectral estimation then gives significant improvement in frequency resolution compared to FFT-based algorithms and can resolve sharp closely spaced peaks as well as broadband content. Using AR spectral estimation in connection with Burg's equations removes the need for windowing. Performance indices were introduced to benchmark the performance of the subjects which showed an increase in motion and/or brain activity at certain bias angles. Doing the same or a similar task over and over again decreases stimulus or difficulty and were identified as learning effects. Two groups of people could be identified within the set of subjects: the group of *frequent* users and the group of *heavy* users. Both achieved high characteristic distances but the brain activities are different for each group.

Table 4: Expected bias angle φ for highest motion/brain activity for each task and both user groups.

	Task 1	Task 2	Task 3
frequent user	$+90°$ $-120°/-90°$	$+120°$ $-120°$	$+90°/120°$ $-120°$
heavy user (gamer)	$+90°$ $-120°$	$+120°$ $-120°$	$+90°$ $-120°$

The sets of bias angles for which an increased motion and/or brain activity was observed are summarised in Table 4. Interestingly, for some directions it is especially hard to perform the tasks. Hence, bias angles exist for which the brain activity in the β-band are persistently higher ($\approx \pm 100°$), and are able to sustain a persistent stimulus even after several trainings. Regarding the learning effect, a decrease in one or more performance indices, the group of heavy users experienced a learning effect at the bias angles listed for all three tasks, a small learning effect for Task 1 and a significant one for Tasks 2 and 3. The group of frequent users encountered a small learning effect only for Task 2. Consequently, for this group of users a repeated stimulation of brain/motion activity can be achieved by Tasks 1 and 3 at the bias angles listed in Table 4. It is remarkable that the learning effect was not observed for the frequent users during Task 3, since this task is trained directly after Tasks 1 and 2 for the same bias angle. Hence, although the same mouse characteristic was trained by the frequent users during Tasks 1 and 2 for a total number of 12 trainings, no sign of a decrease in performance or difficulty can be observed. This suggests that the proposed continuous and simultaneous direction and distance control during Task 3 is the most efficient task to achieve a repeated and persistent stimulus for the brain.

A few comments on the setup arrangement follow. In the main literature on goal-directed arm movements (see e.g. the journals of *Human Movement Science* or *Experimental Brain Research*) it is very common to consider only a small number of subjects. This is mainly due to the difficulty to find or have access to a sufficiently large group of subjects with a specific feature (syndrome, disability, etc.). In the majority of studies the number of tested subjects is something below 10 and is reduced due to unpredictable reasons or other arguments to a number of 3 to 5 test persons. This is absolutely legitimate if omitting some of the acquired data can be sufficiently justified. Nevertheless, a statistical analysis based on such a small number is mathematically unjustified, although in many cases standard deviations are listed. In this trial, a small number (12) of subjects is analysed, too, of which 5 are not considered due to aborted trials or software conflicts. The data of the remaining 7 subjects is evaluated and split into two groups consisting of 4 and 3 subjects. Valuable information can still be obtained (similar to established literature), however, here a statistical analysis is omitted and only the more conservative minimum and maximum values are determined.

In this trial, a bias angle is introduced which could be compensated for simply by rotating the mouse device. Since the hand is screened from the subject as well as the supervisor this could be achieved unnoticed. Small bias angles of $\pm 15°$, maybe $\pm 30°$ could be compensated by twisting the wrist accordingly. For larger bias angles, one might think of compensating a bias rotation by keeping the hand in its natural position and rotating the device only. This is prevented for Tasks 1 and 2 by asking the subject to move the cursor quickly and to click on each target which assured a proper hand placement. For Task 3 this is theoretically possible but unlikely due to the preceding trainings and familiarisation at a certain bias angle.

7 Recommendations and comments

The proposed measurement setup shows that bias angles for increased motion and/or brain activity lie somewhere between $\pm 90°$ and $\pm 120°$. Introducing a higher angular resolution in these regions instead of the 30°-steps in this study would give a more accurate prediction of increased activity. Also, the sequence of rotations should be randomised in order to minimise learning effects and verify the expected bias angles with more confidence.

Another improvement of this trial should be achieved with respect to the EEG acquisition. Tighter error bands of the analysed power in certain frequency bands could be realised by several different methods which are listed here:

- Towards the end of this study I found this work on artefact removal in a single channel measurement [34, 38], which seems to be an interesting and robust alternative to the data processing in Section 5.2.2.
- The data processing of EEG signals could be improved by implementing non-stationary models, for instance, based on Kalman filters. See Section 5.2 for some literature. A concise overview on qualitative EEG analysis (qEEG) can be found in [19].
- Filtering muscle artifacts more accurately could be achieved if a second EEG electrode would be used to measure the muscular activity on the neck synchronously to the brain activity with a subsequent PCA decomposition.
- A single channel EEG was measured in this trial. PCA-methods (e.g. `Matlab` command `princomp`) or even ICA-methods would become applicable [22] if multi-channel EEG signals are measured. Also, free software under the GNU licence agreement exist with libraries of several tested procedures, including ICA and PCA. However, due to their complexity, e.g. [21], more time is needed to operate this software in its full functionality.
- Measuring a single EEG during a phase of dedicated thinking and being bored would generate reference signals that could be used to define an accurate state of increased brain activity. In doing so, a precise range for the frequency band β for each subject could be derived individually instead of the average values listed in Table 1.
- All subjects in this trial were daily office users between 18 and 32 years. A higher confidence in the results would be gained by a larger*number* and *diversity* of subjects.
- Use deformed circular path (rotated ellipse): possibility to train specific arm muscles?

In this trial the EEG was acquired at the point `Cz`. After the seminar presentation it became clear that measuring at `C3` for right-handed people and at `C4` for left-handed people could increase the EEG signal amplitudes of a factor of 2.

Recently mass-produced equipment to acquire brain activity became available, see for instance the headsets [44, 45]. These might be a sufficiently good alternative to the Biopac Systems since for this trial the time resolution of the brain activity is not important and only an average value over a single training is considered. Another alternative could be to use HEG (Hemoencephalography) instead of EEG measurements, see for instance the headsets [46].

An algorithm was developed that translates the acquired trajectory data into line coordinates, see Fig. 12. Based on these data sets a simplified control model could be developed which is described by a PT1 or PT2 transfer function or a trained neural network.

Finally, instead of an easily movable object such as the optical mouse, a hand device with controlled resistance could be used. Still, one of the major aims of this project was to investigated the possibility of goal-directed trainings at *home* where a device as developed in [16] cannot be used. An alternative

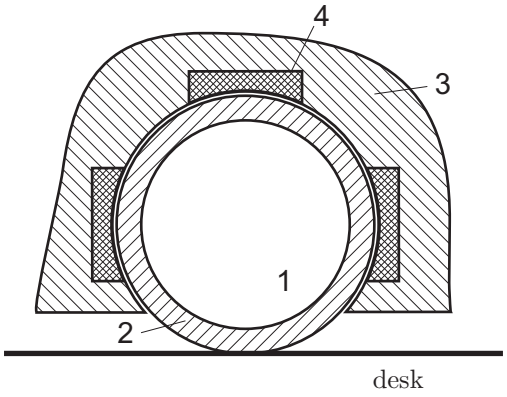


Figure 17: Hand device with controlled resistance: (1) ball or wheel, (2) sufficiently thick iron skin, (3) housing and (4) electromagnets.

hand device would be of course less efficient than the professional device in [16] but accessible virtually anytime. A possibility of such a device might be a magnetic device sketched in Fig. 17. A resistance in motion is achieved here by eddy currents induced in a moving part. The key point for this device would be to keep the air gap between the iron skin of a ball or wheel and the guiding as small as possible in order to keep the necessary currents in the electromagnets small so that it can be used easily in connection with a PC and a conventional transformer. The device should be sufficiently heavy to avoid slipping. There might be saftey consideration to avoid injury to a subject. Since the hand needs to be screened anyway, the whole setup could be placed in a box that fits on a standard table. The arrangement sketched in Fig. 17 could be realised as a single larger component in the centre of the hand device with weight supporting wheels, or 3 to 4 smaller wheels mounted at the edges of the hand device. A starting point to define the design aims of such a device might be [47].

Once enough EEG data has been gathered to confirm values for the bias angles which lead to an increased activity, it might be possible to to correlate the performance indices time and distance with the β-activity. If so, then an EEG measurement becomes obsolete and trajectory data are sufficient to generate a stimulus and to interpret the results. At this point, performing the tasks in the browser window (`Microsoft`, `Mozilla`, `Opera`, etc.) and acquiring the trajectories on a server, would gain access to a virtually countably infinite number of subjects. A bias in the cursor motion in the browser independent script software `JavaScript` was implemented and can be found in the same folder. Scripts for data acquisition would need to be developed.

References

[1] G Flodgren, M Heiden, E Lyskov and AG Crenshaw, Characterization of a laboratory model of computer mouse use – Applications for studying risk factors for musculoskeletal disorders, *Applied Ergonomics* **38** (2007) 213–218.

[2] AAE Ahmed and I Traore, A new biometric technology based on mouse dynamics, *IEEE Transactions on dependable and secure computing* **4**(3) (2007) 165–179.

[3] W Abend, E Bizzi and P Morasso, Human arm trajectory formation, *Brain* **105** (1982) 331-348.

[4] L Hay, and C Redon, The control of goal-directed movements in children: Role of proprioceptive muscle afferents, *Human Movement Science* **16** (1997) 433-451.

[5] JJ van den Dobbelsteen, E Brenner and JB Smeets, Endpoints of arm movements to visual targets, *Exp Brain Res* **138** (2001) 279-287.

[6] B Rohrer, S Fasoli, HI Krebs, R Hughes, B Volpe, WR Frontera, J Stein and N Hogan, Movement smoothness changes during stroke recovery, *Journal of Neroscience* **22** (2002) 8297-8304.

[7] JJ van den Dobbelsteen, E Brenner and JBJ Smeets, Adaptation of movement endpoints to perturbations of visual feedback, *Exp Brain Res* **148** (2003) 471–481.

[8] A-C Eliasson, B Rösblad, and H Forssberg, Disturbances in programming goal-directed arm movements in children with ADHD, *Developmental Medicine & Child Neurology* **46** (2004) 19-27.

[9] JA Saunders and DC Knill, Visual feedback control of hand movements, *Journal of Neuroscience* **24** (2004) 3223-3234.

[10] M Roerdink, CE Peper and PJ Beek, Effects of correct and transformed visual feedback on rhythmic visuo-motor tracking: Tracking performance and visual search behavior, *Human Movement Science* **24** (2005) 379-402.

[11] L Hay, C Bard, C Ferrel, I Olivier and M Fleury, Role of proprioceptive information in movement programming and control in 5 to 11-year old, *Hum,an Movement Science* **24** (2005) 139-154.

[12] RA Scheidt, MA Conditt, EL Secco and FA Mussa-Ivaldi, Interaction of visual and proprioceptive feedback during adaptation of human reaching movements, *Journal of Neurophysiology* **93** (2005) 2300-3213.

[13] JA Saunders and DC Knill, Humans use continuous visual feedback from the hand to control both the direction and distance of pointing movements, *Exp Brain Res* **162** (2005) 458-473.

[14] D Goble, Validity of using reaction time as a basis for determining motor laterality, *Journal of Neurophysiology* **97** (2007) 1868-1868.

[15] DA Lantero and SD Ringenbach, Developmental differences in the use of visual information during a contiuous bimanual cooridnation task, *Journal of Motor Behavior* **39** 2007 139-155.

[16] Article *University to trial new technology to help stroke patients* published in the *Bulletin – the weekly newspaper of the University of Southampton* **13**(47), Monday 24 September 2007.

[17] T Akerstedt, Subjective and objective sleepiness in the active individual, Int. Journal of Neuroscience **52** (1990) 29-37.

[18] H Jasper, Report of the committee on methods of clinical examination in electroencephalography, Electrophysiology and Clinical Neurophysiology **10** 370-375.

[19] NV Thakor, and S Tong, Advances in quantitative electroencephalogram analysis methods, *Annu. Rev. Biomed. Eng.* **6** (2004) 453-495.

[20] A Erfani, and A Erfanian, The effects of mental practise and concentration skills on EEG brain dynamics during motor imagery using independent component analysis, In: Proceddings of the 26th Annual International Conference of the IEEE EMBS, San Francisco, CA, USA, Sept 1-5, 2004.

[21] A Delorme, and S Makeig, EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis, *Journal of Neuroscience Methods* **134** (2004) 9-21.

[22] T-P Jung, S Makeig, C Humphries, T-W Lee, MJ Mckeown, V Iragui, and TJ Sejnowski, Removing electroencephalographic artifacts by blind source separation, *Psychophysiology* **37** (2000) 163-175.

[23] J Anemüller, TJ Sejnowski, and S Makeig, Complex independent component analysis of frequency-domain electroencephalographic data, *Neural Networks* **16** (2003) 1311-1323.

[24] J Annemüller, TJ Sejnowski, S Makeig, Complex independent component analysis of frequency-domain electroencephalographic data, *Neural Network* **16** (2003) 1311-1323.

[25] S Makeig, A Delorme, M Westerfield, T-P Jung, J Townsend, E Courchesne and TJ Sejnowski, Electroencephalographic brain dynamics following visual targets requiring manal responses, *Public Library of Science Biology*, 2004.

[26] AV Oppenheim, and RW Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, 1989.

[27] Abd-Krim Seghouane and Maiza Bekara, A small sample model selection criterion based on Kullback's symmetric divergence, *IEEE Transactions on Signal Processing* **52** (2004) 3314-3323.

[28] J Pardey, S Roberts, L Tarassenko and J Stradling, A review of parametric moelling techniques for EEG analysis, Medical Engineering and Physics **18** 2 - 11.

[29] JM Spyers-Ashby, PG Bain and SJ Roberts, A comparison of fast fourier transform (FFT) and autoregressive (AR) spectral estimation techniques for the analysis of tremor data, *Journal of Neuroscience Methods* **83** (1998) 35-43.

[30] JN Knight, Signal fraction analysis and artifact removal in EEG, Master Thesis, Colorado State University, 2003.

[31] AD Krystal, R Prado and M West, New methods of time series analysis of non-stationary EEG data: eigenstructure decompositions of time varying autoregressions, *Clinical Neurophysiology* **110** (1999) 2197-2206.

[32] C James and S Wang, Blind source separation in single-channel EEG analysis: an application to BCI, In *Proceedings 28th Annual International Conference IEEE Enginering in Medicine and Biology Society* (EMBS), Piscataway, USA, Engineering in Medicine and Biology Society (2006) 4pp.

[33] H Ramoser, J Müller-Gerking and G Pfurtscheller, Optimal spatial filtering of single trial EEG during imagined hand movement, *IEEE Transactions on Rehabilitation Engineering* **8** (2000) 441-446.

[34] AM Tomé, AR Teixeira, EW Lang and A Martins da Silva, Greedy Kernel PCA applied to single-channel EEG recordings, In *Proceedings of the 29th Annual International Conference of the IEEE EMBS Cit Internationale*, Lyon, France August 23-26, (2007) 4pp.

[35] W De Clercq, A Vergult, B Vanrumste, W Van Paesschen and S Van Huffel, Canonical correlation analysis applied to remove muscle artifacts from the electroencephalogram, *IEEE Transactions on Biomedical Engineering* (2006) 2583-2587.

[36] WD Penny and SJ Roberts, Dynamic models for nonstationary signal segmentation, *Computers and Biomedical Research* **32** (1999) 483-502.

[37] VP Oikonomou, AT Tzallas, DI Fotiadis, A Kalman filter based methodology for EEG spike enhancement, *Computer Methods and Programs in Biomedicine* **85** (2007) 101-108.

[38] ME Davies and CJ James, Source separation using single channel ICA, *Signal Processing* **87** (2007) 1819-1832.

[39] M Rohál'ová, P Sykacek, M Koska and G Dorffner, Detection of the EEG Artifacts by the Means of the (Extended) Kalman Filter, *Measurement Science Review* **1** (2001) 59-62.

[40] DW Skagen, Estimation of running frequency spectra using a Kalman filter algorithm, *Journal of Biomedical Engineering* **10** (1988) 275-279.

[41] J Shlens, A tutorial on principal component analysis, Student notes downloaded at http://www.snl.salk.edu/ shlens/pub/notes/pca.pdf

[42] http://www.dsprelated.com/showmessage/16628/1.php

[43] http://www.koders.com

[44] Headsets from Emotiv Systems, www.emotiv.com

[45] Headsets from NeuroSky, www.neurosky.com

[46] www.pocket-neurobics.com

[47] S Mina, L Ehrenstrasse, R Lurf and B Prazak, Evaluation of reaction forces during human computer interaction for optimization and development – a pilot research, in: ICCHP 2006, K Miesenberger et al (Eds.), SpringerVerlag Berlin Heidelberg, 2006, 415-420.

A Analysis for a single subject

A.1 Task 1 – moving towards target

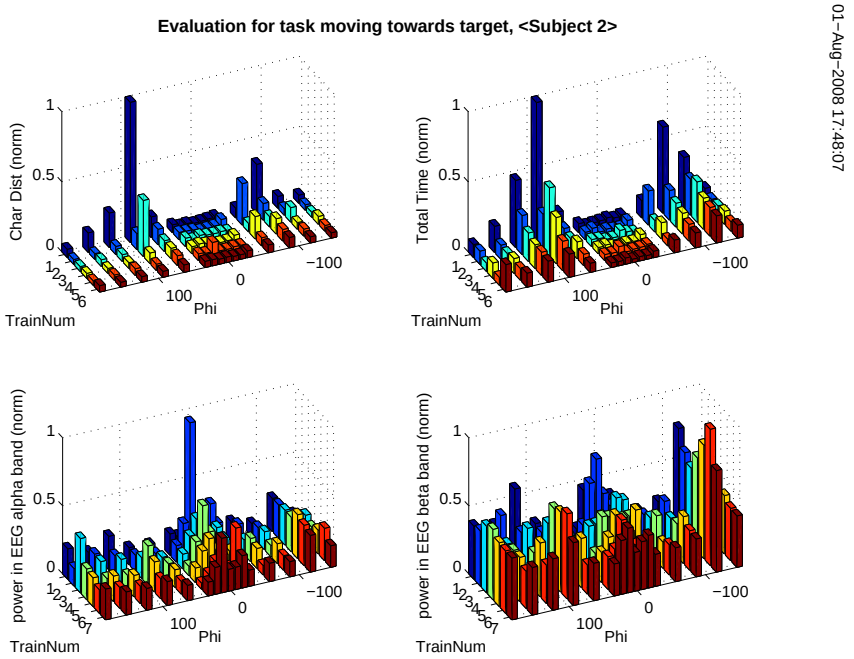


Figure 18: Characteristic distribution during Task 1 for subject 2.

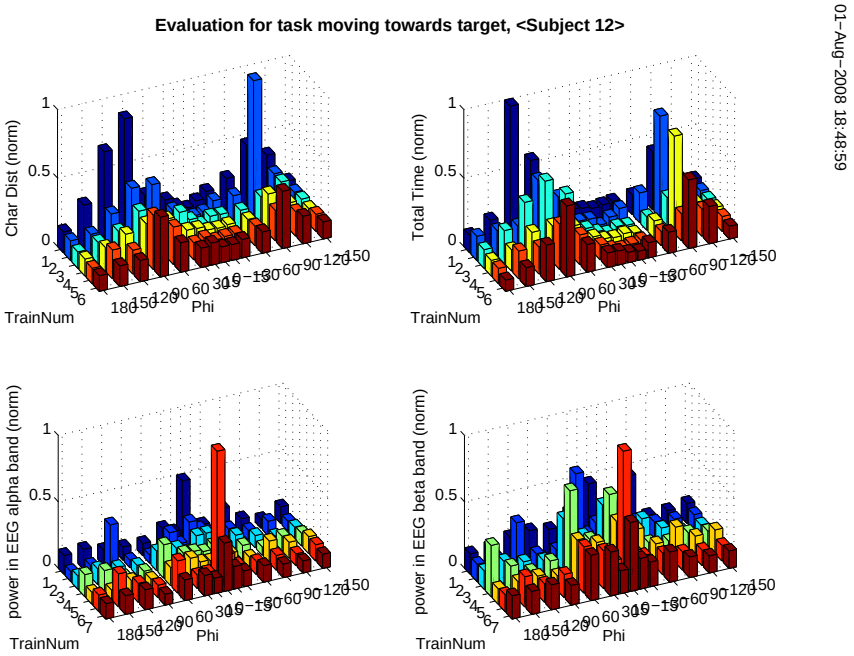


Figure 19: Characteristic distribution during Task 1 for subject 12.

A.2 Task 2 – tracking a linear path

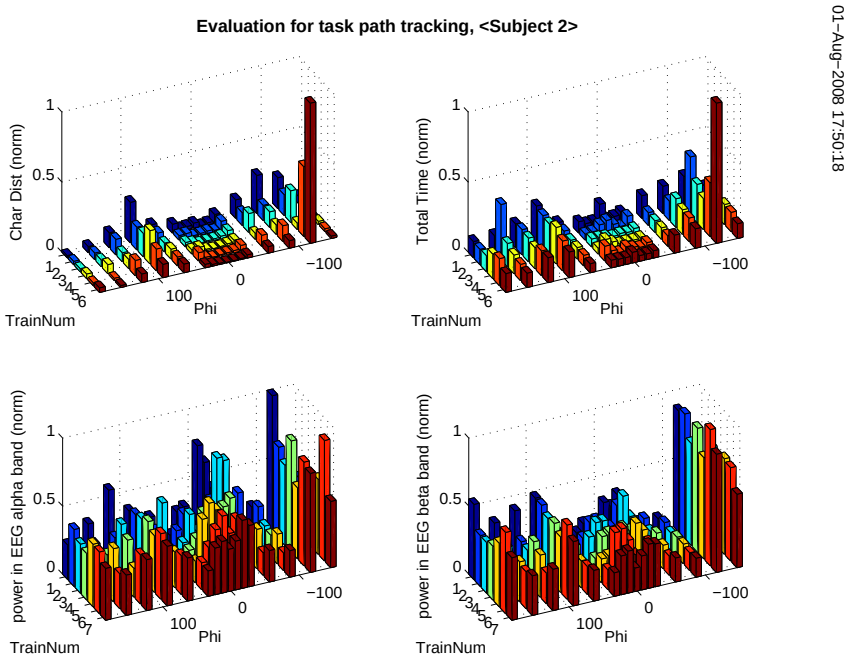


Figure 20: Characteristic distribution during task 1 for subject 2.

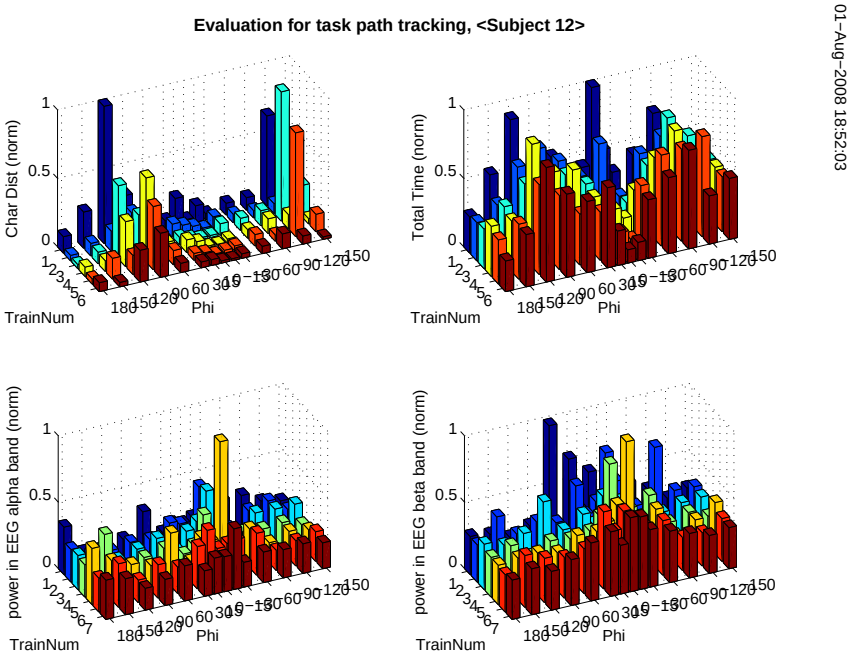


Figure 21: Characteristic distribution during Task 2 for subject 12.

A.3 Task 3 – tracking circular path

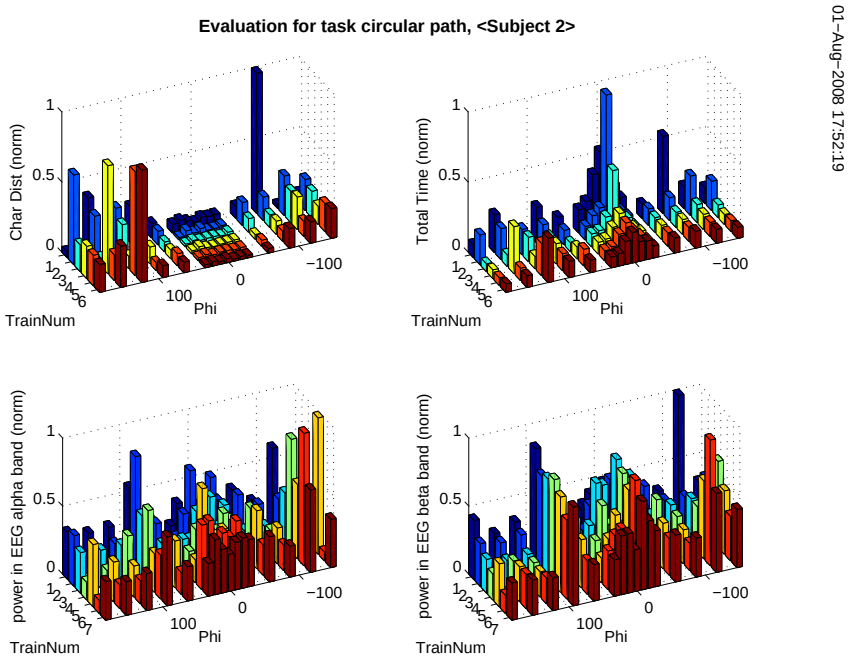


Figure 22: Characteristic distribution during Task 3 for subject 2.

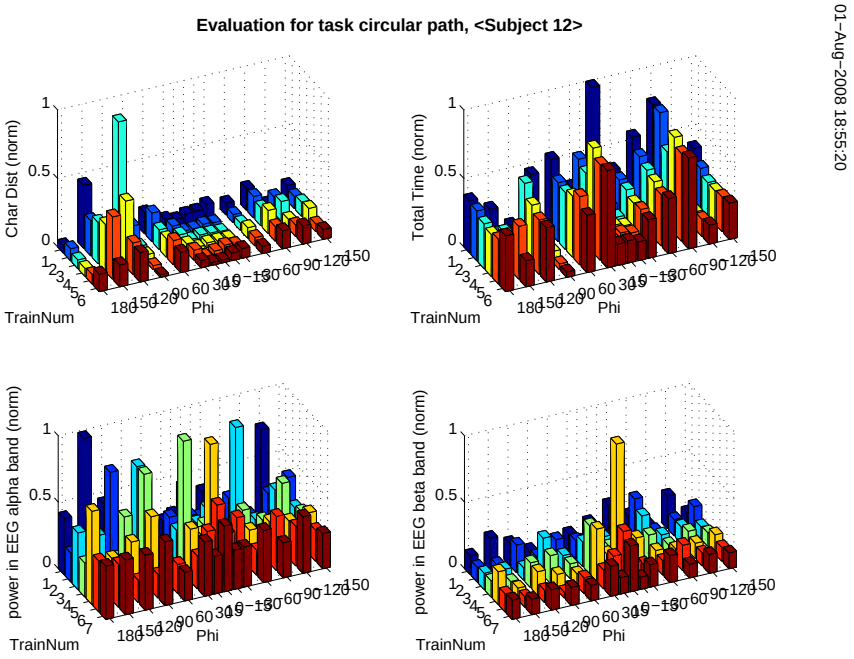
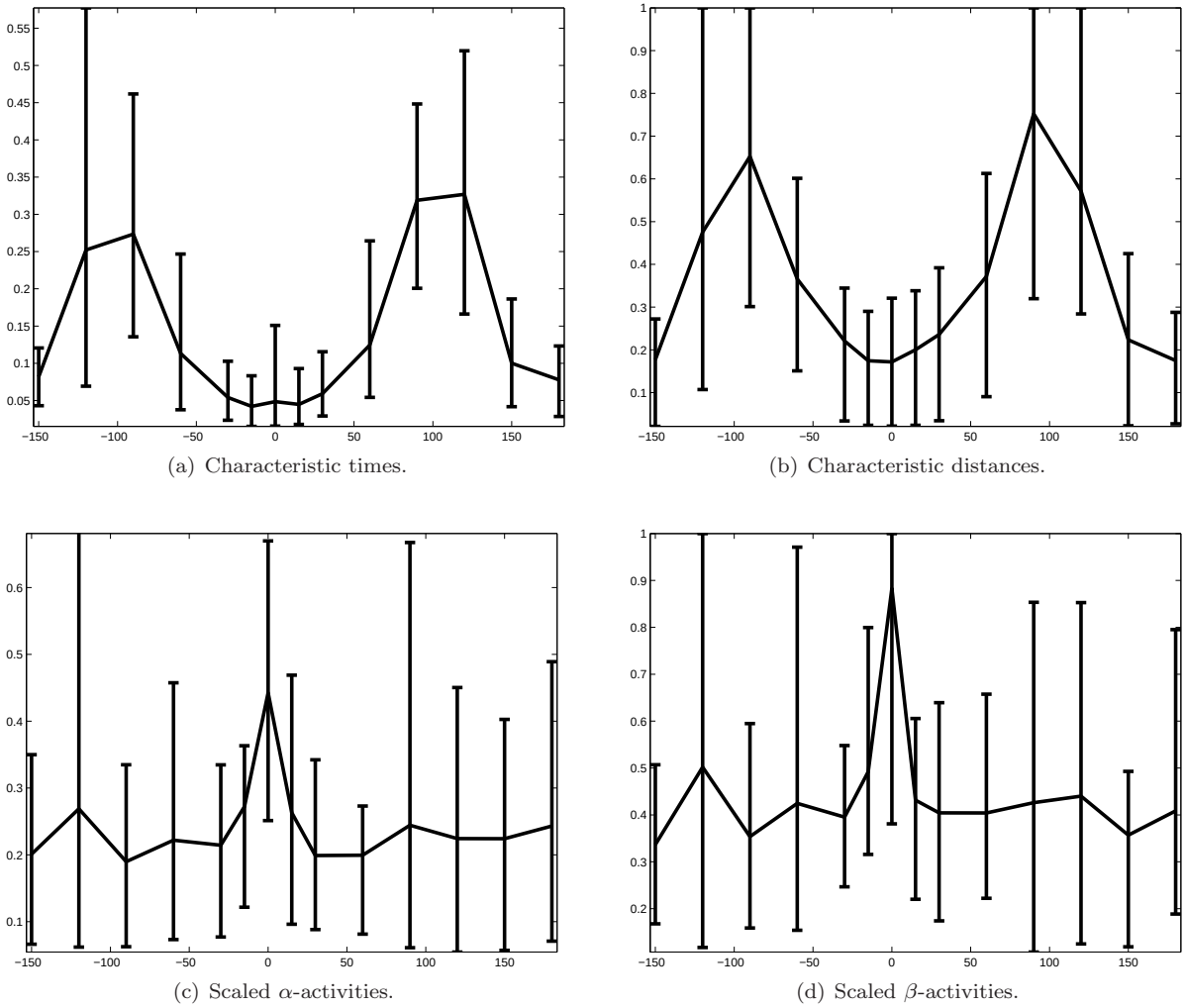


Figure 23: Characteristic distribution during Task 3 for subject 12.

B Evaluation for all subjects



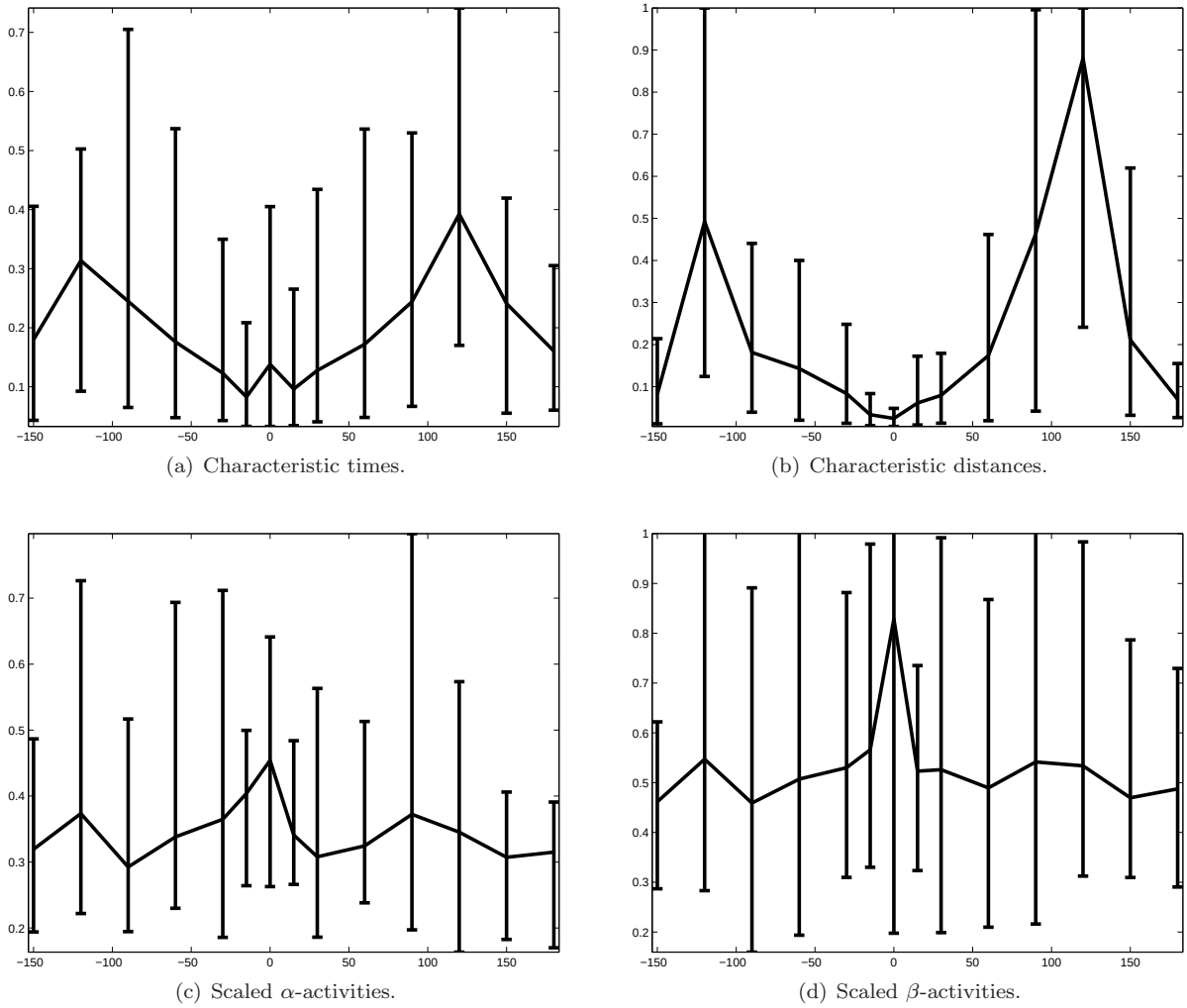
(a) Characteristic times.

(b) Characteristic distances.

(c) Scaled α-activities.

(d) Scaled β-activities.

Figure 24: Task 1 – Performance indices for all subjects in dependence of the bias angle φ.



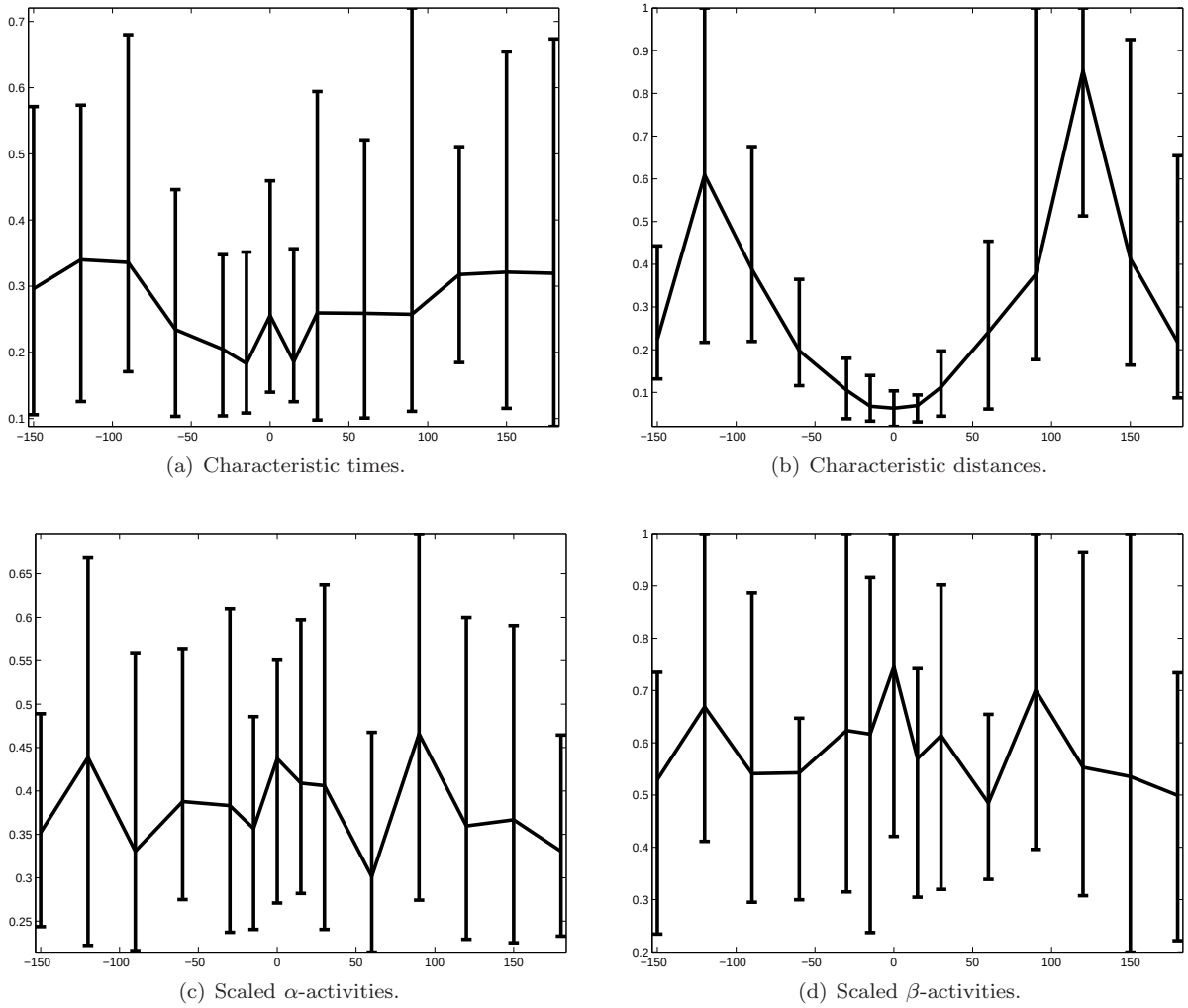
(a) Characteristic times.

(b) Characteristic distances.

(c) Scaled α-activities.

(d) Scaled β-activities.

Figure 25: Task 2 – Performance indices for all subjects in dependence of the bias angle φ.



(a) Characteristic times.

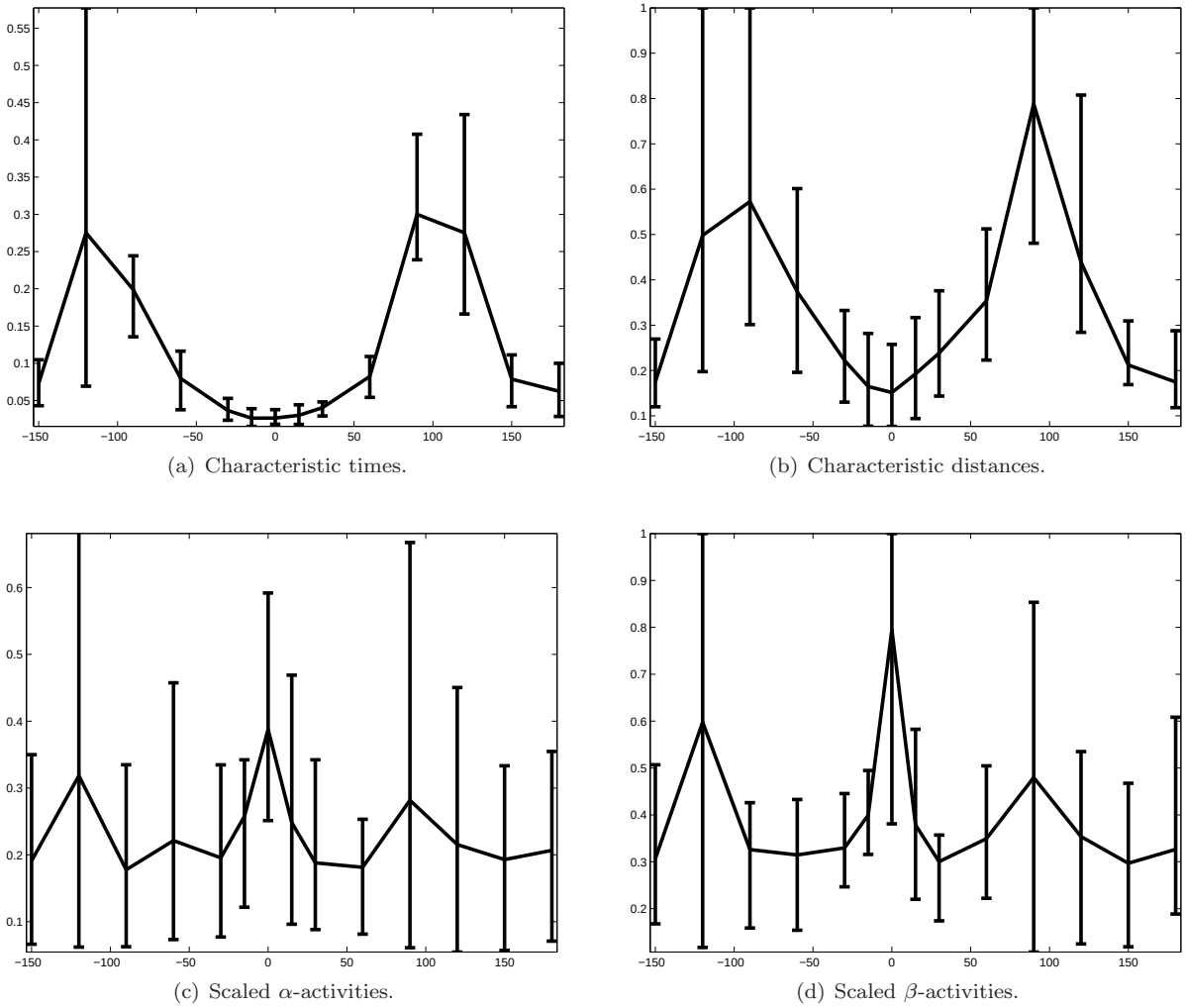
(b) Characteristic distances.

(c) Scaled α-activities.

(d) Scaled β-activities.

Figure 26: Task 3 – Performance indices for all subjects in dependence of the bias angle φ.

C Evaluation for frequent users



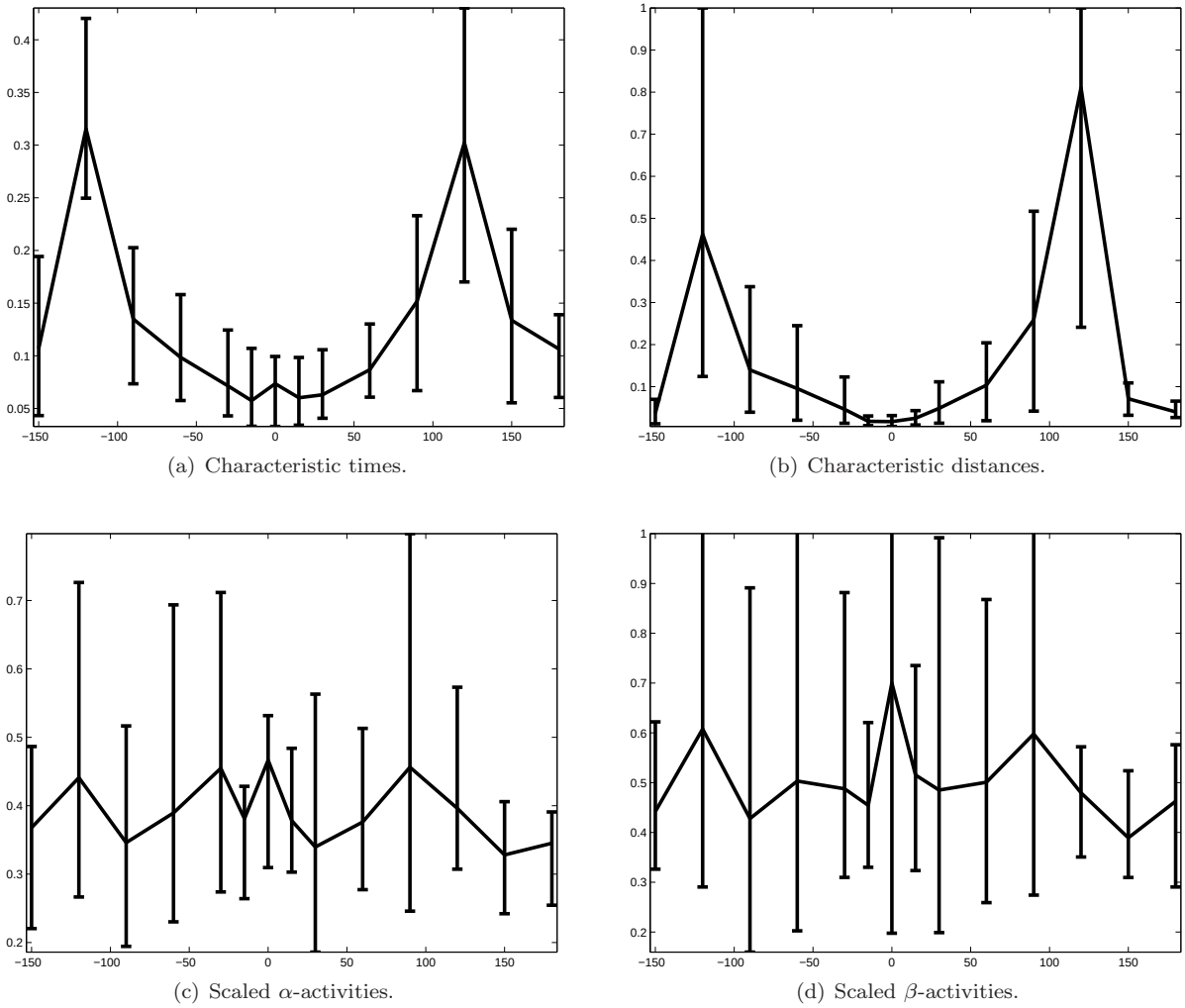
(a) Characteristic times.

(b) Characteristic distances.

(c) Scaled α-activities.

(d) Scaled β-activities.

Figure 27: Task 1 – Performance indices for frequent users in dependence of the bias angle φ.



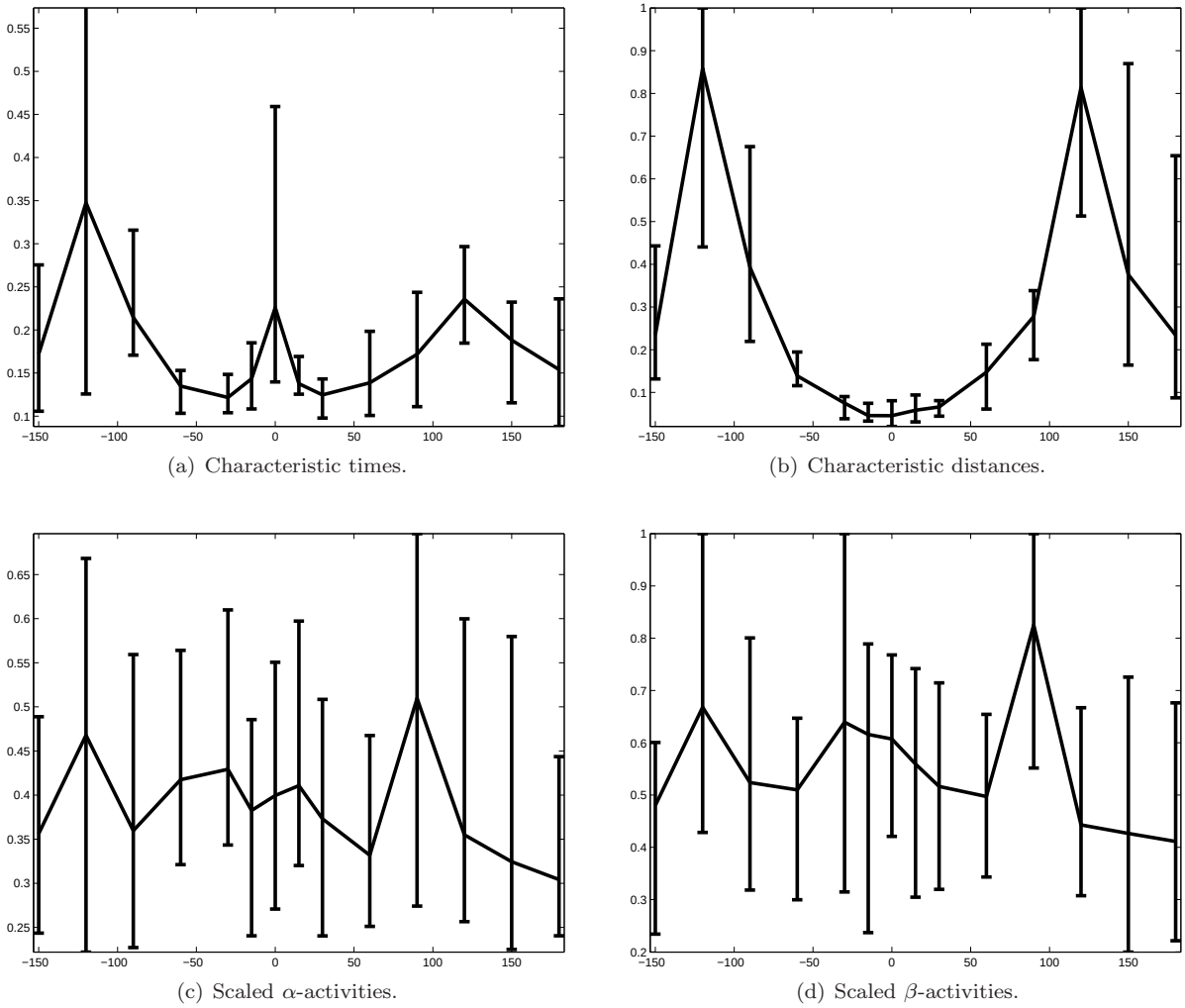
(a) Characteristic times.

(b) Characteristic distances.

(c) Scaled α-activities.

(d) Scaled β-activities.

Figure 28: Task 2 – Performance indices for frequent users in dependence of the bias angle φ.



(a) Characteristic times.

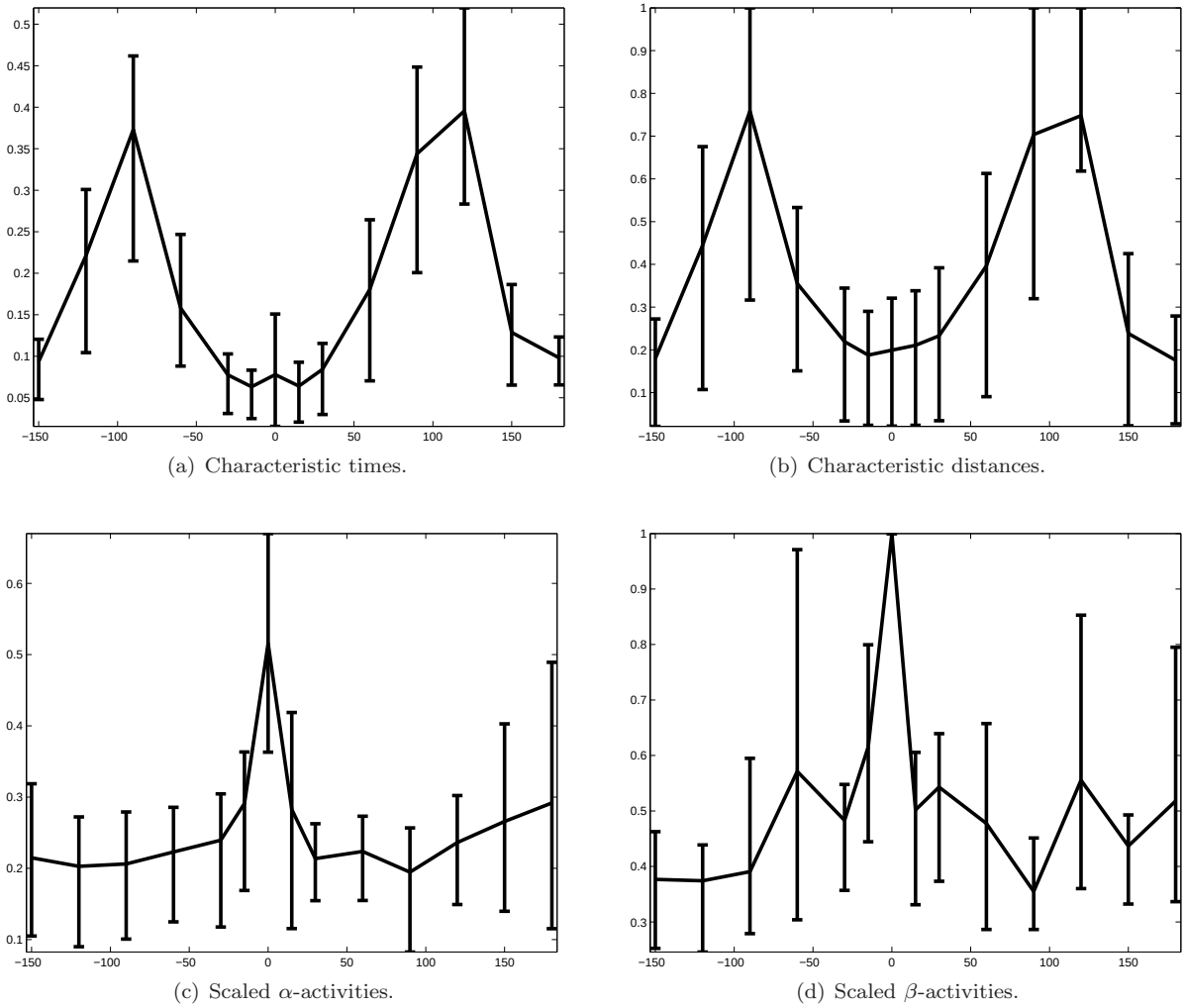
(b) Characteristic distances.

(c) Scaled α-activities.

(d) Scaled β-activities.

Figure 29: Task 3 – Performance indices for frequent users in dependence of the bias angle φ.

D Evaluation for frequent users



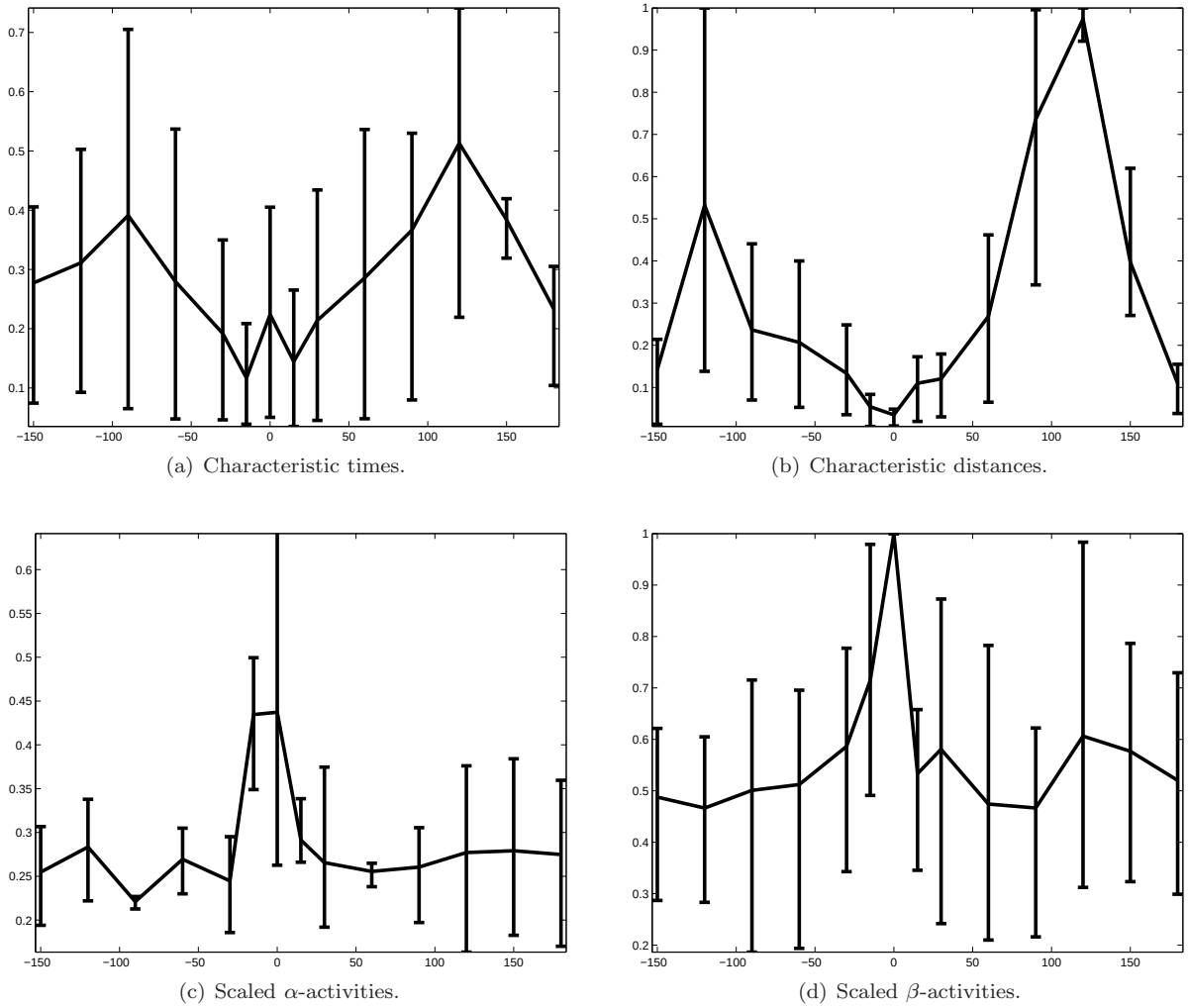
(a) Characteristic times.

(b) Characteristic distances.

(c) Scaled α-activities.

(d) Scaled β-activities.

Figure 30: Task 1 – Performance indices for heavy users in dependence of the bias angle φ.



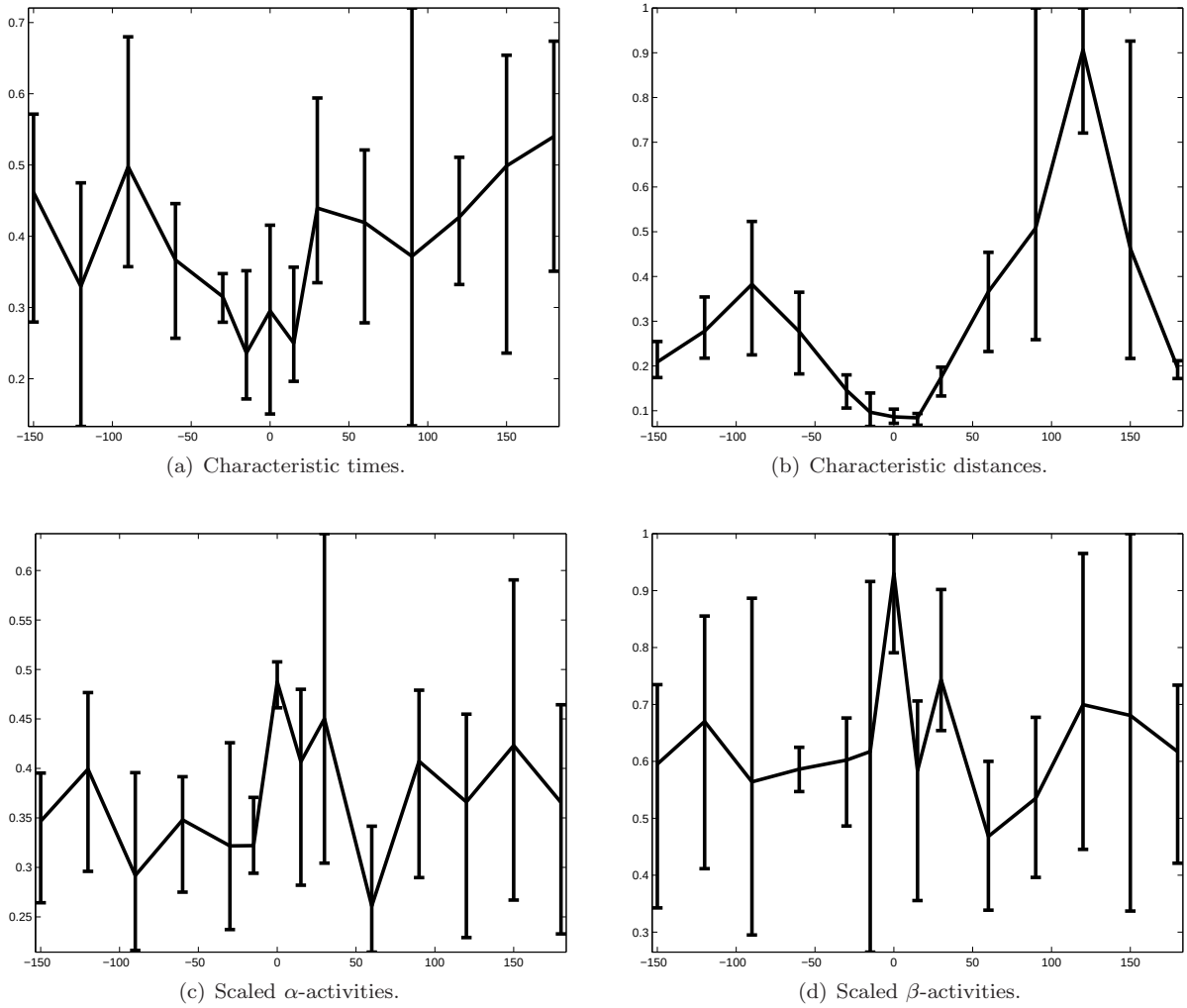
(a) Characteristic times.

(b) Characteristic distances.

(c) Scaled α-activities.

(d) Scaled β-activities.

Figure 31: Task 2 – Performance indices for heavy users in dependence of the bias angle φ.



(a) Characteristic times.

(b) Characteristic distances.

(c) Scaled α-activities.

(d) Scaled β-activities.

Figure 32: Task 3 – Performance indices for heavy users in dependence of the bias angle φ.