Evolutionary Brain Computer Interfaces

Riccardo Poli[1], Caterina Cinel[2], Luca Citi[1,3], and Francisco Sepulveda[1]

[1] Department of Computer Science, University of Essex, UK
[2] Department of Psychology, University of Essex, UK
[3] IMT Institute for Advanced Studies, Lucca, Italy

Abstract. We propose a BCI mouse and speller based on the manipulation of P300 waves in EEG signals. The 2–D motion of the pointer on the screen is controlled by directly combining the amplitudes of the output produced by a filter in the presence of different stimuli. This filter and the features to be combined within it are optimised by a GA.

1 Introduction

Brain-Computer Interfaces (BCIs) measure signals (often EEG) of brain activity intentionally and unintentionally induced by the user and translate them into device control signals (see [10] for a comprehensive review). BCI studies have showed that non-muscular communication and control is possible and might serve useful purposes for those who cannot use conventional technologies.

Event related potentials (ERPs) are among the signals used in BCI studies to date. ERPs are relatively well defined shape-wise variations to the ongoing EEG elicited by a stimulus and temporally linked to it. ERPs include an exogenous response, due to the primary processing of the stimulus, as well as an endogenous response, which is a reflection of higher cognitive processing induced by the stimulus [5,8]. The P300 wave is a large positive ERP with a latency of about 300 ms which is elicited by rare and/or significant stimuli. P300 potentials vary according to the degree of attention the user dedicates to a stimulus. It is, therefore, possible to use them in BCI systems to determine user intentions.

One important application of BCI is controlling 2–D pointer movements. There have been some attempts to develop BCI systems for this purpose, the most successful of which being based on frequency analysis (e.g., [11]), and those using invasive cortical interfaces (e.g., [6]). The former, however, require lengthy training periods before users can control them, while the latter are not very practical requiring surgery, presenting risks of infections, etc. These problems could be overcome by non-invasive systems based on the use of P300s. To date, however, only very limited studies with this approach have been performed [9,1] and these have reported promising but limited successes.

We use an evolutionary approach to parameter identification and optimisation in a P300-based BCI system for the two-dimensional control of a cursor and spelling on a computer. Because of their effectiveness, EAs are frequently applied in the area of image and signal processing. However, no application of EC in
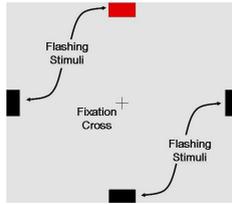
Fig. 1. Our BCI mouse when the stimulus "up" is presented

the area of BCI has been reported with the exception of our own preliminary results [4] where we used an evolutionary approach to process speller data.

The paper is organised as follows. In Sect. 2 we provide details on the participants, the stimuli, and the protocol followed during the training and use of our BCI system. We describe the various phases through which EEG signals are translated into control commands in the system in Sect. 3. In Sect. 4 we provide the details of the GA used. We report on the experimentation of the system in Sect. 5. Finally, we draw some conclusions in Sect. 6.

2   Experimental Methods

In building our BCI system we drew inspiration from Donchin Speller [8], where P300s were used to determine on which stimulus, out of a large set of stimuli concurrently presented, an observer's attention was focused. In particular, we used the following paradigm. Four gray rectangles are constantly superimposed on whatever is shown on a computer screen. They are unobtrusive being relatively small and aligned with the upper, lower, left and right borders of the screen (see Figure 1). Each rectangle corresponds to a possible direction of movement for the mouse cursor.

At 180 ms intervals, this static display is altered by temporarily changing the colour of one of the rectangles from gray to bright red. The stimulus remains brightly coloured for 100 ms, after which it becomes gray again. Which particular rectangle is selected for flashing is determined randomly without replacement. After each flash, the display returns to its standard gray colouring for 80 ms. To limit perceptual errors that can reduce the accuracy of BCI systems [2], the last rectangle to flash in each series is not allowed to be the first to flash in the following series.

The system has three modes of operation: training, validation and normal use. During the phases of acquisition of a training set for our control system, the experimenter selected one of the rectangles on the screen as a target, and participants were asked to focus their attention only on the target stimulus. During training, the screen had a light gray background with no stimuli other than the rectangles mentioned above. During validation phases, participants were asked to perform the same task with the same stimuli and an homogeneous background, except that, at this stage, the system had already been optimised

and we could immediately show participant the trajectory of the mouse pointer produced by their efforts. During normal use, of course, all sorts of windows and icons were present on the screen in addition to the rectangles necessary to control the BCI mouse.

During training and validation, we used the sequences of stimuli described above. Six participants were tested (aged between 23 and 44) of which one has a neuromuscular disability. Each run of our experiment involved presenting a full series of 4 flashing rectangles for 30 times. Participants were requested to count the flashes of one of the rectangles representing direction of motion. The process was repeated for each of the four possible directions, multiple times for each direction. Every few runs, participants were given a few minutes to rest.

We used a 19 channel setup in a Mindset24 System and an electrode cap compliant with the 10-20 international standard to acquire and digitise EEG signals. Efforts were made to obtain impedances below $7\,\text{k}\Omega$ in all experiments.

3   The System

Each EEG channel is lowpass filtered using a FIR filter of order $N = 30$. The coefficients of the filter were obtained via a least mean squares fit between the frequency response of a target ideal filter and the FIR filter's frequency response. The sampling frequency was 256 Hz, the frequency below which the signal was to be left unaltered was $f_{pass} = 34$ Hz, while the frequency above which all components of the signal were required to be totally suppressed was $f_{stop} = 47$ Hz. After low pass filtering, the signal is decimated to 128 Hz. To extract signal features, we then performed a Continuous Wavelet Transform (CWT) on the 19 channels.

The features are extracted for each epoch of the signal. An *epoch* is a window of 1 second starting when a stimulus is presented on the screen. In each epoch the system needs to process the EEG signals and appropriately emphasise and utilise a P300 if this is present in the epoch.

In our system, pointer control is determined by a filter which is applied to each epoch. In principle, the filter can use all of the coefficients of the CWT of an epoch. We use 30 different unequally-spaced scales between 2 and 40. Since P300s occur within a well known time window after a stimulus, we compute CWT only for 40 samples corresponding to a temporal window from 235 ms and 540 ms after the beginning of each epoch. This gives us a 3–D array $\mathbf{V}(c, s, t)$ of features, where c indexes the channel, s the scale and t the time corresponding to a feature. In total we have $22,800$ components.

Our controller performs a linear combination of a subset of elements of the feature matrix $\mathbf{V}$:

$$P(\mathbf{V}) = a_0 + \sum_{j=1}^{N} a_j \cdot \mathbf{V}(c_j, s_j, t_j). \tag{1}$$

where N is the number of terms in the filter, the coefficients c_j, s_j, t_j identify which component of $\mathbf{V}$ is used as the in the j-th term, and finally the values a_j are coefficients weighing the relative effect of each term.

The value $P(\mathbf{V})$ is then passed through a squashing function ϕ to produce a classifier output, $O(\mathbf{V}) = \phi(P(\mathbf{V}))$, in some interval $[-\Phi, \Phi]$, where Φ is a positive constant. The value of $O(\mathbf{V})$ is an indication of the degree to which an epoch contains a *target*, i.e., the stimulus on which a participant is focusing his/her attention for the purpose of selecting a particular direction of movement for the mouse cursor.

Every time a stimulus flashes, an epoch starts. For each epoch the system records the position of the corresponding stimulus on the screen and acquires and processes a 1 second segment of EEG signal. Epochs acquired during the period of training are annotated also with the direction the participant was asked to focus on.

In epochs corresponding to target stimuli, we expect to find a P300 wave, while in epochs where a non-target stimulus flashed this should not be present.

The motion of the pointer is directly determined by the squashed output of the filter. More precisely, the vertical motion of the pointer is proportional to the difference between the output produced by the filter when processing an epoch where the "up" rectangle was flashed and the output produced by the filter when processing an epoch where the "down" rectangle was flashed. Similarly, the horizontal motion of the pointer is determined by the difference between the outputs produced in response to the flashing of the "right" and "left" rectangles.

In order to turn P300s into mouse pointer motion, we divide the stream of epochs into groups of four. Each group contains epochs corresponding to the flashing of all four possible stimuli. As soon as a full group is acquired, the features and the output of the function $P(\mathbf{V})$ are computed for each of the four epochs. As a squashing function ϕ we used arctan.

As a result of these operations we obtain a tuple of output values (O_u, O_d, O_l, O_r), where subscripts refer to the up, down, left and right stimuli, respectively. These are used to compute the motion vector $v = (v_x, v_y)$ for the mouse cursor on the screen, as follows: $v_x = O_r - O_l$ and $v_y = O_u - O_d$.

4   Evolutionary Algorithm

Given the high noise and limited information content in EEG signals and the difficulty of the task of recognising P300s, it appeared immediately obvious that in our BCI mouse we had to search concurrently for the best features and the best parameters to use for control. It was also obvious that the size of the search space and its discontinuities required a robust search algorithm such as a GA. In our system the GA needs to choose the best features and classifier parameters to control the 2–D motion of the pointer.

For each participant, the GA has the task of identifying a high-quality set of parameters for Equation 1. These include: $N + 1$ real-valued coefficients a_j, N feature channels c_j, which are integers in the set $\{0, \cdots, 18\}$, N integer feature scales s_j in the set $\{0, \cdots, 29\}$, and N integer feature samples t_j in the set $\{0, \cdots, 39\}$.

The operation of feature selection is performed by the GA choosing N tuples (c_j, s_j, t_j), while the classifier training is performed by optimising the corresponding coefficients a_j. These operations are performed jointly by the GA.

The representation used here is simply a concatenation of the floating-point and integer parameters of the linear classifier in Equation 1. For simplicity, we decided to encode integer parameters c_j, s_j and s_j as real numbers, taking care, of course, of rounding them to the nearest integer before using them in Equation 1 (e.g., during fitness evaluation). We were then able to use any operators available for real-coded GAs. In particular, we decided to use blend crossover [7], also known as BLX-α, where the offspring $(a_1^s, \ldots, a_i^s, \ldots, a_n^s)$ is obtained, component by component, as $a_i^s = a_i^1 + c_i(a_i^2 - a_i^1)$, where, for each i, a different value of c_i is drawn uniformly at random from the interval $[-\alpha, 1 + \alpha]$, α being a suitable non-negative constant. We used $\alpha = 0.1$.

As a selection operator we chose tournament selection with tournament size 3. To maximally speedup evolution we used a steady state GA. As a replacement strategy we used negative tournament selection.

As a mutation operator we used headless chicken crossover, i.e., the recombination (via BLX) between the individual to be mutated and a randomly generated individual.

Given the complexity of the task we used very large populations including $50,000$ individuals.

While we were able to use a standard representation and standard genetic operators, the design of the fitness function involved much more work and required numerous successive refinements. The fitness function we eventually settled for is described below.

The natural objective function for a mouse is, of course, the extent to which the pointer was moved in the desired direction. So, this is clearly a necessary component in our fitness function. However, this is not enough. For example, it is possible that the pointer moved a great deal in the desired direction, while at the same time drifting significantly in a direction orthogonal to the desired one. A fitness function based on the natural objective would reward this behaviour, effectively leading to very sensitive but not very precise controllers. So, clearly the problem is multi-objective, that is we want to obtain both maximum motion in the desired direction *and* minimal motion in the orthogonal direction.

To deal with this problem we adopted a penalty method to combine the multiple objectives into a single fitness measure. So, the fitness function f is a linear combination of the objectives ω_i of the problem:

$$f = \sum_i \lambda_i \omega_i \qquad (2)$$

where λ_i are appropriate coefficients (with λ_1 conventionally being set to 1).

In the mouse control problem we used three different objectives. The first objective, ω_1, assesses the extension of the motion in the target direction, the other two, ω_2 and ω_3, evaluate the motion in orthogonal direction. In particular, ω_2 assesses the average extension of the motion in such a direction at the end of runs. It, therefore, ignores any errors that have been later cancelled by errors

with opposite sign (i.e., in the opposite direction). Instead, ω_3 evaluates the average absolute error deriving from motion orthogonal to the target direction. So, it assesses the extent to which the trajectory towards the target is convoluted.

The performance of the controller was evaluated on the basis of its behaviour over groups of 30 repetitions of a command (up, down, left or right), which we will term *runs*. In order to ensure that the controller performed well in all directions, these runs were acquired for all possible directions (and, in fact, multiple times for each direction to limit risks of over-fitting). All the resulting trials formed the controller's training set.

In each of the examples in the training set the controller produced a velocity vector. Let us call $v^{i,t} = (v_d^{i,t}, v_o^{i,t})$ the velocity vector produced at repetition i in the t-th run. This vector is expressed in a reference system where the component $v_d^{i,t}$ represents the motion in the target direction, while $v_o^{i,t}$ represents the motion produced in the direction orthogonal to the desired direction.

The objectives can then be expressed as as follows:

$$\omega_1 = \nu \sum_{t=1}^{N_r} \sum_{i=1}^{30} v_d^{i,t} \qquad \omega_2 = \nu \sum_{t=1}^{N_r} \sum_{i=1}^{30} v_o^{i,t} \qquad \omega_3 = \nu \sum_{t=1}^{N_r} \sum_{i=1}^{30} |v_o^{i,t}|$$

where N_r is the number of runs and $\nu = \frac{1}{30 \times N_r}$ is a normalisation factor. Since we want to maximise ω_1, but minimise $|\omega_2|$ and ω_3, the penalty coefficients were set as follows: $\lambda_1 = 1$, $\lambda_2 = -0.2$ and $\lambda_3 = -0.2$.

5   Results

5.1   Controlled Experiments

A *4-fold cross-validation* has been applied to train the system and test its performance and generalisation ability. So, for each participant the total of 16 runs has been split in groups of four each containing one run for each direction (1–4, 5–8, etc.). Therefore there were 4 groups for each participant. Then two of these groups have been used as training set while a third one as validation set. For each of the 12 combinations a population (i.e., a different classifier) has been evolved. Runs lasted 40 generations.

The three different objectives (Sect. 4) have been evaluated for the training-set as well as for the validation-set. The analysis of the results (see [3] for details) shows that all participants were able to move the pointer in the target directions (as quantified by ω_1) with very limited lateral error at the end of a sequence of commands (quantified by ω_2). In fact, if we consider ω_1 as the amplitude of the signal and ω_2 as the standard error of the noise in our system, then $\omega_1/|\omega_2|$ gives us an idea of the signal to noise ratio (SNR) during the movement of the pointer. Averaged over our 6 participants, $\omega_1/|\omega_2|$ is 28.1, which gives us a good SNR = 29.0 dB.

These results have been obtained by processing offline data acquired with the BCI mouse in *training* mode. In this mode stimuli are presented to a participant

but no feedback is provided as to the extend and direction of motion of the pointer. This is done to ensure that the cleanest possible dataset is gathered. As soon as the acquisition of an appropriate training set is completed the system performs all the necessary preparatory steps, including filtering and feature detection, for the application of the GA. GA runs are typically successful within 30 generations. The data preparation and the training process require between 5 to 10 minutes. The system can then be used.

5.2   Realistic Applications

Having established that the approach works and appears to be accurate, we decided start exploring more realistic uses of the mouse with participant D – the oldest and worst performing of our subjects. So, the systems we describe below and the results obtained with this participant can only be considered as first, conservative indications of what can be achieved. We will need to perform much deeper investigations of these systems in future research.

The first application we considered was the use of the mouse to control a standard computer system (in particular a personal computer running the Windows XP operating system). In this application, the screen included our four flashing rectangles and a fixation cross as in the tests of the BCI mouse described above. However, naturally, the background was not uniformly gray: instead it included the standard windows and icons normally available in a Windows machine (see Figure 2). In order to make the system as user friendly as possible for people with limited or no saccadic control, instead of moving the mouse pointer on a fixed screen, we decided to scroll the screen, thereby ensuring that the entities of interest for the user were always near the fixation cross. In addition, we used a zoom factor of 2 to ensure maximum readability.

We did not retrain the system with this richer set of stimuli. Nonetheless, the user was immediately able to move the pointer to the desired locations on the screen, albeit slowly at first, despite the large number of moving elements on the screen being a clear distracting factor. After a few minutes the participant had clearly improved his ability to control the system, as shown in the "film strip" in Figure 2, where the participant was able to scroll from the top left corner to the lower left corner of the main window in around 13 seconds, which we find encouraging. We expect that much better performance could be obtained by retraining the control system for this specific application and by giving participants some time to adjust to the system. Also, as we mentioned before, participant D was the worst performing in our study. So, we should expect to see better average performance when looking at a larger set of participants.

Naturally, the availability of 2–D pointer motion makes it possible to input data of any other type. Numerous systems exist, for example, that can turn mouse movements into text. So, as our second test application we designed a simple prototype speller system. We wanted to test the viability of spelling by BCI mouse and determine whether the accuracy of the control is sufficient for the task. In our BCI speller system, in the centre of the screen a circle with 8 sectors is drawn. Each sector represents one of the 8 most used characters in English
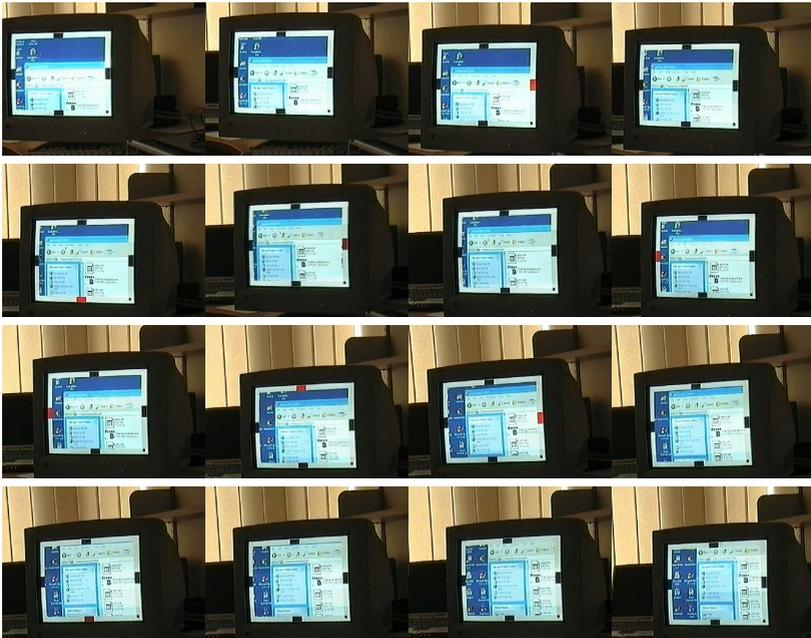
Fig. 2. Snapshots (ordered from left to right and from top to bottom) taken every 20 frames (800ms) from a video recording of our BCI mouse in action. The user wanted to scroll down.

(Fig. 3). The idea is for the system to use a T9 cell-phone-type predictor which makes it possible to input complete words with very few key presses. However, we made not use of this feature during our tests.

The system starts with the pointer at the centre of the screen. In order to enter a character, the user moves the cursor in the desired sector of the screen. To give time to the user to move the pointer into the desired sector, character input is prevented for a certain amount of time (15 s in our tests). The character is acquired as soon as the cursor reaches the perimeter of the circle or a maximum time has elapsed. The entered character is appended to the current string shown at the upper right in the screen and the pointer is repositioned at the centre.

The preliminary testing of this system with participant D has been promising, despite, again, our not retraining the system for this application, and the tests being hampered by technical problems. The objective of the tests was to input the sentence "she has a hat". The user repeated the task three times. The first time the user made 2 errors out of 7 characters inputting characters at a rate of one every 35 seconds. In the second attempt the user was able to spell "she has" without any errors but at a much slower rate (one character every 80 seconds). The participant was able to do a further test spelling again without any errors "she has" plus an additional space, before feeling drained and unable to continue. In this last test the user inputted characters at a rate of one every 54 seconds.
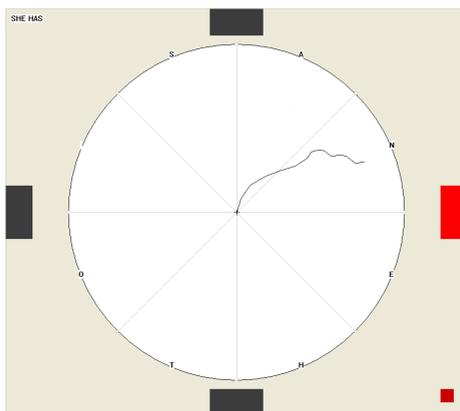
Fig. 3. Our BCI-mouse based speller after the user was been able to correctly enter the text "SHE HAS"

So, despite us running these tests with a user who was tired after the many other tests he had run during his session, after rapidly adjusting to the new application, the user was able to control the device and input data without errors, although this was achieved with a considerable concentration effort and at a very slow rate of 3.2 bits/minute. Again, we should expect much better performance with well-rested and better performing participants. Also, training the control system specifically for this application and giving participants some time to adjust to the system would certainly provide significant performance improvements. We plan to further develop this system in future research.

6   Conclusions

In this paper we have proposed a BCI system based on the use of P300 waves. The system is analogue, in that at no point a binary decision is made as to whether or not a P300 was actually produced in response to a particular stimulus. Instead, the motion of the pointer on the screen is controlled by directly combining the amplitudes of the output produced by a filter in the presence of different stimuli.

Beyond providing carefully designed stimuli, a rich set of features (wavelet coefficients) and a flexible combination mechanism (a filter) through which we thought a solution to the problem of controlling a pointer via EEG could be found, we actually did not do any other design. The biggest part of the design in this system (i.e. the feature selection, the selection of the type, order and parameters of the controller) was entirely left to a genetic algorithm.

The performance of our system has been very encouraging. As mentioned in Sect. 5.1, all participants have been able to use the system within minutes. The GA was very effective and efficient at finding good designs for the system. Indeed, it succeed in every run, suggesting that we had chosen the infrastructure for the system and the feature set reasonably well. In validation, the trajectories

of the pointer have achieved high accuracy. The system issues control commands at a much faster rate (approximately once per second) than other P300-based computer mice previously described in the literature.

These encouraging results indicate that there may be a lot more information about user intentions in EEG signals, and that perhaps, traditional design techniques may be a limiting factor. There is very limited knowledge as to how to manually design analogue BCI mouse systems. Evolution, on the other hand, being entirely guided by objective measures of success (the fitness function), was able to achieve this almost effortlessly. The systems evolved were also rather robust. For example, it was possible to control the mouse even in situations very different from the ones originally considered in training, such as in tests with control in a real Windows environment and in our BCI speller, without retraining the system. So, we suspect that other single-trial BCI applications might benefit from the use of properly guided evolutionary algorithms. We plan to explore this in future research.

References

1. F. Beverina, G. Palmas, S. Silvoni, F. Piccione, and S. Giove. User adaptive BCIs: SSVEP and P300 based interfaces. *PsychNology Journal*, 1(4):331–354, 2003.
2. C. Cinel, R. Poli, and L. Citi. Possible sources of perceptual errors in P300-based speller paradigm. *Biomedizinische Technik*, 49:39–40, 2004. Proceedings of 2nd International BCI workshop and Training Course.
3. L. Citi, R. Poli, C. Cinel, and F. Sepulveda. P300-based brain computer interface mouse with genetically-optimised analogue control. Technical Report CSM-451, Department of Computer Science, University of Essex, May 2006.
4. L. Citi, R. Poli, and F. Sepulveda. An evolutionary approach to feature selection and classification in P300-based BCI. *Biomedizinische Technik*, 49:41–42, 2004. Proceedings of 2nd International BCI workshop and Training Course.
5. E. Donchin and M. G. H. Coles. Is the P300 a manifestation of context updating? *Behavioral and Brain Sciences*, 11:355–372, 1988.
6. J. Donoghue. Connecting cortex to machines: recent advances in brain interfaces. *Nature Neuroscience*, 5:1085–1088, 2002.
7. L. J. Eshelman and J. D. Schaffer. Real-Coded Genetic Algorithms and Interval Schemata. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, San Mateo, CA, 1993. Morgan Kaufmann Publishers.
8. L. A. Farwell and E. Donchin. Talking off the top of your head: A mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70:510–523, 1988.
9. J. B. Polikoff, H. T. Bunnell, and W. J. B. Jr. Toward a p300-based computer interface. In *Proc. Rehab. Eng. and Assistive Technology Society of North America (RESNA '95)*, pages 178–180, Arlington, Va, 1995. RESNAPRESS.
10. J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan. Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113(6):767–791, Jun 2002.
11. J. R. Wolpaw and D. J. McFarland. Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proceedings of the National Academy of Sciences*, 101(51):17849–17854, 2004.