**Preprint Version** -The final publication is available at Springer via
http://dx.doi.org/10.1007/978-3-319-02753-1_30

# PolyMorph: A P300 Polymorphic Speller*

Alberto Casagrande[1], Joanna Jarmolowska[2], Marcello Turconi[2], Francesco Fabris[1], and Piero Paolo Battaglini[2]

[1] Dept. of Mathematics and Geosciences, University of Trieste, Via Valerio 12/1, 34127 Trieste, Italy
[2] Dept. of Life Sciences, University of Trieste, via Fleming 22, 34127 Trieste, Italy

*This paper is dedicated to the memory of Col. Paolo Corazzi. He motivated the innovations introduced in this work and passed away on the very day in which the first draft of it had been completed.*

**Abstract.** P300 is an electric signal emitted by brain about 300 milliseconds after a rare, but relevant-for-the-user event. Even if it is hard to identify and it provides a low-rate communication channel, it raises the interest of the scientific community because it can be used in cases in which other evoked potentials fail. One of the applications of this signal is a speller that enables subjects who lost the control of their motor pathways to communicate by selecting one by one each character of a sentence in a matrix containing all the alphabet symbols. This paper provides an improvement of this paradigm and, in particular, it aims at reducing both the error rate and the time required to spell the entire sentence by exploiting the redundancy which is present in all the natural languages.

## 1 Introduction

ERPs (Event Related Potential) are brain activities elicited in response to external or internal events. P300 is a positive peak of an ERP that represents an endogenous cognitive response to a desired stimulus [1]. Its peak has a delay of about 300 milliseconds from the stimulus from which follows the name.

This wave is usually associated to the so called "oddball paradigm" which enables to identify the occurrences of desired events. A subject is presented with a sequence of episodes: some of them are desired stimuli, the others are of no interest to the user. Whenever a desired event arises, it elicits an ERP characterized by a P300 component which can be identified by an electroencephalography (EEG) [2,3]. Among other things, the oddball paradigm enables individuals affected by cerebral stroke, neurodegenerative disease, and locked-in syndrome to communicate by using P300-based spellers. A user observes a set of randomly flashing characters and he focuses his attention on a symbol of interest. Its enlightenment triggers a P300 signal which can be identified and, in this way, it

* This work has been partially supported by Istituto Nazionale di Alta Matematica (INdAM).

Preprint Version -The final publication is available at Springer via
http://dx.doi.org/10.1007/978-3-319-02753-1_30

is possible to spell, character by character, a complete sentence. Unfortunately, EEG reflects thousands of simultaneously ongoing brain processes and the response to a stimulus is not visible in a single trial: many repetitions of the same stimulus are required and this leads to an extremely low spelling rate i.e., the number of symbol selections per time unit.

Because of this, many strategies have been proposed so far to increase the efficiency of P300-based spelling. Farwell *et al.* suggested to dispose all the symbols into a matrix, dubbed *selection matrix*, identify the row and the column of interest (i.e., those that contain the desired character), and, ultimately, deduce the desired character itself [4]. This is achieved by flashing entire rows and columns in place of single elements and it requires a set of EEG measurements per selection that is proportional to the sum of rows and columns of the selection matrix. Of course, the smallest ratio between the number of measurements and the number of selectable characters per selection can be obtained by adopting a square selection matrix. Blankertz *et al.* proposed a two step character selection on a tree whose nodes are visually-presented as hexagons [5]: the first selection discriminates between six groups of six symbols each, while the second selections identifies the aimed symbol in the selected group. Pires *et al.* presented an analogous strategies, but they also noted that the group transition rate depends on the organization of groups themselves and suggested how to improve it [6]. Ryan *et al.* integrated suggestions, based on prefix of the current word, in the classical row-column selectors [7]. The suggestions themselves were not presented inside the selection matrix, but in additional windows. Despite an improvement in the character per minute rate, the proposed system significantly decreased the accuracy with respect to the classical paradigm. More recently, D'Albis *et al.* described a predictive speller whose symbols are dynamically organized [8].

We propose an enhancement of the classical row-column speller, named *PolyMorph*, that suggests how to complete the current word based on what has already been written. The suggestions are inferred from a knowledge base and take into account the spelled prefixes of both words and sentences. By a priori choosing the dictionary, PolyMorph is able to both reduce the number of selectable symbols according to the past selections and dynamically resize the selection matrix exhibiting a sort of polymorphism from which follows its name. Moreover, it exploits the free space left by the missing symbols and maximizes the size of the displayed fonts. In some cases, the proposed speller also admits the selection of sequences of characters rather than of single symbols. Not only all these features considerably reduce the number of stimuli required to spell a complete sentence, but apparently increase the accuracy of the spelling process.

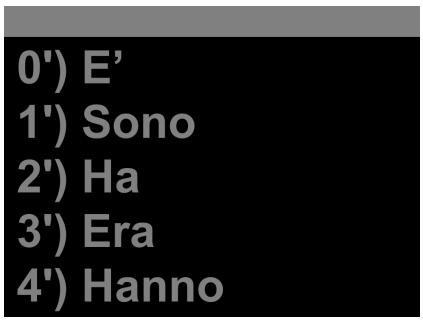
The paper is organized as follows: Section 2 presents PolyMorph and its features, while Section 3 briefly describes the implementation. The results of some tests are shown and analyzed in Section 4 and, finally, Section 5 draws conclusions and suggests future developments.

Preprint Version - The final publication is available at Springer via
http://dx.doi.org/10.1007/978-3-319-02753-1_30

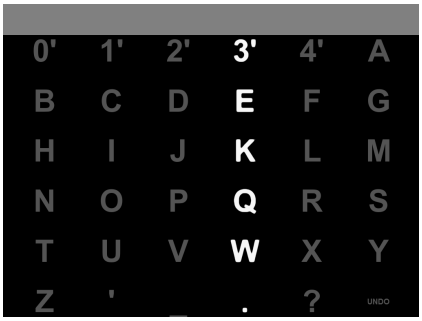
## 2 PolyMorph's Features

We aim at decreasing the number of selections needed to "write" sentences by exploiting the redundancies that are always present in natural languages. We try to both minimize the number of selectable characters per selection and suggest the most probable words completing the already written prefix. On one hand, we want to leave the chance to compose known words to form new sentences, but, on the other hand, we want to avoid the sequences of characters that are not admissible in the user dictionary.

PolyMorph maintains a knowledge base, initialized by using a phrasebook, that, in some sense, summarizes the user language. We clearly identify two distinct levels in this base: a *lexicographic level*, which stores all the known words and all the admissible character combinations, and a *syntactic level*, which memorizes all the sentences that either are contained into the phrasebook or have already been selected by the user.




(a) Suggestion phase: the most selected/frequent words are suggested and associated to unique numeric IDs.




(b) Selection phase: the selection matrix is shown and the P300 measurement proceeds.

**Fig. 1.** The working cycle of PolyMorph is split into two phases.

The lexicographic level allows to present to the user only those symbols that, taking into account the given dictionary, are compatible with the already spelled string (see Fig. 2(b)). The variability in the set of the proposed symbols leads to a polymorphic selector (from which the name *PolyMorph*) which tries to minimize the size of the selection matrix at each selection. This has two main effects: it decreases the number of the P300 measurements that are required for the selection of each symbol, reducing the selection time, and it enables to increase the size of the fonts used in the symbol presentation (see e.g., Fig. 1(b) and 2(b)). Since the amplitude of the P300 component is related with the strength of the stimulus that causes it [9], by increasing the font size we decrease the probability of an error in the identification of the aimed symbol even if it is

well known that decreasing the matrix size may reduce the accuracy [10]. The lexicographic level can also identify the words that complete the current word selection and either have been selected more times so far or are the most used in the original dataset. These are then suggested to the user who can spell them with a single selection.

As the lexicographic level, the syntactic level is used to identify words that are worth to be suggested, but it takes into account the entire spelled string in place of the word prefix that user has spelled. In particular, it can furnish the list of words that follow a prefix $p$ and together with $p$ either have already been selected at least once or are present in the initial phrasebook.

*Example 1.* Let "the word th" be the string spelled by the user. The lexicographic level might propose the words "those", "the", or "that". However, neither "the word the" nor "the word those" appear to be a prefix of an English sentence and there are many chances that they both have been never spelled and are not even contained in the initial phrasebook. If this is the case, the syntactic level would suggest exclusively "that".

The syntactic level does not constrain sentence spelling and users can combine known words to obtain sentences that are not present in the original phrasebook. This is the main difference with respect to the lexicographic level whose set of stored words cannot be upgraded. Because of this, the syntactic level has no impact on the characters proposed by the speller, but it may affect the set of words suggested by PolyMorph.

Beyond the grammatical aspect, this level contains, in some sense, semantic information about sentences: if a phrase is a non-sense, then it will never be selected by the user and will not be stored in the knowledge base.

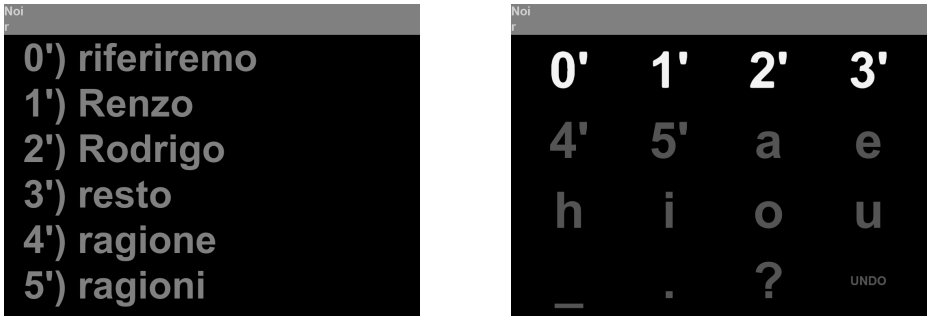
Each selection process is split into two phases: the *suggestion phase* and the *identification phase* (see Fig. 1). The former presents the suggested words and associates them to numeric IDs dubbed *suggesting symbols*. The latter shows a selection matrix, containing also the suggesting symbols, and performs the P300 measurements as done in the row-column paradigm. Whenever a suggesting symbols is chosen, the unwritten suffix of the word associated to it is selected.

## 3 Implementation

While the PolyMorph user interface is massively based on the row-column speller, named *P3Speller*, included into the BCI2000 framework [11], its internals have been developed from the scratch. At the beginning of the development, we had planned the use of web search engines, such as Google or Yahoo!, to both compose the spelling matrix and identify the words to be suggested. This choice would have avoided the need of an initial phrasebook, but at the same time it would have prevented us from achieving a user specific knowledge base. Because of this, we dropped this idea and adopted *radix tree* as main data structure.

A *radix tree* [12] is a tree used to memorize a set of strings. The edges of the tree are labeled by texts and, in the case of edges leaving the same node, the

Preprint Version - The final publication is available at Springer via
http://dx.doi.org/10.1007/978-3-319-02753-1_30




(a) The number of suggested words is the least natural, greater than a given parameter, such that all the selection matrix elements are not empty.

(b) The size of the matrix (and of fonts) depends on both the letters that begin the suffices of the current word and the suggested words.

**Fig. 2.** Both the number of suggested words and the size of the selection matrix may change during the computation.

labels are pairwise distinct. Any node is associated to the string corresponding to the concatenation of all the labels in the path that connects the root to the node itself. In particular, each internal node of the tree represents a string that is a maximal common prefix of at least two strings in the original data set. It follows that the root of the tree is associated to the empty string and the leafs represent the strings stored in the tree itself.

Thanks to the radix tree representation, PolyMorph is able to both suggest the suffices that complete a given string and reduce the number of selections required to spell both words and sentences. For example, let us consider the knowledge base presented in Fig. 3. Whenever the user selects the character "g", PolyMorph spells the entire prefix "goo" as all the words contained in the knowledge base that begin with "g" share "goo" as prefix. Since this feature corresponds to spelling an entire label of a radix tree as a consequence of a single selection, we call it *label selection*.

By enriching each node of a radix tree with both the number of occurrences of $s_n$ in the original data set and the number of user selections of $s_n$, we are able to identify the strings that most likely complete a given prefix. We call this data structure *statistical radix tree*.

PolyMorph maintains two statistical radix trees: one for the lexicographic and one of the syntactic level. The former stores all the words that may occurs in a sentence. The latter contains all the sentences that either are in the original dataset or have been selected by the user. The statistical information stored in the two trees is updated at each sentence selection and, whenever a new sentence is selected, the sentence itself is memorized into the tree of the syntactic level.

In order to select a subset of all the words that can be suggested, we established an order relation $\succ$ between strings that takes into account the statistical

Preprint Version -The final publication is available at Springer via
http://dx.doi.org/10.1007/978-3-319-02753-1_30

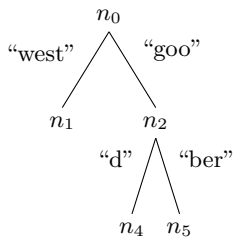



**Fig. 3.** The radix tree of { "good", "goober", "west" }.

information associated to the strings themselves. The relation $s_1 \succ s_2$ holds if and only if either $s_1$ has been spelled by the user more times than $s_2$ or if they have been spelled the same number of times and there are more occurrences of $s_1$ than of $s_2$ in the initial phrasebook. PolyMorph suggests the greater $m_c$ words with respect to $\succ$, where $m_c$ is a parameter. In doing so, it gives priority to the strings extracted from the syntactic radix tree and, whenever they are not enough, it recovers the missing from the lexicographic level. In those cases in which the same string is returned by both the levels, PolyMorph pays attention not to suggest it twice. The chosen words are presented during the suggestion phase and each of them is associated to a suggestion ID (see both Fig. 1 and 2). The suggesting cells are labelled by suggestion IDs and, whenever one of them is selected, the missing suffix of the word corresponding to its ID is spelled.

## 4 Tests and Analysis

In order to validate PolyMorph, we performed two kinds of tests: an *in-vivo* set of tests, which aimed at evaluating PolyMorph on real users and proposed the selection of two sentences, and an *in-silico* set of tests, which statistically strengthened the analysis and considered a wider set of phrases. In both the cases, we also used P3Speller to compare the efficiency of the two spellers.

For the *in-vivo* set of tests, we considered 10 healthy subjects. All of them were Italian native speakers and, because of this, we built the PolyMorph knowledge base by using an Italian phrasebook. In particular, we collected 111176 Italian sentences, containing 51590 distinct words, from books, internet, and journals. The mean sentence length was 37.2 characters and that of words is 5.3. By exploiting only the label selection feature and with no suggestions, PolyMorph could spell them with an average of 9.5 and 4.6 selections, respectively.

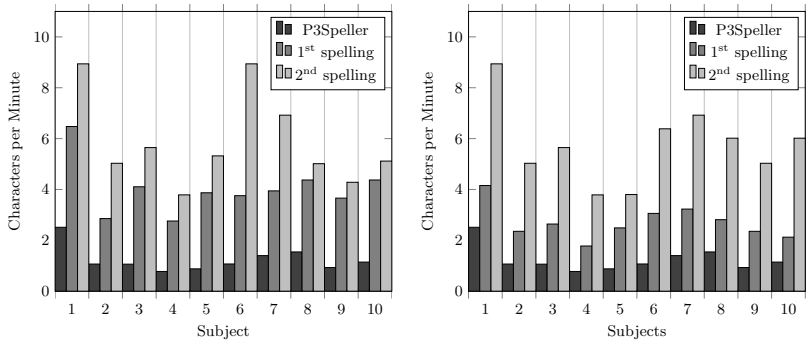
Before the experiments, we set some of the speller parameters. Both the stimulus duration and the time between two consecutive stimuli (ISI) were set to 125ms and the time between the appearance of the selection matrix and the first stimulus (pre-sequence duration) to 3s for both P3Speller and PolyMorph. The time between the last stimulus and the change of selection matrix (post-sequence duration) in P3Speller was set to 3s, while we forced PolyMorph to show

Preprint Version - The final publication is available at Springer via
http://dx.doi.org/10.1007/978-3-319-02753-1_30

the word suggestions for 10s. The number of repetitions of the same stimulus (sequence stimulus) was chosen user by user by performing a calibration on P3Speller: from the first to the tenth user were set to 6, 14, 12, 20, 13, 6, 9, 11, 14, and 11, respectively.

The *in-vivo* experiments consist in the spelling of two sentences: "*Piace tanto alla gente.*", sentence A, and "*Sono andato sulla luna.*", sentence B (i.e., "People like it very much." and "I have been on the moon."). All their words are included in the phrasebook, but only the former sentence is contained in it. We demanded to spell each sentence twice and we also asked the subjects to use P3Speller and spell sentence A character by character. Let us notice that spelling the same sentence twice in a row does not furnish statistically meaningful data since it does not occur quite often in normal conditions. However, it provides an upper bound (first spelling) and a lower bound (second spelling) for the number of selections required by the process. All the results of the *in-vivo* experiments will be statistically strengthen by the *in-silico* tests. Moreover, in order to avoid bias, the order of tests fed to each user was randomly selected.

PolyMorph outperforms P3Speller in writing both the sentences (see Fig. 4). The differences between P3Speller and PolyMorph on the first spelling of sentence B are due to the lexicographic level. On the contrary, the increased efficiency of the second spelling with respect to the first spelling of the same sentence is due to the syntactic level and so it is for the differences between the first spelling of sentence A, which is initially included into the knowledge base, and the first spelling of sentence B, which is a new sentence.

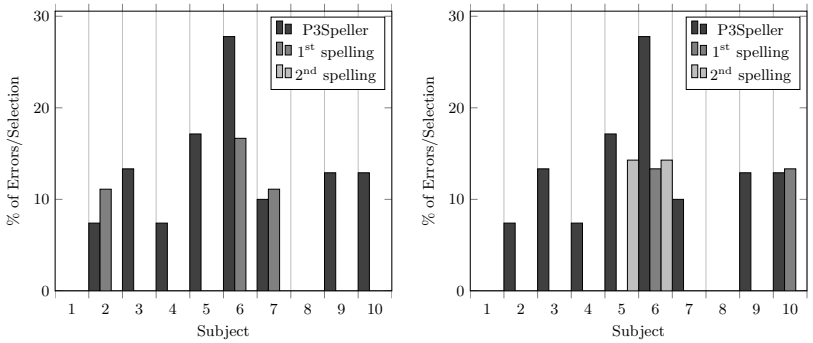



(a) Sentence A (b) Sentence B

**Fig. 4.** *In-vivo* experiments: characters per minute.

PolyMorph also decreases the error-rate with respect to both spelled characters and selections (see Fig. 5). This may be explained by four reasons: first, it reduces the number of selections required to spell a sentence and, as a consequence, the probability of wrong selections. Second, due to the polymorphism,

the selection matrix sometimes contains a low number of symbols. In such cases, the font size is increased and the P300 signal is more detectable. Third, since the suggested words appear always on the first two rows of the selection matrix, the users tend to focus their attention there and, probably, they filter the noisy stimulus coming from the remaining part of the matrix. As soon as the aimed word is suggested, the error-rate decreases. Finally, PolyMorph reduces the number of stimuli required to spell a sentence: this increases the user comfort and, thus, his ability of focusing on a single event.




(a) Sentence A (b) Sentence B

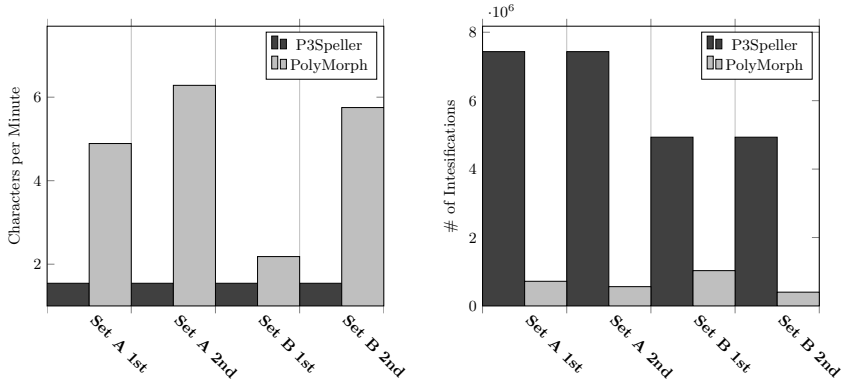
**Fig. 5.** *In-vivo* experiments: errors per selection.

For the *in-silico* tests, we maintained the same phrasebook and set the sequence stimulus to 12: the average of that used during the *in-vivo* tests. We wrote a program to automatically spell sentences with PolyMorph and we built two sets of 500 phrases to be spelled: the set A, which contains 20137 characters, is a subset of the original phrasebook, while none of the sentences of the set B, counting 13200 symbols, are initially included into the knowledge base. All their words are present in the phrasebook. As done for the *in-vivo* experiments, the spells were repeated twice and the sentence order was randomly chosen.

With respect to the spelling time, the *in-silico* experiments confirm the results obtained *in-vivo* (see Fig. 6(a)). They also underline that PolyMorph bids a number of visual stimuli much smaller than that of P3Speller (see Fig. 6(b)). In particular, in the worst case, i.e., trying to select a phrase that is not memorized into the knowledge base, the former exhibits an average of one fifth of the stimuli necessary to the latter. Despite the *in-silico* experiments do not provide any piece of information about selection errors, these data suggest an improved user experience and we are confident that this is connected with a reduced error-rate as underlined during the analysis of *in-vivo* experiments.

All the files used during above experiments and a set of tables reporting all the results are available at the URL `http://polymorph.units.it`.

Preprint Version - The final publication is available at Springer via
http://dx.doi.org/10.1007/978-3-319-02753-1_30




(a) Throughput (b) # of Stimuli

**Fig. 6.** *In-silico* experiments: the number of repetitions per stimulus was set to 12.

## 5 Conclusions and Future Works

This paper presents a new P300-based speller, named PolyMorph, which, among the other features, suggests the most probable word that follows what has already been spelled. While this idea is not new (e.g., see [7] and [8]), our approach differs from the literature in four main aspects: the interface used to suggest words, sentence-based suggestions, the presence of a polymorphic speller matrix, and the, so called, label selection. As far as we know, all the hint-based selectors proposed so far show the selection matrix and the suggested words at same time in two different windows. This feature forces the use of large displays and requires to rapidly move the attention from one side to the other of the screen. On the contrary, PolyMorph splits each working cycle into a suggestion phase and a identification phase and requires to associate the wanted word to a suggesting symbol before the identification phase begins.

All the remaining features introduced in PolyMorph cooperate to increase the speller efficiency. Sentence-based suggestions exploit a knowledge base encoding the linguistic habits of users and increase the odd of identifying the next word to be spelled. The polymorphism allows to reduce the P300 measurements and, as a consequence, the overall selection time by removing unnecessary symbols from the selection matrix. Finally, the label selection minimizes the number of selections needed to distinguish all the words in the dictionary. The only way to further reduce this number is to change the alphabet, but this would make the selector harder to use. In order to adopt the last two features, we assumed to store the complete user's dictionary. We do not consider this constraint particularly restrictive, however, we plan to remove it in future works.

We carried out some *in-vivo* and *in-silico* experiments which highlighted both a reduction in the time required to spell a complete sentence and an increased accuracy. Although these tests are limited in number and do not guarantee the

same results in subjects with disabilities, they furnish a cheering picture and push us to further investigate PolyMorph. The spelled source code, which has been released under the GNU GPL license, and all the data obtained during the experiments are available at URL `http://polymorph.units.it`.

In the future, we will test PolyMorph on locked-in subjects and we will remove some of the imposed constraints, for instance, by allowing users to dynamically enrich the vocabulary. Moreover, we would like to integrate in PolyMorph a knowledge graph providing the most likely sentences with respect to the user context. A deduction algorithm that encodes this context-aware mechanism will increase the odd of suggesting the word aimed by the user. Finally, we will target the selection time trying to either reduce or remove the suggestion phase.

## References

1. Sutton, S., Braren, M., Zubin, J., John, E.R.: Evoked-potential correlates of stimulus uncertainty. Science **150**(3700) (Nov 1965) 1187–8
2. Squires, N.K., Squires, K.C., Hillyard, S.A.: Two varieties of long-latency positive waves evoked by unpredictable auditory stimuli in man. Electroencephalogr Clin Neurophysiol **38**(4) (Apr 1975) 387–401
3. Picton, T.W.: The P300 wave of the human event-related potential. J Clin Neurophysiol **9**(4) (1992) 456–79
4. Farwell, L.A., Donchin, E.: Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. Electroencephalogr Clin Neurophysiol **70**(6) (Dec 1988) 510–23
5. Blankertz, B., Dornhege, G., Krauledat, M., Schröder, M., Williamson, J., Murray-Smith, R., Müller, K.R.: The Berlin Brain-Computer Interface presents the novel mental typewriter Hex-o-Spell. In: Proceedings of the 3rd International Brain-Computer Interface Workshop and Training Course, Verlag der Technischen Universität Graz (2006) 108–109
6. Pires, G., Nunes, U., Castelo-Branco, M.: GIBS block speller: toward a gaze-independent P300-based BCI. Conf Proc IEEE Eng Med Biol Soc **2011** (2011) 6360–4
7. Ryan, D.B., Frye, G.E., Townsend, G., Berry, D.R., Mesa-G, S., Gates, N.A., Sellers, E.W.: Predictive spelling with a P300-based brain-computer interface: Increasing the rate of communication. Int J Hum Comput Interact **27**(1) (Jan 2011) 69–84
8. D'Albis, T., Blatt, R., Tedesco, R., Sbattella, L., Matteucci, M.: A predictive speller controlled by a brain-computer interface based on motor imagery. ACM Trans. Comput.-Hum. Interact. **19**(3) (October 2012) 20:1–20:25
9. Chapman, R.M., Bragdon, H.R.: Evoked responses to numerical and non-numerical visual stimuli while problem solving. Nature **203** (1964) 1155–1157
10. Li, Y., Soon Nam, C., Shadden, B.B., Johnson, S.L.: A P300-Based Brain-Computer Interface: Effects of Interface Type and Screen Size. Int. J. Hum. Comput. Interaction **27**(1) (2011) 52–68
11. Schalk, G., McFarland, D.J., Hinterberger, T., Birbaumer, N., Wolpaw, J.R.: BCI2000: a general-purpose brain-computer interface (BCI) system. IEEE Trans. Biomed. Engineering **51**(6) (2004) 1034–1043
12. Morrison, D.R.: PATRICIA - Practical Algorithm To Retrieve Information Coded in Alphanumeric. J. ACM **15**(4) (1968) 514–534