Automatically Extracting P3 Latencies Using a Dynamic Template Matching Algorithm

Sven Lesche[1], Kathrin Sadus[1], Anna-Lena Schubert[2], Christoph Löffler[2], and & Dirk Hagemann[1]

[1] Heidelberg University

[2] University of Mainz

Author Note

The authors acknowledge support by the state of Baden-Württemberg through bwHPC.

Correspondence concerning this article should be addressed to Sven Lesche, Psychologisches Institut, Hauptstraße 47-51, 69117 Heidelberg. E-mail: sven.lesche@psychologie.uni-heidelberg.de

Abstract

In this study, we introduce a template matching algorithm using the grand average as a dynamic template to extract P3 latencies. This new algorithm outperforms peak latency and area latency algorithms in both empirical data and a simulation. Template matching algorithms showed the highest correlation with latencies extracted by expert researchers and the most accurate recovery of simulated latency shifts. Our results highlight the robustness of template matching algorithms across various tasks, preprocessing steps, and algorithm hyperparameters. Additionally, template matching provides a fit statistic that researchers can use to automatically discard ERPs with poor matches or flag certain ERPs for manual review. This template matching algorithm is objective, efficient, reliable and more valid than previous methods. Template matching can be used in the final latency estimation steps of multiverse studies and automated pipelines.

Keywords: event-related potentials, latency extraction, P3, template matching

Automatically Extracting P3 Latencies Using a Dynamic Template Matching Algorithm

Introduction

The latency of event-related potentials (ERPs) is a valuable measure of the speed of neuro-cognitive operations. This measure has been used to investigate the effect of experimental manipulations or individual differences in mental speed and has provided substantial insight into brain processes underlying cognitive functions. For example, a latent factor over individual latencies of the N2, P2, and P3 components in simple decision tasks has been shown to explain 80% of the variance in general intelligence (Schubert et al., 2017). However, testing differences between conditions or measuring individual differences in the latency of components, requires the extraction of latency estimates per subject. Such research based on individual estimates of latency requires extraction methods that provide consistently reliable and valid measures to ensure the replicability of results.

Several approaches to measure the latency of ERP components have been proposed throughout the years. These approaches show considerable variability in reliability and validity. Algorithms aiming to automate latency extraction, such as peak latency algorithms or fractional area latency algorithms, are time-effective and objective, but often lack in reliability or validity (Kiesel et al., 2008; Sadus et al., 2024; Schubert et al., 2023). Manually extracting latencies by inspecting each subject-level signal and directly picking peaks or adjust the measurement window by hand improves reliability and validity (Sadus et al., 2024; Schubert et al., 2023). However, this impairs objectivity and is inefficient, especially in the face of large data. So far, no approach has proven consistently reliable and valid as well as objective and efficient. This paper introduces a novel algorithm for the automatic extraction of P3 latencies based on template matching. We aim to show that this new algorithm improves on existing approaches and enables more efficient, objective, reliable, and valid extraction of ERP latencies.

As a first step, we focus our attention in this paper on the P3. The P3 is a

positive-going deflection around 300 - 500ms after stimulus onset and one of the most widely used ERP components (Donchin, 1981; Luck, 2014). It is related to higher-order cognitive processing and often associated with updating of information (Polich, 2007), stimulus classification (Duncan-Johnson, 1981) or response selection (Polich, 2007). P3 latencies show robust correlations with intelligence (Hilger et al., 2022; Schubert et al., 2023). We chose to restrict our analysis to the P3 and first provide proof-of-concept for this component before extending our work to other components.

Latency Extraction Algorithms

The two approaches most commonly used to automatically extract component latencies are either based on the maximum voltage deflection within a fixed measurement window (peak latency) or the area under the ERP signal [fractional area latency; Luck (2014)]. Peak latency algorithms are based on the assumption that the point in time where maximum voltage deflection occurs is the best estimate of the latency of the true underlying component. However, high voltage deflections are sensitive both to high frequency noise and the superimposed signal of surrounding components (Luck, 2014). This influence of unrelated sources on the maximum voltage deflection challenges the central assumptions of peak latency algorithms (Luck, 2014). Additionally, Kiesel et al. (2008) presented an empirical challenge to peak latency approaches. They showed that the peak latency algorithm was unable to accurately recover simulated latency differences across a range of different components, including the P3. In addition to issues in recovering experimental effects, Sadus et al. (2024) demonstrate considerable issues in the reliability of P3 latencies quantified by the peak latency approach. In their multiverse study with varying pre-processing steps and experimental tasks, the peak latency approach often produced low reliabilities, showed poor homogeneity, and resulted in questionable effect size estimates. Manual extraction of individual P3 latencies proved to yield measures that were more consistently reliable and valid. Taken together, these methodological issues and empirical findings demonstrate that the peak latency algorithm is not a sufficiently reliable

and valid algorithm that could replace manual quantification of ERP latencies.

Fractional area latency algorithms hope to remedy some of the methodological issues associated with the peak latency approach. Here, the area enclosed by the signal and some horizontal axis, most often the time-axis, within a fixed measurement window is used to estimate the latency of a component. Component latency is specified as the time that divides this area into a given fraction. The 50% area latency algorithm, for example, finds the time point that divides the area in two halves. Fractional area latency algorithms are unaffected by high frequency noise (Luck, 2014). Nonetheless, they remain highly sensitive to the measurement window specified by the researcher (Luck, 2014). A wider measurement window may include signal from surrounding components while a shorter measurement window may capture only a part of the signal of the component of interest. Compared to peak latency algorithms, fractional area latency algorithms showed better recovery of simulated latency differences, but also failed to consistently recover simulated experimental effects (Kiesel et al., 2008). Similarly, Sadus et al. (2024) revealed that fractional area latency algorithms were unable to consistently produce good reliabilites, homogeneities, and plausible effect sizes when extracting P3 latencies.

It should be noted that according to Kiesel et al. (2008), the combination of jackknifing and the fractional area latency algorithm leads to the best recovery of simulated experimental effects in P3 latencies. In contrast, Sadus et al. (2024) found that jackknifing may lead to implausible effect sizes. Their results challenge the validity of using jackknifing in large samples. Furthermore, as jackknifing does not produce individual latency estimates, its application to individual differences research is inherently problematic.

Due to the issues of both peak latency algorithms and fractional area latency algorithms, manually extracting component latencies seems to remain the gold-standard regarding reliability and validity when individual differences are of interest (Sadus et al., 2024). All of the measurement pipelines that consistently produced desirable psychometric properties included manual extraction of component latencies (Sadus et al., 2024).

However, manual extraction of latencies suffers from poor objectivity and low efficiency. An extraction algorithm that is sufficiently reliable and valid would greatly improve objectivity and efficiency.

Template Matching

We aim to present an algorithm that can match the reliability and validity of human performance, while improving objectivity and efficiency. With the intention of matching human performance, we sought to embed some of the strategies used by human researchers into the latency extraction algorithm.

Human researchers may be better at extracting ERP latencies because they use more information than traditional extraction algorithms. Before even investigating individual subject ERPs, most researchers inspect the grand averages in order to better understand the underlying ERP structure in their specific task, condition, and subject sample. When extracting the latency of subject-level ERPs, they are looking to identify the part of the signal that most closely resembles the component of interest in the grand average. This might not always be an exact match, amplitudes for the individual might be higher or the component might appear later, but the general shape of the component of interest will still be present. Expert researchers identify the part of the individual signal that best resembles the grand average component and then extract the subject's latency accordingly.

Similarly, our algorithm finds the closest match of the grand average to the individual subject-level ERP. To achieve this, we based our new algorithm on the concept of template matching. We use the grand average over all subjects as a template, which includes an idealized version of the component of interest. Then, the algorithm uses this template to identify the P3 in subject-level ERPs, by figuring out which part of the ERP best reflects the structure in the template. This mirrors the process expert ERP researchers use to extract individual latencies and allows the algorithm to use more of the available information.

Using the grand average as a template is ideal as it contains the average latency and average amplitude of a particular component. Nonetheless, subject-level ERPs differ from this grand average in both latency and amplitude. Quantifying these individual differences is the goal of our algorithm. With an estimate of the latency of the component of interest in the grand average and an estimate of the difference between the subject-level ERP and the grand average, the algorithm can recover individual latencies. To quantify individual differences in relation to the grand average, we use a dynamic template with variable latency and amplitude instead of only using the static grand average. This dynamic template is derived from the grand average by adding two free parameters that allow compressing or stretching both the latency and amplitude of the original grand average. Our algorithm then determines which transformed template shows the best fit to the subject-level ERP. For example, if the template amplitude needs to be stretched by a factor of 1.1 and its latency multiplied by 0.9 to best fit the subject-level ERP, then the subject-level latency is quantified as 0.9 times the grand average's component latency (see Figure 1).

Optimization of these free parameters is based on so called *similarity measures*. The similarity measure quantifies the fit of the transformed template to the subject-level signal. Different approaches to similarity measures are present in the template matching literature and fall into one of two camps: distance-based or correlation-based (Brunelli, 2009; Mahalakshmi et al., 2012). These different approaches may result in different performance of the template matching algorithm (Brunelli & Poggiot, 1997; Goshtasby et al., 1984). We therefore implemented a version of both types of similarity measures. The first is based on minimizing the squared distance (MINSQ) and the second is based on maximizing the correlation between the transformed template and the signal (MAXCOR).

If used without any further modification, all points of the signal would contribute equally to the similarity measure. However, as the component of interest in the grand average only consists of a part of the signal and does not incorporate all points in time, we

tested the use of weighting windows that weigh the similarity measure depending on the time of the component of interest in the grand average. When extracting P3 latencies, which usually occur around 300-500ms (Luck, 2014), early and late activity in the grand average should not influence the template matching procedure as much as the activity central to the P3. Using these weighting windows helps the algorithm to more closely mirror human behavior. Similarly, researchers manually extracting latencies focus on those parts of the signal where the component of interest occurs.

A focus on specific parts of the signal is also present in peak and area latency algorithms. Here it is introduced by the use of measurement windows. Activity outside of these fixed windows is not considered in peak latency or fractional area latency algorithms. Additionally, these measurement windows are applied to each subject-level ERP and directly influence what part of the signal is included. In our template matching algorithm, these windows are based on the grand average and do not directly indicate what part of the subject-level signal is included. Rather, they influence the template that is used during the template matching algorithm. To underline this difference, we will refer to windows set in the context of the template matching algorithm as *weighting windows*.

These weighting windows form the basis of a weighting vector used during the template matching procedure. Generally speaking, the difference (or correlation) between template and signal outside of that window is less important than the difference inside of the window.

Testing

We used the same data used by Sadus et al. (2024) and investigated both the psychometric properties of our new algorithm as well as the correlation with manually extracted latencies. This allows us to evaluate whether our new algorithm is a suitable replacement for the current gold-standard - manual extraction. Additionally, using data from this multiverse study allows us to investigate the impact of different tasks and pre-processing steps on the evaluated algorithms and provide insight into the conditions

under which each algorithm performs best or worst.

We expanded this validation approach by a simulation study based on data used by Schubert et al. (2023) in order to investigate how well a particular algorithm can recover a pre-specified true value. Since no simulation approach exists that can generate realistic ERPs with a known true latency, we simulated shifts in latency between experimental conditions similar to Kiesel et al. (2008). We then evaluated how well peak latency, fractional area latency, and template matching algorithms recover those true simulated latency differences.

We hope to show that a template matching algorithm using the grand average as a variable template can successfully extract subject-level P3 component latencies. Ideally, use of this algorithm improves psychometric properties in comparison to prior algorithms, show high correlations with manually extracted data, accurately recover simulated latency shifts, and present an objective and efficient way to extract ERP latencies.

Template Matching

The template matching algorithm is implemented in MATLAB (Version 2024a)[1]. Broadly, the algorithm can be divided into four steps: generation of the dynamic template, application of similarity weights, optimization, and recovery of latency from free parameters.

Template Generation

We use the grand average ERP over all participants as a template and introduce two free parameters to allow the template to match individual differences in amplitude and latency. Both parameters lead to linear transformations of the template. The first parameter a scales the entire template and adjusts for individual differences in amplitude. The second parameter b stretches or compresses the entire template and adjusts for individual differences in time course.

[1] Additionally, we use the Database Toolbox, the Optimization Toolbox, the Global Optimization Toolbox, the Parallel Computing Toolbox, and the Signal Processing Toolbox.

Implementation of the parameter a is straightforward: the entire template signal is multiplied by a. This results in linear transformations of the template along the amplitude axis. To implement the parameter b that enables linear transformations along the time axis, we simply stretch or compress the signal along the x-axis. We then use spline interpolation to estimate the signal for a given transformed time-point. For example, if the template has a peak signal strength of 9.4 μV at 370 ms and the transformation parameter $b = 1.1$, the transformed template will have the signal strength of 9.4 μV at 370×1.1 ms. The signal strength at 370 ms will be equal to the spline interpolated signal strength of the untransformed template at $\frac{370}{1.1}$ ms (see Figure 2).

The algorithm allows these two free parameters to transform the template along the time- and amplitude-dimension and then optimizes the parameters for maximum similarity between the transformed template and subject-level ERP.

Similarity Measures

We chose to test two different approaches to similarity measures in this paper. The first approach minimizes the (weighted) squared distance between transformed template $t(a, b)$ and the signal s. It will be referred to as the MINSQ approach.

$$\underset{a,b}{\text{argmin}} = \sum_i \omega_i (t_i(a, b) - s_i)^2.$$

The second approach aims to maximize the (weighted) correlation between transformed template $t(a, b)$ and signal s. It will be referred to as the MAXCOR approach.

$$\underset{a,b}{\text{argmax}} = \frac{cov(t(a, b), s; \omega)}{\sqrt{cov(t(a, b), t(a, b); \omega) cov(s, s; \omega)}},$$

where

$$cov(t(a, b), s; \omega) = \frac{\sum_i \omega_i \cdot (t_i(a, b) - m(t(a, b); \omega))(s_i - m(s; \omega))}{\sum_i \omega_i},$$

and

$$m(t(a, b); \omega) = \frac{\sum_i \omega_i t_i(a, b)}{\sum_i \omega_i}.$$

However, since the parameter a linearly scales the template $t_i(a, b)$ it can be rewritten as $at_i(b)$. From this follows:

$$cov(t(a, b), s; \omega) = cov(at(b), s; \omega),$$

and thus

$$\frac{cov(t(a, b), s; \omega)}{\sqrt{cov(t(a, b), t(a, b); \omega)cov(s, s; \omega)}} = \frac{cov(at(b), s; \omega)}{\sqrt{cov(at(b), at(b); \omega)cov(s, s; \omega)}},$$

extracting a out of the covariance terms

$$\frac{cov(at(b), s; \omega)}{\sqrt{cov(at(b), at(b); \omega)cov(s, s; \omega)}} = \frac{acov(t(b), s; \omega)}{\sqrt{a^2cov(t(b), t(b); \omega)cov(s, s; \omega)}},$$

and finally

$$\frac{acov(t(b), s; \omega)}{\sqrt{a^2cov(t(b), t(b); \omega)cov(s, s; \omega)}} = \frac{cov(t(b), s; \omega)}{\sqrt{cov(t(b), t(b); \omega)cov(s, s; \omega)}}.$$

This shows that a does not impact the weighted correlation. We thus only optimized b when using the MAXCOR approach and set $a = 1$.

Weighting Vectors

In order to impart weights on the matching procedure that allow placing more importance on those parts of the signal where the component of interest occurs, we use a weighting vector ω. Weighting windows indicate when the component of interest occurs.

We tested three different weighting windows in order to investigate which size leads to the best template matching results. We implemented the two measurement windows used in Sadus et al. (2024) in order to compare our results to theirs. The first window has a range from 250 ms to 700 ms and the second window has a range from 250 ms to 900 ms. In order to include some of the activity of earlier components, we also tested a weighting window from 200 ms to 700 ms.

Weighting windows are then combined with a weighting function to generate the weighting vector used in the template matching algorithm. A simple function to generate

this weighting vector would be to assign 0 to all values outside of the window and 1 to all values inside of that window. This rectangular window would correspond to a Tukey window with $\alpha = 0$ (Bloomfield, 2004) and is also referred to as a Dirichlet window. The rectangular shape of this weighting window suggests a sudden drop in the relevance of a mismatch just at the weighting window's borders. However, such a sudden cut-off is difficult to justify. Therefore, we tested two versions of weighting functions that include tapered edges.

We tested a Tukey window with $\alpha = 0.25$ and the Hamming window to slowly raise the weights from 0 outside of the measurement window to 1 inside the measurement window (Bloomfield, 2004). We also tested an additional weighting function that assigns weights based on the maximum-normalized amplitude of the grand average. Here the difference between transformed template and signal is of maximum importance at the peak of the component in the grand average and decreases in importance at lower amplitudes. See Figure 3 for an overview of the different weighting functions. Additional information about the weighting functions can be found in the Appendix / supplementary materials.

These weighting windows are defined based on the location of the component of interest in the grand average. In order to simplify the optimization, we chose to rescale the subject-level ERP along the time- and amplitude-dimension and fit it to the static grand average. Therefore, the actual implementation runs exactly opposite to the argument outlined here. This allows us to keep the weighting windows and weighting vectors constant and only transform the subject-level ERP signal. It has no impact on the fit statistic or the recovered latency.

Penalty

Additionally, we observed that the algorithms would sometimes converge on extreme values of the transformation parameter b if no clear solution can be found. To combat this, we also ran all combinations of similarity measures, weighting windows, and weighting functions with a penalty function that penalizes the MAXCOR and MINSQ function for

$b \leq 0.\overline{6}$ or $b \geq 1.5$ by multiplying the value of the function by e^b (or $e^{\frac{1}{b}}$ for $b < 1$).

Optimization

All combinations of similarity measures, weighting windows, weighting functions, and penalty functions optimize the similarity measure using on the GlobalSearch algorithm (Ugray et al., 2007) implemented in the Global Optimization Toolbox (The Math Works, 2024). This optimization algorithm uses a multistart heuristic approach to find the global minimum of a nonlinear differentiable function by evaluating multiple starting points for local minima. We constrained the parameter a with lower and upper bounds of $[0.2, 20]$ and the parameter b with bounds of $[0.3, 2]$[2]. The algorithm returns both the optimal parameters as well as the resulting optimal similarity measure.

Fit

The optimal similarity measure returned by the algorithm can be used as an indicator of certainty. Very close fits of transformed template to subject-level ERP suggest a high degree of certainty and low impact of noise. Poor similarity measures, like correlations $r < 0.3$ suggest that even the optimal transformation of the component in the template does not closely resemble the component structure in the subject-level ERP. Therefore, we used of this fit statistic to remove latency estimates where even the best fit shows poor similarity between template and signal. Because the (weighted) correlation is easier to interpret, we used the correlation as an indicator of fit for both the MAXCOR and MINSQ approach and removed those latency estimates with correlations $r < 0.3$.

[2] For the lower bound of b, we chose a factor of three as this seemed more conservative to us. However, this is not optimal for the upper bound, as the signal is compressed by b in our implementation. This leads to $> 50\%$ missing values in cases where $b > 2$. Therefore, we chose 2 as the upper and 0.3 as the lower boundary. For the parameter a, we chose 0.2 in order to prevent the algorithm to converge on a value close to 0 in cases with inconclusive signal. We added the upper bound of 20 as a very liberal bound on the factor by which the grand average may differ from the individual signal.

Recovery of component latency

We can use the set of optimal transformation parameters to recover the latency of subject-level ERPs. Prior to this, the latency of the component of interest in the grand average needs to be quantified. This can be done either manually or using one of the standard algorithms, like peak latency or fractional area latency. Due to the high signal-to-noise ratio of the grand average, applying these algorithms is much less problematic than in noisy subject-level ERPs. In our analysis, we use the 50% area latency algorithm proposed by Liesefeld (2018) with the respective weighting window as a measurement window to determine the latency in the grand average.

The latency of the component in the grand average l_{GA} is then multiplied by the optimal latency shift parameter of subject j to obtain the latency in the subject-level ERP,

$$l_{GA} \cdot b_j = l_j.$$

Other extraction methods

We defined peak latency as the time of maximum positive voltage deflection within a fixed measurement window which is also larger than the average of the three points both to the right as well as to the left of it (Luck, 2014).

We determined fractional area latency using the 50% area latency approach. All signal below 0 was set to 0. Then we summed up the signal within the measurement window and extracted the two time-points between which the signal grows larger than 50% of the total. The exact latency was then determined through linear interpolation.

We also tested the modifications to fractional area latency suggested by Liesefeld (2018) and subtracted half of the maximum amplitude from the signal as a relative baseline (see also Wascher et al., 2022). Essentially, we set all signal below 50% of the maximum amplitude to 0. All other steps were identical to the steps taken in the regular 50% area latency algorithm.

Study 1

Method

All analyses presented here are based on data originally published in Löffler et al. (2024). The original data contain EEG recordings from 6 different tasks. We focused on the same three tasks that have been already selected by Sadus et al. (2024).

Participants

The empirical evaluation of our algorithm that is presented in the first part of this paper follows Sadus et al. (2024). Their analysis used a sample of 30 young (18-21 years old, $M_{age} = 19.37$, $SD_{age} = 0.76$) and 30 old participants (50-60 years old, $M_{age} = 55.83$, $SD_{age} = 2.87$), representing the 30 youngest and 30 oldest participants from the overall study (Löffler et al., 2024).

All participants had normal or corrected to normal vision. None of the participants had neurological or mental disorders, used psychotropic drugs, wore a pacemaker or suffered from red-green color vision deficiency. All participants provided informed consent prior to participation and received 75€ or course credit for participation.

Tasks

All participants completed a set of three tasks: a Flanker Task, an Nback Task, and a Switching Task. Each assesses one of the executive functions proposed by Miyake et al. (2000). Löffler et al. (2024) programmed all tasks using MATLAB (The Math Works, 2022) and the software package Psychtoolbox (Version 3-0.13) (Brainard & Vision, 1997; Kleiner et al., 2007; Pelli & Vision, 1997). We presented stimuli centrally against a black background and instructed participants to respond as quickly and accurately as possible.

Flanker Task. We employed a standard Arrow Flanker task (Eriksen & Eriksen, 1974) to evaluate a participant's *inhibition* ability. A central arrow was flanked by additional arrows appeared on the screen, with the flanking arrows either pointing in the same or opposite direction to the central arrow. All participants had to identify the direction of the central arrow while ignoring the flanking arrows. Each participant

completed practice trials followed by 100 congruent and 100 incongruent trials.

Nback Task. An adapted Nback task (Scharinger et al., 2015) assessed participants' *updating* abilities. During presentation of a stream of letters, participants had to indicate whether the current letter matched either a predefined target-letter (0-back condition) or the letter presented immediately before (1-back condition). In the original study, participants also completed a 2-back condition. Following Sadus et al. (2024), we excluded this condition from the analysis. All participants completed practice trials and 96 trials per condition.

Switching Task. We administered a Switching task to measure participants' *shifting* ability. We presented colored digits ranging from 1 to 9. Participants had to categorize them based on predefined rules signaled by a colored fixation cross before each trial. They had to either indicate whether the digit was greater than or less than 5 or whether the digit was odd or even. Participants either maintained the same rule as the previous trial or switched to the alternate rule. Each participant completed practice trials followed by 192 trials in both the repeat and switch conditions.

Procedure

The original study consisted of three test sessions. The three tasks analyzed here were all conducted in the first session. The original study included two additional measurement occasions which we will not discuss further here. During EEG measurement, participants were seated approximately 140 cm from a monitor in a sound-attenuated room.

EEG recording and processing

We continuously recorded EEG data using 32 equidistant Ag/AgCl electrodes and took additional electrooculogram (EOG) measures with two electrodes placed above and below the left eye to correct for ocular artifacts. We maintained all impedances below 5 kΩ. We recorded the signal with a sampling rate of 1000 Hz and online-referenced it to Cz. We removed artifacts using ICA on a cloned dataset down-sampled to 200 Hz and filtered with a high-pass filter of 1 Hz. We further removed line-noise using the CleanLine function

(Mullen, 2012). We detected bad channels using a z-value threshold of 3.29 as per the EPOS pipeline (Rodrigues et al., 2021) and interpolated channels that were removed following this procedure. Then, we re-referenced the data using an average referencing scheme. Based on a threshold of 1000 μV and excluding data more than 5 SDs away from the mean, we automatically detected and removed a maximum of 5% of segments per iteration containing artifacts in the ICA-dataset. We used the InfoMax algorithm in the ICA and applied the resulting decomposition to the original dataset. We removed ICs determined to be less than 50% likely to be brain activity by the ICLabel Algorithm (Pion-Tonachini et al., 2019). Then, we low-pass filtered the data using a Butterworth filter with varying cutoff frequencies (4 Hz, 8 Hz, 16 Hz, 32 Hz, and no filter) and segmented it into 1200 ms epochs starting 200 ms prior to stimulus onset. Again, we automatically detected and removed segments containing artifacts. Finally, we baseline corrected segments using the 200 ms prior to stimulus onset. We conducted ERP analyses in MATLAB (Version 2024a) (The Math Works, 2024). We only included correct trials in the analysis and investigated the P3 at the electrode Pz in accordance with existing literature (Polich, 2012; Verleger, 2020).

Latency extraction techniques

We compared all versions of a template matching algorithm resulting from the combinations of similarity measures, weighting windows, weighting functions, and penalty methods to traditional approaches such as peak latency, fractional area latency, and the fractional area latency algorithm proposed by Liesefeld (2018). For peak latency and 50% area latency algorithms, we used the respective measurement windows used by Sadus et al. (2024). See Figure 4 for an overview.

Empirical evaluation

In order to compare our newly proposed algorithm to existing extraction methods, we applied all versions of the template matching algorithm as well as the peak latency algorithm, the 50% area latency algorithm, and the modified area latency algorithm

(Liesefeld, 2018; Wascher et al., 2022). We focused on the reliability of latency values that are extracted by a particular algorithm and the intra-class correlation of latencies extracted by an algorithm vs. an expert ERP researcher (i.e. we used the manually extracted latencies from; (Sadus et al., 2024).

Results

We conducted all data preprocessing and statistical analyses using R [Version 4.3.0; R Core Team (2022)][3].

The grand averages for the three tasks in the empirical evaluation and the data containing the full set of participants used in the simulation are displayed in Figure 5.

Descriptive Statistics

The percentage of missing values varied systematically depending on the extraction algorithm (see Table 1). Across tasks, preprocessing steps, weighting windows, weighting functions, and without a penalty the MINSQ algorithm resulted in 20.00 % missing values and the MAXCOR algorithm in 11.00 % missing values. Penalizing more extreme transformation parameters reduced the percentage of missing values to 5.54 % for the MINSQ algorithm and 3.15 % for the MAXCOR algorithm. The peak latency algorithm resulted in 0.00 % missing values, with 2.17 % and 2.33 % missing values for the area latency and modified area latency algorithm, respectively. For reference, the expert ERP researcher in Sadus et al. (2024) classified 4.05 % of individual ERPs as missing values.

Reliability

The reliability coefficients estimated using split-half correlations are reported in Table 2. Across tasks and preprocessing steps, the 50% area latency algorithm with the window used in Sadus et al. (2024) (250 ms - 700 ms) resulted in the highest average

[3] We, furthermore, used the R-packages *afex* (Version 1.3.1; Singmann et al., 2023), *emmeans* (Version 1.10.2; Lenth, 2023), *flextable* (Version 0.9.6; Gohel & Skintzos, 2023), *knitr* (Version 1.46; Xie, 2015), *papaja* (Version 0.1.2; Aust & Barth, 2022), *rmarkdown* (Version 2.26; Xie et al., 2018, 2020), and *tidyverse* (Version 2.0.0; Wickham et al., 2019).

reliability ($r_{tt} = 0.91$). Across tasks, preprocessing steps, weighting windows, weighting functions, and penalty settings, the MINSQ algorithm showed a mean reliability of $r_{tt} = 0.88$ and the MAXCOR algorithm a mean reliability of $r_{tt} = 0.87$. The best reliability for the template matching algorithms was achieved when using the MAXCOR algorithm with no weights and an exponential penalty ($r_{tt} = 0.89$) or the MINSQ algorithm with normalized weights and an exponential penalty ($r_{tt} = 0.89$). The modified fractional area latency algorithm resulted in an average reliability of $r_{tt} = 0.90$ and the peak latency algorithm in an average reliability of $r_{tt} = 0.78$.

Validity

Intra-class correlations between automatically extracted ERP latencies and those extracted by Sadus et al. (2024) can be found in Table 3. The choice of weighting function had considerable impact on the ICC. Hamming and Tukey weighting functions showed a mean $ICC = 0.80$ across tasks, preprocessing steps, weighting windows, and template matching algorithms. The grand average normalized weighting functions showed a mean $ICC = 0.89$.

Using the MINSQ or MAXCOR algorithm in conjunction with normalized weights yielded mean ICCs of $ICC = 0.89$ and $ICC = 0.90$, respectively. The peak latency algorithm showed a mean $ICC = 0.76$, the 50% area latency algorithm a mean $ICC = 0.77$, and the modified area latency algorithm a mean $ICC = 0.88$.

Discussion

Study 2

Method

All analyses presented here based on data originally published in Schubert et al. (2023). The original data contain EEG recordings from 6 different tasks. The simulation is limited to the task and condition that yielded the grand average with the most typical P3 - the congruent condition of the Flanker task.

Participants

The simulation presented in the second part of this paper is based on the complete set of N = 148 participants (M_{age} = 31.52, SD_{age} = 13.91).

All participants had normal or corrected to normal vision. None of the participants had neurological or mental disorders, used psychotropic drugs, wore a pacemaker or suffered from red-green color vision deficiency. All participants provided informed consent prior to participation and received 75€ or course credit for participation.

Tasks

For detailed information regarding the Flanker task, see the method section of study 1 or Schubert et al. (2023).

Procedure and EEG recording

The procedure was identical to the procedure described in study 1 except that the simulation focuses on only the flanker task.

EEG recording and preprocessing deviated only slightly from the protocol described in study 1. Instead of using the original recording frequency 1000 Hz, we down-sampled the data to 500 Hz. Additionally, we chose not to investigate the data where no low-pass filter was applied. All other preprocessing steps were identical to study 1.

Latency extraction techniques

Again, we compared all versions of a template matching algorithm resulting from the combinations of similarity measures, weighting windows, weighting functions, and penalty methods to traditional approaches such as peak latency, fractional area latency, and the fractional area latency algorithm proposed by Liesefeld (2018). For peak latency and 50% area latency algorithms, we used the respective measurement windows used by Sadus et al. (2024). See Figure 4 for an overview.

Simulation protocol

We chose to deviate from the simulation protocol used by Kiesel et al. (2008) in two ways. Firstly, we refrain from shifting the entire signal by a set distance and rather applied a linear transformation along the time-dimension to the entire signal. Scaling the signal has the benefit that no missing values around the origin are created and that this approach may be more realistic as latency shifts are usually not constant for all components but rather increase in magnitude the later the component occurs (Luck, 2014). Therefore, we chose to stretch the signal by a value λ using the same spline interpolation technique that we employ in the template matching process.

We introduced individual differences in the magnitude of the latency shift by randomly drawing the magnitude of the shift from a normal distribution with a mean of $\mu_\lambda = 0.9$ and a standard deviation of $\sigma_\lambda = 0.05$. We then set this random value as that subject's true latency shift parameter λ_j for all iterations of the simulation and for all latency extraction methods.

At the beginning of each iteration, we randomly divided each subject's trials into control and experimental trials and then averaged them to generate control and experimental ERPs. We then stretched the experimental ERPs by that subject's experimental shift parameter λ_j. We applied all extraction algorithms to the control and experimental ERPs and recorded the recovered component latencies and fit statistics. Finally, we compute the recovered experimental shift of iteration i for person j $\lambda_{i,j}$ by dividing the latency in the control condition $l_{i,j|control}$ by the latency in the experimental condition $l_{i,j|exp}$,

$$\lambda_{i,j} = \frac{l_{i,j|control}}{l_{i,j|exp}}.$$

We repeated this process 250 times for each pre-processing step, each time randomly splitting the available trials into a set of control and experimental trials. We removed latency estimates with bad template matching fit statistics ($r < 0.3$) and unreasonable

estimates of β ($\beta \leq 0.5$ or $\beta \geq 1.9$). Then we averaged $\lambda_{i,j}$ over all simulations to obtain the recovered shift parameter per person $\hat{\lambda_j}$. This reduces the error variance that is introduced by the random split into control and experimental trial. We removed subjects where less than 50% of the simulations yielded a valid latency estimate and excluded subjects whose recovered shift parameters were more than three standard deviations away from the mean. Then, we computed the intra-class correlation focusing on absolute agreement between the true shift parameter λ_j and the average recovered shift parameter $\hat{\lambda_j}$.

Results

Descriptive Statistics

The percentage of missing values per extraction method is reported in Table 4. Across preprocessing steps, weighting windows, weighting functions, and without a penalty, the MINSQ algorithm resulted in 10.39 % missing values and the MAXCOR algorithm in 7.59 % missing values. Employing normalized weights reduced the percentage of missing values to 7.23 % and 4.41 % for the MINSQ and MAXCOR algorithm, respectively. Penalizing more extreme transformation parameters reduced the percentage of missing values to 3.02 % for the MINSQ algorithm and 2.33 % for the MAXCOR algorithm. The peak latency algorithm resulted in 0.16 % missing values. The area latency and modified area latency algorithm resulted in 1.73 % and 1.92 % missing values, respectively.

Validity

Intra-class correlations between true experimental shifts and average recovered shifts can be found in Table 5. The choice of weighting function had considerable impact on the recovery of simulated shifts. Hamming and Tukey weighting functions showed a mean $ICC = 0.77$ across preprocessing steps, weighting windows, and template matching algorithms. The grand average normalized weighting functions showed a mean $ICC = 0.94$.

Using the MINSQ or MAXCOR algorithm in conjunction with normalized weights yielded mean ICCs of $ICC = 0.95$ and $ICC = 0.93$, respectively. The peak latency

algorithm showed a mean $ICC = 0.70$, the 50% area latency algorithm a mean $ICC = 0.78$, and the modified area latency algorithm a mean $ICC = 0.85$.

Discussion

General discussion

Manual extraction of ERP latencies has so far proven more reliable and valid than algorithmic approaches (Sadus et al., 2024) but presents a time- and resource-intensive process. The newly proposed template matching algorithm replicated human behavior and recovered simulated latency shifts better than any of the previous approaches like peak latency or fractional area latency. Template matching algorithms using normalized weights showed a mean intraclass correlation (ICC) with manually extracted P3 latencies of $ICC = 0.89$ across tasks, filter settings, similarity measures, weighting windows, and penalties. This indicates that template matching algorithms are able to replicate manual extraction accurately while presenting a more objective and efficient approach to latency extraction. In addition, template matching algorithms were able to closely recover simulated latency shifts. Template matching algorithms were best at recovering true latency shifts ($ICC = 0.94$) across filter settings, similarity measures, weighting windows, and penalties. Application of template matching algorithms based on the grand average can increase replicability and scalability. Furthermore, it significantly reduces the time and resources spent on latency extraction while proving consistently reliable and valid.

The MINSQ and the MAXCOR approach in combination with normalized weights and a penalty for extreme values worked well across tasks, filter settings and weighting windows. Nonetheless, the exact specification of similarity measures, weighting windows, weighting functions, and penalties depends on a variety of factors.

MAXCOR vs. MINSQ

Both the template matching algorithm maximizing the correlation (MAXCOR) and the algorithm minimizing the squared distance between template and signal (MINSQ) outperformed previous approaches. In both the simulation and the empirical data, the

MAXCOR approach lead to fewer cases where no latency could be specified than the MINSQ approach. We attribute this to cases where few parts of the signal have positive-going amplitude in the P3 window. In this case, the MINSQ approach fails to find a proper match because the optimal transformation would include $a = 0$. We did prevent this by setting a lower bound of 0.2, which will force the optimization to try and find a more reasonable match. However, this issue does prevail in some cases, leading to bad fit statistics and thus resulting in missing values. The MAXCOR approach does not suffer from this issue because the value of the amplitude is irrelevant to this approach.

The MAXCOR approach slightly outperformed the MINSQ approach in replicating an expert ERP researcher. On the other hand, the MINSQ approach displayed higher reliability in the empirical data and showed better recovery in the simulation. Importantly, even expert ERP researchers are not perfect when extracting ERP latencies. Therefore, we place more emphasis on the results of the simulation, where the MINSQ approach gained the upper hand. In practice, both approaches performed very well and applying them will lead to improved quality in the extracted latencies. The tradeoff between a higher amount of missing values or slightly lower validity should be tailored to the specific requirements of the research project. In cases where large amounts of data are available or analysis methods are less affected by missing values, the MINSQ approach seems to prove best.

Finally, **a param not mit MAXCOR here**

Weighting functions

We tested four different weighting functions. Not applying any weighting function did perform well in the simulation but perfomed poorly in empirical data. The Hamming and Tukey weighting functions performed poorly in both the empirical data and the simulation. Using the maximum-normalized amplitude in the template to generate weights lead to the best results across the board. While the exact shape of the weighting function is open to discussion, these results support the notion that a weighting function that is based on the amplitude in the template itself will lead to the best results. We argue that

these normalized weights are also the closest formalization of the process a human researcher employs when identifying ERP component latencies. The shape around the peak of the component is most important and the shape around the onset and offset of the components are less important. If there is a pronounced surrounding component, like the N2 in our case, researchers may use this as an additional indicator. The normalized weighting function best captures this behaviour.

Weighting windows

We investigated three different weighting windows in order to gauge the impact this has on our template matching algorithms. For both approaches and across tasks and filter settings, the weighting window did not impact the validity greatly. In some cases, the wide window (250 - 900ms) seems to lead to slightly worse validity than the other windows. We therefore recommend specifying a weighting window in a similar manner to a measurement window for an area-based latency algorithm. Any window that includes a part of the onset, the peak and a part of the offset should perform reasonably well.

This invariance to the weighting window is an additional benefit of template matching approaches. In contrast to traditional algorithms, the weighting window is not applied to each subject level ERP. Measurement windows, as used in traditional algorithms, may not include the component of interest fully in a given subject-level ERP. Often, it is then the expert's job to readjust this measurement window for each individual. In the template matching algorithms, the weighting window is only applied to the template and then used to generate the weighing vector. Therefore, it will always include exactly those parts of the component that were previously specified by the researcher.

Penalty

Inclusion of a penalty term for extreme values of b during the optimization reduced the percentage of missing values greatly. We suspect that this is because extreme values are often the result of spurious high matches with very late or very early parts of the signal. This can occur in cases where the signal to noise ratio of a subject's ERP is low.

Forcing the algorithm to converge on a more typical value thus leads to more reasonable latency estimates that also still generate acceptable fit statistics. Importantly, this is a penalty and not an upper/lower bound. The algorithm can converge on extreme values in cases where the latency of the component in the subject does deviate from the template greatly and less extreme values of b do not lead to significantly better similarity measures. Additionally, no penalty is being applied as long as $0.\overline{6} \leq b \leq 1.5$. Accordingly, no bias towards the mean is present during the optimization process for most ERPs.

A bias towards the mean would manifest in a reduction in validity. We did observe a reduction in validity in the simulation, but only for the MINSQ approach. In the empirical evaluation, the MINSQ approach matched the expert more closely when a penalty was used. The MAXCOR approach had higher validity in both the simulation and the empirical evaluation. Therefore, we see no signs of a reduction in validity that a bias towards the mean would entail.

In practice, whether to apply a penalty should be determined by the degree of automation needed and the amount of missing values that are tolerable. If there is no time to review cases with extreme values of b and missing values have to be avoided, a penalty term should be applied during optimization. This ensures a lower number of missing values without reducing validity. In order to ensure optimal validity, we suggest running the algorithm without a penalty term and then manually reviewing those cases with the worst fit statistics and the most extreme values of b.

Fit statistic

Template matching algorithms allow researchers to selectively review those subject-level ERPs where the correlation between template and signal indicates low similarity. For example, researchers can use the algorithm to efficiently and objectively extract the majority of ERP latencies and only spend time on those cases with the worst fit statistics. Here, a human may be able to quantify some ERPs that the algorithm fails on, thus increasing the amount of usable data. By adjusting the cutoff after which manual

review is necessary, researchers can be as liberal or conservative as they choose. The algorithm performs well without a review, but the validity of the extracted data will probably increase after human inspection.

No other algorithms allow this type of efficient manual intervention or adjustable automatic rejection of the worst cases. In traditional approaches, subject-level metrics of data quality, like the standardized measurement error (SME) (Luck et al., 2021) may be used to exclude those subjects with insufficient quality metrics (Wascher et al., 2022). This does lead to improved results and may be considered as an alternative. However, the SME is directly influence by the number of trials (Zhang & Luck, 2023) and the number of data-points (Rodrigues et al., 2024). The fit statistic generated by template matching approaches is unaffected by both of them. Furthermore, the correlation between template and signal provides a more intuitive quantification of data quality. It directly reflects the extent to which the component of interest is present in the data and how much it is corrupted by noise.

Importantly, noise in the sense of correlation between template and signal is any unsystematic difference in the covariance term between the signal and the transformed template. Trial-to-trial variability, which may be an important part in the neurocognitive processes underlying the task, will not influence the fit statistic as long as this variability occurs in most subjects. While the SME would classify this as noise, the fit statistic of template matching algorithm allows for intertrial variability. For example, variability in the true latency of the component will lead to broader components in both the subject-level ERPs as well as the template.

Limitations and future research

Most importantly, this work is limited in scope. As a first step, we used template matching algorithms to extract P3 latencies only. In order to improve generalizability, we included three different tasks and varied the low-pass filter during preprocessing. While the different template matching approaches differed in validity between the tasks and filter

settings, we did not observe severe issues in any of these 15 different conditions. We are confident that template matching algorithms can be used to extract P3 latencies in other tasks or using other preprocessing pipelines. Nonetheless, we did not investigate any other components and will remedy this shortcoming in future work.

Secondly, application of the template matching algorithm depends heavily on the template itself. If there is no clear component structure in the grand average, a template matching algorithm will not be of any help. However, we suspect that other, more problematic issues are at hand in this case.

Additionally, application of a template matching algorithm is more complex and computationally demanding than the previous algorithms. We want to address the complexity issue in future work and simplify the application of template matching algorithms to make them more accessible. Quantification of a single ERP takes around 0.05-0.2 seconds depending on the hardware. For extremely large data and low computing power, using the fractional area latency algorithm with the modifications proposed by Liesefeld (2018) may be the better choice. However, with access to a high performance computing cluster running 25 nodes in parallel, we were able to run the quantification of 4.8 million ERPs in under 24 hours.

Lastly, we used an arbitrary cutoff for the fit statistic that was based on our previous experience in working with the algorithm. As with any cutoff, the exact value is up to debate and can be specified by the researcher on a study-by-study basis.

Conclusion

Template matching algorithms using the grand average as a dynamic template were able to extract P3 latencies better than any previous algorithm both in empirical data as well as in our simulation. Template matching algorithms show the best correlation with latencies extracted by an expert researcher and the best recovery of true simulated latency shifts. Both the MAXCOR and MINSQ approach in combination with normalized weights work well across tasks, filter settings, weighting windows, and penalties. The algorithms

also provide a fit statistic that may be used to automatically discard those ERPs with low matches to the template. It may also be used to flag certain ERPs for later manual review.

The exact specification of which similarity measure to use, which weighting function to apply, whether to apply a penalty, and which cutoff to use for the fit statistic is up to each individual researcher. We recommend using the MINSQ approach with normalized weights and a weighting window that includes the on- and offset of the component. If no manual inspection is possible, we recommend using a penalty for extreme values of b during the optimization process. Otherwise, we recommend using the unpenalized MINSQ approach and manually reviewing only cases with the worst fits and most extreme values of b.

We showed that this template matching algorithm improves objectivity and efficiency while maintaining consistently good reliability and superior validity over previous approaches. This algorithm could be used as a final step for latency estimation in multiverse studies (Sadus et al., 2024; Zhang & Luck, 2023) and other automated pipelines (Clayson et al., 2021; Rodrigues et al., 2021; Wascher et al., 2022). We hope to demonstrate the efficacy of template matching algorithms for other components in future work.

References

Aust, F., & Barth, M. (2022). *papaja: Prepare reproducible APA journal articles with R Markdown*. https://github.com/crsh/papaja

Bloomfield, P. (2004). *Fourier analysis of time series: An introduction*. John Wiley & Sons.

Brainard, D. H., & Vision, S. (1997). The psychophysics toolbox. *Spatial Vision*, *10*(4), 433–436.

Brunelli, R. (2009). *Template matching techniques in computer vision: Theory and practice*. John Wiley & Sons.

Brunelli, R., & Poggiot, T. (1997). Template matching: Matched spatial filters and beyond. *Pattern Recognition*, *30*(5), 751–768.

Clayson, P. E., Brush, C. J., & Hajcak, G. (2021). Data quality and reliability metrics for event-related potentials (ERPs): The utility of subject-level reliability. *International Journal of Psychophysiology*, *165*, 121–136.

Donchin, E. (1981). Surprise!... surprise? *Psychophysiology*, *18*(5), 493–513.

Duncan-Johnson, C. C. (1981). Young psychophysiologist award address, 1980: P300 latency: A new metric of information processing. *Psychophysiology*, *18*(3), 207–215.

Eriksen, B. A., & Eriksen, C. W. (1974). Effects of noise letters upon the identification of a target letter in a nonsearch task. *Perception & Psychophysics*, *16*(1), 143–149. https://doi.org/10.3758/BF03203267

Gohel, D., & Skintzos, P. (2023). *Flextable: Functions for tabular reporting*. https://CRAN.R-project.org/package=flextable

Goshtasby, A., Gage, S. H., & Bartholic, J. F. (1984). A two-stage cross correlation approach to template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *3*, 374–378.

Hilger, K., Spinath, F. M., Troche, S., & Schubert, A.-L. (2022). The biological basis of intelligence: Benchmark findings. *Intelligence*, *93*, 101665.

Kiesel, A., Miller, J., Jolicœur, P., & Brisson, B. (2008). Measurement of ERP latency

differences: A comparison of single-participant and jackknife-based scoring methods. *Psychophysiology*, *45*(2), 250–274.

Kleiner, M., Brainard, D., & Pelli, D. (2007). *What's new in psychtoolbox-3?*

Lenth, R. V. (2023). *Emmeans: Estimated marginal means, aka least-squares means.* https://CRAN.R-project.org/package=emmeans

Liesefeld, H. R. (2018). Estimating the timing of cognitive operations with MEG/EEG latency measures: A primer, a brief tutorial, and an implementation of various methods. *Frontiers in Neuroscience*, *12*, 765.

Löffler, C., Frischkorn, G. T., Hagemann, D., Sadus, K., & Schubert, A.-L. (2024). The common factor of executive functions measures nothing but speed of information uptake. *Psychological Research*, 1–23.

Luck, S. J. (2014). *An introduction to the event-related potential technique.* MIT press.

Luck, S. J., Stewart, A. X., Simmons, A. M., & Rhemtulla, M. (2021). Standardized measurement error: A universal metric of data quality for averaged event-related potentials. *Psychophysiology*, *58*(6), e13793. https://doi.org/10.1111/psyp.13793

Mahalakshmi, T., Muthaiah, R., & Swaminathan, P. (2012). Image processing. *Research Journal of Applied Sciences, Engineering and Technology*, *4*(24), 5469–5473.

Miyake, A., Friedman, N. P., Emerson, M. J., Witzki, A. H., Howerter, A., & Wager, T. D. (2000). The unity and diversity of executive functions and their contributions to complex "frontal lobe" tasks: A latent variable analysis. *Cognitive Psychology*, *41*(1), 49–100.

Mullen, T. (2012). CleanLine EEGLAB plugin. *San Diego, CA: Neuroimaging Informatics Toolsand Resources Clearinghouse (NITRC).*

Pelli, D. G., & Vision, S. (1997). The VideoToolbox software for visual psychophysics: Transforming numbers into movies. *Spatial Vision*, *10*, 437–442.

Pion-Tonachini, L., Kreutz-Delgado, K., & Makeig, S. (2019). ICLabel: An automated electroencephalographic independent component classifier, dataset, and website.

NeuroImage, *198*, 181–197.

Polich, J. (2007). Updating P300: An integrative theory of P3a and P3b. *Clinical Neurophysiology*, *118*(10), 2128–2148.

Polich, J. (2012). Neuropsychology of P300. *The Oxford Handbook of Event-Related Potential Components*, 159–188.

R Core Team. (2022). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Rodrigues, J., Müller, S., Paelecke, M., Wang, Y., & Hewig, J. (2024). Exploration of the influence of the quantification method and reference scheme on feedback-related negativity and standardized measurement error of feedback-related negativity amplitudes in a trust game. *Cortex; a Journal Devoted to the Study of the Nervous System and Behavior*.

Rodrigues, J., Weiß, M., Hewig, J., & Allen, J. J. (2021). EPOS: EEG processing open-source scripts. *Frontiers in Neuroscience*, *15*, 660449.

Sadus, K., Schubert, A.-L., Löffler, C., & Hagemann, D. (2024). An explorative multiverse study for extracting differences in P3 latencies between young and old adults. *Psychophysiology*, *61*(2), e14459.

Scharinger, C., Soutschek, A., Schubert, T., & Gerjets, P. (2015). When flanker meets the n-back: What EEG and pupil dilation data reveal about the interplay between the two central-executive working memory functions inhibition and updating. *Psychophysiology*, *52*(10), 1293–1304. https://doi.org/10.1111/psyp.12500

Schubert, A.-L., Hagemann, D., & Frischkorn, G. T. (2017). Is general intelligence little more than the speed of higher-order processing? *Journal of Experimental Psychology: General*, *146*(10), 1498–1512. https://doi.org/10.1037/xge0000325.supp

Schubert, A.-L., Löffler, C., Hagemann, D., & Sadus, K. (2023). How robust is the relationship between neural processing speed and cognitive abilities? *Psychophysiology*, *60*(2), e14165.

Singmann, H., Bolker, B., Westfall, J., Aust, F., & Ben-Shachar, M. S. (2023). *Afex: Analysis of factorial experiments*. https://CRAN.R-project.org/package=afex

The Math Works, Inc. (2022). *MATLAB version: 9.13.0 (r2022b)*. The MathWorks Inc. https://www.mathworks.com

The Math Works, Inc. (2024). *MATLAB version 24.1.0.2537033 (R2024a)*. The Mathworks, Inc. https://www.mathworks.com

Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., & Martí, R. (2007). Scatter search and local NLP solvers: A multistart framework for global optimization. *INFORMS Journal on Computing*, *19*(3), 328–340.

Verleger, R. (2020). Effects of relevance and response frequency on P3b amplitudes: Review of findings and comparison of hypotheses about the process reflected by P3b. *Psychophysiology*, *57*(7), e13542.

Wascher, E., Sharifian, F., Gutberlet, M., Schneider, D., Getzmann, S., & Arnau, S. (2022). Mental chronometry in big noisy data. *PLOS ONE*, *17*(6), e0268916. https://doi.org/10.1371/journal.pone.0268916

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Chapman; Hall/CRC. https://yihui.org/knitr/

Xie, Y., Allaire, J. J., & Grolemund, G. (2018). *R markdown: The definitive guide*. Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown

Xie, Y., Dervieux, C., & Riederer, E. (2020). *R markdown cookbook*. Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook

Zhang, G., & Luck, S. J. (2023). Variations in ERP data quality across paradigms,

participants, and scoring procedures. *Psychophysiology*, *60*(7), e14264.

Table 1

Missing values for different algorithms: empirical evaluation

		[200 700]		[250 700]		
approach	weight	penalized	none	penalized	none	penalized
maxcor	none	0.05	0.08	0.05	0.08	0.05
	hamming	0.03	0.16	0.02	0.17	0.04
	tukey	0.03	0.15	0.03	0.15	0.04
	normalized	0.01	0.10	0.01	0.10	0.01
minsq	none	0.06	0.21	0.06	0.21	0.06
	hamming	0.05	0.19	0.04	0.19	0.09
	tukey	0.07	0.22	0.06	0.21	0.09
	normalized	0.02	0.15	0.03	0.15	0.03
peak	none		0.00		0.00	
area	none		0.02		0.02	
liesefeld	none		0.02		0.02	

Note. Frequency of missing values per algorithm. The rows indicate combinations of similarity measure and weigh

Table 2

Reliability for different algorithms: empirical evaluation

		[200 700]		[250 700]		
approach	weight	penalized	none	penalized	none	penalized
maxcor	none	0.89	0.88	0.89	0.88	0.89
	hamming	0.88	0.91	0.86	0.86	0.85
	tukey	0.90	0.90	0.87	0.87	0.86
	normalized	0.85	0.87	0.85	0.88	0.85
minsq	none	0.88	0.89	0.88	0.89	0.88
	hamming	0.88	0.89	0.89	0.85	0.86
	tukey	0.88	0.88	0.89	0.89	0.86
	normalized	0.89	0.88	0.89	0.88	0.89
peak	none		0.81		0.83	
area	none		0.91		0.91	
liesefeld	none		0.90		0.90	

Note. Reliability estimated by the split-half correlation. The rows indicate combinations of similarity measure and

Table 3

ICC for different algorithms: empirical evaluation

		[200 700]		[250 700]		
approach	weight	penalized	none	penalized	none	penalized
maxcor	none	0.81	0.78	0.81	0.78	0.81
	hamming	0.87	0.83	0.84	0.79	0.80
	tukey	0.87	0.82	0.81	0.76	0.80
	normalized	0.90	0.90	0.91	0.90	0.90
minsq	none	0.74	0.74	0.74	0.74	0.74
	hamming	0.84	0.82	0.83	0.78	0.68
	tukey	0.83	0.81	0.81	0.79	0.73
	normalized	0.89	0.89	0.90	0.89	0.89
peak	none		0.81		0.83	
area	none		0.79		0.77	
liesefeld	none		0.86		0.88	

Note. Intra-class correlation focusing on absolute agreement. The rows indicate combinations of similarity measur

Table 4

Missing values for different algorithms: simulation

		[200 700]		[250 700]		
approach	weight	penalized	none	penalized	none	penalized
maxcor	none	0.03	0.06	0.03	0.06	0.03
	hamming	0.02	0.09	0.02	0.13	0.03
	tukey	0.02	0.09	0.03	0.12	0.04
	normalized	0.00	0.04	0.00	0.04	0.00
minsq	none	0.03	0.11	0.03	0.11	0.03
	hamming	0.02	0.09	0.03	0.10	0.05
	tukey	0.03	0.10	0.03	0.11	0.06
	normalized	0.01	0.07	0.01	0.07	0.01
peak	none		0.00		0.00	
area	none		0.02		0.02	
liesefeld	none		0.02		0.02	

Note. Frequency of missing values per algorithm. The rows indicate combinations of similarity measure and weigh

Table 5

ICC between true and recovered experimental shift for different algorithms: simulation

		[200 700]		[250 700]		
approach	weight	penalized	none	penalized	none	penalized
maxcor	none	0.95	0.95	0.94	0.94	0.93
	hamming	0.78	0.66	0.76	0.77	0.81
	tukey	0.60	0.64	0.73	0.78	0.86
	normalized	0.90	0.93	0.96	0.94	0.92
minsq	none	0.93	0.96	0.92	0.95	0.93
	hamming	0.79	0.74	0.73	0.71	0.77
	tukey	0.81	0.81	0.77	0.75	0.83
	normalized	0.94	0.94	0.95	0.96	0.94
peak	none		0.84		0.81	
area	none		0.81		0.78	
liesefeld	none		0.89		0.85	

Note. Intra-class correlation focusing on absolute agreement. The rows indicate combinations of similarity measur

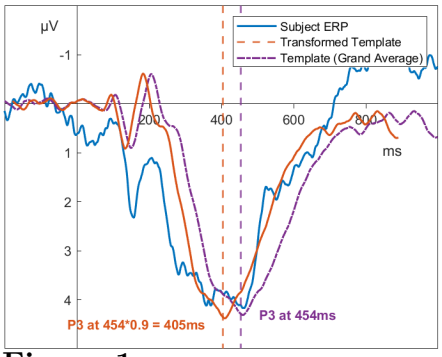

Figure 1

Transforming the template to match an ERP

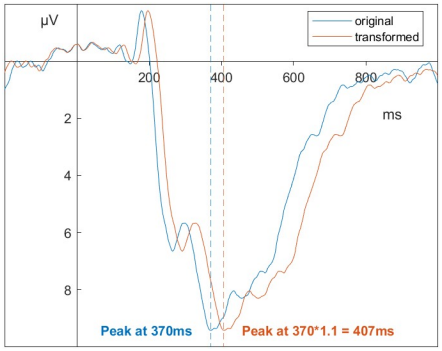

Figure 2

Transforming the latency of a signal using b

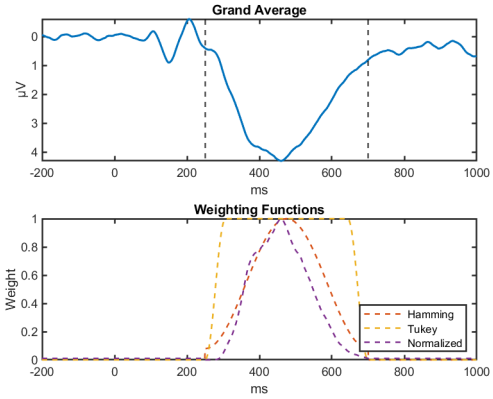

Figure 3

Overview of weighting functions

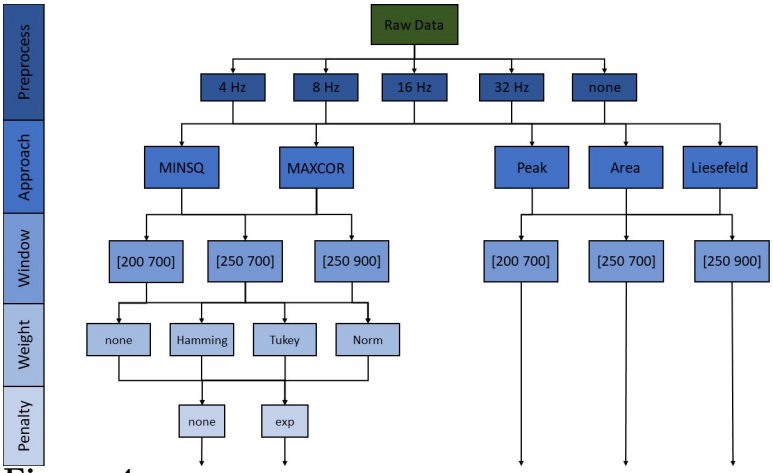

Figure 4

Overview of extraction algorithms

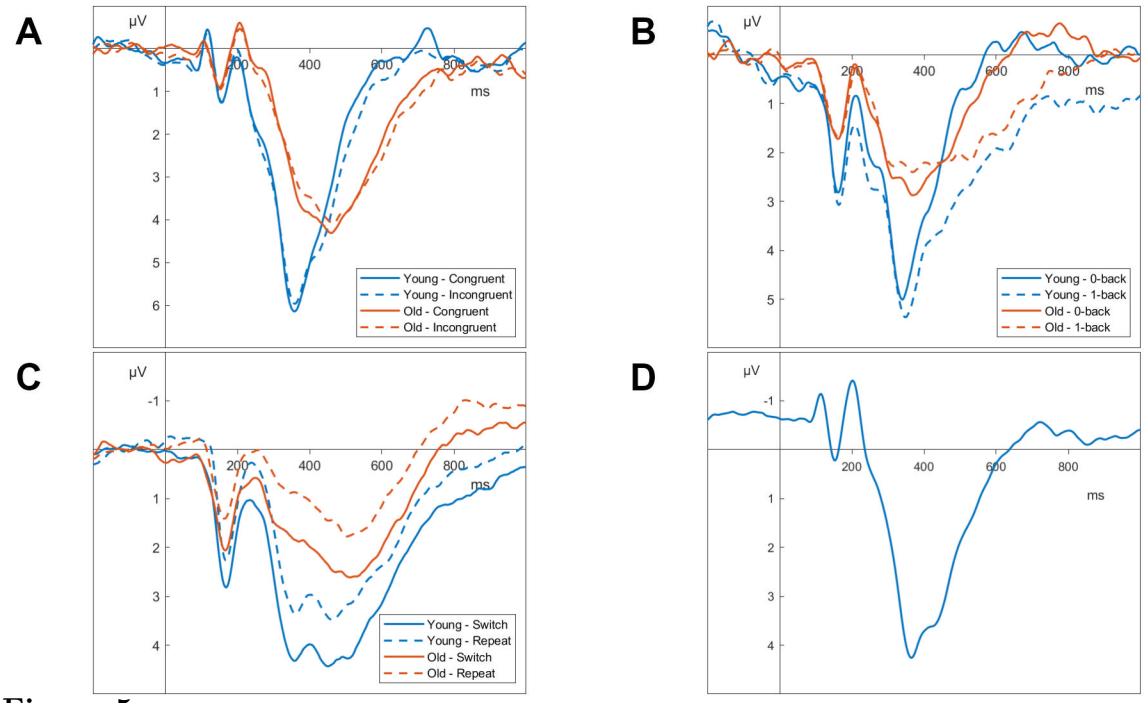

Figure 5

Grand Averages of the Tasks used at Pz with a 32 Hz low-pass filter

Appendix A

Empirical data

Missing values

Reliability

Validity

Appendix B

Simulation

Table B1

Missing values of different algorithms by task

			[200 700]		[250 700]		
approach	weight	task	penalized	none	penalized	none	penalized
maxcor	none	flanker	0.02	0.04	0.02	0.04	0.02
	none	nback	0.07	0.11	0.07	0.11	0.07
	none	switching	0.06	0.09	0.06	0.09	0.06
	hamming	flanker	0.00	0.07	0.02	0.13	0.04
	hamming	nback	0.04	0.12	0.04	0.11	0.06
	hamming	switching	0.03	0.28	0.01	0.26	0.01
	tukey	flanker	0.00	0.06	0.02	0.10	0.03
	tukey	nback	0.05	0.14	0.04	0.10	0.06
	tukey	switching	0.05	0.26	0.02	0.24	0.02
	normalized	flanker	0.00	0.03	0.00	0.03	0.00
	normalized	nback	0.02	0.11	0.03	0.11	0.02
	normalized	switching	0.01	0.15	0.01	0.15	0.00
	none	flanker	0.02	0.12	0.02	0.12	0.02
	none	nback	0.07	0.21	0.07	0.21	0.07
	none	switching	0.09	0.28	0.09	0.28	0.09
	hamming	flanker	0.01	0.07	0.02	0.08	0.06
	hamming	nback	0.08	0.22	0.08	0.24	0.12
	hamming	switching	0.05	0.29	0.03	0.26	0.08
	tukey	flanker	0.02	0.09	0.03	0.10	0.06
	tukey	nback	0.10	0.22	0.09	0.24	0.10
	tukey	switching	0.10	0.35	0.07	0.29	0.09
	normalized	flanker	0.00	0.05	0.00	0.06	0.00
	normalized	nback	0.05	0.19	0.06	0.18	0.06

minsq

Table B1

Missing values of different algorithms by task

			[200 700]		[250 700]		
approach	weight	task	penalized	none	penalized	none	penalized
	normalized	switching	0.02	0.21	0.02	0.21	0.02
	none	flanker		0.00		0.00	
peak	none	nback		0.00		0.00	
	none	switching		0.00		0.00	
	none	flanker		0.02		0.02	
area	none	nback		0.01		0.01	
	none	switching		0.03		0.04	
	none	flanker		0.02		0.02	
liesefeld	none	nback		0.01		0.01	
	none	switching		0.03		0.04	

Note. Frequency of missing values per algorithm. The rows indicate combinations of similarity measure and weigh

Table B2

Missing values of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
maxcor	none	0	0.05	0.07	0.05	0.07	0.05
	none	4	0.06	0.09	0.06	0.09	0.06
	none	8	0.05	0.08	0.05	0.08	0.05
	none	16	0.05	0.08	0.05	0.08	0.05
	none	32	0.05	0.08	0.05	0.08	0.05
	hamming	0	0.02	0.15	0.02	0.17	0.03
	hamming	4	0.03	0.12	0.02	0.12	0.04
	hamming	8	0.03	0.16	0.03	0.19	0.04
	hamming	16	0.02	0.18	0.03	0.19	0.03
	hamming	32	0.02	0.16	0.02	0.17	0.03
	tukey	0	0.04	0.15	0.03	0.15	0.04
	tukey	4	0.04	0.12	0.03	0.12	0.04
	tukey	8	0.04	0.17	0.03	0.16	0.04
	tukey	16	0.03	0.16	0.03	0.16	0.04
	tukey	32	0.02	0.16	0.03	0.15	0.04
	normalized	0	0.01	0.11	0.01	0.11	0.01
	normalized	4	0.02	0.07	0.02	0.06	0.02
	normalized	8	0.02	0.09	0.01	0.09	0.01
	normalized	16	0.01	0.10	0.01	0.11	0.01
	normalized	32	0.01	0.11	0.01	0.11	0.01
	none	0	0.06	0.22	0.06	0.22	0.06
	none	4	0.06	0.20	0.06	0.20	0.06
	none	8	0.06	0.20	0.06	0.20	0.06

Table B2

Missing values of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
minsq	none	16	0.06	0.22	0.06	0.22	0.06
	none	32	0.06	0.20	0.06	0.20	0.06
	hamming	0	0.04	0.16	0.04	0.17	0.09
	hamming	4	0.05	0.23	0.04	0.21	0.09
	hamming	8	0.04	0.21	0.04	0.21	0.08
	hamming	16	0.05	0.19	0.04	0.20	0.09
	hamming	32	0.05	0.17	0.04	0.17	0.08
	tukey	0	0.08	0.21	0.07	0.19	0.09
	tukey	4	0.07	0.22	0.06	0.21	0.10
	tukey	8	0.08	0.25	0.06	0.23	0.09
	tukey	16	0.07	0.22	0.07	0.22	0.08
	tukey	32	0.07	0.20	0.06	0.19	0.07
	normalized	0	0.02	0.15	0.02	0.14	0.02
	normalized	4	0.03	0.15	0.04	0.15	0.04
	normalized	8	0.02	0.16	0.02	0.16	0.03
	normalized	16	0.02	0.15	0.02	0.15	0.02
	normalized	32	0.02	0.15	0.02	0.16	0.02
peak	none	0		0.00		0.00	
	none	4		0.00		0.00	
	none	8		0.00		0.00	
	none	16		0.00		0.00	
	none	32		0.00		0.00	
	none	0		0.02		0.02	

Table B2

Missing values of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
area	none	4		0.03		0.04	
	none	8		0.02		0.02	
	none	16		0.02		0.02	
	none	32		0.02		0.02	
liesefeld	none	0		0.02		0.02	
	none	4		0.04		0.04	
	none	8		0.02		0.02	
	none	16		0.02		0.02	
	none	32		0.02		0.02	

Note. Frequency of missing values per algorithm. The rows indicate combinations of similarity measure and weigh

Table B3

Reliability of different algorithms by task

			[200 700]		[250 700]		
approach	weight	task	penalized	none	penalized	none	penalized
maxcor	none	flanker	0.93	0.91	0.93	0.91	0.93
	none	nback	0.87	0.87	0.87	0.87	0.87
	none	switching	0.87	0.85	0.87	0.85	0.87
	hamming	flanker	0.92	0.94	0.95	0.94	0.92
	hamming	nback	0.85	0.88	0.80	0.84	0.84
	hamming	switching	0.87	0.92	0.83	0.81	0.78
	tukey	flanker	0.95	0.94	0.91	0.92	0.94
	tukey	nback	0.87	0.85	0.85	0.83	0.82
	tukey	switching	0.89	0.89	0.84	0.86	0.82
	normalized	flanker	0.88	0.90	0.87	0.90	0.84
	normalized	nback	0.81	0.83	0.82	0.85	0.83
	normalized	switching	0.87	0.88	0.87	0.87	0.89
	none	flanker	0.92	0.94	0.92	0.94	0.92
	none	nback	0.88	0.90	0.88	0.90	0.88
	none	switching	0.86	0.83	0.86	0.83	0.86
	hamming	flanker	0.92	0.95	0.90	0.88	0.87
	hamming	nback	0.85	0.87	0.91	0.87	0.89
	hamming	switching	0.86	0.84	0.87	0.79	0.82
	tukey	flanker	0.91	0.93	0.92	0.92	0.88
	tukey	nback	0.87	0.86	0.89	0.89	0.88
	tukey	switching	0.86	0.87	0.85	0.86	0.83
	normalized	flanker	0.91	0.90	0.91	0.91	0.91
	normalized	nback	0.87	0.89	0.85	0.88	0.87

Table B3

Reliability of different algorithms by task

			[200 700]		[250 700]		
approach	weight	task	penalized	none	penalized	none	penalized
	normalized	switching	0.89	0.85	0.90	0.85	0.90
	none	flanker		0.82		0.86	
peak	none	nback		0.77		0.77	
	none	switching		0.85		0.86	
	none	flanker		0.93		0.93	
area	none	nback		0.89		0.90	
	none	switching		0.90		0.89	
	none	flanker		0.92		0.94	
liesefeld	none	nback		0.87		0.87	
	none	switching		0.91		0.91	

Note. Reliability estimated by the split-half correlation. The rows indicate combinations of similarity measure and

Table B4

Reliability of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
maxcor	none	0	0.88	0.86	0.88	0.86	0.88
	none	4	0.90	0.89	0.90	0.89	0.90
	none	8	0.90	0.89	0.90	0.89	0.90
	none	16	0.89	0.87	0.89	0.87	0.89
	none	32	0.88	0.86	0.88	0.86	0.88
	hamming	0	0.84	0.89	0.85	0.84	0.85
	hamming	4	0.91	0.90	0.85	0.87	0.86
	hamming	8	0.90	0.93	0.87	0.88	0.83
	hamming	16	0.90	0.91	0.86	0.83	0.84
	hamming	32	0.87	0.92	0.87	0.89	0.86
	tukey	0	0.90	0.85	0.80	0.81	0.88
	tukey	4	0.92	0.91	0.88	0.90	0.84
	tukey	8	0.90	0.91	0.89	0.90	0.84
	tukey	16	0.89	0.92	0.88	0.86	0.85
	tukey	32	0.90	0.89	0.88	0.88	0.88
	normalized	0	0.84	0.82	0.85	0.83	0.76
	normalized	4	0.89	0.91	0.90	0.90	0.89
	normalized	8	0.88	0.88	0.88	0.89	0.90
	normalized	16	0.89	0.88	0.85	0.87	0.90
	normalized	32	0.77	0.86	0.78	0.88	0.81
	none	0	0.86	0.86	0.86	0.86	0.86
	none	4	0.89	0.90	0.89	0.90	0.89
	none	8	0.89	0.91	0.89	0.91	0.89

Table B4

Reliability of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
minsq	none	16	0.89	0.91	0.89	0.91	0.89
	none	32	0.89	0.88	0.89	0.88	0.89
	hamming	0	0.90	0.89	0.91	0.86	0.84
	hamming	4	0.82	0.86	0.84	0.82	0.86
	hamming	8	0.92	0.93	0.90	0.87	0.86
	hamming	16	0.87	0.90	0.90	0.82	0.87
	hamming	32	0.87	0.86	0.92	0.88	0.87
	tukey	0	0.90	0.89	0.90	0.90	0.87
	tukey	4	0.83	0.87	0.84	0.86	0.88
	tukey	8	0.90	0.87	0.90	0.90	0.86
	tukey	16	0.88	0.91	0.90	0.92	0.87
	tukey	32	0.89	0.87	0.90	0.88	0.83
	normalized	0	0.87	0.84	0.84	0.86	0.87
	normalized	4	0.91	0.90	0.91	0.88	0.89
	normalized	8	0.91	0.90	0.92	0.92	0.91
	normalized	16	0.89	0.88	0.89	0.88	0.91
	normalized	32	0.89	0.88	0.88	0.87	0.89
peak	none	0		0.81		0.83	
	none	4		0.85		0.85	
	none	8		0.83		0.85	
	none	16		0.80		0.82	
	none	32		0.79		0.80	
	none	0		0.91		0.91	

Table B4

Reliability of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
area	none	4		0.92		0.91	
	none	8		0.91		0.91	
	none	16		0.90		0.90	
	none	32		0.91		0.90	
liesefeld	none	0		0.88		0.89	
	none	4		0.92		0.92	
	none	8		0.91		0.90	
	none	16		0.90		0.91	
	none	32		0.90		0.91	

Note. Reliability estimated by the split-half correlation. The rows indicate combinations of similarity measure and

Table B5

Validity of different algorithms by task

			[200 700]		[250 700]		
approach	weight	task	penalized	none	penalized	none	penalized
maxcor	none	flanker	0.89	0.82	0.89	0.82	0.89
	none	nback	0.81	0.81	0.81	0.81	0.81
	none	switching	0.72	0.71	0.72	0.71	0.72
	hamming	flanker	0.92	0.91	0.91	0.88	0.85
	hamming	nback	0.89	0.81	0.79	0.74	0.79
	hamming	switching	0.81	0.76	0.82	0.76	0.76
	tukey	flanker	0.92	0.91	0.86	0.84	0.84
	tukey	nback	0.89	0.80	0.78	0.75	0.79
	tukey	switching	0.79	0.74	0.80	0.67	0.78
	normalized	flanker	0.93	0.94	0.93	0.93	0.93
	normalized	nback	0.88	0.90	0.91	0.91	0.90
	normalized	switching	0.88	0.87	0.88	0.87	0.86
	none	flanker	0.82	0.77	0.82	0.77	0.82
	none	nback	0.75	0.73	0.75	0.73	0.75
	none	switching	0.65	0.72	0.65	0.72	0.65
	hamming	flanker	0.90	0.90	0.85	0.85	0.71
	hamming	nback	0.80	0.75	0.77	0.71	0.67
	hamming	switching	0.83	0.82	0.87	0.79	0.66
	tukey	flanker	0.89	0.89	0.87	0.87	0.83
	tukey	nback	0.78	0.75	0.75	0.73	0.71
	tukey	switching	0.82	0.78	0.81	0.77	0.65
	normalized	flanker	0.93	0.93	0.93	0.92	0.93
	normalized	nback	0.88	0.89	0.89	0.88	0.86

Table B5

Validity of different algorithms by task

			[200 700]		[250 700]		
approach	weight	task	penalized	none	penalized	none	penalized
	normalized	switching	0.87	0.86	0.87	0.86	0.87
	none	flanker		0.86		0.88	
peak	none	nback		0.77		0.78	
	none	switching		0.81		0.84	
	none	flanker		0.84		0.81	
area	none	nback		0.80		0.77	
	none	switching		0.72		0.74	
	none	flanker		0.93		0.92	
liesefeld	none	nback		0.84		0.84	
	none	switching		0.82		0.87	

Note. Intra-class correlation focusing on absolute agreement. The rows indicate combinations of similarity measur

Table B6

Validity of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
maxcor	none	0	0.80	0.80	0.80	0.80	0.80
	none	4	0.80	0.79	0.80	0.79	0.80
	none	8	0.81	0.77	0.81	0.77	0.81
	none	16	0.81	0.77	0.81	0.77	0.81
	none	32	0.82	0.77	0.82	0.77	0.82
	hamming	0	0.86	0.81	0.82	0.81	0.80
	hamming	4	0.89	0.87	0.87	0.80	0.83
	hamming	8	0.88	0.83	0.82	0.76	0.81
	hamming	16	0.88	0.79	0.85	0.77	0.77
	hamming	32	0.87	0.83	0.83	0.82	0.78
	tukey	0	0.85	0.80	0.82	0.73	0.79
	tukey	4	0.88	0.87	0.81	0.80	0.84
	tukey	8	0.87	0.82	0.78	0.70	0.79
	tukey	16	0.87	0.80	0.82	0.76	0.79
	tukey	32	0.86	0.81	0.83	0.79	0.81
	normalized	0	0.89	0.88	0.89	0.88	0.88
	normalized	4	0.95	0.95	0.95	0.95	0.95
	normalized	8	0.91	0.91	0.93	0.91	0.91
	normalized	16	0.86	0.89	0.89	0.90	0.88
	normalized	32	0.88	0.89	0.89	0.88	0.88
	none	0	0.74	0.75	0.74	0.75	0.74
	none	4	0.73	0.72	0.73	0.72	0.73
	none	8	0.73	0.75	0.73	0.75	0.73

Table B6

Validity of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
minsq	none	16	0.74	0.73	0.74	0.73	0.74
	none	32	0.76	0.75	0.76	0.75	0.76
	hamming	0	0.84	0.82	0.84	0.78	0.70
	hamming	4	0.85	0.85	0.81	0.78	0.65
	hamming	8	0.84	0.83	0.83	0.79	0.68
	hamming	16	0.83	0.78	0.82	0.78	0.68
	hamming	32	0.87	0.82	0.85	0.79	0.67
	tukey	0	0.82	0.80	0.80	0.78	0.75
	tukey	4	0.86	0.84	0.81	0.80	0.71
	tukey	8	0.81	0.80	0.81	0.79	0.73
	tukey	16	0.82	0.79	0.80	0.78	0.72
	tukey	32	0.85	0.79	0.83	0.79	0.74
	normalized	0	0.86	0.85	0.87	0.84	0.83
	normalized	4	0.94	0.93	0.94	0.93	0.94
	normalized	8	0.90	0.90	0.90	0.90	0.89
	normalized	16	0.88	0.90	0.89	0.89	0.89
	normalized	32	0.89	0.88	0.89	0.88	0.89
peak	none	0		0.75		0.80	
	none	4		0.90		0.90	
	none	8		0.80		0.81	
	none	16		0.82		0.84	
	none	32		0.80		0.82	
	none	0		0.75		0.75	

Table B6

Validity of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
area	none	4		0.83		0.80	
	none	8		0.80		0.78	
	none	16		0.76		0.76	
	none	32		0.78		0.78	
liesefeld	none	0		0.81		0.84	
	none	4		0.90		0.91	
	none	8		0.89		0.89	
	none	16		0.85		0.87	
	none	32		0.85		0.87	

Note. Intra-class correlation focusing on absolute agreement. The rows indicate combinations of similarity measur

Table B7

Validity of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
maxcor	none	4	0.93	0.94	0.93	0.91	0.92
	none	8	0.95	0.95	0.93	0.93	0.93
	none	16	0.94	0.95	0.96	0.96	0.93
	none	32	0.96	0.97	0.96	0.97	0.93
	hamming	4	0.76	0.81	0.72	0.85	0.87
	hamming	8	0.84	0.50	0.65	0.72	0.77
	hamming	16	0.81	0.70	0.86	0.81	0.82
	hamming	32	0.73	0.66	0.85	0.71	0.78
	tukey	4	0.88	0.91	0.78	0.83	0.88
	tukey	8	0.75	0.56	0.69	0.65	0.80
	tukey	16	0.51	0.55	0.74	0.81	0.89
	tukey	32	0.47	0.64	0.71	0.89	0.88
	normalized	4	0.99	0.96	0.97	0.96	0.98
	normalized	8	0.92	0.95	0.91	0.97	0.83
	normalized	16	0.86	0.88	0.98	0.90	0.89
	normalized	32	0.84	0.91	0.98	0.94	0.97
	none	4	0.92	0.94	0.93	0.94	0.91
	none	8	0.94	0.97	0.93	0.94	0.95
	none	16	0.93	0.97	0.90	0.97	0.91
	none	32	0.94	0.95	0.93	0.95	0.94
	hamming	4	0.81	0.79	0.78	0.72	0.78
	hamming	8	0.78	0.62	0.71	0.60	0.75
	hamming	16	0.78	0.81	0.75	0.78	0.80

Table B7

Validity of different algorithms by filter setting

			[200 700]		[250 700]		
approach	weight	filter	penalized	none	penalized	none	penalized
minsq	hamming	32	0.78	0.75	0.71	0.75	0.76
	tukey	4	0.85	0.84	0.81	0.82	0.80
	tukey	8	0.80	0.77	0.73	0.69	0.78
	tukey	16	0.83	0.79	0.81	0.78	0.87
	tukey	32	0.76	0.84	0.75	0.74	0.90
	normalized	4	0.98	0.99	0.98	0.96	0.95
	normalized	8	0.91	0.87	0.96	0.94	0.92
	normalized	16	0.93	0.96	0.95	0.97	0.96
	normalized	32	0.93	0.93	0.93	0.98	0.91
peak	none	4		0.91		0.87	
	none	8		0.91		0.79	
	none	16		0.84		0.78	
	none	32		0.73		0.81	
area	none	4		0.81		0.80	
	none	8		0.81		0.79	
	none	16		0.81		0.77	
	none	32		0.82		0.77	
liesefeld	none	4		0.91		0.91	
	none	8		0.83		0.82	
	none	16		0.91		0.82	
	none	32		0.92		0.86	

Note. Intra-class correlation focusing on absolute agreement. The rows indicate combinations of similarity measur