Behav Res (2011) 43:1194–1201
DOI 10.3758/s13428-011-0106-8

A simple framework (ASF) for behavioral and neuroimaging experiments based on the psychophysics toolbox for MATLAB

Jens Schwarzbach

Published online: 26 May 2011

Abstract The cognitive neurosciences combine behavioral experiments with acquiring physiological data from different modalities, such as electroencephalography, magnetoencephalography, transcranial magnetic stimulation, and functional magnetic resonance imaging, all of which require excellent timing. A simple framework is proposed in which uni- and multimodal experiments can be conducted with minimal adjustments when one switches between modalities. The framework allows the beginner to quickly become productive and the expert to be flexible and not constrained by the tool by building on existing software such as MATLAB and the Psychophysics Toolbox, which already are serving a large community. The framework allows running standard experiments but also supports and facilitates exciting new possibilities for real-time neuroimaging and state-dependent stimulation.

Keywords Behavioral experiments · MEG/EEG · TMS · Functional MRI · MATLAB · Psychophysics toolbox · Slideshow · Online rendering · State-dependent stimulation · Real-time fMRI

Introduction

A common problem in experimental psychology and cognitive neuroscience is that experimenters either program experiments from scratch by using programming languages such as C++ or MATLAB, which often leads to highly specific code with limited reusability, or alternatively, use software packages that, after some time, reveal their limits in adaptability to the specific problems the experimenters need to address. These problems get aggravated when the experimenters work at a center that uses different modalities, such as electroencephalography (EEG), magnetoencephalography (MEG), transcranial magnetic stimulation (TMS), and functional magnetic resonance imaging (fMRI) to study the human mind and brain. Different labs develop different tools and different paradigmatic views on how to conceptualize an experiment. To give an example, in TMS, the experiment drives the machine, while in fMRI, the timing of the machine basically drives the timing of an experiment, and multimodal experiments require both. This can easily lead to fractionation or inefficient reprogramming of the same experiment, depending on which modality one wants to use.

The goal of the project described here is to provide a framework for programming experiments that guarantees near real-time or submillisecond precision and that gives the freedom to implement all sorts of conceivable designs and to embed interfaces with complex hardware while encouraging programming reusable and clear code. At the same time the framework is usable for teaching and provides beginners with fast success with little effort—hence, the name *A Simple Framework* (ASF).

ASF is built around the Psychophysics Toolbox Version 3 (Brainard, 1997; Kleiner, Pelli, & Brainard, 2007; Pelli, 1997), which is a free set of MATLAB and GNU/Octave functions for vision research that runs on multiple platforms (Windows, Mac OSX, Linux) and that makes it easy to synthesize and show accurately controlled visual, auditory,and tactile stimuli. ASF is freely available at

Electronic supplementary material The online version of this article (doi:10.3758/s13428-011-0106-8) contains supplementary material, which is available to authorized users.

J. Schwarzbach (✉)
Department of Cognitive Science and Education,
Center for Mind/Brain Sciences, Trento University,
Trento, Italy
e-mail: jens.schwarzbach@unitn.it

http://code.google.com/p/asf/, with a discussion forum hosted at http://groups.google.com/group/asf-forum. ASF provides a certain view on how to structure an experiment, without limiting the user to that view, and it provides interfaces with the different hardware components of different experimental modalities, such that the experimenter can concentrate on the core paradigm. ASF requires the installation of MATLAB and the Psychophysics Toolbox. ASF emphasizes accurate timing, easy interfacing with different hardware, and clearly documented interfacesfor input (stimulus and design) and resulting data (responses, timestamps, hard- and software settings).

General structure of ASF

A typical experiment can be conceived of as involving randomization, calculation of timing, stimulus rendering, stimulus presentation, response collection, and data analysis.

ASF separates these parts in a way that users can set up a range of different experiments, without programming at all. ASF takes care of those tasks that are inherent in every computer-controlled experiment: stimulus presentation and response collection. The basic version of ASF is conceptualized as a slideshow with maximally achievable temporal accuracy. It has three modules that cover the tasks of an experiment: Design, Runtime, and Analysis (see Fig. 1).

1. Design (randomization, calculation of timing, stimulus rendering)
   a) The user provides a stimulus definition file (.STD) that contains a list of filenames that point to the stimulus material (picture or sound files) (see example.std in the Supplementary Material section).
   b) The user provides a trial definition file (.TRD) that contains a sorted list of trial definitions. Each line in this text file defines a trial by providing information about what condition a trial belongs to, plus information about which stimulus material shall be shown when and how to handle responses (see Fig. 4 and example.trd in the Supplementary Material section).
   c) The user provides configuration settings (Cfg) for hard- and software (e.g., refresh rate, type of response device, synchronization to MR scanner, EEG/MEG triggers, interfacing with an eyetracker, screen background color)
2. Runtime (stimulus presentation, response collection)
   The experiment is conducted by the runtime engine, which the beginner can treat as a black box that presents pictures and sounds as determined by the information contained in the stimulus definitions, trial definitions, and configuration settings. The advanced user may want to create so called plugins that extend stimulus presentation from showing prefabricated slides to complex online-generated multisensory stimulation. Further explanations on how to create plugins can be found in the Supplementary Material section.
3. Analysis
   ASF creates an output file that contains a structurevariable that contains information about the environment in which the experiment was run (hardware, operating system, default and user-specified settings) and trial-by-trial information about what was shown when and what the participant responded and when. Timestamps are provided with microsecond resolution. This allows checking for accuracy of the implemented experiment. Basic functionality for data export and analysis is provided.

Thus, design aspects, such as ordering of trialsand aspects of balancing and timing, are addressed outside the

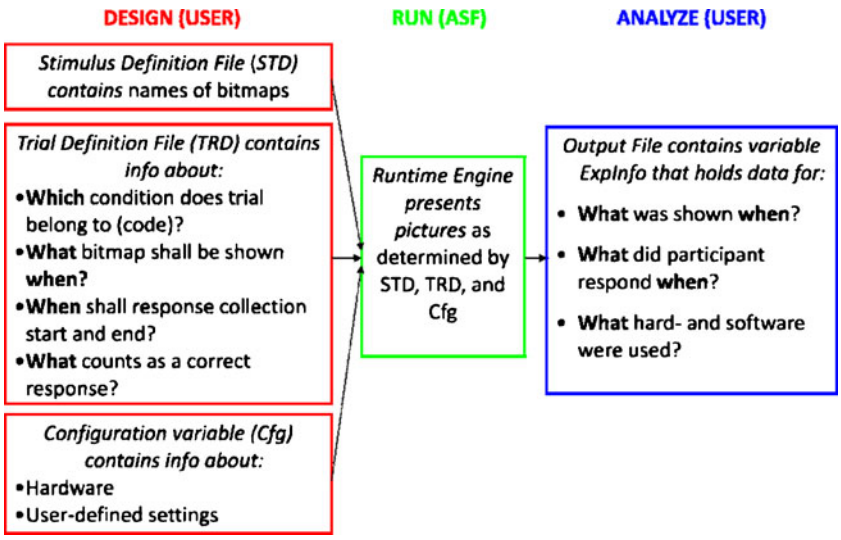

Fig. 1 Partitioning of empirical experiments into modules within ASF

runtime module. This allows beginners to use a text editor to set up the design, while the advanced user with programming experience may decide to write a procedure that automatically creates the text files that determine the design (see the Supplementary Material section). Keeping design and runtime modules separate leads to higher flexibility in adapting designs to a researcher's needs.

Example

To demonstrate the basic functionality of ASF, the following section describes how to implement a masked priming experiment similar to one in a previously published study using metacontrast masking (see Fig. 2; Vorberg, Mattler, Heinecke, Schmidt, & Schwarzbach, 2003). This experiment requires high accuracy in timing and lends itself to testing in other modalities, such as MEG/EEG, TMS, and fMRI (see below).

In masked priming, one is interested in whether visibility of a prime stimulus affects its ability to prime motor responses to a target stimulus. To this end, Vorberg et al. (2003) presented small arrows that pointed to the left or to the right (primes) and that were masked by a larger leftward- or rightward-pointing arrow (mask). Stimulus onset asynchrony (SOA) between prime and mask was varied from 0 to 100 ms. For the sake of simplicity, we will reduce the experiment to a 2 × 2 design here, with $\text{congruence}_{\text{prime–mask}}$ (congruent, incongruent) and $\text{SOA}_{\text{prime–mask}}$ (50, 100 ms) as experimental factors.

Assume that stimuli are presented on a 60-Hz display. This means that the duration of any presented stimulus is quantized in steps of 1-frame duration (1/60th of a second, or 16.67 ms). Each trial begins with a fixation cross being shown at the center of the screen for 30 frames (500 ms). Primes are shown above fixation for 1 frame. Primes point either to the left or to the right. $\text{SOA}_{\text{prime–mask}}$ is either 3 or 6 frames (50 or 100 ms). The mask is shown for 6 frames (100 ms). Masks appear centered around the same location as the prime, and they point either to the left or to the right. After extinguishing the mask, there is a response period of 90 frames (1,500 ms), during which a fixation cross is shown and during which participants can respond to the orientation of the mask by pressing the left or the right mousebutton. Between trials, there is a fixed intertrial interval of 30 frames (500 ms). Reaction time measurement starts with presentation of the mask.

The entire experiment (see the Supplementary Material section for full listings) can be realized by presenting six different pictures or slides. By means of a texteditor or a custom-written routine, the user needs to create a textfile (stimulus definition file with the ending std; see example.std in the Supplementary Material section) that contains the filenames of all images that make up an experiment(e.g., S01_empty.bmp, S02_fix.bmp, S03_prime_left.bmp, S04_prime_right.bmp, S05_mask_left.bmp, S06_mask_right.bmp).

In a second step, the user needs to create trial descriptions. For example, a congruent trial in which a prime and a target both point to the left with an SOA of 50 ms can be described as a sequence of five pairs of numbers, with the first number referring to the slide to be shown and the second number referring to the slide's respective duration in units of vertical screen refresh (frame; see Fig. 3). Thus, such a trial can be represented by [2, 30], [3, 1], [2, 2], [5, 6], [1, 90].

Such descriptions need to be written into a trial definition (.trd) file (see Fig. 4), which consists of two sections. The first line contains the factorial information for your experiment. If you want to code your experiment—for example, as a 2 × 2 design—this line would read '2 2'. The remainder of the file contains one entry per trial. It starts

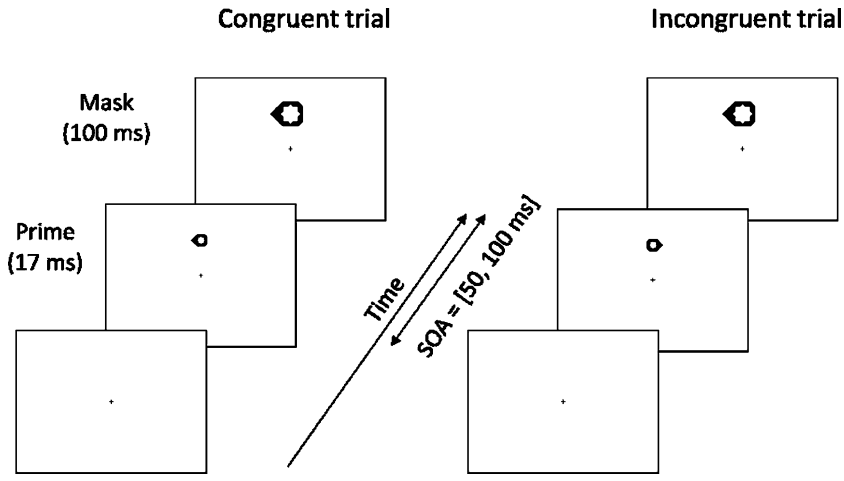

Fig. 2 A masked priming experiment in which primes and masks are presented at different SOAs and response congruencies. Researchers investigate whether presenting a masked stimulus can influence reaction times to a target stimulus, which is, in this case, the mask itself (see, e.g., Vorberg, Mattler, Heinecke, Schmidt, & Schwarzbach, 2003)

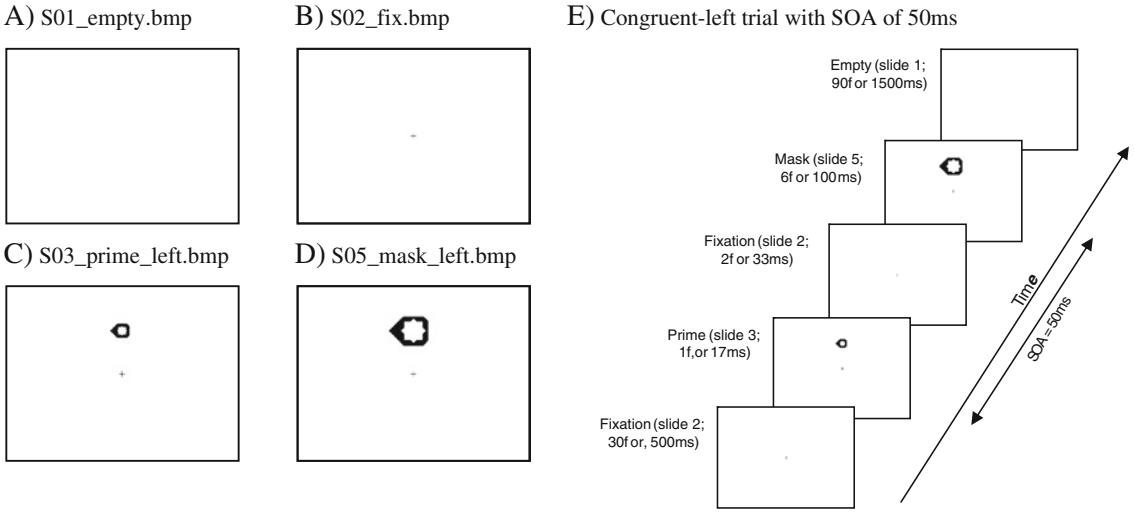

Fig. 3 **a**–**d** Sample bitmaps needed on a congruent trial in which both the prime and mask are pointing to the left. **e** Sequence of events (pages) on a trial on which the prime and target point to the left and are presented with an SOA of 50 ms: fixation cross for 500 ms (slide 2 for 30 frames), prime (small arrow left, slide 3 for 1 frame), fixation (slide 2 for 2 frames), mask (large arrow left, slide 5 for 6 frames), empty screen (slide 1 for 90 frames). Reaction time measurement starts with the onset of the mask

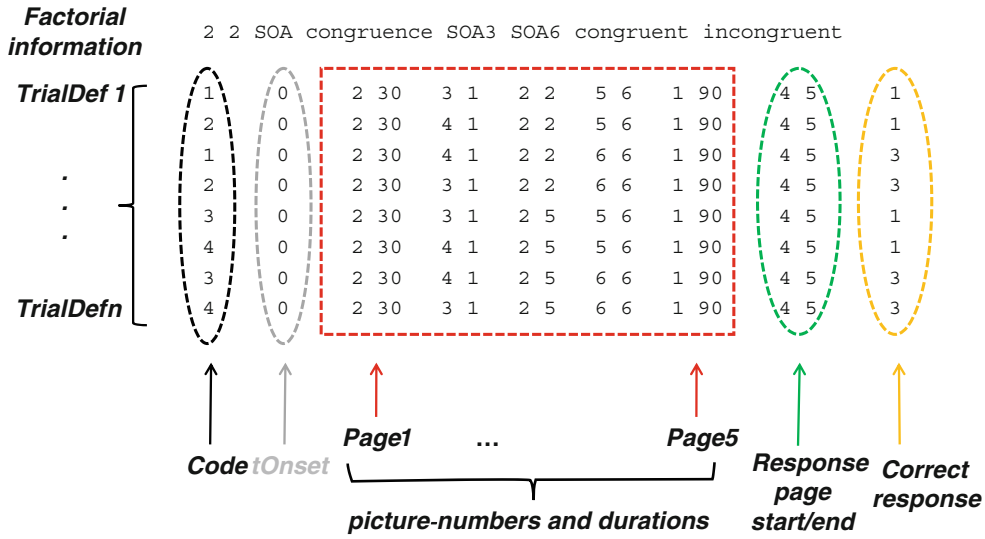

Fig. 4 Annotated trial definition file for a masked priming experiment with congruence (two levels) and SOA (two levels) as factors indicated in the first line, followed by the factor names and the names of each factorial level. The remainder of the file contains one line per trial. The first (*black*) column provides a code for data analysis condition (code 1: congruent, SOA 50 ms; code 2: incongruent, SOA 50 ms; code 3: congruent, SOA 100 ms; code 4: incongruent, SOA 100 ms). The second column (*gray*) contains each trial's onset time with respect to the start of the experiment (used in fMRI experiments; here, the value 0 means that there are no waiting periods and that trials are played backtoback). What follows (*framed in red*) is a series of pairs of columns, with two columns for each picture-event or page—namely, slide number and slideduration. In this sample experiment, each trial consists of five pages, with page 1 being the prestimulus period of 30 frames, page 2 being the prime of one frame, page 3 being the interstimulus interval, page 4 being the mask of 6 frames, and page 5 being an additional responsewindowof 90 frames. Different experiments may differ in the number of pages they require. ASF can also handle experiments in which different trials consist of a different number of pages. The last three columnsdescribe response parameters: the number of the page at which response collection is supposed to start (in the sample experiment, this is page 4, when the mask is presented), the page number when response collection shall end. The last column contains the keycode for a correct response, which can be used for feedback or data analysis. In this example, a correct response is determined by the direction of the mask. Trials on which the mask points to the left require pressing the left mouse button (code 1), while trials on which the mask points to the right require pressing the right mouse button (code 3). The first trial definition refers to the trial described in Fig. 3 (congruent, left, SOA = 50 ms). The last trial definition describes an incongruent trial in which the prime points to the left and the mask points to the right, with an SOA of 100 ms

with a trialcode anda trial onset time (in seconds) with respect to experimentstart (by default not used), followed by pairs of entries that describe a picture-event or page (slide number, slide duration in frames). The second-to-last entry contains the page-number at which reaction time measurement on a given trial should start. The last entry contains a number for the correct response.

While creating trialdescriptions for experiments by using a texteditor or a spreadsheet program is conceivable, one may, in the long run, wish to create such files with a customized program that allows randomizing trials or flexibly changing design aspects, such as stimulus durations or experimental factors. Readers can find such a program in the Supplementary Material section.

Within MATLAB, the experiment is run by invoking ASF with the following command:

$$\mathrm{ExpInfo} = \mathrm{ASF}('\mathrm{example.std}', '\mathrm{example.trd}', '\mathrm{example}', []).$$

The user will see an opening screen with information on screen resolution and refreshrate. The experiment starts with default settings after a mousebutton has been pressed and returns a structurevariable ExpInfo, which contains information about the environment in which the experiment was run (hardware, operating system, default and user-specified settings) and trial-by-trial information about what was shown when and what the participant responded and when. For each event, timestamps are provided with microsecond resolution and an accuracy that is, in general, better than 1 ms (see Fig. 5).

After the experiment has been run, its timing accuracy can be assessed by calling

$$\mathrm{ASF\_timingDiagnosis}(\mathrm{ExpInfo}),$$

which creates a graph (Fig. 5) depicting the differences between expected time of events and their respective time-stamps when run on a particular computer. This allows checking for framedrops, which would indicate either that the hardware is insufficient (see the Supplementary Material section on recommended hard- and software) or that the code (especially of plugins) has some parts that require too much computation time. The experimenter can thus try to change the code or can try out different hardware. If this is not an option anymore, because the experiment has already been conducted, this timingcheck can help the experimenter to discard certain trials at the time of analysis. Subframe variability up to a few milliseconds informs the experimenter about the eventual variability of received or emitted triggers, which, for example, can be taken into account in trigger-based averaging of signals in EEG/MEG or neurophysiology.

Data necessary for analyzing reaction times and error rates can be extracted from ExpInfo by invoking

$$\mathrm{dat} = \mathrm{ASF\_readExpInfo}(\mathrm{ExpInfo}),$$

which returns a matrix dat that contains as many rows as trials and four columns (trialcode, reaction time, correctresponse as determined by the experimenter, actual response of the participant). Such data can then be further analyzed in MATLAB or exported for analysis in other software packages.

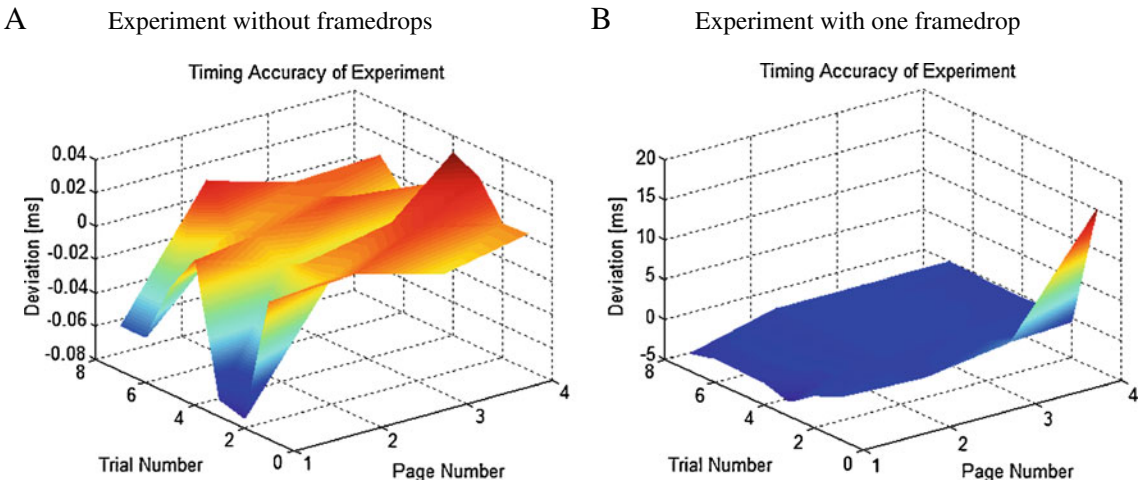

Fig. 5 Output of ASF_timingDiagnosis, depicting accuracy of timing for two replications of the example experiment consisting of eight trials. **a** A session without framedrops. **b** A session with one framedrop on page 4 of trial 1 (note the different scales for the *z*-axis). Each trial (*y*-axis) consisted of a subsequent presentation of four pages (*x*-axis). The dependent measure (*z*-axis) shows the deviation in milliseconds of the timestamp for each page on a given trial from the expected timestamp derived from the trialdefinitions. Negative values indicate that the event occurred too early, and positive numbers indicate that the event occurred too late, with respect to the expected occurrence of a stimulus. This graph demonstrates the high accuracy of timing that can be achieved with the psychophysics toolbox. The maximum deviation was smaller than one tenth of a millisecond. The example experiment was run under Windows 7 on a DELL PRECISION M6400 laptop equipped with a NVIDIA Quadro FX 3700 graphics card operating with a refresh rate of 60 Hz

Function list

The following functions (MATLAB m-files) are part of the ASF distribution (Table 1). In order to remain compatible with future updates, it is recommended not to edit ASF.m. However, its core function for stimulus delivery, ASF_showTrialSample.m, can be used as a template for user-defined stimulus delivery using MATLAB, PsychToolbox, and the ASFfunctions described below.

Tutorials

The ASF distribution contains four PowerPoint presentations to introduce basic concepts, dataanalysis and export, and pluginwriting. A "HowTo" section covers timing of visual stimuli, timing of auditory stimuli, using different methods (wavplay, audioplayer, PTB-snd, and PsychPortAudio), and audiovisual synchronization. Furthermore, it contains a "Tutorial Projects" section with a

Table 1 Description of functions implemented in ASF

For running experiments	
ASF	The core programfor running an experiment (reads stimulus and trial definition; manages playback and logging; communicates with devices)
For creating trd-files	
ASF_encode	Encode factorial information to single number code
For plugins	
ASF_showTrialSample	Copy and make your own experiment with online rendering, adaptive testing, etc.
ASF_xFlip	Flip display synchronized with vertical retrace (and much more). One of the most important functions of ASF
ASF_playSound	Wrapper function for different methods to play sound
ASF_sendMessageToEyelink	For logging stimulus or other events in the stream of eye-tracking data
ASF_checkforuserabort	Checks whether somebody has pressed the q key to quit program
ASF_sendMessageToEyelink	For logging stimulus or other events in the stream of eye-tracking data
ASF_checkforuserabort	Checks whether somebody has pressed the q key to quit program
ASF_decode	Decoding codes to factorial information
ASF_setTrigger	Send signals to parallel port or an Arduino board (useful for TMS, EEG, MEG)
ASF_waitForMousePressBenign	Just waits for a mouse press
ASF_waitForResponse	Waits for a response. Can handle many different response devices (mouse, lumina parallel, lumina serial, voicekey, keyboard)
ASF_waitForScannerSynch	Pauses until an MRscanner trigger arrives
For data analysis	
ASF_timingDiagnosis	To check whether everything happened when it was supposed to
ASF_readExpInfo	Reads logfiles created by an ASF experiment
ASF_getTrialOnsetTimes	Returns a vector for trial onset times
Internal helper functions	
ASF_initEyelinkConnection	Establishes initialization of an EyeLink eyetracker
ASF_shutdownEyelink	Closes connection with EyeLinkeyetracker
ASF_initParallelPortInput	Initializes the parallel port using the data acquisition toolbox
ASF_initResponseDevice	Initializes response device (mouse, voicekey, Lumina-parallel, Lumina-serial, keyboard)
ASF_arduinoTrigger.m	Function for triggering using an Arduino board
ASF_readTrialDefs	Reads trdfiles into a structure
ASF_readStimulus	Reads bitmaps and avi files
ASF_makeTexture	Internal function for mapping bitmaps onto textures
ASF_PTBExit	Generates a graceful exit (save logs, shut down hardware, remove objects from memory)
Experimental	
ASF_checkTrials	MATLAB visualization of trialscheme
ASF_onlineFeedback	On dual screen setups, this shows the current cumulative info on errors and reaction time (even for factorial experiments)
ASF_pulseTrainNI	Can create pulse trains on a national instruments card for rTMS

growing number of projects, which currently comprises the masking project covered in this article, a working memory experiment that introduces plugins, and an fMRI-ready experiment for mapping out the motion complex MT/MST, using random-dot kinematograms.

Extensions

Additional built-in features

ASF allows users to change many parameters for the display (e.g., screen resolution, refresh rate), for stimulation (e.g., Quaerosys piezo stimulator for tactile stimulation, http://www.quaerosys.com/), for response devices (mouse, keyboard, voicekey, eyetracker), and for triggering (input and output). Here, only triggering will be discussed, since it shows the multimodal capabilities of ASF. For a full documentation of configuration settings, consult ASF's documentation or type "help ASF" on MATLAB's command line.

Eye-tracking It is often desirable to measure the position of gaze during an experiment, either to ensure that participants keep their eyes fixated at a certain location or in order to have participants respond with eyemovements instead of or in addition to manual responses. ASF fully supports the EyeLink toolbox (Cornelissen, Peters, & Palmer, 2002). By adding one configuration setting with which ASF is called, the experiment will start with a calibration procedure and will record eyemovement data, provided the experimental system is connected to an eyetracker supported by the EyeLink toolbox:

```
Cfg.Eyetracking.useEyelink = 1;
ExpInfo = ASF('example.std', 'example.trd', 'example', Cfg)
```

This configures ASF to put time-stamped markers into the eye-tracking data whenever a trial starts ('TRIALSTART'), and whenever a new page is presented ('PAGE xyz', where xyz stands for the page number as defined in the .std file). This allows offline analyses of eye-tracking data (e.g., saccadic reaction times) with respect to the timing of events in the experiment. ASF contains a documented tutorial in which the above described example experiment is turned into an eye-tracking experiment with saccadic responses. The tutorial also provides a documented function for the analysis of the eye-tracking data.

EEG/MEG For the analysis of evoked potentials in EEG or evoked magnetic fields in MEG, one usually puts an eventtrigger in the data stream of the data acquisition device. This can be achieved by setting a configuration flag at startup of an experiment:

```
Cfg.issueTriggers = 1; ExpInfo
  = ASF('example.std', 'example.trd', 'example', Cfg).
```

This will, for each page time-locked to its presentation, send a trigger code (i.e., the picture number) to the parallel port, which can be connected to the dataacquisition device.

fMRI For running an fMRI experiment, two built-in features are provided. First, the onset of the experiment can be synchronized to the onset of the MR data acquisition by waiting for a trigger sent by the MR scanner via a parallel or serial port or a TTL signal to a digital input device (e.g., by National Instruments). Furthermore, the user can provide trial onset times in the trd-file with respect to the first synch-pulse from the scanner. This feature is extremely important for creating event-related designs with jitter or nulltrials (Friston, Zarahn, Josephs, Henson, & Dale, 1999).The script makeExampleTrd.m in the Supplementary Material section implements some simple trialjitter. The respective call for running the fMRI version of the example experiment would be

```
Cfg.useTrialOnsetTimes = 1;
  Cfg.synchToScanner = 1;
ExpInfo = ASF('example.std', 'exampleScripted.trd', 'example', Cfg).
```

User-definable extensions via plugins: TMS, online rendering, real-time fMRI

ASF uses the function ASF_ShowTrial() (defined in ASF_ShowTrial.m) to present stimuli. In order to enhance the functionality of stimulus presentation, the user can create a plugin. A plugin is a MATLAB m-file the user writes—for example, using ASF_ShowTrial.m as a template and saving it under a different name, such as myShowTrialPlugin.m.

For example, a TMS-researcher may wish to investigate the effect of a TMS pulse time-locked to the prime in order to see whether TMS of a certain area alters the effect of priming (see, e.g., Sack, van der Mark, Schuhmann, Schwarzbach, & Goebel, 2009). This can be achieved by extending the trialdefinition file with information about when, in a given trial, a TMS pulse should be issued. The plugin myShowTrialPlugin.m would have to interpret the additional information from the trial definition file and contain a piece of code for issuing the TMS pulse. When

starting the experiment, one can tell ASF to exchange its built-in ASF_ShowTrial() function with the plugincode. An example of such a plugin is contained in the tutorial that comes with ASF.

Using plugins allows implementing experiments in which the approach of presenting prefabricated images is not practical anymore or a predetermined trialstructure cannot be used, such as in adaptive procedures. Here, the ShowTrial function should be altered to perform online rendering of the stimulus controlled by information provided by the trdfile. Such an approach has been used to investigate attention to motion in random-dot kinematograms (Furlan & Schwarzbach, 2011).

Plugins allow for a highly interesting new approach to experimenting in the cognitive neurosciences—that is, state-dependent stimulation. ASF contains an example plugin that processes real-timefMRI information from an ongoing MR scan that can be used to decide what stimulus the participant will see at a given time. ASF provides a host of functions and interfaces that help in creating such plugins (see the Supplementary Material and Table 1).

Conclusions

Here, a framework has been presented that different groups from the cognitive neurosciences can use to run experiments that employ one or several methods, even simultaneously. The framework builds upon MATLAB and the Psychophysics Toolbox, which run on different operating systems (Windows, MacOSX, Linux). The standard use of ASF does not require any programming. It just requires putting together text, image, and soundfiles that are transformed by ASF to a (near) real-time multimedia slideshow that allows collecting responses. Programming can be used to automatically generate such text files—for example, for randomization procedures and balancing (see the Supplementary Material section). Finally, MATLAB- or C/C++ programming can be used to create more complex stimulation procedures that require onliner-endering. The framework allows running standard experiments but also facilitates exciting new research paradigms, such as real-time neuroimaging and state-dependent stimulation.

If you want to acknowledge the use of ASF when you publish your research, please cite not only this article, but also those describing the underlying technologies, such as the Psychophysics Toolbox (Brainard, 1997; Pelli, 1997) and Eyelink Toolbox extensions (Cornelissen et al., 2002).

Acknowledgements The author wishes to thank Angelika Lingnau for interfacing ASF with the EyeLink Toolbox (Cornelissen et al., 2002) and for continuous feedback during the development of ASF and on earlier versions of the manuscript; Gianpaolo Demarchi for interfacing ASF with the parallel port and the Quaerosys tactile stimulator; and the two anonymous expert reviewers of this article. This work has been supported by the Provincia autonoma di Trento and the Fondazione Cassa di Risparmio di Trento e Rovereto.

References

Brainard, D. H. (1997). The psychophysics toolbox. *Spaialt Vision, 10,* 433–436.

Cornelissen, F. W., Peters, E. M., & Palmer, J. (2002). The eyelink toolbox: Eye tracking with MATLAB and the psychophysics toolbox. *Behavioral Research Methods, Instruments, & Compuerst, 34,* 613–617.

Friston, K. J., Zarahn, E., Josephs, O., Henson, R. N., & Dale, A. M. (1999). Stochastic designs in event-related fMRI. *Neuroimage, 10,* 607–619.

Furlan, M., & Schwarzbach, J. (2011). *Spatial attention does not lead to input- or response gain in processing motion coherence*. Manuscript in preparation.

Kleiner, M., Pelli, D. G., & Brainard, D. H. (2007). What's new in Psychtoolbox-3? *Perception*, *36*(Abstract Suppl.).

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

Sack, A. T., van der Mark, S., Schuhmann, T., Schwarzbach, J., & Goebel, R. (2009). Symbolic action priming relies on intact neural transmission along the retino-geniculo-striate pathway. *Neuroimage, 44,* 284–293.

Vorberg, D., Mattler, U., Heinecke, A., Schmidt, T., & Schwarzbach, J. (2003). Different time courses for visual perception and action priming. *Proceedings of the National Academy of Sciences, 100,* 6275–6280.