Audiovisual presentation of video-recorded stimuli at a high frame rate

Björn Lidestam

Linköping University Post Print

N.B.: When citing this work, cite the original article.

The original publication is available at www.springerlink.com:

Björn Lidestam, Audiovisual presentation of video-recorded stimuli at a high frame rate, 2014, Behavior Research Methods, (46), 2, 499-516.

<http://dx.doi.org/10.3758/s13428-013-0394-2>

Copyright: Springer Verlag (Germany)

<http://www.springerlink.com/?MUD=MP>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-110292>

Audiovisual Presentation of Video-recorded Stimuli at High Frame-rate

Björn Lidestam

Linköping University

Author Note

Björn Lidestam, Department of Behavioural Sciences, Linköping University, Sweden

This research was funded by the Swedish Research Council, grant no. 2006-6917. Matlab programming was done by Mattias Ragnehed (visual presentation), Amin Saremi (auditive presentation), and Carl-Fredrik Neikter (audiovisual presentation). Jerker Rönnberg and Shahram Moradi are thanked for good cooperation, discussions, advice, and the empirical data collected. Katarina Marjanovic spoke all stimuli for the recordings. Teddy Fredriksson provided valuable technical advice. Robert Eklund provided valuable advice on acoustics. Felix-Sebastian Koch, Tomas Jungert, and two anonymous reviewers provided valuable comments on the manuscript.

Correspondence concerning this article should be addressed to Björn Lidestam, Department of Behavioural Sciences, Linköping University, SE-581 83 Linköping, Sweden.

E-mail: bjorn.lidestam@liu.se

Telephone: +46 13 282155

Abstract

A method for creating and presenting video-recorded synchronized audiovisual stimuli at a high frame-rate, which is highly useful in psychophysical studies on for example JNDs and gating, is presented. Methods for accomplishing this comprise of recording audio and video separately using an exact synchronization signal, editing the recordings and finding exact synchronization points, and presenting the synchronized audiovisual stimuli with a desired frame-rate on a CRT display using Matlab and Psychophysics Toolbox 3. The methods from an empirical gating study (Moradi, Lidestam, & Rönnberg, 2013) are presented as an example of implementation of playback at 120 fps.

Keywords: psychophysics, frame rate, audiovisual, synchronization, temporal resolution

Audiovisual presentation of video-recorded stimuli at high frame-rate

This paper presents a new method for playback of synchronized audiovisual video-recorded stimuli at high frame-rate, which cannot be obtained with standard playback softwares and procedures.

The problem is that standard frame rate of video is today low: 25 or 29,97 fps in television broadcasts and on DVD video, and 24 fps on BD (Bluray Disc) video and in cinemas. Most commercial softwares for psychological research (e.g., E-prime, Superlab, and Presentation) implement video playback by media player softwares (e.g., Windows Media Player and Quicktime), also at standard frame-rate. However, standard playback rate does not render very smooth and natural motion, which can for example readily be observed when watching camera-sweeps on a high definition monitor—motion is jerky, which can be annoying.

For research purposes, standard playback rate is potentially even more problematic, since ecological validity may be compromised. For example, in experiments on object tracking and detection of fast motion, motion should preferably be smooth and natural, not jerky. Another example is that high-enough frame rate is required in order to accurately present fast periodical events, such as fast blinks and oscillations. Artifacts such as the wagon-wheel effect (the illusion that a wheel appears to rotate in the opposite direction of its true rotation) should also be avoided for naturalistic presentation.

It is also sometimes a problem that the temporal resolution of the stimuli do not permit manipulations and measurements with high-enough resolution. This is because the temporal resolution of video stimuli is discrete, and the standard frame rate of 25 fps, for example, means a resolution of $1 \text{ s} / 25 = 40 \text{ ms}$. Audio stimuli, in comparison, are continuous and therefore have infinitely high time resolution, which allows much more exact manipulations and therefore also better possibilities to optimize research designs.

The purpose was to accomplish higher frame-rate in visual and audiovisual video playback than present-day standard technology offers, on standard computer equipment that is available in most laboratories doing studies in perception and psychophysics.

A subordinate purpose was to accomplish audiovisual synchronization that is more precise and reliable than possible with the standard method that is used in film production, for video-recorded stimuli. This was needed since there are currently no video cameras that record audiovisually at higher frame rates than 30 fps on the market, since high-speed recording on video cameras is only designed for slow-motion playback (i.e., playback at standard frame-rate, at a fraction of the recording speed). Therefore audio and video, respectively, need to be recorded separately and synchronized later when recording at high frame-rate. Capturing *exactly* and reliably when the parts of a clapper (as used in film production) "clap" together on video is virtually impossible, unless frame rate were infinitely high. The time resolution of the video is proportional to frame rate, and each frame therefore spans a specific number of milliseconds. Establishing exactly when—within one frame—that the pieces of the clapper make contact, is virtually impossible. In film production the synchronization error may therefore be as large as $1 \text{ s} / 24 \approx 42 \text{ ms}$. The method presented here can be used to precisely and reliably ascertain synchronization well below 1 ms, and was successfully implemented in Moradi, Lidestam, and Rönnberg (2013).

A method for presenting video recordings as fast as a screen can present

The method presented in the present paper allows presentation rate to be as fast as a CRT (cathode ray tube) display can operate; which can be upwards 160 Hz. In order to accomplish this with standard computer equipment, Matlab (2009b, 32-bit) and Psychophysics Toolbox 3 (Brainard, 1997; Pelli, 1997; Kleiner, Brainard, & Pelli, 2007) can be used. In Moradi et al. (2013), 120 fps was used. However, the computer's processing speed determines the spatial resolution that can be used for video presentation at the desired rate. Too many pixels to process may result in VBL (vertical blanking or retrace interval) errors, if the data flow is too fast for the VBL rate;

that is, if there are too many pixels to render at the set VBL rate. Moradi et al. chose to set the spatial resolution close to the screen setting of 800×600 pixels at 120 fps, and to crop the video files such that superfluous areas were excluded, thereby lowering processing demands and avoiding VBL errors.

As stated above, the upper limit for temporal resolution is currently restricted by the frame rate on a CRT display. This is because a CRT display is needed to ascertain that presentation is correct and that there are no VBL errors; it provides feedback to the computer, to instantaneously tell whether presentation was correct with regard to timing and all frames being presented. Timing errors and missed frames may cause problems in their own right for visual-only presentation, as well as cause asynchrony in audiovisual presentation (Kleiner, Brainard, Pelli, Ingling, Murray, et al., 2007). Most laboratories conducting research within visual perception and psychophysics therefore still use CRT displays, even though they have gone out of fashion for most other applications.

More modern display technologies, such as plasma, LCD (liquid chrystal displays), LED (light emitting diod), and OLED (organic light emitting diod) screens, allow faster presentation rate and generally have better technical specifications, but they do not provide feedback to the computer on whether presentation was correct with regard to timing and that all frames actually were displayed. Checking correctness of presentation would then be much more cumbersome, suggestedly by using a photo cell mounted on the screen to measure a small area designated for presentation check by using white odd frames and black even frames, providing a flicker of the desired presentation rate. It would also be necessary to develop a procedure to organize and analyze the quality control data. Developing and describing such a procedure was however not within the scope of the present paper. Therefore Moradi et al. (2013) used a CRT display, with the desired result.

Benefit of high frame rate in research on perception and psychophysics

As of today, playback of video-recordings at frame rates exceeding 30 fps, with or without synchronized sound, is not available by standard consumer technologies (but 48 fps playback is beginning to be used in some cinemas, see HFR, n.d.). Whenever naturalness and smoothness of motion is required in stimulus presentation, fast-enough frame rate is needed. For research purposes, some types of studies even require use of small time-steps as a manipulation of the independent variable. There are many types of studies where fast audiovisual (or just visual) presentation of video-recordings is useful. Some examples are presented below.

Psychophysical studies. In order to study *JNDs* of video-recorded stimuli, a fast-enough time resolution can be essential for different reasons. Today's highest standard video presentation frame-rate is 30 fps (frames per second), which means that each video frame is displayed for $1/30 \text{ s} = 33,33 \text{ ms}$. In psychophysical experiments this can be regarded as a large step size (Friberg & Sundberg, 1995; Tervaniemi et al., 2006), not small enough for, for example, finding absolute *JNDs* (just noticeable differences) for discrimination of vowel *duration* in Swedish (Lidestam, 2009). Higher frame rate is therefore required.

When studying *feature detection*, the critical features such as vibration, should be conveyed. The Nyquist–Shannon sampling theorem (see Lüke, 1999) states that in order to capture signal frequency, the sampling must at minimum have the double frequency. A standard video-recording and playback at 25 fps can therefore not convey oscillations (or other periodical events) that have a frequency above about 12 Hz. For example, in standard Swedish, the consonants /l/ and /r/ have the same place of articulation, and only differ such that /l/ is pronounced with the tongue in a static position, while /r/ is pronounced with the tip of the tongue vibrating at about 28 Hz. If this somehow were to be recorded and presented with the standard rate of 25 or 29.97 fps, this tongue vibration would not be captured, since the sampling rate is roughly the same as the stimulus rate. For instance, the recording might only show the tongue in its lowest position, or as fuzzy, since the tongue moved during exposure time for each frame. With 120 fps, on the

other hand, the tongue will be captured in each position, and will be displayed as less fuzzy in each frame.

The frame rate is also potentially crucial to *object tracking*, since motion becomes fuzzy or jerky at too low frame rates. For example, in order to develop a valid and reliable video-based test of how accurately a human can track a tennis ball and call a serve on or very close to the line "in" or "out", very fast frame rate would be needed. This is why the Hawk-Eye ball tracking system (Hawk-Eye Innovations, 2013) used in major tennis tournaments uses high-speed cameras for tracking balls travelling at high velocities. For example, after a fast serve the ball can travel at 161 kph (100 mph) just before bouncing by the service line. It then moves as far as 1.79 m in one frame at 25 fps (i.e., 40 ms), compared to 4.47 cm in one frame at 1000 fps. High frame-rate is also interesting to use for the afore mentioned reasons in psychocinematic research (e.g., Gross & Levenson, 1995; Hasson et al., 2008; Schwan & Ildirar, 2010; Shimamura, 2013).

Gating paradigm studies. In gating studies, successively more of a stimulus is presented until the stimulus, typically a word, is identified and the percept is stable (cf. Grosjean, 1980, 1996). Narrow gates means higher resolution in the measurement, and therefore better reliability as well. Also, with the opportunity to use small time steps (i.e., frames), presentation becomes more flexible, since it is possible to use multiples of the frames to optimize gate size for specific purposes. It not always desirable to use as narrow gates as possible; presentation may become too tedious and time consuming, and there may therefore be a risk that the participants lose motivation or become fatigued (see Moradi et al., 2013), hence risking poor validity.

Method

In order to present the method and how it can be implemented, the methods (including the Matlab script used for stimulus presentation) from a recent gating study by Moradi et al. (2013) are presented. The stimuli were consonants presented in vowel–consonant–vowel (/aCa/) format and words presented in consonant–vowel–consonant (/CVC/) format, presented in noise and in silence. Note that the materials should be viewed as examples: technical specifications do

not need to be identical, but should generally not be below those of the materials used for stimulus presentation.

Recording

Video. A RED ONE video camera (RED Digital Cinema, 2013) was used to record the video at 120 fps, in 2048×1536 pixels. At this high frame rate, the camera cannot record sound, which therefore had to be recorded separately. There are many consumer cameras that can record at 120 fps and higher, at lower spatial resolutions, on the market.

Audio. A Sony HVR-V1E video camera with a Sony EVM-NV1 directional electret condenser stereo microphone (Sony, 2013) placed 1.7 m from the speaker recorded the sound at 48 kHz sampling rate and 16 bits in stereo, while recording video at PAL standard 25 fps and 1920×1080 pixels. As a benefit of having the video recorded in the same file as the audio, handling and matching of the audio and video files, respectively, was facilitated.

Synchronization signal. In order to provide an audiovisual signal to allow precise synchroniztion of the separate audio and video recordings, an electronic voltage test pen (Voltage Tester GK7, manufactured by Zhangzhou Eastern Intelligent Meter Co., Ltd., China, see Figure 1), purchased at a major national hardware and electronics chain-store, was used together with a looped electric AC cord connected to a switch, which was clicked on and off to activate the voltage test pen. Any similar device with a an LED and buzzer should provide synchronized blinks and beeps. The voltage test pen blinked and sounded simultaneously (i.e., each blink and beep were synchronized), with a frequency of 18 blinks-and-beeps per second, at about 2700 Hz. Each blink had a duration of approximately 35 ms, whereas each beep was about 36 ms in duration.

In order to identify the separate items to be used, a running stop watch was rigged together with the voltage test pen, on the side of the actress speaking the items (Figure 1), together constituting a cinema clapperboard with very high time resolution. Therefore each item had a synchronization signal from the test pen, as well as a time code.

Editing

Video. Separate files of the recorded items, including the entire synchronization signals, were made using Final Cut Pro 7.0.3, of files delivered in Apple ProRes 4444 and 2048×1024 pixels (see Figure 1). The files were named according to their respective speech token and time code (e.g., 'bad_1.31.34' for the word token /bad/, meaning "bath", recorded at 1.31.34 according to the stop watch, see Figure 1). Then these video files were cropped such that the voltage test pen, stop watch, and irrelevant areas surrounding the talker were removed, so the number of pixels to be processed was reduced to 600×670 pixels and saved as noncompressed MOV files, using Quicktime Pro 7. This also lowered processing demands for playback, ensuring that presentation could be executed without VBL errors according to Psychophysics Toolbox (Brainard, 1997; Pelli, 1997; Kleiner, Brainard, & Pelli, 2007). The number of pixels was also chosen to match the screen settings used for presentation, meaning no loss in spatial resolution. Actual presentation on the screen was 600×600 pixels, with the presentation format further slightly cropped at top and bottom (due to the screen setting), see Figure 2. All frames were kept, to allow synchronization.

Figure 1. Video frame of the talker (2048×1024 pixels), voltage test pen used as clapper (blinking red), and running stop watch (for identification of separate items), as used for editing and synchronization of audio and video.

Figure 2. Cropped video frame of the talker (600×600 pixels), as presented to the participants.

Audio. Audio files, one per item, were made from the recordings with the Sony HVR-V1E video camera. Each file included both synchronization signal (i.e., beeps) and the speech token to be presented. As with the video files, all audio files were named according to the token and time code. Bit depth and resolution was kept as 16 bits and 48kHz, and the files were saved in WAV format.

Inspecting and choosing items

The audiovisual files from the standard frame-rate Sony HVR-V1E video camera were inspected to choose the best items from different recordings of the same item. The video was inspected for eyeblinks and anything else that could distract the participants. The audio was inspected for noise, clicks, and audibility, primarily phoneme distinctiveness, in order to avoid phonetic ambiguity. The best items, with regard to both audio and video, were chosen.

Synchronization. First, the number of blinks from the voltage test pen were counted in all video files; and all beeps were counted in the audio files, ensuring that the synchronization signal

was correct in all recordings (i.e., that the numbers of blinks and beeps matched). The median number of blinks and beeps was seven.

For each *video* file, onset and offset time of the blinks were found and registered as a number of frames from the beginning of the file. Since the LED in the voltage test pen was lit up and went out virtually instantaneously (less than 100 ns, cf. Philips Lumec, n.d.), light intensity within frames was used to estimate the length of time within each frame that the LED was lit, allowing time estimates for onset and offset to be fractions of the 8.33 ms that was the exposure time for each frame. The mean of the onset and offset time was then used as a synchronization point.

In the *audio* files, onset of the first beep and offset of the last beep were measured and registered for each file, by checking waveforms with Sound Studio 4 (Felt Tip, n.d.), see Figure 3. The mean of the onset and offset times was then used as synchronization point.

Figure 3. Waveforms for the auditory synchronization signal (beeps, marked in blue) and the speech token (the word "bad", meaning "bath"), respectively.

Whenever the time between the synchronization point and the start point of the designated audio playback did not match the video frame duration of 8.33 ms, the audio synchronization point was adjusted to match the number of frames that represented the synchronization point of the video file, thereby optimizing synchronization.

Since the mean of many samples (i.e., frames) of a relatively long synchronization signal (as compared to a clapper used when recording motion pictures) was used, synchronization was virtually perfect. For example, when the visual synchronization point was calculated for a

synchronization signal with 7 blinks, the blinks were captured in a series of 47 frames. Since the frame resolution was 8.33 ms, the resolution for the mean of the 47 frames, used as synchronization point, was $8.33 / 47 = 0.18$ ms.

Stimulus presentation

Scripts and setup. The experiment was written and run in Matlab (2009b, 32-bit), using the Psychophysics Toolbox 3 extensions (Brainard, 1997; Pelli, 1997; Kleiner, Brainard, & Pelli, 2007). The script is included in the Appendix. In the experiment settings, gate size (in terms of time increments in the audio presentation, and the number of frames presented as video), and signal-to-noise ratio (with chosen noise audio file) could be set. Item presentation orders and time settings for start and stop of playback (standard stimuli and start of first gate, respectively) were specified in separate text files. Before playback, each video (MOV) file was converted to an array of still pictures. The still pictures were then read and displayed, one per VBL update, with Psychophysics Toolbox 3. All presentations in terms of lists, items, times, and potential VBL errors were logged into log files. The experiment was carried out with dual-screen setup, such that stimuli were presented on a CRT screen in front of the participant, and the Matlab interface was presented on another screen in front of the experimenter, allowing the experimenter control over all events without distracting the participant.

Computer equipment. An Apple Mac Pro 4.1 running OSX 10.6.8, equipped with a 2.93 GHz Quad-core Intel Xeon processor, 32 GB 1066 MHz DDR3 RAM, and an ATI Radeon 4870 graphics processor was used. In order to get short interstimulus intervals (i.e., avoid delay in presentation from slow transfer of data from the hard drive to the RAM), a solid-state drive with 555 Gb/s read time and SATA-III connection at 6 Gb/s were used. The largest file used was a 1.19 GB video file of a sentence (743 frames with 600×670 pixels each), and it was retrieved and played back in about 5 s with this configuration.

Visual stimuli were displayed on a 22 inch Mitsubishi Diamond Pro 2070SB CRT monitor (Mitsubishi Electric, 2013), set at 800×600 pixels and 120 Hz, and viewed from a distance of

about 55 cm. Audio signals were presented binaurally at approximately 65 dB via Sennheiser HDA 200 headphones (Sennheiser, 2012), with sound pressure level initially and individually set and fixed at comfort level for each participant.

Discussion

The major benefit of this method is its high temporal resolution for presentation of video-recorded stimuli, meaning that motion and fast periodical events can be displayed accurately and with good ecological validity. Another benefit is that the method allows precisely synchronised audiovisual presentation of video-recorded stimuli. Further, the high temporal resolution allows precise stimulus manipulations and measurements. The method can be implemented with standard equipment and softwares used in most psychological and psychophysical laboratories, and if certain equipment is lacking, it is easy to acquire and relatively inexpensive.

The method was used and evaluated in the gating paradigm study by Moradi et al. (2013) to present synchronized audiovisual speech stimuli. There were no technical mishaps (e.g., no crashes or VBL errors), and good results were yielded. For example, there was a very large audiovisual benefit over auditory-only presentation in terms of how accurately (error rate) and early (isolation points, cf. Grosjean, 1980) the speech was identified. Also, the need for invoking top-down processing was reduced, such that the correlations that speech identification scores had with attention and working memory scores in an auditory-only condition were absent in the audiovisual condition. This shows that the synchronized audiovisual presentation was very well accepted (i.e., effortlessly and automatically processed) by the participants in Moradi et al. The methods presented here can therefore be claimed to be valid and reliable.

With regard to ecological validity and accurate display of motion and periodical events, the benefit of 120 fps over 25 fps became very evident during preparations of the materials in Moradi et al. (2013). The electronic voltage tester, used as synchronisation signal, simultaneously blinked and "chirped" at a rate of 18 blinks-and-beeps per second. Audio and video were recorded with a video camera at standard frame-rate of 25 fps. At the same time, the video to be

used for stimulus presentation in the experiment was recorded with the high frame-rate camera at 120 fps (i.e., 4.8 times faster recording rate than 25 fps). Each blink was recorded in approximately four frames in the high frame-rate video—while the standard frame-rate video failed to show about half the number of blinks, since the number of frames per second just exceeded the number of blinks per second by a factor of 1.4. In order to ascertain that all blinks are captured, the frame rate must be at minimum twice the blink rate, according to the Nyquist–Shannon sampling theorem—in this case, at least 36 fps. Thus, high frame-rate could capture and display this periodical event accurately, but standard frame-rate could not.

Due to the high temporal resolution that comes with high frame-rate, there is also previously unparalleled flexibility with regard to resolution for finding small JNDs in video-recorded visual and audiovisual stimuli, and gate size in gating paradigm studies. For example, Moradi et al. (2013) could use gates as narrow as 8.33 ms, but multiples by two and four (i.e., 16.67 ms and 33.33 ms, respectively), were chosen for pragmatic purposes.

Correctness of synchronization at playback could be further objectively validated by video-recording the synchronized playback of the recordings of the test pen (displayed on the CRT monitor) with a high-speed video camera, at no less than 240 fps. Twice the frame rate of 120 fps is required to reliably record each frame presented on the monitor, according to the Nyquist–Shannon theorem. It would however not be possible to get truly audiovisual recordings (i.e., recordings with audio and video in the same, synchronized, file as in a standard video recording), since no current video camera records sound at such high frame-rates. To remedy this problem a fast audio-to-video device (e.g., an LED VU-meter) could be used. At synchronized audiovisual playback of the recorded audio and video (as presented to the participants), the sound from the playback of recordings of the test pen's beeps could trigger an LED as soon as sound exceeds a certain level, allowing fast and synchronized blinks to be triggered by the audio playback.

However, it would be difficult to ascertain that this conversion is indeed instantaneous, but this approach could validate that synchronization was reliable, since a small offset would be constant.

By inspecting the ≥ 240 fps video-recordings with both the blinks on the monitor and the blinks from the sound-triggered device in the same picture, synchrony (or asynchrony) could be established.

Recommendations for future studies are as follows. Firstly, to show the dependency between frame rate for visually and audiovisually presented stimuli, and JNDs for temporal events (e.g., object tracking and feature detection). Secondly, to test sensitivity to audiovisual synchrony when high frame-rate is used. If sensitivity to visual temporal events benefits from higher frame-rates, then sensitivity to audiovisual asynchrony should also increase, and especially for events of short duration. Thirdly, research and development of methods and technology for visual and audiovisual playback of video-recordings at even higher frame-rates than CRT displays allow, and with full control over timing, is strongly suggested. Finally, with regard to how much better and more fluent video-recorded motion looks in 120 fps compared to standard presentation rate, implementation of the methods presented here is suggested.

References

- Brainard, D. H. (1997). The Psychophysics Toolbox. *Spatial Vision*, 10, 433–436.
- Felt Tip (n.d.). *Sound Studio 4*. Retrieved from <http://feltp.com/ss/>
- Friberg, A., & Sundberg, J. (1995). Time discrimination in a monotonic, isochronous sequence. *The Journal of the Acoustical Society of America*, 5, 2524–2531.
- Grosjean, F. (1980). Spoken word recognition processes and the gating paradigm. *Perception & Psychophysics*, 28, 267–283.
- Grosjean, F. (1996). Gating. *Language and Cognitive Processes*, 11, 597–604.
- Gross, J. J., & Levenson, R. W. (1995). Emotion elicitation using films. *Cognition and Emotion*, 9, 87–108.
- Hasson, U., Landesman, O., Knappmeyer, B., Vallines, I., Rubin, N., & Heeger, D. J. (2008). Neurocinematics: The neuroscience of film. *Projections*, 2, 1–26.
- Hawk-Eye Innovations (2013). *Hawk-Eye: Tennis*. Retrieved from <http://www.hawkeyeinnovations.co.uk/page/sports-officiating/tennis>
- HFR (n.d.). *High frame rate movie news, info, downloads & discussion*. Retrieved from <http://www.hfrmovies.com>
- Kleiner, M., Brainard, D., & Pelli, D. (2007). What's new in Psychtoolbox-3? *Perception*, 36, ECVF Abstract Supplement.
- Kleiner, M., Brainard, D., Pelli, D., Ingling, A., Murray, R., et al. (2007). *What's new in Psychtoolbox-3? A free cross-platform toolkit for psychophysics with Matlab & GNU/Octave*. (PDF document with slides from ECVF 2007, Arezzo, Italy). Retrieved from <http://psychtoolbox.org/PsychtoolboxLinks>
- Lidestam, B. (2009). Visual discrimination of vowel duration. *Scandinavian Journal of Psychology*, 50, 427–435.
- Lüke, H. D. (1999). The origins of the sampling theorem. *IEEE Communications Magazine*, 37, 106–108. doi: 10.1109/35.755459

Mitsubishi Electric (2013). *Diamond Pro 2070SB*. Retrieved from

<http://www.mitsubishielectric.com.au/2345.htm>

Moradi, S., Lidestam, B., & Rönnberg, J. (2013). Gated audiovisual speech identification in

silence versus noise: Effects on time and accuracy. *Frontiers in Psychology*, 4:359. doi:

10.3389/fpsyg.2013.00359

Pelli, D. G. (1997). The VideoToolbox software for visual psychophysics: Transforming numbers

into movies. *Spatial Vision*, 10, 437–442.

Philips Lumec (n.d.). *Advantages and disadvantages of LEDs*. Retrieved from

http://www.lumec.com/learning-center/led/advantages_and_disadvantages.html

RED Digital Cinema (2013). *RED ONE: The original*. Retrieved from

<http://www.red.com/products/red-one>

Schwan, S., & Ildirar, S. (2010). Watching film for the first time: How adult viewers interpret

perceptual discontinuities in film. *Psychological Science*, 21, 1–7.

Sennheiser (2012). *HDA 200*. Retrieved from [http://en-us.sennheiser.com/audiometers-high-](http://en-us.sennheiser.com/audiometers-high-frequency-testing-closed-dynamic-ear-protector-headphones-hda-200)

[frequency-testing-closed-dynamic-ear-protector-headphones-hda-200](http://en-us.sennheiser.com/audiometers-high-frequency-testing-closed-dynamic-ear-protector-headphones-hda-200)

Shimamura, A. P. (2013). Presenting and analyzing movie stimuli for psychocinematic research.

Tutorials in Quantitative Methods for Psychology, 9, 1–5.

Sony (2013). *Sony HVR-V1E*. Retrieved from

<http://www.sony.nl/pro/lang/en/nl/product/hdvcamcorders/hvr-v1e/overview>

Tervaniemi, M., Jacobsen, T., Röttger, S., Kujala, T. Widmann, A., Vainio, M., Näätänen, R.,

& Schröger, E. (2006). Selective tuning of cortical sound-feature processing by

language experience. *European Journal of Neuroscience*, 23, 2538–2541.

Appendix

Matlab script

```

-----
%% This script plays movies from separate movies (with no sound) and audio
% files. The playback is synchronized so the movie and audio is starting
% at the correct moment. Between each playback the user is asked if he or she
% recognized the word, and if he or she wants to continue.
%
%% Inputs: none
% A config file is selected that instructs the function of which experiment
% folder that is involved. Each experiment folder should contain
% sub-folders with the included movie and audio files. There are also
% settings that represent values for synchronization, start, stop etc.
%
%% Some details
% Loads a movie high fps clip and displays it using PsychToolbox.
function psych_word_experiment()

    %% SOME SETUP THINGS
    DEBUG = false;
    HomeScreen = 0;
    ExperimentScreen = 1;
    NO_VIDEO = 0;
    windowSize = [];
    RESULT_DIREC = 'Results';
    LOG_DIREC = 'Logs';
    %CLIP_DIREC = 'Media';
    DEF_DIR = 'Definitions';
    EXP_DIR = 'Experiments';
    REINIT_CYCLE_COUNT = 1;

    if DEBUG
        %ExperimentScreen = 0
        NO_VIDEO = 0;
        %windowSize = [0 0 400 200];
    end
    clc;
    home;
    Screen('Preference', 'SkipSyncTests', 0); % must be '0' during real experiment

    % setup working directory and logging
    %WORK_DIR = '[specify file path]';
    %WORK_DIR = '[specify file path]';
    [scriptPath, scriptName, scriptExt] = fileparts(which(mfilename));
    WORK_DIR = fullfile(scriptPath, '..');
    cd(WORK_DIR);

    % set up as logging
    RUN_LOG_FILE = sprintf('RUN-%s.log', datestr(now, 'yyyy-mm-dd-HHMMSS'));
    diary(fullfile(LOG_DIREC, RUN_LOG_FILE));
    diary('on');

    RESULT_LOG_FILE = fullfile(RESULT_DIREC, 'EXP_RESULTS.txt');
    fid_log = fopen(RESULT_LOG_FILE, 'a', 'native', 'UTF-8');

    function log_line(line)
        % print to Command Window and results log file
        disp(line);

```

```

    fprintf(fid_log, sprintf('%s\n', line));
end

function log_text(text)
    % print to Command Window and results log file
    disp(text);
    fprintf(fid_log, text);
end

% psych toolbox settings
% ListenChar(2); % prevent key presses to interfere with matlab,
output_level = 2; % 1-5; 1: minimal, 5: maximal;
% more output = slower execution, change when running
% real experiment (1-3?).
Screen('Preference', 'Verbosity', output_level);

vblPerFrame = 1; % 1 in real experiment
PAUSE_INTERVAL = 1000; % pause after every N frame;

%% get config file that specifies the experiment
cd(DEF_DIR);
[conf_file, conf_dir, filter_ix] = uigetfile('*.txt', 'Select config file');
if filter_ix == 0
    return;
end
cd(WORK_DIR);
fid = fopen(fullfile(conf_dir, conf_file), 'r');

%% extract experiment info
% DIR
temp = textscan(fid, 'DIR %s', 1);
%experiment_dir = fullfile(WORK_DIR, EXP_DIR, temp{1}{:});
experiment_dir = fullfile(conf_dir, '..', EXP_DIR, temp{1}{:});

% NOISE_FILE
temp = textscan(fid, 'NOISE_FILE %s', 1, 'commentStyle', '%');
temp = temp{1};
noiseFile = "";
useNoise = 0;
if (numel(temp) > 0)
    useNoise = 1;
    noiseFile = fullfile(conf_dir, '..', EXP_DIR, temp{:});
    %disp(noiseFile);
end;

% SNR
temp = textscan(fid, 'DEFAULT_SNR %d', 1);
defaultSnr = temp{1};

% PLAYBACK_PAUSE_MS
temp = textscan(fid, 'PLAYBACK_PAUSE_MS %d', 1);
playbackPauseMs = temp{1};

% ORDER_FILE
temp = textscan(fid, 'ORDER_FILE %s', 1, 'commentStyle', '%');
temp = temp{1};
orderFile = "";
useOrderFile = 0;
if (numel(temp) > 0)
    useOrderFile = 1;
    orderFile = fullfile(experiment_dir, temp{:});
end;

```

```

fclose(fid);

if (useOrderFile)
    fid = fopen(char(orderFile), 'r');
    orderDirs = textscan(fid, '%s', 'delimiter', ',');
    orderDirs = orderDirs{1};
    fclose(fid);
end;

try

    % prepare for the experiment

log_line('=====');
log_line('NEW EXPERIMENT');
log_line(['START: ' datestr(now, 31)]);
log_line(['CONFIG: ' fullfile(conf_dir, conf_file)]);

winExperiment = 0;
if (~NO_VIDEO)
    winExperiment = Screen('OpenWindow', ExperimentScreen, ...
        [150 155 170], windowSize);
    % set PTB to have high priority to minimize OS scheduling of other
    % processes.
    Priority(MaxPriority(winExperiment));
end;

% Initialize for high-precision, low-latency sound output.
InitializePsychSound(1);

if (useNoise)
    % read noise file.
    [noiseData, noiseFs, noiseBits] = wavread(noiseFile);
    % convert from N channels to mono.
    noiseData = noiseData(:);
    numNoiseSamples = numel(noiseData);
end;

%%% get data from the experiment folder
if (useOrderFile)
    % prepend path to subdirs
    subDirs = cellfun(@(x) fullfile(experiment_dir, x), ...
        orderDirs, 'UniformOutput', false);
else
    dirData = dir(experiment_dir);
    dirDataIx = [dirData.isdir];
    subDirs = {dirData(dirDataIx).name};
    % exclude . and .. folders
    subDirs = subDirs(~(strcmp('.', subDirs) | strcmp('..', subDirs)));
    % prepend path to subdirs
    subDirs = cellfun(@(x) fullfile(experiment_dir, x), ...
        subDirs, 'UniformOutput', false);
end;

subDirsIx = zeros(1, numel(subDirs));
movieFiles = cell(1, numel(subDirs));
audioFiles = cell(1, numel(subDirs));
execFiles = cell(1, numel(subDirs));

if NO_VIDEO
    flipInterval = (1 / 120); % 120 Hz

```

```

else
    flipInterval = Screen('GetFlipInterval', winExperiment);
end;
flipIntervalMs = flipInterval * 1000;

% find movie, audio and exec files.
for subIx = 1:numel(subDirs)
    subDir = subDirs{subIx};

    dirData = dir(fullfile(subDir, '*.mov'));
    movieFiles{subIx} = "";
    audioFiles{subIx} = "";
    execFiles{subIx} = "";
    if (numel(dirData) >= 1)
        movieFiles{subIx} = fullfile(subDir, dirData(1).name);
    else
        dirData = dir(fullfile(subDir, '*.avi'));
        if (numel(dirData) >= 1)
            movieFiles{subIx} = fullfile(subDir, dirData(1).name);
        end
    end

    dirData = dir(fullfile(subDir, '*.wav'));
    if (numel(dirData) >= 1)
        audioFiles{subIx} = fullfile(subDir, dirData(1).name);
    end

    dirData = dir(fullfile(subDir, 'exec.txt'));
    if (numel(dirData) >= 1)
        execFiles{subIx} = fullfile(subDir, dirData(1).name);
    end;

    if (numel(movieFiles{subIx}) == 0 || numel(audioFiles{subIx}) == 0 || ...
        numel(execFiles{subIx}) == 0)
        log_text(sprintf('No movie/audio/exec file found in %s', ...
            subDirs{subIx}));
    else
        subDirsIx(subIx) = 1; % folder content OK
    end;
end;

subDirsIx = logical(subDirsIx); % make logical array for logical indexing
% filter out directories and files that have no complete content.
subDirs = subDirs(subDirsIx);
movieFiles = movieFiles(subDirsIx);
audioFiles = audioFiles(subDirsIx);
execFiles = execFiles(subDirsIx);

waveSamples = cell(1, numel(subDirs));
clipStartNo = zeros(1, numel(subDirs));
clipStopNo = zeros(1, numel(subDirs));
clip1stGateNo = zeros(1, numel(subDirs));
clipGateWidth = zeros(1, numel(subDirs));
audioStartMs = zeros(1, numel(subDirs));
audioSnr = zeros(1, numel(subDirs));

audioFs = 0;
snrOverriden = 0;
prevAudioFs = 0;
prevAudioBits = 0;

% read exec files and audio data.

```

```

for subIx = 1:numel(subDirs)
    fid = fopen(char(execFiles{subIx}), 'r');

    % CLIP_START_FRAME
    temp = textscan(fid, 'CLIP_START_FRAME %d', 1);
    clipStartNo(subIx) = temp{1};

    % CLIP_STOP_FRAME
    temp = textscan(fid, 'CLIP_STOP_FRAME %d', 1);
    clipStopNo(subIx) = temp{1};

    % CLIP_1ST_GATE_FRAME
    temp = textscan(fid, 'CLIP_1ST_GATE_FRAME %d', 1);
    clip1stGateNo(subIx) = temp{1};

    % CLIP_GATE_WIDTH
    temp = textscan(fid, 'CLIP_GATE_WIDTH %d', 1);
    clipGateWidth(subIx) = temp{1};

    % AUDIO_START_MS
    temp = textscan(fid, 'AUDIO_START_MS %d', 1);
    audioStartMs(subIx) = temp{1};

    % AUDIO_START_MS
    temp = textscan(fid, 'AUDIO_SNR %d', 1);
    if (numel(temp{1}) > 0) % overridden?
        audioSnr(subIx) = temp{1};
    else
        audioSnr(subIx) = defaultSnr;
    end;

    fclose(fid);

    % read audio wave file
    [audioData, audioFs, audioBits] = wavread(char(audioFiles{subIx}));
    if (prevAudioFs == 0)
        prevAudioFs = audioFs;
        prevAudioBits = audioBits;
    else
        if prevAudioFs ~= audioFs
            error('Audio files cannot have mixed sample frequency.');
```

end;

```

        if useNoise && audioFs ~= noiseFs
            error('Audio and noise files cannot have mixed sample frequency.');
```

end;

```

        waveSamples{subIx} = audioData;
    end;

    % calculate good-to-have audio data.
    audioStopMs = audioStartMs + (clipStopNo - clipStartNo + 1) * ...
        flipIntervalMs;
    audioStartSample = round(audioStartMs * 1e-3 * audioFs);
    audioStopSample = round(audioStopMs * 1e-3 * audioFs);
    numAudioSamples = audioStopSample - audioStartSample;

    % convert from N channels to mono.
    for subIx = 1:numel(subDirs)
        % numRows == num samples, numCols == num channels
        numSamples = size(waveSamples{subIx}, 1);

```

```

    numChannels = size(waveSamples{subIx}, 2);
    waveData = zeros(numSamples, 1);
    for channelIx = 1:numChannels
        waveData = waveData + waveSamples{subIx}(:,channelIx);
    end
    waveSamples{subIx} = waveData / numChannels;
end

% do rms normalization on the audio data.
waveSamples = rms_normalization(waveSamples);

% mix with noise with the appropriate SNR
if (useNoise)
    waveSamples = mix_noise(waveSamples, noiseData, audioSnr);

    % do rms normalization (again) on the audio data to make sure that
    % the energy of audio files are equal and no overflow occurs.
    waveSamples = rms_normalization(waveSamples);
end;

% Cut before start and after stop samples.
% It is important that this cut is coming after RMS normalization.
% Otherwise the RMS data computations might not work correctly.
% The RMS computations should be done on all audio samples.
for subIx = 1:numel(subDirs)
    if (useNoise && numNoiseSamples < numAudioSamples(subIx))
        log_text(subDirs{subIx});
        error('Audio content (between start and stop) cannot have less samples than noise content.');
```

end

```

    if (audioStartSample(subIx) > numel(waveSamples{subIx}))
        log_text(subDirs{subIx});
        error('Audio start time must be less than total audio time.');
```

end

```

    if (audioStopSample(subIx) > numel(waveSamples{subIx}))
        log_text(subDirs{subIx});
        error('Audio stop time must be less than total audio time.');
```

end

```

    waveSamples{subIx} = waveSamples{subIx}(audioStartSample(subIx):audioStopSample(subIx));
end

% Open the default audio device [], with default mode [] (==Only playback),
% and a required latencyclass of zero 2 == low-latency mode, as well as
% a frequency of freq and nrchannels sound channels.
% This returns a handle to the audio device:
nrChannels = 1;
latencyClass = 0; % > 0 makes the Matlab window flicker with high CPU usage.
pahandle = PsychPortAudio('Open', [], [], latencyClass, ...
    audioFs, nrChannels);
autoPlayback = 1;
reinitCycle = 0;

% playback of video and sound.
for subIx = 1:numel(subDirs)

    if (~autoPlayback)
        input('Press Enter twice when ready.', 's');
        input('', 's');
    end;

    if (reinitCycle == REINTT_CYCLE_COUNT)
        %% Re-initialize PsychToolbox.
        % This will trigger the immediate release of textures as well,
```



```

% although we have already released them.
reinitCycle = 0;

if (winExperiment ~= 0)
    %Screen('Close');
    %Screen('Close', winExperiment);
    Screen('CloseAll')

    winExperiment = Screen('OpenWindow', ExperimentScreen, ...
        [150 155 170], windowSize);

end
end;

reinitCycle = reinitCycle + 1;

% in case NO_VIDEO is true.
movieTex = zeros(0);
log_line(' ');

if ~NO_VIDEO
    %% read movie file.
    % Comment: Actual frame rate at recording is 120 fps.
    % However, no such standard for playback exists. Hence, a standard playback
    % rate must be chosen for the video file. Playback at the default rate
    % will therefore be in slow motion. Playback at original rate (as in
    % this script) will look as the original rate.
    % open display , fullscreen
    log_text(sprintf('Reading of %s', char(subDirs{subIx})));

    [moviePtr movDuration movFps movWidth movHeight movFrameCount] = ...
        Screen('OpenMovie', winExperiment, char(movieFiles{subIx}));

    % validity checks
    if (clipStartNo(subIx) > movFrameCount)
        error(sprintf('Video start frame must be less than or equal movie frame count %d.', movFrameCount));
    end
    if (clipStopNo(subIx) > movFrameCount)
        error(sprintf('Video stop frame must be less than or equal movie frame count %d.', movFrameCount));
    end

    %% create a pause image
    % initialize imgPause with ones in a dimension according to video resolution.
    %imgPause = ones(vid_size([2 1]), 'uint8');
    imgPause = ones([movHeight movWidth], 'uint8');
    % imgPause(:) syntax produces a way to fetch and assign a matrix from
    % a vector (array).
    imgPause(:) = 128*(1:numel(imgPause)) / (numel(imgPause) + 2*eps);
    %break_tex = Screen('MakeTexture', winExperiment, imgPause, 0);
    repmat(imgPause, [1,1,3]);
    imgPause(:,1) = 150;
    imgPause(:,2) = 155;
    imgPause(:,3) = 170;
    respond_tex = Screen('MakeTexture', winExperiment, imgPause, 0);

    %% create movieTex for all frames
    frameIndex = clipStartNo(subIx):clipStopNo(subIx);

    movieTex = zeros(1, numel(frameIndex)); % pre-allocate array for performance reasons
    % cut before start frame
    for n = 1:clipStartNo(subIx)-1
        [tex pts] = Screen('GetMovieImage', winExperiment, moviePtr);

```

```

        Screen('Close', tex); % Free immediately
    end
    % cut after stop frame
    ix = 0;
    for n = frameIndex
        ix = ix + 1;
        [tex pts] = Screen('GetMovieImage', winExperiment, moviePtr);
        movieTex(ix) = tex;
    end
end

toFrameNo = clip1stGateNo(subIx) - clipStartNo(subIx) + 1;
response = 'N';
%response = 'No';

log_text(sprintf('Playback of %s', char(subDirs{subIx})));

while (~strcmp(response, 'Y') && ~(numel(movieTex) > 0 && ...
    toFrameNo > numel(movieTex)))

    % Finally, do the real playback
    playback(playbackPauseMs, movieTex, waveSamples{subIx}, toFrameNo, flipInterval);

    %response = questdlg('Did you recognize the word?', 'Alert', 'Yes', 'No', 'No');
    response = input('Did you recognize the word? Y/N [Y]: ', 's');
    response = upper(response);
    %log_line(response);

    pause(.3);
    toFrameNo = toFrameNo + clipGateWidth(subIx);
end

%% Clear PsychToolbox resources
Screen('Close', movieTex);
clear movieTex;

autoPlayback = 0;
end

log_line(' ');
log_line(['END OF EXPERIMENT: ' datestr(now, 31)]);

log_line('=====');

catch ex
    len = length(ex.stack);
    disp(['Error: ' ex.message]);
    disp(ex.identifier);

    for mm = 1:len
        disp('');
        disp(ex.stack(mm))
    end

end

%% FINAL CLEAN-UP
% Take care of experiment data and clean up
if (winExperiment ~= 0)
%     Screen('Close', winExperiment);
    Screen('CloseAll')
end

```

```

% Close the audio device:
if (exist('pahandle'))
    PsychPortAudio('Close', pahandle);
end;

Priority(0);
fclose('all');
%ListenChar(0); % respond to keys again
diary('off');
clear all;

function playback(pauseMs, movieTex, waveSamples, toFrameNo, flipInterval)

    vblPerFrame = 1; % 1 in real experiment

    % calculate good-to-have audio data (remember that audio data has been
    % cut before start and after end.
    audioGateMs = toFrameNo * flipIntervalMs;
    audioGateSample = floor(audioGateMs .* 1e-3 .* audioFs);
    waveSamples = waveSamples(1:audioGateSample);

    % transpose audio data since PsychToolbox wants every row to be
    % the channel data (the columns represents the channels in wavread).
    waveSamples = waveSamples';

    % Fill the audio playback buffer with the audio data 'wavedata':
    % 'bufferdata' is usually a matrix with audio data in double() or
    % single() format. Each row of the matrix specifies one sound channel,
    % each column one sample for each channel.
    PsychPortAudio('FillBuffer', pahandle, waveSamples);

    if ~NO_VIDEO
        flipIntervalMs = flipInterval * 1000;

        vbIs = zeros(1, toFrameNo);
        vbl = Screen('Flip', winExperiment);
        %vbl = now;
        %disp(vbl);
        %disp(now);

        if (pauseMs < 100)
            pauseMs = 100; % pause at least 100 ms
        end;

        % Show first frame in movie
        Screen('DrawTexture', winExperiment, movieTex(1));
        vbl = Screen('Flip', winExperiment, vbl + flipInterval/2, 1);
        % Pause for some configurable time, at a multiple of
        % flipInterval.
        vbl = vbl + round(double(pauseMs * 0.001) / flipInterval) * flipInterval;
        %vbl = WaitSecs('UntilTime', vbl);

        % Start audio playback for one repetition of the sound data,
        % start it at vbl and don't wait for the playback to start,
        % return onset timestamp.
        PsychPortAudio('Start', pahandle, 1, vbl, 0);

        ix = 0;
        for frame = 1:toFrameNo
            ix = ix + 1;

```

```

    Screen('DrawTexture', winExperiment, movieTex(frame));
    % "when" specifies when to flip: If set to zero (default),
    % it will flip on the next possible retrace. If set to a
    % value when > 0, it will flip at the first retrace after
    % system time 'when' has been reached.
    vbl = vbl - flipInterval/2;
    % dontClear=2 is to:
    % prevent PTB3 from clearing the framebuffer to background
    % color after each flip. This will increase performance!
    vbl = Screen('Flip', winExperiment, vbl, 2);
    vbl = vbl + vblPerFrame*flipInterval;
    vbls(ix) = vbl;
end
else
    vbl = PsychPortAudio('Start', pahandle, 1, 0, 0);
end;

% Stop playback
waitForAudioEnd = 1;
PsychPortAudio('Stop', pahandle, waitForAudioEnd);

% short break between clips
if ~NO_VIDEO
    Screen('DrawTexture', winExperiment, respond_tex);
    % dontClear=0, which will clear the framebuffer to background
    % color after each flip. This will prevent the buffer to be in
    % undefined state during start of next playback.
    vbl = Screen('Flip', winExperiment, 0, 0);

    % check for sync errors
    interval = diff(vbls);
    %disp(interval);
    interval = round(interval/flipInterval);
    sync_errors = sum(interval-1);
    if (sync_errors)
        log_text(sprintf('Warning! Sync errors (%d) with VBL.', sync_errors));
    end

    if DEBUG
        figure('Name','sync_err');
        plot(sync_errors);
    end
end
end

function pause_experiment()
    h = warndlg('Press OK to continue','!! PAUSE !!');
    set(h, 'Position', [200 600 200 75]);
    uiwait(h);
    drawnow();
end
end

% RMS normalization allows players to normalize loudness for individual
% wave signal data.
% This avoids the common problem of having to manually adjust volume
% levels when playing audio files that have been mastered at different
% loudnesses.
% autoData contains sample data, where each cell is the sample data.
function y = rms_normalization(audioData)

    %% initialize

```

```

y1 = cell(1, numel(audioData));
y = cell(1, numel(audioData));

% rms for each wave file.
numFiles = numel(audioData);
rms = zeros(1, numFiles);
max_sample = zeros(1, numFiles);

%% calculate the input energy for the audio data
for i = 1:numFiles
    rms(i) = sqrt(sum(audioData{i}.^2)/numel(audioData{i}));
end

max_rms = max(abs(rms));

for i = 1:numFiles
    factor = max_rms/rms(i);
    y1{i} = audioData{i} .* factor;
    max_sample(i) = max(abs(y1{i}));
end

peak_sample = max(max_sample);
%% normalise so that no overflow occurs.
% each sample must be <= 1.0
for i = 1:numFiles
    y1{i} = y1{i}./peak_sample;
end
end

% This function mixes noise with original audio according to the
% specified SNR (signal-to-noise ratio).
%  $SNR = P_{signal}/P_{noise} = (A_{signal}/A_{noise})^2$ 
% Read more at http://en.wikipedia.org/wiki/Signal-to-noise\_ratio
% audioData contains sample data, where each cell is the sample data.
% noiseData contains noise data (directly, no cell).
function y = mix_noise(audioData, noiseData, snr)

    %% initialize
    y = cell(1, numel(audioData));

    numFiles = numel(audioData);
    rmsAudio = zeros(1, numFiles);
    rmsNoise = zeros(1, numFiles);

    %% calculate the input energy for the audio data
    for i = 1:numFiles
        numSamples = length(audioData{i});
        % Calculate the rms of the real audio signal
        rmsAudio(i) = sqrt(sum(audioData{i}.^2)/numSamples);
        % Calculate the rms of the noise signal
        rmsNoise(i) = sqrt(sum(noiseData(1:numSamples).^2)/numSamples);
    end

    %c1 = 10^(double(snr)/20);

    for i = 1:numFiles
        c1 = 10^(double(snr(i))/20);

        numSamples = length(audioData{i});
        % the desirable level of noise to create the specified SNR
        wantedRmsNoise = rmsAudio(i) / c1;
        noiseModifier = wantedRmsNoise / rmsNoise(i);
    end

```

```
% re-scale the noise according to the specified SNR
scaledNoise = noiseModifier .* noiseData(1:numSamples);
% mix the speech signal with the noise
y{i} = audioData{i} + scaledNoise;
end
end
```