Journal of
open research software

Hansen, M 2016 PyVDT: A PsychoPy-Based Visual Sequence Detection Task. *Journal of Open Research Software,* 4: e22, DOI: http://dx.doi.org/10.5334/jors.117

SOFTWARE METAPAPER

PyVDT: A PsychoPy-Based Visual Sequence Detection Task

Mads Hansen

Developer; Department of Psychology and Behavioural Sciences, Aarhus University, Denmark
d@taba.se

PyVDT is a computerized test consisting of two brief visual sequence detection tasks in which participants watch single digits displayed on screen and respond whenever target digit sequences (even – odd – even) are displayed. The total duration of the test is around five minutes. PyVDT is a reimplementation of the Visual Monitoring Task (VMT), a task thought to measure working memory.

PyVDT uses the PsychoPy API to display digits, to plot diagnostic information, and to output log files and results. It is available for download on Figshare and GitHub. PyVDT is free software and has minimal software and hardware requirements. Thus, PyVDT provides a readily available visual monitoring task for use in experiments within cognitive science and related fields.

Keywords: working memory; visual working memory; visuospatial working memory; visual detection; visual monitoring task; psychopy; python
Funding statement: The software did not result from funded research.

(1) Overview

Introduction

Working memory is a theoretical concept originating in the field of cognitive psychology [1, 2]. Baddeley [3] describes working memory as "a limited capacity system, which temporarily maintains and stores information" (p. 829). According to Baddeley (p. 829), working memory "supports human thought processes by providing an interface between perception, long-term memory and action." Based on this description, one would expect working memory to play a crucial role in the performance of many cognitive tasks. Indeed, this appears to be the case when reviewing the working memory literature. These studies contain a multitude of findings showing that measures of working memory correlate with many different cognitive abilities, ranging from general intelligence to reading ability (see [4] for a review).

Baddeley's [5] model of working memory is split into components corresponding to different sensory modalities. For example, Baddeley conceptualizes auditory and visuospatial working memory as discrete, yet related, components or abilities. This distinction is central to the current article, since visual monitoring tasks, such as the PyVDT described here, are thought to measure visual working memory [6].

Visual monitoring tasks have been used in studies of hearing impairment (see [7] for a review). Knutson et al. [8] appear to have been the first to use this type of monitoring task to study cognitive factors assumed to be predictive of speech perception following cochlear implantation (i.e., the surgical insertion of an electrode into the cochlea which allows profoundly deaf individuals to regain the sense of hearing). Knutson et al. used the Visual Monitoring Task (VMT), a test that requires participants to "watch the computer monitor as single-digit numbers are presented one at a time" (p. 819) and to "respond when those digits [...] produce the specified pattern" (p. 819), an even-odd-even pattern of digits (e.g., "2–5–4").

Knutson et al. [8] found that VMT scores predicted post-implant audiological outcomes (i.e., measures of speech perception). Others (e.g., [9]) have since reported similar associations between speech reception in noise and visual monitoring scores. Lunner and Sundewall-Thorén used a translated version of the Visual Letter Monitoring test (VLM) originally developed by the MRC Institute of Hearing Research [10] – a test which is similar in principle to the VMT used by Knutson et al. Instead of digits, however, the VLM uses three-letter words (e.g., P–E–N) as target sequences.

Akeroyd [7] and Gfeller et al. [11] view the VMT as a measure of attention, reaction time, and working memory. In line with this, Knutson et al. [8] state that the VMT requires the participant to "maintain the last two digits in working memory" (p. 819) and to "update the two digits in working memory" (p. 819). This notion of updating is

echoed by Akeroyd [7] who describes the VMT as requiring "continuous performance" (p. S58). More recently, Knutson [6] highlighted the "probable importance of working memory in the VMT" (p. 156) and stated that the test includes "a memory component" (p. 154). Also, since these monitoring tasks display stimuli that can be verbalized (i.e., digits and letters), they likely also reflect the ability of participants to overtly or covertly verbalize stimuli.

The association between working memory and speech perception first reported by Knutson et al. [8] is central to the Ease of Language Understanding (ELU) model [12]. Rönnberg et al. [12] theorize that working memory is beneficial to speech perception in difficult listening conditions (i.e., background noise), since – according to the ELU model – working memory helps maintain relevant information while inhibiting irrelevant information. The software described here allows researchers to test theories such as the ELU model and to attempt to replicate findings such as those reviewed in the preceeding paragraphs.

Traditionally, researchers wanting to investigate the concept of working memory have had to purchase standardized tests or create their own. The former often ensures greater reliability and validity, but can be expensive; the latter is often cheaper, but less reliable and valid.

The PyVDT represents a first step toward providing a reliable and valid visual monitoring task as free software. Since the PyVDT very closely mimics the original Visual Monitoring Task [8], the PyVDT can be assumed to be largely as reliable and valid as the VMT. In any case, since the PyVDT code is freely available, anyone can scrutinize the code, verify the reliability and validity of the test, and carry out independent replication studies. Additionally, bug reports, patches, forks and suggestions for improvement are encouraged.

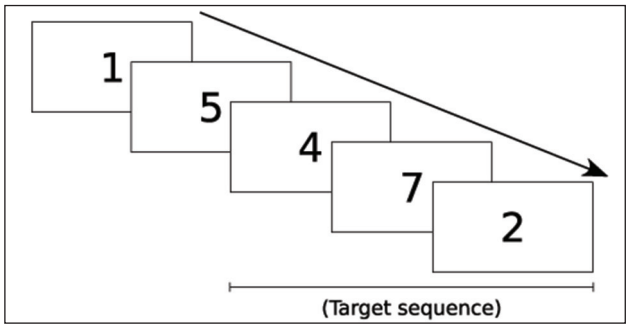

Figure 1: Schematic representation of the presentation of digits during the PyVDT. The software shows single digits, one at a time, on screen. Digits change according to the presentation rate. The three rightmost (i.e., most recently presented) digits make up a target sequence (i.e., any sequence beginning with an even digit, followed by an odd digit, and ending with an even digit). Pressing the space bar while the final digit in a target sequence (here, the number two) is displayed would count as a "hit", whereas failing to press the space bar in this case would count as a "miss". Pressing the space bar at the end of a non-target sequence (e.g., the three leftmost digits) would count as a "false alarm", whereas failing to press the space bar would count as a "correct rejection". See Macmillan and Creelman [14] for a detailed discussion of these four response types.

Recently, Stone & Towse [13] made a Java-based working memory test battery available as free software. Stone and Towse's battery consists of seven span tests: digit, matrix, arrow, reading, operation, rotation, and symmetry span. The PyVDT adds to this work by providing an additional, complementary measure of working memory.

Within hearing aid and cochlear implant research, the majority of studies investigating the relationship between monitoring scores and speech perception have used either the Visual Monitoring Task [8] or the MRC Visual Letter Monitoring task [10]. Neither of these tests are available as free software, however. It is not straightforward, therefore, for researchers to inspect the source code of these tests or to attempt to replicate published findings. The PyVDT, again, aims to resolve these issues.

Like the VMT, the PyVDT displays single digits one at a time. Participants are instructed to press the space bar whenever the three most recently displayed digits match the pattern "even digit – odd digit – even digit" (i.e., the target sequence; see **Fig. 1**).

The PyVDT consists of two subtests. The default configuration displays one digit every two seconds during the first subtest, and one digit every second during the second subtest. Thus, this configuration is identical to the one used by Knutson et al. [8]. The digit presentation rates can be adjusted using the configuration dialog (**Fig. 2**). The duration of each subtest is two minutes and 32 seconds. Participant performance is measured using d' ("dee-prime"), a measure of sensitivity originating in signal detection theory (or simply "detection theory") [14].

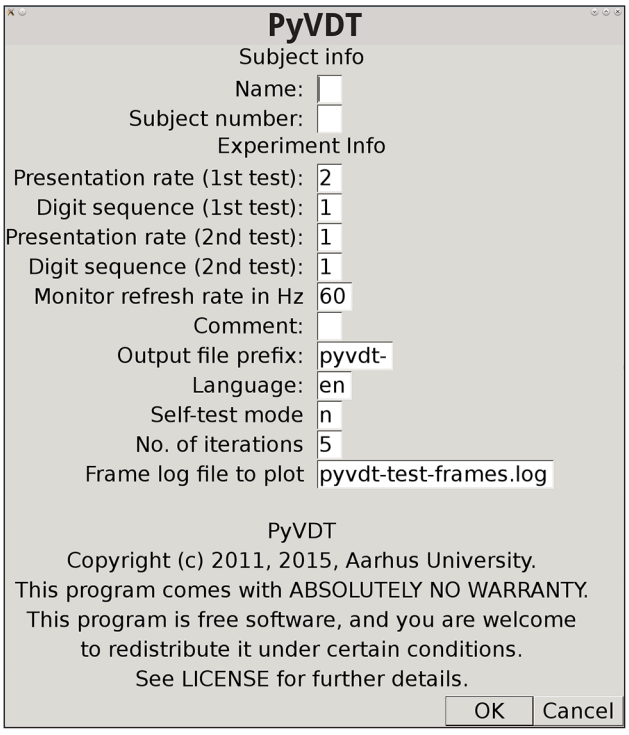

Figure 2: PyVDT configuration dialog.

Implementation and architecture

PyVDT is based on the PsychoPy software suite [15, 16] which facilitates the development of computer-based experiments by providing an application programming interface (API) for presenting auditory and visual stimuli, among other things. PyVDT leverages the PsychoPy API to display stimuli and to log detailed diagnostic data about each test run (e.g., timestamped information about keypresses and dropped frames). Most importantly, PsychoPy's `TextStim` function is used to display digits, and its `logging` function is used to save diagnostics to log files. These log files (named `<prefix>-<subject number><subject name>.log`) can be used for general troubleshooting or to evaluate the performance of the graphics processing unit (GPU) installed in the computer running PyVDT, in order to verify that stimuli are presented at the correct rate. The frame log files generated by PyVDT (named `*-frames.log`) contain a comma-separated list of the durations in milliseconds of all presented frames. These frame log files can be used to identify "dropped" frames (see "Quality control" section below).

PyVDT comes with a number of pre-defined (pseudo-random) digit sequences, each containing the same amount of targets. The targets make up roughly one quarter of the total number of stimuli presented. The digit sequences are defined in the two .csv files provided with the software (`pyvdtSequences-rate1.csv` and `pyvdtSequences-rate2.csv`).

Instructions for use

In order to run the PyVDT, two software packages have to be installed: PsychoPy and PyVDT itself. After installation, the PyVDT can be launched by opening the file `pyvdt.py` via PsychoPy Coder and clicking the green "run" button. This brings up the configuration dialog (**Fig. 2**) which allows the user to input participant names and numbers, digit presentation rates, prefixes for the output files, monitor refresh rates, etc. After clicking "OK", the program switches to full-screen mode. That is, the normal GUI interface of the operating system (e.g., icons, taskbars, etc.) is replaced with a white background taking up the entire screen, and a welcome message is displayed. After pressing a key on the keyboard, a fixation dot is briefly shown in the center of the screen. Immediately afterwards, the first digit is displayed and the test is running. Following the first test, a message is shown, asking the participant to press any key to continue with the second test. After the second test ends, a "thank you" message is shown. The messages shown in full-screen mode can be customized by editing the file `pyvdt.ini`.

PyVDT saves output data to .csv files that can be imported into a statistical software package for further analysis. The .csv files named `<prefix>-data-{1,2}.csv` (one file per subtest) contain the main output variables of interest (e.g., d'). These output files contain data from multiple PyVDT runs, provided that these runs use the same output file prefix (defined in the configuration dialog; see **Figure 2**). This allows experimenters to save the results of different experiments in separate .csv files (e.g., a pilot study might use the prefix `pilot`, the `main` study might use main, etc.).

Aside from the main output .csv files, two additional .csv files are generated per run. These files (named `<prefix>-<timestamp>-{1,2}.csv`) contain raw, mostly unprocessed data relating to the presentation of stimuli and the calculation of the sensitivity measures. These .csv files are primarily provided for diagnostic purposes. In most cases, experimenters can safely ignore these particular files.

Sample output .csv files are included with PyVDT. The different variables saved in these .csv files are described in the comments found in the file `csvfunc.py`.

Quality control

The PyVDT has been successfully used in a study of 25 cochlear implant users [17]. Additionally, PyVDT uses the Python `doctest` module [18] to verify that the measures reported by the PyVDT (e.g., the d' measure of sensitivity; [14]) are calculated correctly. To see the results of these doctests, run Python with verbose output enabled (e.g., `python pyvdt.py -v`). The formulae used to calculate the sensitivity measures are those reported by Macmillan and Creelman [14] (p. 21).

The PyVDT also includes two self-tests that are accessible via the configuration dialog (see the text box labeled "self-test mode" in **Figure 2**). The first self-test (enabled by entering `y` in the self-test text box) allows users to simulate participant responses by generating sets of random responses and corresponding output data, while the second self-test (enabled by entering `p` in the self-test text box) plots frame durations from previous PyVDT runs. Sample output data can be generated by running the former self-test. Sample digit sequences are provided in two .csv files (`pyvdtSequences-rate1.csv` and `pyvdtSequences-rate2.csv`).

Running these self-tests before using the PyVDT in an experimental study is highly recommended in order to verify that testing conditions are similar for all participants and that results are reliable. Ideally, digit presentation rates should be constant across experiments. Therefore, it is important to ensure that the computer used to run the PyVDT is performing adequately. Computer monitors are updated a large number of times per second (as indicated by the monitor's refresh rate in Hz). These rapid screen updates are known as frames. If the computer hardware is unable to present the stimuli at the desired rate, frames may be discarded, or "dropped". Thus, dropped frames often signify performance problems. The second self-test (see above) allows users to identify performance problems; dropped frames show up as spikes on the plot.

The software has been tested and found to work on Windows (8.1), Mac (Snow Leopard), and Linux (Debian Jessie) platforms. Bugs can be reported to the author via the "Issues" function on GitHub (https://github.com/criticalmads/pyvdt/issues).

(2) Availability

Operating system
Linux, Windows, Mac.

Programming language
Python.

Additional system requirements
PyVDT was designed to run on laptops. The hardware requirements are modest. No unusual equipment is needed. The software has been extensively tested on a 2009 13-inch MacBook Pro (model identifier "MacBookPro5,5"; 2.26 GHz Intel Core 2 Duo processor, 2 GB DDR3 RAM) and should run smoothly on desktop PCs and laptops with dedicated GPUs supporting OpenGL 2.0. When using slower hardware, however, it is advisable to verify (by examining the log files or by plotting frames via the built-in self-test) that no frames are dropped or delayed significantly during testing. PyVDT itself takes up less than 1 megabyte of disk space, while PsychoPy requires a few hundred megabytes of disk space, depending on the platform.

Dependencies
PsychoPy version 1.73 or higher (http://www.psychopy.org).

List of contributors
Mads Hansen, developer, Department of Psychology and Behavioural Sciences, Aarhus University, Denmark.

Software location:
Archive

Name: Figshare

Persistent identifier: https://dx.doi.org/10.6084/m9.figshare.1583394.v3

Licence: GPLv3

Publisher: Mads Hansen

Date published: 23/10/15 (v1); 27/01/16 (v2); 05/02/16 (v3)

Code repository
Name: GitHub

Identifier: https://github.com/criticalmads/pyvdt

Licence: GPLv3

Date published: 23/10/15

Language
English. PyVDT comes with user-selectable English and Danish participant instructions to be shown on screen. Additional languages can be defined in the configuration file (`pyvdt.ini`).

(3) Reuse potential
The PyVDT is well-suited for experimental studies of working memory within psychology, neuroscience, cognitive science, and related fields. The software could easily be modified or extended to use letters as stimuli (like the MRC letter monitoring task [10]). Also, the software could be modified to use adaptive presentation rates rather than the fixed rates currently used.

Technical support is provided by the author via GitHub.

Acknowledgements
The author would like to thank Eva Bang-Olsen, Senior Legal Adviser, Corporate Relations and Technology Transfer, Aarhus University, for legal advice regarding licensing of the PyVDT software. Additionally, the author would like to thank John F. Knutson for providing technical information about the original Visual Monitoring Task. Lastly, the author would like to thank Leif Østergaard and Kristjar Skajaa, CFIN, Aarhus University, who generously gave permission to release the PyVDT code as free software.

Competing Interests
The author declares that they have no competing interests.

References
1. **Miller, G A, Galanter, E** and **Pribram, K H** 1960. Plans and the structure of behavior. Holt, New York. DOI: http://dx.doi.org/10.1037/10039-000
2. **Baddeley, A D** and **Hitch, G** 1974. Working Memory, in: Bower, G H (Ed.), *Psychology of Learning and Motivation*, Academic Press, pp. 47–89. DOI: http://dx.doi.org/10.1016/s0079-7421(08)60452-1
3. **Baddeley, A** 2003. Working memory: looking back and looking forward. *Nat Rev Neurosci,* 4, pp. 829–839. DOI: http://dx.doi.org/10.1038/nrn1201
4. **Baddeley, A** 2007. Working Memory, Thought, and Action. Oxford University Press, New York. DOI: http://dx.doi.org/10.1093/acprof:oso/9780198528012.001.0001
5. **Baddeley, A** 2012. Working Memory: Theories, Models, and Controversies. Annual Review of Psychology 63, pp. 1–29. DOI: http://dx.doi.org/10.1146/annurev-psych-120710-100422
6. **Knutson, J F** 2006. Psychological aspects of cochlear implantation, in: Cochlear Implants: A Practical Guide. Whurr Publishers Limited, London, pp. 151–178.
7. **Akeroyd, M A** 2008. Are individual differences in speech reception related to individual differences in cognitive ability? A survey of twenty experimental studies with normal and hearing-impaired adults. *Int J Audiol,* 47, pp. S53–S71. DOI: http://dx.doi.org/10.1080/14992020802301142
8. **Knutson, J F, Hinrichs, J V, Tyler, R S, Gantz, B J, Schartz, H A** and **Woodworth, G** 1991. Psychological predictors of audiological outcomes of multichannel cochlear implants: preliminary findings. *Ann. Otol. Rhinol. Laryngol.* 100, pp. 817–822. DOI: http://dx.doi.org/10.1177/000348949110001006
9. **Lunner, T** and **Sundewall-Thorén, E** 2007. Interactions between cognition, compression, and listening conditions: Effects on speech-in-noise performance in a two-channel hearing aid. *Journal of the American Academy of Audiology* 18, pp. 604–617. DOI: http://dx.doi.org/10.3766/jaaa.18.7.7
10. **Gatehouse, S, Naylor, G** and **Elberling, C** 2003. Benefits from hearing aids in relation to the interaction between the user and the environment.

Int J Audiol 42, pp. 77–85. DOI: http://dx.doi.org/10.3109/14992020309074627

11. **Gfeller, K, Christ, A, Knutson, J F, Witt, S, Murray, K T** and **Tyler, R S** 2000. Musical backgrounds, listening habits, and aesthetic enjoyment of adult cochlear implant recipients. *J Am Acad Audiol 11*, pp. 390–406.
12. **Rönnberg, J, Lunner, T, Zekveld, A, Sorqvist, P, Danielsson, H, Lyxell, B** and **Rudner, M** 2013. The Ease of Language Understanding (ELU) model: theoretical, empirical, and clinical advances. *Front. Syst. Neurosci* 7, 31. DOI: http://dx.doi.org/10.3389/fnsys.2013.00031
13. **Stone, J** and **Towse, J** 2015. A Working Memory Test Battery: Java-Based Collection of Seven Working Memory Tasks. *Journal of Open Research Software 3*. DOI: http://dx.doi.org/10.5334/jors.br
14. **Macmillan, N A** and **Creelman, C D** 2004. Detection Theory: A User's Guide, 2 edition. ed. Psychology Press, Mahwah, N.J.
15. **Peirce, J W** 2007. PsychoPy—Psychophysics software in Python. *Journal of Neuroscience Methods* 162, pp. 8–13. DOI: http://dx.doi.org/10.1016/j.jneumeth.2006.11.017
16. **Peirce, J W** 2009. Generating stimuli for neuroscience using PsychoPy. Front. Neuroinform. 2, 10. DOI: http://dx.doi.org/10.3389/neuro.11.010.2008
17. **Hansen, M, Petersen, B, Ovesen, T** and **Bærentsen, K B** 2015. Postoperative Predictors of Speech Reception in Cochlear Implant Users. [Unpublished manuscript].
18. **Martelli, A** 2006. Python in a Nutshell, Second Edition, O'Reilly Media, Beijing ; Sebastopol, CA.

How to cite this article: Hansen, M 2016 PyVDT: A PsychoPy-Based Visual Sequence Detection Task. *Journal of Open Research Software,* 4: e22, DOI: http://dx.doi.org/10.5334/jors.117

Submitted: 05 February 2016 **Aceepted:** 02 June 2016 **Published:** 10 June 2016