The Journal of Open Source Software

stimupy: A Python package for creating stimuli in vision science

Lynn Schmittwilken[1,2¶], **Marianne Maertens**[1,2], **and Joris Vincent**[2]

1 Science of Intelligence, Technische Universität Berlin, Germany **2** Computational Psychology, Technische Universität Berlin, Germany ¶ Corresponding author

DOI: 10.21105/joss.05321

Software

- Review
- Repository
- Archive

Editor: Elizabeth DuPre

Reviewers:

- @alexander-pastukhov
- @JonathanReardon

Submitted: 24 March 2023
Published: 09 June 2023

Summary

Visual stimuli are at the heart of perception research. They may come as visual illusions which demonstrate the striking differences between the perceptual and physical world, they may involve minuscule stimulus changes which are used to probe the limits of visual sensitivity, or they may be used to probe any other aspect of visual processing. `stimupy` is a free and open-source Python package which allows the user to create visual stimuli of different complexity as they are commonly used in the study of visual perception (Figure 1).

`stimupy` provides functions to generate:

- basic components, including shapes, lines, gratings, checkerboards, and Gaussians
- different types of visual noise textures
- visual stimuli such as Gabors, plaids, edges, and a variety of so-called illusions (e.g., Simultaneous Brightness Contrast, White's illusion, Hermann grid, Ponzo illusion), and many more
- stimulus sets from prior research papers, providing exact stimulus recreations (e.g., ModelFest, Carney et al. (1999))
- utility functions for stimulus import, export, manipulation (e.g., contrast, size), or plotting
- documentation, including interactive demonstrations of stimulus functions
- unit and integration tests

`stimupy` has been designed to:

- generate (novel) visual stimuli in a reproducible, flexible, and easy way
- recreate exact stimuli as they have been used in prior vision research
- explore large parameter spaces to reveal relations between formerly unconnected stimuli
- provide classic stimulus sets (e.g., ModelFest), exactly as described in the original manuscripts (including experimental data)
- build new stimulus sets or benchmarks (e.g., for testing computational models), and easily add them to `stimupy`
- support vision science by providing a large, openly-available and flexible battery of relevant stimulus functions
- unify and automate stimulus creation

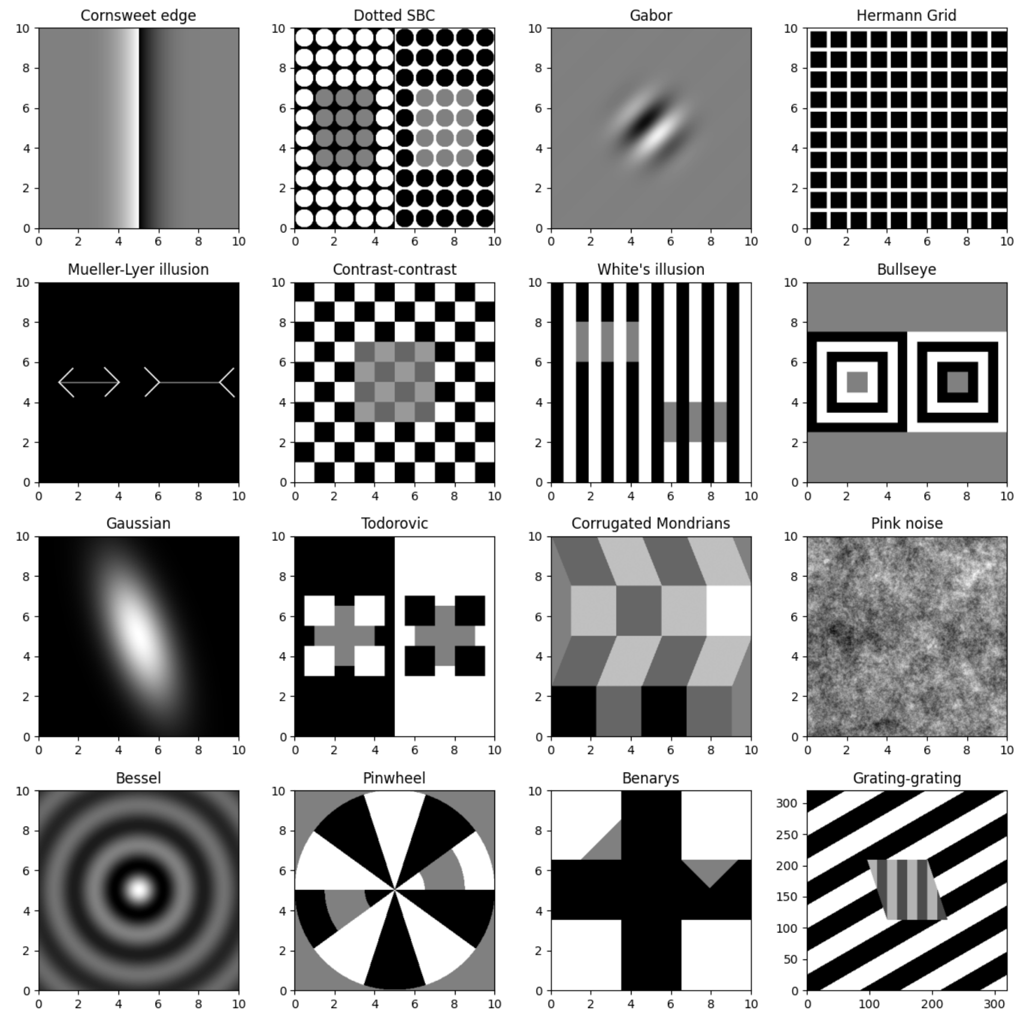

Figure 1: A small fraction of the stimulus variety that `stimupy` can produce

State of the field

Creating visual stimuli is a central task in vision research. To generate stimuli, it is common practice to either write your own stimulus functions from scratch; reuse existing code; or import a static stimulus version from an image or data file (see e.g., Carney et al. (1999), Murray (2020)). The alternative to these idiosyncratic approaches is to use existing software which provides more flexible stimulus functions.

We are currently aware of

- Psychtoolbox (Brainard, 1997),
- Psychopy (Peirce et al., 2019),
- Pyllusion (Makowski et al., 2021),
- OCTA (Van Geert et al., 2022).

Psychtoolbox and Psychopy both provide functions to generate a number of visual stimuli. However, stimulus generation is integrated into their main purpose which is to run psychophysical experiments. The design focus of both Psychtoolbox and Psychopy has therefore been to support the user to interface between computer hardware and MATLAB and Python, respectively, to enable temporal precision and high dynamic range stimulus delivery.

The design focus of stimupy is on stimulus creation. This allows stimupy to include many more stimuli than included in Psychtoolbox or Psychopy. It also allows the user to interact with the stimulus arrays directly. This makes it easy to manipulate the stimulus and use it for other purposes than psychophysical experimentation (e.g., computational modeling, visualization). This also means that in order to present the stimuli on a computer monitor, the user may still want to use Psychopy, Psychtoolbox or another delivery system for hardware control.

Pyllusion is a Python package to generate a number of well-known illusions such as the Müller-Lyer, Ponzo or Zöllner illusions, and more. Pyllusion provides functions for each of these illusions using high-level parameters (e.g., illusion strength). The parametric approach of Pyllusion is similar in spirit to stimupy. However, in Pyllusion each illusion-function stands alone: it produces only that stimulus, and its arguments are unique to that stimulus. In contrast, stimupy provides a unified interface to stimulus creation, where many functions share the same —intuitive— parameters. This makes it easier to explore parameters and to create novel stimuli.

OCTA is also a Python package to generate stimuli, specifically grids of multiple elements that show regularity and variety along various stimulus dimensions. These stimuli are of particular use to studies on Gestalt vision, aesthetics and complexity. The parametric variation of stimulus dimensions as well as the compositionality of displays are features found in both OCTA and stimupy. Both packages also have a strong focus on ease-of-use, replicability, and open science. stimupy currently focuses on a different class of stimuli: mainly displays used to study early and low-level visual processes, as well as visual features such as brightness, contrast, and orientation. Thus, OCTA and stimupy cover complementary use cases.

Another design decision that sets stimupy apart from existing software such as OCTA and Pyllusion, is that all stimupy stimuli are generated as NumPy-arrays representing pixel-based raster-graphics (NumPy, Harris et al. (2020)). This has several advantages over existing, vector-graphics or custom object-based approaches, mainly that any standard array-manipulation tooling can be used to further process a stimulus.

Statement of Need

Many visual stimuli are used time and again. Despite their relevance, there is no standard way of implementing stimuli which considers function parameters in a way that is targeted towards vision science. Hence, in practice, researchers implement their own stimuli from scratch or are lucky enough to find some existing implementation online, from colleagues or in the above mentioned software packages. Depending on the complexity or specificity of the desired stimulus manipulation, this endeavor is (1) time-consuming, (2) prone to error, and (3) makes comparisons with other research difficult. Hence, we developed stimupy to simplify, unify and automate visual stimulus generation while at the same time allowing the flexibility to create entirely new stimuli and build stimulus benchmarks.

As far as we know stimupy is the only package that:

- contains a wide variety of visual stimuli, from simple geometric shapes to complex illusions
- includes ready-to-use replications of existing stimulus sets (e.g., ModelFest)
- makes it easy to create new stimuli because (1) stimulus functions use parameters which are familiar to vision scientists, and (2) it provides building blocks and masks which can be used to assemble more complicated geometries
- uses flexible output structures (NumPy arrays, and Python dictionaries) and hence makes it easy to interact with the stimulus arrays and store additional information (e.g., stimulus descriptions, stimulus masks, experimental data)
- is modular and therefore easy to extend with new stimulus functions, and new stimulus sets

- is hierarchical in a sense that more complex stimulus functions (e.g., visual illusions) use more basic stimulus functions (e.g., components)
- comes with application-oriented documentation, including interactive Jupyter Notebooks (Kluyver et al., 2016)

`stimupy` is a free and open-source Python package which can be easily downloaded and installed via standard package managers, or directly from its GitHub source. We think that using `stimupy` will improve the consistency and accessibility of visual stimuli while helping to avoid bugs. A key feature in `stimupy` is that its functions are parameterized with parameters that are relevant to vision scientists (e.g., visual angle, spatial frequency, target placements). Moreover, `stimupy` is designed in a modular fashion, i.e. more complex stimuli are composed of less complex stimuli, which supports the understanding of existing stimuli, makes connections between stimuli explicit, and facilitates the creation of novel stimuli. The output of all stimulus functions is a dictionary which contains the stimulus-image as a NumPy-array together with other useful stimulus information (e.g., masks, stimulus parameters, and experimental data). Having the stimulus-image as a NumPy-array makes it easy to work and interact with the stimulus, e.g., using common NumPy tooling and/or utility functions provided by `stimupy`. This is useful for manipulating the stimulus as well as for using the stimulus for other purposes than psychophysical experimentation on a computer screen (e.g., for visualizations or for computational modeling). The main advantage of using dictionaries as function outputs is that Python dictionaries are mutable data structures which allow you to add additional information easily. Moreover, the stimulus dictionaries contain all parameter information relevant for describing a stimulus (e.g., size, resolution, spatial frequency, orientation, phase, etc), and hence help to report stimulus parameters, e.g. in a research paper. `stimupy` also provides utility-functions to strip the stimulus dictionaries down to just the raw parameters that the original stimulus function expects. Hence, it is sufficient to share only the stimulus dictionaries to recreate exact stimulus replications, which makes it easy to create slight variations of this stimulus. Taken together, these design choices make `stimupy` a flexible and versatile Python package which facilitates the (re)creation and use of visual stimuli for a variety of purposes.

Another important use case for `stimupy` is the evaluation of computational vision models. A common strategy to validate computational vision models is to test them with benchmark datasets; e.g. in spatial vision (Carney et al., 1999), lightness perception (Murray, 2021), object recognition (Deng et al., 2009), or object segmentation (Martin et al., 2001). However, visual stimuli from prior research are not always publicly available and it is thus difficult and time-consuming to test model performance on stimuli from prior research. On top of that, creating and agreeing on benchmark datasets is a challenging task. Hence, to support the accessibility of previously used stimuli and encourage the creation of stimulus benchmarks, `stimupy` provides a collection of existing stimulus sets (including ModelFest) as they have been used in the original manuscripts. Due to `stimupy`'s versatility, entire stimulus sets (including experimental findings) can be accessed via a single line of code, and more stimulus sets can be added at any point in time.

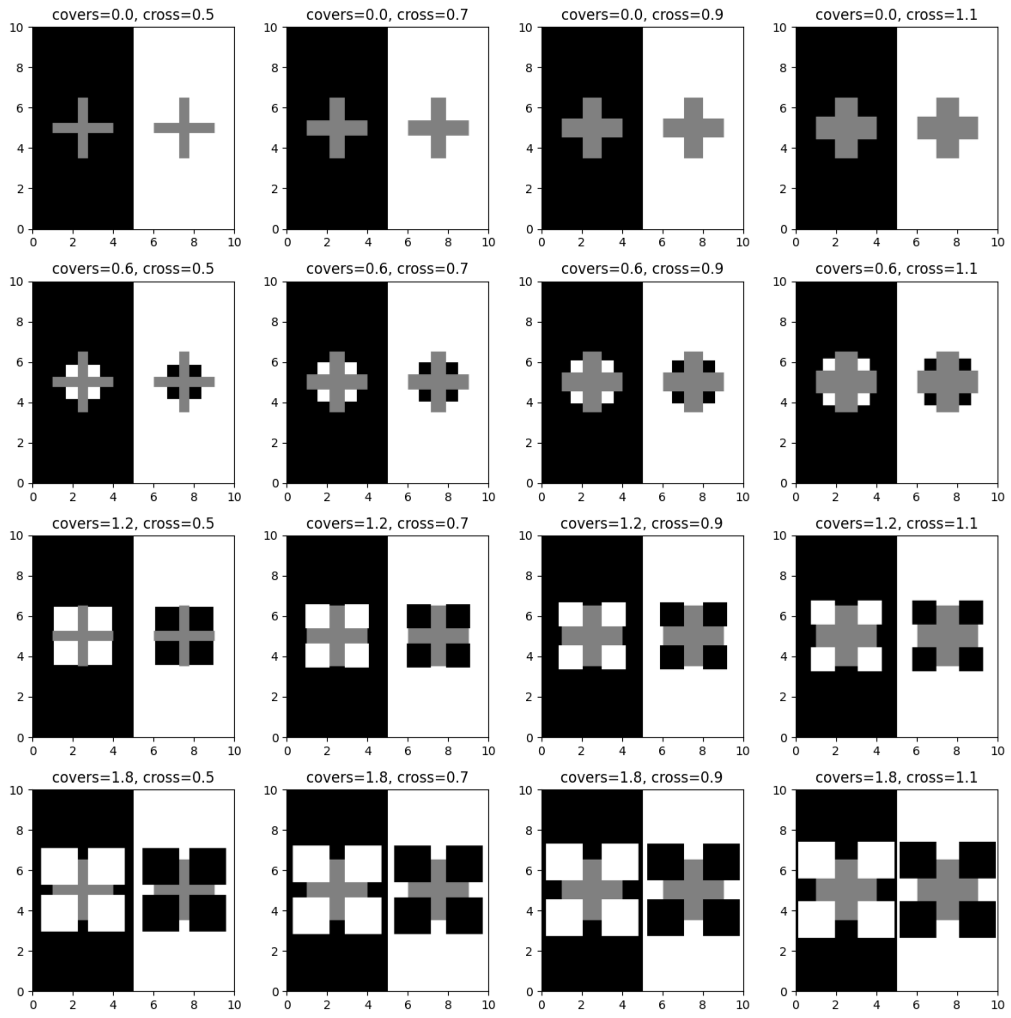

Figure 2: Samples from a parameter space of a single `stimupy` stimulus function:

`stimupy`'s high degree of parameterizability allows for extensive explorations of stimulus parameter spaces (Figure 2). On the one hand, this can be useful for vision experimentation because varying stimuli along one or multiple dimensions can be directly mapped to certain experimental designs and research questions. On the other hand, this feature can also guide theoretical work because among other things it allows to find so-called maximally-differentiable stimuli (Wang & Simoncelli, 2008). The basic idea of maximum differentiation is to analyze model predictions for systematically varied stimuli to find the ones which differentiate best (maximally) between models. Like this, the number of stimuli tested experimentally can be reduced to the most relevant stimuli. Since collecting (human) data is resourceful, maximal differentiation is a useful method to reduce theoretically large stimulus parameter spaces to a testable number of stimuli.

Last but not least, `stimupy` can be a useful aid in teaching contexts because it provides students with a basic framework in which they can design and interact with stimuli in a playful way. Since `stimupy` is focused on stimulus creation rather than stimulus presentation, a user can quickly generate complex and innovative stimuli – even with beginner knowledge of Python. The parameterized functions and the interactive documentation allow for easy teaching and communication of how various stimulus parameters affect perception.

Projects Using the Software

As stimulus creation is relevant for many vision science projects, stimulus functions which are part of `stimupy` or a pre-release version of the software have been used in almost all of the work of our laboratory within the last two years. Some of `stimupy`'s noise functions have been used to generate the narrowband noise masks of varying center frequency in (Schmittwilken & Maertens, 2022b). A pre-release version was used in multiple conference contributions in which we compared structural elements between existing models of brightness perception (Vincent & Maertens, 2021a); in which we compared existing models of human brightness perception on a large battery of brightness stimuli (Schmittwilken et al., 2022); in which we evaluated how to quantitatively link output of computational models to human brightness perception data (Vincent & Maertens, 2021b); in which we demonstrate that a family of computational models fails to account for novel brightenss perception data (Aguilar et al., 2022; Vincent et al., 2022b, 2022a) and in which we studied human edge processing with Cornsweet stimuli in various kinds of noise (white, pink, brown, several narrowband noises) (Schmittwilken & Maertens, 2022a). All these stimuli were created with `stimupy` or functions that are included in the software. Moreover, we are using `stimupy` in ongoing work in our laboratory and in many student projects.

Future Work

In theory, there is an infinite number of stimuli which are or could be interesting in the future of vision research. Hence, `stimupy` will by default remain under active development. In future versions, we want to add more visual stimuli and more stimulus sets – particularly dynamic stimuli which are currently not included. Finally, we want to foster the development of stimulus benchmarks in vision science which will be added to `stimupy`.

Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy -EXC 2002/1 "Science of Intelligence"- project number 390523135 and individual grants MA 5127/4-1 and 5-1 to M. Maertens

References

Aguilar, G., Maertens, M., & Vincent, J. (2022). Characterizing perceptual brightness scales for White's effect using conjoint measurement. *Journal of Vision*, *22*, 3519. https://doi.org/10.1167/jov.22.14.3519

Brainard, D. H. (1997). The psychophysics toolbox. *Spatial Vision*, *10*(4), 433–436. https://doi.org/10.1163/156856897X00357

Carney, T., Klein, S. A., Tyler, C. W., Silverstein, A. D., Beutter, B., Levi, D., Watson, A. B., Reeves, A. J., Norcia, A. M., Chen, C.-C., & others. (1999). Development of an image/threshold database for designing and testing human vision models. *Human Vision and Electronic Imaging IV*, *3644*, 542–551. https://doi.org/10.1117/12.348473

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant,

T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), *Positioning and power in academic publishing: Players, agents and agendas* (pp. 87–90). IOS Press. https://doi.org/10.3233/978-1-61499-649-1-87

Makowski, D., Lau, Z. J., Pham, T., Paul B., W., & Annabel C., S. (2021). A parametric framework to generate visual illusions using python. *Perception*, *50*(11), 950–965. https://doi.org/10.1177/03010066211057347

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *8th IEEE Internatinonal Conference on Computer Vision*, *2*, 416–423. https://doi.org/10.1109/ICCV.2001.937655

Murray, R. F. (2020). A model of lightness perception guided by probabilistic assumptions about lighting and reflectance. *Journal of Vision*, *20*(7), 28. https://doi.org/10.1167/jov.20.7.28

Murray, R. F. (2021). Lightness perception in complex scenes. *Annual Review of Vision Science*, *7*. https://doi.org/10.1146/annurev-vision-093019-115159

Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Hoechenberger, R., Sogo, H., Kastman, E., & Lindelov, J. K. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods*, *51*(1), 195–203. https://doi.org/10.3758/s13428-018-01193-y

Schmittwilken, L., & Maertens, M. (2022a). Medium spatial frequencies mask edges most effectively [Poster]. *Journal of Vision*, *22*. https://doi.org/10.1167/jov.22.14.4041

Schmittwilken, L., & Maertens, M. (2022b). Fixational eye movements enable robust edge detection. *Journal of Vision*, *22*(8), 1–12. https://doi.org/10.1167/jov.22.8.5

Schmittwilken, L., Matic, M., Maertens, M., & Vincent, J. (2022). BRENCH: An open-source framework for b(r)enchmarking brightness models [Talk]. *Journal of Vision*, *22*, 36. https://doi.org/10.1167/jov.22.3.36

Van Geert, E., Bossens, C., & Wagemans, J. (2022). The order & complexity toolbox for aesthetics (OCTA): A systematic approach to study the relations between order, complexity, and aesthetic appreciation. *Behavior Research Methods*. https://doi.org/10.3758/s13428-022-01900-w

Vincent, J., & Maertens, M. (2021a). A history and modular future of multiscale spatial filtering models. *Journal of Vision*, *21*, 2824. https://doi.org/10.1167/jov.21.9.2824

Vincent, J., & Maertens, M. (2021b). *The missing linking functions in computational models of brightness perception* [Talk]. OSF. osf.io/9bca7

Vincent, J., Maertens, M., & Aguilar, G. (2022a). Perceptual Brightness Scales for White's Effect Constrain Computational Models of Brightness Perception. *Journal of Vision*, *22*, 4160. https://doi.org/10.1167/jov.22.14.4160

Vincent, J., Maertens, M., & Aguilar, G. (2022b). Perceptual brightness scales in a White's effect stimulus are not captured by multiscale spatial filtering models of brightness perception [Poster]. *Journal of Vision*, *22*, 20. https://doi.org/10.1167/jov.22.3.20

Wang, Z., & Simoncelli, E. P. (2008). Maximum differentiation (MAD) competition: A methodology for comparing computational models of perceptual quantities. *Journal of Vision*, *8*(12), 8–8. https://doi.org/10.1167/8.12.8