The Journal of Open Source Software

CogStim: Reproducible visual stimulus generation for cognitive science, neuroscience, and vision research

Eudald Correig-Fraga[1]

1 Innovamat Education, Sant Cugat del Vallès, Catalonia (Spain)

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

CogStim is an open-source Python library for reproducible, parameterized offline generation of visual stimuli for psychology, neuroscience, and computer vision. It produces PNG/JPEG/SVG assets for common paradigms (two-colour ANS arrays, match-to-sample pairs with optional total-area equalization, single-colour dot arrays, geometric shapes, oriented lines/stripes, and fixation targets). Deterministic seeding and robust geometric routines enforce non-overlap, boundary validity, and equalization to minimise perceptual confounds. A compact Python API and command-line interface make the tool accessible to both programmers and non-programmers, enabling quick creation of controlled stimuli without special setup.

Statement of need

Designing visual stimuli is a routine requirement in psychology, neuroscience, and vision research. It requires precise control over numerosity, size, spacing, color, and layout, alongside reproducible randomization and strict study dependent constraints. When built ad hoc, these demands make stimulus creation tedious and can introduce small inconsistencies that affect comparisons across studies and model evaluations. Researchers therefore benefit from a simple way to generate controlled stimuli offline as standard image assets ready to use wherever needed.

CogStim addresses this need by providing an offline, parameterized generator for a range of paradigms: Approximate Number System (ANS) (Halberda et al., 2008), Match-To-Sample (MTS) (Sella et al., 2013), geometric shapes, oriented lines (Srinivasan, 2021), and fixation targets (Thaler et al., 2013). It produces images in any major image format (e.g. PNG, JPEG, SVG) that can be dropped into experiment builders, web or desktop presentation software, and computer-vision pipelines without heavy setup. A Python API and a command-line interface enable both quick prototyping and large batch generation; deterministic seeds ensure the same configuration yields identical outputs; and built-in algorithms enforce non-overlap and allow total-area equalization when required. In this way the library serves dual needs: controlled stimuli for behavioral and neuro-cognitive experiments, and well-specified synthetic datasets for model development and evaluation.

State of the field

Several established frameworks dominate the landscape of behavioral research: `PsychoPy` (Peirce et al., 2019) is the standard Python library for experiment creation, offering precise timing and a vast array of stimuli; `Psychtoolbox` (Brainard, 1997) provides similar capabilities within the MATLAB/C environment with a focus on low-level hardware control; and `jsPsych` (Leeuw, 2015) has become the de facto standard for web-based behavioral experiments. These tools are designed primarily as runtime engines: they generate and render stimuli in real-time during the experimental loop, coupling the stimulus generation logic with the display backend.

CogStim was built rather than contributing to these existing projects for several strategic reasons. First, it decouples generation from presentation. While engines like PsychoPy are powerful, automating them to batch-export thousands of static images for external use (e.g., training a neural network or loading onto a tablet) often requires hacking the windowing system or running "headless" modes that are resource-intensive. CogStim is lightweight and backend-agnostic by design, treating file export as a first-class citizen. Second, CogStim fills a specific niche regarding algorithmic validity: it prioritizes the construction logic of the stimulus (e.g., ensuring area equalization in ANS arrays or preventing overlap in crowded scenes) over the rendering speed required for runtime presentation. Finally, by producing standard assets (PNG/SVG) rather than experiment code, it ensures interoperability; the resulting datasets can be deployed across different platforms, from a custom web app to machine learning pipelines, making CogStim a more versatile tool for researchers.

Software design

CogStim architecture prioritizes modularity and simplicity to remain accessible to researchers with varying programming expertise. We adopted a template method pattern where a base generator class handles infrastructure concerns—such as file I/O, directory structure, and random seed management—while specialized subclasses focus solely on the geometric logic of specific stimuli. This design decouples the "how" of file management from the "what" of stimulus creation, allowing researchers to extend the library with new generators without navigating complex boilerplate code.

To ensure efficiency and maintainability, we minimized external dependencies, relying only on NumPy for vectorised geometric calculations and Pillow for rasterization. This lightweight footprint enables headless execution on servers or CI/CD pipelines, a trade-off we favoured over including heavy GUI dependencies often found in rendering engines.

Research impact statement

CogStim has demonstrated its utility in diverse research contexts, bridging behavioral science and computational modeling. It has been used to generate stimuli for psychometric assessments in educational settings (Correig-Fraga et al., 2025; E. Correig-Fraga et al., 2024) and to create synthetic datasets for evaluating visual computation models in neuroscience (under review). These applications validate the library's capability to support both traditional psychological experiments and modern data-driven approaches.

To facilitate broad community adoption and ensure long-term reliability, CogStim adheres to rigorous software engineering standards. It includes a comprehensive test suite, continuous integration (CI) pipelines, and is distributed via PyPI under a permissive MIT license. Recognizing the varying technical expertise in the field, the project features a novel documentation strategy: an LLM-optimized manual designed to help non-programmers generate complex CLI commands via natural language prompting. This combination of robust engineering and accessibility effectively democratizes access to rigorous stimulus generation, ensuring that high-quality, reproducible stimuli are available to the wider research community.

Software description

Design and key features

- **Task coverage**: ANS two-colour dot arrays; single-colour dot arrays; MTS pairs with optional area equalization; geometric shapes (circle, star, triangle, square); oriented line/stripe patterns; fixation targets.
- **Determinism & reproducibility**: global seed handling for Python/NumPy; same parameters and same seed will yield identical images.
- **Robust dot engine**: `DotsCore` enforces non-overlap, boundary validity, optional area equalization, and (for MTS) pair equalization within tolerances.
- **Stimulus equalization algorithms**: CogStim implements robust geometric equalization methods that adjust dot radii so that total surface areas are matched either within two-colour ANS arrays or between sample–match pairs. These procedures guarantee perceptually fair stimuli for numerosity and matching tasks, maintaining non-overlap and boundary validity while achieving precise area ratios within configurable tolerances.
- **CLI & Python API**: consistent configuration via dictionaries in code and ergonomic subcommands in the CLI.

Implementation and dependencies

CogStim is implemented in Python (≥ 3.10) (Python Software Foundation, 2023) and builds upon a small number of widely used open-source libraries. Image creation and drawing operations are handled through Pillow (Clark, 2015), while all geometric computations and randomization routines rely on NumPy (Harris et al., 2020). The library uses tqdm (Costa-Luis & others, 2022) to provide progress bars during generation processes and adopts standard Python modules such as argparse for command-line interfaces and pytest for automated testing.

The codebase is organized around a small set of generator classes that call these dependencies through a unified interface. Each generator defines the parameters of a particular task (e.g., ANS, MTS, shapes, lines, fixation) and uses Pillow for rendering, NumPy for geometric calculations, and tqdm for user feedback. This results in a lightweight, portable implementation that can run on any system supporting Python without special dependencies or graphical backends.

All dependencies are open source, actively maintained, and available through the Python Package Index (PyPI), ensuring long-term accessibility and compatibility with typical research workflows. The project is licensed under MIT, and available as a Git repository in Github.

Example figures

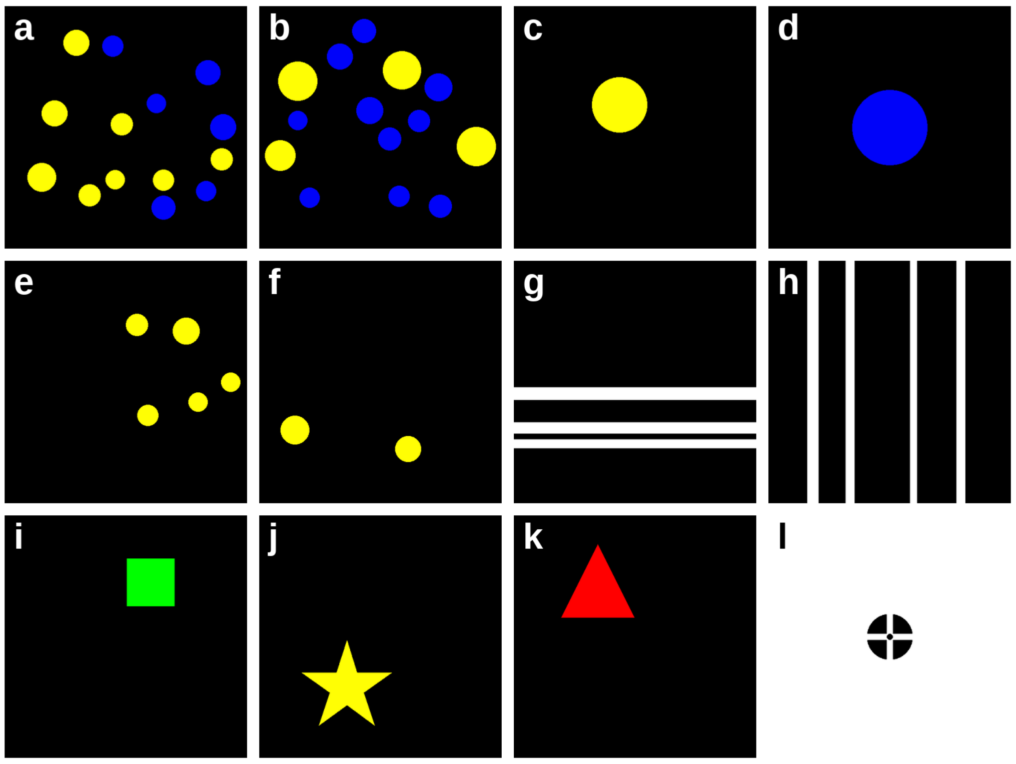

Figure 1: Representative stimuli generated by CogStim.

Figure 1. Representative stimuli generated by CogStim across different task paradigms. **(a, b)** Approximate Number System (ANS) two-colour dot arrays for numerosity discrimination tasks, with (b) showing area-equalized dots between colours. **(c, d)** Single geometric shapes (circle) in different colours, used in shape discrimination tasks. **(e, f)** Single-colour dot arrays suitable for numerosity estimation, match-to-sample (MTS) paradigms, or as components in multi-feature discrimination tasks. **(g, h)** Oriented line/stripe patterns for orientation discrimination experiments. **(i, j, k)** Additional geometric shapes (square, star, triangle) in various colours, demonstrating the library's shape generation capabilities for categorical perception and visual search tasks. **(l)** Fixation cross stimulus for experimental trial preparation and gaze control.

Availability

- Repository: https://github.com/eudald-seeslab/cogstim
- License: MIT
- Issue tracker: enabled and publicly readable
- Archive: upon acceptance, we will create a tagged release, archive on Zenodo and include the DOI here.

Acknowledgements

We thank Innovamat Education for their support in the development of this open source work.

AI usage disclosure

The core architecture and foundational algorithms of CogStim were developed prior to the widespread adoption of generative AI tools. However, during recent development cycles, AI assistants were utilized to assist in refactoring legacy code, homogenizing interfaces across generator modules, and expanding specific functionalities to ensure consistency. Regarding this manuscript, generative AI tools were employed to review and refine the English language and narrative flow. The authors have manually verified all AI-generated suggestions and retain full responsibility for the accuracy and integrity of both the software and the publication.

Conflicts of Interest

Authors declare no competing interests.

References

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

Clark, A. (2015). *Pillow (PIL fork) documentation*. readthedocs. https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf

Correig-Fraga, E., Vilalta-Riera, A., & Calvo-Pesce, C. (2024). Development and validation of a semi-automated, scalable response to intervention framework in mathematics. *SN Social Sciences*, *4*(2), 1–19. https://doi.org/10.1007/s43545-024-00835-7

Correig-Fraga, Sales-Pardo, & Guimerà, R. (2025). Interplay between children's cognitive profiles and within-school social interactions is nuanced and differs across ages. *Communications Psychology*, *3*(1). https://doi.org/10.1038/s44271-025-00227-4

Costa-Luis, C. da, & others. (2022). *Tqdm: A fast, extensible progress bar for python and CLI*. https://github.com/tqdm/tqdm.

Halberda, J., Mazzocco, M. M. M., & Feigenson, L. (2008). Individual differences in non-verbal number acuity correlate with maths achievement. *Nature*, *455*(7213), 665–668. https://doi.org/10.1038/nature07246

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

Leeuw, J. R. de. (2015). jsPsych: A JavaScript library for creating behavioral experiments in a web browser. *Behavior Research Methods*, *47*(1), 1–12. https://doi.org/10.3758/s13428-014-0458-y

Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., Kastman, E., & Lindeløv, 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

Python Software Foundation. (2023). *Python: A programming language for scientific computing*. https://www.python.org/.

Sella, F., Lanfranchi, S., & Zorzi, M. (2013). Enumeration skills in down syndrome. *Research in Developmental Disabilities*, *34*(11), 3798–3806. https://doi.org/10.1016/j.ridd.2013.07.038

Srinivasan, M. V. (2021). Vision, perception, navigation and "cognition" in honeybees and applications to aerial robotics. *Biochemical and Biophysical Research Communications*, *564*, 4–17. https://doi.org/10.1016/j.bbrc.2020.09.052

Thaler, L., Schütz, A. C., Goodale, M. A., & Gegenfurtner, K. R. (2013). What is the best fixation target? The effect of target shape on stability of fixational eye movements. *Vision Research*, *76*, 31–42. https://doi.org/10.1016/j.visres.2012.10.012