

# <sup>1</sup> CogStim: Reproducible visual stimulus generation for cognitive science, neuroscience, and vision research

<sup>3</sup> **Eudald Correig-Fraga**  <sup>1</sup>

<sup>4</sup> 1 Innovamat Education, Sant Cugat del Vallès, Catalonia (Spain)

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- <sup>5</sup> [Review](#) 
- <sup>6</sup> [Repository](#) 
- <sup>7</sup> [Archive](#) 

**Editor:** 

**Submitted:** 08 November 2025

**Published:** unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

## Summary

<sup>8</sup> 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

<sup>15</sup> 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.

<sup>22</sup> 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

<sup>34</sup> 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

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

## 53 Software design

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

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

## 65 Research impact statement

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

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

## 80 Software description

### 81 Design and key features

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

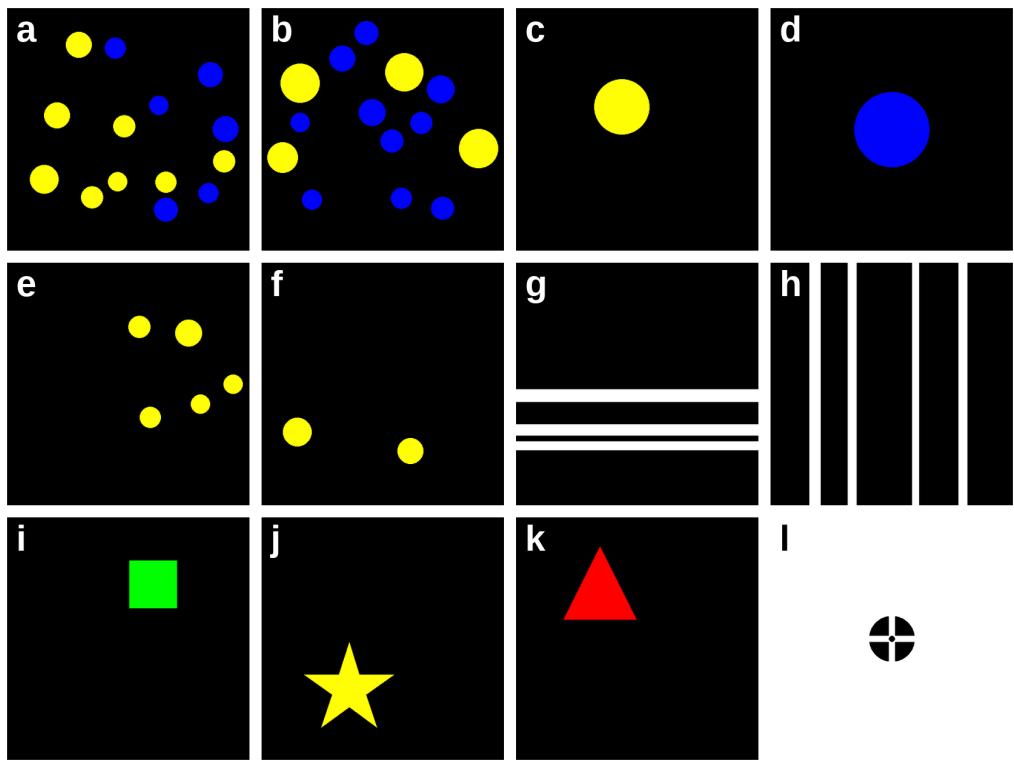
### 96 Implementation and dependencies

97     CogStim is implemented in Python ( $\geq 3.10$ ) ([Python Software Foundation, 2023](#)) and builds  
98         upon a small number of widely used open-source libraries. Image creation and drawing  
99         operations are handled through Pillow ([Clark, 2015](#)), while all geometric computations and  
100         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  
101         Python modules such as argparse for command-line interfaces and pytest for automated testing.

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

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

<sup>112</sup> **Example figures**



**Figure 1:** Representative stimuli generated by CogStim.

<sup>113</sup> **Figure 1.** Representative stimuli generated by CogStim across different task paradigms. (a, b)  
<sup>114</sup> Approximate Number System (ANS) two-colour dot arrays for numerosity discrimination tasks,  
<sup>115</sup> with (b) showing area-equalized dots between colours. (c, d) Single geometric shapes (circle)  
<sup>116</sup> in different colours, used in shape discrimination tasks. (e, f) Single-colour dot arrays suitable  
<sup>117</sup> for numerosity estimation, match-to-sample (MTS) paradigms, or as components in multi-  
<sup>118</sup> feature discrimination tasks. (g, h) Oriented line/stripe patterns for orientation discrimination  
<sup>119</sup> experiments. (i, j, k) Additional geometric shapes (square, star, triangle) in various colours,  
<sup>120</sup> demonstrating the library's shape generation capabilities for categorical perception and visual  
<sup>121</sup> search tasks. (l) Fixation cross stimulus for experimental trial preparation and gaze control.

<sup>122</sup> **Availability**

- <sup>123</sup> ▪ Repository: <https://github.com/eudald-seeslab/cogstim>  
<sup>124</sup> ▪ License: MIT  
<sup>125</sup> ▪ Issue tracker: enabled and publicly readable  
<sup>126</sup> ▪ Archive: upon acceptance, we will create a tagged release, archive on Zenodo and include  
<sup>127</sup> the DOI here.

<sup>128</sup> **Acknowledgements**

<sup>129</sup> We thank Innovamat Education for their support in the development of this open source work.

## 130 AI usage disclosure

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

## 138 Conflicts of Interest

139 Authors declare no competing interests.

## 140 References

- 141 Brainard, D. H. (1997). The psychophysics toolbox. *Spatial Vision*, 10(4), 433–436.
- 142 Clark, A. (2015). *Pillow (PIL fork) documentation*. readthedocs. <https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf>
- 143 Correig-Fraga, E., Vilalta-Riera, A., & Calvo-Pesce, C. (2024). Development and validation of  
144 a semi-automated, scalable response to intervention framework in mathematics. *SN Social  
145 Sciences*, 4(2), 1–19. <https://doi.org/10.1007/s43545-024-00835-7>
- 146 Correig-Fraga, Sales-Pardo, & Guimerà, R. (2025). Interplay between children's cognitive  
147 profiles and within-school social interactions is nuanced and differs across ages.  
148 *Communications Psychology*, 3(1). <https://doi.org/10.1038/s44271-025-00227-4>
- 149 Costa-Luis, C. da, & others. (2022). *Tqdm: A fast, extensible progress bar for python and  
150 CLI*. <https://github.com/tqdm/tqdm>.
- 151 Halberda, J., Mazzocco, M. M. M., & Feigenson, L. (2008). Individual differences in non-  
152 verbal number acuity correlate with maths achievement. *Nature*, 455(7213), 665–668.  
153 <https://doi.org/10.1038/nature07246>
- 154 Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D.,  
155 Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk,  
156 M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., ... Oliphant,  
157 T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- 158 Leeuw, J. R. de. (2015). jsPsych: A JavaScript library for creating behavioral experiments in a  
159 web browser. *Behavior Research Methods*, 47(1), 1–12. <https://doi.org/10.3758/s13428-014-0458-y>
- 160 Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., Kastman,  
161 E., & Lindeløv, J. K. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior  
162 Research Methods*, 51(1), 195–203. <https://doi.org/10.3758/s13428-018-01193-y>
- 163 Python Software Foundation. (2023). *Python: A programming language for scientific  
164 computing*. <https://www.python.org/>.
- 165 Sella, F., Lanfranchi, S., & Zorzi, M. (2013). Enumeration skills in down syndrome. *Research  
166 in Developmental Disabilities*, 34(11), 3798–3806. <https://doi.org/10.1016/j.ridd.2013.07.038>
- 167 Srinivasan, M. V. (2021). Vision, perception, navigation and “cognition” in honeybees and  
168 applications to aerial robotics. *Biochemical and Biophysical Research Communications*,
- 169

<sup>173</sup> 564, 4–17. <https://doi.org/10.1016/j.bbrc.2020.09.052>

<sup>174</sup> Thaler, L., Schütz, A. C., Goodale, M. A., & Gegenfurtner, K. R. (2013). What is the best  
<sup>175</sup> 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>

DRAFT