

¹ GEFF: Graph Exchange File Format

² Morgan Schwartz  ^{1*}, Caroline Malin-Mayor  ^{1*}, Talley Lambert  ², Teun Huijben  ³, Jan Funke  ¹, Laura Xénard  ^{4,13}, Mark Kittisopkul  ¹, Draga Doncila Pop  ⁵, Yohsuke Fukai  ⁶, Jordão Bragantini  ³, Melisande Croft  ⁷, Anniek Stokkermans  ⁸, Georgeos Hardo  ^{9,12}, Benjamin Gallusser  ³, Kasia Kedziora  ¹⁰, Jean-Yves Tinevez  ⁴, Ko Sugawara  ¹¹, Tobias Pietzsch  ¹⁴, and Ilan Theodoro  ³

⁸ 1 Janelia Research Campus, Howard Hughes Medical Institute, Ashburn, VA, USA ² Harvard Medical School, Boston, MA, USA ³ Biohub, San Francisco, CA, USA ⁴ Institut Pasteur, Université Paris Cité, Image Analysis Hub, Paris, France ⁵ Monash University, Melbourne, Australia ⁶ RIKEN Pioneering Research Institute, Kobe, Hyōgo, Japan ⁷ Human Technopole, Milan, Italy ⁸ Hubrecht Institute, Utrecht, Netherlands ⁹ United Arab Emirates University, Al Ain, UAE ¹⁰ University of Pittsburgh, Pittsburgh, PA, USA ¹¹ RIKEN Center for Biosystems Dynamics Research, Kobe, Japan ¹² University of Cambridge, Cambridge, UK ¹³ Institut Pasteur, Université Paris Cité, INSERM U1225, Paris, France ¹⁴ Independent Consultant, Germany * These authors contributed equally.

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

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: George K. Thiruvathukal  

Submitted: 20 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Published: unpublished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/)).

¹⁶ Summary

¹⁷ GEFF (Graph Exchange File Format) is a file format specification for exchanging graph data. ¹⁸ Its main application today is focused on exchange of animal, cell and organelle tracking data ¹⁹ in the life sciences. It is not intended to be mutable, editable, chunked, or optimized for use ²⁰ in an application setting. As an exchange format with a strict specification, GEFF enables ²¹ interoperability between tools written in various programming languages.

²² The geff repository contains two Python packages: `geff-spec`, the specification of GEFF ²³ metadata written with `pydantic.BaseModels`, which are exported to a JSON schema for use ²⁴ in other languages, and `geff`, the Python library that reads and writes GEFF files to and from ²⁵ several Python in-memory graph data structures (`networkx` ([Hagberg et al., 2008](#)), `rustworkx` ²⁶ ([Treinish et al., 2022](#)), and `spatial-graph`). A Java implementation of the GEFF v1 spec, ²⁷ `geff-java` is in progress in a separate repository.

²⁸ Statement of Need

²⁹ Cell and organelle tracking is an active area of research with many tools for performing tracking ³⁰ and visualizing results. At the [2023 Janelia Trackathon](#), a two-week workshop gathering ³¹ cell tracking researchers, there was widespread agreement that a common graph file format ³² would benefit the field by reducing code duplication and increasing standardization between ³³ projects. However, attempts made at that time were intended to be mutable and optimized, ³⁴ which introduced barriers to code generation and adoption by tools with different types of ³⁵ optimization needs. The 2025 Janelia Trackathon brought together all the authors of GEFF to ³⁶ decide on the specification and initial implementation, which was accomplished in a week-long ³⁷ hackathon.

³⁸ Research Impact Statement

³⁹ GEFF allows different research tools to all track and visualize the same data, reducing barriers
⁴⁰ to pipelining analysis and visualization tools, even across languages. As of submission time,
⁴¹ the following tools all support either saving and/or loading GEFF files: [motile-tracker](#),
⁴² [traccuracy](#), [ultrack](#), [track_gardener](#), [laptrack](#), [trackastr](#), [TrackMate](#), [InTRACKtive](#),
⁴³ [tracksdata](#) and [napari-geff](#). The developers have already fielded inquiries through GitHub
⁴⁴ and email from researchers, both developers and end-users, about how best to use GEFF to
⁴⁵ accelerate their research workflow. We hope that GEFF will become the standard for storing
⁴⁶ and exchanging tracking information in the bio-image analysis community, and potentially
⁴⁷ even other fields that require exchanging graph-based information.

⁴⁸ State of the Field

⁴⁹ There are many formats used to store and exchange tracking solutions. A commonly used one
⁵⁰ is the Cell Tracking Challenge (CTC) ([Maška et al., 2014](#)) format, which combines TIFF files
⁵¹ with segmentation masks and a CSV file to provide division edges. However, some tracking
⁵² applications such as particle tracking do not operate on segmentations, but instead utilize
⁵³ point detections, making this format not applicable. As such, individual tracking tools often
⁵⁴ define their own format for saving tracking results; for example, TrackMate ([Tinevez et al.,
55](#) [2017](#)) has a specific XML file format, Mastodon ([Pietzsch et al., 2025](#)) saves and loads from a
⁵⁶ binary file, the [Motile Tracker](#) exports and loads to CSV files with specific node ID, parent
⁵⁷ ID, and location columns, and Ultrack ([Bragantini et al., 2025](#)) has a custom SQL database.
⁵⁸ In these existing file formats, there is limited support for storing additional properties on
⁵⁹ either nodes or edges. Additionally, none of these tools shared a common file format, which
⁶⁰ prevented interactions between them and strongly limited the scope and ambition of track
⁶¹ analysis pipelines. In contrast to prior efforts to create a common file format ([Gonzalez-Beltran
62](#) [et al., 2020](#)) ([Kyoda, 2020](#)), we worked with tool authors to integrate support for GEFF directly
⁶³ into the tools themselves rather than only creating a third-party library. Each of them can now
⁶⁴ export to and import from GEFF in addition to their custom formats, enabling interoperability
⁶⁵ with minimal code change in each library.

⁶⁶ Implementation

⁶⁷ GEFF is built on zarr ([Zarr Developers, 2019](#)), a common file format used in bioimage analysis.
⁶⁸ Graphs are represented as an array of node IDs and an array of edge IDs where each edge
⁶⁹ ID is a tuple of two node IDs. Nodes and edges can have properties, which are stored in
⁷⁰ a properties array with corresponding indices. The specification includes support for nodes
⁷¹ and edges with missing properties, as well as variable-length properties. To support the cell
⁷² tracking community, the GEFF specification also provides specific metadata with standardized
⁷³ meaning, including positional axes, tracklet and lineage IDs, and linking to related objects
⁷⁴ such as image and segmentation arrays.

⁷⁵ GEFF's object specification supports a wide range of shapes, enabling its application across
⁷⁶ diverse fields in the life sciences. The library integrates multiple representation formats, from
⁷⁷ binary masks (2D/3D), commonly used in cell biology, developmental biology, and natural
⁷⁸ image tracking—to geometric primitives (points, circles, ellipses, spheres, and ellipsoids),
⁷⁹ which are essential in super-resolution microscopy, virology, and developmental studies. It
⁸⁰ also includes polygons and meshes for detailed structural analysis in cell biology, microbiology,
⁸¹ and complex shape modeling, as well as pose-based representations for markerless tracking of
⁸² anatomical keypoints in multi-animal and behavioral research.

⁸³ Each object, track, or lineage, is represented by a simple directed graph, where each edge
⁸⁴ is a link from one biological object detected in a frame, to its detection in the earliest next

85 frame. GEFF can therefore be used to harness tracking data with gaps (an edge extends over
86 more than two adjacent time-points, because of a missing detection or object exit and reentry),
87 object divisions and objects fusions. Multiple GEFFs can refer to the same set of segmentations
88 or detections which makes it possible to easily store multiple tracking solutions or hypotheses
89 without duplicating the underlying data, which is a limitation of the commonly-used CTC
90 format.

91 By integrating these diverse object and link representations, GEFF facilitates seamless data
92 exchange between segmentation algorithms, morphometric analysis tools, and tracking pipelines,
93 ensuring compatibility with both established and emerging imaging workflows. This versatility
94 makes the library a valuable resource for researchers working across disciplines, from high-
95 throughput cell biology to fine-grained anatomical studies in developmental and computational
96 biology.

97 Software Design

98 The GEFF specification emphasizes simplicity over optimization. It would be infeasible to
99 make a single graph format, or even tracking format, that was optimized for all use cases.
100 Implementing a simple exchange format allows a variety of tools with different goals to
101 exchange information, while retaining their optimized internal formats. Additionally, focusing
102 on an exchange format allows cross-language compatibility. In addition to a hand-written
103 specification, the GEFF specification package provides Pydantic classes and a JSON schema
104 for creating and validating the metadata properties.

105 The Python reference implementation has a highly modular design. The core_io module is the
106 heart of the implementation, with core read and write implementations that can be used by any
107 graph library, converting from an InMemoryGeff data structure to an on-disk zarr and back.
108 This crucial abstraction improves maintainability as well as extensibility; new graph library
109 implementations only need to convert to and from the InMemoryGeff data structure, and any
110 improvements to performance or bugfixes in the core_io module are automatically propagated
111 to the individual graph library implementations. GEFF supports Zarr specification v2 and v3,
112 and has minimal dependencies, making it a lightweight dependency for other libraries.

113 The _graph_libs currently implements three graph backends. networkx and rustworkx are
114 two of the most common Python graph libraries, making GEFF adoption simple for most
115 Python programmers. spatial_graph is a newer graph library with improved efficiency when
116 searching for all graph elements in a spatial region. While the Python implementation contains
117 significant internal typing and abstraction logic that reduces code duplication and enhances
118 maintainability, the external API is quite simple; most users will only need to use the public
119 read and write functions.

120 The other three modules provide additional functionality for new users and developers. The
121 convert module contains a CLI tool for converting existing on-disk formats to GEFF, including
122 TrackMate XML, the Cell Tracking Challenge format, and any CSV-like format that can be
123 loaded into a Pandas dataframe. The testing module provides InMemoryGeff objects with a
124 variety of valid GEFF data combinations for testing both existing and new implementations.
125 Finally, the validate module provides helper functions for testing the validity of on-disk
126 GEFFs with varying levels of inspection, from fast, structure-only validation to intensive data
127 validation.

128 Extensibility

129 While GEFF was developed by the cell tracking research community, it is a generic graph
130 exchange format that could be easily extended to other use cases with additional metadata to
131 specify the meaning of standard properties.

132 Acknowledgments

133 We would like to thank HHMI Janelia Research Campus for hosting the 2025 Janelia Trackathon
134 and the other attendees of the trackathon for their discussion. LX and JYT acknowledge
135 support from the French National Research Agency (France Biolimaging, ANR-24-INBS-0005
136 FBI BIOGEN). LX acknowledges support from the INCEPTION project (PIA/ANR-16-CONV-
137 0005) and the FIRE PhD program funded by the Bettencourt Schueller foundation and the
138 EURIP graduate program (ANR-17-EURE-0012). KS acknowledges support from the JST
139 CREST, Japan (JPMJCR2124).

140 Competing interests

141 KS is employed part-time by LPIXEL Inc.

142 AI Usage Disclosure

143 All specification and paper content was written manually and reflects the careful thought and
144 input of the community. GEFF is an open source project, and as such contributors are free to
145 use any tools, AI or otherwise, to generate code contained in pull requests. All pull requests
146 are reviewed by a core developer and often iterated on multiple times; therefore, all content in
147 the repository represents the effort and judgment of the authors.

148 References

- 149 Bragantini, J., Theodoro, I., Zhao, X., Huijben, T. A., Hirata-Miyasaki, E., VijayKumar,
150 S., Balasubramanian, A., Lao, T., Agrawal, R., Xiao, S., & others. (2025). Ultrack:
151 Pushing the limits of cell tracking across biological scales. *Nature Methods*, 1–14. <https://doi.org/10.1101/2024.09.02.610652>
- 152 Gonzalez-Beltran, A. N., Masuzzo, P., Ampe, C., Bakker, G.-J., Bésson, S., Eibl, R. H., Friedl,
153 P., Gunzer, M., Kittisopkul, M., Le Dévédec, S. E., Leo, S., Moore, J., Paran, Y., Prilusky,
154 J., Rocca-Serra, P., Roudot, P., Schuster, M., Sergeant, G., Strömlad, S., ... Martens, L.
155 (2020). Community standards for open cell migration data. *GigaScience*, 9(5), gaa041.
156 <https://doi.org/10.1093/gigascience/gaa041>
- 157 Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring Network Structure,
158 Dynamics, and Function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman
159 (Eds.), *Proceedings of the 7th Python in Science Conference* (pp. 11–15). <https://doi.org/10.25080/tcwv9851>
- 160 Kyoda, K. H. L. A. T., Koji AND Ho. (2020). BD5: An open HDF5-based data format
161 to represent quantitative biological dynamics data. *PLOS ONE*, 15(8), 1–11. <https://doi.org/10.1371/journal.pone.0237468>
- 162 Maška, M., Ulman, V., Svoboda, D., Matula, P., Matula, P., Ederra, C., Urbiola, A., España,
163 T., Venkatesan, S., Balak, D. M., & others. (2014). A benchmark for comparison of
164 cell tracking algorithms. *Bioinformatics*, 30(11), 1609–1617. <https://doi.org/10.1093/bioinformatics/btu080>
- 165 Pietzsch, T., Tinevez, J.-Y., Sugawara, K., Arzt, M., Ulman, V., & Hahmann, S. (2025).
166 Mastodon – a large-scale tracking and track-editing framework for large, multi-view images.
167 <https://mastodon.readthedocs.io/en/latest/>
- 168 Tinevez, J.-Y., Perry, N., Schindelin, J., Hoopes, G. M., Reynolds, G. D., Laplantine, E.,
169 Bednarek, S. Y., Shorte, S. L., & Eliceiri, K. W. (2017). TrackMate: An open and
170 extensible image analysis library for cell tracking. *bioRxiv*, 152032. <https://doi.org/10.1101/152032>
- 171

- 174 extensible platform for single-particle tracking. *Methods*, 115, 80–90. <https://doi.org/10.1016/j.ymeth.2016.09.016>
- 175 Treinish, M., Carvalho, I., Tsilimigkounakis, G., & Sá, N. (2022). Rustworkx: A high-
176 performance graph library for python. *Journal of Open Source Software*, 7(79), 3968.
<https://doi.org/10.21105/joss.03968>
- 177
- 178
- 179 Zarr Developers. (2019). *Zarr-specs*. <https://github.com/zarr-developers/zarr-specs>

DRAFT