




LinkingLines: Using the Hough Transform to Cluster Line Segments and Mesoscale Feature Extraction

Allison Kubo Hutchison ¹, Leif Karlstrom ¹, and Tushar Mittal ²

1 University of Oregon, Eugene, OR, USA 2 Pennsylvania State University, University Park, PA, USA

DOI: [10.21105/joss.06147](https://doi.org/10.21105/joss.06147)

Software

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

Editor: [Hugo Ledoux](#)  

Reviewers:

- [@evetion](#)
- [@nialov](#)

Submitted: 30 October 2023

Published: 07 June 2024

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Linear feature analysis plays a fundamental role in geospatial applications, from detecting infrastructure networks to characterizing geological formations. In this paper, we introduce `linkinglines`, an open-source Python package tailored for the clustering and feature extraction of linear structures in geospatial data. Our package leverages the Hough Transform, commonly used in image processing, performs clustering of line segments in the Hough Space, and then provides unique feature extraction methods and visualization. `linkinglines` empowers researchers, data scientists, and analysts across diverse domains to efficiently process, understand, and extract valuable insights from linear features, contributing to more informed decision-making and enhanced data-driven exploration. We have used `linkinglines` to map dike swarms with thousands of segments associated with Large Igneous Provinces in Kubo Hutchison et al. (2023).

Statement of Need

The `linkinglines` Python package addresses the need for quantitative and automated line clustering and analysis in geospatial data. In geology and related fields, there is a need to analyze and extract patterns in overlapping and sometimes incomplete datasets of linear features such as fractures, dikes, or roads.

The primary needs that the `linkinglines` package fulfills include:

1. **Dissected Line Extraction or Data Reduction:** In areas where land cover or data availability affects the complete mapping of linear features, this package can link together similarly oriented segments into lines in a quantitative automated way. This is also an data reduction technique.
2. **Feature Extraction and Analysis:** The `linkinglines` package provides functions to extract and then compute essential metrics and statistics on extracted linear, radial or circumferential type features which are commonly seen in dike swarms or fracture networks.
3. **Custom Plotting and Visualization:** Effective visualization is critical for data interpretation. The package offers custom plotting scripts, making it easier to visualize results and communicate findings.

State of the Field

This package was originally developed to tackle the issue of mapping dike segments. Rugged terrain, vegetation cover, and a large area made it impossible to accurately map dikes. The length, density, and structure of the dike swarm affects how magma is transported and erupted. Scaling analysis indicated that for segments of widths of 10 m, dikes could be tens to thousands

of kilometers long; however observed segments were two orders of magnitude shorter (Morris et al., 2020). Additionally, the complex overlapping structure of the dike swarm was difficult to analyze. We designed linkingLines to extract not only lines from line segments but also help analyze the mesoscale structure of the dike swarm. Using the unique properties of the Hough Transform we can extract several unique mesoscale structures within a group of lines.

Issues with data coverage, incompleteness, and complexity occur in a wide range of Earth and Planetary science data products. As an example we show in our documentation how this work can also be applied to lineaments on Venus (Grosfils et al., 2011) and fracture networks (Ovaskainen et al., 2023). Clustering and machine learning methods are commonly applied geosciences within the subdisciplines of seismology, geochemistry, and planetary science (Li et al., 2023). There are currently available packages to work with geospatial data clustering, examples are geopandas, PySAL, and others which perform the Hough Transform on images such as scikit-image or OpenCV. However this seems to be the first union of the two methods and specific to geoscience data applications (Gaboardi et al., 2021; Jordahl et al., 2020; Walt et al., 2014). Related packages which are specific to geosciences, such as fractopo which focuses on network analysis of fractures, could be utilized in parallel with linkingLines for additional analysis (Ovaskainen, 2023).

Algorithm

linkingLines can read in any common geospatial data format using geopandas such as ESRI Shapefiles, GeoJSON, or Well Known Text (Jordahl et al., 2020). Preprocessing is applied to the dataset so that only straight lines are considered in the algorithm. This is done by loading in the points from each vector object and performing linear regression, only considering those that yield a line with a $p > 0.05$. The data is then formatted into a pandas DataFrame. This package heavily uses pandas as the database structure for ease of use, data manipulation, and integration with numpy and scipy (McKinney, 2010).

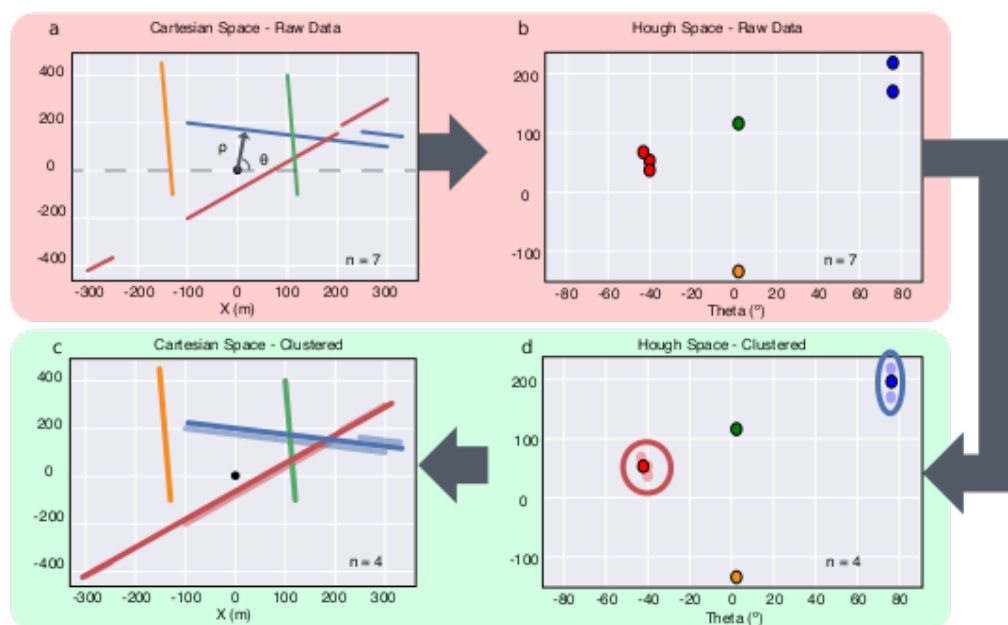


Figure 1: Dike linking algorithm using the Hough Transform. First, raw data in Cartesian space is converted into Hough space (a and b). Agglomerative clustering is then performed on the data in Hough coordinates (d). In this example, there are four dikes in total and two (red and blue) clusters. The clusters are redrawn by connecting the endpoints of the segments in the cluster (c).

The Hough Transform is an image processing technique used for detecting straight lines and other patterns in images (Ballard, 1981; Duda & Hart, 1972). After loading in the data, it is assumed to be already line structures, so the accumulator array of the Hough Transform is skipped. First, the angle of the line segment is found. In many methods, it is the left-hand corner of the image (Ballard, 1981), but we choose the average midpoint of the line segments (Figure 1b). Other origins can be specified in certain functions using the x_c and y_c arguments. After the coordinate transform, ρ and θ become the basis for the next step, where we utilize Scipy's clustering algorithm (Virtanen et al., 2020).

After labels are assigned in the clustering portion of the algorithm, new lines are drawn using the endpoints of the clustered lines (Figure 1d). For each cluster, the nearest neighbors of the segment midpoints are calculated in Cartesian space, allowing for an analysis of the Cartesian spatial clustering of the lines. We also introduce an optional filtering step, which analyzes the maximum nearest neighbor difference of midpoints normalized by the total cluster length. This filters out clusters with segments that are not evenly clustered in Cartesian space.

Feature Extraction

Additionally, we leverage the unique properties of the Hough Transform to combine clustering with feature extraction. In the original usage case of overlapping complex dike swarms, two potential end members of swarm types are linear and radial or circumferential swarms (Figure 2). Using the equation of the Hough Transform, we can apply curve-fitting to quantitatively fit the data to a radial pattern.

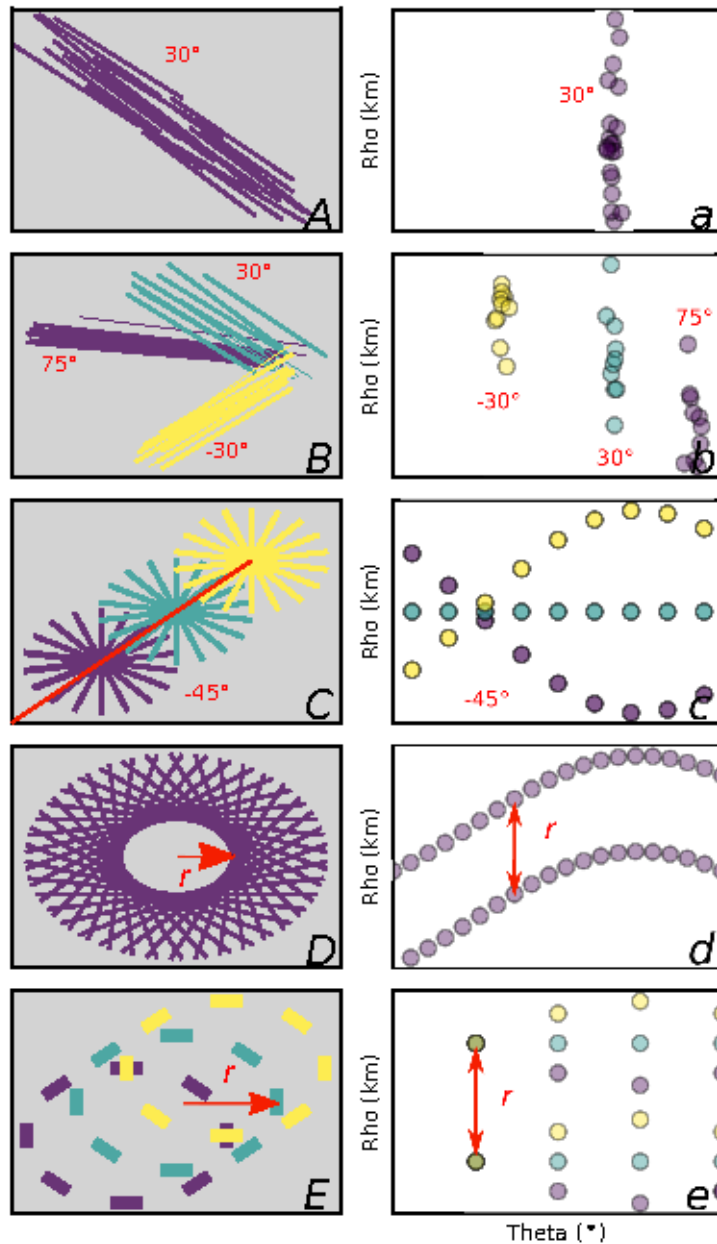


Figure 2: Synthetic dike swarms in a Cartesian space (gray background, uppercase label) and Hough Transform space (white background, lowercase label). (A) Shows a simple linear swarm oriented at 30° . (B) Shows three linear swarms at -30° , 30° , 75° . (C) Shows three radial swarms aligned at a -45° angle. The angle at which radial swarms intersect in the Hough space (HS) is the angle of their relative orientation in Cartesian space. (D) Shows a circumferential swarm with the lines extending to show how it converges to a radial swarm. The radius of the circumferential swarm is equal to the spacing of the parallel two curves in HS. (E) Shows three circumferential swarms with the same radius aligned at a -45° angle.

For extraction of linear features we can apply the Hough accumulator array, a 2D histogram of θ and ρ . You set the size of bins in the histogram and if clusters fall within those boxes they can be thought of as mesoscale clusters. We allow for flexibility of cutoffs for these mesoscale feature extraction, so it can be tailored to each research or engineering application.

Future Work

This package takes geospatial or other types of line segment data and clusters them based on their orientation. Currently, the implementation only works with linear features, however it could be generalized to arbitrary shapes for more flexibility (Ballard, 1981). Additionally, future work could incorporate other shapes or patterns in the Hough Space and could extend the feature extraction methods laid out here. We invite collaboration to increase the capabilities of this code.

Acknowledgements

This paper was made possible by the “Crafting Quality Research Software and Navigating Publication in Software Journals” held by the Computational Infrastructure for Geodynamics in September 2023.

References

- Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2), 111–122. [https://doi.org/10.1016/0031-3203\(81\)90009-1](https://doi.org/10.1016/0031-3203(81)90009-1)
- Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1), 11–15. <https://doi.org/10.1145/361237.361242>
- Gaboardi, J. D., Rey, S., & Lumnitz, S. (2021). Spaghetti: Spatial network analysis in PySAL. *Journal of Open Source Software*, 6(62), 2826. <https://doi.org/10.21105/joss.02826>
- Grosfils, E., Long, S., Venechuk, E., Hurwitz, D., Richards, J., Kastl, B., Drury, D., & Hardin, J. (2011). Geologic map of the Ganiki Planitia quadrangle (v-14). *Venus. United States Geological Survey, Scientific Investigations Map*, 3121. <https://doi.org/10.3133/sim3121>
- Jordahl, K., Bossche, J. V. den, Fleischmann, M., Wasserman, J., McBride, J., Gerard, J., Tratner, J., Perry, M., Badaracco, A. G., Farmer, C., Hjelle, G. A., Snow, A. D., Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur, A., ... Leblanc, F. (2020). *Geopandas/geopandas: v0.8.1* (Version v0.8.1). Zenodo. <https://doi.org/10.5281/zenodo.3946761>
- Kubo Hutchison, A., Karlstrom, L., & Mittal, T. (2023). Multiscale spatial patterns in giant dike swarms identified through objective feature extraction. *Geochemistry, Geophysics, Geosystems*, 24(9), e2022GC010842. <https://doi.org/10.1029/2022GC010842>
- Li, Y. E., O'malley, D., Beroza, G., Curtis, A., & Johnson, P. (2023). Machine learning developments and applications in solid-earth geosciences: Fad or future? In *Journal of Geophysical Research: Solid Earth* (No. 1; Vol. 128, p. e2022JB026310). Wiley Online Library. <https://doi.org/10.1029/2022JB026310>
- McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th python in science conference: SciPy 2010* (pp. 56–61). SciPy. <https://doi.org/10.25080/Majora-92bf1922-00a>
- Morriss, M. C., Karlstrom, L., Nasholds, M. W., & Wolff, J. A. (2020). The Chief Joseph Dike Swarm of the Columbia River flood basalts, and the legacy data set of William H. Taubeneck. *Geosphere*, 16(4), 1082–1106. <https://doi.org/10.1130/GES02173.1>
- Ovaskainen, N. (2023). Fractopo: A Python package for fracture network analysis. *Journal of Open Source Software*, 8(85), 5300. <https://doi.org/10.21105/joss.05300>
- Ovaskainen, N., Skyttä, P., Nordbäck, N., & Engström, J. (2023). Detailed investigation

of multi-scale fracture networks in glacially abraded crystalline bedrock at Åland islands, finland. *Solid Earth*, 14(6), 603–624. <https://doi.org/10.5194/se-14-603-2023>

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., & others. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261–272. <https://doi.org/10.1038/s41592-019-0686-2>

Walt, S. van der, Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., & contributors, the scikit-image. (2014). Scikit-image: Image processing in Python. *PeerJ*, 2, e453. <https://doi.org/10.7717/peerj.453>