The Journal of Open Source Software

PyLithics: A Python package for stone tool analysis

Jason J. Gellis[*][1, 2, 3], **Camila Rangel Smith**[1], **and Robert A. Foley**[1, 2, 3]

1 The Alan Turing Institute **2** University of Cambridge **3** Leverhulme Centre for Human Evolutionary Studies

DOI: 10.21105/joss.03738

Software

- Review
- Repository
- Archive

Editor: Nikoleta Glynatsi

Reviewers:

- @MichaelHoltonPrice
- @steko

Submitted: 10 September 2021
Published: 24 January 2022

Summary

Archaeologists have long used stone tools (lithics) to reconstruct the behaviour of prehistoric hominins. While techniques have become more quantitative, there still remain barriers to optimizing data retrieval (Andrefsky,William, 2012). Machine learning and computer vision approaches can be developed to extract quantitative and trait data from lithics, photographs and drawings. `PyLithics` has been developed to capture data from 2D line drawings, focusing on the size, shape and technological attributes of flakes. The problems addressed in the software are: one, capturing data in a form that can be quantified, and information maximized; two, solving the challenges of data that is not a simple linear sequence of bases but complex 3D objects or 2D image representations; and three, transforming and exporting these into systematic data for analysis. The goal is to enhance the size and quality of lithic databases for analysing ancient technology and human behaviour.

Statement of need

`PyLithics` is an open-source, free for use software package for processing lithic artefact illustrations scanned from the literature. Accurately measuring lithic artefacts is difficult and especially time-consuming as lithics and their features are incongruous shapes and sizes. This is especially problematic for the researcher as certain features, such as flake scar size, are useful in elucidating the manufacturing process of an artefact. Thus, while even the best, most complete illustrations are able to visually capture an immense amount of information about an artefact, much of this information is under-utilized or not used at all.

`PyLithics` alleviates these issues by accurately identifying, outlining, and computing lithic shape and linear measures, and returns user ready data. It has been optimized for feature extraction and measurement using a number of computer vision techniques including pixel intensity thresholding, edge detection, contour finding, custom template matching and image kernels. On both conventional and modern drawings, `PyLithics` can identify and label platform, lateral, dorsal, and ventral surfaces, as well as individual dorsal surface scar shape, size, orientation, diversity, number, and order of flake size from greatest to least. Complete size and shape metrics of individual scars and whole flakes can be calculated and recorded. Orientation and flaking direction of dorsal scars can also be calculated. The resulting data can be used for metrical analysis, extracting features indicative of both typologies and technological processes. Data output can easily be employed to explore patterns of variation within and between assemblages.

[*]corresponding author

Methods and workflow

PyLithics is devised to work with illustrations of lithic objects common to publications in archaeology and anthropology. Lithic illustrators have established conventions regarding systems of artefact orientation and proportions. Stone artefacts made by hominins have clear diagnostic features of manufacture that are used to infer technological patterns. A basic division is made between cores, which are the source material that is modified by striking, and flakes (also known as debitage), which are the pieces struck from the cores. Both provide important information, but flakes have been the focus for PyLithics development.

Lithics are normally drawn at a 1:1 scale, with the vertical axis orthogonal to the striking platform – that is, the point on the flake which was struck to remove it from the core. A preferred method is to orient and illustrate various aspects of a flake as a series of adjacent surfaces at 90-degree rotations. Four illustrations are normal: the ventral surface, the surface that was last attached to the core, and is usually a smooth, unbroken surface; the dorsal surface, which is usually marked by scars from previous flake removals; the proximal or platform view, from the top where the flake was struck, and which is approximately at 90 degrees to the ventral surface; and the lateral view, which essentially shows the thickness of the flake and its longitudinal shape. Each aspect, but especially the dorsal surface, contains internal details (i.e., flake scars, cortical areas, etc.), indication of flaking direction radial lines (ripples), and the inclusion of a metric scale (for more information about lithic drawings see (Martingell & Saville, 1988)). Currently PyLithics is optimised to work with unifacial flakes and bifaces, which are relatively flat, two-dimensional objects. For best performance and accurate measurement, images loaded into PyLithics should be:

1. Oriented with the platform orthogonal to the top of the page.
2. Use arrows to indicate flaking direction rather than ripples.
3. Have arrows with longer tails than arrow points.
4. Avoid having arrows or other indicators overlapping or immediately adjacent to flake scars.

While PyLithics can identify and measure surfaces and surface features on illustrations with ripples, too many ripples will reduce measurement accuracy of flake scars. The inputs for PyLithics are images of lithic objects, images of their associated scales, and a metadata CSV file linking the two and giving the scale measurement in millimetres. PyLithics processes the images with the following steps and as illustrated in (Figure 1):

1. Import images and match image name to associated image ID and scale image from CSV metadata file.
2. Calculate a conversion of pixels to millimetres based on the size of the associated scale from CSV metadata file. If no scale is present, measurements will be in pixels
3. Apply noise removal and contrast stretching to images to minimise pixel variation.
4. Pixel intensity thresholding of images to prepare for contour finding.
5. Apply edge detection and contour finding to thresholded images.
6. Calculate metrics of lithic surface features from found contours – area, length, breath, shape, number of vertices.
7. Select contours which outline an entire lithic object's surfaces or select contours of inner scars greater than 3% and less than 50% of the total size of its surface.
8. Classify these selected surface contours as "Dorsal," "Ventral," "Lateral," and/or "Platform" depending on presence or absence. Assign scar contours to these surfaces.
9. If present, find arrows using connected components and template matching, measure their angle and assign angle to associated scar.
10. Plot resulting surface and scar contours on the original images for validation.
11. Output data in a hierarchical JSON file detailing measurement of surface and scar contours.

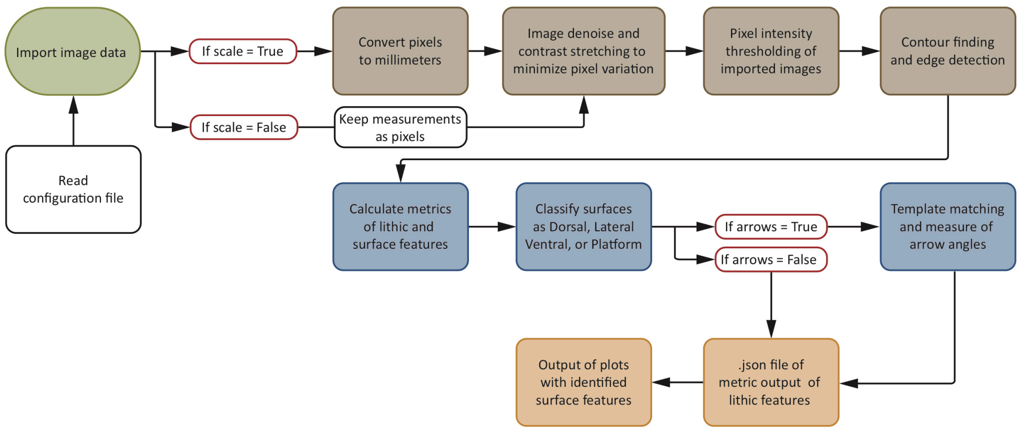

Import image data
If scale = True
Convert pixels to millimeters
Image denoise and contrast stretching to minimize pixel variation
Pixel intensity thresholding of imported images
Contour finding and edge detection
If scale = False
Keep measurements as pixels
Read configuration file
Calculate metrics of lithic and surface features
Classify surfaces as Dorsal, Lateral Ventral, or Platform
If arrows = True
If arrows = False
Template matching and measure of arrow angles
Output of plots with identified surface features
.json file of metric output of lithic features

Figure 1: PyLithics program workflow.

`PyLithics` depends on common Python packages such as NumPy (Harris et al., 2020), SciPy (Virtanen et al., 2020), Pandas (McKinney, 2010) for data processing, Matplotlib (Hunter, 2007) for plotting and scikit-image (Walt et al., 2014) and OpenCv (Bradski, 2000) for image processing and computer vision tasks.

Results

`PyLithics` generates two outputs:

1) An image set comprised of the original input images with superimposed contour identification and derived metrics (see Figure 2 and Figure 3), and if arrows are present in the illustration, angles of flaking direction (see Figure 4).
2) A JSON file with data for lithic objects and surface features found in each image Figure 5. These data are hierarchically organised, first by type of object surface (i.e., ventral, dorsal, lateral, and platform); and second by metrics (in mm) from scars and arrows associated to each object surface. Output includes: object id, pixel conversion rate (based on provided scale), number of surfaces identified (e.g., dorsal, ventral, etc.); surface and scar area, maximum breadth and length, scar count (ordered by largest to smallest), percentage of surface area represented by scars, surface area of individual scars, and flaking angle of scars. `PyLithics` also produces a polygon count – a way of characterising the shape of each scar by approximating a polygon to it and counting how many sides it has. Output files can be used for statistical analyses with any software that can read JSON files.

Output images Figure 2 serve as validation of the output data Figure 3.

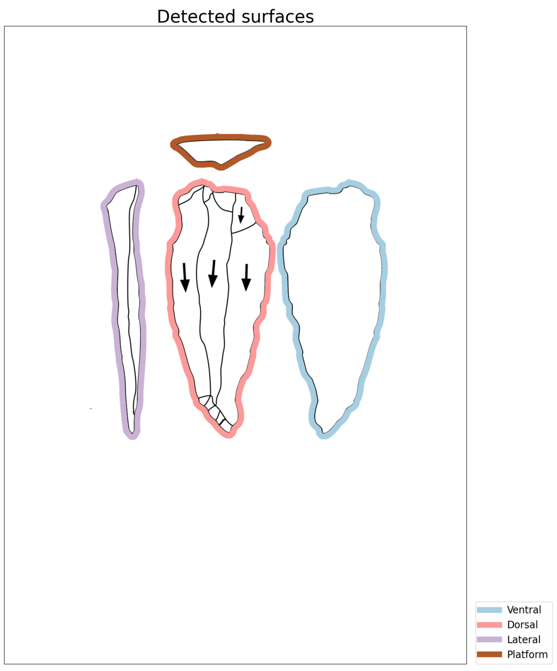


Figure 2: PyLithics output - detected surfaces.

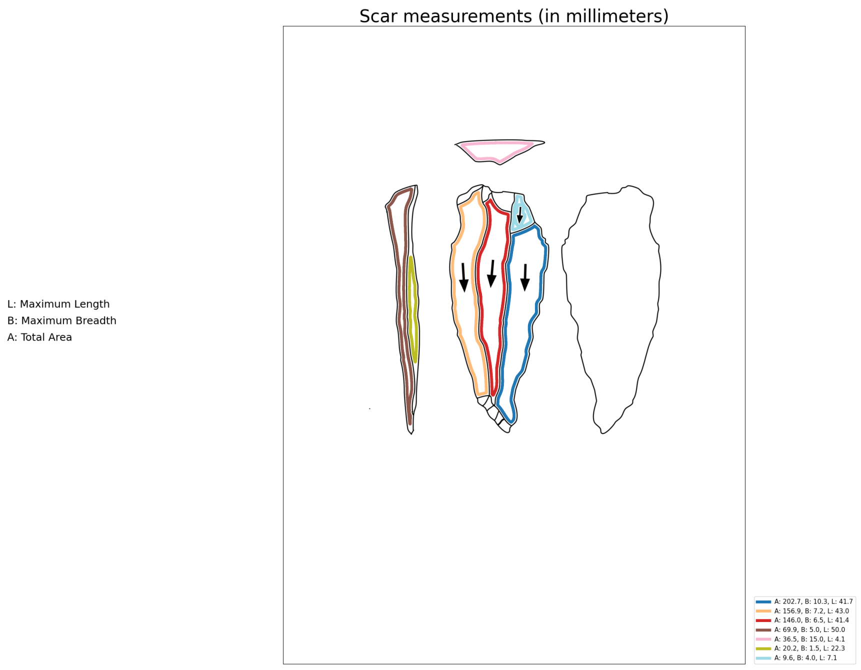


Figure 3: PyLithics output - detected scars.

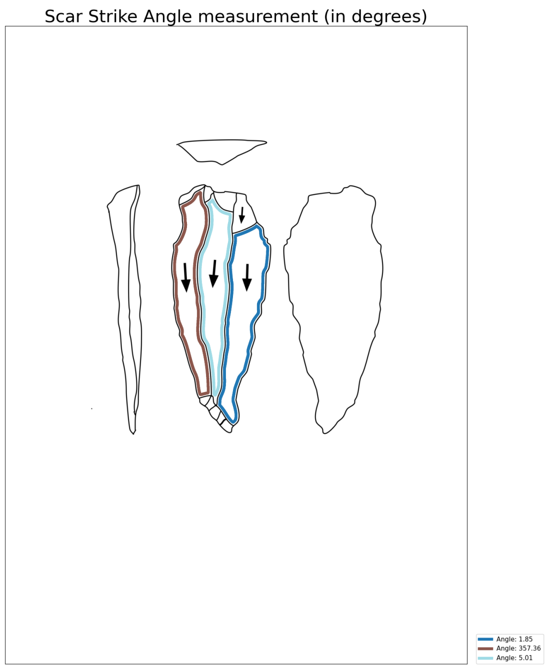
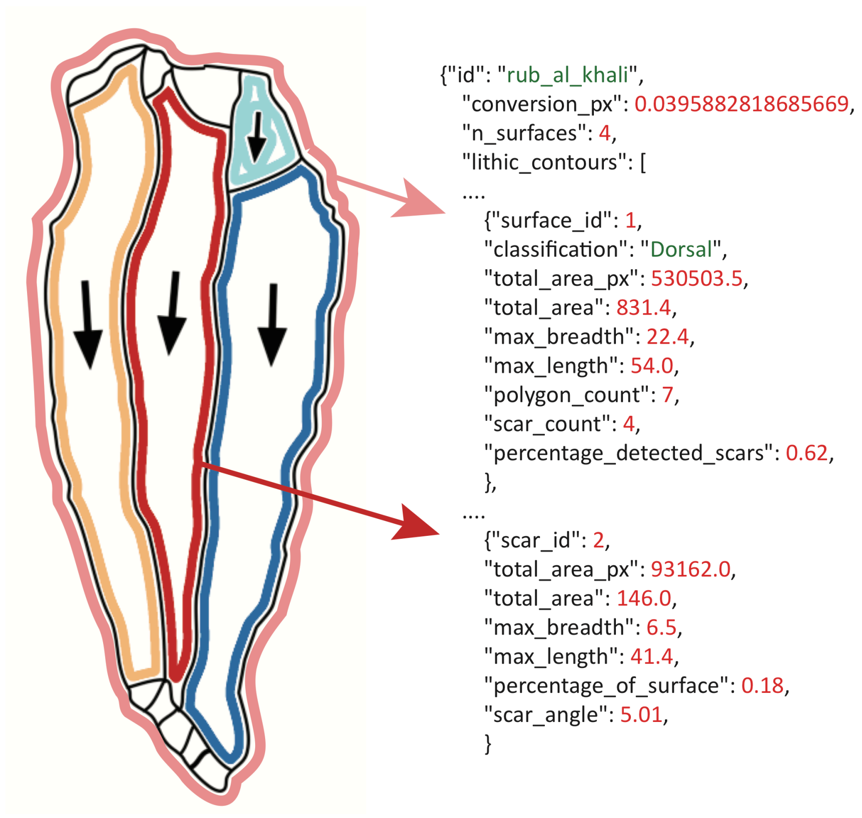


Figure 4: PyLithics output - flake angles.



```
{"id": "rub_al_khali",
   "conversion_px": 0.0395882818685669,
   "n_surfaces": 4,
   "lithic_contours": [
   ....
      {"surface_id": 1,
      "classification": "Dorsal",
      "total_area_px": 530503.5,
      "total_area": 831.4,
      "max_breadth": 22.4,
      "max_length": 54.0,
      "polygon_count": 7,
      "scar_count": 4,
      "percentage_detected_scars": 0.62,
      },
   ....
      {"scar_id": 2,
      "total_area_px": 93162.0,
      "total_area": 146.0,
      "max_breadth": 6.5,
      "max_length": 41.4,
      "percentage_of_surface": 0.18,
      "scar_angle": 5.01,
      }
```

Figure 5: PyLithics output - JSON data file.

Outlook

Evolutionary biology, and the study of human evolution in particular, has been transformed by the impact of genomics and the development of ancient DNA methodologies (Moody, 2004). One of the reasons that genomics has had such an impact is the sheer scale of the data now available, and power of the analytical techniques used. Although current approaches to lithic analysis have become more quantitative, they remain based on relatively univariate attribute assignments and limited metrics, variably collected and reported. `PyLithics` aims to expand data collection with the goal of building expansive, comprehensive, and standardized high-dimensional lithic artefact datasets for integration with genomic and fossil data.

Acknowledgements

The `PyLithics` package was developed by researchers from The Alan Turing Institute, University of Cambridge, and the Leverhulme Centre for Human Evolutionary Studies. Funding was provided by the Alan Turing Institute (grant number G109254). We would like to acknowledge support from Professor Katharine Robson Brown and Doctor Sebastian Ahnert.

References

Andrefsky,William, J. (2012). *Lithics: Macroscopic Approaches to Analysis*. Cambridge University Press. ISBN: 9780521849760

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

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

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Martingell, H., & Saville, A. (1988). *The Illustration of Lithic Artefacts: a guide to drawing stone tools for specialist reports* (p. 30). Association of Archaeological Illustrators & Surveyors. ISBN: 0951324608

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

Moody, G. (2004). *Digital code of life : how bioinformatics is revolutionizing science, medicine, and business*. Wiley. ISBN: 0471327883

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Mulbregt, P. van. (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