Automated rapid artefact surface area measurement from imagery with computer vision

Wesley J. Weatherbee [1], Jonathan Fowler [1], Danika van Proosdij [1]

1. Saint Mary's University, 923 Robie Street, Halifax, Nova Scotia. B3H 3C3 Canada.
Email: wesley.weatherbee@smu.ca; Fowler email: jonathan.fowler@smu.ca;
van Proosdij email: dvanproo@smu.ca

Abstract:

Automated surface area measurements have been of interest to archaeologists since digital imagery began allowing researchers to remotely collect artefact metrics. We present a method of automatically measuring 2D surface area from artefact planform images employing computer vision in Python. The Python script, provided as a .py file in supplementary data, creates boundaries around regions of relatively homogeneous pixels (artefacts) in the image. These bounded regions are called contours. A count of the number of pixels within each contour provides a surface area in pixels. A circular reference object provides a conversion factor for the contours, as well as a point of reference for geometric accuracy of outputs.

Measurements of 2D artefact surface area can be used in combination with measurements of length, width, thickness, and mass, or in some cases, replace such measurements. As presented, this technique provides utility to archaeology with applications to new documentation of artefacts, archived artefact images containing a scale, as well as landscape geoarchaeology and sedimentary contexts. Limitations of this type of surface area measurement include the requirement of the image background being of a solid colour heavily contrasting the artefacts being measured. Effectively, the background requirement limits deployment supporting collection of rapid field measurements from in-situ surface scatters without modification to the script or manipulation of the artefacts. Analytical applications utilizing this technique include studies of relative artefact abundance, shape and size class characterizations in artefact scatters, and redistribution of artefacts by geomorphological processes.

Keywords: imagery; automation; digital archaeology; surface area; archaeometry

1. Introduction and background

Automatic extraction of measurements and shape data from artefacts is not a new trend in archaeology. In the early 1990s, the Graphically Oriented Archaeological Database Project produced an environment allowing archaeologists to automatically extract and compare shape information from line drawings of artefacts, primarily pottery vessels (Lewis & Goodson 1991). Within a decade, this approach was employed by archaeologists studying lithic artefacts justifying the process as a means to extract information and metrics that may be hard to extract using a calliper and other traditional methods (McPherron & Dibble 1999).

Journal of Lithic Studies (2023) vol. 10, nr. 1, 14 p. DOI: https://doi.org/10.2218/jls.5623

Published by the School of History, Classics and Archaeology, University of Edinburgh
ISSN: 2055-0472. URL: http://journals.ed.ac.uk/lithicstudies/

Over two decades later, the pace of technological advancement has continued to increase, and digital cameras, laptops, tablets, and mobile phones are now commonplace in archaeological fieldwork. While adoption of digital technologies is widespread in archaeology, digital data collection methodologies have not enjoyed the same appreciation. Byrd & Owens (1997: 316) identify one factor impeding the adoption of new methods measuring surface area of artefacts stating that measuring "surface area … will not likely be adopted by archaeologists unless a relatively simple technique is developed for obtaining such data".

Here we present a step toward the technique forecasted by Byrd & Owens (1997) by automating the extraction of 2D surface area using computer vision from the OpenCV package in Python 3 (Bradski 2000; Van Rossum & Drake 2009). The method allows archaeologists to extract surface area measurements from an assemblage, collection, or sample of lithic flakes, or other relatively flat artefacts, using images. Images can contain many artefacts, and measurements are automatically output to a CSV file.

2. Methods

The script used to derive artefact surface area was modified from a post originally on PyImageSearch using computer vision to measure objects in images (Rosebrock 2016). The resulting method employs computer vision to automatically quantify the 2D surface area of experimentally produced lithic flakes in a planform image. User inputs for the script are the path to the input image and diameter of the circular reference object entered during command line execution of the script. Required materials (Table 1) include a circular reference object, camera, background of contrasting colour, artefacts, and a Python 3 installation including the packages.

A crucial requirement was the placement of a circular reference object with a known diameter at the left of the image (Figure 1). Initial measurements were converted from a count of pixels to user input units from the diameter of the reference object allowing subsequent measurements to use the same units. Our reference object is a Canadian Quarter with a reported diameter of 2.388 cm from Royal Canadian Mint technical specifications (25 Cents, Government of Canada, n.d.).

A selection of differentially shaped, experimental flakes was used to demonstrate the process of extracting surface area measurements using Python. The flakes consist of a light grey Georgetown flint with a white cortex. To contrast the flake material, a matte black background was selected. Flakes were placed to the right of the reference object in the image frame.

Figure 1: Input image of experimental flakes with circular reference at left.

2.1. Script

Python 3 code is located below, highlighting how the technique works. For a more detailed explanation on this technique, please visit the tutorials on DigitalArchNS (Weatherbee 2020) or PyImageSearch (Rosebrock 2016).

To employ the script, install the required packages (Table 1), navigate to the directory holding the python script within a command prompt window. Ensure the subdirectories "images" and "csv" exist, the "images" subdirectory contains the image to be measured, and that the diameter measurement of the circular reference object is known. To start measuring enter the following, modifying red text to reference user input:

python artifact-area.py --image [path to image] --width [diameter]

Where [path to image] is the path of the image files relative to the directory of the python script, and [diameter] is the diameter of the reference object in the desired unit of measurement. Note that a hardcoded threshold of 10000 pixels is set as a minimum size of objects measured by the script. This value can be modified to include smaller objects or exclude background noise from measurements. After entering this information, press the Enter key to begin measuring.

First, the input image is converted to a single channel composite greyscale image. Gaussian blur is executed twice prior to edge detection using an initial kernel of 9x9 followed by a 5x5 kernel iteration promoting smoothed edges while minimizing overall image blur. Edge detection is then applied to the composite greyscale image using the auto_canny method defined at the top of the script (Canny 1986; Rosebrock 2015b). Erode and dilate functions then decrease and increase the thickness of the contour in pixels, respectively. These functions help derive the contours from the input image, therefore providing information fundamental to the process of extracting surface area. Three iterations of the dilate function are followed by two iterations of erode to promote contour connectivity along artefact perimeters in the edge map. Contours are sorted from left to right, then enter a for loop measuring X, Y, and surface area. For each contour the user is required to press any key to cycle through an image displayed for

each contour. Once all contours are visualized, the script will append measurements to a CSV available in the 'csv' folder.

Table 1: Summary table of Python packages used in the script.

Packages				
Name	Summary	Reference	Installation	Alias
OpenCV	Open-source computer vision and machine learning library built to simplify and streamline adoption of machine learning and computer vision. *Version 4.1.2.30*	(Bradski 2000)	pip install opencv-python==4.5.5.62	cv2
SciPy	Open-source mathematical algorithm and convenience function library built on-top of NumPy. *Version 1.3.3*	(Virtanen *et al*. 2020)	pip install scipy==1.3.3	distance
imutils	Open-source convenience function and visualization library built on OpenCV, matplotlib, and NumPy. *Version 0.5.3*	(Rosebrock 2015a)	pip install imutils==0.5.3	perspective; contours
NumPy	Open-source scientific computing library fundamental to many Python workflows. *Version 1.17.4*	(Harris *et al*. 2020)	pip install numpy==1.17.4	np
ArgParse	Native Python library simplifying user input to scripts through CMD. *Python 3 Version 3.7.5*	(Van Rossum 2021)	n/a	argparse
CSV	Native Python library simplifying read and write of CSV files from multiple sources. *Python 3 Version 3.7.5*			csv
matplotlib	Open-source 2D graphics library for visualization and app development. *Version 3.1.2*	(Hunter 2007)	pip install matplotlib==3.1.2	n/a
pandas	Open-source data structure and statistical tools library designed to make scientific Python a more attractive and practical statistical computing environment for academic and industry practitioners alike (McKinney 2010, 56). *Version 0.25.3*	(McKinney 2010)	pip install pandas==0.25.3	pd

```
"""
Title: Automated Rapid Artifact Surface Area Measurement from Imagery using Computer Vision
Author: Wesley Weatherbee
Date: February 2020
Description: This script is intended to rapidly collect measurements
        from multiple artifacts in a single image using computer vision.

Modified from: https://www.pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/
Original Author: Adrian Rosebrock
```

```
 Date: March 28, 2016
"""

# USAGE
# open command-line (cmd.exe) and change the path to refer to the
# path where the file is loaded using: cd [path to directory here]
# after changing the directory, enter the following code in command-line:
# python artifact-area.py --image images/test_01.jpg --width 2.381

# import the necessary packages
from scipy.spatial import distance as dist
from imutils import perspective
from imutils import contours
import numpy as np
import argparse
import imutils
import cv2
import csv
import pandas as pd

# define the automatic canny edge detection method
def auto_canny(image, sigma=0.33):
    # compute the median of the single channel pixel intensities
    v = np.median(image)

    # apply automatic Canny edge detection using the computed median
    lower = int(max(0, (1.0 - sigma) * v))
    upper = int(min(255, (1.0 + sigma) * v))
    edged = cv2.Canny(image, lower, upper)

    # return the edged image
    return edged

# define midpoint method
def midpoint(ptA, ptB):
    return ((ptA[0] + ptB[0]) * 0.5, (ptA[1] + ptB[1]) * 0.5)

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
    help="path to the input image")
ap.add_argument("-w", "--width", type=float, required=True,
    help="width of the left-most object in the image (user defined measurement system)")
args = vars(ap.parse_args())

# load the image, convert it to grayscale, and blur it slightly, twice
image = cv2.imread(args["image"])
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (9, 9), 0)
gray = cv2.GaussianBlur(gray, (5, 5), 0)
```

```
# perform edge detection, then perform a dilation + erosion to
# close gaps in between object edges
edged = auto_canny(gray)
edged = cv2.dilate(edged, None, iterations=3)
edged = cv2.erode(edged, None, iterations=2)

# find contours in the edge map
cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
   cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)

# sort the contours from left-to-right and initialize the
# 'pixels per metric' calibration variable
(cnts, _) = contours.sort_contours(cnts)
pixelsPerMetric = None

# create list object
measure = []

# loop over the contours individually
for c in cnts:
   # if the contour is not sufficiently large, ignore it
   if cv2.contourArea(c) < 10000:
      continue

   # compute the rotated bounding box of the contour
   orig = image.copy()
   box = cv2.minAreaRect(c)
   box = cv2.cv.BoxPoints(box) if imutils.is_cv2() else cv2.boxPoints(box)
   box = np.array(box, dtype="int")

   # order the points in the contour such that they appear
   # in top-left, top-right, bottom-right, and bottom-left
   # order.
   box = perspective.order_points(box)

   # unpack the ordered bounding box, then compute the midpoint
   # between the top-left and top-right coordinates, followed by
   # the midpoint between bottom-left and bottom-right coordinates
   (tl, tr, br, bl) = box
   (tltrX, tltrY) = midpoint(tl, tr)
   (blbrX, blbrY) = midpoint(bl, br)

   # compute the midpoint between the top-left and top-right points,
   # followed by the midpoint between the top-right and bottom-right
   (tlblX, tlblY) = midpoint(tl, bl)
   (trbrX, trbrY) = midpoint(tr, br)

   # compute the Euclidean distance between the midpoints
```

```
    dA = dist.euclidean((tltrX, tltrY), (blbrX, blbrY))
    dB = dist.euclidean((tlblX, tlblY), (trbrX, trbrY))

    # if the pixels per metric has not been initialized, then
    # compute it as the ratio of pixels to supplied metric
    # (in this case, inches)
    if pixelsPerMetric is None:
        pixelsPerMetric = dB / args["width"]

    # computer centre of the contour
    M = cv2.moments(c)
    cX = int(M["m10"] / M["m00"])
    cY = int(M["m01"] / M["m00"])

    # compute the size and surface area of the object
    dimA = (dA / pixelsPerMetric)
    dimB = (dB / pixelsPerMetric)
    SA = ((cv2.contourArea(c) / pixelsPerMetric) / pixelsPerMetric)

    # add measurements to list
    measure.append((dimA, dimB, SA))

    # draw contours in red
    cv2.drawContours(orig, [c.astype("int")], -1, (0, 0, 255), 2)

    # draw the object area on the image
    cv2.putText(orig, "{:.2f}sqcm".format(SA),
        (int (cX), int (cY)), cv2.FONT_HERSHEY_SIMPLEX,
            0.65, (0, 0, 0), 3)
    cv2.putText(orig, "{:.2f}sqcm".format(SA),
        (int (cX), int (cY)), cv2.FONT_HERSHEY_SIMPLEX,
            0.65, (255, 255, 255), 2)

    # show the output image
    origS = cv2.resize(orig, (1536, 1024))
    cv2.imshow("Image", origS)
    cv2.waitKey(0)

# take X, Y, and surface area measurements and append them to a CSV
col_titles = ('X', 'Y', 'SA')
data = pd.np.array(measure).reshape((len(measure) // 1, 3))
pd.DataFrame(data, columns=col_titles).to_csv("csv/Measurements.csv", index=False)
```

3. Results

Running the script results in contours being created around each artefact in the image, from which surface area measurements are extracted. Our experimental results indicate an error of 2.23% in the geometric measurements of surface area using the equation for percent error (PercentError = ((Experimental - Expected) x 100) / Expected). This calculation was based on an expected surface area of our circular reference scale (Canadian quarter) being 4.48 cm^2

compared to the experimental surface area of 4.58 cm^2 obtained by our method. The expected value for surface area of a quarter is calculated with the diameter measurement provided by the Royal Canadian Mint ("25 Cents Coins," The Royal Canadian Mint, n.d.). The experimental value for surface area is provided by the contours in the script. This percent error can be attributed to camera angle misalignment or lens distortion and is used to calculate the uncertainty of surface area values in Table 2.

The pixelsPerMetric value is a count of pixels corresponding to the user input diameter of the circular reference object. Our experimental results report 71.19pixels per 2.388 cm, a value that will change depending on resolution of the image and distance to the objects being measured. This is equal to a pixel dimension of 0.034cm along each side meaning each pixel reflects a real-world area of 1.13×10^{-3} cm^2.

The output CSV file holds measurements of surface area, and X and Y dimensions of each object (Table 2). The X and Y values are measured in centimetres, but do not equate to length and width values measured using a calliper (Andrefsky 2005: 99-101). Rather, these values represent the minimum and maximum length and width of the bounding box of minimum dimensions fitting the contour. Similar measurements have been considered to represent the intermediate axis of flakes, in prior lithic research (Brown 2001; Cadieux 2013). In some cases, these values are similar, but should not be used interchangeably.

Table 2: Output tabular results from experimental data. User entered diameter bolded. SA = surface area.

X (cm)	Y (cm)	SA (cm^2)
2.430	2.388	4.58 ±0.10
2.345	5.644	10.29 ±0.23
2.717	1.841	3.69 ±0.08
1.633	4.721	5.71 ±0.13
1.299	2.172	2.15 ±0.05
2.660	1.642	3.56 ±0.08
2.306	2.497	3.80 ±0.08
2.644	2.079	3.72 ±0.08
1.308	1.968	2.10 ±0.05
4.244	3.077	10.33 ±0.23
3.947	2.306	6.89 ±0.15
1.637	2.299	2.80 ±0.06
3.891	2.338	6.51 ±0.15
4.127	4.705	13.95 ±0.31
2.559	3.085	5.77 ±0.13
4.325	3.421	10.27 ±0.23
2.917	4.763	8.96 ±0.20
1.691	1.949	2.37 ±0.05
1.140	3.286	2.63 ±0.06
4.544	2.476	7.71 ±0.17
1.898	4.649	6.06 ±0.14
1.942	1.951	2.66 ±0.06
2.094	5.013	7.47 ±0.17
3.655	1.891	4.45 ±0.10
4.160	2.340	6.56 ±0.15

4. Applications & Limitations

We have presented an automated, rapid technique for extracting surface area measurements from artefacts. This technique is lightweight, running on a Python script that can be executed in a command prompt using in-line arguments. The script instructs the computer to translate the image to greyscale, blur the image, perform Canny edge detection using thresholds defined by the range of values within the image, place contours around relatively homogeneous groupings of pixels fitting a size criterion, and output the measurements (Figure 2) to a CSV file.

Our method is powerful, yet simple, and with careful consideration, has a variety of archaeological applications, including extracting information from scans of artefact images in published literature. Integration of this technique within larger, more advanced morphometric workflows in archaeological research (*e.g.*, Barceló 2010), and the addition of camera calibration (see: Sadekar & Mallick 2020) are practical next steps for this technique. The applications of the method presented not only apply to archaeology and artefact analysis, but can percolate through the disciplines of geography and geology ─ from where Byrd & Owens (1997) drew their inspiration.

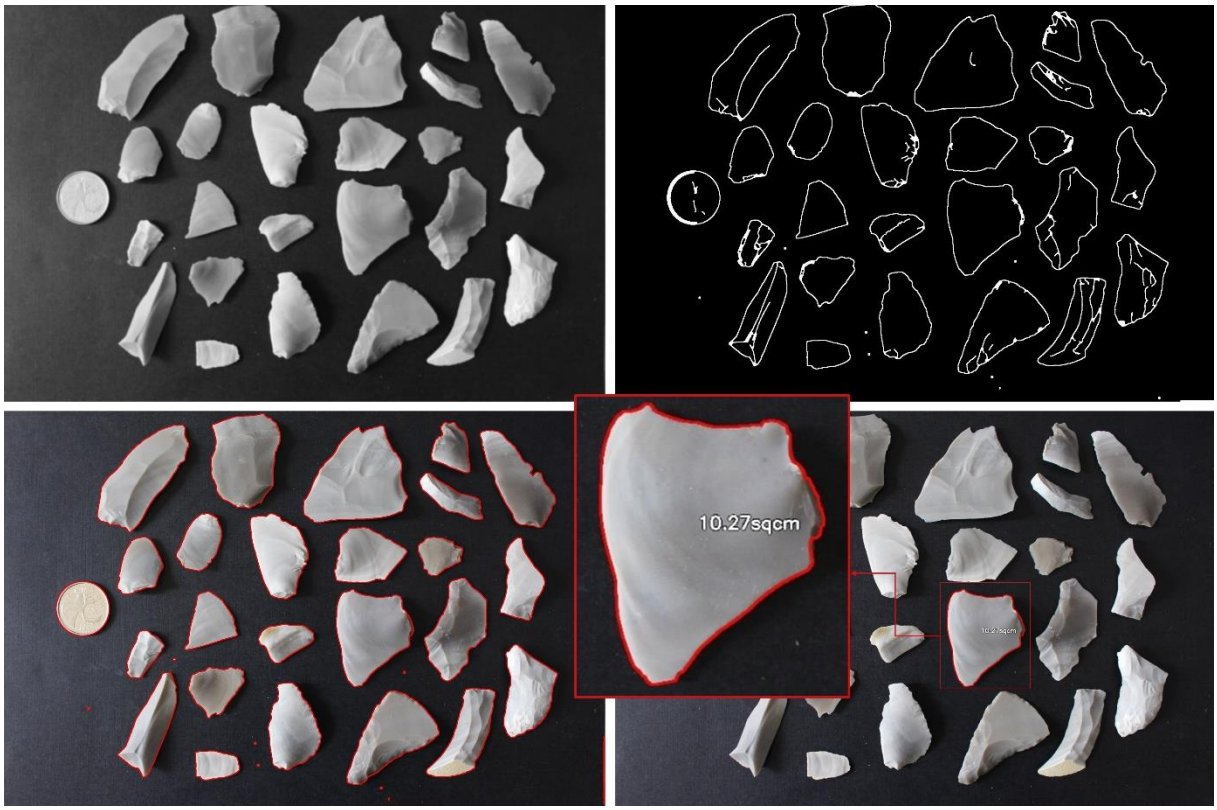

Figure 2: Process of surface area extraction from an image. Top Left - blurred composite greyscale image. Top Right – edge detection after dilation and erosion. Bottom Left – contours identified in the image using computer vision. Bottom Right & Inset – calculated surface area measurement of an artifact from the contour.

Automated, rapid measurement of surface area also identifies a point of intersection for geologic, geographic, and archaeological research along past and present coastline contexts by considering how object shape influences its relocation by geomorphic processes. The intersection is expressed in emerging relational studies of geomorphology and archaeology known as landscape geoarchaeology (Contreras 2010; Contreras 2017; Holliday 2009; Holliday *et al*. 2019;). Flake shape often resembles that of beach shingles found within the swash-zone of coastal systems having a relatively small mass compared to 2D surface area. By creating a

way of readily quantifying the geometry of swash-zone gravel shingles, geographers will be able to create more thorough conceptual and numerical models of how materials such as beach shingles and flakes are transported in coastal areas. Research in such areas will certainly provide insights into geoarchaeological problems of taphonomy, site-formation processes, and post-depositional movements of artefacts within coastal environments.

Deploying this tool on larger assemblages with well-defined spatial provenience can give insights into the relationships that surface area holds in relation to well-defined expected distributions of debitage in flintknapping areas (Kvamme 1988; 1997; 1998). Surface area can be used in combination with object mass, and material mass density to estimate the thickness and volume of materials (Cadieux 2013). Both applications present novel ways to observe patterns in spatial and non-spatial assemblages through attributes too complex to compute from traditional measurement techniques.

This technique does not come without some limitations. The ability of this method to measure objects comes from an apparent contrast between object and background. This means that employing this method on images with varying material types may provide inaccurate measurements between objects. One approach to solve this problem is to separate each assemblage based on material type. This way, consistent measurements and associated errors can be obtained for each material class.

Understanding what the measurements represent clarifies this method's limitations and applicability to other problems. The distinction between 2D and 3D surface area is key. A 2D surface area is measured only using the horizontal axis, while a 3D surface area uses the vertical (depth) axis in addition to horizontal. This effectively means a 2D surface area represents a footprint of an artefact viewed in planform. The relevance of this distinction is that our 2D surface area measurement is in fact a generalized potential area for lift or drag forces to be applied to the object in conceptualizing artefact transportation whether it be due to waves on a beach, currents in a river, or striking flakes from a core.

5. Conclusions

The opportunities for research provided by this technique are vast. The convenience offered by our automated, rapid artefact surface area measurement technique is instrumental in its service of accelerating data collection and analysis. Techniques such as this bridge digital analysis skills and computer literacy with the material culture remains of the past. By providing links to blog posts inspiring this publication, we are presenting sources of information that can be used to develop computer literacy in relation to programming and analysis.

Though still quite rudimentary, this technique could be applied as one step in a much larger research workflow. Automated metric extraction from unpublished images, and archival scans of artefacts offers an avenue for archival projects to benefit from this technique as well. The extracted contours could perceivably be one input layer into an algorithm extracting points along the contour for typological or morphometric analysis from an image. Alternatively, surface area used as a proxy for lift and drag forces may be able to help identify a feeder source for artefacts redistributed within coastal areas by longshore drift. Countless present and future applications are presented by this technique, and the authors encourage input into future developments by interested parties in any discipline.

Acknowledgements

Authors would like to acknowledge the author of the original post and script inspiring this application and publication, Adrian Rosebrock. Without the effective communication of computer programming topics at PyImageSearch this may not have been possible. The corresponding author would also like to express his gratitude for family, friends, colleagues,

and Saint Mary's University for facilitating an environment allowing this idea to come to fruition, even during COVID-19 restrictions.

Data accessibility statement

All data and hosted repository are licensed by the corresponding author under a derivative of the original license available at PyImageSearch.com. Data is hosted at Codeberg and indexed on Zenodo. Please cite use of supplementary data in publications employing or modifying this technique with the following citation:

Weatherbee, Wesley J., Fowler, Jonathan, and van Proosdij, Danika. 2021. "Supplementary Data: Automated Rapid Artefact Surface Area Measurement from Imagery with Computer Vision". Zenodo. http://doi.org/10.5281/zenodo.4792203.

List of supplementary files

Supplementary data for this article is available as an organized directory containing the Python script, input, and output data used in this article:

Supplementary data repository
Supplementary Data: artifact-area-v0.1

This repository contains Supplementary Data accompanying the publication *Automated Rapid Artefact Surface Area Measurement from Imagery with Computer Vision* submitted to the *Journal of Lithic Studies*. Please download the complete repository for easiest deployment of the script.

References

Government of Canada n.d., *25 Cents*. royal canadian mint. Retrieved October 17, 2023. URL: https://www.mint.ca/en/discover/canadian-circulation/25-cents

Andrefsky, W. 2005, *Lithics: macroscopic approaches to analysis*. (Second ed.) Cambridge Manuals in Archaeology. Cambridge, UK: Cambridge University Press, 301 p. DOI: https://doi.org/10.1017/CBO9780511810244

Barceló, J.A. 2010, Visual analysis in archaeology. An artificial intelligence approach. In: *Morphometrics for nonmorphometricians* (Ashraf M.T., Ed.), Lecture notes in Earth Sciences Vol. 124. Springer, Berlin Heidelberg: p. 93-156. DOI: https://doi.org/10.1007/978-3-540-95853-6

Bradski, G. 2000, The OpenCV library. *Dr. Dobb's Journal of Software Tools* 120: 122–25.

Brown, C.T. 2001, The fractal dimensions of lithic reduction. *Journal of Archaeological Science* 28(6): 619–31. DOI: https://doi.org/10.1006/jasc.2000.0602

Byrd, J.E. & Owens, D.D. 1997, A method for measuring relative abundance of fragmented archaeological ceramics. *Journal of Field Archaeology*, 24(3): 315–20. DOI: https://doi.org/10.1179/009346997792208168

Cadieux, N. 2013, Size matters: measuring debitage area and getting it right with a digital scanner. *Lithic Technology*, 38(1): 46–70. DOI: https://doi.org/10.1179/0197726113Z.0000000004

Canny, J. 1986, A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8(6): 679–98. DOI: https://doi.org/10.1109/TPAMI.1986.4767851

Contreras, D.A. 2010, Reconstructing an engineered environment in the central andes: landscape geoarchaeology at Chavín de Huántar, Peru. In: *The Archaeology of Anthropogenic Environments* (Dean, R. Ed.), Occasional Paper, No. 37. Southern Illinois University Carbondale Center for Archaeological Investigations, Carbondale, Illinois: p. 225-249.

Contreras, D.A. 2017, (Re)Constructing the sacred: landscape geoarchaeology at Chavín de Huántar, Peru. *Archaeological and Anthropological Sciences*, 9(6): 1045–57. DOI: https://doi.org/10.1007/s12520-014-0207-2

Harris, C.R., Jarrod Millman, K., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., et al. 2020, Array programming with NumPy. *Nature*, 585: 357–62. DOI: https://doi.org/10.1038/s41586-020-2649-2

Holliday, V.T. 2009, Geoarchaeology and the search for the first americans. *CATENA, Developments in International Geoarchaeology*, 78(3): 310–22. DOI: https://doi.org/10.1016/j.catena.2009.02.014

Holliday, V.T., Harvey, A., Cuba, M.T. & Weber, A.M. 2019, Paleoindians, paleolakes and paleoplayas: landscape geoarchaeology of the Tularosa Basin, New Mexico. *Geomorphology*, 331: 92–106. DOI: https://doi.org/10.1016/j.geomorph.2018.08.012

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

Kvamme, K.L. 1988, A simple graphic approach and poor man's clustering technique for investigating surface lithic scatter types. *Plains Anthropologist*, 33(121): 385–94. DOI: https://doi.org/10.1080/2052546.1988.11909418

Kvamme, K.L. 1997, Patterns and models of debitage dispersal in percussion flaking. *Lithic Technology*, 22 (2): 122–38. DOI: https://doi.org/10.1080/01977261.1997.11754538

Kvamme, K.L. 1998, Spatial structure in mass debitage scatters. In: *Surface Archaeology*, (Sullivan, A.P. Ed.), University of New Mexico Press, Albuquerque: p. 127-141.

Lewis, P.H. & Goodson, K. J. 1991, Images, databases and edge detection for archaeological object drawings. In: *Computer Applications and Quantitative Methods in Archaeology 1990*. (Rahtz, S. & Lockyear, K. Eds.), BAR International Series Vol. 565, Tempus Reparatum, Oxford: p. 149-153. URL: https://eprints.soton.ac.uk/250828/1/Lewis%2526Goodson.pdf

McKinney, W. 2010, Data structures for statistical computing in Python. In: *Proceedings of the 9th Python in Science Conference*. (van der Walt, S. & Millman, J. Eds), SciPy Organizers, Austin, Texas: p. 56–61. DOI: https://doi.org/10.25080/Majora-92bf1922-00a

McPherron, S.P. & Dibble, H.L. 1999, stone tool analysis using digitized images: examples from the Lower and Middle Paleolithic. *Lithic Technology*, 24(1): 38–52. DOI: https://doi.org/10.1080/01977261.1999.11720944

Rosebrock, A. 2015a, *My imutils package: a series of opencv convenience functions*. PyImageSearch (blog). Retrieved: February 17, 2020.

URL: https://www.pyimagesearch.com/2015/02/02/just-open-sourced-personal-imutils-package-series-opencv-convenience-functions/

Rosebrock, A. 2015b, *Zero-Parameter, automatic canny edge detection with Python and OpenCV*. PyImageSearch (blog). Retrieved: February 17, 2020. URL: https://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/

Rosebrock, A. 2016, *Measuring size of objects in an image with opencv*. PyImageSearch (blog). Retrieved: February 17, 2020. URL: https://www.pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/

Van Rossum, G. 2021, *Python 3.7 reference manual*. Python Software Foundation, Beaverton, Oregon. Accessed: October 17, 2023. URL: https://docs.python.org/3.7/download.html.

Sadekar, K. & Mallick, S. 2020, *Camera calibration using OpenCV*. Learn OpenCV (blog). 2020. URL: https://learnopencv.com/camera-calibration-using-opencv/

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., et al. 2020, SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods,* 17: 261–72. DOI: https://doi.org/10.1038/s41592-019-0686-2

Weatherbee, W. 2020, *A quarter and a camera ─ measuring relative abundance of artifacts with computer vision in Python*. DigitalArchNS (blog). Retrieved: February 18, 2020. URL: https://github.com/weslyfe/weslyfe.github.io/blob/master/_posts/2020-02-18-measureartifacts.md

Mesure automatisée et rapide de la surface d'un artefact à partir d'images issues de la vision par ordinateur

Wesley J. Weatherbee [1], Jonathan Fowler [1], Danika van Proosdij [1]

1. Saint Mary's University, 923 Robie Street, Halifax, Nova Scotia. B3H 3C3 Canada.
Email: wesley.weatherbee@smu.ca; Fowler email: jonathan.fowler@smu.ca;
van Proosdij email: dvanproo@smu.ca

Abstract:

Les mesures automatisées de surface intéressent les archéologues depuis que l'imagerie numérique a commencé à permettre aux chercheurs de collecter à distance des données sur les artefacts. Nous présentons une méthode de mesure automatique de la surface 2D à partir d'images de formes d'artefacts en utilisant un déploiement Python 3 de vision par ordinateur. Le script Python, disponible sous la forme d'un fichier .py dans les données supplémentaires, crée des limites autour des régions de pixels relativement homogènes (artefacts) dans l'image. Ces limites de régions sont appelées contours. Un décompte du nombre de pixels à l'intérieur de chaque contour fournit une superficie en pixels. Un objet de référence circulaire fournit un facteur de conversion pour les contours, ainsi qu'un point de référence pour la précision géométrique des résultats.

Les mesures de la surface des artefacts en 2D peuvent être utilisées en combinaison avec les mesures de la longueur, de la largeur, de l'épaisseur et de la masse ou, dans certains cas, remplacer ces mesures. Telle qu'elle est présentée, cette technique est utile à l'archéologie en s'appliquant à la nouvelle documentation des artefacts, aux images d'artefacts archivées contenant une échelle, ainsi qu'à la géoarchéologie du paysage et aux contextes sédimentaires. Les limites de ce type de mesure de surface exigent l'utilisation d'un fond d'image de couleur unie contrastant fortement avec les artefacts mesurés. En effet, l'exigence d'un arrière-plan limite le déploiement de la collecte de mesures rapides sur le terrain à partir de dispersions de surface in situ sans modification du script ou manipulation des artefacts. Les applications analytiques utilisant cette technique comprennent des études sur l'abondance relative des artefacts, la caractérisation des classes de forme et de taille dans les dispersions d'artefacts et la redistribution des artefacts par les processus géomorphologiques.

Keywords: imagerie; automatisation; archéologie numérique; surface; archéométrie