



Sep 22, 2021

outlineR: Artefact Processing and Extraction Protocol

Forked from [outlineR: Artefact Processing and Extraction Protocol](#)David Nicolas Matzig¹¹Department of Archaeology and Heritage Studies, Aarhus University, Denmark

1 Works for me

Share

dx.doi.org/10.17504/protocols.io.bygaptse

David Matzig

Department of Archaeology and Heritage Studies, Aarhus Unive...

DISCLAIMER

THE PROTOCOL IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

ABSTRACT

Geometric morphometric methods (GMM) in archaeology are experiencing a sharp increase in application and popularity since the last decade or so and seem to be more popular now than ever. In general, they constitute a major advance vis-à-vis earlier qualitative descriptions, typological assessment, or linear measurements of artefacts. GMM approaches can be divided into methods that use landmarks, and those that use trigonometric descriptions of whole outlines. The bulk of archaeological applications of GMM have so far relied on landmark-based approaches, although a surge of recent studies is demonstrating the utility of whole-outline approaches using so-called elliptical Fourier analysis and cognate approaches. There currently exist various standalone software applications as well as some R-packages for the extraction and analysis of landmarks and whole-outlines. However, the extraction step always involves a considerable amount of manual processing and manual tracking of either the landmarks or whole-outlines, which proves to be the definite bottleneck of many studies. In this protocol I introduce the R-package *outlineR* (Matzig 2021) that allows a fast and efficient extraction of whole-outlines from multiple artefacts on images, as well as all necessary preparatory steps that lead up to it.

References

- Barthelme et al. 2020:** Barthelme, S., Tschumperle, D., Wijnfjels, J., Assemlal, H. E., & Ochi, S. (2020). imager: Image Processing Library Based on "CImg" (0.42.3) [Computer software]. <https://CRAN.R-project.org/package=imager>
- Bonhomme et al. 2014:** Bonhomme, V., Picq, S., Gaucherel, C., & Claude, J. (2014). Momocs: Outline Analysis Using R. Journal of Statistical Software, 56(13). <https://doi.org/10.18637/jss.v056.i13>
- Matzig 2021:** outlineR: An R package to derive outline shapes from (multiple) artefacts on JPEG images. Zenodo. <https://doi.org/10.5281/ZENODO.4527469>
- Pau et al. 2010:** Pau, G., Fuchs, F., Sklyar, O., Boutros, M., & Huber, W. (2010). EBImage—An R package for image processing with applications to cellular phenotypes. Bioinformatics, 26(7), 979–981. <https://doi.org/10.1093/bioinformatics/btq046>

DOI

dx.doi.org/10.17504/protocols.io.bygaptse

EXTERNAL LINK

<https://github.com/yesdavid/outlineR>

PROTOCOL CITATION

David Nicolas Matzig 2021. outlineR: Artefact Processing and Extraction Protocol. **protocols.io**
<https://dx.doi.org/10.17504/protocols.io.bygaptse>



FORK NOTE

added file.path() command to be able to work on different operating systems

FORK FROM

Forked from outlineR: Artefact Processing and Extraction Protocol, David Matzig

KEYWORDS

R, Archaeology, Stone Tools, Geometric Morphometrics, Outline Analysis, Image processing, Momocs, outlineR, Rstats, Lithic Studies, Lithics, 2D, Digitization, Digital Humanities

LICENSE

————— This is an open access protocol distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

CREATED

Sep 22, 2021

LAST MODIFIED

Sep 22, 2021

PROTOCOL INTEGER ID

53474

MATERIALS TEXT

Morar Quartz Industry (Wellcome Collection, <https://wellcomecollection.org/works/th7egtj>, CC BY 4.0)

SAFETY WARNINGS

Developed under Ubuntu 18.04

DISCLAIMER:

THE PROTOCOL IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Workspace Set-up

- 1 For this project, create a new folder on your computer. In the following protocol, this project folder will be referred to as "/project_folder".

- 1.1 Inside of "/project_folder" create the following subfolders:
 - one for the raw images ("/project_folder/raw"),
 - one for the prepared images ("/project_folder/prep"),
 - one for the single artefacts as a subfolder of the prepared images folder ("/project_folder/prep/single").

Expected folder structure:

```
|
├─ project_folder/
│   └─ raw/      # raw image data
│       └─ prep/  # prepared image data
│           └─ single/ # single artefacts
```

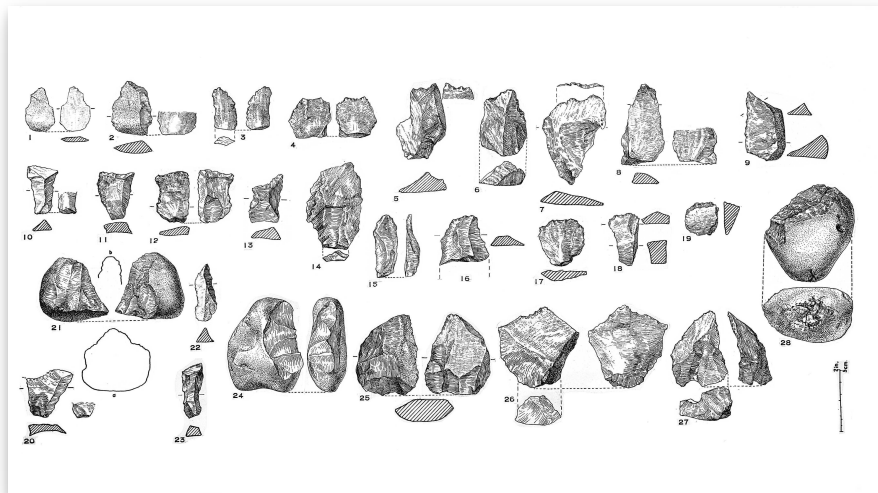
- 1.2 For the purpose of this protocol, we are going to use a picture of Morar Quartz Industry from [Wellcome Collection](https://wellcomecollection.org/works/th7eqtfj) (<https://wellcomecollection.org/works/th7eqtfj>; CC BY 4.0). The image serves as proxy for a typical figure in a catalogue of an archaeological publication.

Save it under "/project_folder/raw" as "morar_quartz_industry.jpg".



When saving your raw image(s) make sure to save them in ".jpg" format with no special characters or empty spaces in their filename.

Morar Quartz Industry (CC BY 4.0)



Morar Quartz Industry (<https://wellcomecollection.org/works/th7eqtfj>; CC BY 4.0)

Image Preparation

- 2 Typically, figures of archaeological (stone) artefacts contain scale bars, numberings, and other descriptions or details, or in the case of photographs, heterogenous backgrounds. This section describes, how to manually remove all elements in a picture, that do not belong to the artefact itself.

For every raw image:

2.1 Open GIMP

GNU Image Manipulation Program (GIMP) 2.8 [↗](#)

Ubuntu 18.04

by GIMP Development Team

- 2.2 Load raw, unprepared picture ("/project_folder/raw/morar_quartz_industry.jpg").
- 2.3 Remove all numbering, scales, text, ventral and lateral views, incomplete artefacts, etc. using i.e. the **Rectangle select tool** in GIMP in combination with the "delete" key on your computer's keyboard.
- 2.4 Check if picture is in grayscale. Go to:
Image → Mode → Grayscale
- 2.5 Threshold/binarize the image to get clear, thick lines around the artefacts. Go to:
Color → Threshold

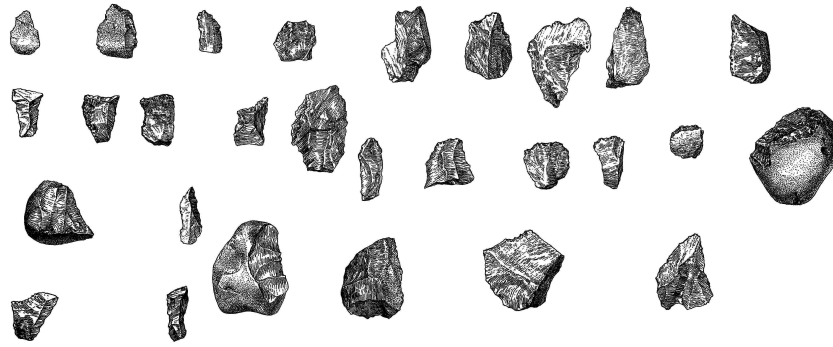


Check, whether there is some pixels in the picture outside of the artefacts that got created in the process of thresholding. If so, remove them.




Check whether all lines around the artefacts are closed. If not, use the **Pencil tool** with **Hardness 100** and **Pixel Size 1** to fill the gaps.


- 2.6 Expected result:



Prepared image of Morar Quartz Industry (<https://wellcomecollection.org/works/th7egtftj>; CC BY 4.0)

2.7 Save the prepared image under "/project_folder/prep" using the same filename as the raw image.

 Save at 100% detail

 Do not change the size/scaling/dpi/... of the picture in any of the steps!

Scaling Factor - Create TPS File

3


Build a .tps file into which all filenames of the prepared images are written. This file is needed for **Step 4: Scaling Factor - Get Scaling Factor**.

This step requires tpsUtil32 by F. James Rohlf (<http://www.sbmorphometrics.org/soft-utility.html>).

tpsUtil32 1.81 

Microsoft Windows
by F. James Rohlf

3.1 Start tpsUtil and navigate to:
Operations → Build tps file from images

 Such a .tps file must only be build, if all pictures in the respective folder are definite and no

images are added or deleted.
Ideally, the .tps file is build after all images have been prepared.

- 3.2 In the field **Input directory** click on **Input**:
 1. navigate to “/project_folder/raw”,
 2. select the first (or any) picture,
 3. under **Files of type** select **JPEG Bitmap**,
 4. click **Open**.
- 3.3 In the field **Output file** click on **Output**:
 1. navigate to “/project_folder/raw”,
 2. create a .tps file with a sensible name that uses underscores instead of empty spaces, and no special characters.
- 3.4 In the field **Actions** click on **Setup**:
 1. make sure to **Include all**,
 2. do not **include path**,
 3. **Create**.

Scaling Factor - Get Scaling Factor

4

With tpsDig we are going to derive the pixel-to-cm scaling factor, using the scale bars on the raw images.

tpsDIG2w32 2.31

Microsoft Windows

by F. James Rohlf

- 4.1 Load the .tps file:
File → Input source → File
and select the .tps file created in tpsUtil in **Step 3: Scaling Factor - Create TPS File**.
- 4.2 For each image, set the pixel to centimeter ratio.
In the “Image Tools” window (**Options → Image tools...**):
 1. **Set scale**,
 2. click on the starting point and the end point of the scale in the image,
 3. type in the length of the measured scale/reference length,
 4. make sure it's the correct measure (i.e. centimeters, millimeters,...),
 5. click **OK**.
 6. If there is more than one raw image to scale, go to the next image by clicking on the **red arrow** pointing to the right in the top left corner.
- 4.3 **File → save data**

outlineR

- 5 The R package *outlineR* (Matzig 2021) is a helpful wrapper around functions from mainly the *Momocs* (v. 1.3.0; Bonhomme et al. 2014), *EBImage* (v. 4.28.1; Pau et al. 2010), and *imager* (v. 0.42.3; Barthelme et al. 2020) packages. It is designed for the fast and easy extraction of single outline shapes from images containing multiple thereof, such as

the one we prepared in the steps above.

To use the package we require a current version of the statistical programming language R ($\geq 3.6.3$; R Core Team 2020). An integrated development environment such as RStudio is recommended.

R $\geq 3.6.3$ 

by R Core Team (2020)

RStudio $\geq 1.2.5033$ 

by RStudio Team (2020)

Inside of R, outlineR can be installed via the following command:

```
# from https://github.com/yesdavid/outlineR
remotes::install_github("yesdavid/outlineR",
                        dependencies = TRUE)
```

- 5.1 Start Rstudio and create a new project:
1. **File \rightarrow New Project... \rightarrow Existing Directory**,
 2. navigate to `"/project_folder"`,
 3. **Create Project**.

- 5.2 Inside Rstudio, open an R script:
File \rightarrow New File \rightarrow R Script.
Inside the R script we will execute all following commands.

- 5.3 Load outlineR:

```
library(outlineR)
```

- 5.4 Define paths:

```
# Define where the prepared images containing multiple artefacts are right now.
inpath <- file.path(".", "prep")

# Define where the single artefacts should be saved.
outpath <- file.path(".", "prep", "single")
```



The paths as depicted here are relative paths. If there is no RStudio project set-up in the "project_folder"-folder as described above, the absolute (full) paths to the `"/prep/"` and `"/single/"` directories has to be provided instead.

5.5 Extract the artefacts from the image(s):

```
outlineR::separate_single_artefacts(inpath = inpath,  
                                     outpath = outpath)
```

Afterwards, the images of the single artefacts should be saved in the `./prep/single` folder which was defined as `outpath` in the code. The single artefact images get individual names based on the original file name and an extension consisting of `"_pseudo_no_"` and a consecutive number. The extraction process does not follow a left-to-right, top-to-bottom scheme and therefore the pseudo numbers will not reflect any (potential) prior numbering on the raw image.



If there is empty images in the `./prep/single` folder, re-check the prepared images in `./prep` for single pixels outside the artefacts or open outlines. If necessary, delete all images in `outpath`. Then, re-run this command.

5.6 Import the .tps-file:

```
tps_df <- outlineR::tps_to_df(file.path("path", "to", "file.tps"))
```



If no `.tps`-file exists, skip this step.

5.7 Extract the outlines of all separated artefacts and combine them with their scaling factor:

```
single_outlines_list <- outlineR::get_outlines(outpath = outpath,  
                                                tps_file_rescale = tps_df)
```

If no `.tps`-file exists, run the following command:

```
single_outlines_list <- outlineR::get_outlines(outpath = outpath,  
                                                tps_file_rescale = NULL)
```

5.8 Combine the single outlines into a common "Out" file (a specific filetype from the Momocs R package):

```
outlines_combined <- outlineR::combine_outlines(single_outlines_list =  
single_outlines_list)
```

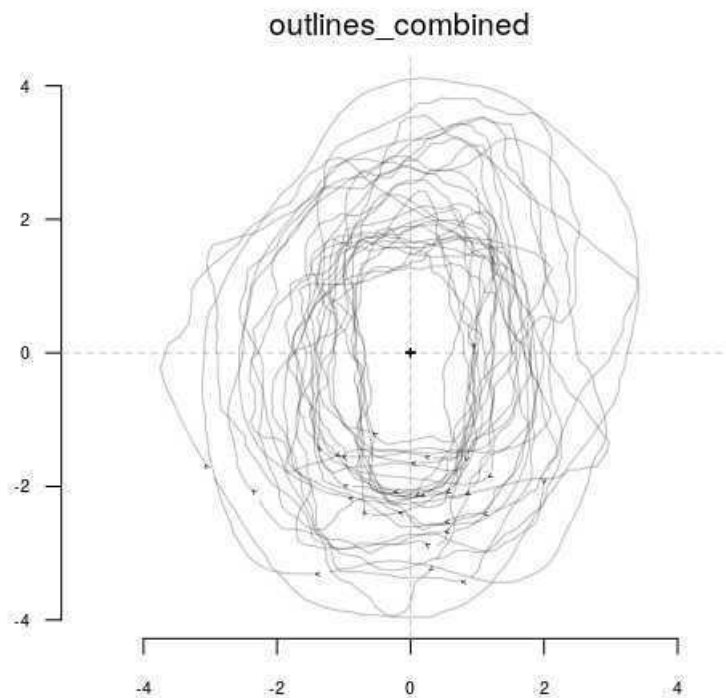
5.9 In a last step before starting to work with your outlines, you should inspect the just created outlines for validity and possible errors:

```
length(outlines_combined) #how many outlines do you have?  
stack(outlines_combined) # shows all outlines above one another(you might want  
to center and scale them first using Momocs)
```

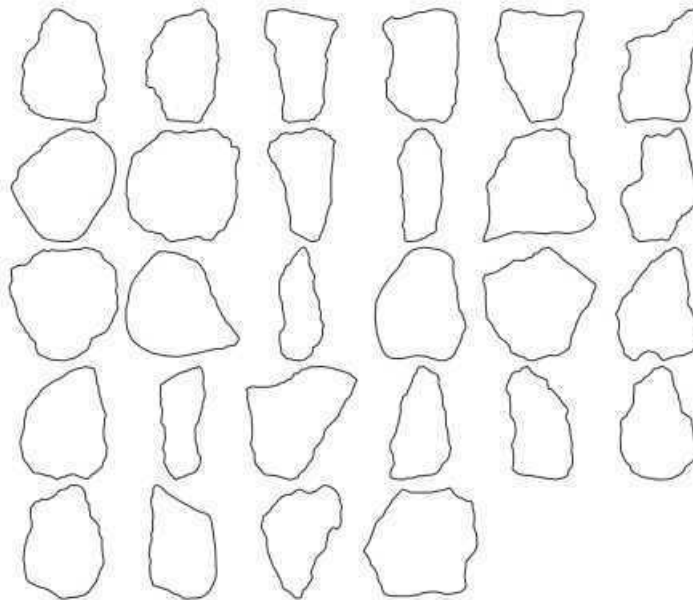


```
Momocs::panel(outlines_combined) # shows all outlines next to each other  
Momocs::inspect(outlines_combined) # shows only a single outline at a time.
```

Expected result:



Stacked outlines.



All extracted outlines from outlines_combined visualised using the Momocs::panel() command.