Amphora_Analysis

User Manual

Jenny Vo-Phamhi
jennyvo@stanford.edu
Class of 2020
B. S. Candidate, Computer Science
B. A. Candidate, Classics
Notation in Science Communication

CS Major Advisor and CS 191W Advisor: Chris Gregg
Department of Computer Science, Stanford University

Contents

Overview

Amphora_Analysis is the user interface for a set of tools which enable archaeologists to perform effective measurement and comparison of pre-modern ceramics using point clouds generated from structured light scans (e.g. Artec) or photogrammetry (e.g. Agisoft Photoscan).

This user manual walks you through the different use cases and examples shown in the Jupyter notebooks which comprise the Amphora_Analysis user interface. Each section covers one area of the analysis. We walk through the analysis material together together, using the sample data included with this package. At each step of the way, we will talk about how you would load and run your own data, accompanied by trouble shooting help and answers to commonly encountered questions. Diagrams with screenshots from the notebooks are included for easy reference and clarity. Throughout, there are also links to background reading if you want to dig in a little more deeply.

Background: Why do we need Amphora_Analysis?

Analysis of ceramic standardization and variation provides a powerful way to evaluate the scale, organization, and technological practices behind pre-modern production and for gauging the coordination and complexity of past economic systems. The systematic study of jars (amphoras) is required to answer archaeological research questions about standardization. However, the current methods of performing measurement and comparison of complex shapes that are common in archaeology – linear measurements of diameter and height – are limited in their ability to encapsulate jar shapes, which often vary in ways that cannot be captured by those simple linear measurements. Amphora_Analysis is the user interface for a set of tools which provides this functionality using point clouds of amphoras generated by structured light scanners (and, eventually, photogrammetry). Developed entirely with open-source Python libraries, these tools are free for all to use. This removes the cost barrier is often a problem for many archaeological projects, many of which work with limited funding. They would simply input their point clouds and generate a suite of analyses. Already, archaeologists in Sicily, Naples, Turkey, Canada, and the U.S. have expressed interest in using these tools for their projects. As a user-friendly interface, Amphora_Analysis lowers the entry barrier to using these tools and makes them accessible to these researchers.

Directory organization:

The Python code files which the notebooks use are in the code folder (Fig. 1). You do not need to edit these files.

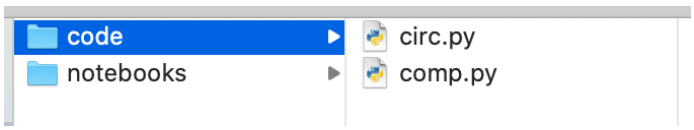

Fig. 1

The notebooks folder (Fig. 2) contains subdirectories which each contain a section of the analysis.

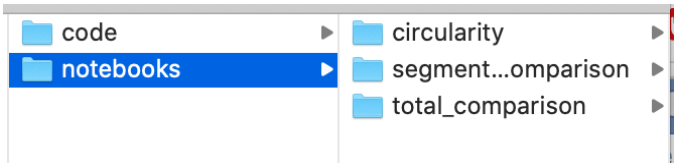

Fig. 2

Software license:

This project uses the Gnu Public License (GPL) version 3.0. More info here, but one of the key features is that not only is this project open-source and free to use for all, any derivative works must be also be open-source and free to use for all.

General tips:

If you have not used Jupyter notebooks before, start with this gentle and effective beginners' guide. I would skip the first three paragraphs and start where the headers says "Getting Started with Jupyter Notebooks!"

This tool developed by researchers at Penn State can help you choose colors that are colorblind safe, printer friendly, and photocopy safe.

CloudCompare is a great platform for viewing, manipulating, and segmenting point clouds (and preparing them for analysis). It is an independent open source project and totally free software. Check out the CloudCompare wiki for tips and how-tos.

Throughout this document, this blue color will be used for names of files and file paths and this green color will be used for variable names.

Note on data and code (for CS 191W):

The sample dataset which was used to develop, test, and demonstrate Amphora_Analysis is the same dataset we have been using throughout this archaeological project. (Archaeology is not a field in which it is customary to share data – especially data which is expensive to acquire, as the structured light scanners which produce these high-quality point clouds are not cheap – so I do not have access to other amphora point cloud datasets of sufficient quality to be analyzed.) Because the archaeology project is still in the process of being published at this time, I can only include in my CS 191W submission what I have written for my CS 191W submission as outlined in my CS 191 proposal – that is, the frontend code (i.e. the notebooks) which follows the workflow I designed, and this user manual. The dataset will accompany the tool when both are published together with our write-up.

Segmentwise Comparison

With segmentwise comparison, you can look at the differences between two amphoras at graduated sections of the amphoras (Fig. 3).

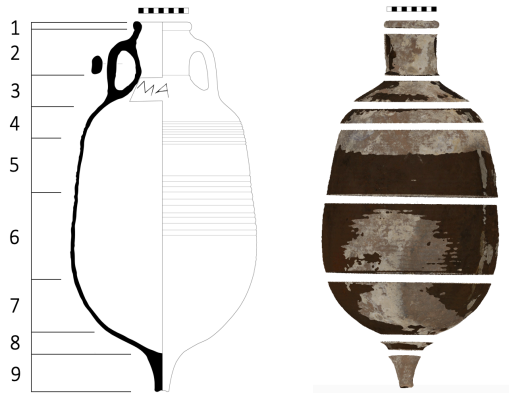

Fig. 3

Segmentwise comparison is executed in segmentwise_comparison.ipynb. The two amphoras in the sample dataset are jars 80E-2 and 4634 from the Gumusluk shipwreck site in Turkey.

Step 1: Check your files

Before you run this notebook, provide a point cloud or mesh in Wavefront Object (.obj) format for each of two amphoras for each graduated segment. If your amphoras are called '80E2' and '4634' and you have segments called '01', '12', '23', etc., your project directory should be organized as follows:

my_segmentwise_analysis (a folder)

- segmentwise_comparison.ipynb (this notebook file)
- segments (a folder)
  - 01 (a folder corresponding with the segment between cuts 0 and 1)
    - 01_80E2.mtl (material file for amphora 80E2)
    - 01_4634.mtl (material file for amphora 01_4634)
    - 01_80E2.obj (file for amphora 80E2)
    - 01_4634.obj (file for amphora 01_4634)
    - 0_SFusion_80E-2_01.bin (image file for amphora 80E2)
    - 0_SFusion_4634_01.bin (image file for amphora 01_4634)
  - 12
    - contents as in 01
  - 23
    - contents as in 01
  - etc. (as many segments as your jars have)

In the filesystem included with this manual, the "my_segmentwise_analysis" folder is called "segmentwise_comparison". You can change the name to whatever you want, e.g. "segmentwise_dataset_summer_2020_3"; this name will not affect the functioning of this tool.

Step 2: List the items you want to analyze

Input the following items...

- The names of the amphora segments
- The maximum diameter of each amphora
- The names of the amphoras you want to compare
- The file path to the segments
- Whether or not to save csv files of point cloud coordinates with all outliers excluded
- Whether or not to save csv files of nearest-neighbor distances between objects.

...into this cell (Fig. 4).

USER INPUT NEEDED!

```
segments = ['01', '12', '23', '34', '45', '56', '67', '78', '89']
max_diameters = [114, 111, 247, 304, 353, 441, 340, 110, 69]    # the maxim
amphora_names = ['80E2', '4634']
path_to_segments = './segments'
save_nonoutliers = True   # Set True if these files do not yet exist, else
save_distances = True     # Set True if these files do not yet exist, else
```

Fig. 4

Run the cell to set these parameters.

Step 3: Run Outlier Removal

Sometimes noise accumulates during the process of scanning an amphora and shows up as points scattered fuzzily around the object at certain places. This step removes points which are not close enough to other points to be considered part of the object.

Execute both cells in Step 3 without changing any code. This step can take a few minutes or so per pair of segments, depending on how large and how dense those point clouds are. You can track progress by watching the printout underneath the code cell (Fig. 5). In the following screenshot, outlier removal is currently being executed on segment '89' for both amphoras.

```
if not os.path.exists('./segment_csv'):
    os.mkdir('./segment_csv')

for segment in segments:

    print(segment) # you can see which segment the code is working on

    # Read Object 0
    obj0_filepath = '%s/%s/%s_%s.obj' % (path_to_segments, segment, segment, amphora_names[0])
    obj0_coords = comp.obj_to_np(obj0_filepath)

    # Remove Object 0 outlier points
    obj0_csv_filepath = './segment_csv/%s_%s.csv' % (segment, amphora_names[0])
    obj0_coords = comp.remove_outliers(obj0_coords, save_nonoutliers, obj0_csv_filepath)

    # Read Object 1
    obj1_filepath = '%s/%s/%s_%s.obj' % (path_to_segments, segment, segment, amphora_names[1])
    obj1_coords = comp.obj_to_np(obj1_filepath)

    # Remove Object 1 outlier points
    obj1_csv_filepath = './segment_csv/%s_%s.csv' % (segment, amphora_names[1])
    obj1_coords = comp.remove_outliers(obj1_coords, save_nonoutliers, obj1_csv_filepath)
```

Fig. 5

If you set save_nonoutliers = True in Step 2, at the end of Step 3, you will have a CSV file for each jar within a new subfolder called segment_csv.

Step 4: Run Cloud-to-Cloud Distance Comparison

Run the first cell to create an empty subfolder called results, which will contain the results of this comparison and the figures. Run the second cell to calculate the mean distance from each point in Object 1 to the nearest neighbor in Object 2 for each segment. Run the last cell if you want to save the average and stdev of the distance data for each segment in a comma-separated value (CSV) file within results (recommended).

Step 5: Visualize distance distributions across segments

You can adjust the color and resolution of your plots in this cell (Fig. 6).

```
bar_color = '#412100'
resolution_dpi = 600      # set resolution of saved figures in dots per inch
```

Fig. 6

Then, run the following cells to produce your results visualizations. Below each plot, there's a cell you can run to save the plot as a high-quality PNG file in the results subfolder.

"Serving Suggestion": Full Visualization

Since the results of average-NND (nearest neighbor distance) shape comparison are most intuitive when the above plot is displayed next to amphora renderings, here are some examples of a layout that has worked well for my team (Fig. 7). (We are actually used figures like these for our own manuscript.)

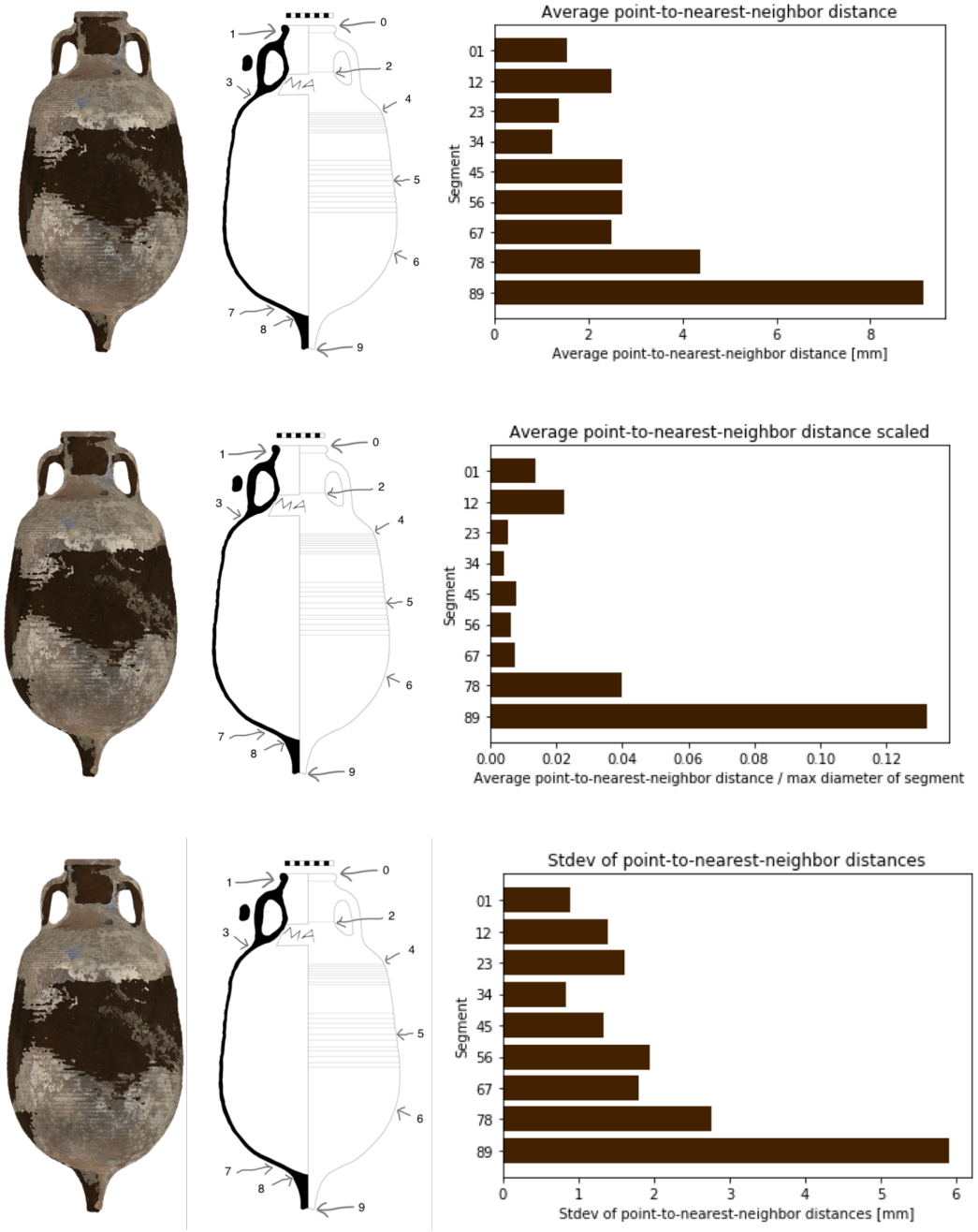
Average point-to-nearest-neighbor distance
Segment
01
12
23
34
45
56
67
78
89
0
2
4
6
8
Average point-to-nearest-neighbor distance [mm]
Average point-to-nearest-neighbor distance scaled
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Average point-to-nearest-neighbor distance / max diameter of segment
Stdev of point-to-nearest-neighbor distances
1
3
5
7
9
Stdev of point-to-nearest-neighbor distances [mm]

Fig. 7

Total Comparison

With total amphora analysis, you can look at morphological difference across large datasets (Fig. 8) and characterize standardization, comparing multiple amphoras to a single reference amphora.

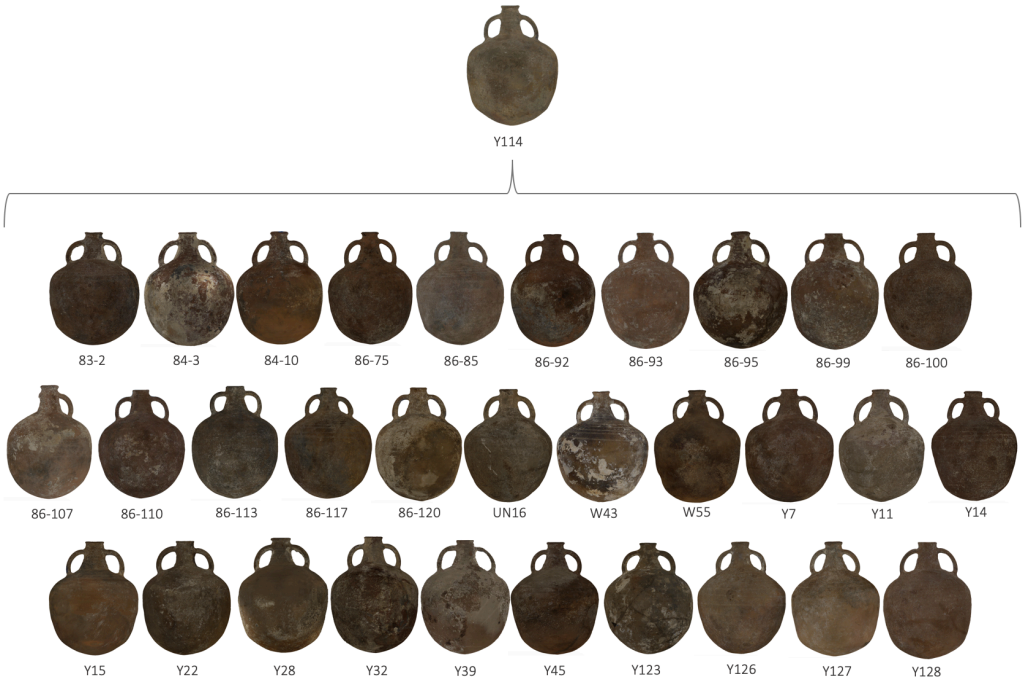

Fig. 8

Total amphora analysis is executed by two notebooks.
total_comparison_part_1_remove_outliers.ipynb removes noise from the models. If your models are already clean, you can dive into total_comparison_part_2_analysis.ipynb.

The amphoras in the sample dataset are from the Yassiada shipwreck in Turkey (Fig. 9).

Fig. 9

Part 1: Outlier (Noise) Removal

Use total_comparison_part_1_remove_outliers.ipynb.

Step 1: Check your files

If your amphoras are named 83-2, etc., your project directory should be organized as follows:

my_total_comparison_project (a folder)

- total_comparison_part_1_remove_outliers.ipynb (this notebook file)
- models
  - 83-2_handleless.mtl (material file for amphora 83-2, handles removed)
  - 83-2_handleless.obj (object file for amphora 83-2, handles removed)
  - YA7_83-2_obj_0.jpg (image file for amphora 83-2)
  - etc. for every jar

In the filesystem included with this manual, the "my_total_comparison_project" folder is called "total_comparison". You can change the name to whatever you want, e.g. "total_comparison_dataset_summer_2020_3"; this name will not affect the functioning of this tool.

Step 2: List the items you want to analyze

List the IDs of the amphoras you want to analyze in this cell (Fig. 10; just the upper part shown, for brevity):

USER INPUT NEEDED!

```
models = ['83-2',
          '84-3',
          '84-10',
          '86-75',
          '86-85',
          '86-92',
          '86-93',
          '86-95',
          '86-99',
          '86-100',
          '86-107',
          '86-110',
          '86-113',
          '86-117',
          '86-120',
          'UN16',
```

Fig. 10

Step 3: Run Outlier Removal

Outlier points are identified as points which are not close enough to other points. (Closeness threshold is the Otsu threshold as computed on the bimodal point-to-point distance distribution. More on Otsu's method threshold here.)

Run the cell below to decide whether or not to save csv files of point cloud coordinates with all outliers excluded.

As in Segmentwise Comparison, this step removes points which are not close enough to other points to be considered part of the object.

In the first cell, decide whether or not to save csv files of point cloud coordinates with all outliers excluded. (Set True if these files do not yet exist.)

```
save_nonoutliers = True    # Set True if these files do not yet exist, else False.
```

Then run the second and third cells without changing any code. This step can take a few minutes or so per amphora, depending on how large and how dense those point clouds are. You can track progress by watching the printout underneath the code cell. In the following screenshot (Fig. 11), outlier removal is currently being executed on amphora 86-95.

```
if not os.path.exists('clean_csv'):
    os.mkdir('clean_csv')
```

```
for model in models:

    print('Now cleaning amphora %s ' % model) # Allows the user to see which model the code is t

    # Read object
    obj_filepath = './models/%s_handleless.obj' % model
    obj_coords = comp.obj_to_np(obj_filepath)

    # Remove object outlier points
    obj_csv_filepath = './clean_csv/%s.csv' % model
    obj_coords = comp.remove_outliers(obj_coords, save_nonoutliers, obj_csv_filepath)
```

```
Now cleaning amphora 83-2
Now cleaning amphora 84-3
Now cleaning amphora 84-10
Now cleaning amphora 86-75
Now cleaning amphora 86-85
Now cleaning amphora 86-92
Now cleaning amphora 86-93
Now cleaning amphora 86-95
```

Fig. 11

Part 2: Analysis

The analysis is executed in total_comparison_part_2_analysis.ipynb.

Step 1: Check your files

If your amphoras are named 83-2, 84-3, etc. your project directory should be organized as follows:

my_total_comparison_project (a folder)

- total_comparison_part_2_analysis.ipynb (this notebook file)
- clean_csv
  - 83-2.csv (csv files of 83-2's coordinates, all outliers excluded)
  - 84-3.csv (csv files of 84-23's coordinates, all outliers excluded)
  - etc. for every jar
- phys_data.csv (a file containing physical data -- max_diam, body_height, group -- measured in-person)

The following files are no longer needed for the analysis at this stage:

- models (folder used in Part 1)
- total_comparison_part_1_remove_outliers.ipynb

However, it is best practice to hold onto them and leave them in this same project folder for future references.

Step 2: List the items you want to analyze

Identify the amphora you want to use as a reference as ref and list the IDs of the amphoras you want to compare with the reference amphora inside the compare list in the following cell (Fig. 12; just the upper part shown for brevity):

USER INPUT NEEDED!

```
ref = 'Y114'
compare = ['83-2',
           '84-3',
           '84-10',
           '86-75',
           '86-85',
           '86-92',
           '86-93',
           '86-95',
           '86-99',
           '86-100',
           '86-107',
```

Fig. 12

Step 3: Run cloud-to-cloud distance comparison

In the first cell (Fig. 13), decide whether or not to save csv files of nearest-neighbor distances between objects. (Set True if these files do not yet exist.)

```
save_distances = True       # set True if these files do not yet exist, els
```

Fig. 13

Then run the second and third (Fig. 14) cells without changing any code. This step can take a few minutes or so per amphora, depending on how large and how dense those point clouds are. You can track progress by watching the printout underneath the code cell. In the following screenshot, outlier removal is currently being executed on amphora 86-95.

```
average_distances = []
stdev_distances = []
for model in compare:

    print(model) # so the user can see which segment the code is working on

    ref_csv_path = './clean_csv/%s.csv' % ref
    model_csv_path = './clean_csv/%s.csv' % model
    dist_filepath = './distances/%s_%s_distances.csv' % (ref, model)

    average_distance, stdev_distance = comp.get_avg_cc_dist(ref_csv_path, m

    average_distances.append(average_distance)
    stdev_distances.append(stdev_distance)
```

```
83-2
84-3
84-10
86-75
86-85
86-92
86-93
86-95
```

Fig. 14

Run the fourth cell of this step to create an empty folder called "results" and run the fifth cell to save the average and stdev distance data for each pair in a comma-separated value (CSV) file.

Step 4: Visualize morphological difference across the dataset

You can customize the look of this figure by adjusting parameters in this cell (Fig. 15).

```
figure_width = 15
figure_height = 6
dist_color = '#412100'    # set color for bars representing nearest neighbor
diam_color = '#d2c295'    # set color for bars representing maximum diameter
bodh_color = '#928b81'    # set color for bars representing body height
resolution_dpi = 600      # set resolution of saved figures in dots per inch
```

Fig. 15

Run the next cell (Fig. 16) without changing any code to visualize the morphological differences between the reference amphora and the comparison amphoras as a bar chart. A high-quality plot is saved to the results subfolder.

```
# order the compared amphoras by avg NND to ref amphora
compare_sorted = [x for _,x in sorted(zip(average_distances, compare))]
average_distances_sorted = sorted(average_distances)

fig, ax = plt.subplots(figsize = (figure_width, figure_height))
x_pos = np.arange(len(compare_sorted))
ax.bar(x_pos, average_distances_sorted, color=dist_color)
ax.set_xticks(x_pos)
ax.set_xticklabels(compare_sorted)
ax.set_xlabel('Compared with %s' % ref)
ax.set_ylabel('Average point-to-nearest-neighbor distance')
ax.set_title('Morphological Difference (Avg. NND) from %s' % ref)
plt.xticks(rotation='vertical')
plt.savefig('./results/Morphological Difference (Avg. NND) from %s.png' % r
plt.show()
```

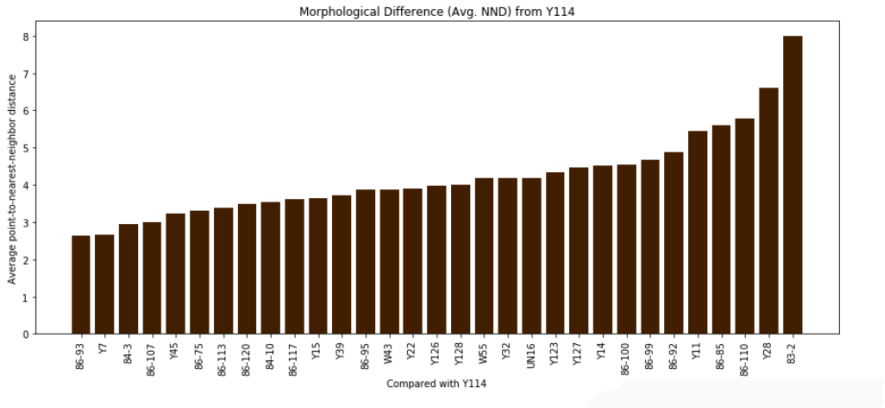

Fig. 16

The analyses in the remainder of this notebook require physical, traditional measurements for body height and maximum diameter and expert typological assessment (e.g. spiral combed, main group, etc.). Before running these analyses, check that these measurements are codified in a CSV file as follows (Fig. 17):

tag	max_diam	body_height	group
83-2	40.7	38.9	Spiral combed - main group
84-3	42	41.3	Spiral combed - main group
84-10	41.8	40.9	Spiral combed - main group
86-75	41.8	41.4	Spiral combed - main group
86-85	41.3	39	Spiral combed - main group
86-92	40.9	41.2	Spiral combed - main group
86-93	42.3	41.3	Spiral combed - main group

Fig. 17

Run the third cell (Fig. 18) in this dataset to load physical data.

Import pertinent data physically measured.

```
import pandas as pd
phys_data = pd.read_csv('phys_data.csv', index_col=0)
print(phys_data)
```

```
        max_diam  body_height                        group
tag
83-2        40.7         38.9  Spiral combed - main group
84-3        42.0         41.3  Spiral combed - main group
84-10       41.8         40.9  Spiral combed - main group
86-75       41.8         41.4  Spiral combed - main group
86-85       41.3         39.0  Spiral combed - main group
86-92       40.9         41.2  Spiral combed - main group
86-93       42.3         41.3  Spiral combed - main group
86-95       43.0         41.5  Spiral combed - main group
```

Fig. 18

Run the rest of the cells in the notebook to visualize morphological difference compared with other physical quantities across the dataset (with the plot saved to results), visualize morphological difference along with differences in physical metrics (with the plot saved to results), and output numerical data that characterizes the dataset as a whole and is useful for reporting in publications.

Circularity Analysis

Circularity analysis is useful for researchers who seek to understand how intent potters were on standardization and to uncover clues regarding the methods different potters may have used. This analysis grades the circularity of the amphora's cross section of greatest diameter (Fig. 19a) on a scale from 0 (asymptotically) to 1 (a perfect circle; Fig. 19b).

Fig. 19a
Aerial view of Amphora 83-2

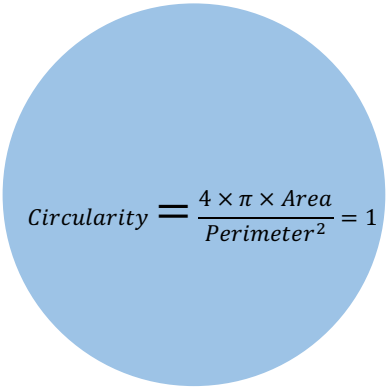

Fig. 19b
Perfect circle

Circularity analysis is executed by circularity.ipynb.

As with total amphora analysis, the amphoras in the sample dataset are from the Yassiada shipwreck in Turkey.

Step 1: Check your files

Save an aerial-view PNG image of each amphora into a folder called 'circles'. The folder 'circles' should live in the same folder as this notebook. Each aerial-view PNG image should be named 'circle_ID.png', where each ID corresponds to the object's identification marker within your project's organizational system.

Here's how to create an aerial-view PNG image file of an amphora:

1. Open the amphora point cloud in your preferred point cloud viewer (CloudCompare, Preview, etc.) and set the background to white (or as close as you can get to white).
2. Remove handles and other protrusions from the body. Orient the amphora for a direct bird's eye view (Fig. 20).
3. Capture the image. If your preferred point cloud viewer does not have in-app capture, you can zoom in as much as you can so the model fills your screen (to maximize resolution) and take a screenshot.

If your amphoras are named 83-2, etc., your project directory should be organized as follows:

my_circularity_analysis_project (a folder)

- circularity.ipynb (this notebook file)
- circles (a folder)
  - circle_83-2.png (an aerial-view PNG image of amphora 83-2, as in Fig. 1)
  - circle_84-3.png
  - circle_84-10.png
  - etc.

In the filesystem included with this manual, the "my_circularity_analysis_project" folder is called "circularity". You can change the name to whatever you want, e.g. "circularity_dataset_summer_2020_3"; this name will not affect the functioning of this tool.

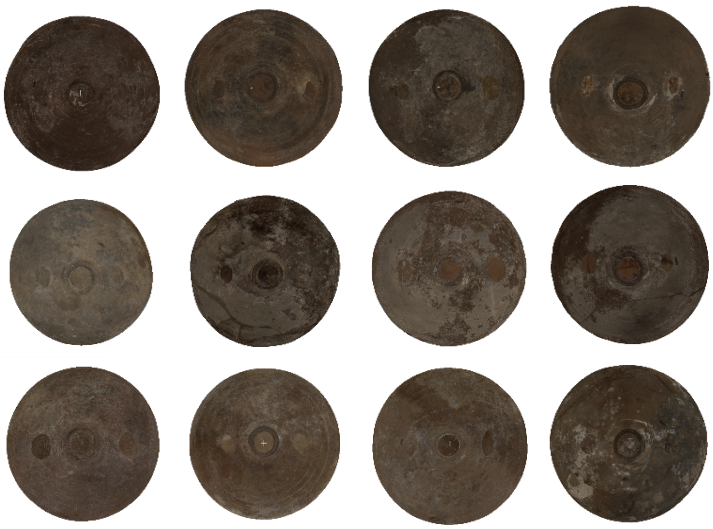

Fig. 20

Step 2: List the items you want to analyze

List the IDs of the amphoras you want to analyze in this cell (Fig. 21; just the upper part shown, for brevity):

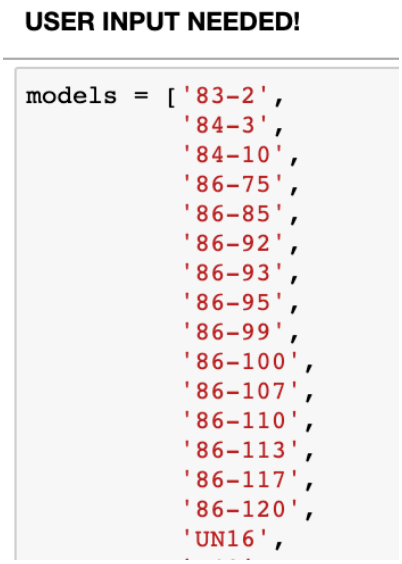

Fig. 21

Step 3: Run the Analysis

Run the two cells in this step without changing any code to run circularity analysis on all your amphoras.

You can track the progress of this processing by following along under the second cell (Fig. 22).

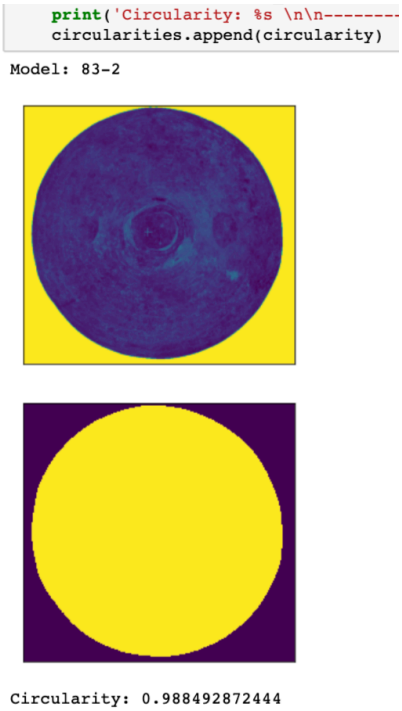

Fig. 22

The first image shows you the binarized input image. The second image shows you the area found corresponding to the shape, which is what the code uses to calculate area and perimeter of the shape.

Step 4: Visualize the results

You can customize the look of this figure by adjusting parameters in this cell (Fig. 23). Run the cell to set the parameters.

```
width = 0.001
min_circularity = 0.975
max_circularity = 0.99
min_num_amphoras = 0
max_num_amphoras = 20
interval_num_amphoras = 2
figure_width = 8
figure_height = 5
bar_color = '#DE956C'
resolution_dpi = 600     # set resolution of saved figures in dots per inch
```

Fig. 23

Run the second cell (Fig. 24) in this step to generate a histogram of amphora circularity.

```
plt.figure(figsize=(figure_width, figure_height))
plt.hist(circularities, bins=np.arange(min_circularity, max_circularity, width)+width/2, color=b
plt.xticks(np.arange(min_circularity, max_circularity, width), rotation=45)
plt.yticks(np.arange(min_num_amphoras, max_num_amphoras, interval_num_amphoras))
plt.xlabel('\nCircularity = 4 * pi * area / (perimeter^s)')
plt.ylabel('Number of Amphoras\n')
plt.title('Circularities for Yassiada Amphoras, n = %s' % len(models))
plt.show()
```

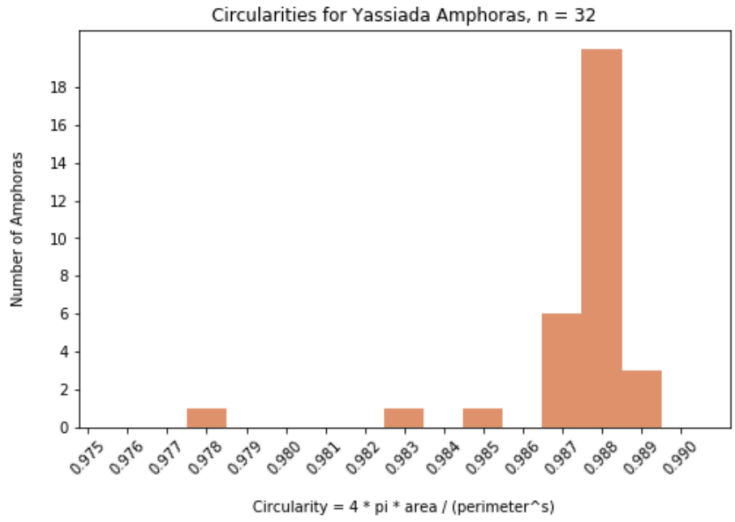

Fig. 24

Step 5: Save the results

Run the first cell (Fig. 25) to create an empty folder called "results".

```
if not os.path.exists('results'):
    os.mkdir('results')
```

Fig. 25

Run the second cell (Fig. 26) if you want to save the circularity data for each model in a comma-separated value (CSV) file within results (recommended)

```
with open('./results/circularities.csv', 'wb') as f:
    writer = csv.writer(f)
    writer.writerows(izip(models, circularities))
```

Fig. 26

Run the third cell (Fig. 27) to save the plot as a high-resolution figure (PNG).

```
plt.savefig('./results/circularities_histogram.png', dpi=resolution_dpi)
```

Fig. 27