APSAP: Shred Matching Tool: a tool to enhance the speed of ceramic matching

Introduction

This document clarifies the terms and methodology used in the ceramic sherd matching application. Our ceramic matching algorithm works in the following way. First, we find the information of a ceramic sherd in its 2d jpeg file, including area, average brightness, standard deviation of the average brightness, and the width and length of its bounding box. Second, we find the same set of information from **all** of the 3d models, by setting up the camera with an appropriate distance from the 3d models and a suitable field of view, taking a screenshot, and analyzing the pixels of the screenshot. Thirdly, by comparing the information of that ceramic sherd jpg, and that of all the ceramic sherd 3d models, we can assess how similar our jpg is to all those 3d models, and generate a sorted list ordered by similarity. In practice, we have a set of 2 jpegs for each ceramic sherd to be compared with all 3d models to make the accuracy higher. Our method yields a significant better performance than matching without a sorted similarity list.

Performance improvement

A provisional assessment of the performance of our matching algorithm has the following results:

1) We considered a total of 109 pairs of ceramic sherd jpegs. A total of 104 are valid(i.e. they have corresponding 3d models.
2) In 64 out of 104 cases, the correct 3d model is on the top of the similarity list.
3) In 85 out of 104 cases, the correct 3d model is on the top 3 of the similarity list.
4) In 93 out of 104 cases, the correct 3d model is on the top 5 of the similarity list.
5) In 100 out of 104 cases, the correct 3d model is on the top 10 of the similarity list.

We consider this as significant improvement. as the chance of a random matching algorithm having a correct prediction is 1/104 = 0.96% of matching when our method yields 64 /104 = 61.5%, a dramatic improvement of performance. The performance increase is similarly significant when we consider the top-n similarity list.

Consider the case of matching 100 pairs of sherd jpegs with 100 3d models. Without a similarity list, one has to go through the 3d models randomly to find the 3d model matched for a certain pair of jpegs. A naive estimate of the number of models one has to go through to find the correct one is 100/2 = 50 on average. With 100 jpegs, one has to check models for a 50 * 100 = 5000 times. Using our method, a naive estimation of the number of times of going through the model is 64 * 1 + (85-64) * 3 + (93 - 85) * 5 + (100 - 93) * 10 + (104 - 100) * 20 = 317 (assuming it takes 20 times of checking the models on average for the worst cases).

In order to achieve this performance, the most important thing is to get the information of the ceramic sherds from the 3d models or 2d jpegs. But to do that, one has to isolate the sherd pixels from the images so that we can analyze the information of the pixels. To isolate the sherd pixels, we need to use neural networks.

Using neural network to get sherd pixels in a 2d jpg

Mask-rcnn is a neural network framework used to create neural networks that can detect 'masks' for objects in a picture. Suppose you have a picture of sheep in a farm, a trained Mask-rcnn neural network can highlight all the sheep as red in the picture, thus detecting where the sheep are in the picture. We trained a **sherd-detector** with around 200 training samples, that is able to accurately estimate the pixel locations of the ceramic sherd. Our application relies on the deep learning library Pytorch.

With our sherd-detector, we can detect the pixels within a jpg that belongs to the ceramic sherd, thus we will be able to get its info. In each comparison, we calculate the information of two jpegs belonging to one ceramic sherds, with hundred(s) of 3d models. To save time, neural networks are only used to analyze the two jpegs.

Figure 1.a and 1.b:
Our sherd detector is able to detect the area where the sherd is in the jpeg, thus able to locate the pixels that are relevant for analysis.

As we need to compare the info of 2d jpegs with info of the 3d models, we need to find similar info of the 3d model. And we have a few discoveries that greatly help with this purpose.

Using Open3d to get sherd pixels in a 3d model

We have two interesting discoveries regarding the 3d models.

1) Unlike highly regular-shaped objects like the cube and the pyramid, which have 6 ways and 4 ways to stand stable on the surface, most ceramic sherds have only 1 or 2 natural ways to stand stably.
2) The 3d model format PLY stores the 3d model in such a way that it is oriented with its length aligning the X axis, its width aligning the Y axis and its depth aligning the Z axis, with the object's origin at (0, 0, 0).

Figure 2.a and 2.b
The highly irregular shape of ceramic sherds makes it highly likely for one to stabilize it on the table with only 1 or 2 possibilities(The pictures on the left). The 3d model(on the right) is at (0, 0, 0), with its length of the bounding box aligning the x axis, width aligning the y axis.

Because of 1), it is highly likely that the way the ceramic sherd is laid under the camera is the same as the way it is laid under the 3d scanner. This leads to an important result: as one views the model with the Open3d viewer, if we set an appropriate camera distance and field of view, by screenshotting, we can replicate an image similar to the jpeg taken with a normal camera. As this image has both the ceramic sherd and the white background with the RGB value, (255, 255, 255), by filtering out these white pixels, we can get all the pixels of the ceramic sherds. Thereby, we can analyze the pixel information of the 3d model.

In short, we can get the pixel information for both the 2d jpegs and the 3d models. As the similarities between a 3d model and a 2d jpg depend on the pixel information, we will explain them in more details.

Pixel Information

In the current version of the application, we have a total number of five types of pixel information we consider. The main criteria of our choice of information we extract from the information depends on two things. First, how costly it is to compute. If it is too costly, the user experience would be bad. This is the reason neural networks are not used to extract the information from the pixels(otherwise, with hundreds of models, our application would be extremely slow). Second, how accurate the type of information is in leading to a more accurate matching result.

Our pixel information includes: area, intensity, intensity standard deviation, bounding-box and color. We shall explain more details for each of these.

Area

The area is the mm squared the ceramic sherd occupies in the real world. Imagine we lay a 100mm x 100mm grid under a ceramic sherd, looking down from the camera on top, we will see the ceramic sherd covering the grid. The area covered on the grid is the area of the ceramic sherd.

As mentioned above, we are able to get the information of the pixels, thus the number of pixels of the 2d jpegs of the ceramic sherds and the screenshots of the 3d models. What we need is a mm-to-pixel_length ratio. If we have the mm-to-pixel_length ratio, we can get the actual area of ceramic sherds with the following formula:

$$\textit{Actual sherd Area} = \text{num\_of\_sherd\_pixels} * (\text{mm-to-pixel\_length})^2$$

Area for the 2d jpegs

For our 2d jpegs, to get the mm-to-pixel_length, we need to know the ratio between the width of an object in actual dimension and its width in pixels. To achieve that, we trained a Mask-rcnn **grid-detector** with 100 training samples. it is able to accurately estimate the pixel locations of the credit card photography scale in the jpegs containing ceramic sherds. The majority of the sherds we consider are pictured with a **credit card photography-scale.** We know the width of a credit card is 53.98 mm. The neural network helps us identify the credit card sized grid's pixels , therefore its width in pixels. As a result, we can get the mm-to-pixel_length with 53.98/width_of_pixels_detected. Notice that, we can train a different grid-detector for a different grid, as long as we know its actual dimension to get the ratio. With the mm-to-pixel_length ratio, we can get the actual area of the ceramic sherd in the jpg.

Figure 3.a, 3.b, 3.c
We are able to get pixel locations for the credit-card color scale, thus knowing the width and height of the credit card in terms of pixels. As we also know the width of height of the actual credit card, we can know the mm-to-pixel ratio to calculate the area of the sherd.

Area for the 3d models.

For the screen shots of the 3d model, we immediately know the width of our object by getting the axis-aligned bounding box's width. To get the pixel width, we show the bounding box as red pixels and detect the red pixels' locations. As a result, we can get the mm-to-pixel_length with the bounding_box_width/width_of_red_pixels. With the areas of the jpegs and the screenshots, we can calculate their similarity in terms of area. As mentioned, as the orientations of the ceramic sherds in the picture and in the 3d model often are the same, areas obtained between the jpeg and the screenshot should be similar.

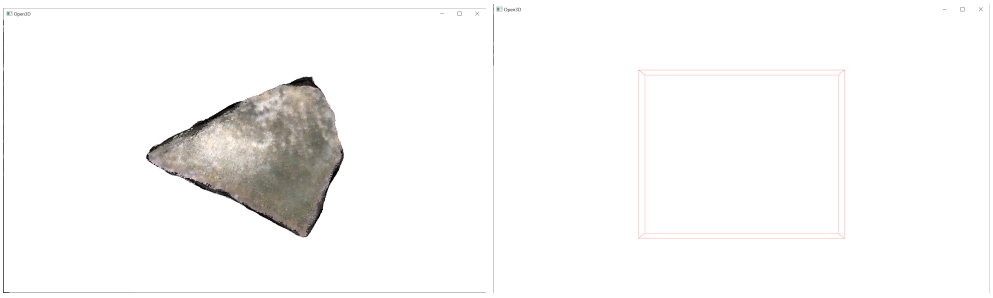

Figure 4.a, 4.b
For each 3d model, we can get its bounding box. As the 3d models record the length of the bounding box in mm, if we calculate the distance between the red lines, we can get the corresponding length of the bounding box in pixel,

Bounding box

Similar to the area approach, this time we calculate the width and length for the 2d jpegs and the screenshots of the 3d models. Similarly, we get the mm-to-pixel_length using the similar method and we can get the actual length and width of the bounding box surrounding the ceramics. Although the orientations between the models and the jpegs are often the same for the ceramic sherds because of the stable-resting principle, the sherd may have rotations. For instance you can rotate a can of Coke Cola along the y axis, and it still stands. As a result, the bounding box is a less accurate measure to calculate the similarity.

Figure 5.a, 5.b
Because there are one or two natural ways to lay the ceramic sherd stable on a surface, the general orientation is the same under a normal camera and the 3d scanner. The problem is that we can rotate the sherd for at most 360 degrees on the surface, and the stable ceramic sherd would still remain stable. This doesn't affect the area estimation as the area stays the same when we rotate it flat on the table.

Average brightness and Average brightness similarity

By turning all pixels of a ceramic sherd into grayscale(i.e. black and white), each pixel has a certain positive number representing its brightness. The higher this number is, the more bright the ceramic sherd is. For each ceramic sherd image, regardless if it is the jpeg or the screenshot of the 3d model, we can average all its pixels' brightness values, resulting in a single number representing the average brightness of that picture. We discovered that the sherds in jpegs are approximately 20% more bright than in the 3d model (this number can be tweaked for higher precision). We take that into account in our calculation of the similarities between 2d jpegs and 3d models.

Similarly, we can get the standard deviation of the black and white pixels and compare them. We noticed that on average the pixels of the 2d jpegs have a higher standard deviation than the 3d models by about 2.1 times(Again this number can be tweaked for higher precision). This is expected as the camera should have a better dynamic range than the 3d scanner because it can better capture the variations of colors and light intensitiy, hence the higher standard deviations. Similarly, we take this 2.1 value into account

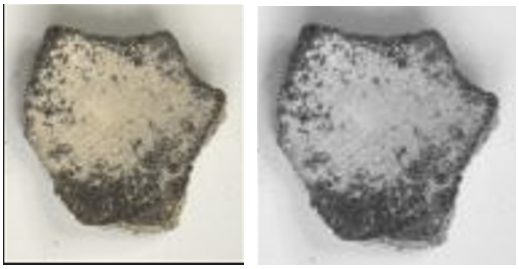

Figure 6.a, 6.b

But turning the sherd's pixels black and white, we get pixels with a 3-element tuple of values to pixels with scalar values. Thus, we can calculate the average brightness value for a sherd and use it for making comparisons.

Color

Last but not least, we calculate the difference between the color pixels of the 3d models and the 2d jpegs using a formula.[1] We noticed that this criterion is not that important for calculating the actual similarity. Perhaps the formula itself is flawed for our application.

Putting the things together

Given all the information, we calculate a total similarity score between a ceramic sherd's jpegs with a 3d model by using this linear formula.

Total similarity = 90 * area_similarity + 65 * brightness_similarity + 40 * bounding_box_similarity + 5 * brightness_std_similarity + 0.15 * color_similarity

The lower the total similarity score is, the more similar the 3d model is to the jpegs. Thus we have a sorted list of 3d models from the most likely to the least likely. This is the key to the acceleration afforded by this application.

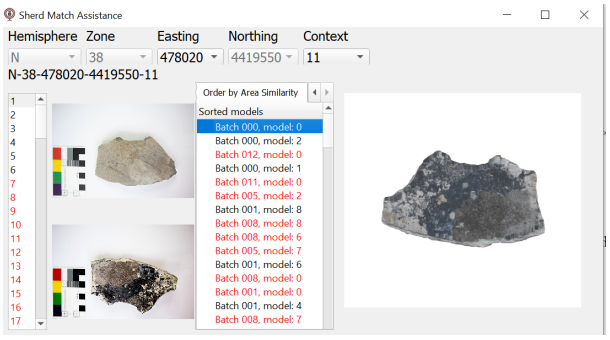

Figure 7.

The 3d models are sorted for each set of jpegs. Notice exactly the first model in the list is the one matching the 2d jpegs, on this occasion, it takes simply 5 seconds to confirm this model then we can check the next ceramic sherd. If we don't have this sorted list, we may need to check 50 models before we find it.

It is highly unlikely the underlying relationship between the individual similarity scores and actual similarity score is a linear one. But this formula is easy to understand and tweak. It also offers really good results. Currently, the factors by which the individual similarities multiply are tested by hand. As a result, an possible improvement in the future is to use machine learning or linear regressions to gauge a set of more accurate values.

[1] https://en.wikipedia.org/wiki/Color_difference#sRGB