

Q1: Signature Matching

The signature image is our identity, banks and other places signatures are used, can you take 10 copies of your signature and show that they represent you, we can use morphological operations like thinning, Skeleton
Can we find bends and curves to identify them
Can we use envelope to make them more robust

Approach

Pre-Processing

- Gaussian Blur
- Morphological Opening
- Otsu Thresholding
- Morphological Opening

Transformation ([Skeletonize](#))

- We use the `skimage` library function for skeleton transformation by thinning the thresholded image. This gives us a signature with only 1 pixel boundary
- Then we can save this as our reference skeleton

Matching ([structural similarity](#))

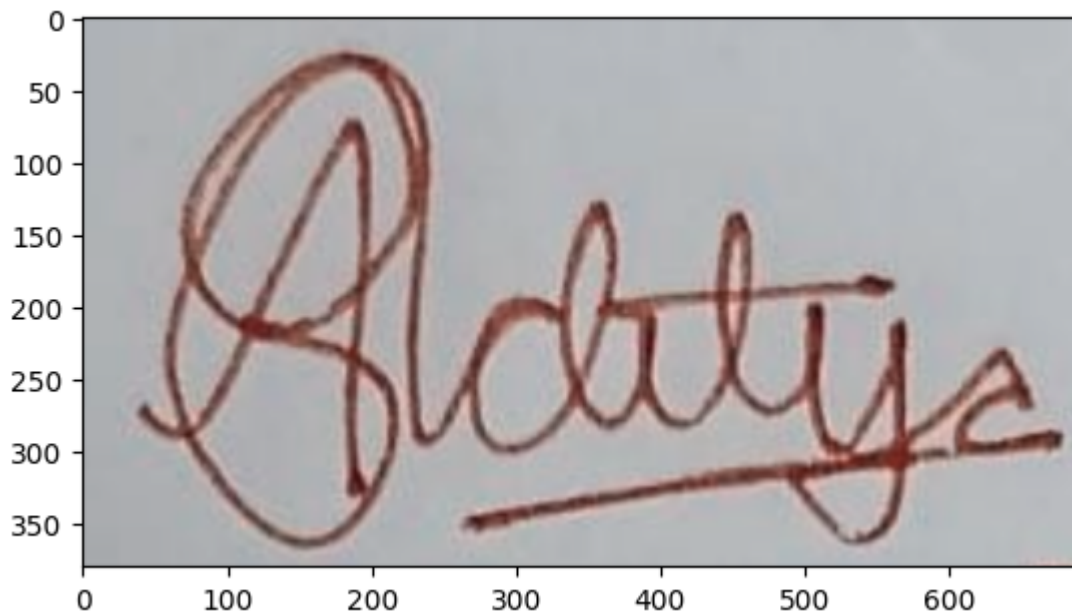
- We will use `skimage`'s structural similarity, to calculate the accuracy of the signature match

```
In [1]: import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import skimage
```

```
In [2]: REFERENCE_IMAGE = "sign1.png"
TEST_IMAGE = "sign2.png"
```

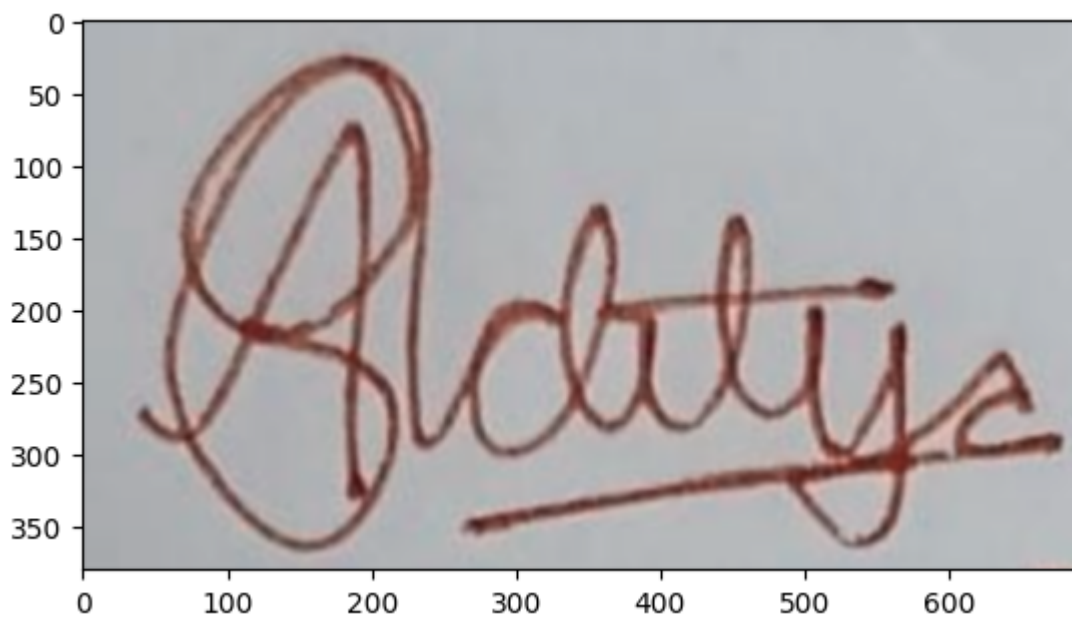
```
In [3]: img = cv.imread(REFERENCE_IMAGE)
plt.imshow(img)
```

```
Out[3]: <matplotlib.image.AxesImage at 0x17bf5bc50>
```



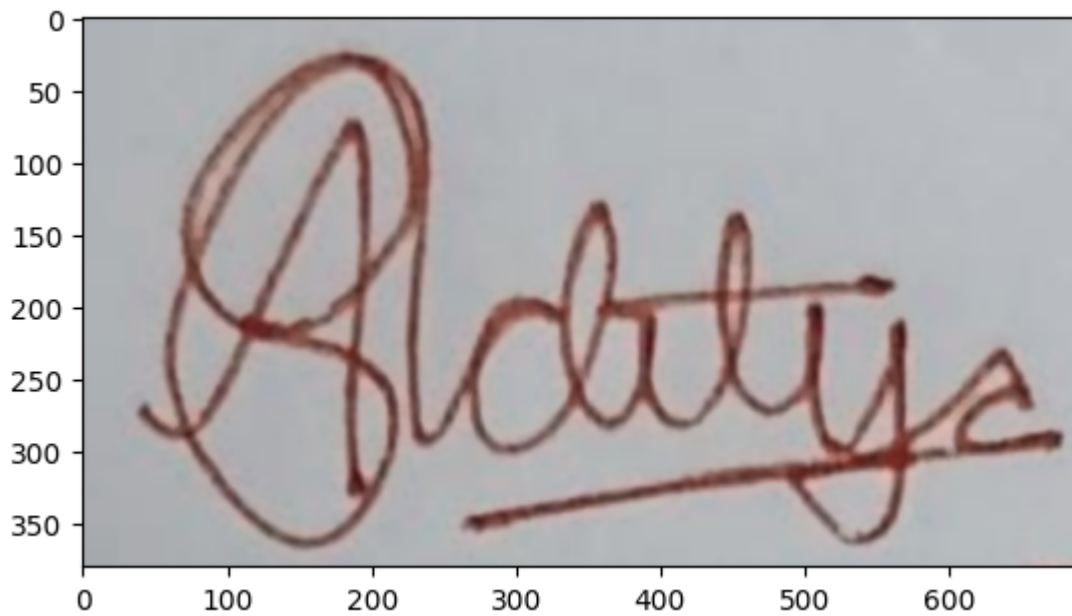
```
In [4]: img_blur = cv.GaussianBlur(img, (5,5), None)
plt.imshow(img_blur)
```

Out[4]: <matplotlib.image.AxesImage at 0x17c1e9b90>



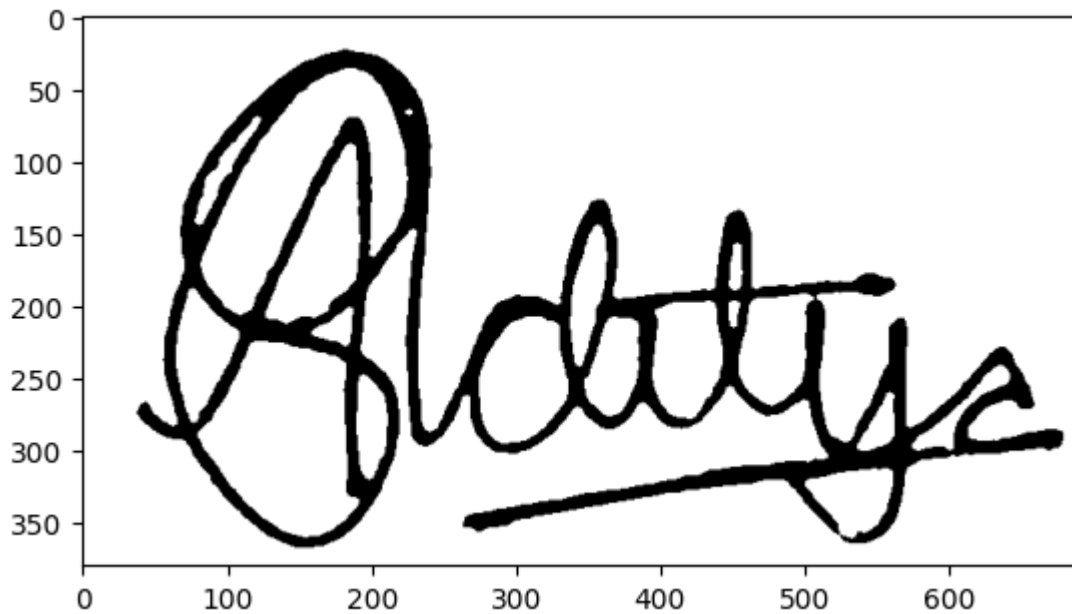
```
In [5]: img_close = cv.morphologyEx(img_blur, cv.MORPH_OPEN, cv.getStructuringEle
plt.imshow(img_close)
```

Out[5]: <matplotlib.image.AxesImage at 0x17c1d1b90>



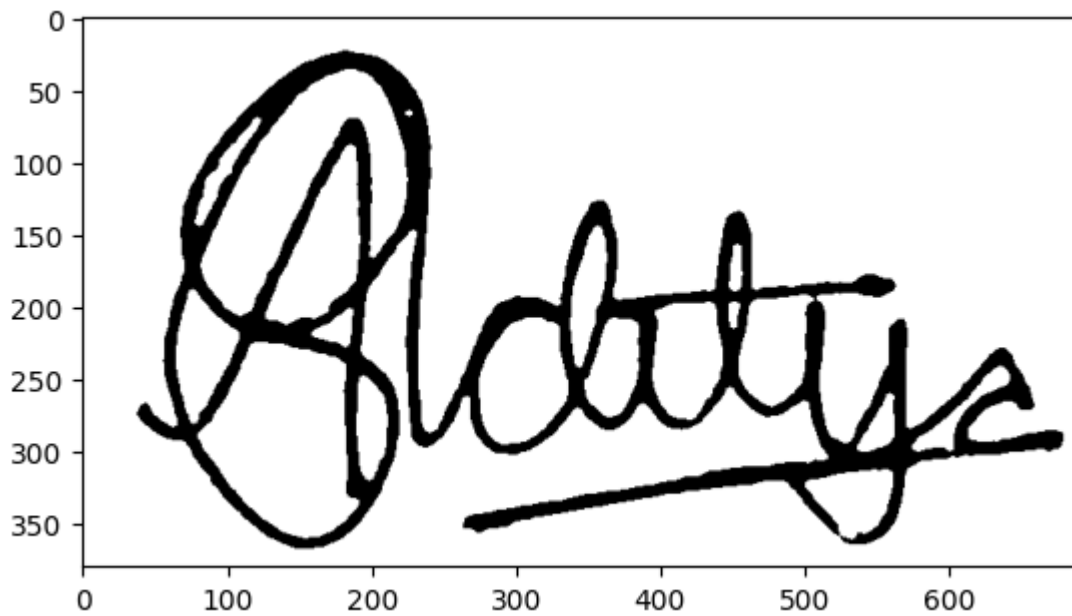
```
In [6]: img_g = cv.cvtColor(img_close, cv.COLOR_BGR2GRAY)
_, bw = cv.threshold(img_g, 0, 255, cv.THRESH_BINARY+cv.THRESH_OTSU)
plt.imshow(bw, cmap="gray")
```

Out[6]: <matplotlib.image.AxesImage at 0x17c989a50>



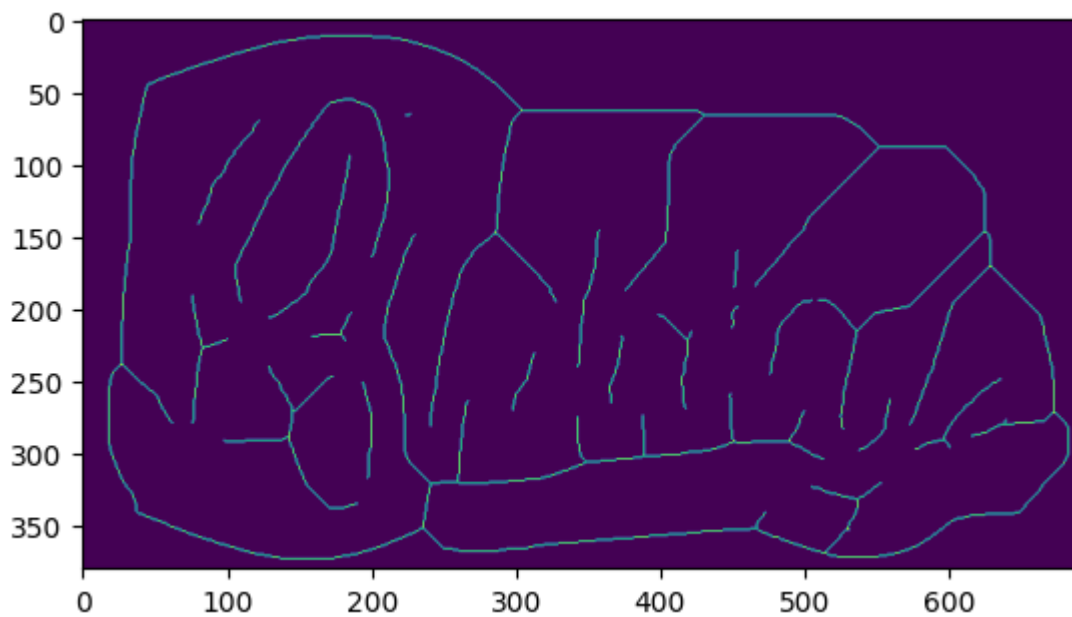
```
In [7]: img_closed = cv.morphologyEx(bw, cv.MORPH_OPEN, cv.getStructuringElement(
plt.imshow(img_closed, cmap="gray")
```

Out[7]: <matplotlib.image.AxesImage at 0x17ca20610>



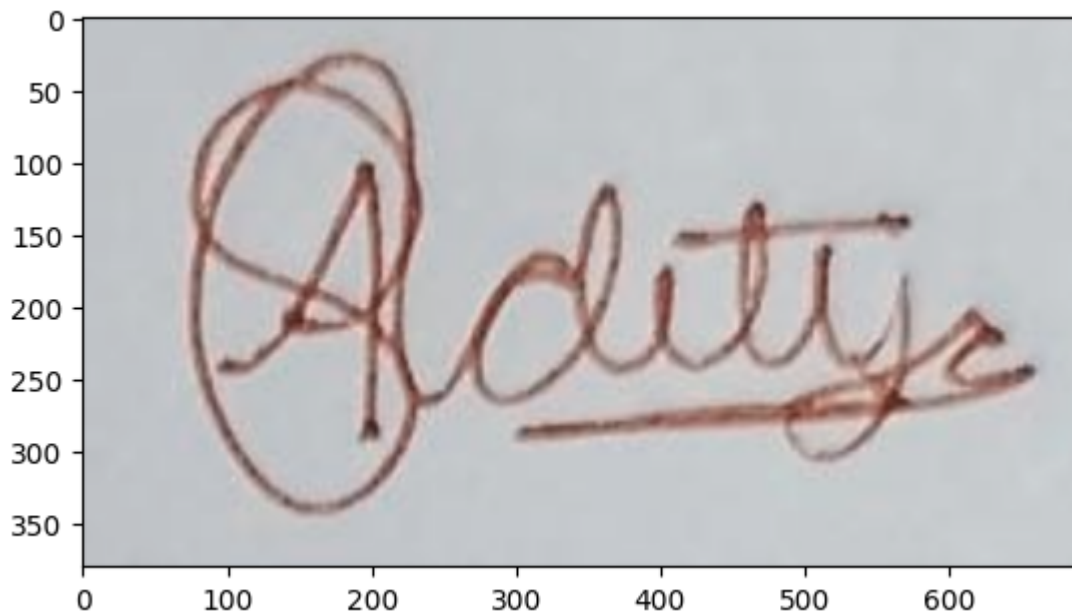
```
In [8]: img_skeleton = skimage.morphology.skeletonize(img_closed)
img_skeleton = skimage.util.img_as_ubyte(img_skeleton)
plt.imshow(img_skeleton)
cv.imwrite("ref.jpg", img_skeleton)
```

Out[8]: True



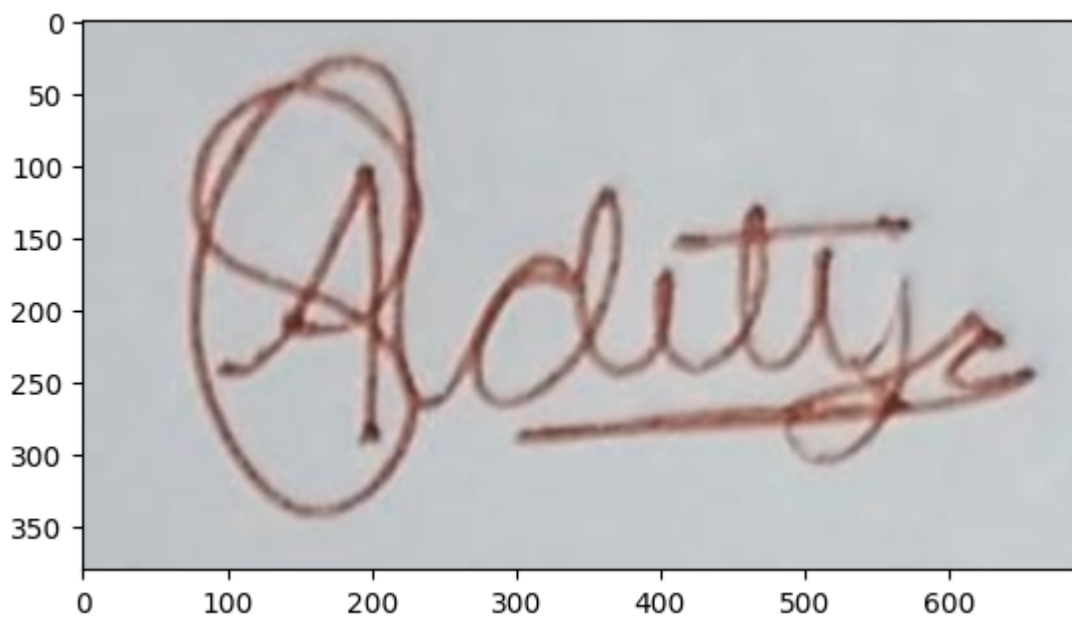
```
In [9]: test = cv.imread(TEST_IMAGE)
plt.imshow(test)
```

Out[9]: <matplotlib.image.AxesImage at 0x194e9cad0>



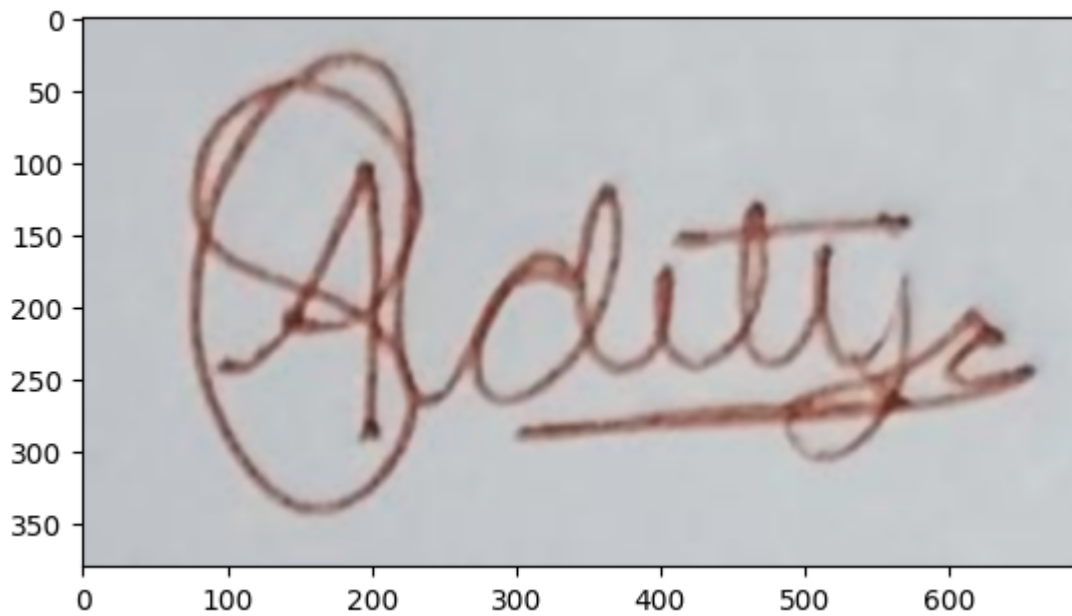
```
In [10]: test_blur = cv.GaussianBlur(test, (5,5), None)
plt.imshow(test_blur)
```

Out[10]: <matplotlib.image.AxesImage at 0x195469a50>



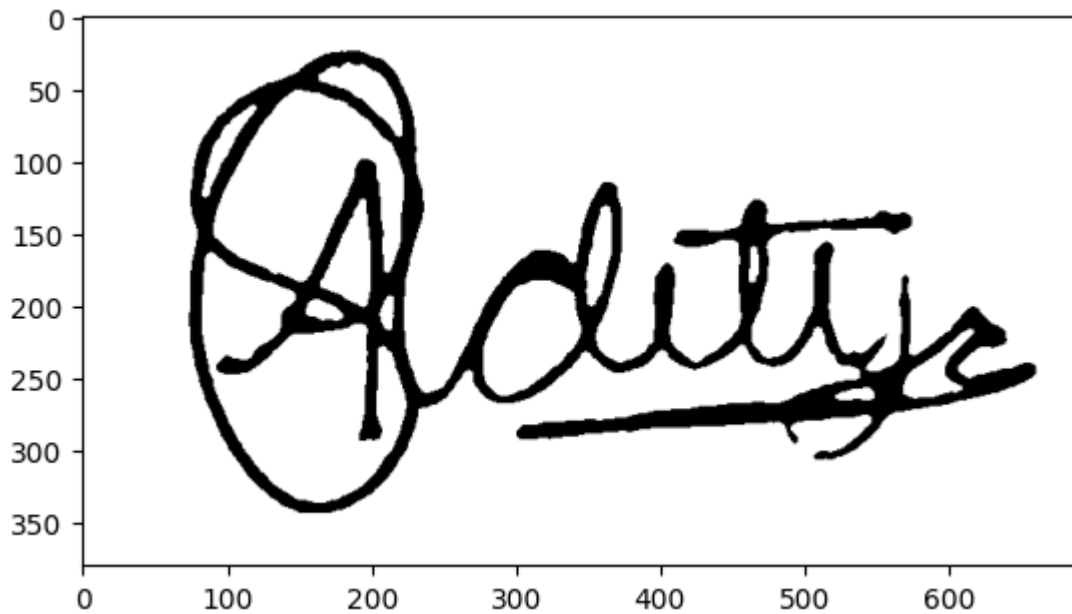
```
In [11]: test_close = cv.morphologyEx(test_blur, cv.MORPH_OPEN, cv.getStructuringE
plt.imshow(test_close)
```

Out[11]: <matplotlib.image.AxesImage at 0x195501a50>



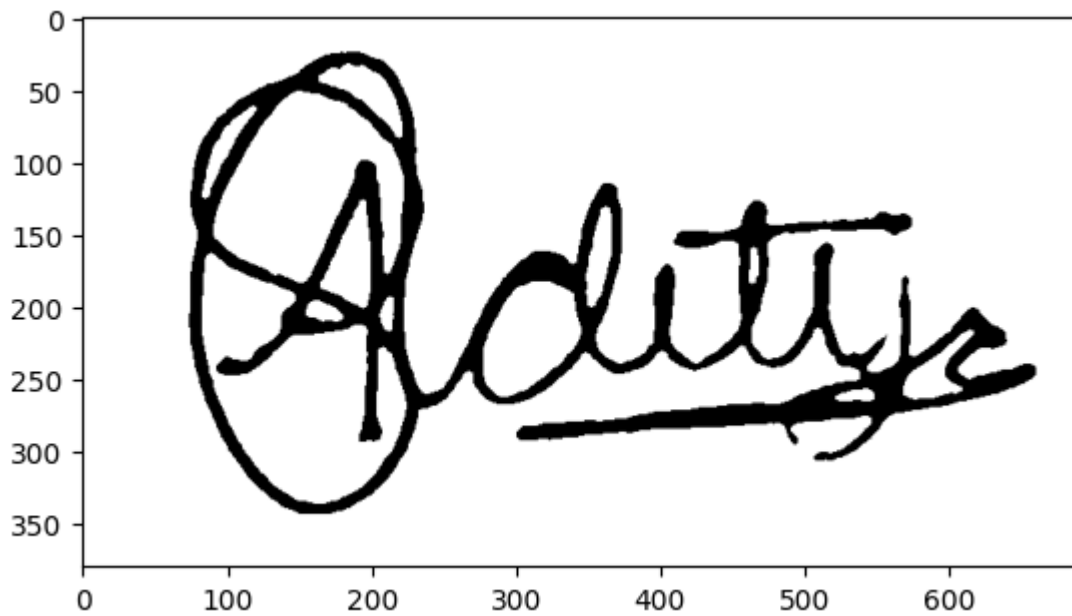
```
In [12]: test_g = cv.cvtColor(test_close, cv.COLOR_BGR2GRAY)
_, test_bw = cv.threshold(test_g, 0, 255, cv.THRESH_BINARY+cv.THRESH_OTSU)
plt.imshow(test_bw, cmap="gray")
```

Out[12]: <matplotlib.image.AxesImage at 0x195598610>



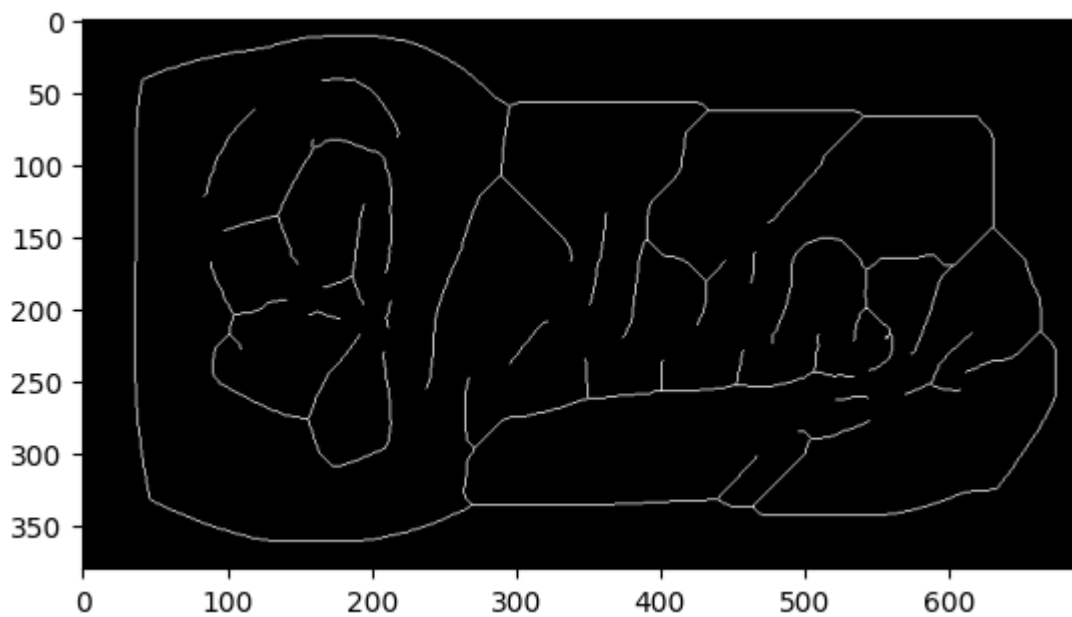
```
In [13]: test_closed = cv.morphologyEx(test_bw, cv.MORPH_OPEN, cv.getStructuringEl
plt.imshow(test_closed, cmap="gray")
```

Out[13]: <matplotlib.image.AxesImage at 0x17d6f0610>



```
In [14]: test_skeleton = skimage.morphology.skeletonize(test_closed)
test_skeleton = skimage.util.img_as_ubyte(test_skeleton)
test_skeleton = cv.resize(test_skeleton, (img_skeleton.shape[1], img_skeleton.shape[0]))
plt.imshow(test_skeleton, cmap="gray")
```

Out[14]: <matplotlib.image.AxesImage at 0x195648610>



```
In [15]: skimage.metrics.structural_similarity(img_skeleton, test_skeleton)
```

Out[15]: 0.7153751527894728

Result

We obtain a 71% match between 2 signatures