MUEI/MNUR
ETSEIB

Course on Medical Imaging
2023-2024
Q1

Filtering and segmentation

Dani Tost

Last week, we studied the concept of histogram, cdf, brighness and contrast.

We experienced applying filters of contrast enhancement, noise removal and mathematical morphology.

Today we'll continue with new filters. Remember that you have to include at least one method of each type in the class RasImage.

Edge detection

Sobel filter

Sobel filter (Sobel–Feldman) operator is used to given an image, create a new image emphasizing edges of the original image. The new image is a 2-D map of the discrete gradient at each pixel. The areas of high gradient are transitions between regions (likely edges). They appear as white lines in the image.

It consists of a 3x3 convolution with the matrices:

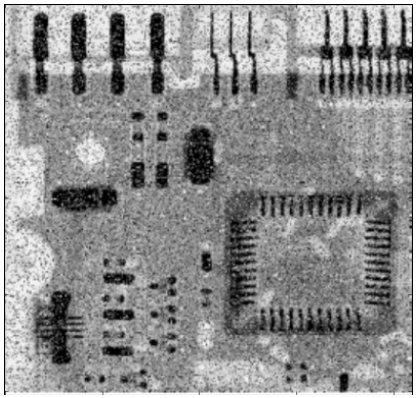
$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

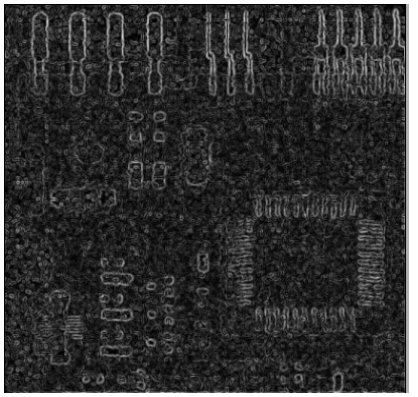
Similar filters are: Laplace, Sharr, Prewitt, Roberts Cross (see a comparison in https://en.wikipedia.org/wiki/Sobel_operator)

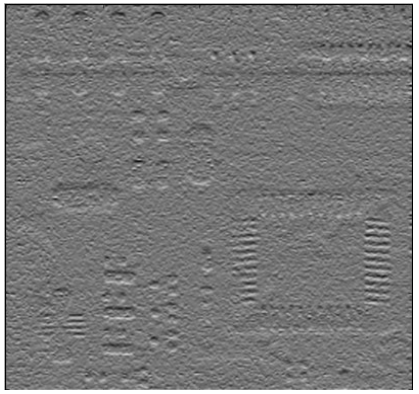
VIP: edge detection does not extract edges, it outlines them

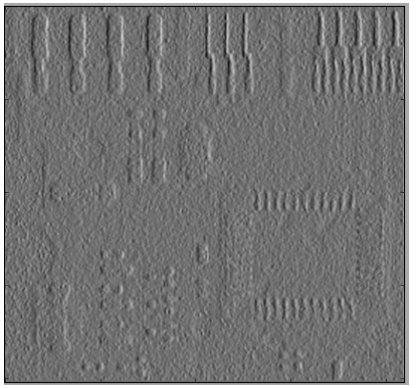
Sobel filter

```
>>> img = io.imread(“dirty.jpg”)
>>> img_sh = filters.sobel_h(img)
>>> img_sv = filters.sobel_v(img)
>>> img_s = filters.sobel(img)
```









Canny filter

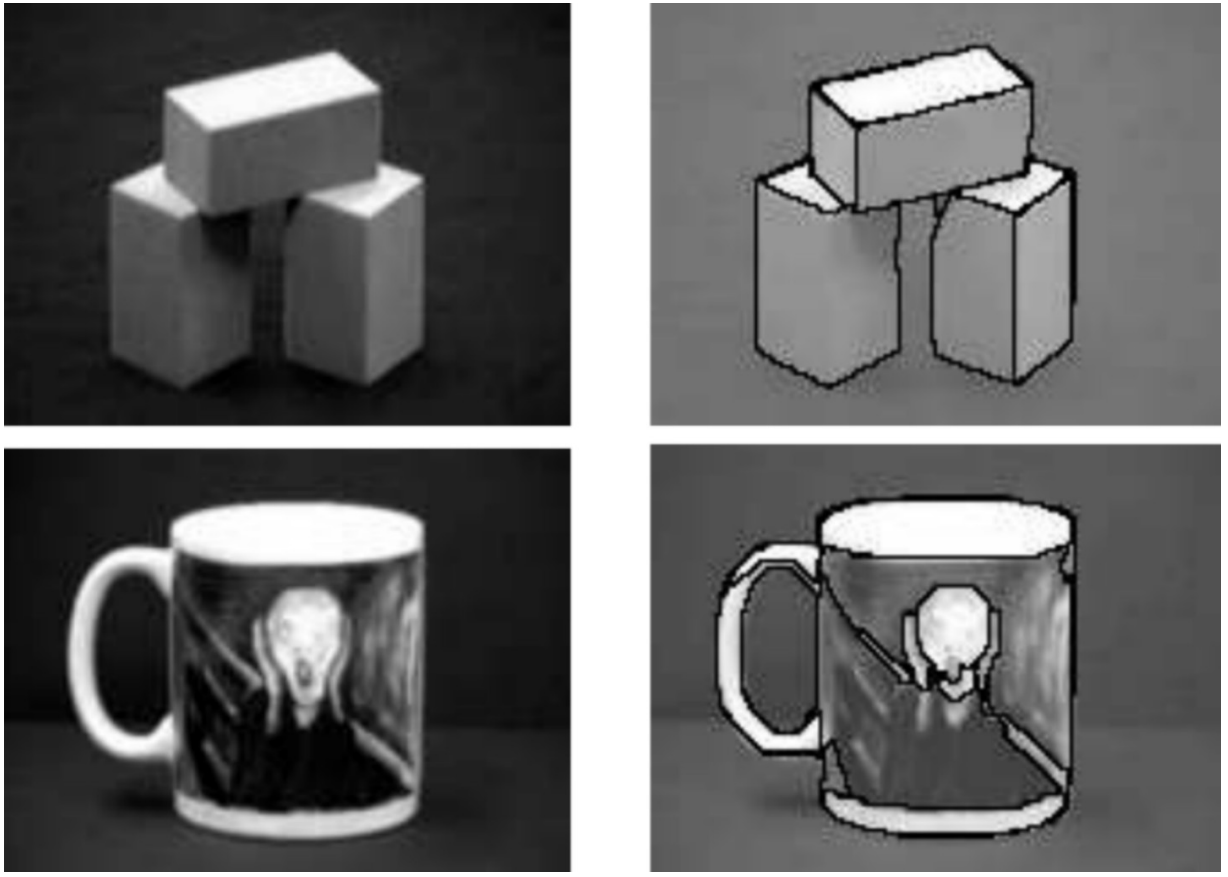
```
>>> img = io.imread(“dirty.jpg”)
>>> img1 = feature.canny(img, sigma=1)
>>> img2 = feature.canny(img, sigma=2)
>>> img3 = feature.canny(img, sigma=5)
```




Segmentation

Segmentation

Clustering of pixels/voxels into regions representing objects or parts of objects.



From Jepson and Fleet, 2011

Segmentation of medical images

Locate tumors and other pathologies

Extract the surface of features

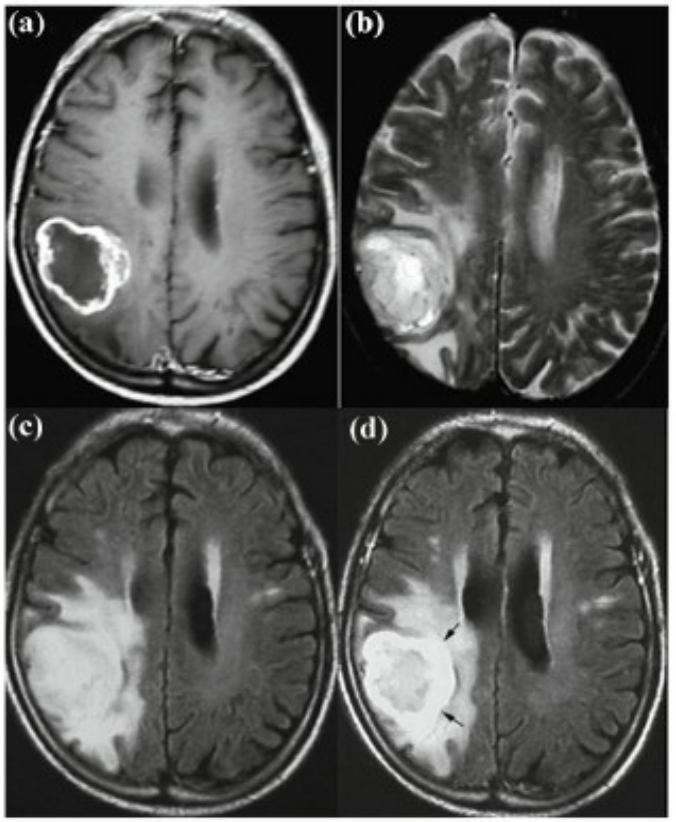


Fig. 1 Four imaging modalities: (a) T1-weighted MRI; (b) T2-weighted MRI; (c) FLAIR; and (d) FLAIR with contrast enhancement[5].

Image from A Survey of MRI-Based Brain Tumor Segmentation Methods, in Liu, Min Li, Jianxin Wang, Fangxiang Wu, Tianming Liu, and Yi Pan, TSINGHUA SCIENCE AND TECHNOLOGY, 1007-021404/10, pp578-595, Volume 19, Number 6, December 2014

Segmentation

Segmentation techniques of medical images are specific to application, imaging modality and body part.

Segmentation often requires pre-filtering to solve problems such as:

- Intra and inter slice variation of intensity
- Partial volume effect
- Different artifacts: example motion artifacts, ring artifacts, etc
- Noise due to sensors and related electronic system.

References:

Withey DJ, Koles ZJ. Three generations of medical image segmentation: Methods and available software. Int J Bioelectromag. 2007;9:67–8.

Sharma, N., & Aggarwal, L. M. (2010). Automated medical image segmentation techniques. Journal of Medical Physics / Association of Medical Physicists of India, 35(1), 3–14. http://doi.org/10.4103/0971-6203.58777

Threshold segmentation

1. Find the value threshold of a region
2. Mark the pixels with a value higher(lower) that the threshold

```
for every pixel in the image:
        if pixel_value >= threshold:    (idem with <)
            mark pixel as inside
        else:
            mark pixel as outside
```

No suitable for segmentation of regions having the same value range.

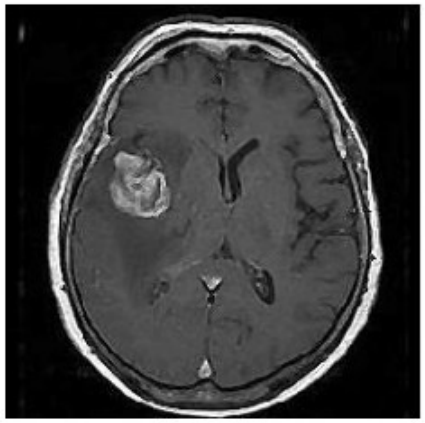
Better on CT than on MR without contrast

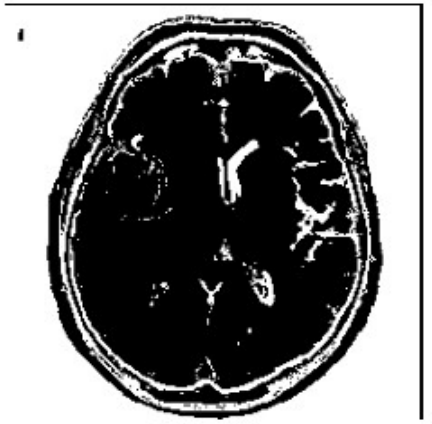
Threshold segmentation

```
# threshold

    filtered = []
    brain = io.imread('brain.jpg')
    filtered.append(brain> 200)
    t_otsu = filters.threshold_otsu(brain)
    filtered.append(brain> t_otsu)
    filtered.append(brain<= t_otsu)
    render_n(brain, filtered, "gray")
```

200 is an arbitrary value. Try others. Which is the Otsu threshold?



RAG segmentation methods

The aim is to split the image into perceptual groups and create a Region Adjacency Graph

There are different methods and in all, the number or size of the groups can be parameterized.

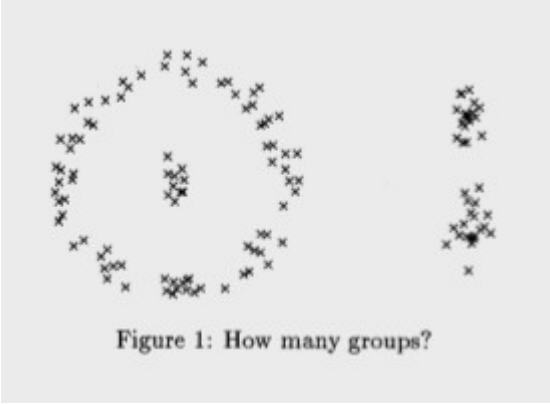


Figure 1: How many groups?

http://scikit-image.org/docs/dev/auto_examples/segmentation/plot_ncut.html#sphx-glr-auto-examples-segmentation-plot-ncut-py

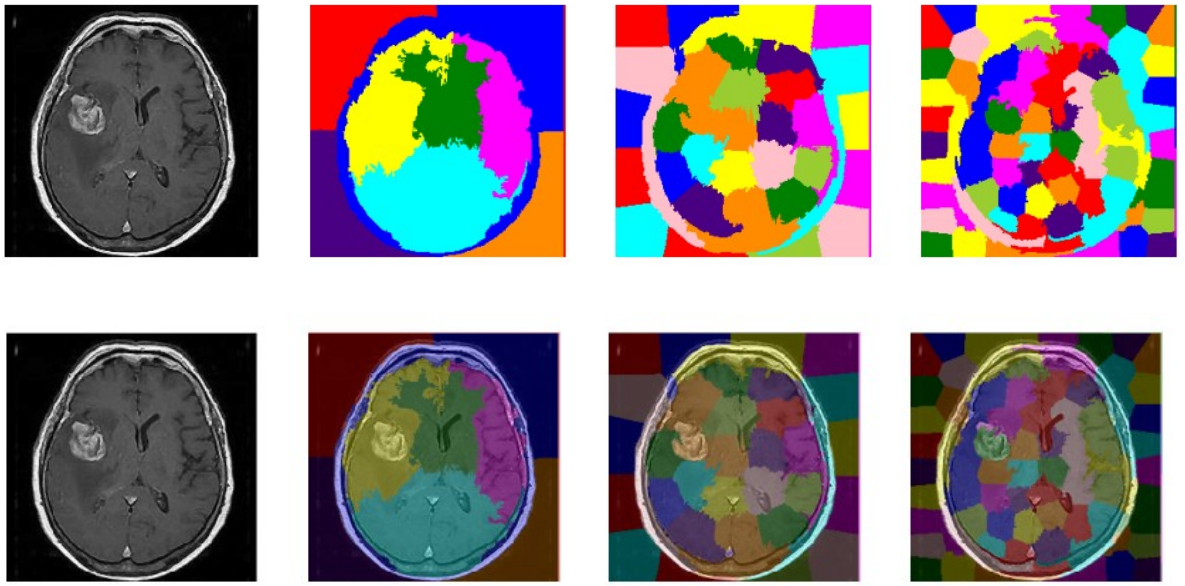
hi, J.; Malik, J., "Normalized cuts and image segmentation", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 22, no. 8, pp. 888-905, August 2000.

SLIC

SLIC (Simple Linear Iterative Clustering) super pixels

```
#segmentation

filtered = []
brain_rgb = color.gray2rgb(brain)
labels = segmentation.slic(brain_rgb, n_segments=10, start_label=1)
filtered.append(color.label2rgb(labels, bg_label=0))
labels = segmentation.slic(brain_rgb, n_segments=50, start_label=1)
filtered.append(color.label2rgb(labels, bg_label=0))
labels = segmentation.slic(brain_rgb, n_segments=100, start_label=1)
filtered.append(color.label2rgb(labels, bg_label=0))
render_n(brain_rgb, filtered, "gray")
```



To view segments on top of brain, add the parameter brain_rgb to label2rgb:

```
color.label2rgb(labels, brain_rgb, bg_label=0
```

RAG & median cut

Computes a region adjacency graph and uses it to merge neighboring regions of similar color.

```
orig2 = data.coffee()

filtered = []

labels = segmentation.slic(orig2, n_segments=50, start_label=1)

filtered.append(color.label2rgb(labels, orig2, bg_label=0))

g = graph.rag_mean_color(orig2, labels, mode='similarity')

labels2 = graph.cut_normalized(labels, g, thresh=90, num_cuts=4)

filtered.append(color.label2rgb(labels2, orig2, bg_label=0))

render_n(orig2, filtered, "gray")
```

Contour extraction

Create contours images

Goal: to reconstruct the polygonal contour of segmented region

Convert raster information to vectorial information

“To vectorize”

We’ll study the Marching square method later, in 3D. Here, we’ll only use it.

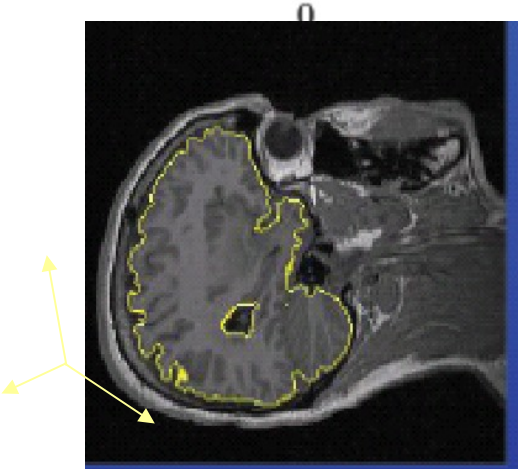
Active contours

Partial Differential Equations methods

Evolve a curve until it fits the boundary of the region

Active contour (snake) method energy function

$$E^*_{snake} = \int_0^1 E_{snake}(\mathbf{v}(s))\,ds = \int_0^1 (E_{internal}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) + E_{con}(\mathbf{v}(s)))\,ds$$



Active contour

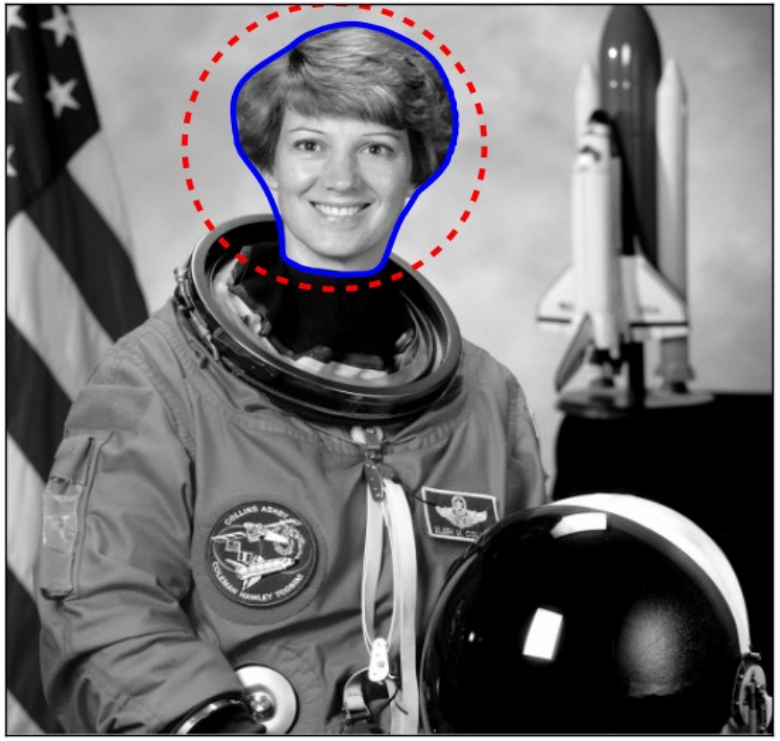
1. Create a first snake (e.g a circular shape)

```
s = np.linspace(0, 2*np.pi, 400)
x = center[0] + radius*np.cos(s)
y = center[1] + radius*np.sin(s)
snake_ini = np.array([x, y]).T
```

2. Apply

```
snake_fi = active_contour(image,
snake_ini)
```

The result can be tuned playing with parameters alpha, betta gamma



http://scikit-image.org/docs/dev/auto_examples/edges/plot_active_contours.html#sphx-glr-auto-examples-edges-plot-active-contours-py

Active contour

```
#snakes
 s = np.linspace(0, 2*np.pi, 400)
 center = [100, 220]
 radius = 100
 r = center[0] + radius*np.sin(s)
 c = center[1] + radius*np.cos(s)
 snake_ini = np.array([r, c]).T
 img = filters.gaussian(orig_gray, 3, preserve_range=False)
 snake_fi = segmentation.active_contour(img, snake_ini, alpha=0.015, beta=10, gamma=0.001)
 render_with_contours(orig_gray, [snake_ini, snake_fi])


def render_with_contours(orig, contours):
    fig, sbplt = plt.subplots(1)
    sbplt.imshow(orig)
    for contour in contours:
        sbplt.plot(contour[:, 1], contour[:, 0])
    sbplt.axis("off")
```

Marching Squares

http://scikit-image.org/docs/dev/api/skimage.measure.html

find_contours

skimage.measure.**find_contours**(*array*, *level*, *fully_connected='low'*, *positive_orientation='low'*)
[source]

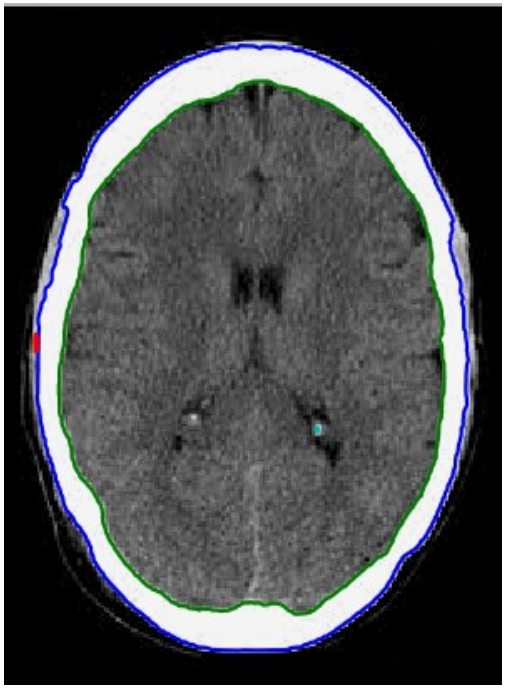
Find iso-valued contours in a 2D array for a given level value.

Uses the "marching squares" method to compute a the iso-valued contours of the input 2D array for a particular level value. Array values are linearly interpolated to provide better precision for the output contours.

Read the brain image
Convert it to gray and then:

```
>>> c = measure.find_contours(im, 0.9)
>>> len(c)
4                                   4 contours found
>>> c[0]
array([[ 238.00235849,  114.        ],
       [ 238.08536585,  113.        ],
       [ 238.13212435,  112.        ],
       ...,
```

Plot the contours (reverse the coordinates)



An example in: http://scikit-image.org/docs/dev/auto_examples/plot_contours.html

Marching Squares

http://scikit-image.org/docs/dev/api/skimage.measure.html

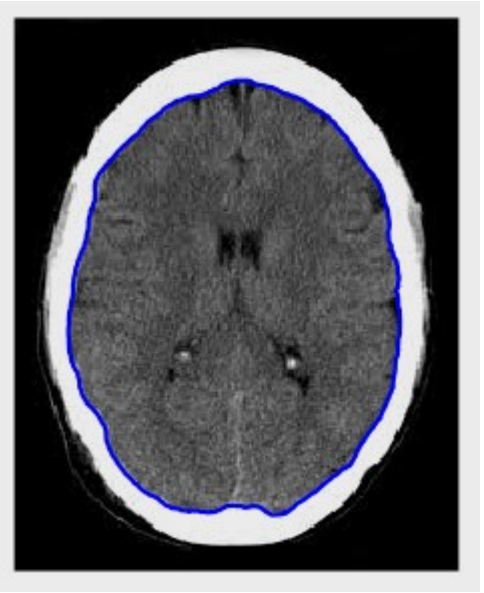
Try to determine which is the brain contour.

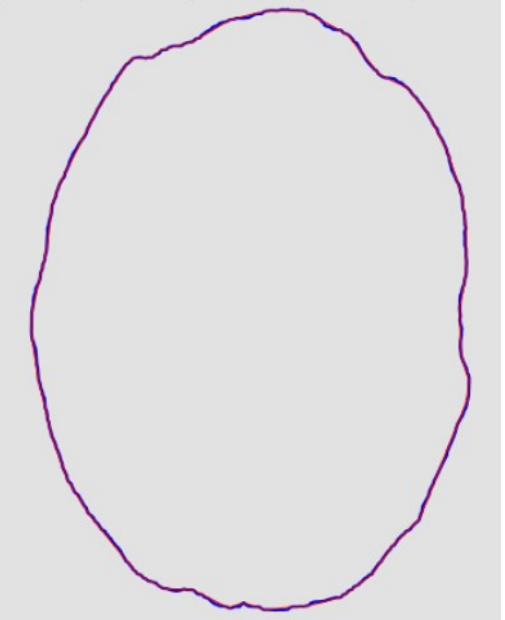
Count how many points it has? (715)

Try to simplify it with *approximate_polygon*

Store the contour in a file.

Read it and plot it with matplotlib.





Next week, after the quiz, we'll talk about vectorial images and rasterization