

Report 7

Contour from mask and image using skimage and active contour

Vibhav Joshi

10th June, 2020

Contour using skimage

Link to google colab file:

<https://colab.research.google.com/drive/1ZBLwAo200uqlbEBSiE-du3Ewf1XHI2s7?usp=sharing>

Contour finding

We use a marching squares method to find constant valued contours in an image. In `skimage.measure.find_contours`, array values are linearly interpolated to provide better precision of the output contours. Contours which intersect the image edge are open; all others are closed.

The `marching squares algorithm` is a special case of the marching cubes algorithm (Lorensen, William and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics SIGGRAPH 87 Proceedings) 21(4) July 1987, p. 163-170).

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from skimage import measure
```

```
# Construct some test data
```

```
x, y = np.ogrid[-np.pi:np.pi:100j, -np.pi:np.pi:100j]
```

```
r = np.sin(np.exp((np.sin(x)**3 + np.cos(y)**2)))
```

```
# Find contours at a constant value of 0.8
```

```
contours = measure.find_contours(r, 0.8)
```

```
# Display the image and plot all contours found

fig, ax = plt.subplots()

ax.imshow(r, cmap=plt.cm.gray)

for n, contour in enumerate(contours):

    ax.plot(contour[:, 1], contour[:, 0], linewidth=2)

ax.axis('image')

ax.set_xticks([])

ax.set_yticks([])

plt.show()
```

Active Contour using skimage

Link to google colab file:

<https://colab.research.google.com/drive/11NzwtqbnCNVyZiv3DfLapuwghzOIVuvb?usp=sharing>

https://colab.research.google.com/drive/1CsG-4x6FJ-ClqDY4ESA_BhqZYda6Tas_?usp=sharing

Active Contour Model

The active contour model is a method to fit open or closed splines to lines or edges in an image ¹. It works by minimising an energy that is in part defined by the image and part by the spline's shape: length and smoothness. The minimization is done implicitly in the shape energy and explicitly in the image energy.

In the following two examples the active contour model is used (1) to segment the face of a person from the rest of an image by fitting a closed curve to the edges of the face and (2) to find the darkest curve between two fixed points while obeying smoothness considerations. Typically it is a good idea to smooth images a bit before analyzing, as done in the following examples.

We initialize a circle around the astronaut's face and use the default boundary condition `boundary_condition='periodic'` to fit a closed curve. The default parameters `w_line=0`, `w_edge=1` will make the curve search towards edges, such as the boundaries of the face.

```
import numpy as np

import matplotlib.pyplot as plt

from skimage.color import rgb2gray

from skimage import data

from skimage.filters import gaussian

from skimage.segmentation import active_contour


img = data.astronaut()

img = rgb2gray(img)


s = np.linspace(0, 2*np.pi, 400)

r = 100 + 100*np.sin(s)

c = 220 + 100*np.cos(s)

init = np.array([r, c]).T


snake = active_contour(gaussian(img, 3),


                        init, alpha=0.015, beta=10, gamma=0.001,

                        coordinates='rc')


fig, ax = plt.subplots(figsize=(7, 7))

ax.imshow(img, cmap=plt.cm.gray)

ax.plot(init[:, 1], init[:, 0], '--r', lw=3)
```

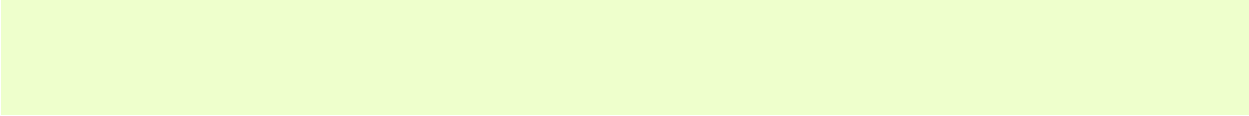


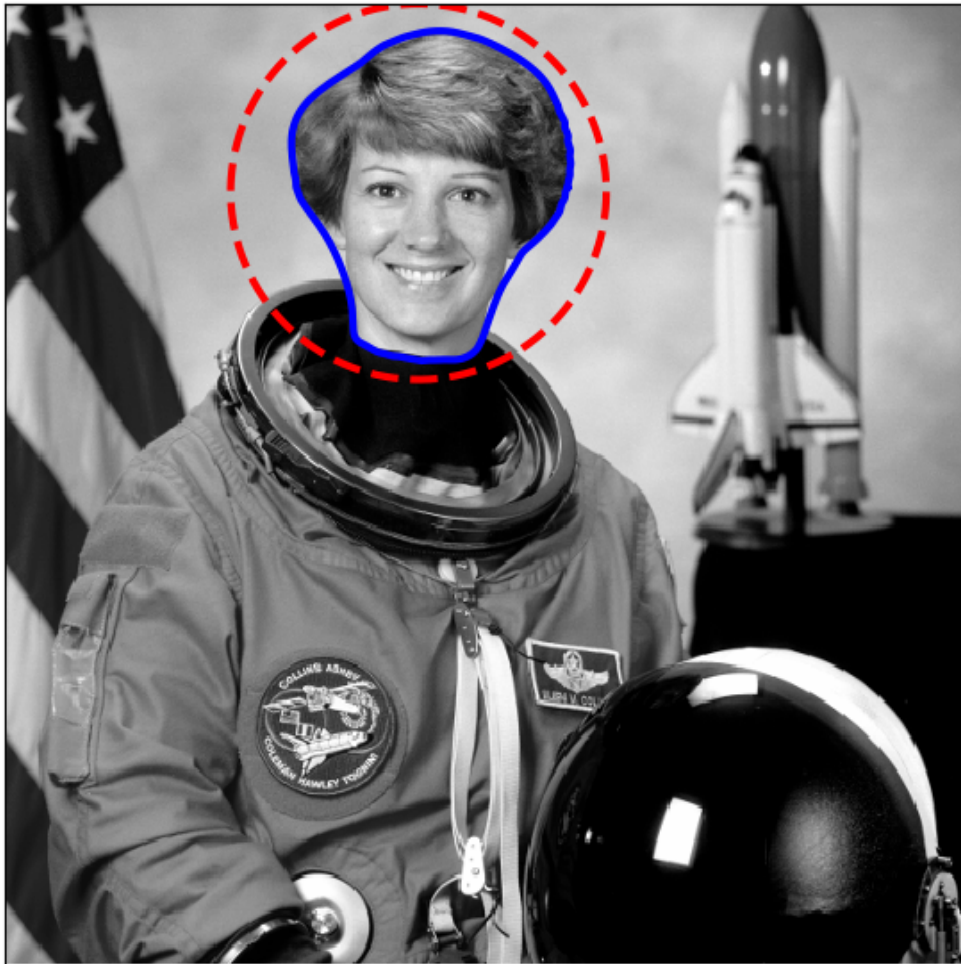
```
ax.plot(snake[:, 1], snake[:, 0], '-b', lw=3)
```

```
ax.set_xticks([]), ax.set_yticks([])
```

```
ax.axis([0, img.shape[1], img.shape[0], 0])
```

```
plt.show()
```





Here we initialize a straight line between two points, (5, 136) and (424, 50), and require that the spline has its end points there by giving the boundary condition *boundary_condition='fixed'*. We furthermore make the algorithm search for dark lines by giving a negative *w_line* value.

```
img = data.text()
```

```
r = np.linspace(136, 50, 100)
```

```
c = np.linspace(5, 424, 100)
```




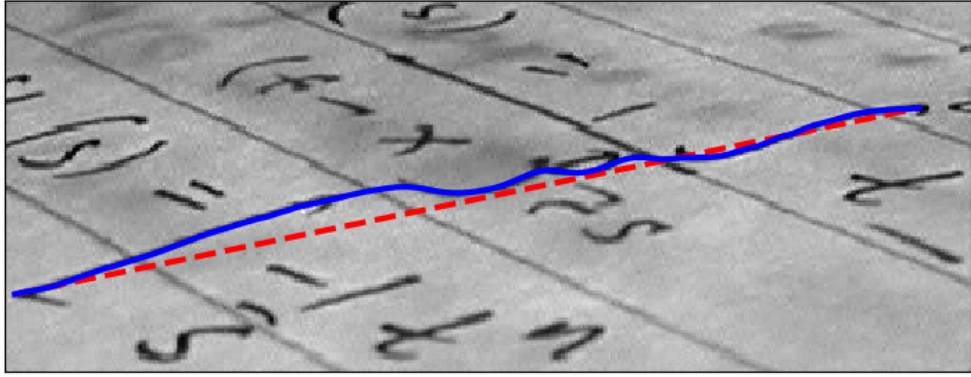
```
init = np.array([r, c]).T
```

```
snake = active_contour(gaussian(img, 1), init, boundary_condition='fixed',  
                        alpha=0.1, beta=1.0, w_line=-5, w_edge=0, gamma=0.1,  
                        coordinates='rc')
```

```
fig, ax = plt.subplots(figsize=(9, 5))  
ax.imshow(img, cmap=plt.cm.gray)  
ax.plot(init[:, 1], init[:, 0], '--r', lw=3)  
ax.plot(snake[:, 1], snake[:, 0], '-b', lw=3)  
ax.set_xticks([]), ax.set_yticks([])  
ax.axis([0, img.shape[1], img.shape[0], 0])
```

```
plt.show()
```





—