EDYCJA I PRZETWARZANIE OBRAZÓW PRZY UŻYCIU NUMPY I SCIPY

Prezentacja z języka Python

Bartosz Sroczyński, Robert Wojtaszek, Agnieszka Mach

Luty 2020

Politechnika Krakowska

OTWIERANIE I ZAPISYWANIE OBRAZÓW

```
from scipy import misc
f = misc.face()
misc.imsave('face.png', f) # uses the Image module (PIL)

import matplotlib.pyplot as plt
plt.imshow(f)
plt.show()
```

```
>>> from scipy import misc
>>> face = misc.face()
>>> misc.imsave('face.png', face) # First we need to create the PNG file

>>> face = misc.imread('face.png')
>>> type(face)
<... 'numpy.ndarray'>
>>> face.shape, face.dtype
((768, 1024, 3), dtype('uint8'))
```

OTWIERANIE I ZAPISYWANIE OBRAZÓW

```
>>> face.tofile('face.raw') # Create raw file
>>> face_from_raw = np.fromfile('face.raw', dtype=np.uint8)
>>> face_from_raw.shape
(2359296,)
>>> face_from_raw.shape = (768, 1024, 3)
```

```
>>> for i in range(10):
...     im = np.random.random_integers(0, 255, 10000).reshape((100, 100))
...     misc.imsave('random_%02d.png' % i, im)
>>> from glob import glob
>>> filelist = glob('random*.png')
>>> filelist.sort()
```

WYŚWIETLANIE OBRAZÓW

```
>>> f = misc.face(gray=True)  # retrieve a grayscale image
>>> import matplotlib.pyplot as plt
>>> plt.imshow(f, cmap=plt.cm.gray)
<matplotlib.image.AxesImage object at 0x...>
```

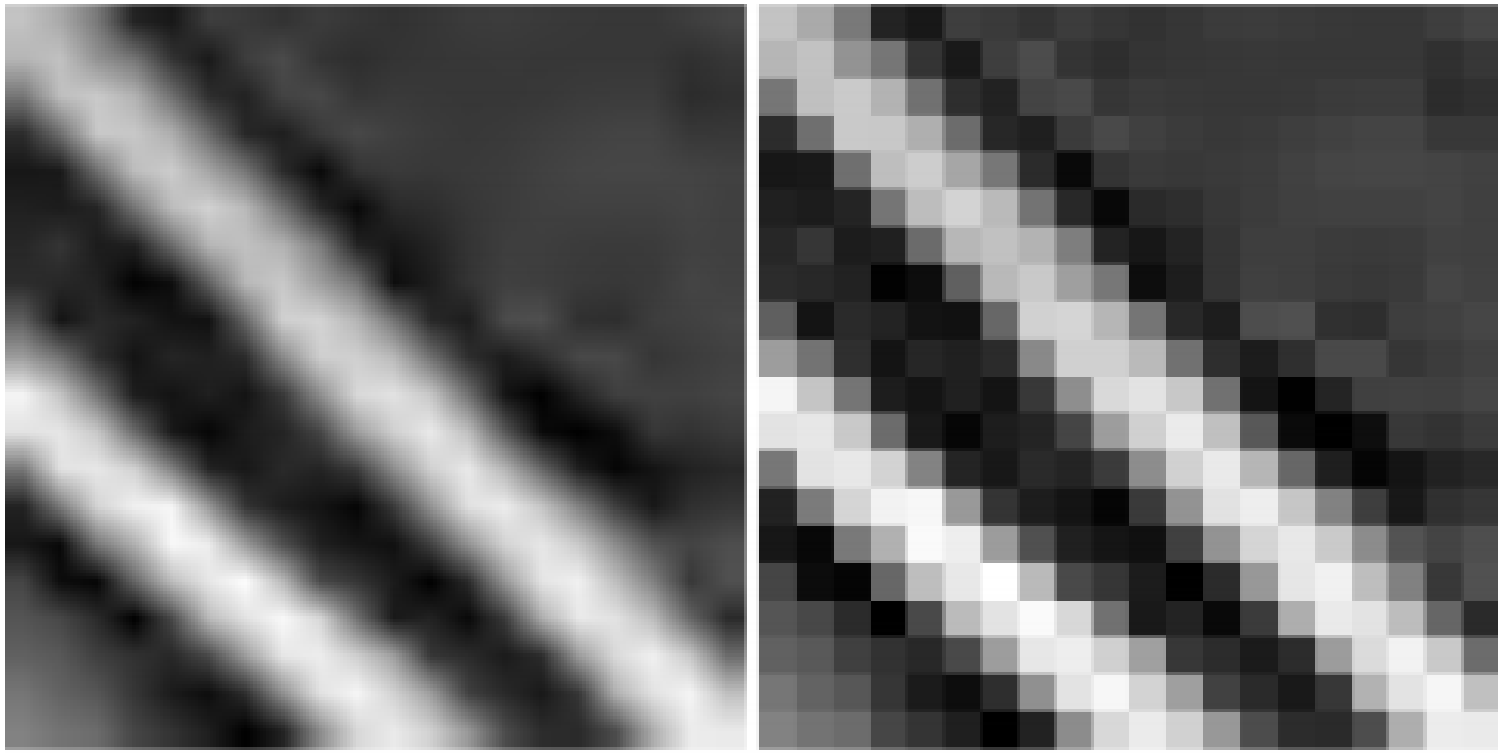
```
>>> plt.imshow(f, cmap=plt.cm.gray, vmin=30, vmax=200)
<matplotlib.image.AxesImage object at 0x...>
>>> # Remove axes and ticks
>>> plt.axis('off')
(-0.5, 1023.5, 767.5, -0.5)
```

WYŚWIETLANIE OBRAZÓW

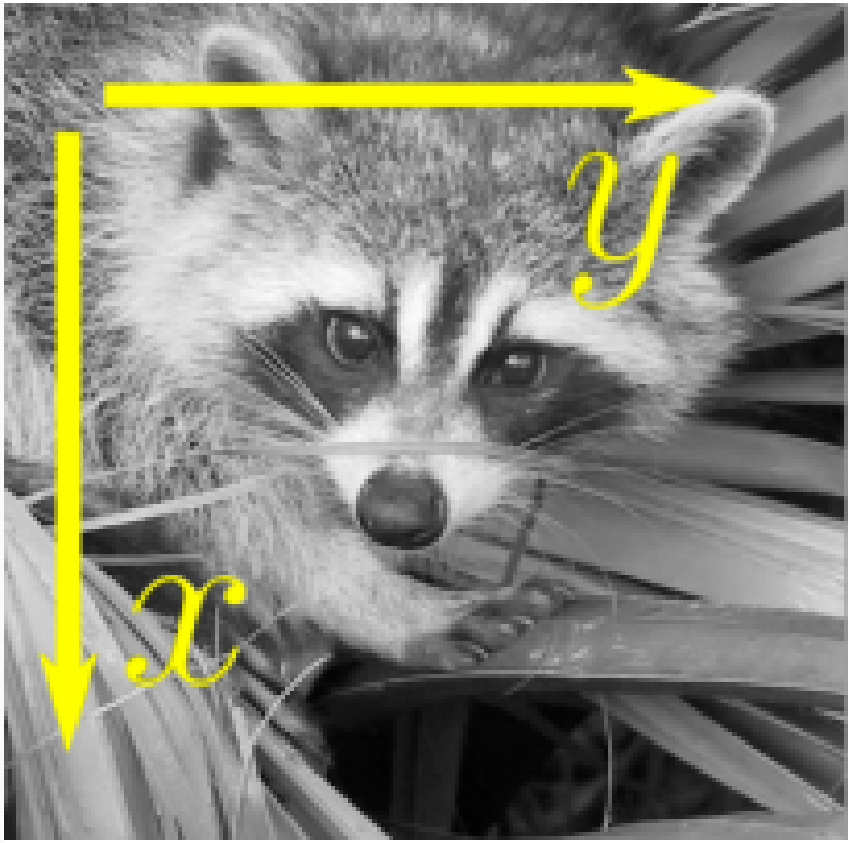
```
>>> plt.contour(f, [50, 200])
<matplotlib.contour.QuadContourSet ...>
```

WYŚWIETLANIE OBRAZÓW

```
>>> plt.imshow(f[320:340, 510:530], cmap=plt.cm.gray)
<matplotlib.image.AxesImage object at 0x...>
>>> plt.imshow(f[320:340, 510:530], cmap=plt.cm.gray, interpolation='nearest')
<matplotlib.image.AxesImage object at 0x...>
```



PODSTAWOWE MANIPULACJE



0	1	2
3	4	5
6	7	8

PODSTAWOWE MANIPULACJE

```
>>> face = misc.face(gray=True)
>>> face[0, 40]
127
>>> # Slicing
>>> face[10:13, 20:23]
array([[141, 153, 145],
       [133, 134, 125],
       [ 96,  92,  94]], dtype=uint8)
>>> face[100:120] = 255
>>>
>>> lx, ly = face.shape
>>> X, Y = np.ogrid[0:lx, 0:ly]
>>> mask = (X - lx / 2) ** 2 + (Y - ly / 2) ** 2 > lx * ly / 4
>>> # Masks
>>> face[mask] = 0
>>> # Fancy indexing
>>> face[range(400), range(400)] = 255
```

PODSTAWOWE MANIPULACJE

INFORMACJE STATYSTYCZNE

```
>>> face = misc.face(gray=True)
>>> face.mean()
113.48026784261067
>>> face.max(), face.min()
(250, 0)
```

MANIPULACJA GEOMETRYCZNA

```
>>> face = misc.face(gray=True)
>>> face.mean()
113.48026784261067
>>> face.max(), face.min()
(250, 0)
```

```
>>> face = misc.face(gray=True)
>>> lx, ly = face.shape
>>> # Cropping
>>> crop_face = face[lx / 4: - lx / 4, ly / 4: - ly / 4]
>>> # up <-> down flip
>>> flip_ud_face = np.flipud(face)
>>> # rotation
>>> rotate_face = ndimage.rotate(face, 45)
>>> rotate_face_noreshape = ndimage.rotate(face, 45, reshape=False)
```

ROZMYCIE

```
>>> from scipy import misc
>>> face = misc.face(gray=True)
>>> blurred_face = ndimage.gaussian_filter(face, sigma=3)
>>> very_blurred = ndimage.gaussian_filter(face, sigma=5)
```

```
>>> local_mean = ndimage.uniform_filter(face, size=11)
```

OSTROŚĆ

```
>>> from scipy import misc
>>> face = misc.face(gray=True).astype(float)
>>> blurred_f = ndimage.gaussian_filter(face, 3)
```

```
>>> filter_blurred_f = ndimage.gaussian_filter(blurred_f, 1)
>>> alpha = 30
>>> sharpened = blurred_f + alpha * (blurred_f - filter_blurred_f)
```

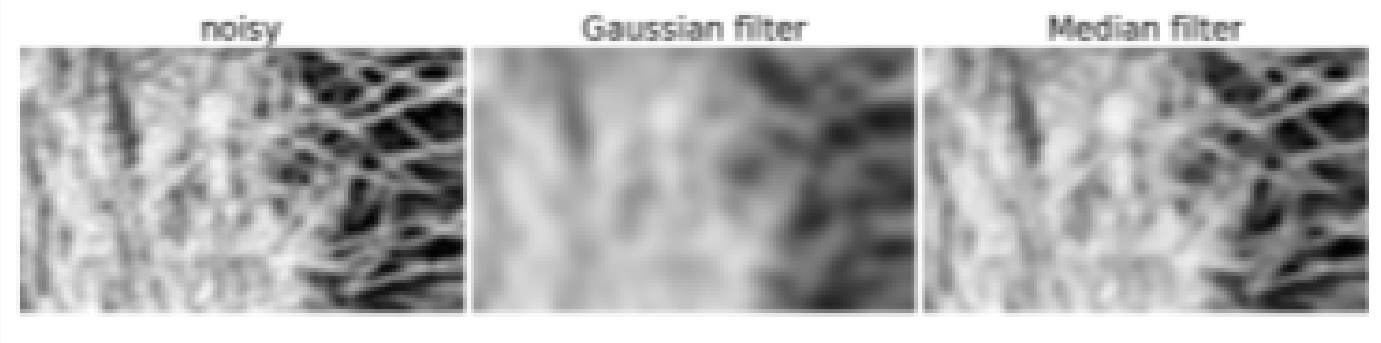


ROZDZIELCZOŚĆ

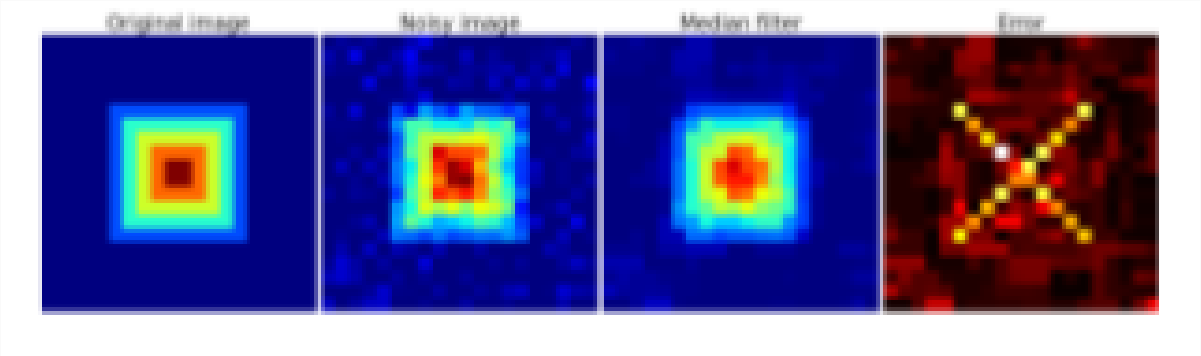
```
>>> from scipy import misc
>>> f = misc.face(gray=True)
>>> f = f[230:290, 220:320]
>>> noisy = f + 0.4 * f.std() * np.random.random(f.shape)
```

```
>>> gauss_denoised = ndimage.gaussian_filter(noisy, 2)
```

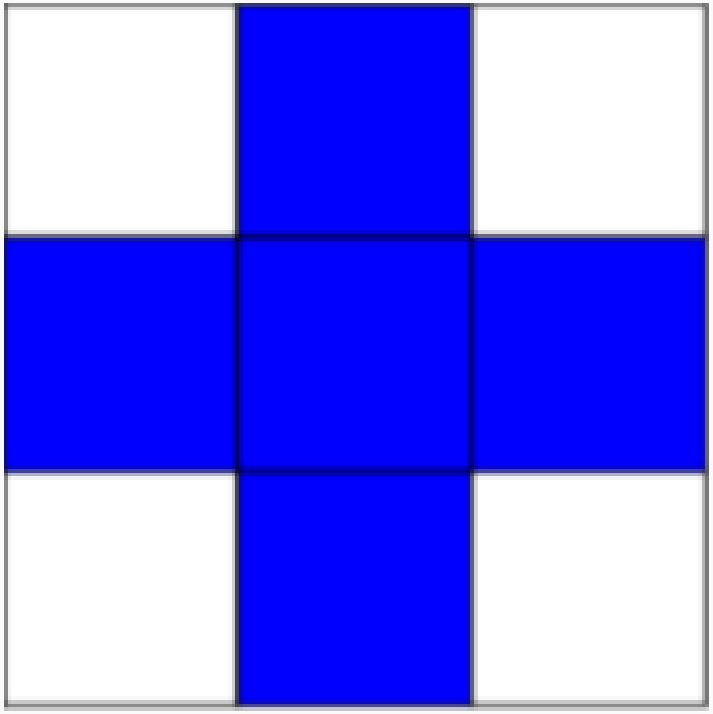
```
>>> med_denoised = ndimage.median_filter(noisy, 3)
```



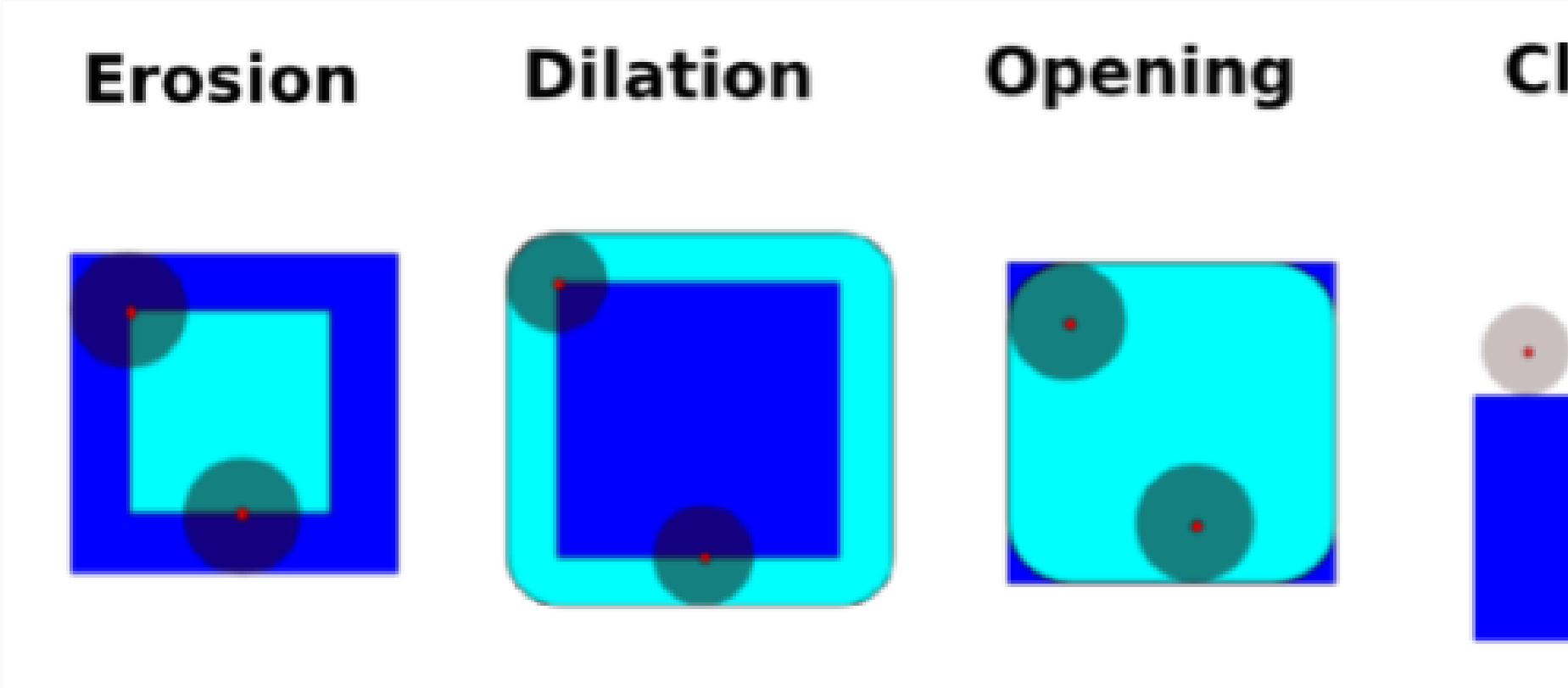
```
>>> im = np.zeros((20, 20))
>>> im[5:-5, 5:-5] = 1
>>> im = ndimage.distance_transform_bf(im)
>>> im_noise = im + 0.2 * np.random.randn(*im.shape)
>>> im_med = ndimage.median_filter(im_noise, 3)
```



```
>>> el = ndimage.generate_binary_structure(2, 1)
>>> el
array([[False,  True, False],
       [ True,  True,  True],
       [False,  True, False]], dtype=bool)
>>> el.astype(np.int)
array([[0, 1, 0],
       [1, 1, 1],
       [0, 1, 0]])
```

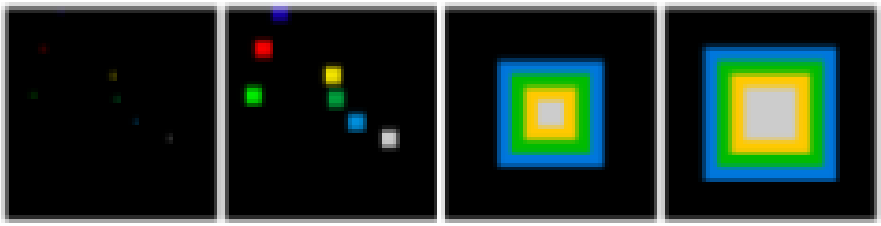


```
>>> a = np.zeros((7,7), dtype=np.int)
>>> a[1:6, 2:5] = 1
>>> a
array([[0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 1, 0, 0],
       [0, 0, 1, 1, 1, 0, 0],
       [0, 0, 1, 1, 1, 0, 0],
       [0, 0, 1, 1, 1, 0, 0],
       [0, 0, 1, 1, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0]])
>>> ndimage.binary_erosion(a).astype(a.dtype)
array([[0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0]])
>>> #Erosion removes objects smaller than the structure
>>> ndimage.binary_erosion(a, structure=np.ones((5,5))).astype(a.dtype)
array([[0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0],
```


Erosion
Dilation
Opening
Cl

```
>>> a = np.zeros((5, 5))
>>> a[2, 2] = 1
>>> a
array([[ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.]])
>>> ndimage.binary_dilation(a).astype(a.dtype)
array([[ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  1.,  0.,  0.],
       [ 0.,  1.,  1.,  1.,  0.],
       [ 0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.]])
```

```
>>> np.random.seed(2)
>>> im = np.zeros((64, 64))
>>> x, y = (63*np.random.random((2, 8))).astype(np.int)
>>> im[x, y] = np.arange(8)

>>> bigger_points = ndimage.grey_dilation(im, size=(5, 5), structure=np.ones((

>>> square = np.zeros((16, 16))
>>> square[4:-4, 4:-4] = 1
>>> dist = ndimage.distance_transform_bf(square)
>>> dilate_dist = ndimage.grey_dilation(dist, size=(3, 3), \
...          structure=np.ones((3, 3)))
```

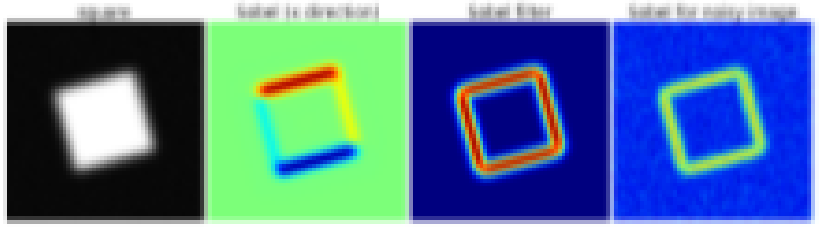


WYKRYWACZ KRAWĘDZI

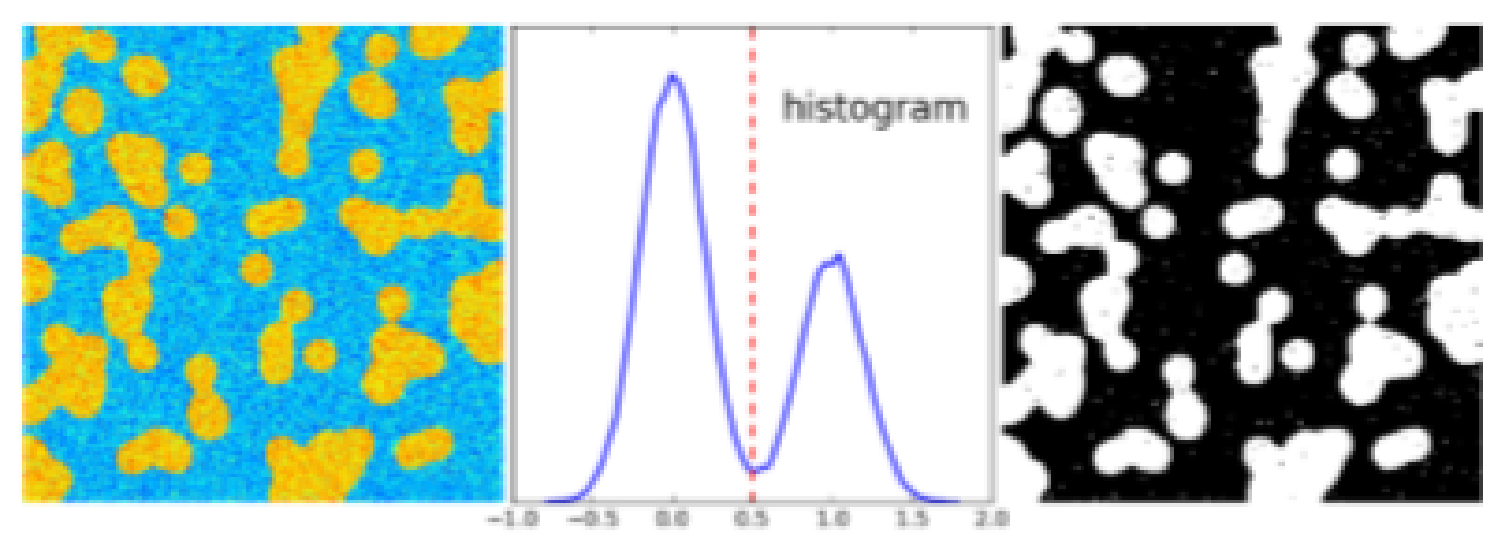
```
>>> im = np.zeros((256, 256))
>>> im[64:-64, 64:-64] = 1
>>>
>>> im = ndimage.rotate(im, 15, mode='constant')
>>> im = ndimage.gaussian_filter(im, 8)
```

Use a **gradient operator** (**Sobel**) to find high intensity variations:

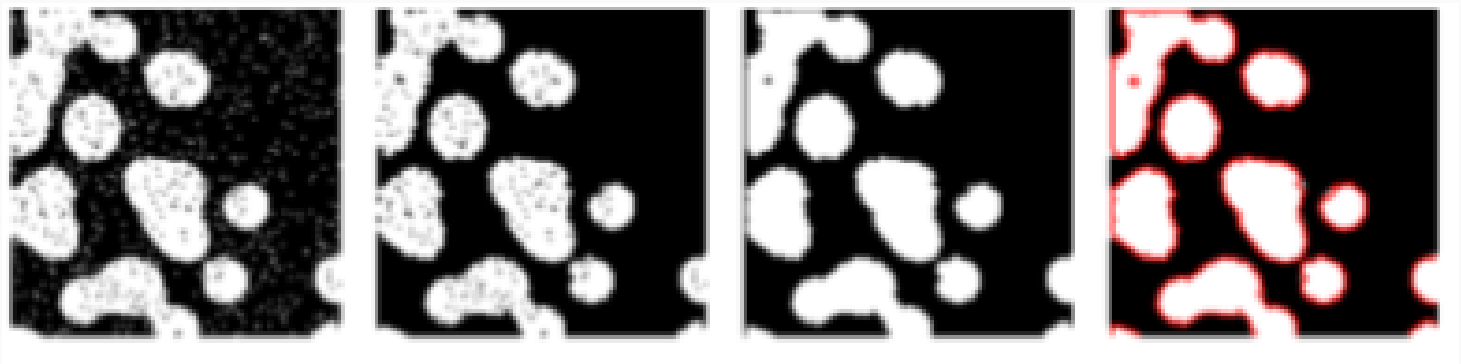
```
>>> sx = ndimage.sobel(im, axis=0, mode='constant')
>>> sy = ndimage.sobel(im, axis=1, mode='constant')
>>> sob = np.hypot(sx, sy)
```



```
>>> n = 10
>>> l = 256
>>> im = np.zeros((l, l))
>>> np.random.seed(1)
>>> points = l*np.random.random((2, n**2))
>>> im[(points[0]).astype(np.int), (points[1]).astype(np.int)] = 1
>>> im = ndimage.gaussian_filter(im, sigma=l/(4.*n))

>>> mask = (im > im.mean()).astype(np.float)
>>> mask += 0.1 * im
>>> img = mask + 0.2*np.random.randn(*mask.shape)

>>> hist, bin_edges = np.histogram(img, bins=60)
>>> bin_centers = 0.5*(bin_edges[:-1] + bin_edges[1:])

>>> binary_img = img > 0.5
```



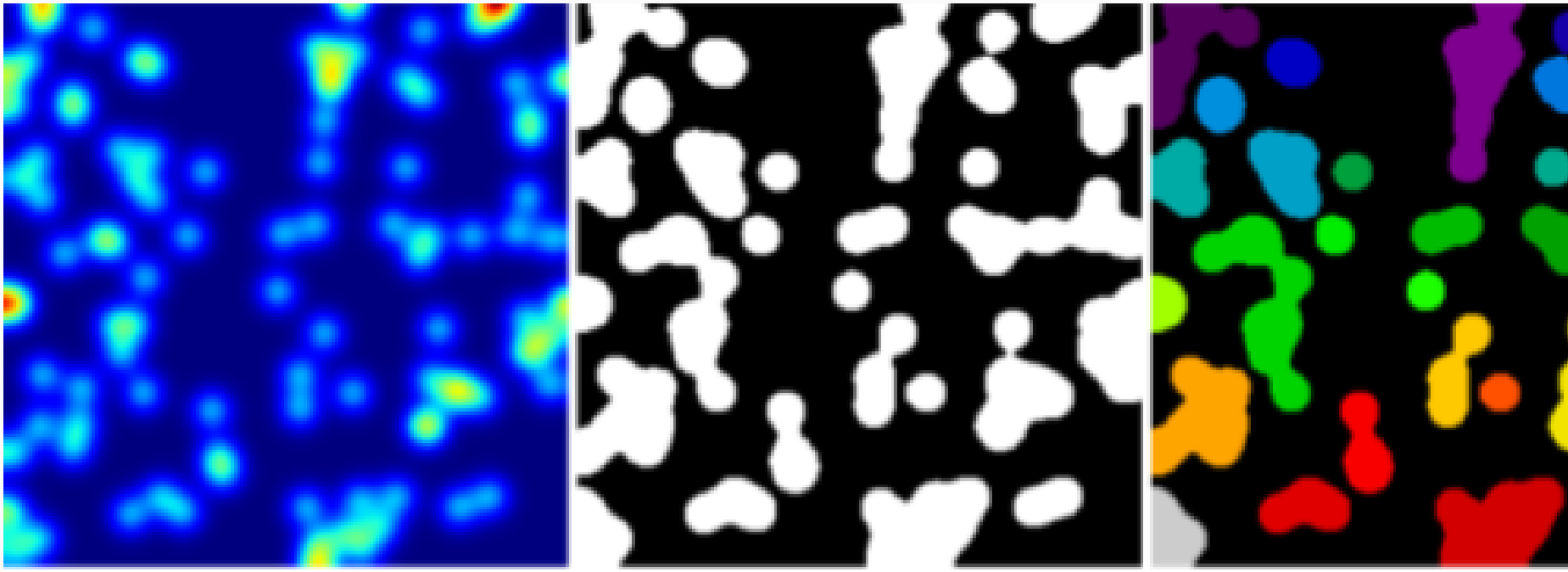

```
>>> # Remove small white regions
>>> open_img = ndimage.binary_opening(binary_img)
>>> # Remove small black hole
>>> close_img = ndimage.binary_closing(open_img)
```



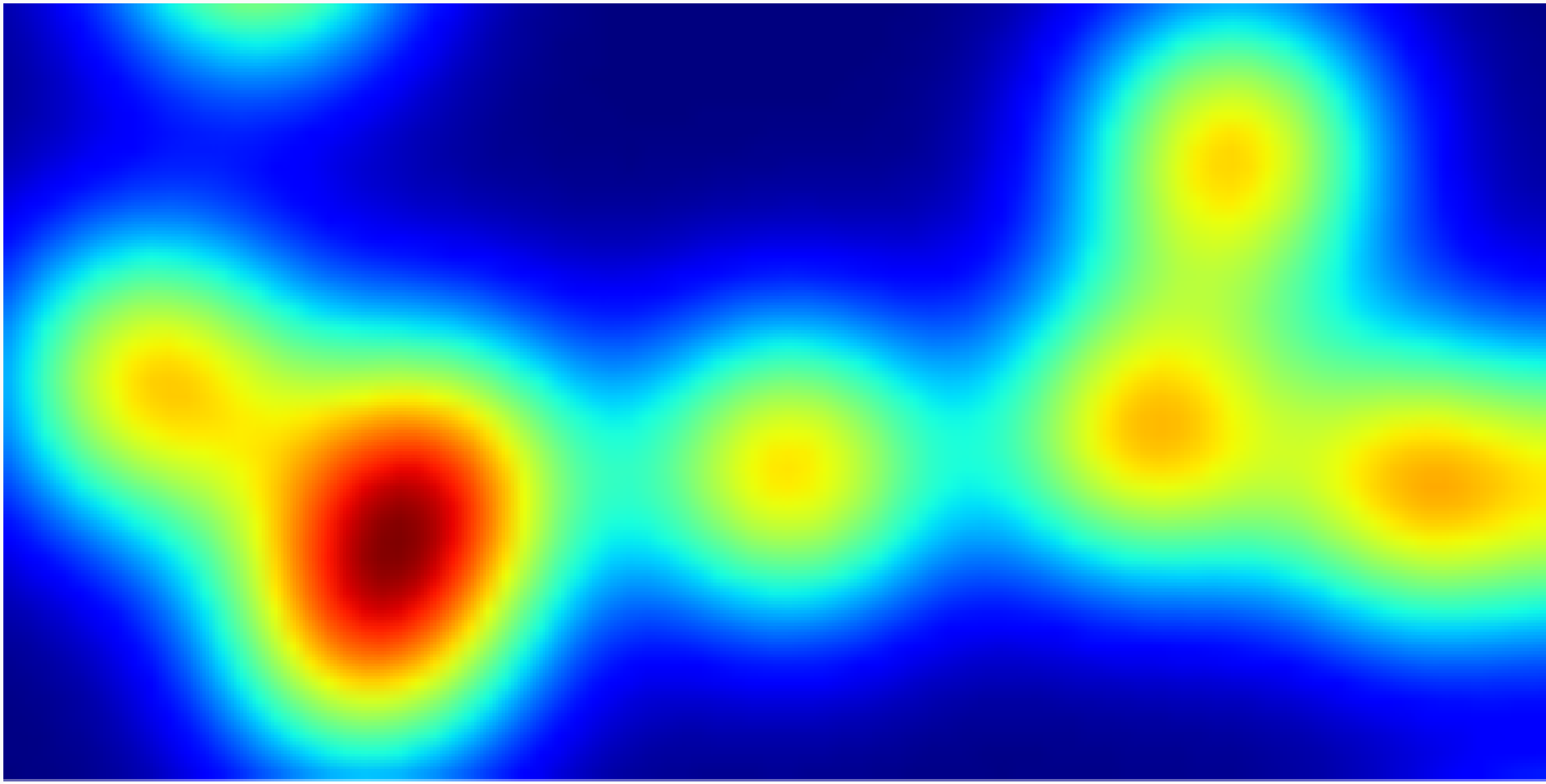
MIERZENIE OBIEKTÓW

```
>>> n = 10
>>> l = 256
>>> im = np.zeros((l, l))
>>> points = l*np.random.random((2, n**2))
>>> im[(points[0]).astype(np.int), (points[1]).astype(np.int)] = 1
>>> im = ndimage.gaussian_filter(im, sigma=l/(4.*n))
>>> mask = im > im.mean()
```

```
>>> label_im, nb_labels = ndimage.label(mask)
>>> nb_labels # how many regions?
16
>>> plt.imshow(label_im)
<matplotlib.image.AxesImage object at 0x...>
```



```
>>> slice_x, slice_y = ndimage.find_objects(label_im==4)[0]
>>> roi = im[slice_x, slice_y]
>>> plt.imshow(roi)
<matplotlib.image.AxesImage object at 0x...>
```



Dziękujemy za uwagę!