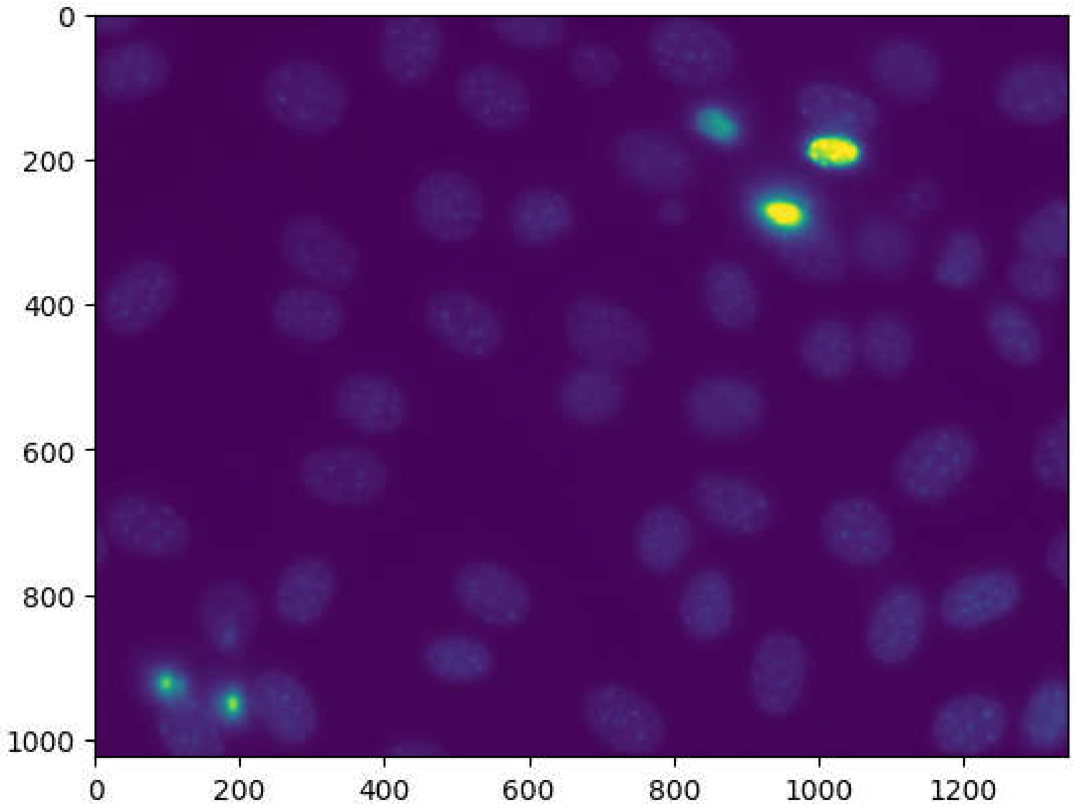
PCD BAB 12

Rahma Fairuz Rania

1. Penyaringan dan pelabelan ulang

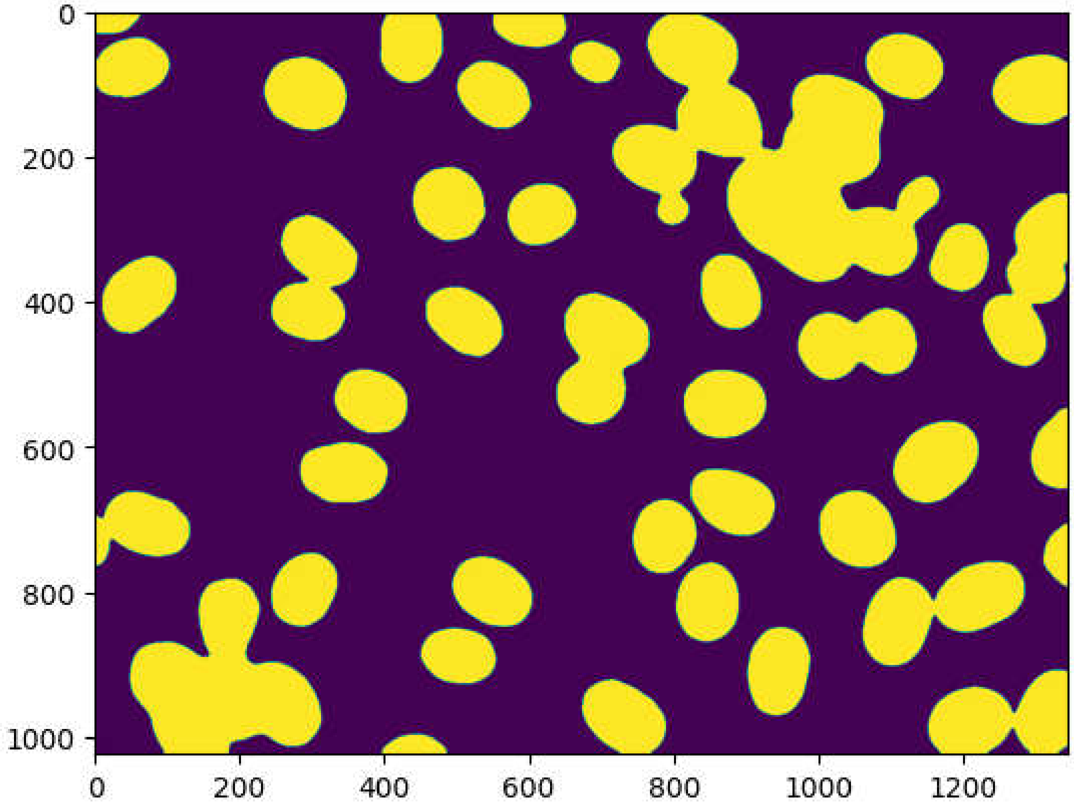
In [3]:

```
import mahotas as mh
import mahotas.demos
import numpy as np
from pylab import imshow, show

f = mh.demos.nuclear_image()
f = f[:,:,0]
imshow(f)
show()
```



In [4]:

```
f = mh.gaussian_filter(f, 4)
f = (f> f.mean())

f = mh.gaussian_filter(f, 4)
f = (f> f.mean())
imshow(f)
show()

labeled, n_nucleus = mh.label(f)
print('Found {} nuclei.'.format(n_nucleus))
```

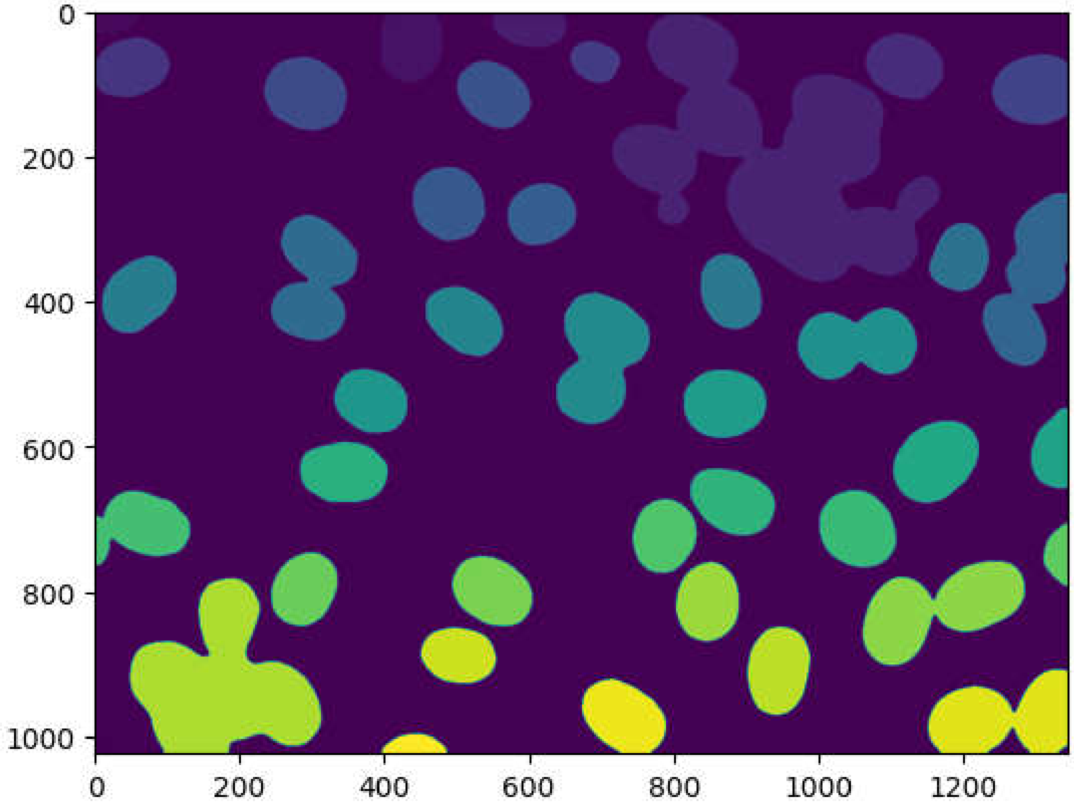


```
Found 40 nuclei.
```

In [5]:

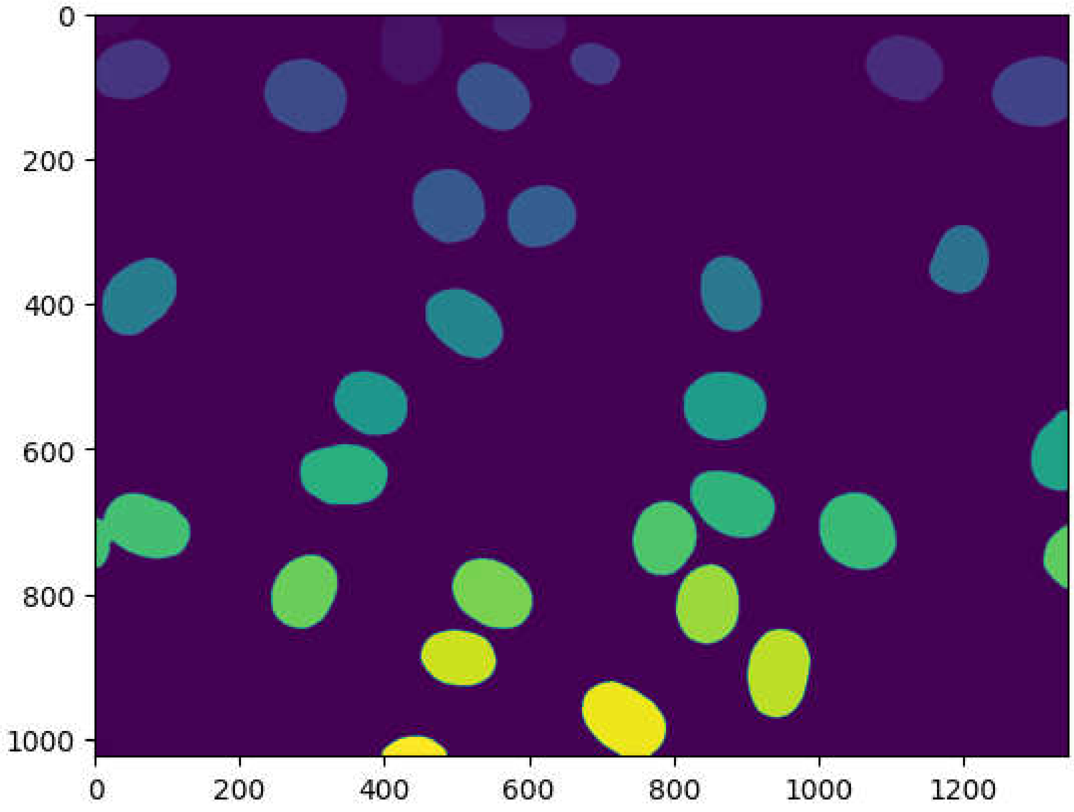
```
labeled, n_nucleus = mh.label(f)
print('Found {} nuclei.'.format(n_nucleus))
imshow(labeled)
show()
```

```
Found 40 nuclei.
```



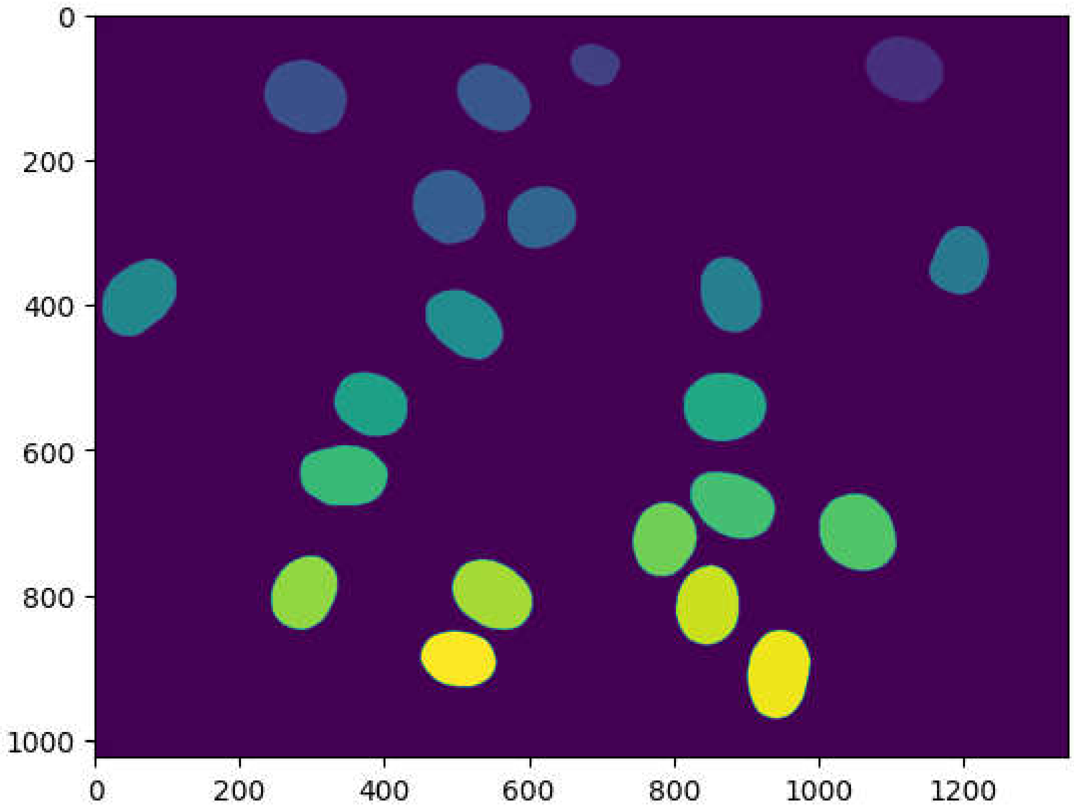
In [7]:

```
sizes = mh.labeled.labeled_size(labeled)
too_big = np.where(sizes > 10000)
labeled = mh.labeled.remove_regions(labeled, too_big)

sizes = mh.labeled.labeled_size(labeled)
too_big = np.where(sizes > 10000)
labeled = mh.labeled.remove_regions(labeled, too_big)
imshow(labeled)
show()
```



In [8]:

```
labeled = mh.labeled.remove_bordering(labeled)

labeled = mh.labeled.remove_bordering(labeled)
imshow(labeled)
show()
```



In [11]:

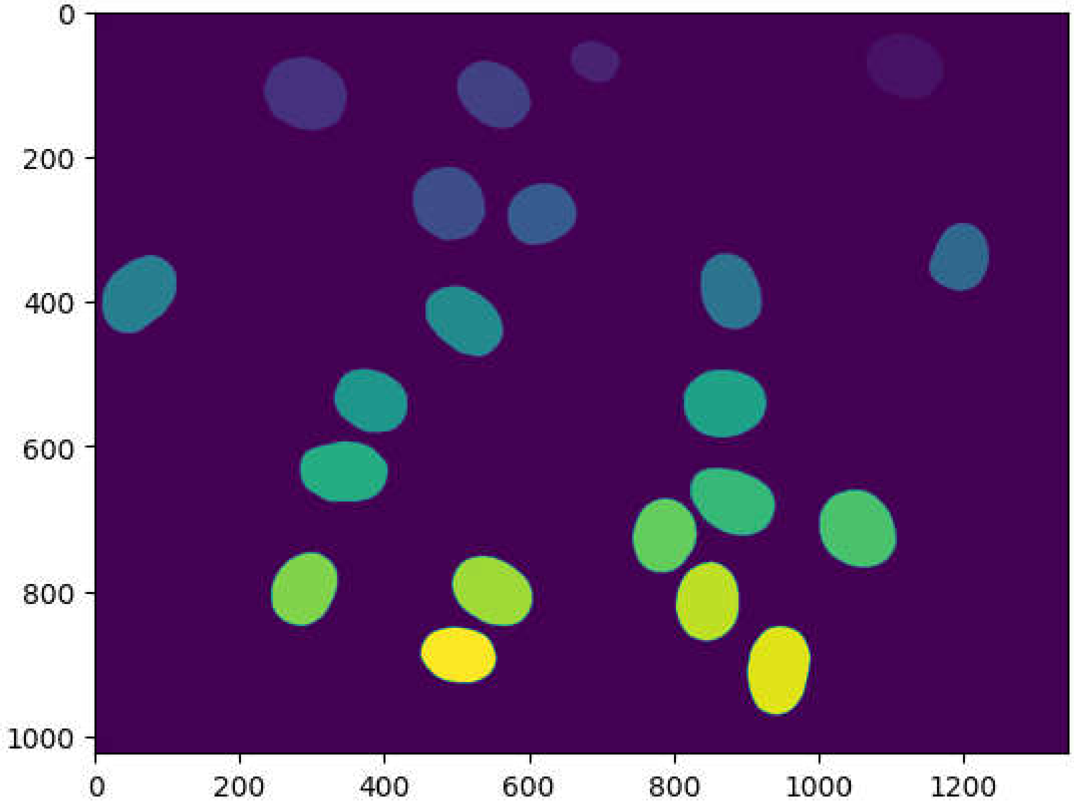
```
relabeled, n_left = mh.labeled.relabel(labeled)
print('After filtering and relabeling, there are {} nuclei left.'.format(n_left))
```

```
After filtering and relabeling, there are 21 nuclei left.
```

In [13]:

```
relabeled, n_left = mh.labeled.relabel(labeled)
print('After filtering and relabeling, there are {} nuclei left.'.format(n_left))
imshow(relabeled)
show()
```

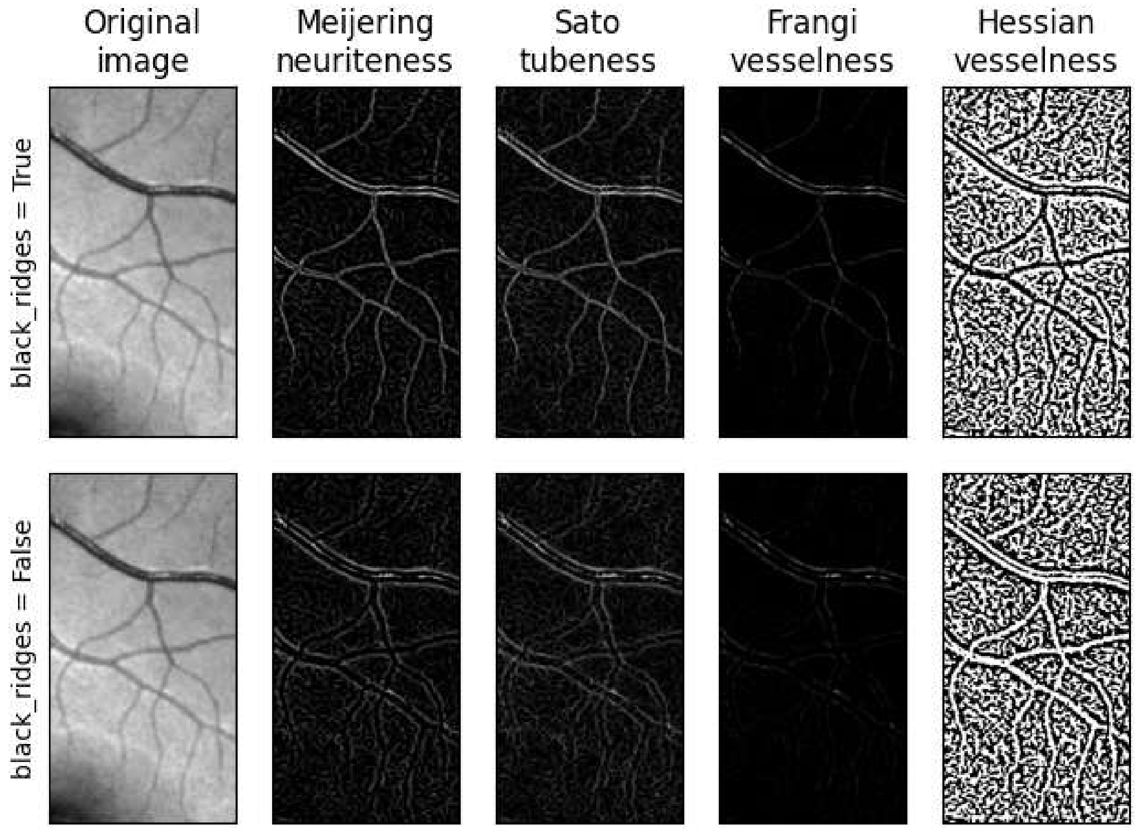
```
After filtering and relabeling, there are 21 nuclei left.
```



2. Filter ridge

In [19]:

```
from skimage import data
from skimage import color
from skimage.filters import meijering, sato, frangi, hessian
import matplotlib.pyplot as plt

def identity(image, **kwargs):
    """Return the original image, ignoring any kwargs."""
    return image

image = color.rgb2gray(data.retina())[300:700, 700:900]
cmap = plt.cm.gray
kwargs = {'sigmas': [1], 'mode': 'reflect'}

fig, axes = plt.subplots(2, 5)
for i, black_ridges in enumerate([1, 0]):
    for j, func in enumerate([identity, meijering, sato, frangi, hessian]):
        kwargs['black_ridges'] = black_ridges
        result = func(image, **kwargs)
        axes[i, j].imshow(result, cmap=cmap, aspect='auto')
        if i == 0:
            axes[i, j].set_title(['Original\nimage', 'Meijering\nneuriteness', 'Sato\ntub
        if j == 0:
            axes[i, j].set_ylabel('black_ridges = ' + str(bool(black_ridges)))
        axes[i, j].set_xticks([])
        axes[i, j].set_yticks([])

plt.tight_layout()
plt.show()
```



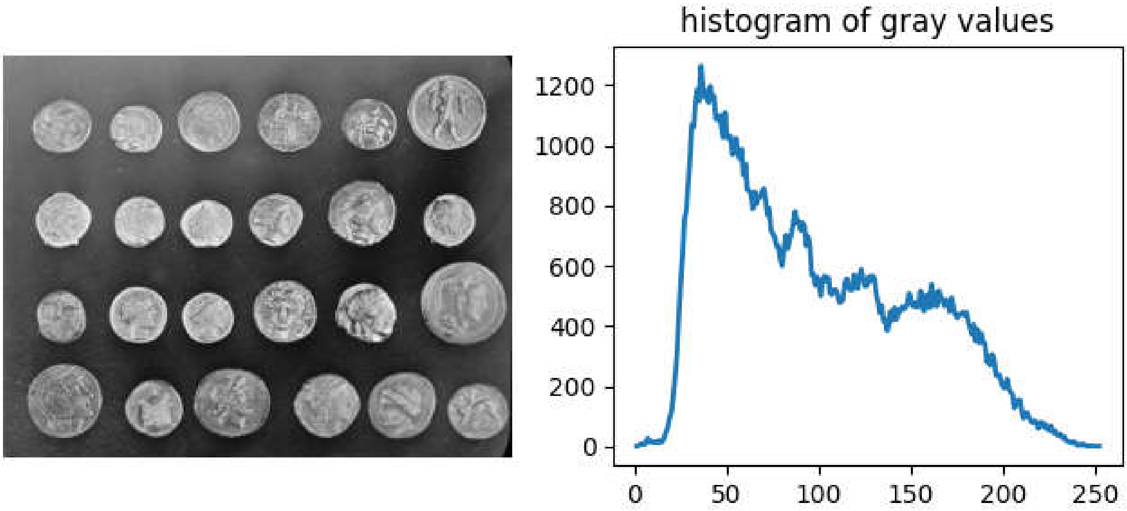
3. Segmentasi dan pelabelan

In [20]:

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
from skimage.exposure import histogram

coins = data.coins()
hist, hist_centers = histogram(coins)
fig, axes = plt.subplots(1, 2, figsize=(8, 3))
axes[0].imshow(coins, cmap=plt.cm.gray)
axes[0].axis('off')
axes[1].plot(hist_centers, hist, lw=2)
axes[1].set_title('histogram of gray values')
```

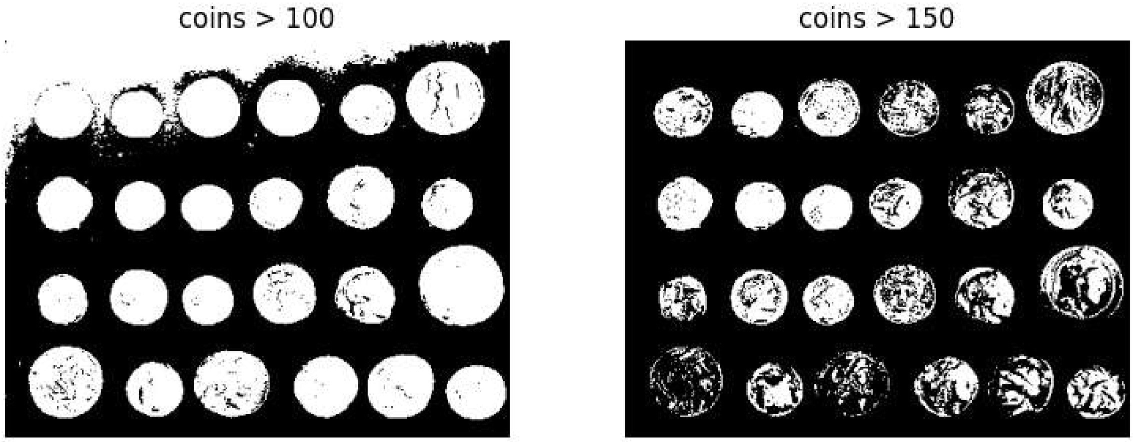
Out[20]:

```
Text(0.5, 1.0, 'histogram of gray values')
```



In [22]:

```
fig, axes = plt.subplots(1, 2, figsize=(8, 3),
sharey=True)
axes[0].imshow(coins > 100, cmap=plt.cm.gray)

axes[0].set_title('coins > 100')
axes[1].imshow(coins > 150, cmap=plt.cm.gray)
axes[1].set_title('coins > 150')
for a in axes:
 a.axis('off')
plt.tight_layout()
```

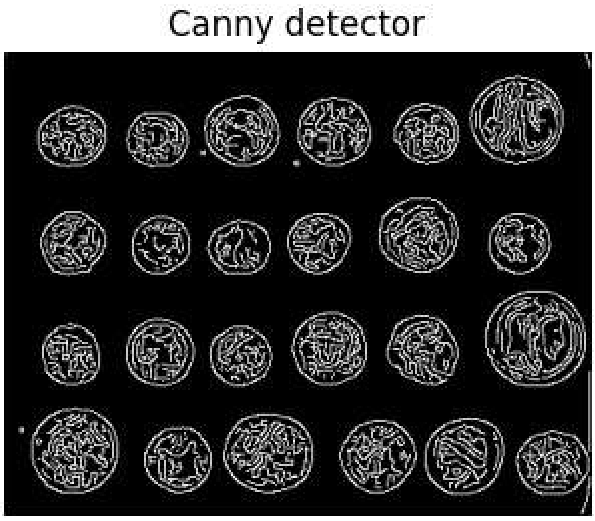


In [23]:

```
from skimage.feature import canny
edges = canny(coins)
fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(edges, cmap=plt.cm.gray)
ax.set_title('Canny detector')
ax.axis('off')
```

Out[23]:

```
(-0.5, 383.5, 302.5, -0.5)
```

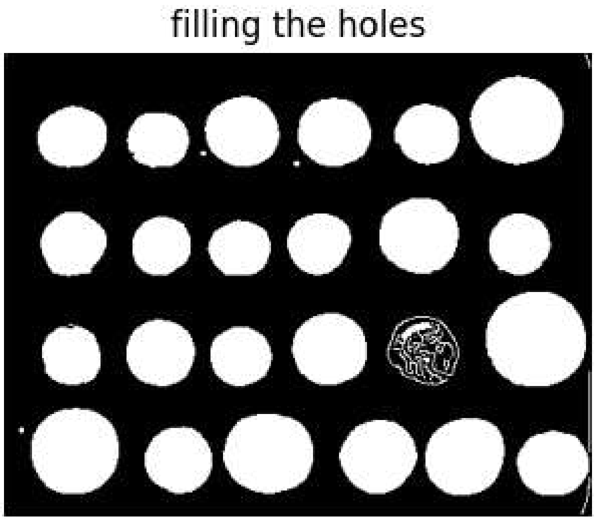


In [24]:

```
from scipy import ndimage as ndi

fill_coins = ndi.binary_fill_holes(edges)
fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(fill_coins, cmap=plt.cm.gray)
ax.set_title('filling the holes')
ax.axis('off')
```

Out[24]:

```
(-0.5, 383.5, 302.5, -0.5)
```

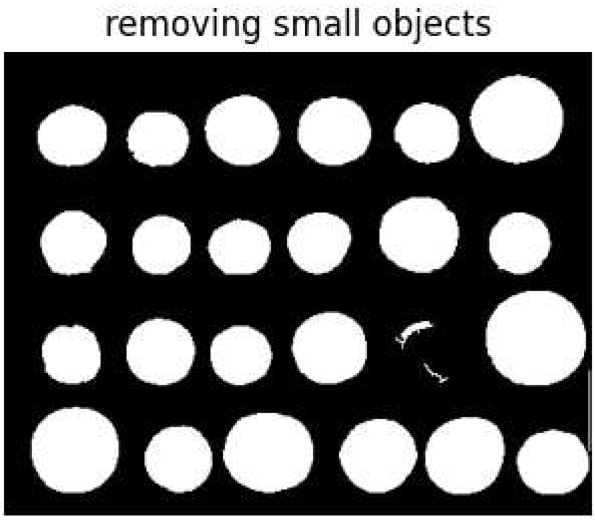


In [26]:

```
from skimage import morphology
coins_cleaned = morphology.remove_small_objects(fill_coins, 21)

fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(coins_cleaned, cmap=plt.cm.gray)
ax.set_title('removing small objects')
ax.axis('off')
```

Out[26]:

```
(-0.5, 383.5, 302.5, -0.5)
```

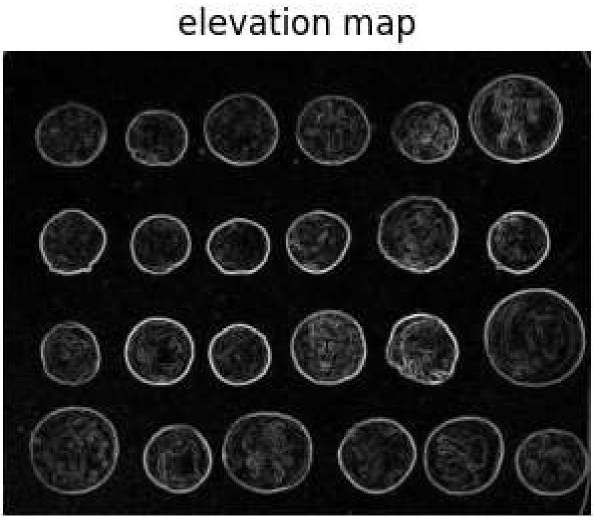


In [27]:

```
from skimage.filters import sobel

elevation_map = sobel(coins)
fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(elevation_map, cmap=plt.cm.gray)
ax.set_title('elevation map')
ax.axis('off')
```

Out[27]:

```
(-0.5, 383.5, 302.5, -0.5)
```

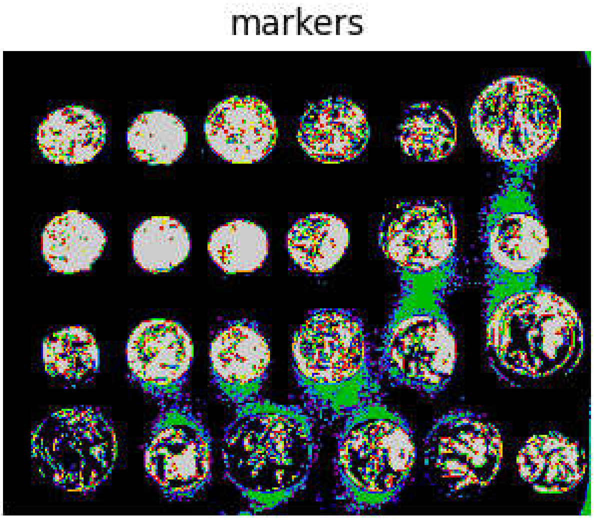


In [28]:

```
markers = np.zeros_like(coins)
markers[coins < 30] = 1
markers[coins > 150] = 2
fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(markers, cmap=plt.cm.nipy_spectral)
ax.set_title('markers')
ax.axis('off')
```

Out[28]:

```
(-0.5, 383.5, 302.5, -0.5)
```

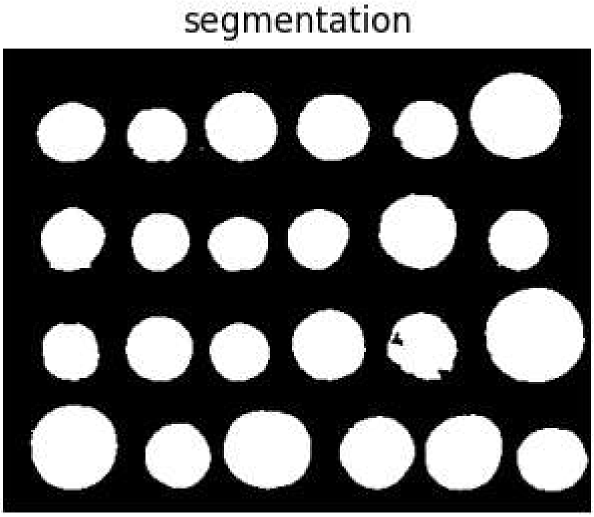


In [29]:

```
from skimage import segmentation
segmentation_coins = segmentation.watershed(elevation_map, markers)
fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(segmentation_coins, cmap=plt.cm.gray)
ax.set_title('segmentation')
ax.axis('off')
```

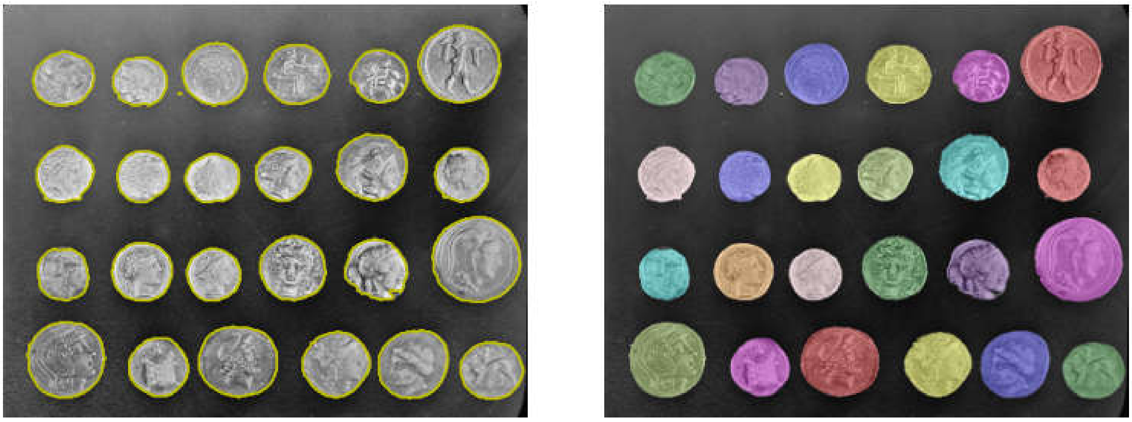
Out[29]:

```
(-0.5, 383.5, 302.5, -0.5)
```



In [30]:

```
from skimage.color import label2rgb

segmentation_coins = ndi.binary_fill_holes(segmentation_coins - 1)
labeled_coins, _ = ndi.label(segmentation_coins)
image_label_overlay = label2rgb(labeled_coins, image=coins, bg_label=0)
fig, axes = plt.subplots(1, 2, figsize=(8, 3), sharey=True)
axes[0].imshow(coins, cmap=plt.cm.gray)
axes[0].contour(segmentation_coins, [0.5], linewidths=1.2, colors='y')
axes[1].imshow(image_label_overlay)
for a in axes:
    a.axis('off')

plt.tight_layout()
plt.show()
```

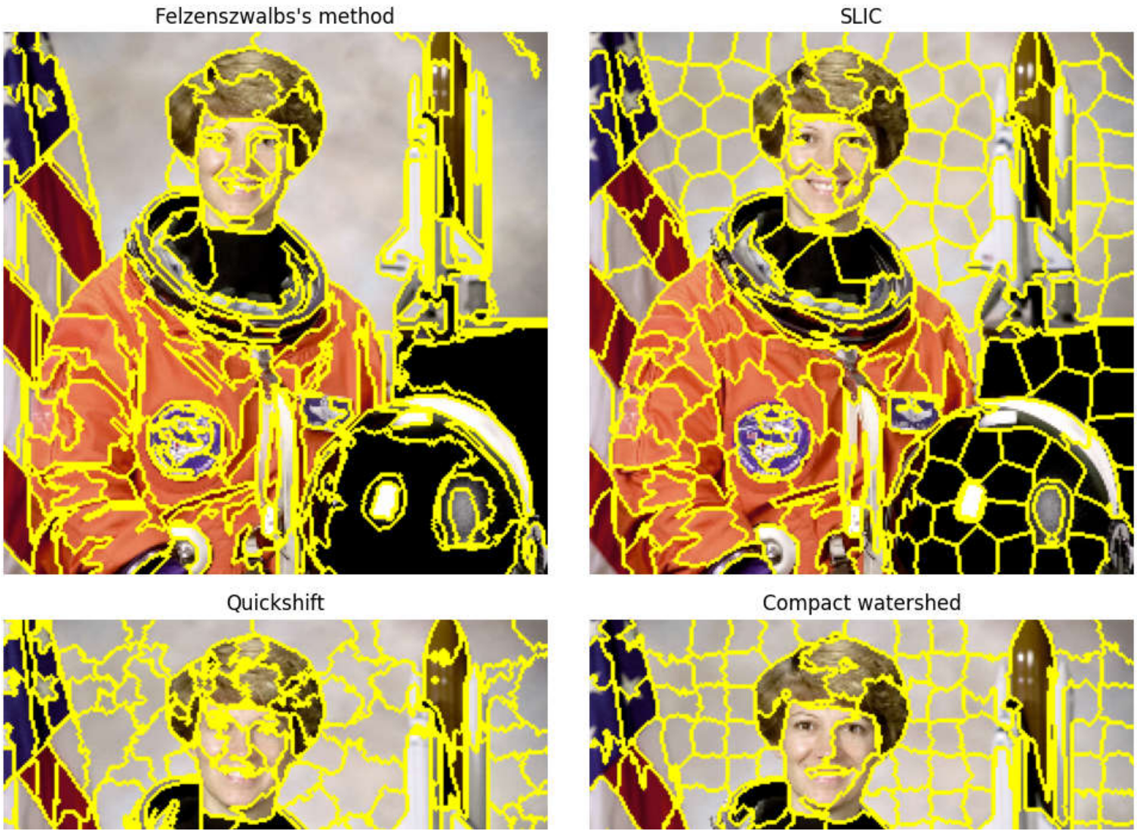


4. Algoritma segmentasi vs superpiksel

In [31]:

```
import matplotlib.pyplot as plt
import numpy as np
from skimage.data import astronaut
from skimage.color import rgb2gray
from skimage.filters import sobel
from skimage.segmentation import felzenszwalb, slic, quickshift, watershed
from skimage.segmentation import mark_boundaries
from skimage.util import img_as_float

img = img_as_float(astronaut()[::2, ::2])
segments_fz = felzenszwalb(img, scale=100, sigma=0.5, min_size=50)
segments_slic = slic(img, n_segments=250, compactness=10, sigma=1, start_label=1)
segments_quick = quickshift(img, kernel_size=3, max_dist=6, ratio=0.5)

gradient = sobel(rgb2gray(img))
segments_watershed = watershed(gradient, markers=250, compactness=0.001)

print(f"Felzenszwalb number of segments: {len(np.unique(segments_fz))}")
print(f"SLIC number of segments: {len(np.unique(segments_slic))}")
print(f"Quickshift number of segments: {len(np.unique(segments_quick))}")

fig, ax = plt.subplots(2, 2, figsize=(10, 10), sharex=True, sharey=True)
ax[0, 0].imshow(mark_boundaries(img, segments_fz))
ax[0, 0].set_title("Felzenszwalbs's method")
ax[0, 1].imshow(mark_boundaries(img, segments_slic))
ax[0, 1].set_title('SLIC')
ax[1, 0].imshow(mark_boundaries(img, segments_quick))
ax[1, 0].set_title('Quickshift')
ax[1, 1].imshow(mark_boundaries(img, segments_watershed))
ax[1, 1].set_title('Compact watershed')
for a in ax.ravel():
    a.set_axis_off()

plt.tight_layout()
plt.show()
```

```
Felzenszwalb number of segments: 194
SLIC number of segments: 196
Quickshift number of segments: 695
```



Felzenszwalbs's method

SLIC

Quickshift

Compact watershed