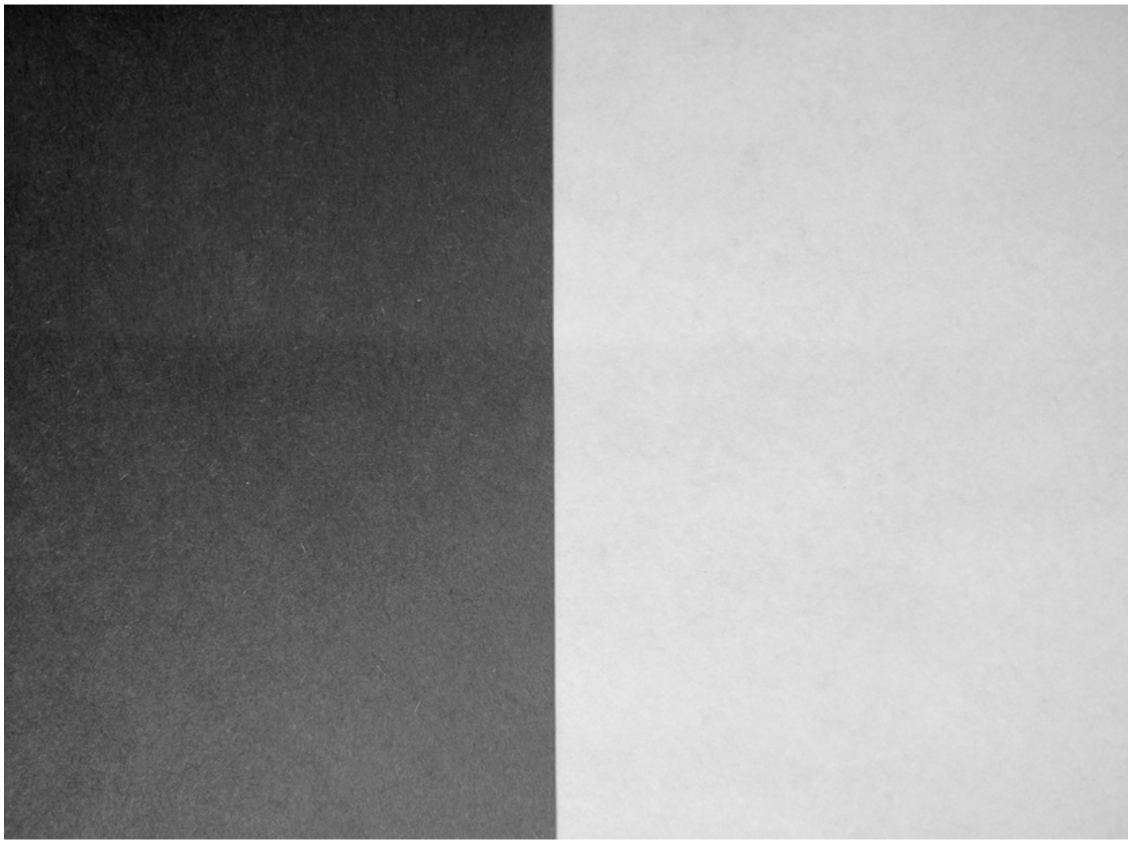
08-edge-detection

January 14, 2020

1 Edge Detection



1.1 Line Profile

```
[ ]: import skimage.io
     from matplotlib import pyplot as plt

     image = skimage.io.imread("../data/07-bw.jpg")
     plt.plot(image[200, :])
     plt.plot(image[400, :])
     plt.plot(image[600, :])
```

[ ]:
```
# zoom in on the image
skimage.io.imshow(image[200:250, 380:430])
```

[ ]:
```
# zoomed in, where would you place the edge?!
plt.plot(range(380, 430), image[200, 380:430])
```

1.2 Canny Edge Detection

[ ]:
```
# created by John Canny in 1986
# fancy algorithm composed of multiple steps
# algorithm steps:
    # gaussian blur, remove noise; !sigma
    # sobel edge detection: derivative of curve fitted to pixels
    # non-maximum suppression
    # double threshold
    # hysteresis
# important part: parameters -> sigma, low, high threshold
# read image as grayscale:
import skimage.feature
import skimage.viewer
import skimage

image = skimage.io.imread("../data/07-shapes.tif", as_gray=True)
sigma = 2
low_threshold = 0.1
high_threshold = 0.3
edges = skimage.feature.canny(
    image=image,
    sigma=sigma,
    low_threshold=low_threshold,
    high_threshold=high_threshold,
)
plt.imshow(edges)
# to get rid of artifacts
# plt.imshow(edges, interpolation='bicubic')
# plt.rc("image", interpolation='bilinear')
```

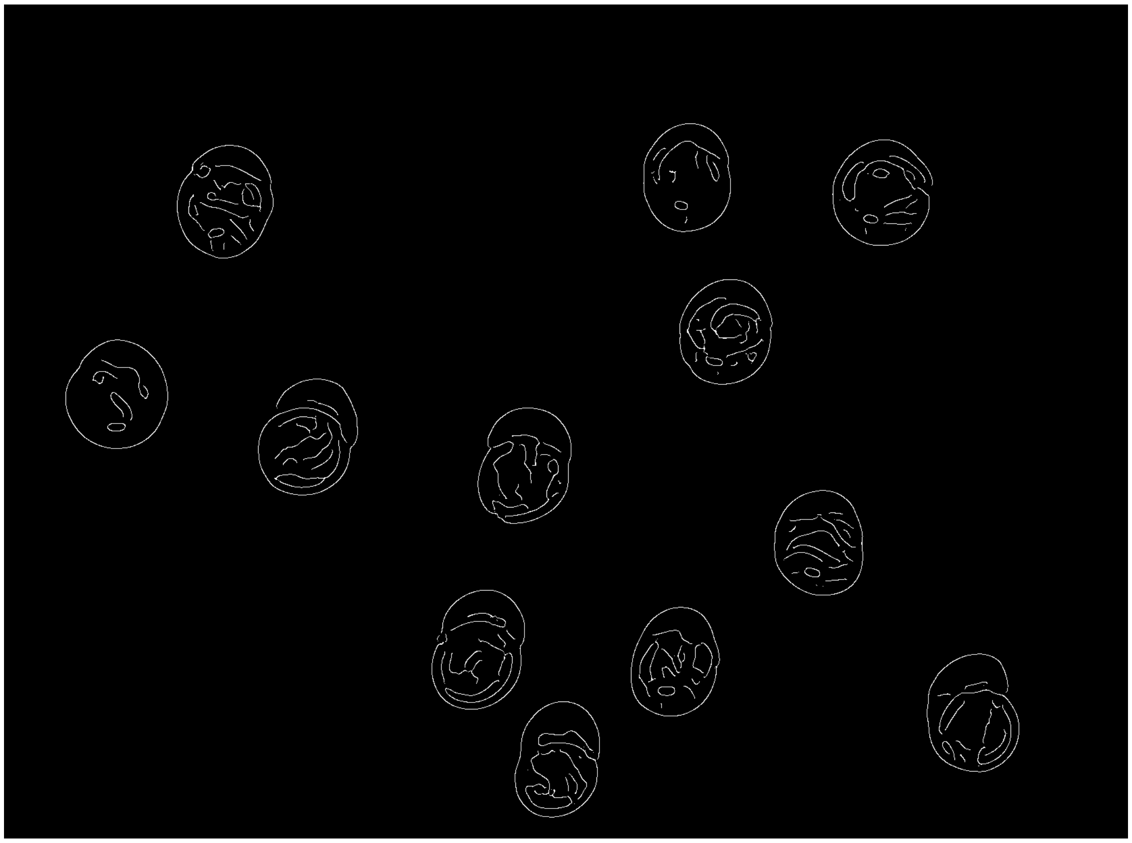
[ ]:
```

```

1.3 Viewer plugins: Interact with the image viewer

[ ]:
```python
# parameters are interdependent, hard to "guess"
image = skimage.io.imread("../data/07-beads.jpg", as_gray=True)

viewer = skimage.viewer.ImageViewer(image)

canny_plugin = skimage.viewer.plugins.Plugin(image_filter=skimage.feature.canny)
canny_plugin.name = "Canny Filter Plugin"

# Add sliders for the parameters
# The filter function will be called with the slider parameters according to␣
 ↪their names as keyword arguments. So it is very important to name the sliders␣
 ↪appropriately.
canny_plugin += skimage.viewer.widgets.Slider(
    name="sigma", low=0.0, high=7.0, value=2.0
)
canny_plugin += skimage.viewer.widgets.Slider(
    name="low_threshold", low=0.0, high=1.0, value=0.1
)
canny_plugin += skimage.viewer.widgets.Slider(
    name="high_threshold", low=0.0, high=1.0, value=0.2
)
viewer += canny_plugin
viewer.show()
```

1.3.1 Exercise:

Load the `beads.jpg` image from the data folder and use the interactive viewer to find the values that produce the following result after filtering:



[ ]:
```
# img = skimage.io.imread("../fig/07-beads.jpg", as_gray=True)
# edge = skimage.feature.canny(img, sigma=5, low_threshold=0.01,␣
↪high_threshold=0.04)
# edge = skimage.img_as_ubyte(edge)
# skimage.io.imsave("../fig/07-beads-out.png", arr=edge)
```

1.3.2 Exercise: Using sliders for thresholding

Let's apply what we know about creating sliders to another, similar situation: Consider this image (`..data/maize-roots.tif`) of a collection of maize seedlings, and suppose we wish to use simple fixed-level thresholding to **mask out** everything that is not part of one of the plants.

[ ]:
```
image = skimage.io.imread("../data/maize-roots.tif", as_gray=True)

def filter_function(image, sigma, threshold):
    masked = image.copy()
    masked[skimage.filters.gaussian(image, sigma=sigma) <= threshold] = 0
    return masked

smooth_threshold_plugin = skimage.viewer.plugins.Plugin(
    image_filter=filter_function
)
```

```
smooth_threshold_plugin.name = "Smooth and Threshold Plugin"

smooth_threshold_plugin += skimage.viewer.widgets.Slider(
    "sigma", low=0.0, high=7.0, value=1.0
)
smooth_threshold_plugin += skimage.viewer.widgets.Slider(
    "threshold", low=0.0, high=1.0, value=0.5
)


viewer = skimage.viewer.ImageViewer(image=image)
viewer += smooth_threshold_plugin
viewer.show()
```

1.3.3 Keypoints

- The `skimage.viewer.ImageViewer` is extended using a `skimage.viewer.plugins.Plugin`.
- We supply a filter function callback when creating a Plugin.
- Parameters of the callback function are manipulated interactively by creating sliders with the `skimage.viewer.widgets.slider()` function and adding them to the plugin.