MUEI/MUNR
ETSEIB

Course on Medical Imaging
2023-2024 Q1

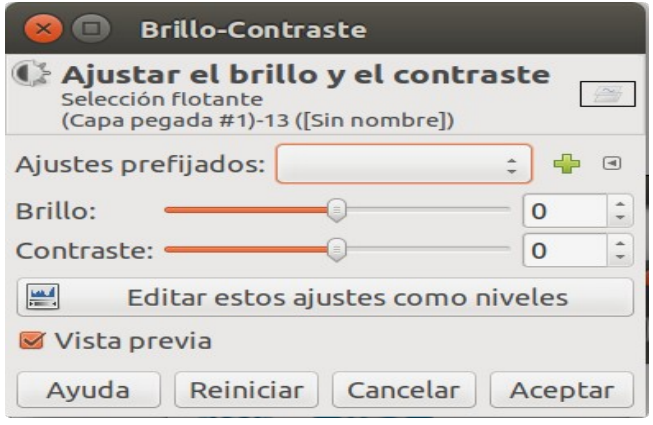
Session 4
Processing images

Dani Tost

Contrast enhancement

Image contrast

Contrast is the difference in luminance that makes an image distinguishable. It can be computed as **Imax - Imin**. If **Imin = Imax** all pixels have the same color: no features are distinguishable.
With low contrast, images seem foggy, and it is difficult to identify the details in them.
Low contrast images have a narrow range of values.
Look at the two histograms below. Which image has the lower contrast?



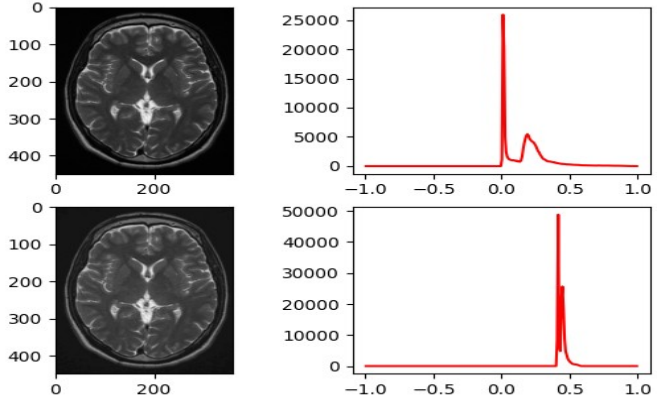
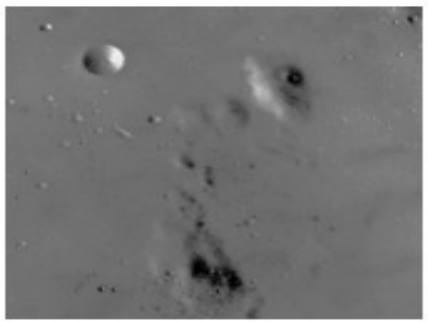
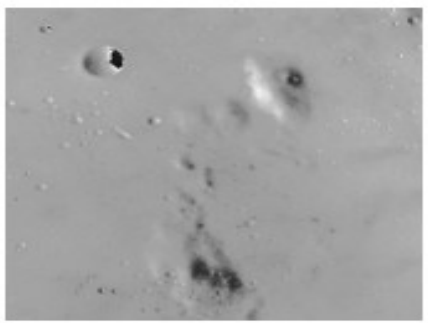


Image brightness

The mean intensity of an image, the higher, the brighter is the image.

original values





+50

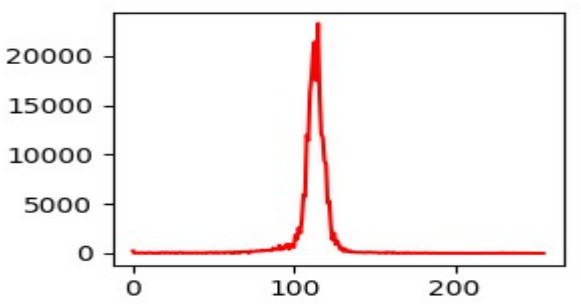
Observe the effects of values larger than

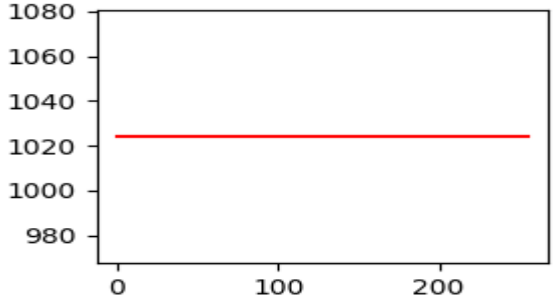
```
>>> from skimage import data
>>> im = data.astronaut()
>>> im2 = im + 50
>>> import matplotlib.pyplot as plt
>>> fig, sp = plt.subplots(1, 2)
>>> sp[0].imshow(im)
>>> sp[1].imshow(im2)
>>> plt.show()
```

Histogram equalization

To adjust image intensities to enhance the contrast of an image by flattening the histogram of the image.

Goal: Ideally to get a flat histogram





$$\frac{n}{nlevels}$$

In a gray uint8 image nlevels=256

Histogram equalization

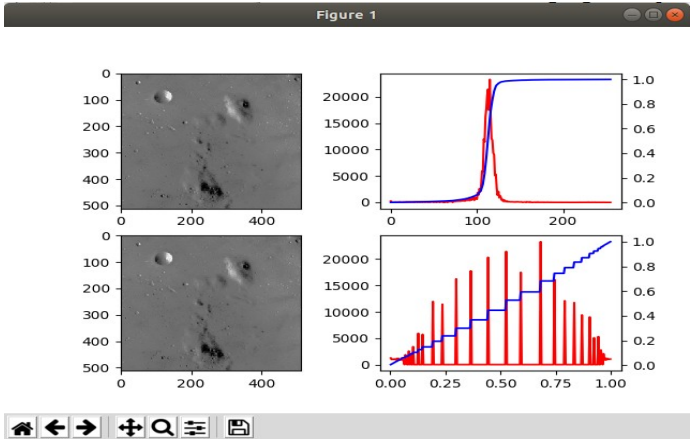
In skimage.exposure: use function equalize_hist

Method for a uint8 gray image:

Compute the cumulative distribution function (see previous slides)

Compute new image substituting all the intensity values by:

$$newvalue(i) = round\left(\frac{cdf(i) - min(cdf)}{n - min(cdf)} \cdot 255\right)$$



n = number of pixels of an image
i = an intensity level of the image 0<=i<=255
n_i = number of pixels with intensity i in the image

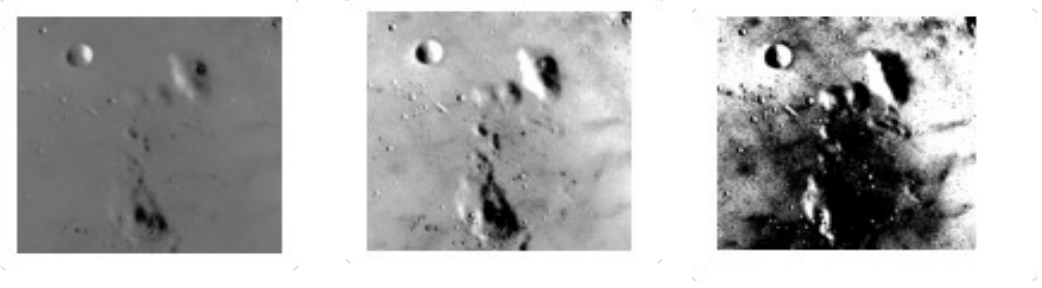
The histogram is flattened (not completely flat) and the cdf approximates a straight line

More info at:
https://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_equalize.html

Contrast stretching

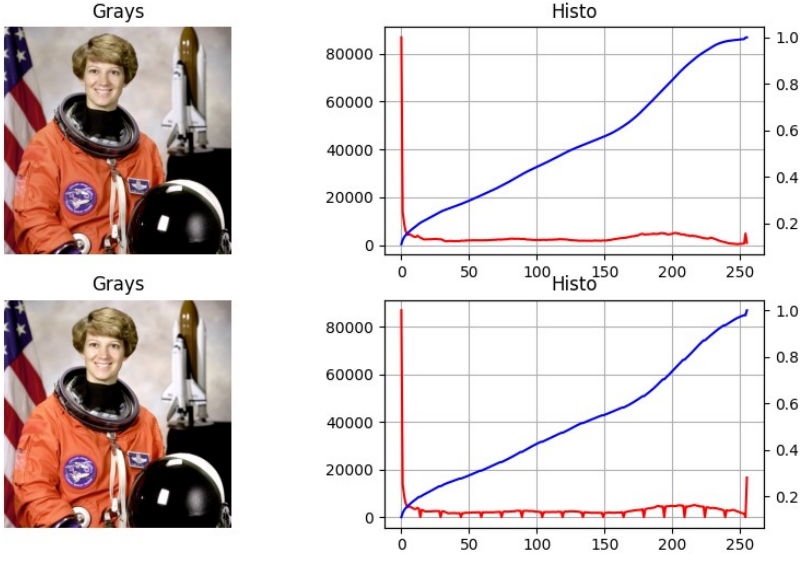
Contrast can be stretched by rescaling all intensities falling within a given percentile

```
>>> from skimage import data, exposure
>>> img = data.moon()
>>> p2, p98 = np.percentile(img, (2, 98))
>>> p2, p98
78.0, 129.0
>>> img2 = exposure.rescale_intensity(img, in_range=(p2, p98))
>>> p20, p80 = np.percentile(img, (20, 80))
>>> p20, p80
109.0, 118.0
>>> img3 = exposure.rescale_intensity(img, in_range=(p20, p80))
```



Histogram equalization

Observe in this image how the cdf is flattened. Where do you perceive more contrast? Can you find the effect in the image? The equalization here is based on the gray level, but it could have been done per channel.

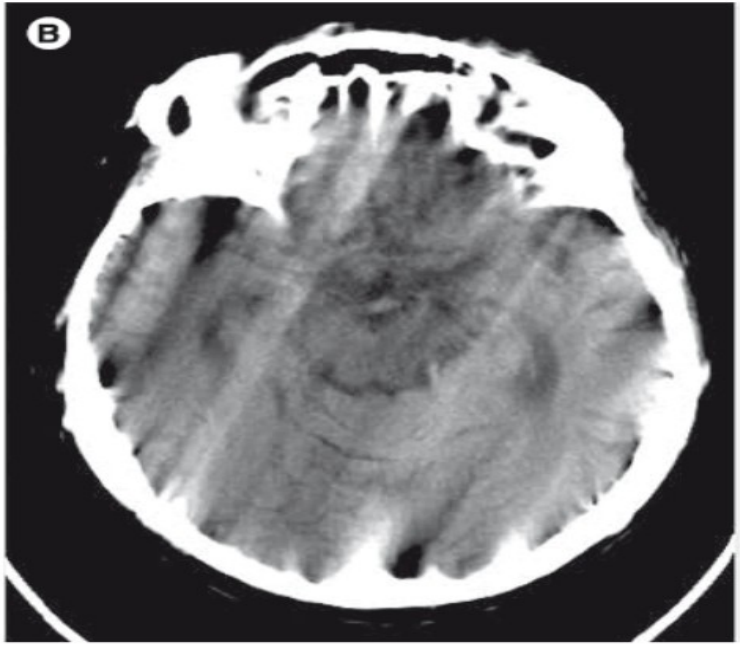


Noise removal

Noise and artifacts

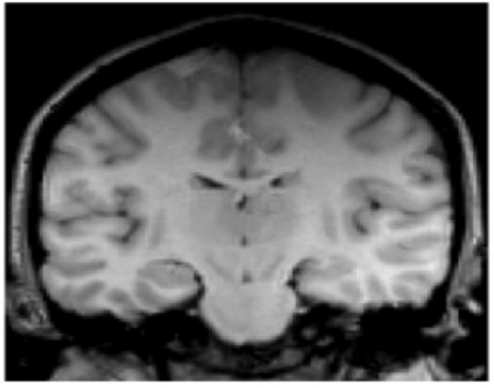
Noise: random variation of brightness or color

Artifacts: noticeable distortions of the images

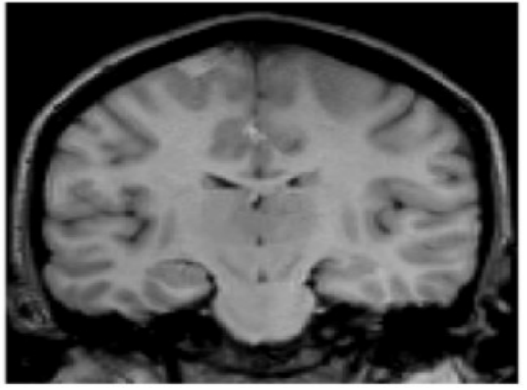


Noise and artifacts

Example of artifact: inhomogeneity of intensity



Before correction



After correction

Image from, «A Review of Methods for Correction of Intensity Inhomogeneity in MRI», Uros Vovk, Franjo Pernu and Bostjan Likar, IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 26, NO. 3, MARCH 200

Noise removal

Gaussian filters (but it blurs)

Follow the example:

https://scikit-image.org/docs/dev/api/skimage.filters.html#skimage.filters.gaussian

Bilateral, tv-chambolle, wavelet transforms (and many others)

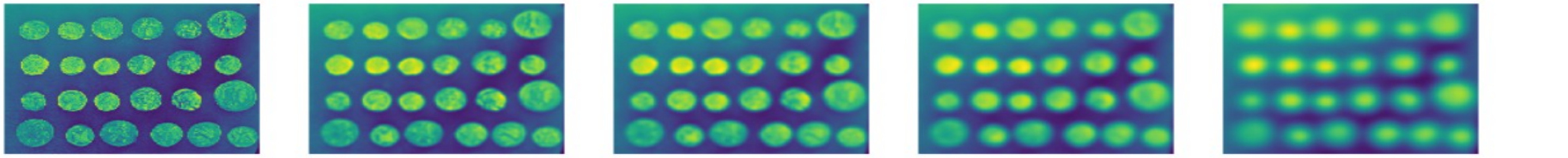
Follow the example (next slide adapted):

https://scikit-image.org/docs/dev/auto_examples/filters/plot_denoise.html

Gaussian filter

It consists of a 3x3 convolution with the matrix:

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

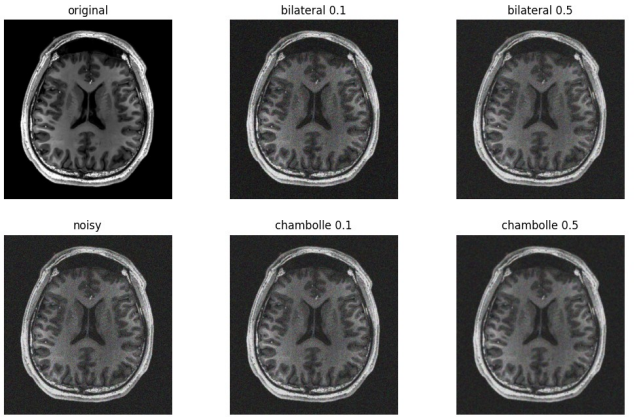



```
>>> img = []
>>> img.append(data.coins)
>>> img.append(filters.gaussian(img[0], sigma=2))
>>> img.append(filters.gaussian(img[0], sigma=3))
>>> img.append(filters.gaussian(img[0], sigma=5))
>>> img.append(filters.gaussian(img[0], sigma=10))
>>> fig, ax = plt.subplots(1, len(img))
>>> for i in range(len(img)):
…         ax[i].imshow(img[i])
…         ax[i].axis('off')
```

Can be applied with larger kernels

Tv Chambolle and bilateral

```
fig, ax = plt.subplots(2, 3)
img = io.imread('brain.jpg')
ax[0, 0].imshow(img)
ax[0, 0].set_title('original')
ims = img_as_float(img)
sigma = 0.355
noisy = util.random_noise(img, var=sigma**2)
ax[1, 0].imshow(noisy)
ax[1, 0].set_title('noisy')
bil1 = restoration.denoise_bilateral(noisy, sigma_color=0.05,  multichannel=True)
ax[0, 1].imshow(bil1)
ax[0, 1].set_title('bilateral 0.1')
bil2 = restoration.denoise_bilateral(noisy, sigma_color=1.0,  multichannel=True)
ax[0, 2].imshow(bil2)
ax[0, 2].set_title('bilateral 0.5')
tv1 = restoration.denoise_tv_chambolle(noisy, weight=0.1, multichannel=True)
ax[1, 1].imshow(tv1)
ax[1, 1].set_title('chambolle 0.1')
tv2 = restoration.denoise_tv_chambolle(noisy, weight=0.5, multichannel=True)
ax[1, 2].imshow(tv2)
ax[1, 2].set_title('chambolle 0.5')
for i in range(2):
    for j in range(3):
        ax[i, j].axis('off')
```

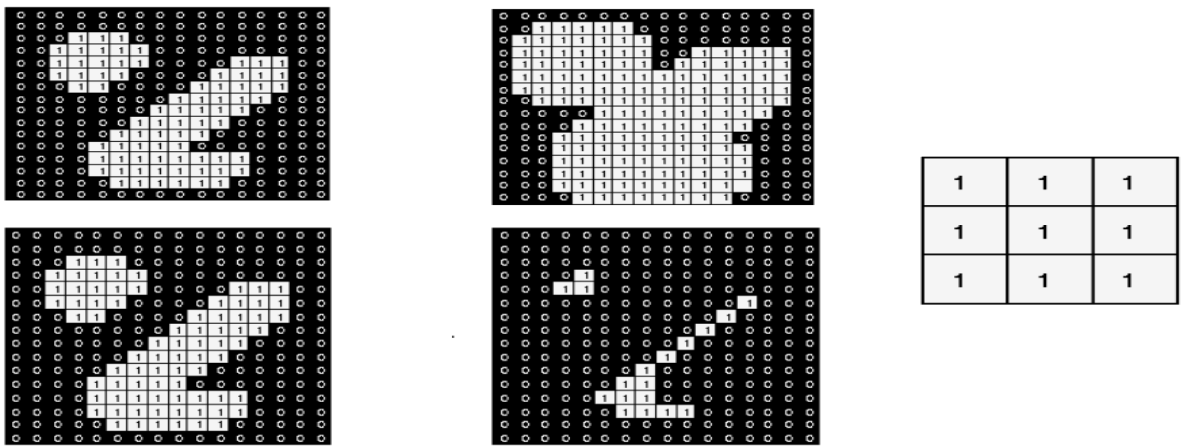


Mathematical morphology to smooth the surface of regions and remove small artifacts

Mathematical morphology

Erode and dilate segmented images to clean segmented images

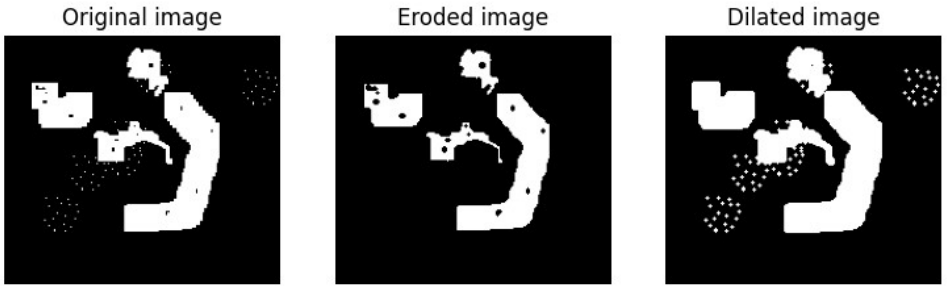
Add/remove in/from the segmented region the pixels using a structuring element



1	1	1
1	1	1
1	1	1

Mathematical morphology

```
img = io.imread('a.png')
img2 = img[:, :, 0]
    img3 = morphology.erosion(img2)
    img4 = morphology.dilation(img2)
```

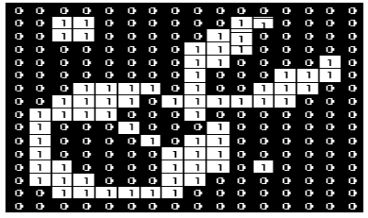
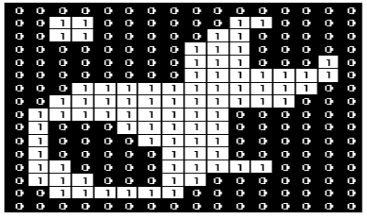
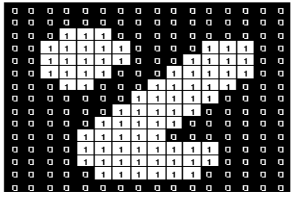
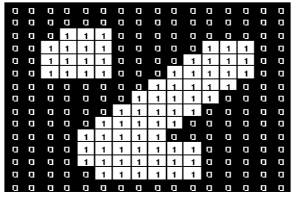


Also look at tutorial
https://scikit-image.org/docs/dev/auto_examples/filters/plot_inpaint.html#sphx-glr-auto-examples-filters-plot-inpaint-py

Mathematical morphology

Opening (and closing) are the result of applying an erosion and a dilation (or vice-versa) with the same structuring elements.

They are used to remove small isolated artifacts and to fill small holes

http://homepages.inf.ed.ac.uk/rbf/HIPR2/open.htm

http://homepages.inf.ed.ac.uk/rbf/HIPR2/close.htm

Next week edge detection, segmentation, contour extraction and rasterization

Now we'll see how to integrate these filters in an interface