Lab 04

Lab 04

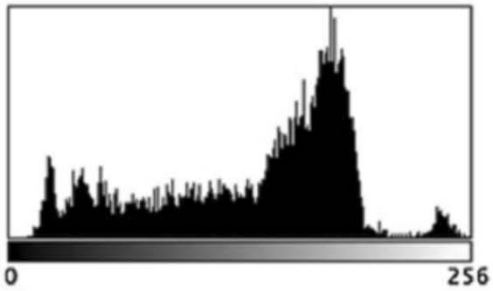
- Histogram Equalization
- Filter

Histogram

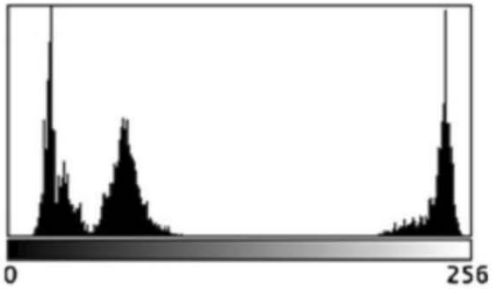
- Histogram of an image provides the frequency of the brightness (intensity) value in the image.

```
def histogram(im):
    h = np.zeros(255)
    for row in im.shape[0]:
        for col in im.shape[1]:
            val = im[row, col]
            h[val] += 1
```

Histogram



Count: 10192
Mean: 133.711
StdDev: 55.391
Min: 9
Max: 255
Mode: 178 (180)



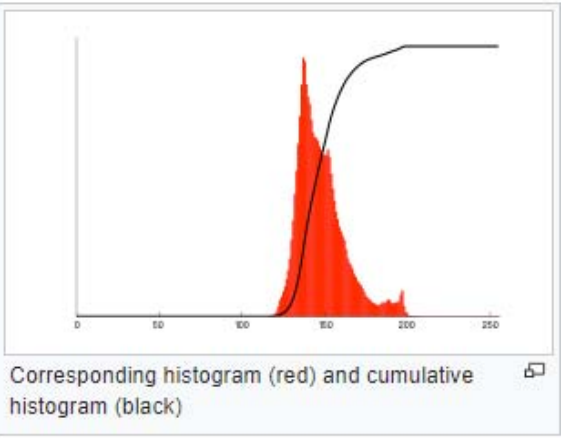
Count: 10192
Mean: 104.637
StdDev: 89.862
Min: 11
Max: 254
Mode: 23 (440)

Slide credit: Dr. Mubarak Shah

Histogram Equalization

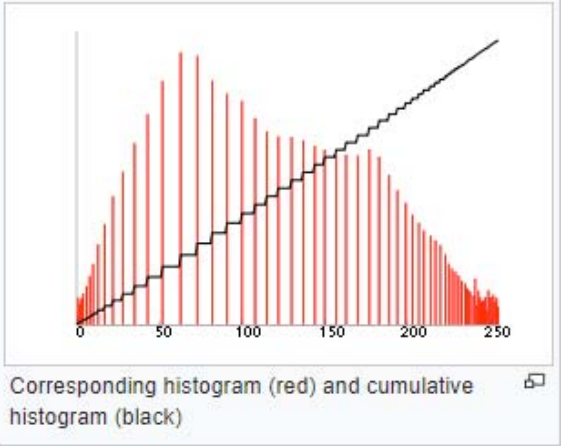
- Straighten cumulative histogram

Before Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

After Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

import

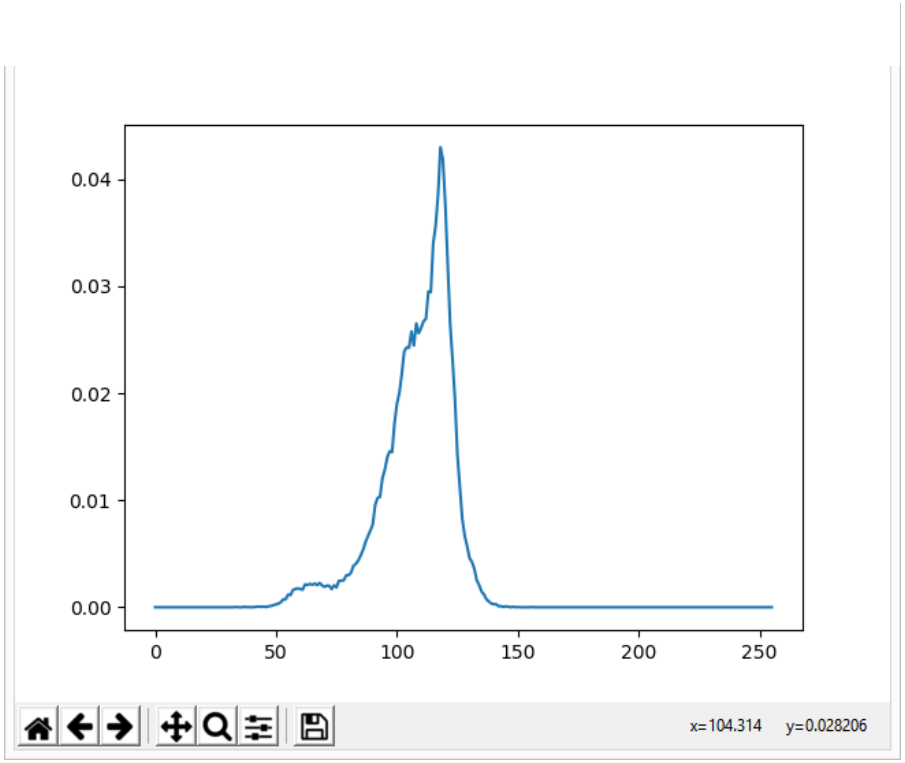
```
import numpy as np
import pylab as plt
from skimage import io
```

Compute and plot histogram

```
def imhist(im):
  # calculates normalized histogram of an image
  # you will see the reason for normalization later
        m, n = im.shape
        h = [0.0] * 256
        for i in range(m):
                for j in range(n):
                        h[im[i, j]]+=1
        return np.array(h)/(m*n)
```

Compute and plot histogram

```
img = np.uint8(io.imread('HB.jpg',as_grey=True)*255.0)
h = imhist(img)
# plot histograms and transfer function
plt.plot(h)
plt.show()
```

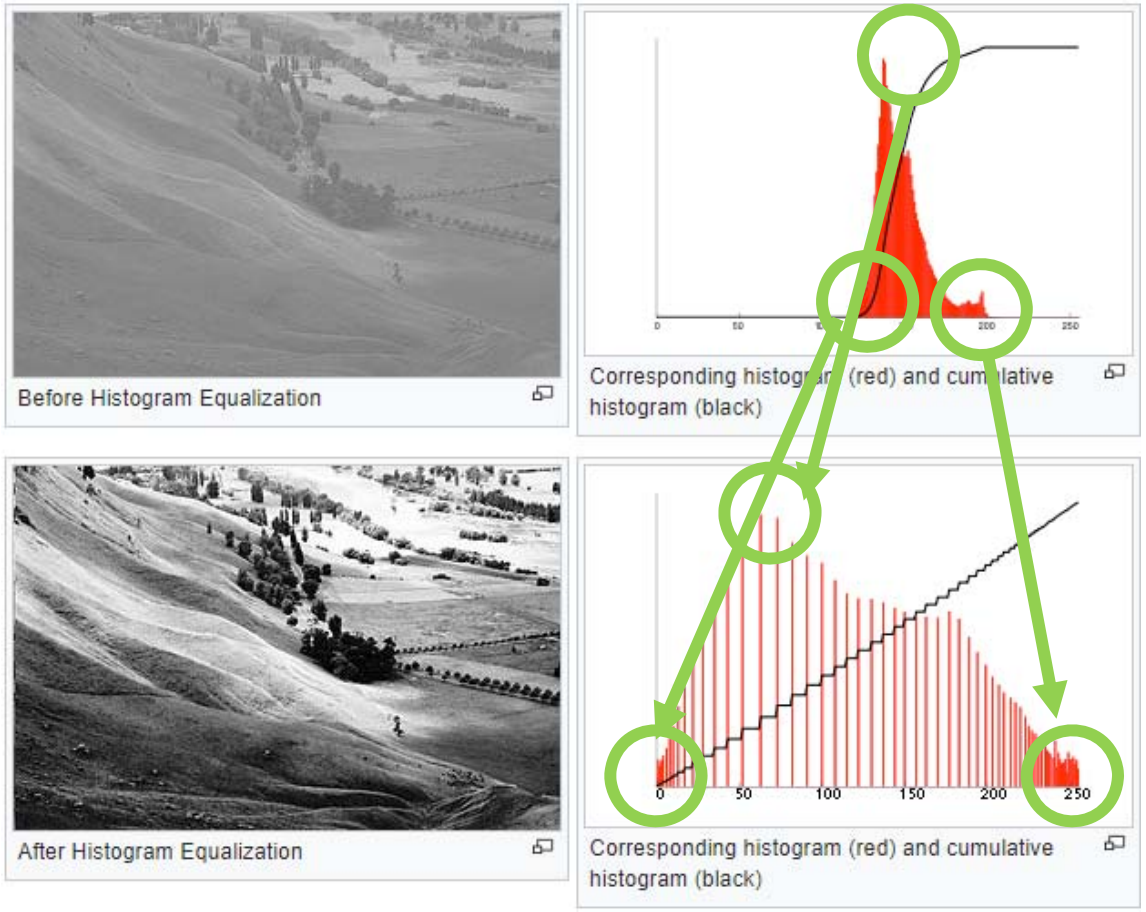


Compute cumulative histogram

```
def cumsum(h):
        # finds cumulative sum of a numpy array, list
        return [sum(h[:i+1]) for i in range(len(h))]


cdf = np.array(cumsum(h)) #cumulative distribution function
```

Transfer function

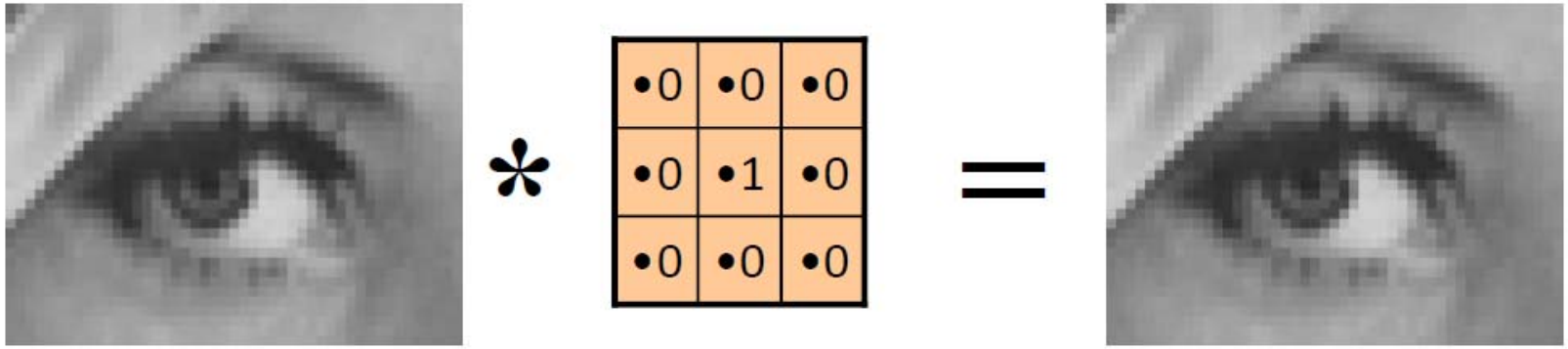
```
transfer = np.uint8(255 * cdf) #transfer function
```



Entire procedure for histeq

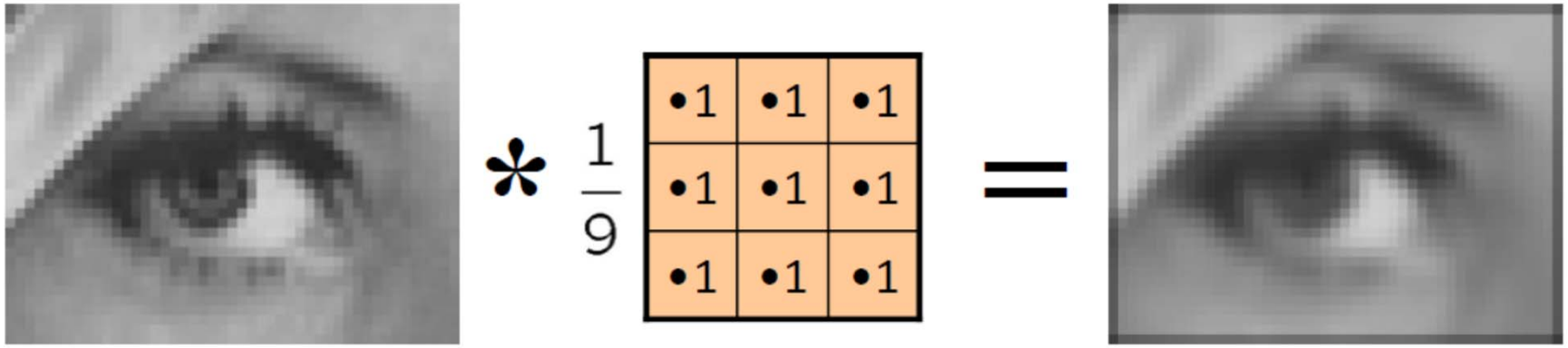
```
def histeq(im):
        # calculate Histogram
        h = imhist(im)

        # cumulative distribution function
        # np.array will enable multiplication below
        cdf = np.array(cumsum(h))

        #transfer function
        transfer = np.uint8(255 * cdf)

        s1, s2 = im.shape
        Y = np.zeros_like(im)
        # apply transfered values for each pixel
        for i in range(0, s1):
                for j in range(0, s2):
                        Y[i, j] = transfer[im[i, j]]

        # new histogram
        H = imhist(Y)

        #return transformed image, original and new histogram,
        # and transform function
        return Y , h, H, transfer
```

filter



Original

Filtered
(no change)



Original

Blur (with a
box filter)

sharpen

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(Note that filter sums to 1)

=

"details of the image"

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right)$$

Apply filter

```
def filter( image, kernel ):
    radius = kernel.shape[0]//2
    kh, kw = kernel.shape
    height, width = image.shape
    result = np.zeros( image.shape )
    # it should be range(radius, height-radius)
    # but omitted for simplicity
    # plz try it by yourself
    # indexing will be quite complicated
    for y in range(height-kh):
        for x in range(width-kw):
            for v in range(kh):
                for u in range(kw):
                    result[y,x] += image[y+v,x+u]*kernel[v,u]
    return result
```

```
# 0.8 is multiplied not to make the result become > 1
im_input = io.imread('_image1.jpg', as_grey=True)*0.8

filter_identity = np.zeros( (3,3) )
filter_identity[1,1] = 1

filter_box = np.ones( (3,3) ) / 9

im_id = filter( im_input, filter_identity )
im_box = filter( im_input, filter_box )
im_detail = im_id - im_box
im_shapen = filter( im_input, filter_identity*2-filter_box )

io.imsave('input.png',im_input)
io.imsave('id.png',im_id)
io.imsave('box.png',im_box)
io.imsave('detail.png',im_detail)

# check below two images are identical
io.imsave('id+detail.png', im_id+im_detail)
io.imsave('shapen.png', im_shapen)
```

homework

- TBA