- **Summary of Week 3 (Chapter 3)**

- Intensity Transformations by Point-Processing
- 1. s = T(r)
- 2. Image negative
- 3. Logarithmic transformation
- 4. Gamma-transformation
- 5. Contrast-stretching (s-curve)

- Bit-Plane Slicing
- 1. Intensities represented by bits

Some images in the slides are from Gonzales and Woods

Intensity transformations are applied on images for contrast manipulation or image thresholding. These are in the spatial domain, i.e. they are performed directly on the pixels of the image at hand, as opposed to being performed on the Fourier transform of the image.

The following are commonly used intensity transformations:

1. *Image Negatives (Linear)*
2. *Log Transformations*
3. *Power-Law (Gamma) Transformations*
4. *Piecewise-Linear Transformation Functions*

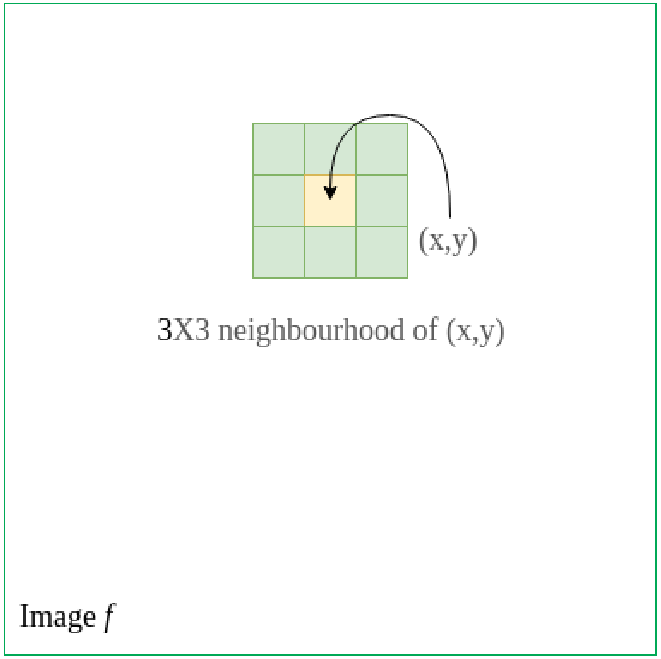
Python | Intensity Transformation Operations on Images - GeeksforGeeks

Spatial Domain Processes

Spatial domain processes can be described using the equation:

$$g(x, y) = T[f(x, y)],$$

where $f(x,y)$ is the input image, T is an operator on f defined over a neighbourhood of the point (x, y), and $g(x, y)$ is the output.



Python | Intensity Transformation Operations on Images - GeeksforGeeks

- Recall contrast-stretching goal

- Want images with high contrast (high dynamic range)
- Hence occupying the entire range of intensity levels
- Can be done by formulating streching curves/transformations

Question

- What does this mean in terms of pixel intensity distribution?
- Are there other ways to achieve contrast stretching?

Image Negatives

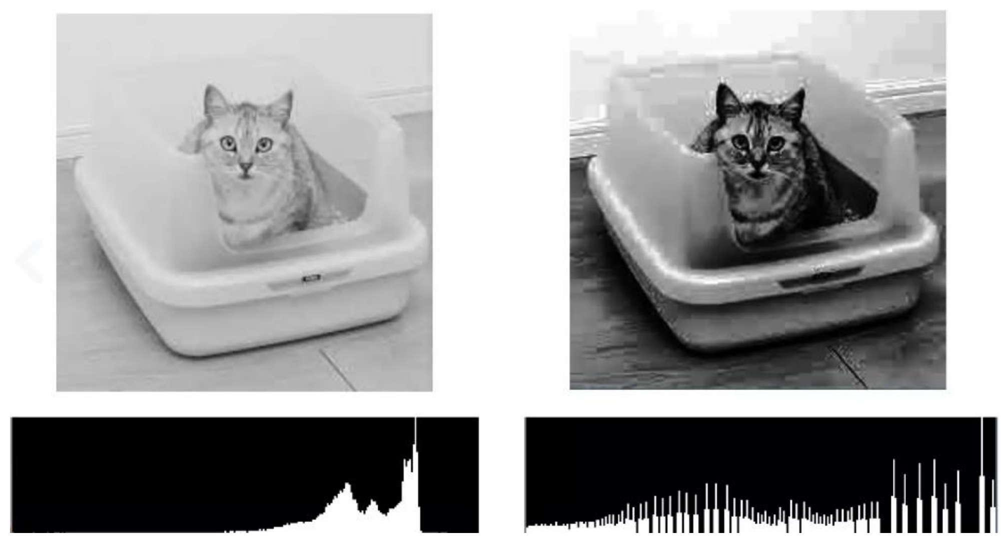
Mathematically, assume that an image goes from intensity levels 0 to (L - 1). Generally, L = 256. Then, the negative transformation can be described by the expression:

$$s = L - 1 - r,$$

where r is the initial intensity level and s is the final intensity level of a pixel. This produces a photographic negative.

- **Contrast Stretching by Histogram Equalization**

1. *The Histogram*
2. *Histogram Equalization*



[COM2304: Intensity Transformation and Spatial Filtering – I (Intensity Transformation and Image Noise) | PPT (slideshare.net)]

Histogram Equalization

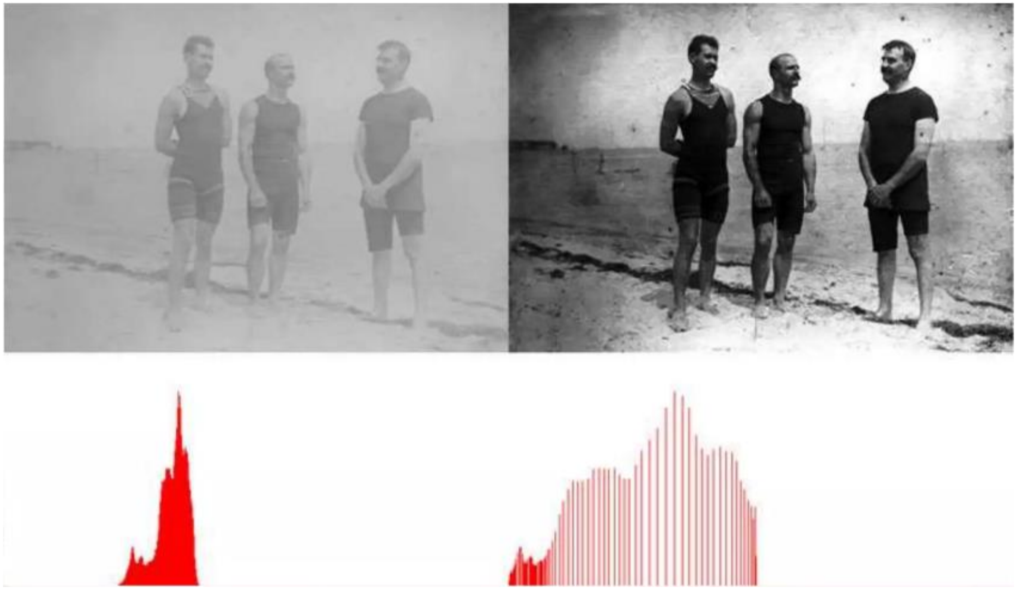
- It’s a method that improves the contrast in an image, in order to stretch out the intensity range.

- Equalization implies mapping one distribution (the given histogram) to another distribution (a wider and more uniform distribution of intensity values) so the intensity values are spread over the whole range.

[COM2304: Intensity Transformation and Spatial Filtering – I (Intensity Transformation and Image Noise) | PPT (slideshare.net)]

Difference between Contrast Stretching and Histogram Equalization

- Histogram Equalization



Contrast = (I_max - I_min)/(I_max + I_min)

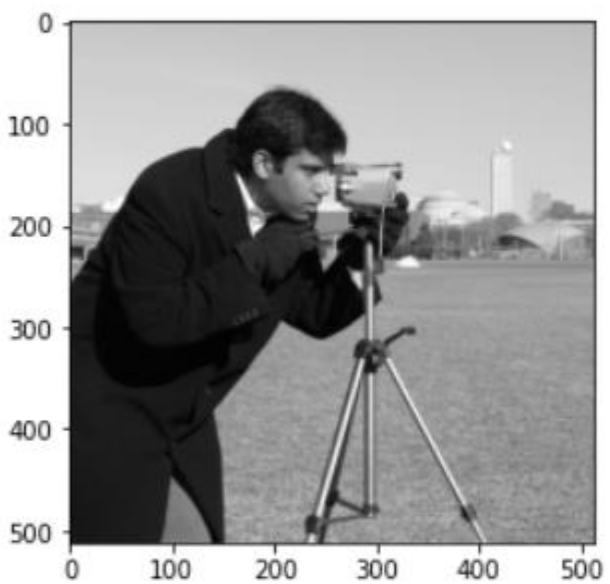
[Python | Intensity Transformation Operations on Images - GeeksforGeeks]
[COM2304: Intensity Transformation and Spatial Filtering – I (Intensity Transformation and Image Noise) | PPT (slideshare.net)]

• Histogram

```
import matplotlib.pyplot as plt
import numpy as np
from skimage import data, img_as_float
```

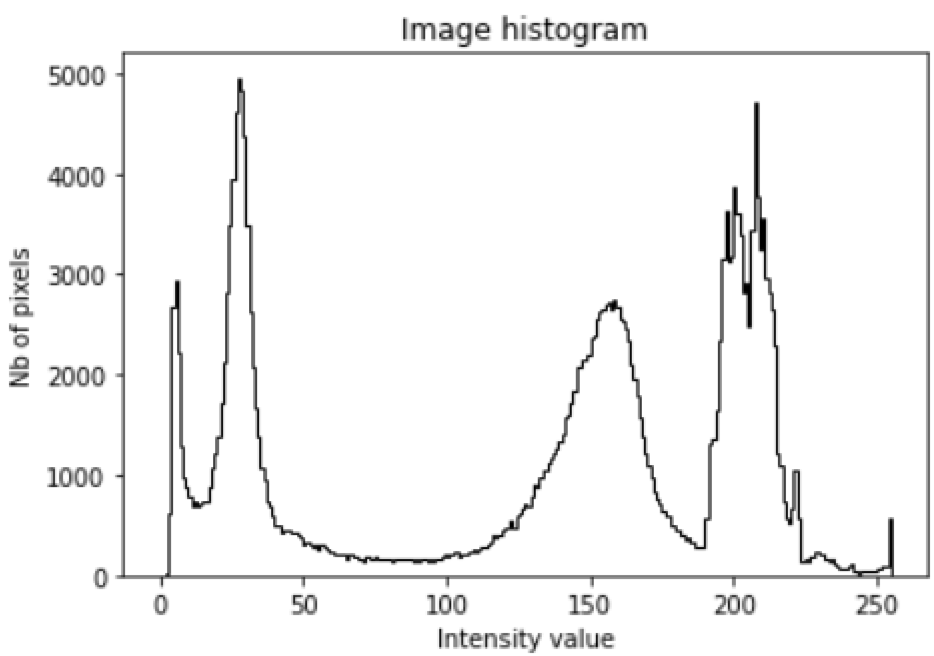
```
image = data.camera()
```

```
plt.imshow(image, cmap=plt.cm.gray)
plt.show()
image.dtype
```



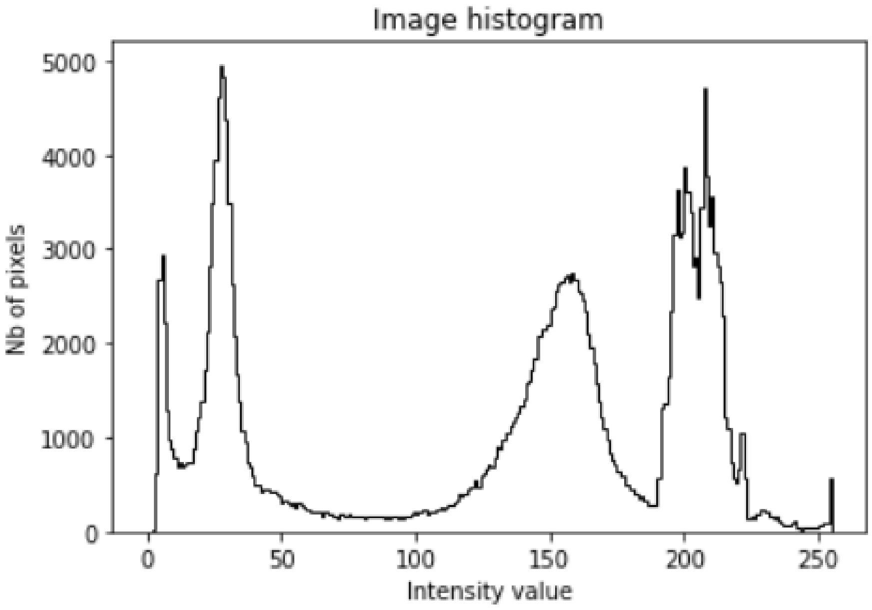

```
dtype('uint8')
```

```
plt.hist(image.ravel(), bins=255,\
        density=False, histtype='step', color='black')
plt.xlabel('Intensity value')
plt.ylabel('Nb of pixels')
plt.title('Image histogram')
plt.show()
```

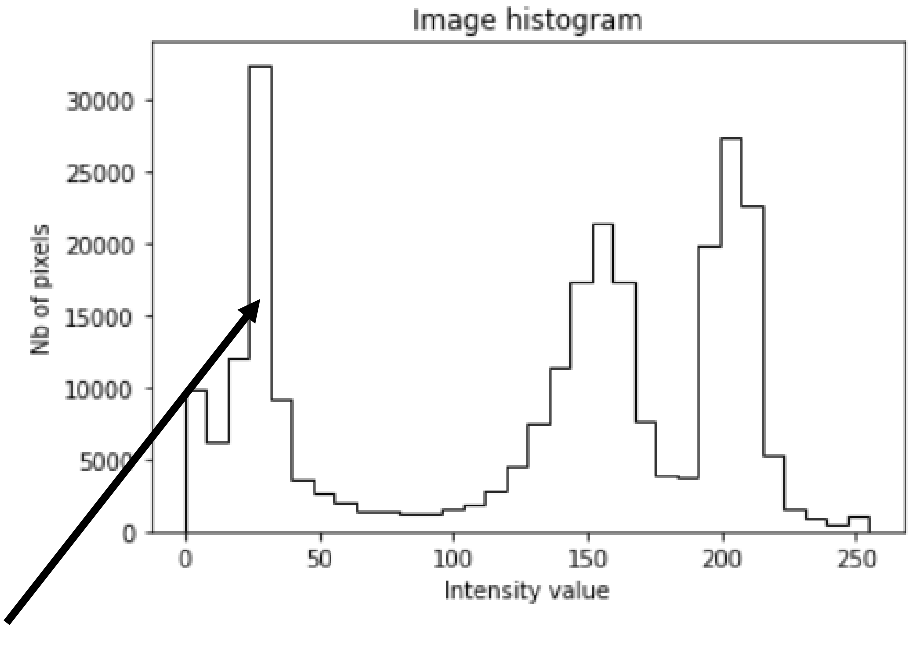


- Histogram

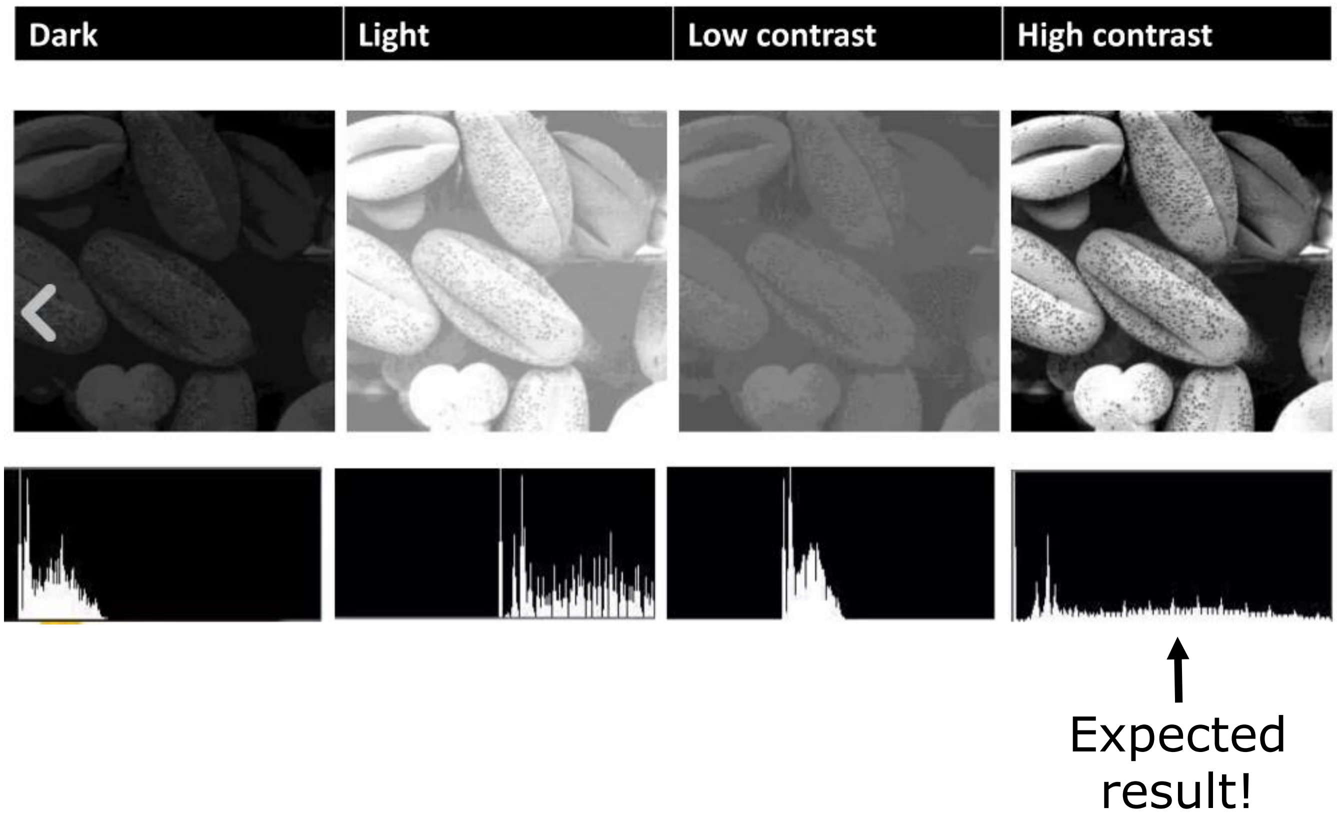
```
plt.hist(image.ravel(), bins=255,\
        density=False, histtype='step', color='black')
plt.xlabel('Intensity value')
plt.ylabel('Nb of pixels')
plt.title('Image histogram')
plt.show()
```




```
# With a different number of bins
plt.hist(image.ravel(), bins=32,\
         histtype='step', color='black')
plt.xlabel('Intensity value')
plt.ylabel('Nb of pixels')
plt.title('Image histogram')
plt.show()
```



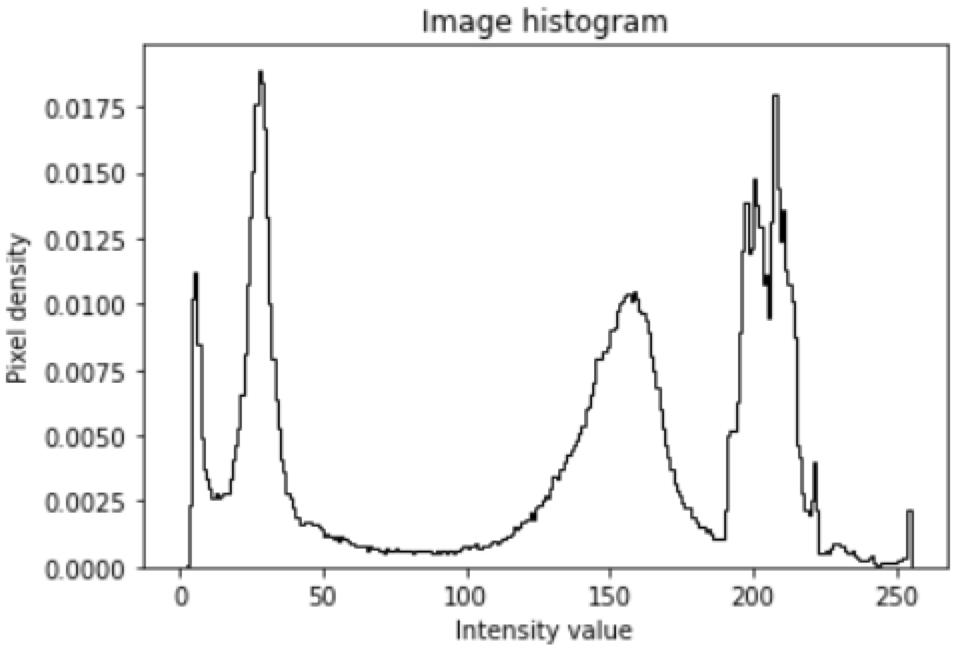
Histogram Processing



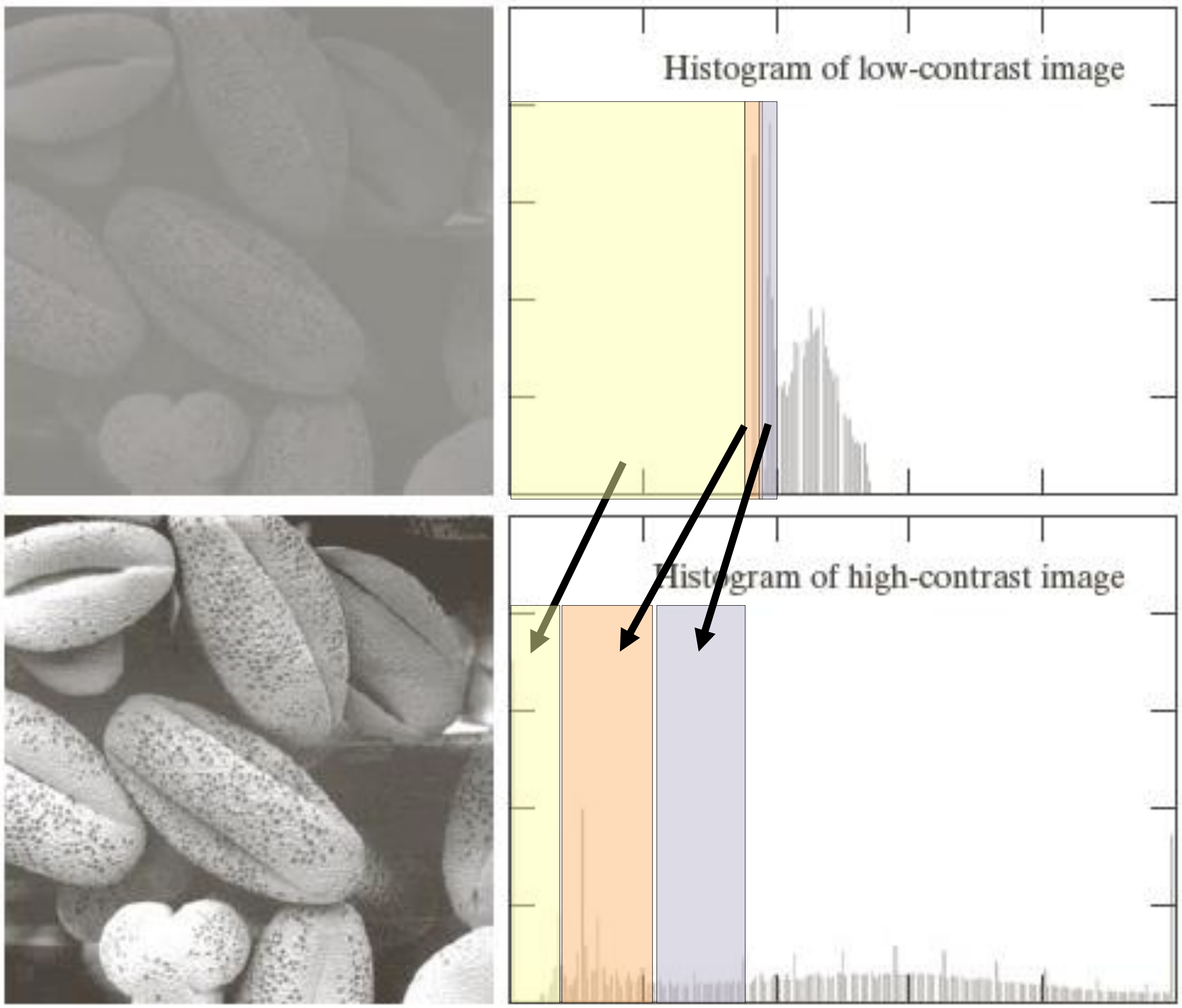
[COM2304: Intensity Transformation and Spatial Filtering – I (Intensity Transformation and Image Noise) | PPT (slideshare.net)]

- **Probability theory can be used to explain contrast stretching**

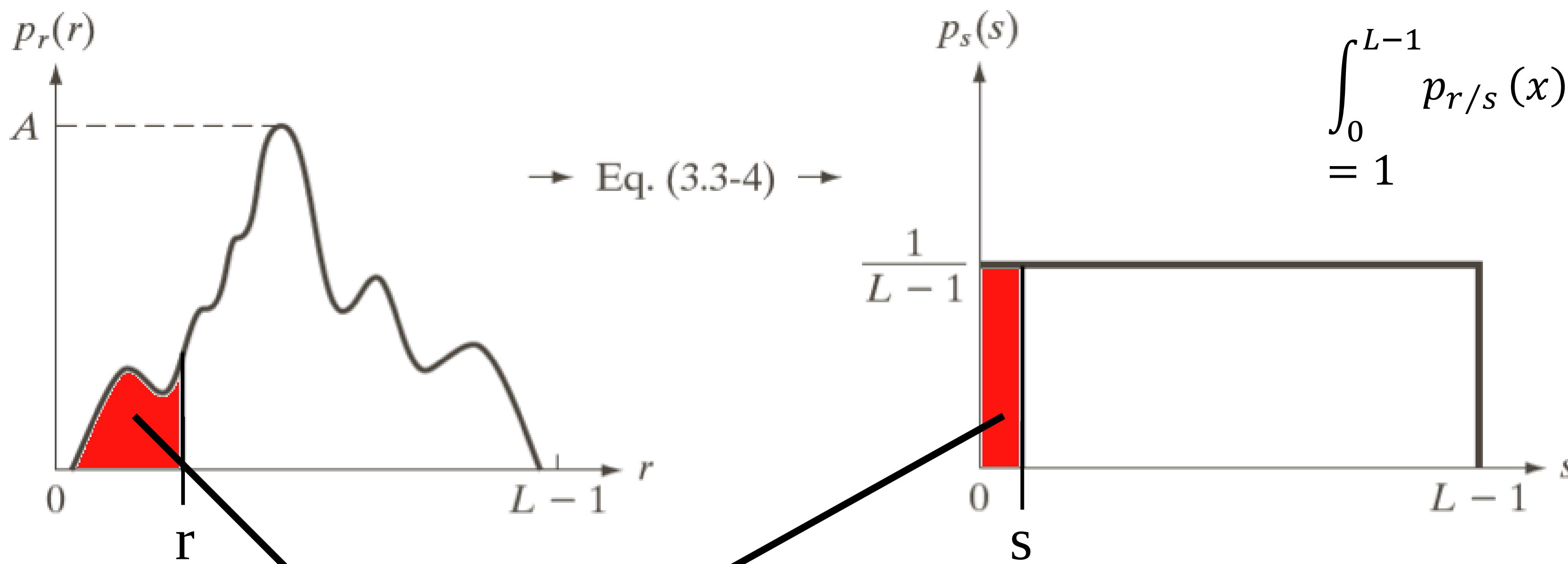
- Pixel intensities as random variable: R
- R takes the value r
- Probability density function: $p_R(r)$
- See the book




```
matplotlib.pyplot.hist(x, bins=None, range=None, density=False, weights=None,
cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical',
rwidth=None, log=False, color=None, label=None, stacked=False, *, data=None,
**kwargs) ¶                                                              [source]
```


Histogram of low-contrast image
Histogram of high-contrast image

Is there a transformation $s = T(r)$ (and an inverse transformation) that can map an arbitrary pdf to a *uniform* pdf?

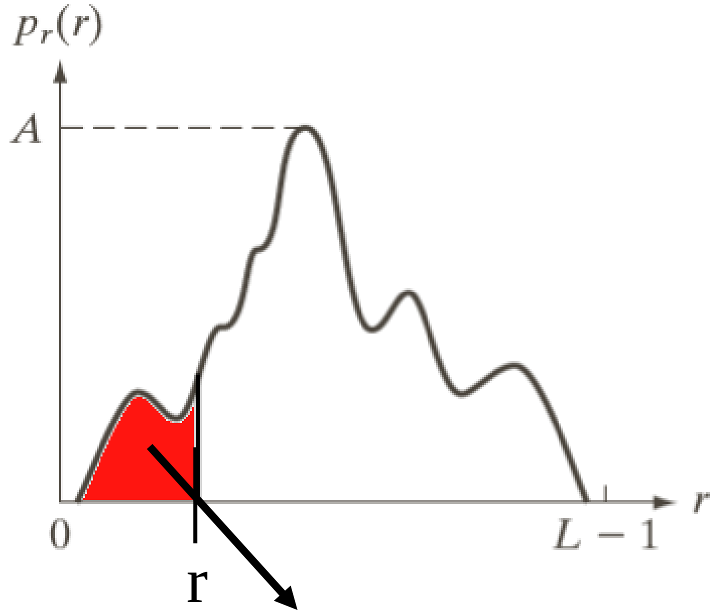


$$\int_0^{L-1} p_{r/s}(x)dx = 1$$

$$\int_0^r p_R(x)dx = \frac{s}{L-1}$$

$$\Leftrightarrow$$

$$s = (L-1)\int_0^r p_R(x)dx$$

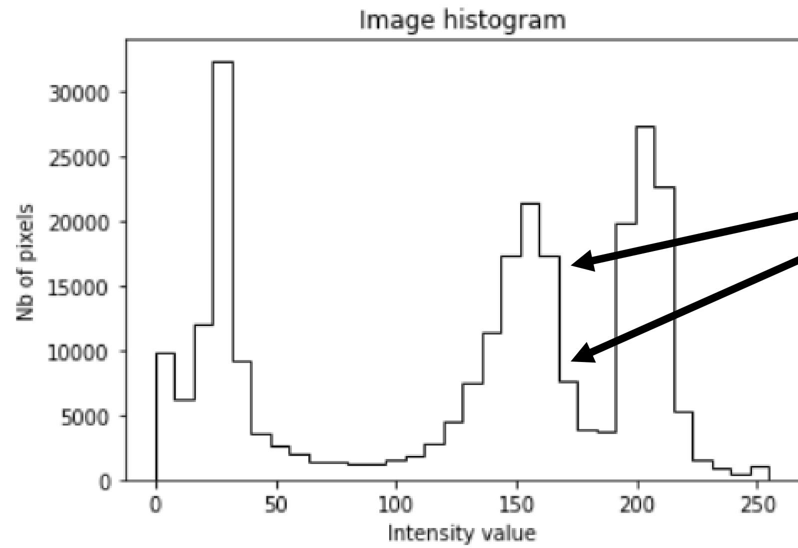


$$s = (L-1)\int_0^r p_R(x)\,dx$$

Our case: discrete values

$$s = (L-1)\sum_{i=0}^{r} p_R(i)$$

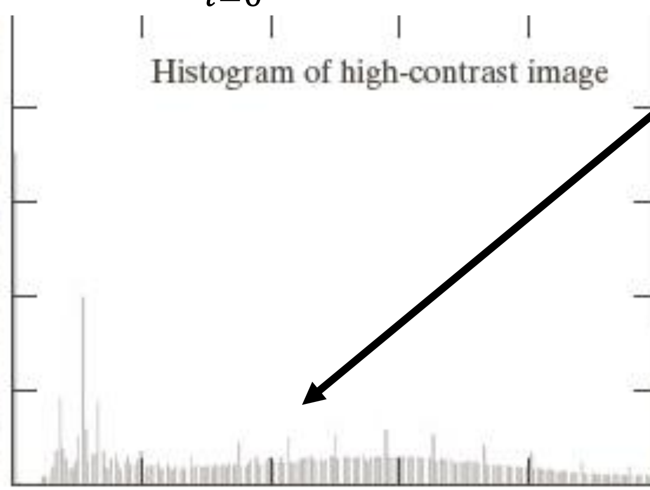
Given by the histogram



Steps

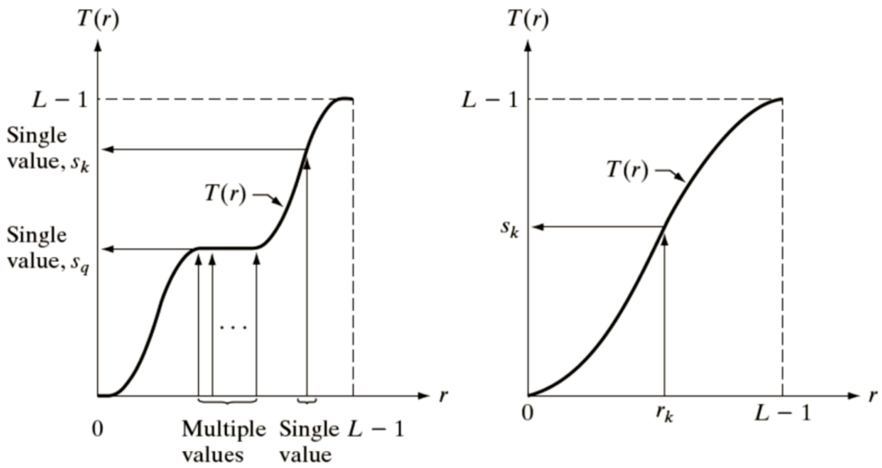
Our case: discrete values $$s = (L-1)\sum_{i=0}^{r} p_R\,(i)$$

Approximate flat histogram



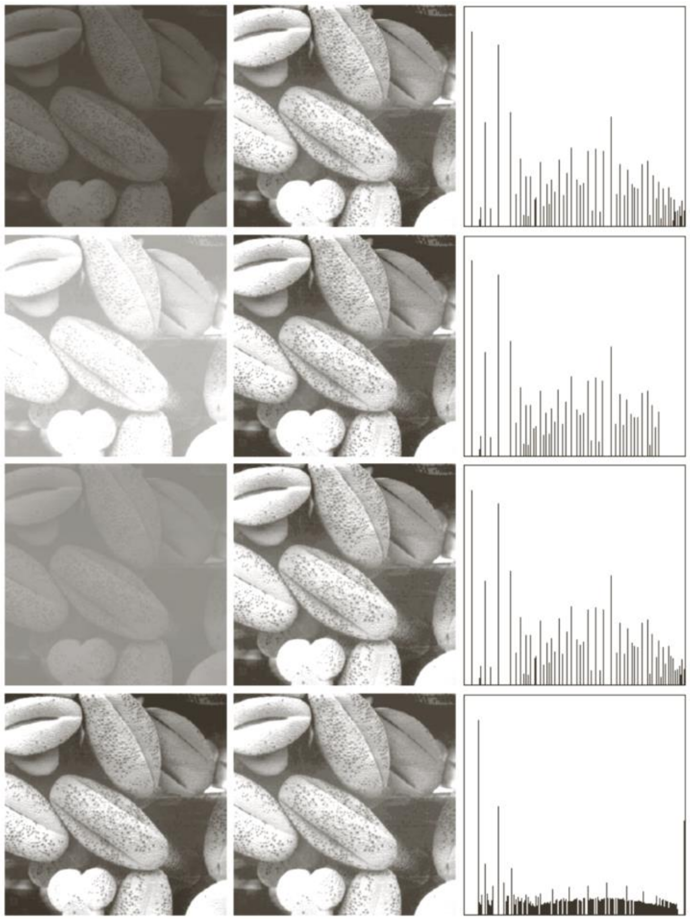
- Transformation Function (with Inverse)
- Cumulative distribution

- Want strictly monotonically increasing function



a b

FIGURE 3.17 (a) Monotonically increasing function, showing how multiple values can map to a single value. (b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.



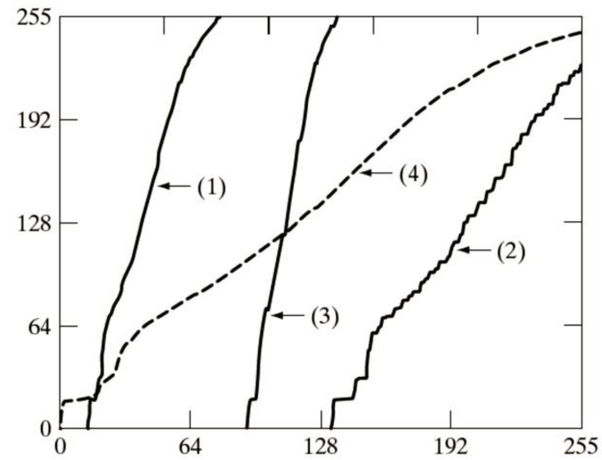


FIGURE 3.21 Transformation functions for histogram equalization. Transformations (1) through (4) were obtained from the histograms of the images (from top to bottom) in the left column of Fig. 3.20 using Eq. (3.3-8).

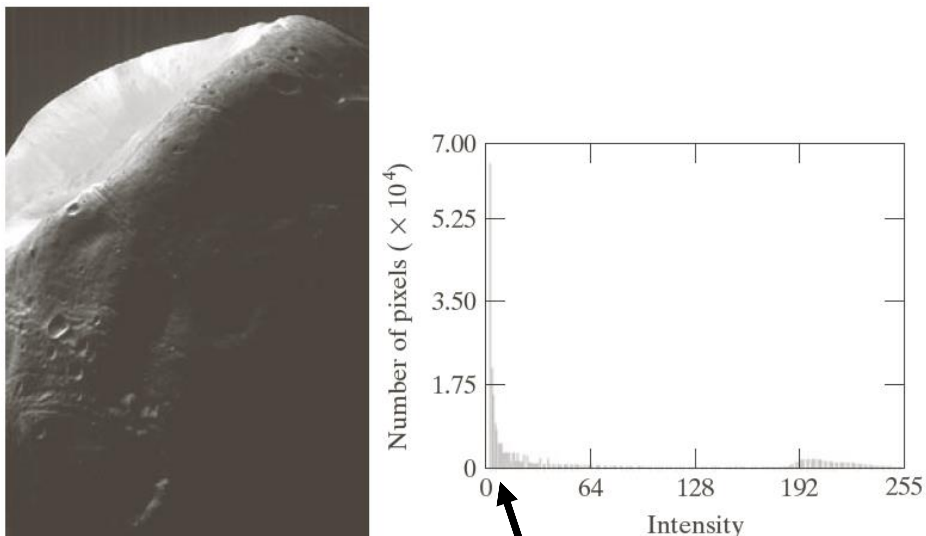
FIGURE 3.20 Left column: images from Fig. 3.16. Center column: corresponding histogram-equalized images. Right column: histograms of the images in the center column.

- We've seen

- Histogram equalization is a powerful tool for contrast stretching
- Fully automatic
- Data adaptive

- May sometimes produce not so good results
- Alternatives?

- **Limitations**: example, Image With Large Dark Areas



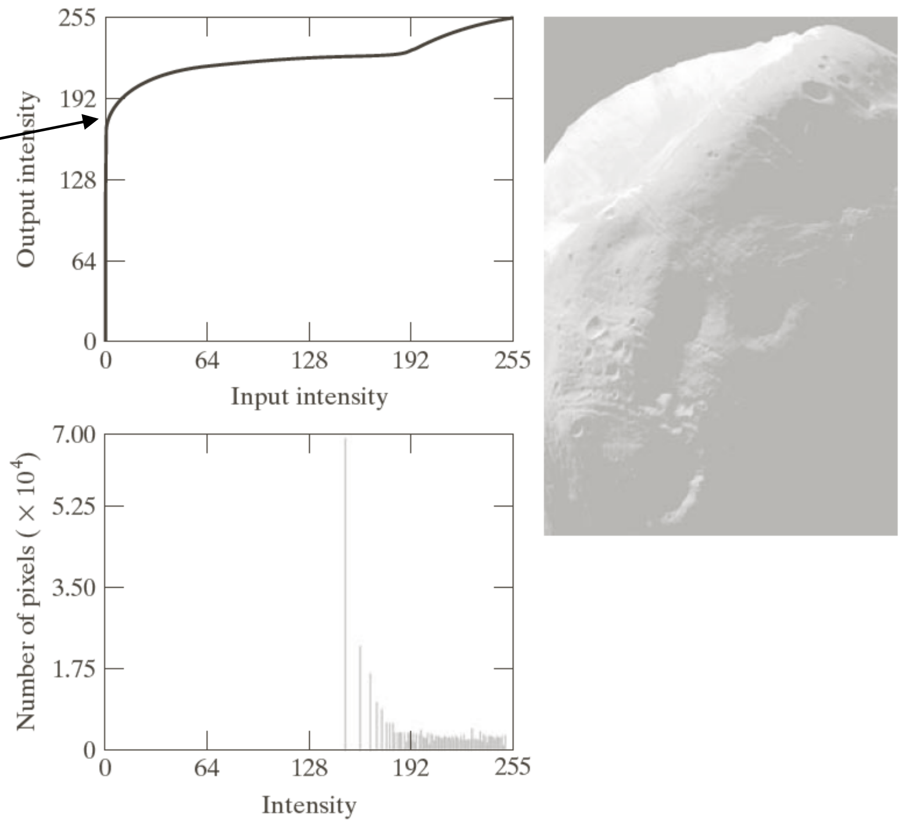
a b

FIGURE 3.23 (a) Image of the Mars moon Phobos taken by NASA's *Mars Global Surveyor.* (b) Histogram. (Original image courtesy of NASA.)

$$binvalues > \frac{1}{L-1}$$

- Histogram Equalized

Too many black pixels lead to a big jump



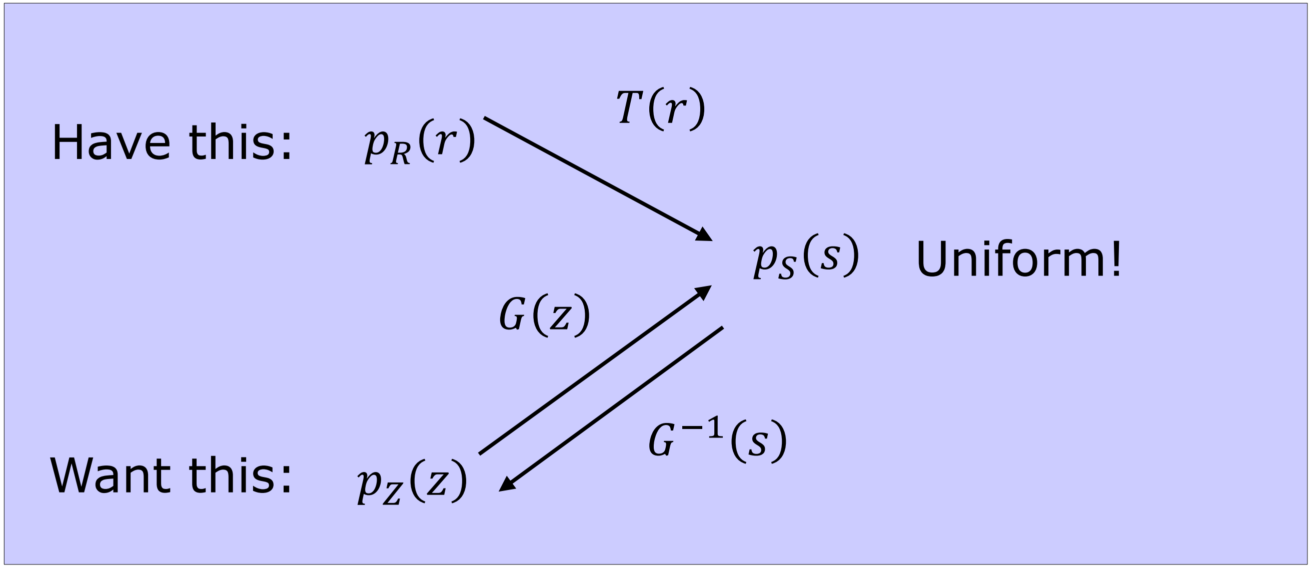
a b
c

FIGURE 3.24
(a) Transformation function for histogram equalization.
(b) Histogram-equalized image (note the washed-out appearance).
(c) Histogram of (b).

Alternatives for a better processing?

- **Generalization:** Histogram Specification

Is there a transformation s = $T(r)$ (and an inverse transformation) that can map an arbitrary pdf to another arbitrary pdf?



1) Histogram Specification Mapping

- Use the inverse relationship

- Where $s = G(z)$ such that $z = G^{-1}(s) = G^{-1}[T(r)]$

$$G(z) = (L-1)\int_0^z p_Z(w)dw \qquad T(r) = (L-1)\int_0^r p_R(w)dw$$

2) Histogram Approximation

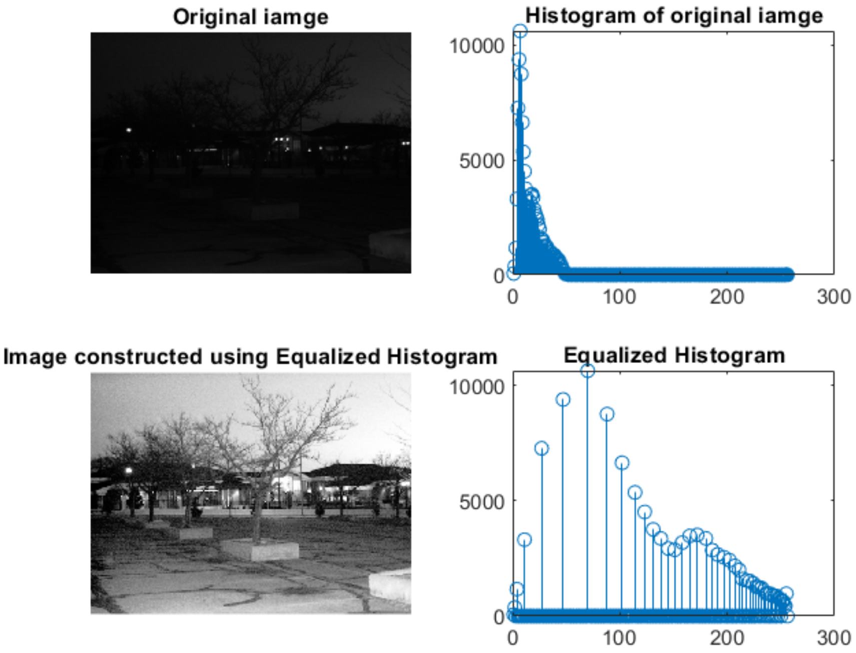
- Specify histogram $p(z_i)$ and compute

$$G(z_q) = (L-1)\sum p(z_i)$$

- for $q = 0, \ldots, L - 1$
- Use a table look-up procedure to find the inverse

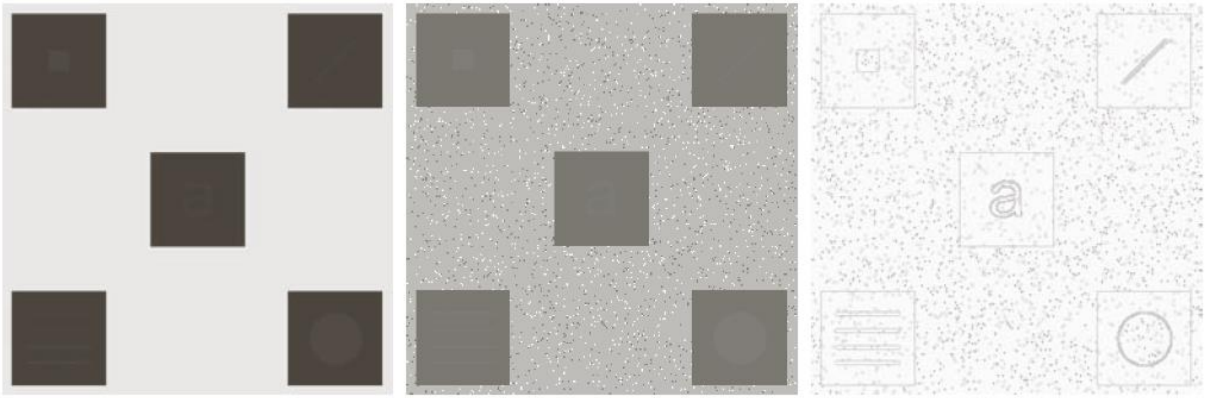
- **Local Histogram Processing**

- It is sometimes useful to transform intensities based on local image regions
  – Slide a neighborhood-mask over the image
  – Compute a histogram at each location
  – Map the center pixel
  – Repeat



[Quick Guide to Histogram Equalization for Clearer Images - Analytics Vidhya]

- Local Histogram Processing



a b c

FIGURE 3.26 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization applied to (a), using a neighborhood of size 3×3.

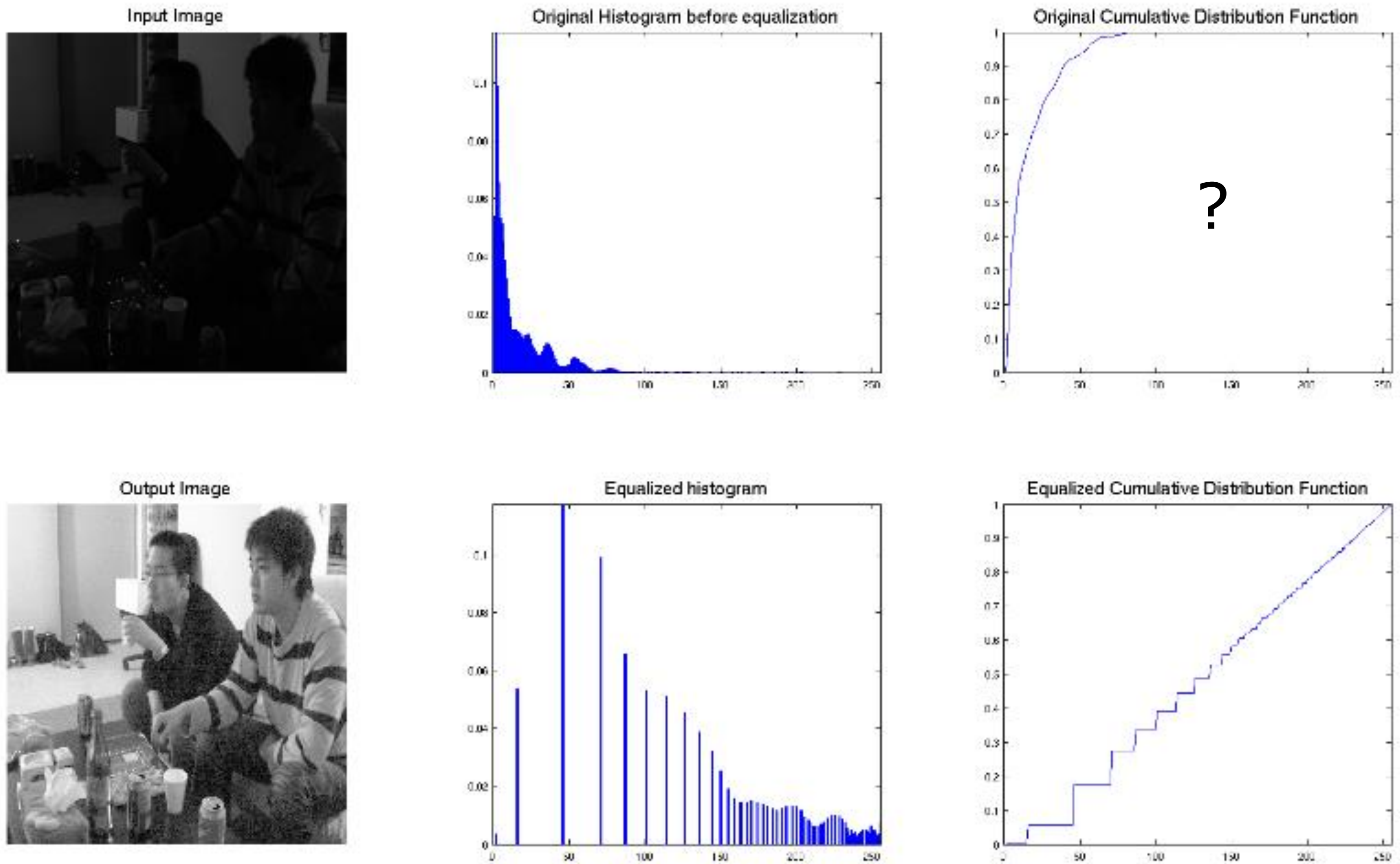
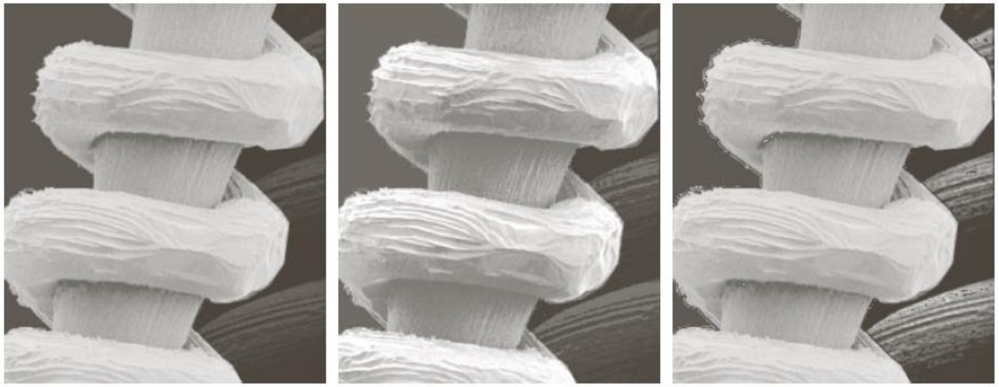


Image Processing (utah.edu)

- Mean and Variance Intensity Transformation

- Book!



a b c

FIGURE 3.27 (a) SEM image of a tungsten filament magnified approximately 130×. (b) Result of global histogram equalization. (c) Image enhanced using local histogram statistics. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

Intensity Transformation

- Intensity transformation can be expressed as:

$$\boldsymbol{s} = \boldsymbol{T}(\boldsymbol{r})$$

Where transformation ***T*** maps a pixel value ***r*** into a pixel values ***s***.
The values of pixels before processing is denoised as ***r*** and after ***s***.

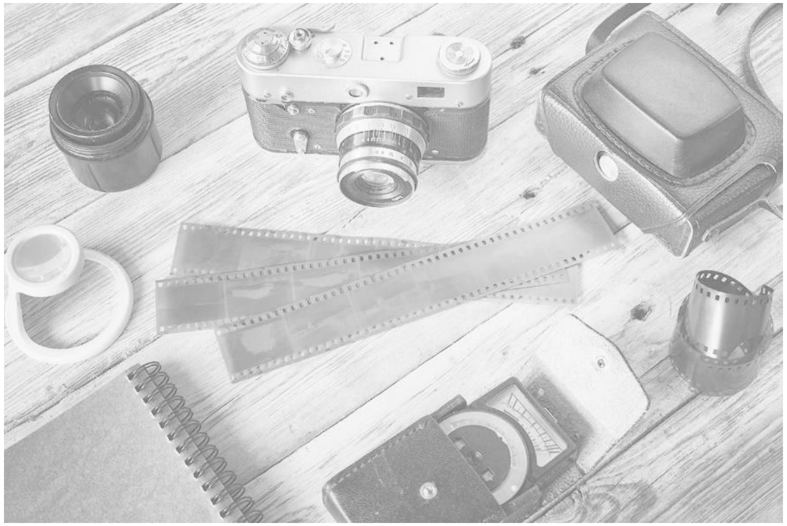
- Image enhancement can be done through gray level transformations and there are three basic gray level transformations:
  - *Linear*
  - *Logarithmic*
  - *Power-law*

COM2304: Intensity Transformation and Spatial Filtering – I (Intensity Transformation and Image Noise) | PPT (slideshare.net)

Mathematically, *log* transformations can be expressed as:

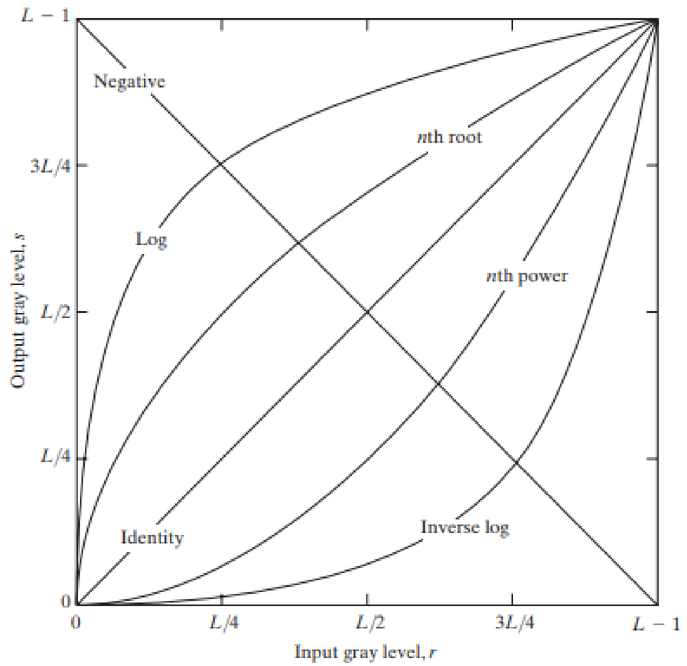
$$s = c \log(1+r).$$

Here, s is the output intensity, $r >= 0$ is the input intensity of the pixel, and c is a scaling constant. c is given by $255/(\log(1 + m))$, where m is the maximum pixel value in the image. It is done to ensure that the final pixel value does not exceed $(L - 1)$, or 255. Practically, *log* transformation maps a narrow range of low-intensity input values to a wide range of output values.



[Python | Intensity Transformation Operations on Images - GeeksforGeeks](https://www.geeksforgeeks.org/)

Log Transformations



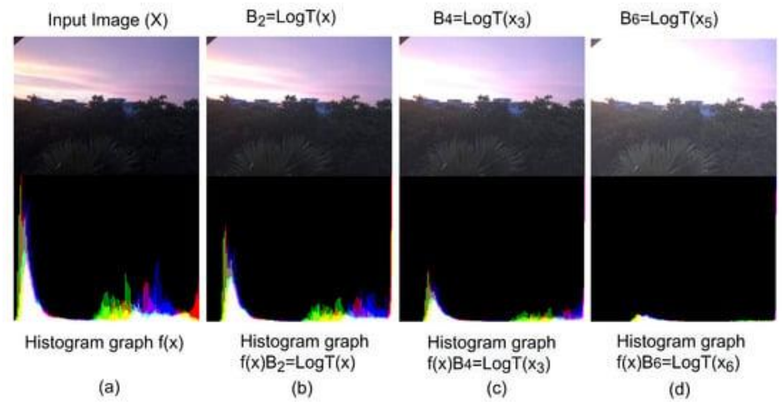
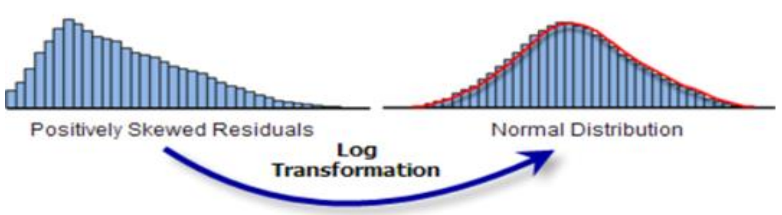


Figure. Analysis of Sequential Multiple Scale Logarithm Transformations of the Dark Input Image. (**a**) Histogram of the Given Image, (**b**) Histogram of the second stage Log Transformed Image, (**c**) Histogram of the fourth stage Log Transformed Image, (**d**) Histogram of the sixth stage Log Transformed Image.



[Applied Sciences | Free Full-Text | Multi-Range Sequential Learning Based Dark Image Enhancement with Color Upgradation (mdpi.com)]

[Improve Performance of your Model With Feature Engineering in Python! (analyticsvidhya.com)]

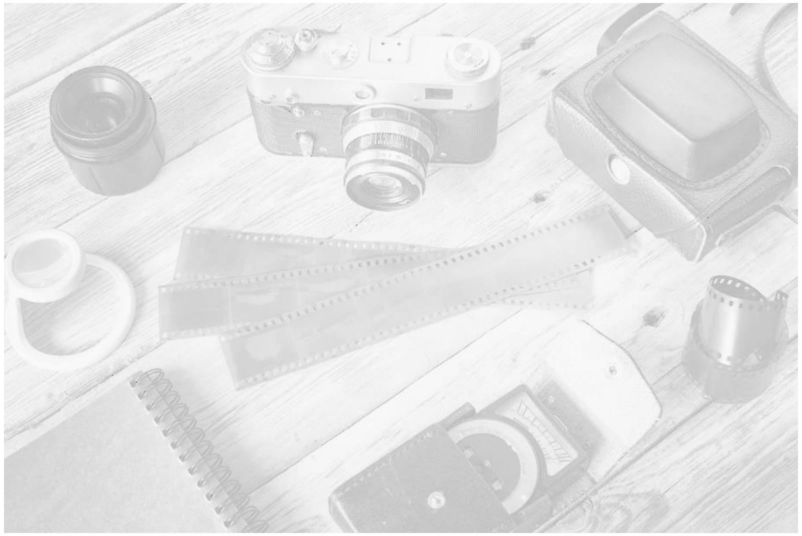
[Python | Intensity Transformation Operations on Images - GeeksforGeeks]

Power-Law (Gamma) Transformation
can be mathematically expressed as:

$$s = c \cdot r^{\gamma}.$$

Gamma correction is important for displaying images on a screen correctly, to prevent bleaching or darkening of images when viewed from different types of monitors with different display settings. This is done because our eyes perceive images in a gamma-shaped curve, whereas cameras capture images in a linear fashion.

Gamma = 0.1:



Gamma = 0.5:

Python | Intensity Transformation Operations on Images - GeeksforGeeks

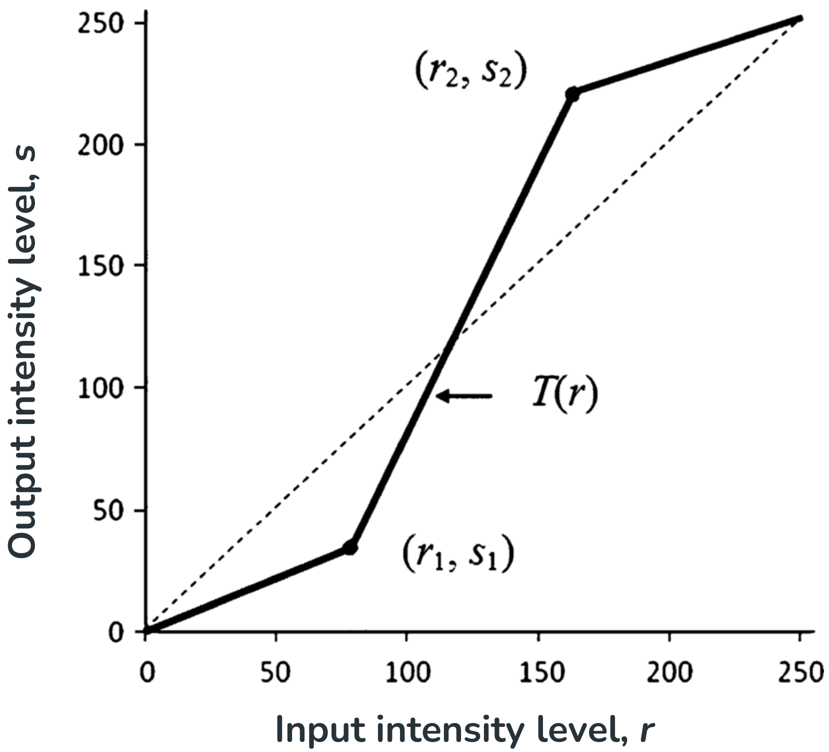
Gamma = 1.2:

Gamma = 2.2:

As can be observed from the outputs as well as the graph, *gamma*>1 (indicated by the curve corresponding to 'n^{th} power' label on the graph), the intensity of pixels decreases, i.e. the image becomes darker. Vice versa, *gamma*<1 (indicated by the curve corresponding to 'n^{th} root' label on the graph), the intensity increases, i.e. the image becomes lighter.

[Python | Intensity Transformation Operations on Images - GeeksforGeeks]

Piecewise-Linear Transformation Functions



With ($r1$, $s1$), ($r2$, $s2$) as parameters, the function stretches the intensity levels by essentially decreasing the intensity of the dark pixels and increasing the intensity of the light pixels. If $r1 = s1 = 0$ and $r2 = s2 = L - 1$, the function becomes a straight dotted line in the graph (which gives no effect). The function is monotonically increasing so that the order of intensity levels between pixels is preserved.

Python | Intensity Transformation Operations on Images - GeeksforGeeks