Imagery numérique

Theme 4

Histograms and Point Operations

Content of course

Semester 1

Theme 1: Introduction to image processing

Theme 2: The HVS perception and color

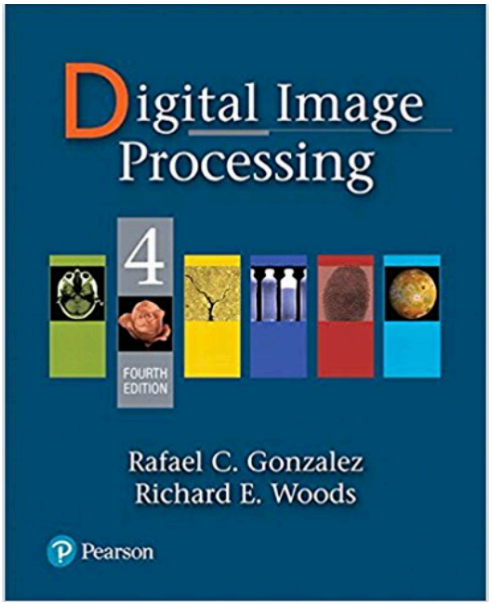
Theme 3: Image acquisition and sensing

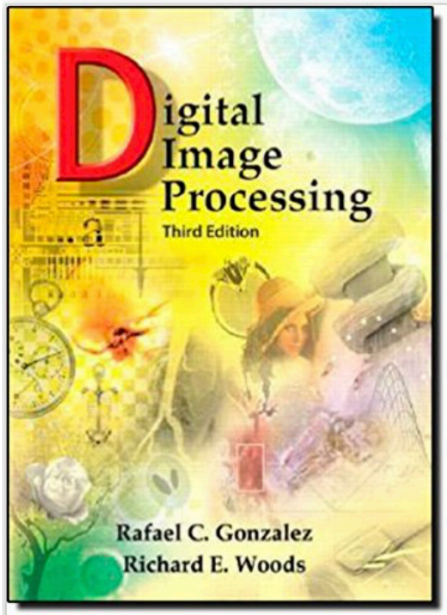
Theme 4: Histograms and point operations

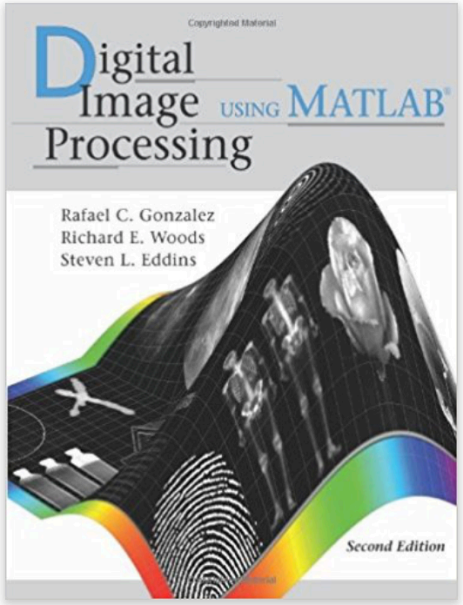
Theme 5: Geometric operations

Theme 6: Spatial filters

Recommended books





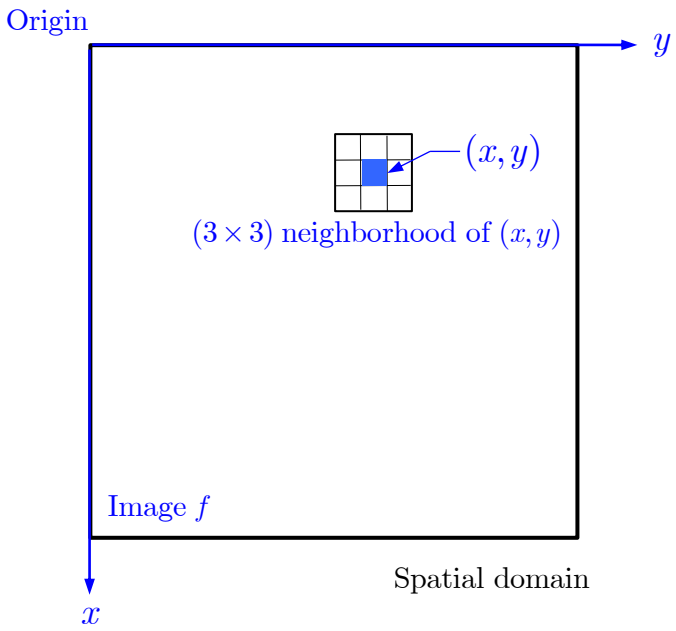


YouTube lectures

Intro to Digital Image Processing (ECSE-4540) Lectures, Spring 2015

by Prof. Rich Radke from Rensselaer Polytechnic Institute

Histograms and Point operations



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

Procedure of **spatial filtering**

1. Consider the neighborhood of pixel (x,y)
2. Apply a transformation of pixels $T\Big[.\Big]$
3. Move to a new pixel location

Spatial filter

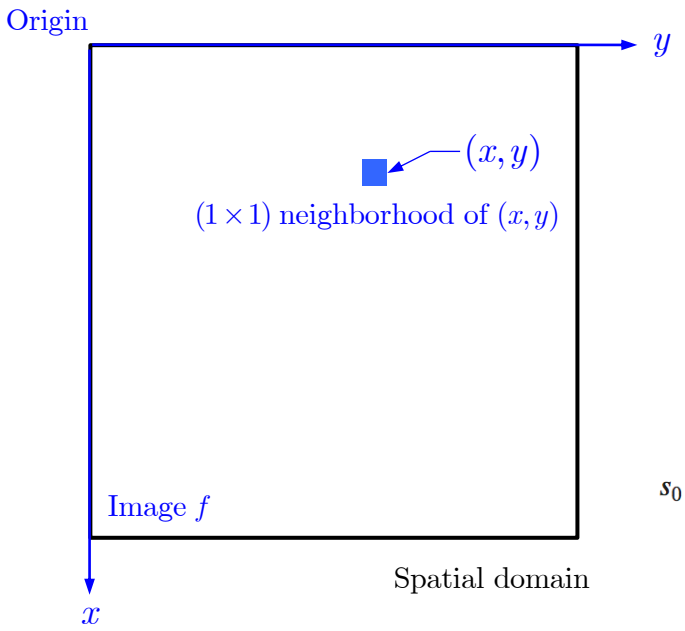
(3×3) neighborhood of (x,y)

Known as:

- *filter*
- *kernel*
- *template*
- *window*

Histograms and Point operations

Simplest form when the size of window is 1×1

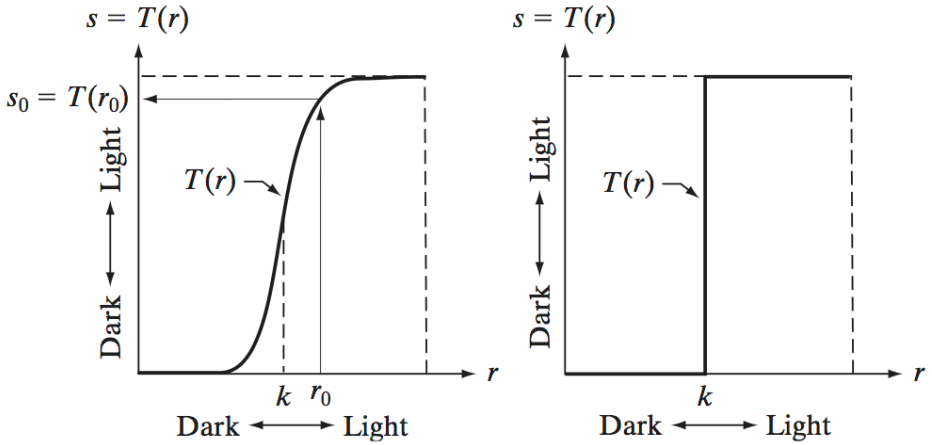


Spatial filtering → point operation

$$g(x,y) = T\big[f(x,y)\big] \qquad s = T\big[r\big]$$

output intensity — input intensity

It is a.k.a. intensity transformation



Histograms and Point operations

Histogram operations

Definition (histogram): given a discrete image $f(x,y)$ with $r \in \{0,1,\cdots,R\}$, the histogram is defined as:

$$hist(r) = \left|\left\{ f(x,y) \middle| f(x,y) = r \right\}\right|$$

That is the histogram corresponds to the number (cardinality) of pixels within the image $f(x,y)$ with the defined intensity r.

Matlab example

imhist

Histogram of image data

colla

Syntax

```
imhist(I)
imhist(I,n)
imhist(X,map)
[counts,binLocations] = imhist(I)
[counts,binLocations] = imhist(gpuarrayI,__)
```

Description

`imhist(I)` calculates the histogram for the intensity image `I` and displays a plot of the histogram. The number of bins in the histogram is determined by the image type.

`imhist(I,n)` calculates the histogram, where `n` specifies the number of bins used in the histogram. `n` also specifies the length of the colorbar displayed at the bottom of the histogram plot.

`imhist(X,map)` displays a histogram for the indexed image `X`. This histogram shows the distribution of pixel values above a colorbar of the colormap `map`. The colormap must be at least as long as the largest index in `X`. The histogram has one bin for each entry in the colormap.

`[counts,binLocations] = imhist(I)` returns the histogram counts in `counts` and the bin locations in `binLocations` so that `stem(binLocations,counts)` shows the histogram. For indexed images, `imhist` returns the histogram counts for each colormap entry. The length of `counts` is the same as the length of the colormap.

`[counts,binLocations] = imhist(gpuarrayI,__)` performs the histogram calculation on a GPU. The input image and the return values are gpuArrays. This syntax requires the Parallel Computing Toolbox™. When the input image is a gpuArray, `imhist` does not automatically display the histogram. To display the histogram, use `stem(binLocations,counts)`.

Code Generation support: Yes.

MATLAB Function Block support: Yes.

Matlab example

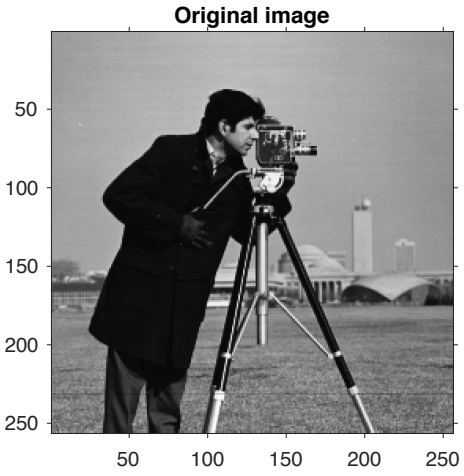
histo.m

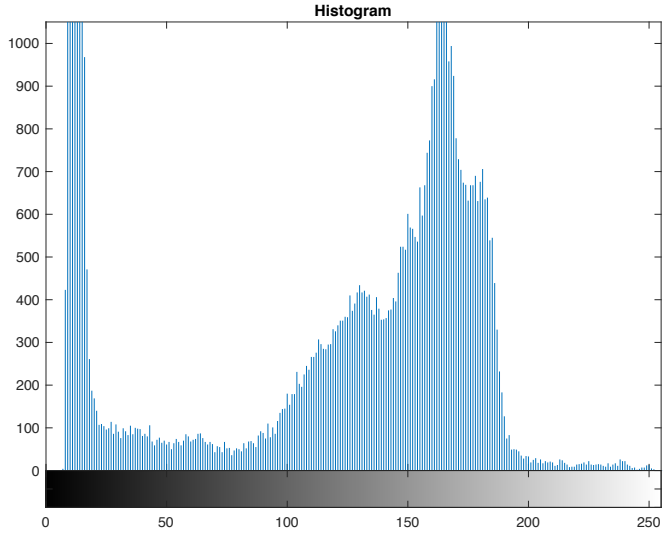
```
% Image histogram
% doc imhist

% Read image
im=imread('cameraman.tif');

% Show image
figure(1); imshow(im); title('Original image')

% Show histogram of image
figure(2); imhist(im); title('Histogram')
```

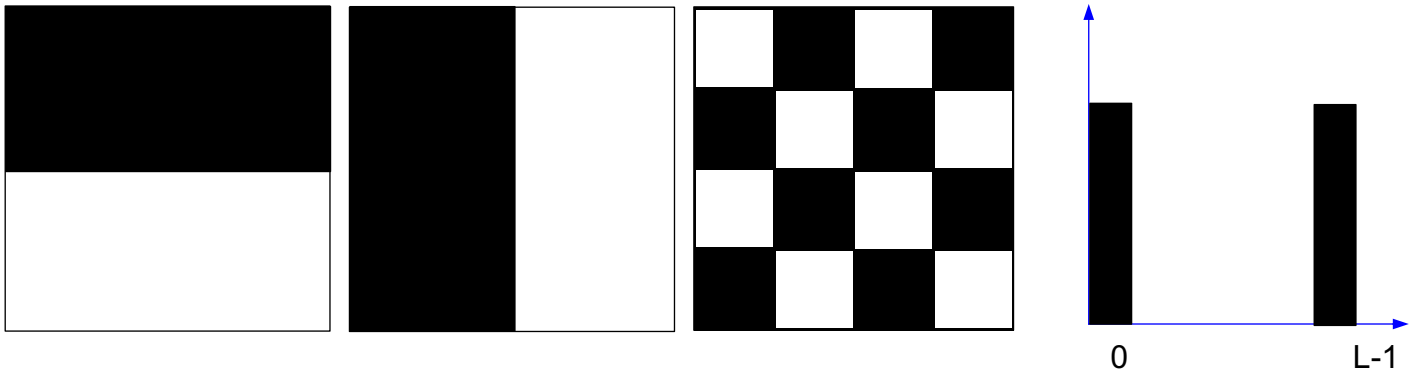




Remarks about histogram

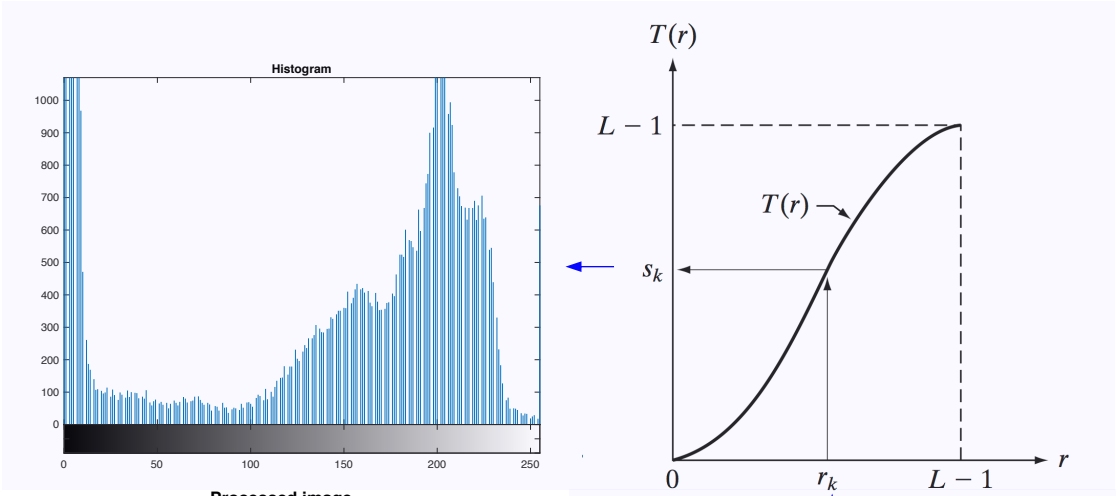
Is histogram informative? Yes

Is histogram unique? Many images might have the same histogram!



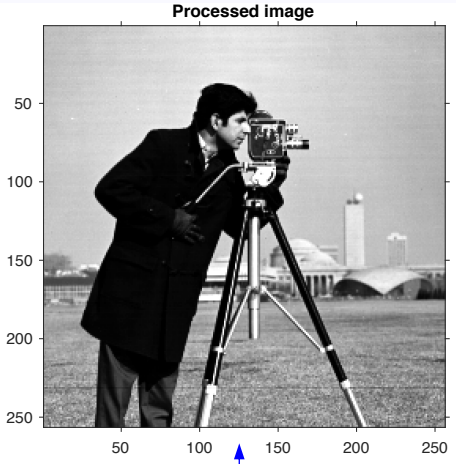
Important: histogram is not a pdf or pmf!

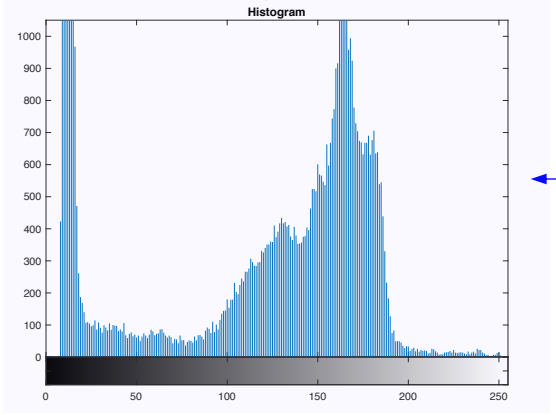
Point operations: schematic representation

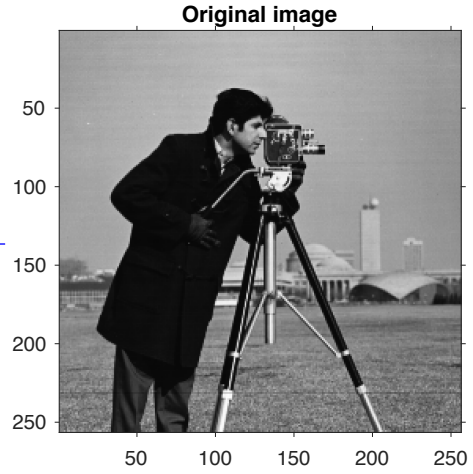


$$g(x,y) = T\left[f(x,y)\right]$$

$$s = T\left[r\right]$$



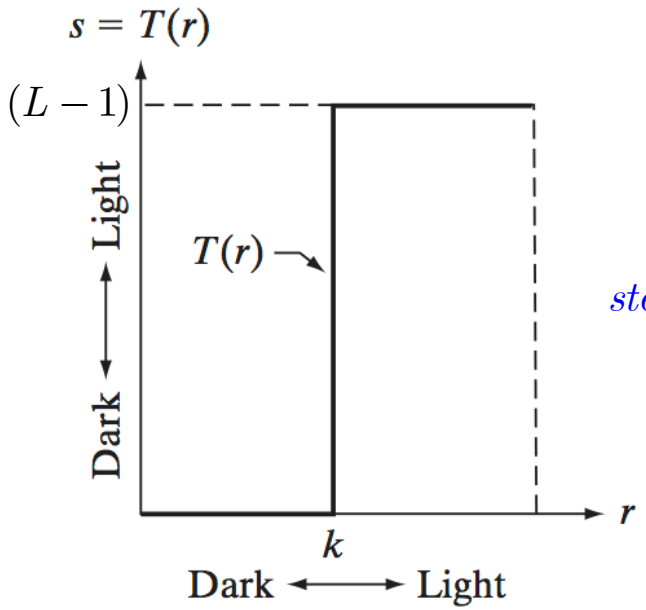




$$g(x,y) = T\left[f(x,y)\right]$$

Examples of point operations: Thresholding

$$s = T\left[r\right] = step\left(r - k\right)(L-1)$$



$$step\left(r - k\right) = \begin{cases} 1, \text{ if } r \geq k, \\ 0, \text{ otherwise.} \end{cases}$$

Examples of point operations: Thresholding

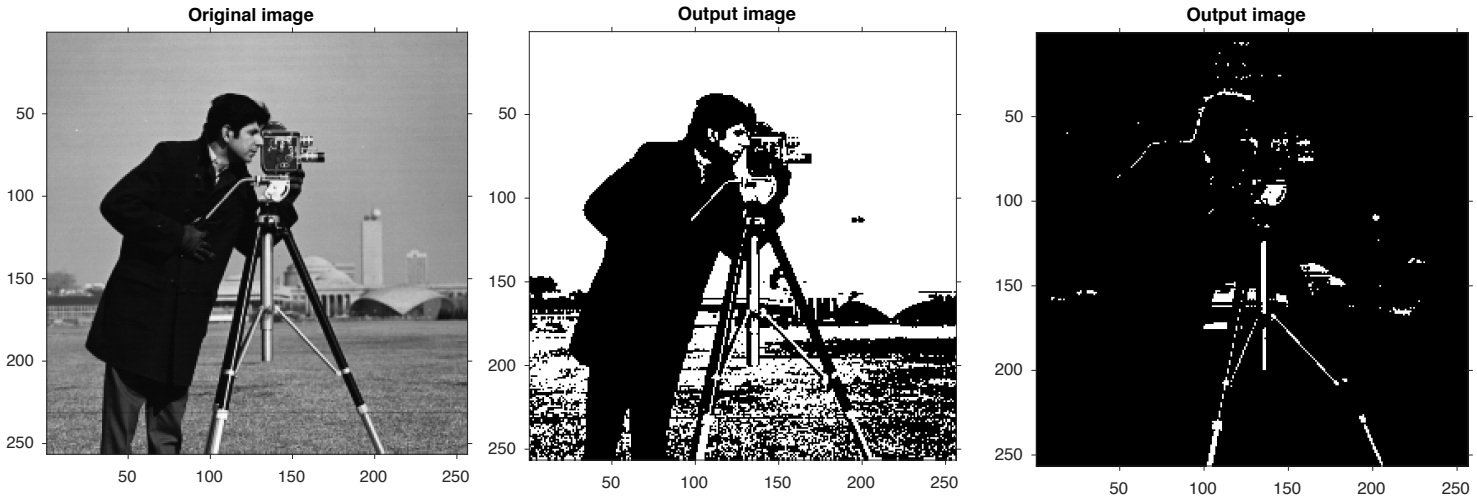
thresholding.m*

```
1     % Image thresholding
2
3
4     % Read image
5 -   im=imread('cameraman.tif');
6
7     % Show image
8 -   figure(1); imshow(im); title('Original image')
9
10 -  out=(im>128);
11    % Show output image
12 -  figure(2); imshow(out); title('Output image')
13
14 -  out=(im>190);
15    % Show output image
16 -  figure(3); imshow(out); title('Output image')
17
```

```
>> whos out
  Name         Size                Bytes  Class      Attributes

  out        256x256               65536  logical
```

Examples of point operations: Thresholding



Examples of point operations: Thresholding

Where the thresholding is used?

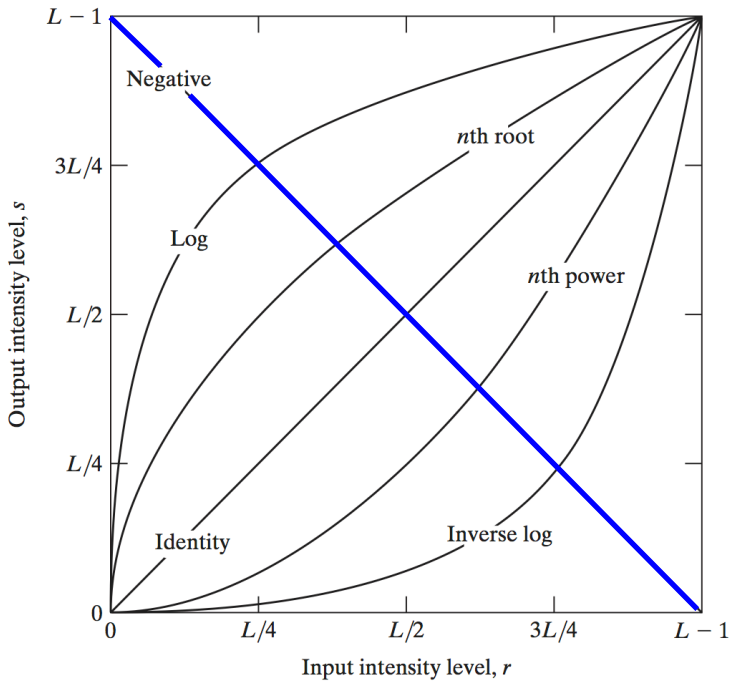
- Image is divided according to the graylevels/colors
- Small parts or parts of special graylevels/colors can be filtered
- Multiple thresholds can be applied
- The resulting image is binary. This is a sort of compression.

The thresholding is:

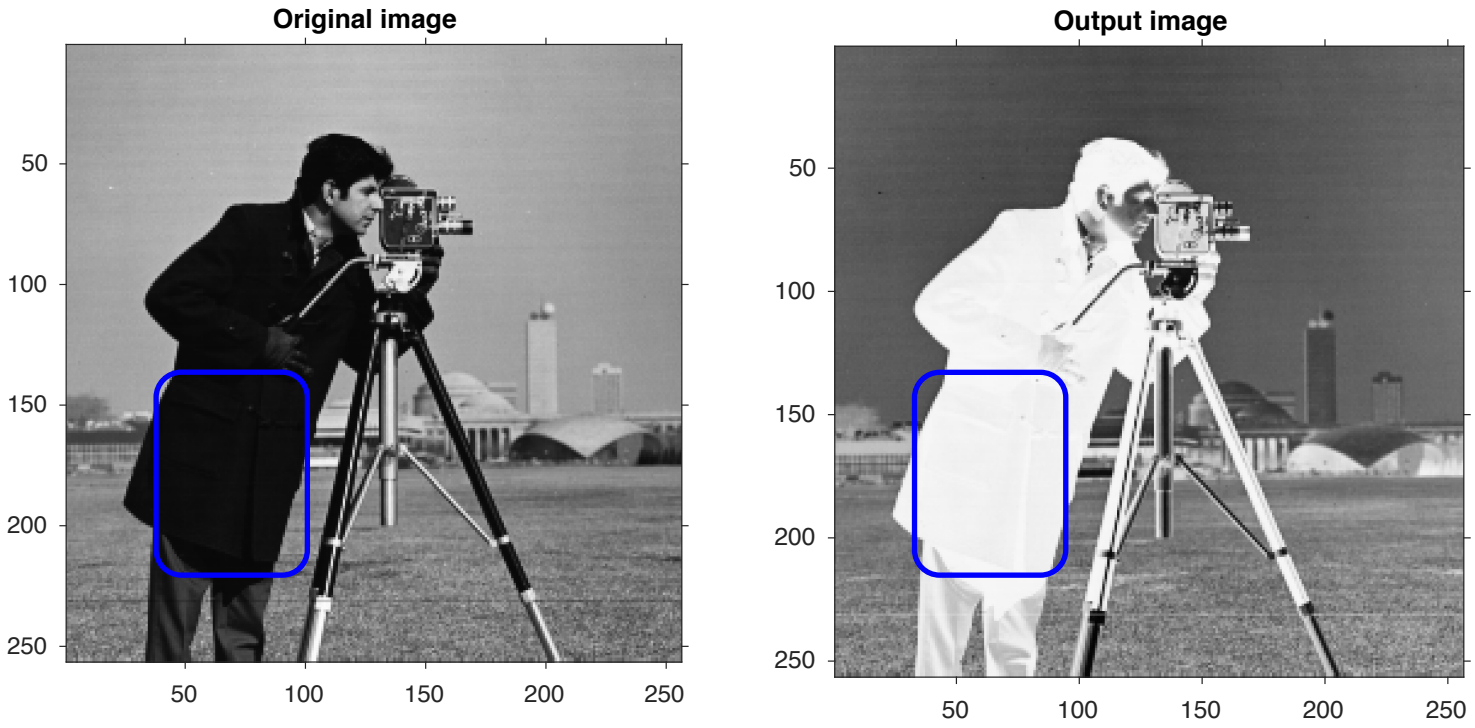
- Information lossy
- Non-invertible operation

Examples of point operations: Digital negative

$$s = T\left[r\right] = (L-1) - r$$

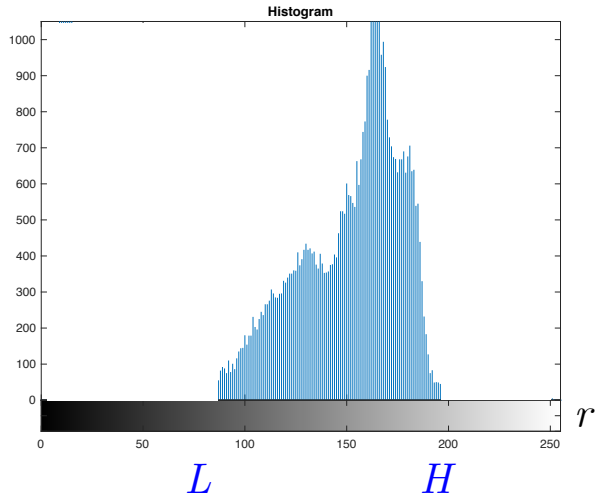


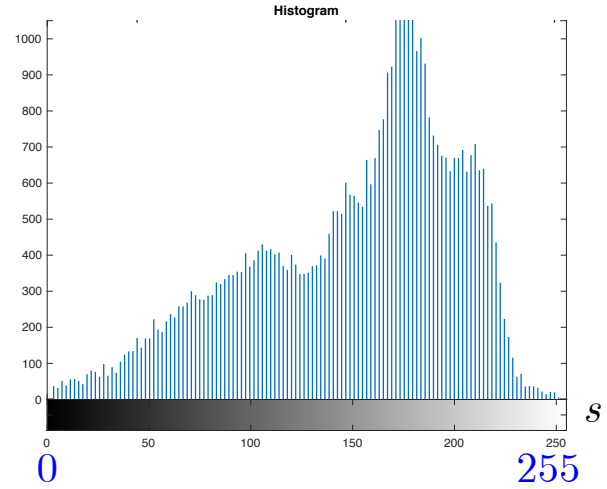
Examples of point operations: Digital negative



Why needed? New interesting things might appear to the HVS (which is non-linear) especially on the dark background.

Examples of point operations: Contrast stretching





The dynamic range is very limited.

Goal is to stretch it to the full dynamic range.

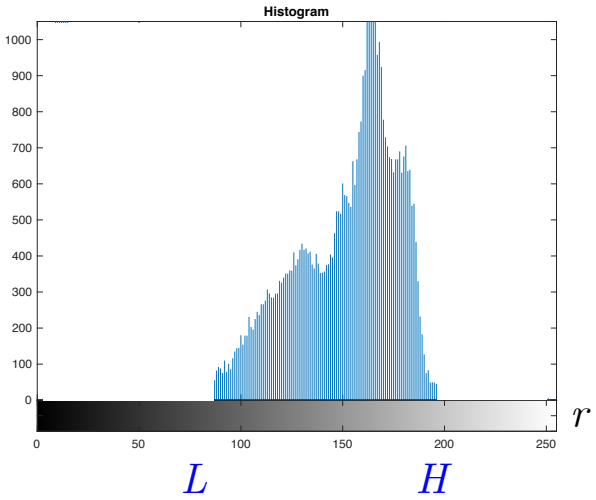
$$s = T\left[r\right] = \frac{\overbrace{255}^{\text{new dynamic range}}}{\underbrace{H - L}_{\text{old dynamic range}}}\left(r - L\right)$$

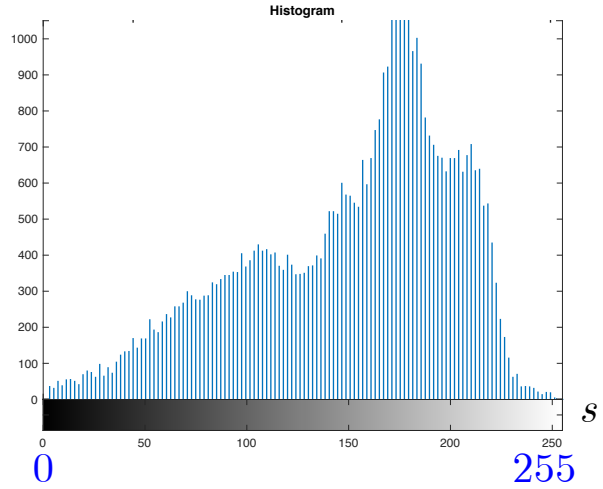
$$r = L : s = \frac{255}{H - L}\left(L - L\right) = 0$$

$$r = H : s = \frac{255}{H - L}\left(H - L\right) = 255$$

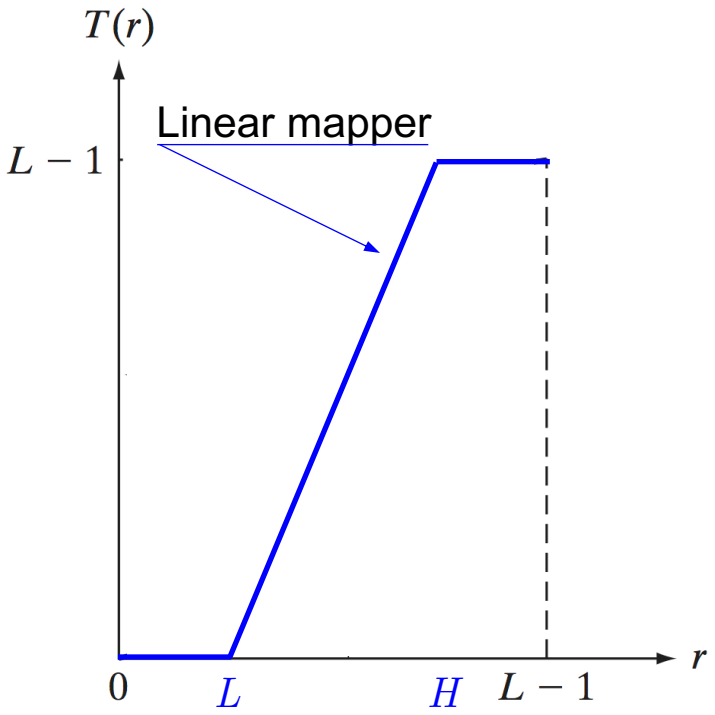
Remark: imshow(im,[]) is doing exactly this mapping between 0 and 255

Examples of point operations: Contrast stretching





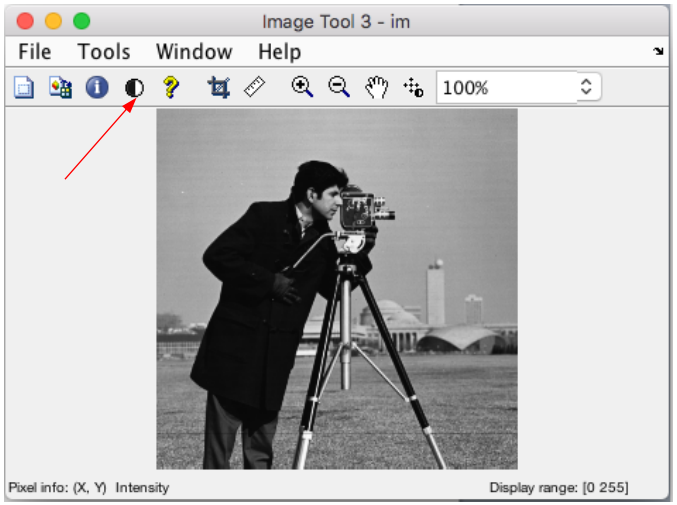
$$s = T[r] = \frac{255}{H - L}(r - L)$$

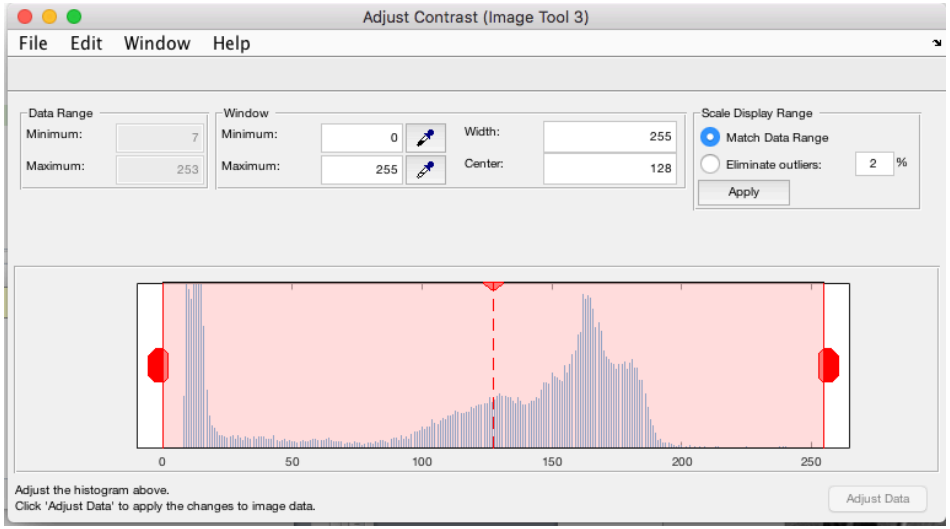


Point operations: Matlab - imtool

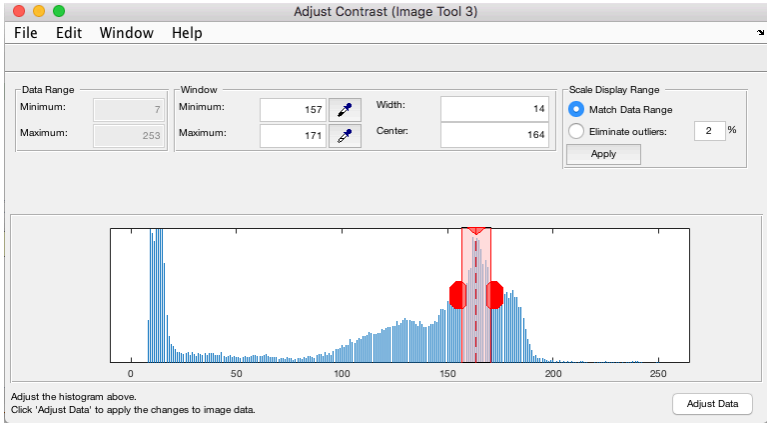
imtool_demo.m

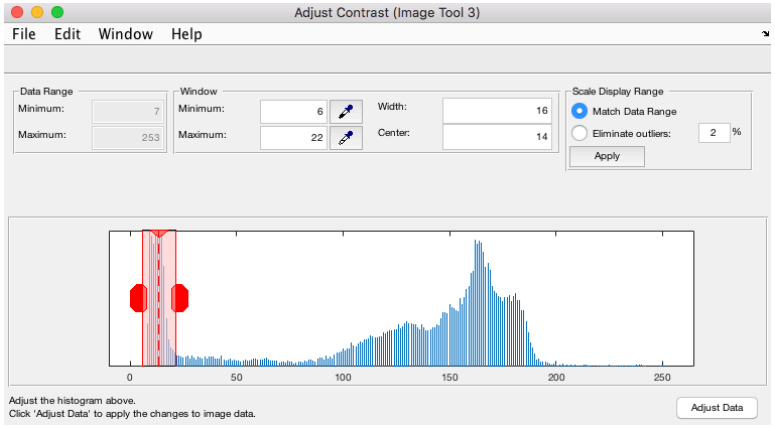
```
1   % Imtool
2
3
4   % Read image
5 - im=imread('cameraman.tif');
6
7   % Show image
8 - figure(1); imshow(im); title('Original image')
9 - imtool(im)
```





Point operations: Matlab - imtool





Unseen details

Remark: it is only for visualization! No new image is produced!

Point operations: imadjust

imadjust

Adjust image intensity values or colormap

collapse all in page

Syntax

```
J = imadjust(I)                                                    example
J = imadjust(I,[low_in; high_in],[low_out; high_out])              example
J = imadjust(I,[low_in; high_in],[low_out; high_out],gamma)
newmap = imadjust(map,[low_in; high_in],[low_out; high_out],gamma)
RGB2 = imadjust(RGB1, __ )                                         example
gpuarrayB = imadjust(gpuarrayA, __ )                               example
```

Description

`J = imadjust(I)` maps the intensity values in grayscale image `I` to new values in `J` such that 1% of data is saturated at low and high intensities of `I`. This increases the contrast of the output image `J`. This syntax is equivalent to `imadjust(I,stretchlim(I))`. example

`J = imadjust(I,[low_in; high_in],[low_out; high_out])` maps the values in `I` to new values in `J` such that values between `low_in` and `high_in` map to values between `low_out` and `high_out`. example

Note If `high_out` is less than `low_out`, `imadjust` reverses the output image, as in a photographic negative.

`J = imadjust(I,[low_in; high_in],[low_out; high_out],gamma)` maps the values in `I` to new values in `J`, where `gamma` specifies the shape of the curve describing the relationship between the values in `I` and `J`.

`newmap = imadjust(map,[low_in; high_in],[low_out; high_out],gamma)` transforms the *m*-by-3 array colormap associated with an indexed image. `low_in`, `high_in`, `low_out`, and `high_out` must be 1-by-3 vectors. `gamma` can be a 1-by-3 vector that specifies a unique gamma value for each channel or a scalar that specifies the value used for all three channels. The rescaled colormap `newmap` is the same size as `map`.

`RGB2 = imadjust(RGB1, __ )` performs the adjustment on each plane (red, green, and blue) of the RGB image `RGB1`. If `low_in`, `high_in`, `low_out`, `high_out`, and `gamma` are scalars, `imadjust` applies the same mapping to the red, green, and blue components of the image. To specify unique mappings for each color component of the image, specify `low_in`, `high_in`, `low_out`, `high_out`, and `gamma` as 1-by-3 vectors. example

`gpuarrayB = imadjust(gpuarrayA, __ )` performs the contrast adjustment on a GPU. The input gpuArray `gpuarrayA` is an intensity image, RGB image, or a colormap. The output gpuArray `gpuarrayB` is the same as the input image. This syntax requires the Parallel Computing Toolbox™. example

Code Generation support: Yes.

MATLAB Function Block support: Yes.

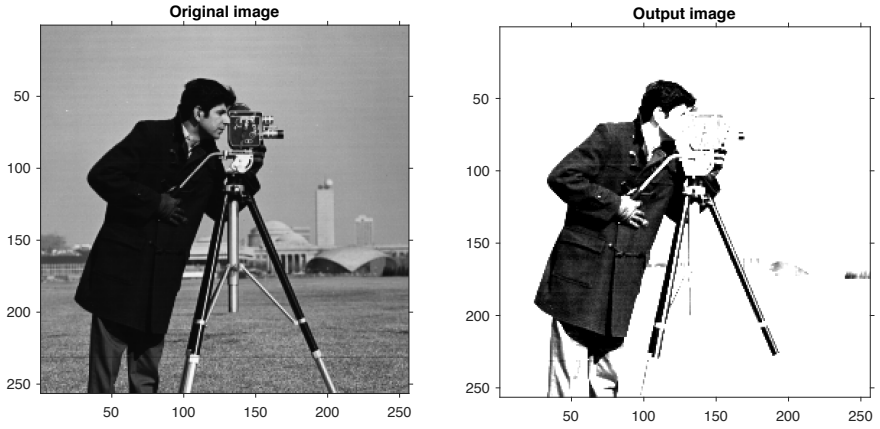
One provides low_in and high_in in the original image, and low_out and high_out in the output image. The chosen method is a linear mapping.

Point operations: imadjust

imadjust_demo.m

```
1    % Imadjust
2
3
4    % Read image
5 -  im=imread('cameraman.tif');
6
7    % Show image
8 -  figure(1); imshow(im); title('Original image')
9
10 - out= imadjust(im,[10/255; 35/255],[0/255; 255/255]);
11   %out= imadjust(im);
12
13 - figure(2); imshow(out); title('Output image')
14
```

We have chosen about the same input and output ranges as in the manually chosen example.

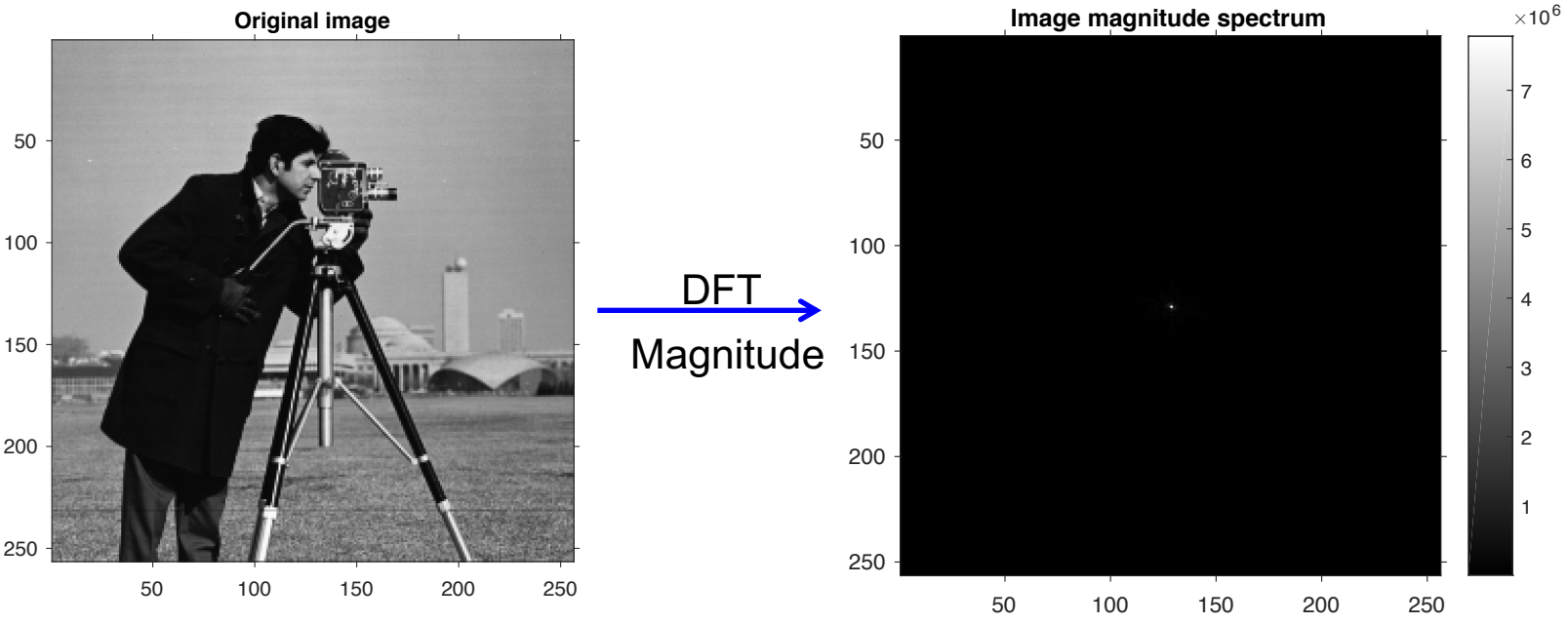


Remark: the dynamic range for the input images is $(0,1)$

Point operations: log transformation

It is very often we need to display the images with a very high dynamic range.

Example: Fourier transform images due to the energy compaction property posses the dynamic range between 0 and 10^6 or higher. [We will study the Fourier transform in Theme 7].

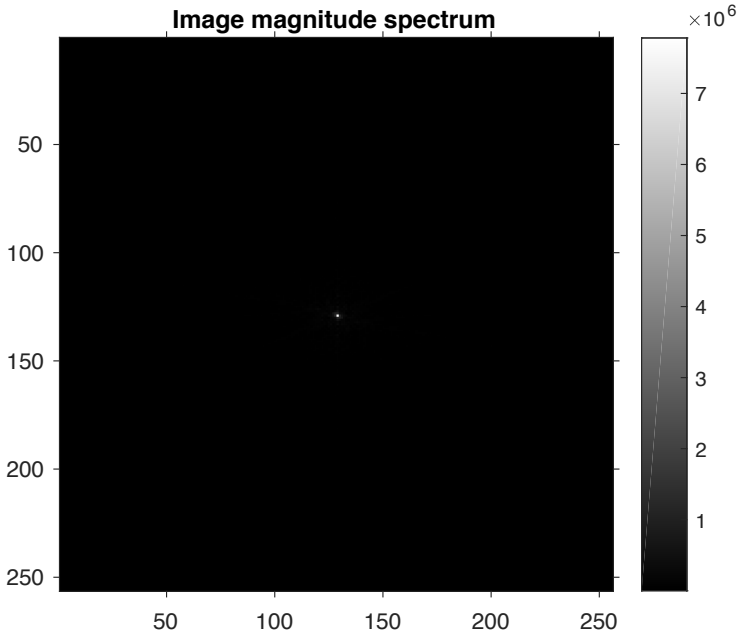


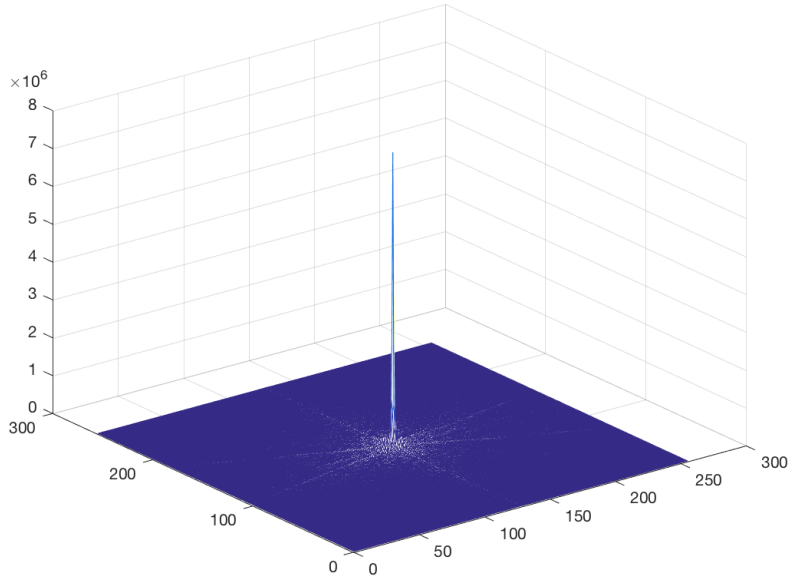
Point operations: log transformation

fft_mag_visualization_demo.m*

```
3      % Read image
4      %im=double(rgb2gray(imread('lena.png')));
5 -    im=double((imread('cameraman.tiff')));
6
7      % Show image
8 -    figure(1); imshow(im, []); title('Original image')
9
10
11     % Computed 2d FFT of image
12 -   out= fft2(im);
13
14     % Computed magnitude of FFT representation
15 -   out_mag= fftshift(abs(out));
16 -   figure(2); imshow(out_mag, []); title('Image magnitude spectrum');colorbar
17 -   figure(3); mesh(out_mag)
18
19
20     % Computed log-magnitude of FFT representation
21 -   out_mag_log=(log(out_mag+1));
22 -   figure(4); imshow(out_mag_log, []); title('Log image magnitude spectrum');colorbar
23 -   figure(5); mesh(out_mag_log)
```

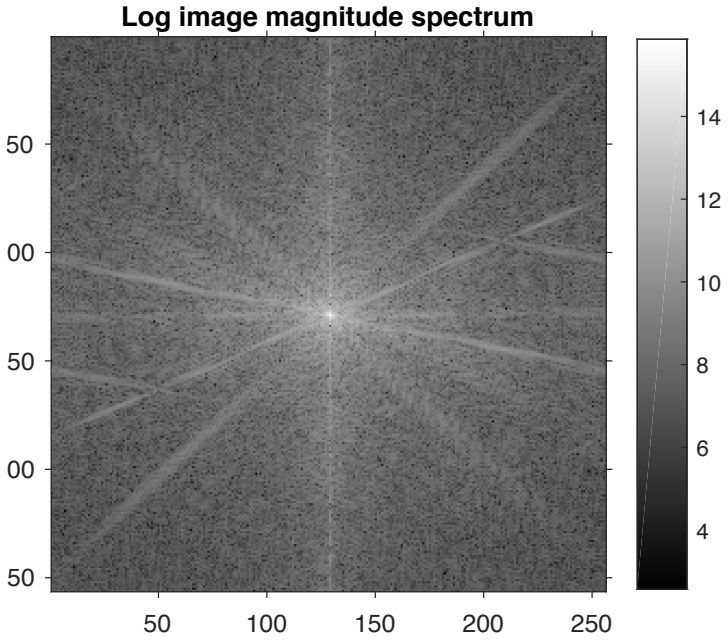
Point operations: log transformation

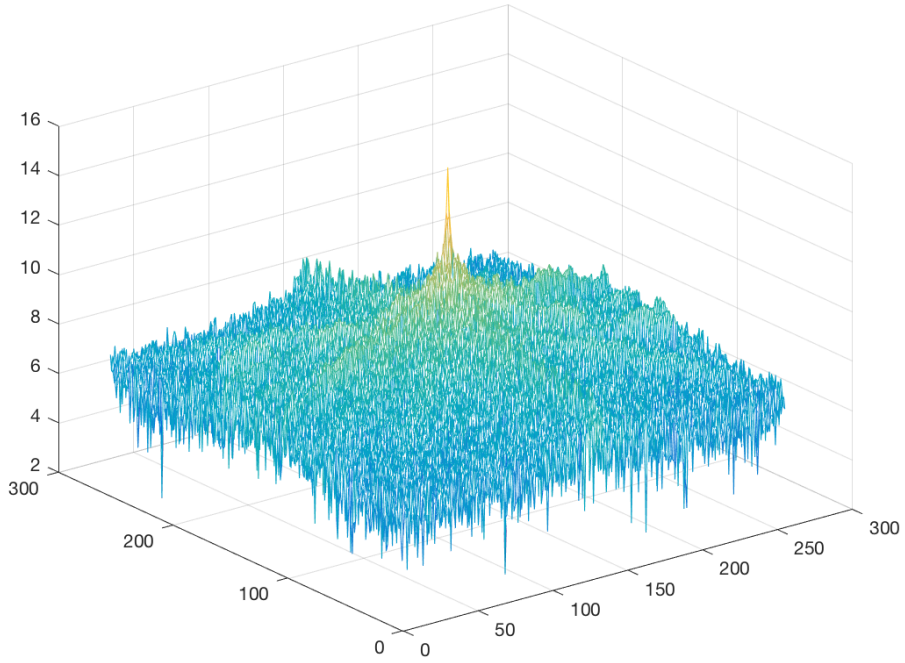




Point operations: log transformation

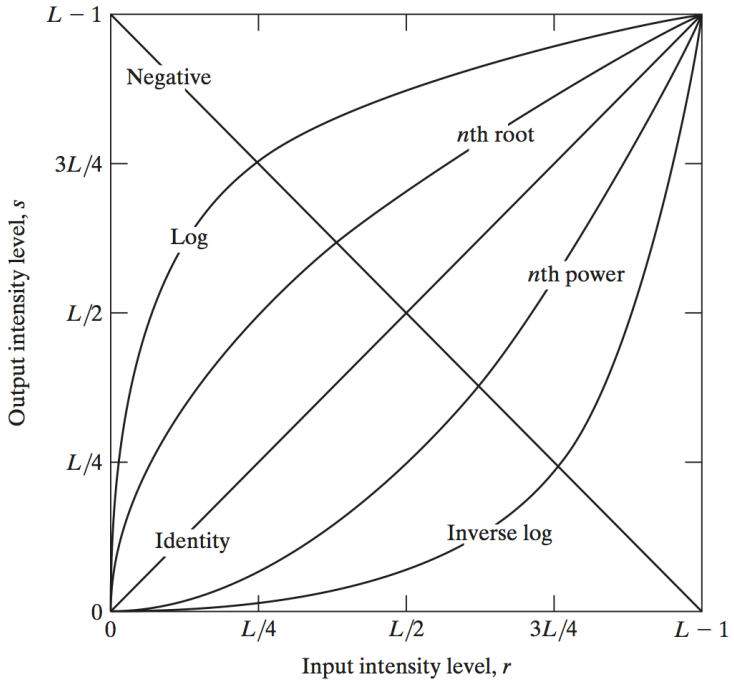
$$s = T\left[r\right] = \log\left(r + 1\right)$$





Point operations: log transformation

$$s = T\left[r \right] = \log\left(r + 1 \right)$$



Point operations – gamma (power law) correction

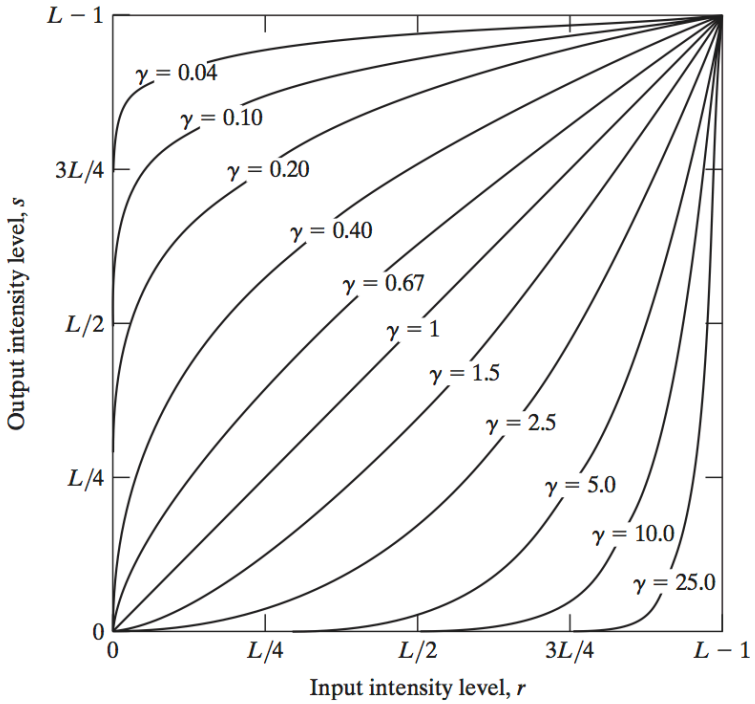
Problem: every display has a different **nonlinear** relationship between pixel input intensive and display output luminance.

In many cases, such a non-linear relationship can be modeled as a **power law** function:

$$s = T\left[r\right] = cr^{\gamma}$$

Point operations – gamma (power law) correction

$$s = T\left[r \right] = cr^{\gamma}$$

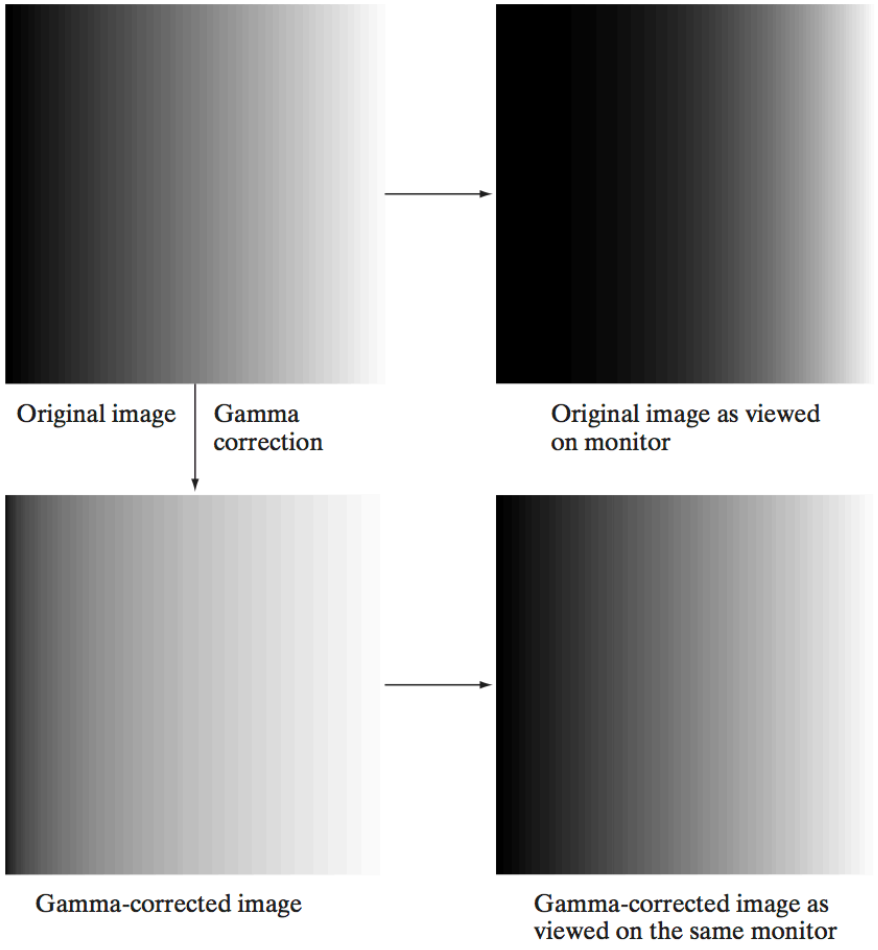


$c = 1$

Solution: if the γ of the display device is known, pre-compensate intensities to get:

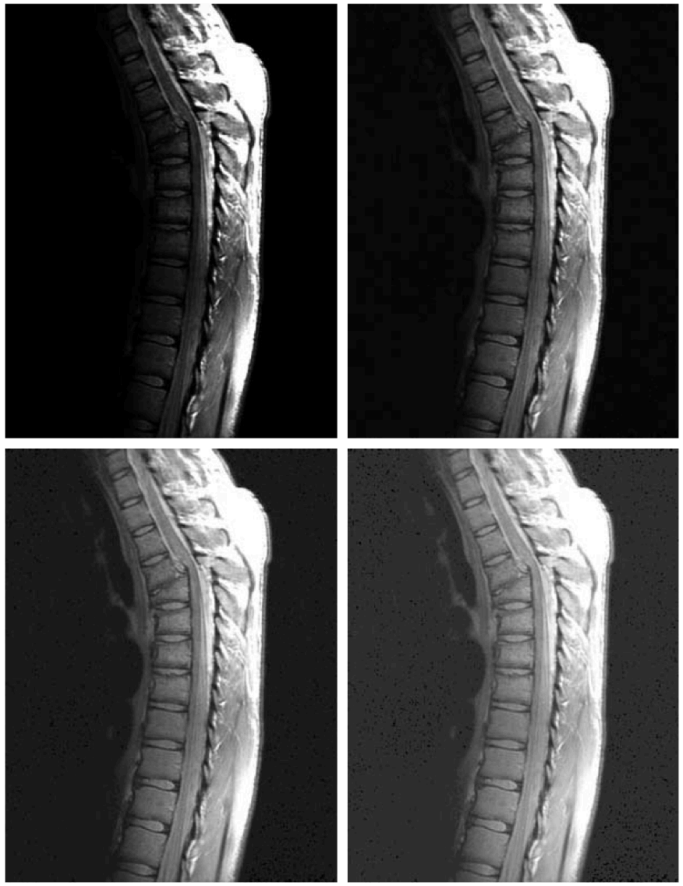
$$\left(r^{\gamma} \right)^{\frac{1}{\gamma}} = r$$

Point operations – gamma (power law) correction



Gonzalez p. 112

Point operations – gamma (power law) correction



a b
c d

FIGURE 3.8
(a) Magnetic resonance image (MRI) of a fractured human spine.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6$, 0.4, and 0.3, respectively. (Original image courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Gonzalez p. 113

Point operations – gamma (power law) correction

a b
c d

FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0, respectively. (Original image for this example courtesy of NASA.)

Gonzalez p. 114

Point operations: Matlab

Contrast stretching + Gamma corection

imadjust

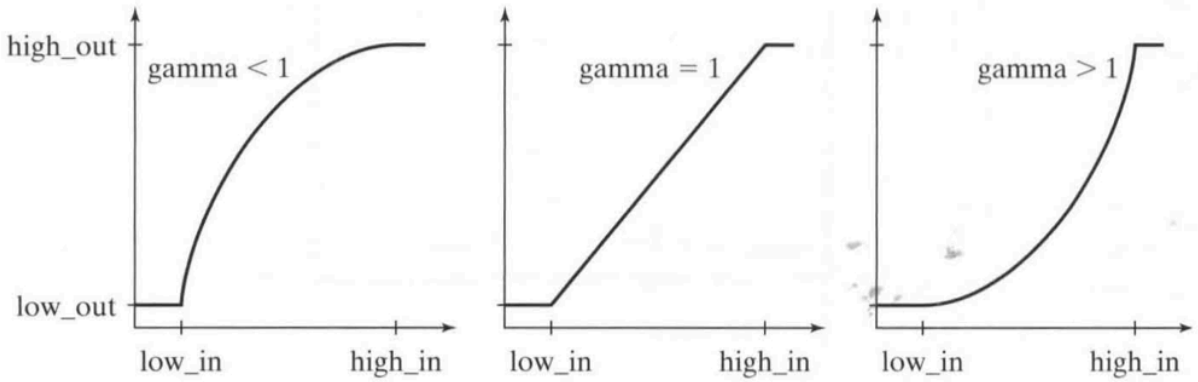
Adjust image intensity values or colormap

Syntax

```
J = imadjust(I)
J = imadjust(I,[low_in; high_in],[low_out; high_out])
J = imadjust(I,[low_in; high_in],[low_out; high_out],gamma)
newmap = imadjust(map,[low_in; high_in],[low_out; high_out],gamma)
RGB2 = imadjust(RGB1,__)
gpuarrayB = imadjust(gpuarrayA,__)
```

a b c

FIGURE 3.2 The various mappings available in function `imadjust`.



Point operations: histogram processing

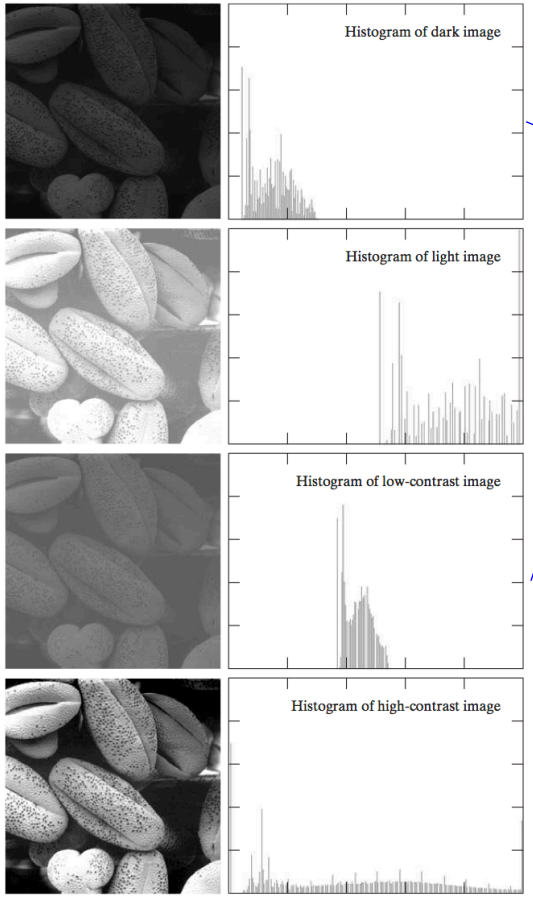


FIGURE 3.16 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms.

Gonzalez p. 121

General issues with "low quality" images:

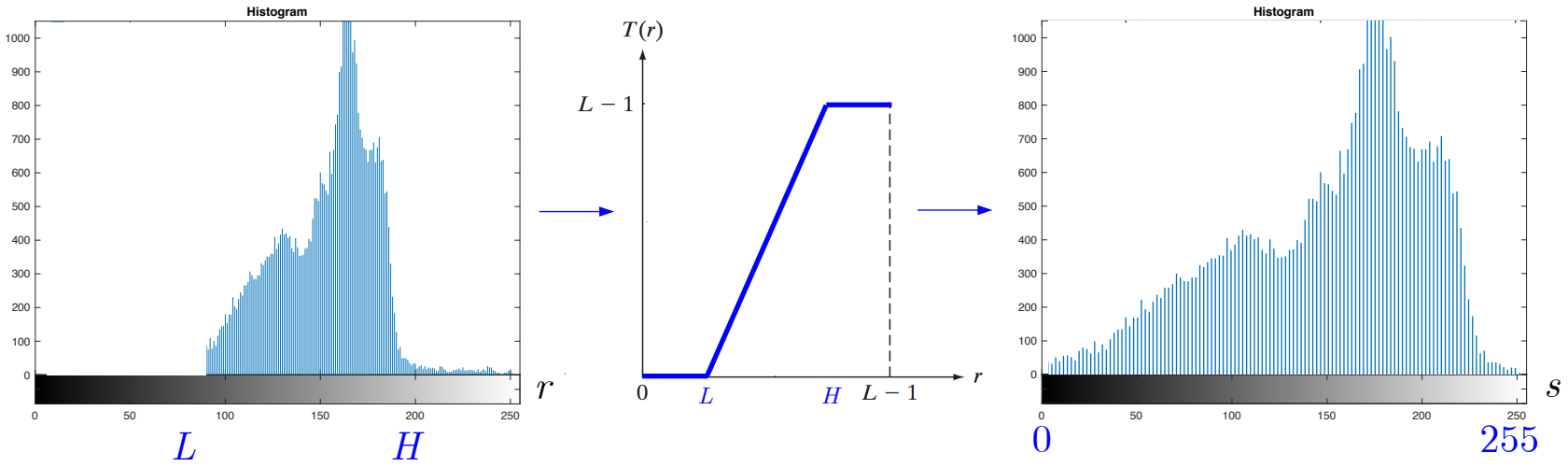
1. Histograms are concentrated in local regions
2. Histograms are not uniform

If the histogram is uniform, the whole dynamic range is used and the image looks with more details.

Point operations: histogram processing

What can be done to correct it?

To address the issue No 1 one can apply **Contrast stretching**



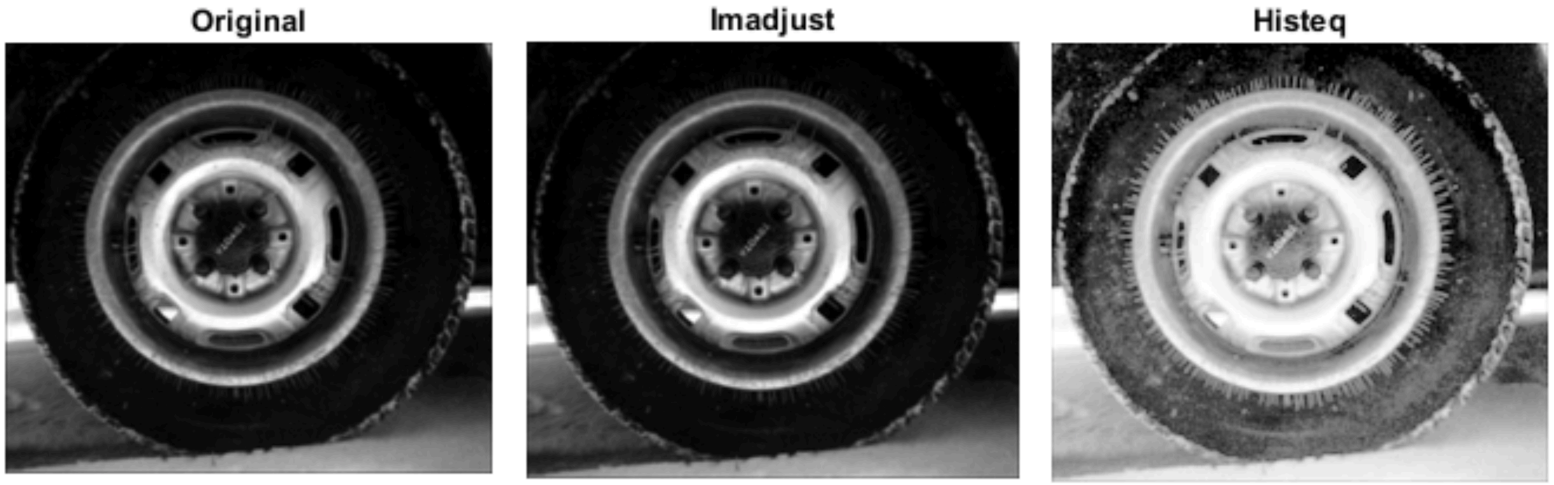
However, the issue No 2 is not resolved! The contrast stretching is linear and we need a non-linear transform!

The resulting histogram is stretched but still not uniform.

What can be done to make the histogram to be more uniform?

Point operations: histogram processing

A solution is **histogram equalization (histeq)**



Point operations: histogram processing

To proceed with the histogram processing operations, we will reconsider the definition of histogram from slide 8.

Let each pixel intensity $f(x,y)$ of MN-image has an intensity r_k.

The **unnormalized histogram** of image is defined as:

$$hist(r_k) = h(r_k) = n_{k,} \; for \; k = 0,1,2,\cdots,L-1$$

n_k is the number of pixels in image f with intensity r_k.

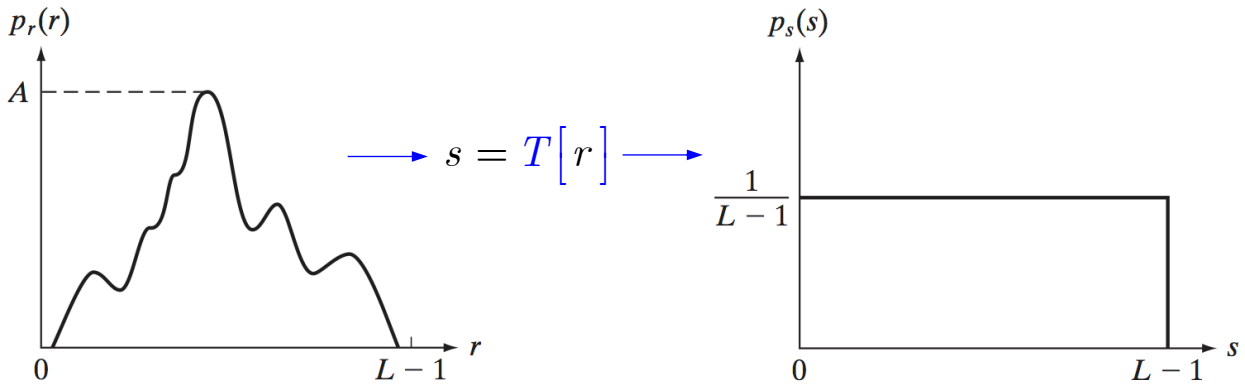
The **normalized histogram** of image is defined as:

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN}$$

Point operations: histogram equalization

Assume that image has continuous intensities $f(x,y) = r \in \left[0, L-1\right]$ with $r = 0$ representing black and $r = L-1$ representing white.

Our goal is to find such a transform $s = T\left[r\right], 0 \le r \le L-1$



Point operations: histogram equalization

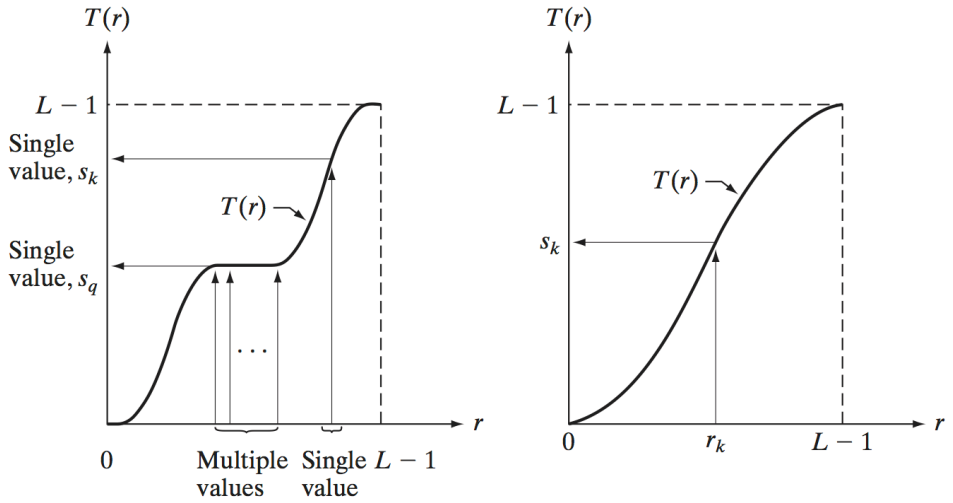
Properties of transform $s = T\left[r\right], 0 \le r \le L-1$

(a) $T\left[r\right]$ is a monotonically increasing function on the interval $0 \le r \le L-1$

(b) $0 \le T\left[r\right] \le L-1, for\ 0 \le r \le L-1$

(c) We might also need the existence of the inverse transform

$r = T^{-1}\left[s\right], 0 \le s \le L-1$



†Recall that a function $T(r)$ is *monotonically increasing* if $T(r_2) \ge T(r_1)$ for $r_2 > r_1$. $T(r)$ is a *strictly monotonically increasing* function if $T(r_2) > T(r_1)$ for $r_2 > r_1$. Similar definitions apply to monotonically decreasing functions.

Point operations: histogram equalization

Why did we assume continuous intensities $f(x,y) = r \in \left[0, L-1\right]$?

We want to explore a link to the transformation of continuous random variables characterized by some pdfs.

Let's assume that (remember that is not a case in general):

- $p_r(r)$ is a probability density function (pdf)
- $T\left[r\right]$ is continuous and differentiable over the range of values of interest.

Recall

The transformation of variable $s = T\left[r\right], 0 \le r \le L-1$ corresponds to the transformation of their pdfs:

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

Remark: $\left|\frac{dr}{ds}\right|$ encodes a local information about $T\left[r\right]$

Point operations: histogram equalization

Recall: transformation of random variables

- X is a discrete r.v. with a pmf $p_x(x)$
- Find: $p_y(y)$ with $y = f(x)$

$$p_y(y) = \sum_{x:f(x)=y} p_x(x) \qquad p_y(y) = \sum_{x=f^{-1}(y)} p_x(x)$$

Example:

Let $$y = \begin{cases} f(x) = 1, \text{ if } x \text{ is even}, \\ f(x) = 0, \text{ if } x \text{ is odd}. \end{cases}$$

For $p_x(x)$ to be uniform on the set $\{1,\cdots,10\}$

$$p_y(1) = \sum_{x\in\{2,4,6,8,10\}} p_x(x) = 5\frac{1}{10} = 0.5$$

$$p_y(0) = \sum_{x\in\{1,3,5,7,9\}} p_x(x) = 5\frac{1}{10} = 0.5$$

Point operations: histogram equalization

Recall: transformation of random variables

- X is a continuous r.v. we cannot use the previous case.
- We use a CDF approach $CDF \underset{differentiation}{\rightarrow} pdf$
- Define

$$CDF_y(y) = F_y(y) = P_y(y) \triangleq \Pr\Big[Y \le y\Big] = \Pr\Big[f(X) \le y\Big]$$
$$Y = f(X)$$

- For a monotonic and hence invertible function $f(x)$

$$P_y(y) = \Pr\Big[f(X) \le y\Big] = \Pr\Big[X \le f^{-1}(y)\Big] = P_x(f^{-1}(y))$$

- PDF (we use the same notation as for pmf)

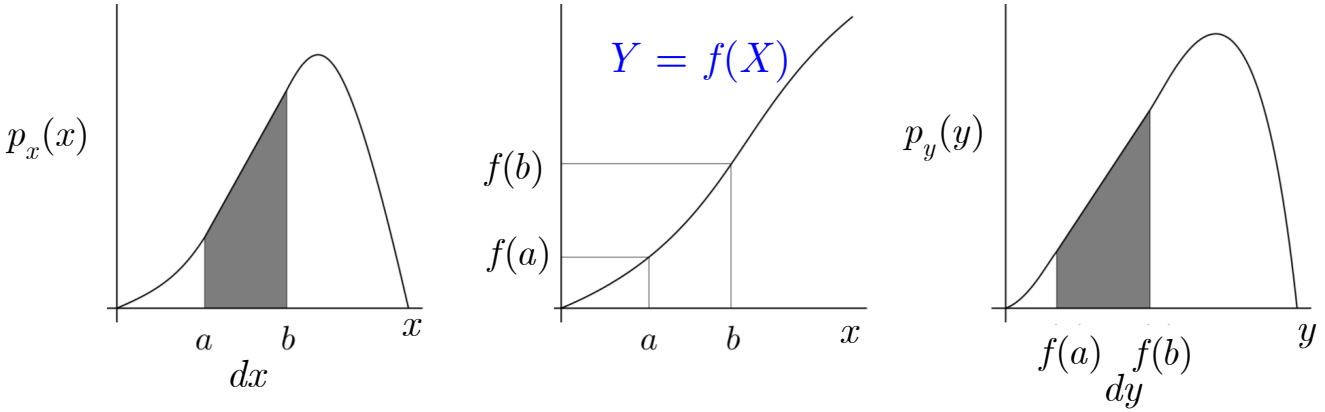
$$p_y(y) \triangleq \frac{d}{dy} P_y(y) = \frac{d}{dy} P_x(f^{-1}(y)) \overset{\text{Chain rule}}{=} \frac{dx}{dy} \underbrace{\frac{d}{dx} P_x(x)}_{p_x(x)} = \frac{dx}{dy} p_x(x)$$

- $\frac{dx}{dy}$ - measures the change in volume. Thus, the sign does not matter.

$$\boxed{p_y(y) = \left|\frac{dx}{dy}\right| p_x(x)} \qquad p_y(y) = \left|\frac{df^{-1}(y)}{dy}\right| p_x(f^{-1}(y))$$

Point operations: histogram equalization

- Example (more details)



$$\Pr\big[\, a \le X < b \,\big] = \int_a^b p_x(x)\, dx = \Pr\big[\, f(a) \le Y < f(b) \,\big] = \int_{f(a)}^{f(b)} p_y(y)\, dy$$

$$\big| p_x(x)dx \big| = \big| p_y(y)dy \big| \;\Rightarrow\; p_y(y) = \left| \frac{dx}{dy} \right| p_x(x)$$

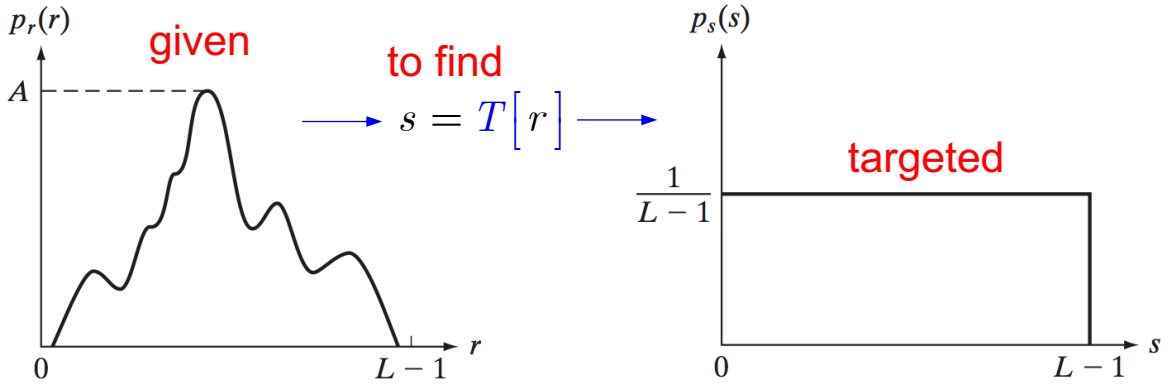
Point operations: histogram equalization

The problem we have considered is:

- Given the pdf $p_r(r)$ and the transform $T[r]$
- Find: $p_s(s)$ using: $p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$

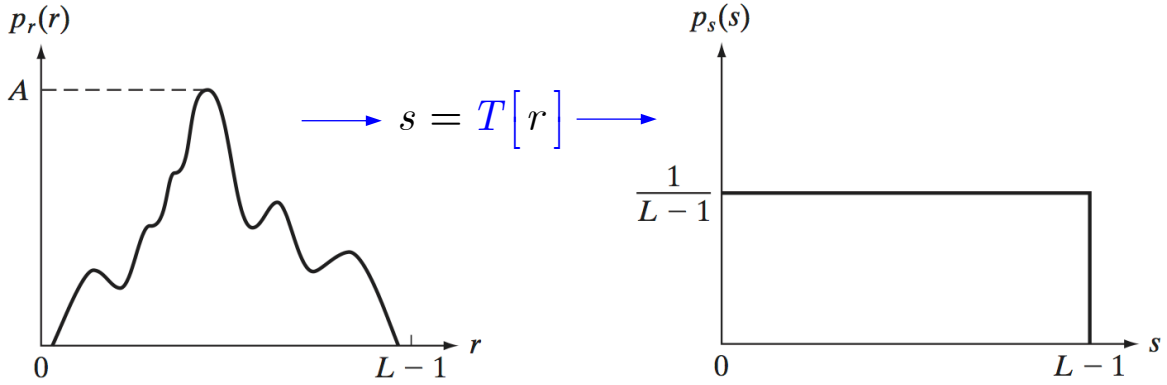
The problem of histogram equalization:

Our goal is to find such a transform $s = T[r], 0 \le r \le L-1$



Point operations: histogram equalization

Reminder: **Our goal** is to find such a transform $s = T\left[r\right], 0 \le r \le L-1$



The transform in a form of CDF

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| \leftarrow \left|\frac{ds}{dr}\right| = \frac{dT\left[r\right]}{dr} = (L-1)\underbrace{\frac{d}{dr}\left[\int_0^r p_r(w)\,dw\right]}_{Leibnitz's\ rule} = (L-1)p_r(r)$$

$$p_s(s) = p_r(r)\left|\frac{1}{(L-1)p_r(r)}\right| = \frac{1}{(L-1)} \rightarrow$$ always uniform disregarding the initial $p_r(r)$

Point operations: histogram equalization

Recall: The **normalized histogram** of image is defined as:

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN}, k \in [0, L-1]$$

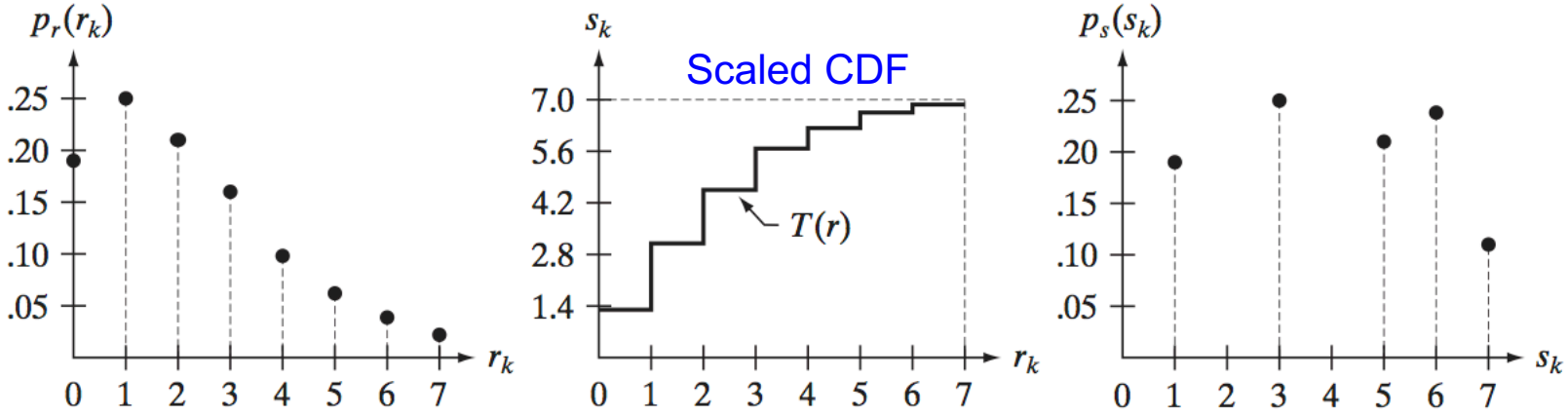
$$s = T[r] = (L-1)\int_0^r p_r(w)\,dw \Rightarrow \boxed{s_r = T[r_k] = (L-1)\sum_{j=0}^{k} p_r(r_j),\ k = 0,\cdots,L-1}$$

$$s_r = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j,\ k = 0,\cdots,L-1$$

Example: given a 3-bit image (L=8) of size 64x64 pixels (MN=4096). Its intensity distribution is shown below.

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

Point operations: histogram equalization



a b c

FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

$$s_0 = T[r_0] = 7\sum_{j=0}^{0} p_r(r_j) = 7p_r(r_0) = 1.33 \qquad s_r = T[r_k] = (L-1)\sum_{j=0}^{k} p_r(r_j), k = 0,\cdots,L-1$$

$$s_1 = T[r_1] = 7\sum_{j=0}^{1} p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) = 3.08$$

Real numbers!

$$s_2 = 4.55, s_3 = 5.67, s_4 = 6.23, s_5 = 6.65, s_6 = 6.86, s_7 = 7.00.$$

Point operations: histogram equalization

We can only reproduce integers in digital images.

Thus, we round them to:

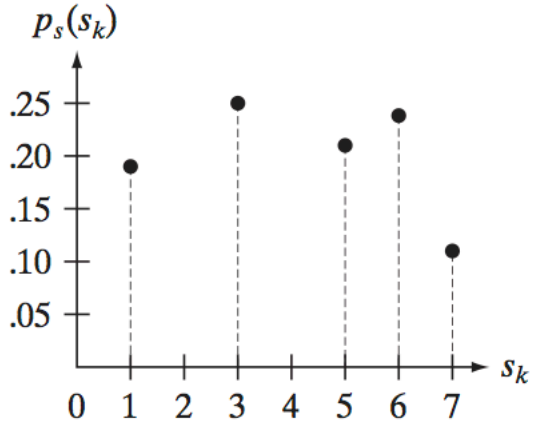
$s_0 = 1.33 \rightarrow 1$ $\quad$ $s_4 = 6.23 \rightarrow 6$

$s_1 = 3.08 \rightarrow 3$ $\quad$ $s_5 = 6.65 \rightarrow 7$

$s_2 = 4.55 \rightarrow 5$ $\quad$ $s_6 = 6.86 \rightarrow 7$

$s_3 = 5.67 \rightarrow 6$ $\quad$ $s_7 = 7.00 \rightarrow 7$

Divide by MN



Note: due to the discrete nature and rounding

- **Some values (like 6:2x and 7:3x) coincide**
- **Some values (like 2) are missed**

Consequence:

The resulting histogram is not perfectly uniform.

Point operations: histogram equalization

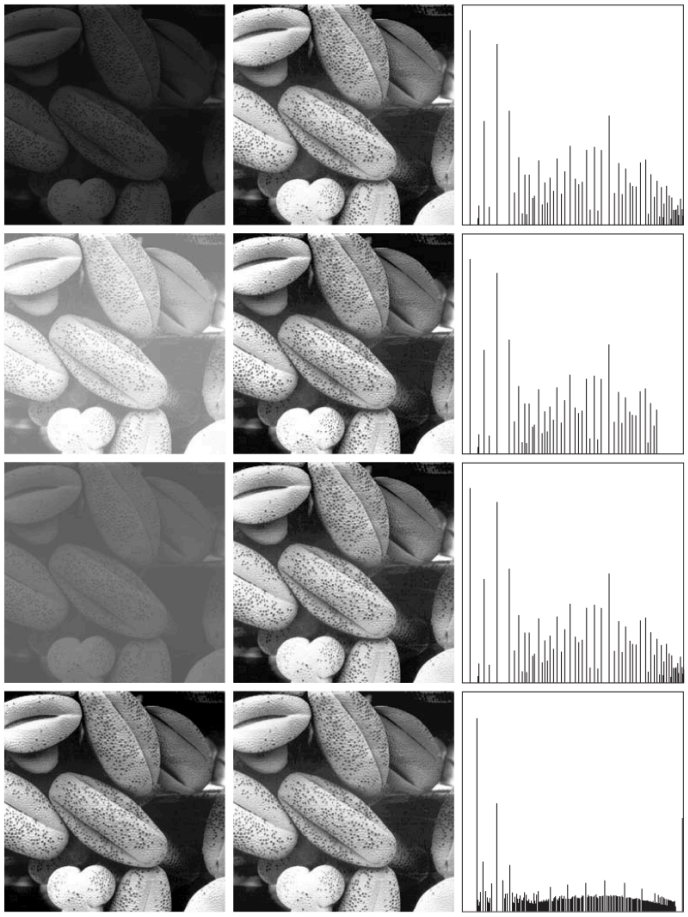
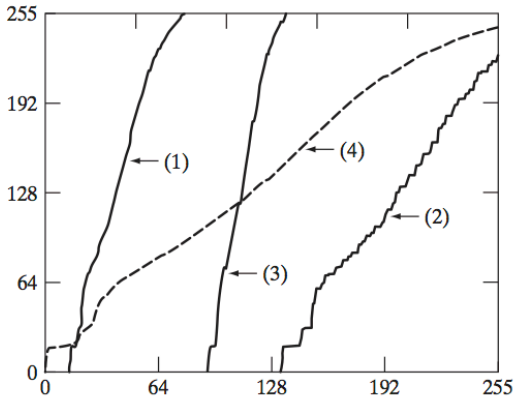


FIGURE 3.20 Left column: images from Fig. 3.16. Center column: corresponding histogram-equalized images. Right column: histograms of the images in the center column.

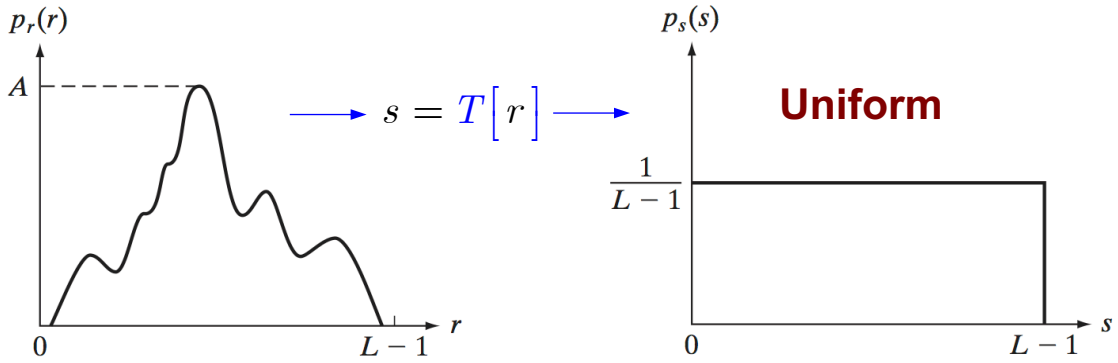
FIGURE 3.21 Transformation functions for histogram equalization. Transformations (1) through (4) were obtained from the histograms of the images (from top to bottom) in the left column of Fig. 3.20 using Eq. (3.3-8).



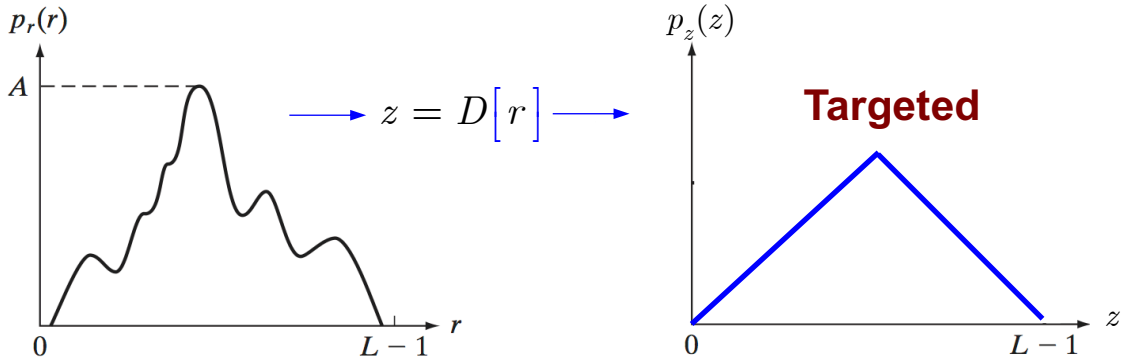
Gonzalez, p. 130

Point operations – histogram matching (specification)

Histogram equalization:

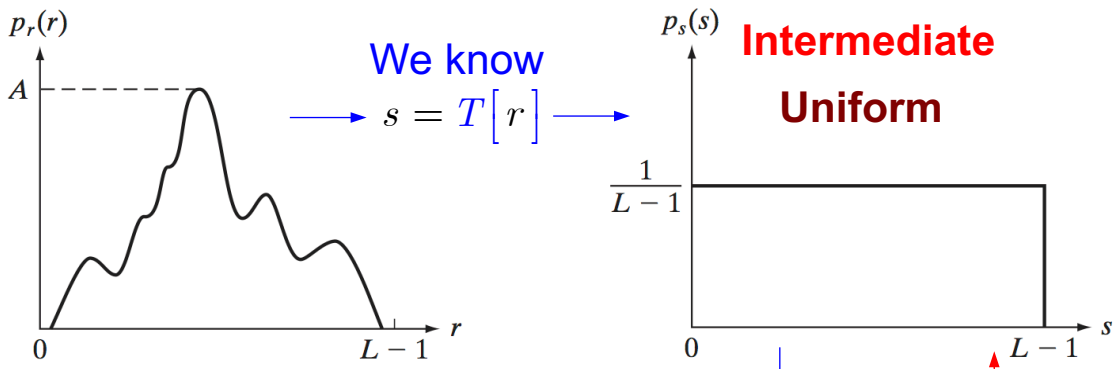


Histogram specification:

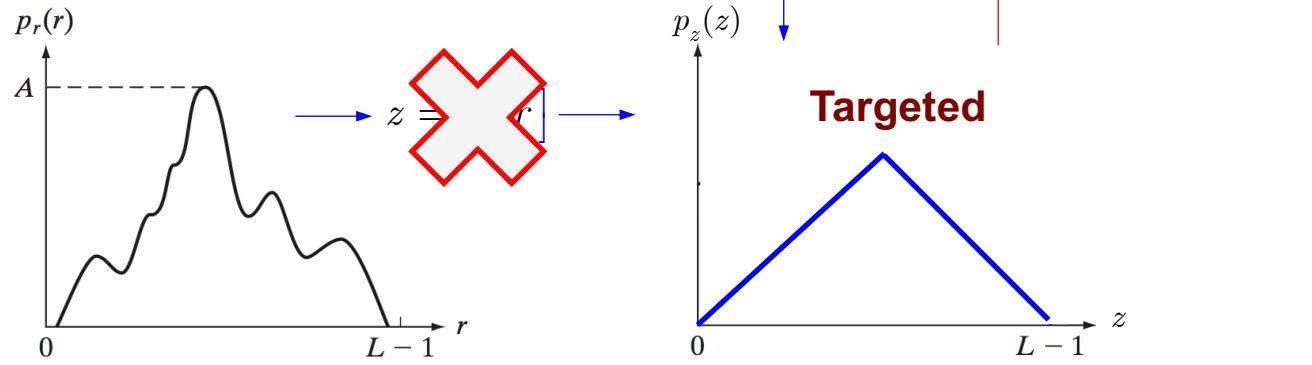


Point operations – histogram matching (specification)

Histogram equalization:

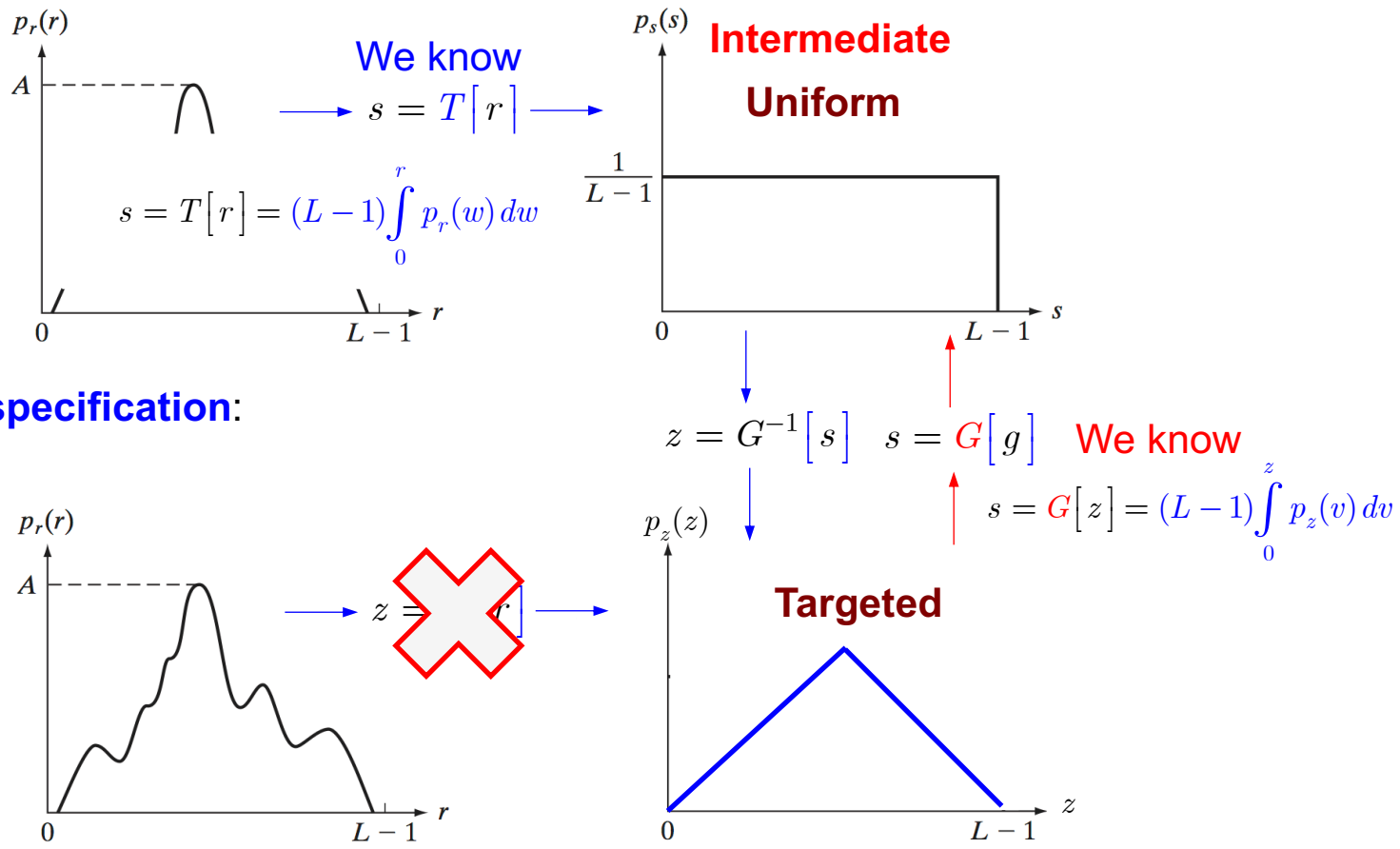


Histogram specification:



Point operations – histogram matching (specification)

Histogram equalization:



Histogram specification:

Point operations – histogram matching (specification)

$$s = T[r] = (L-1)\int_0^r p_r(w)\,dw \qquad s = G[z] = (L-1)\int_0^z p_z(v)\,dv$$

$$T[r] = s = G[z]$$

$$z = G^{-1}[T[r]]$$

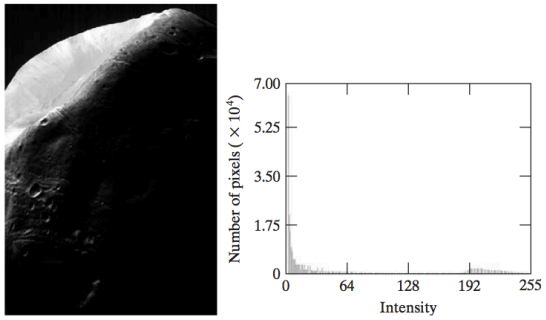
Summary:

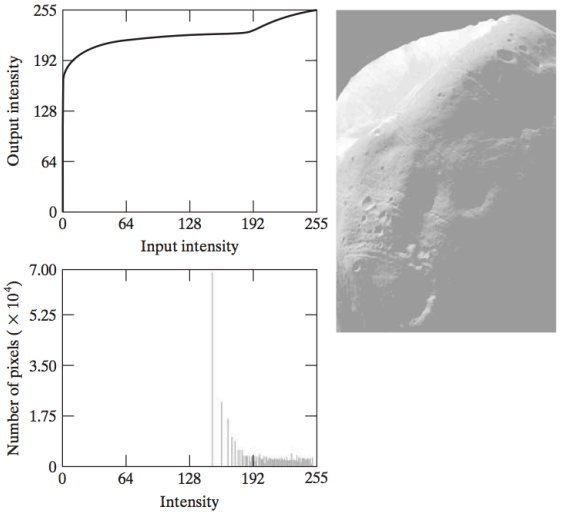
Point operations – histogram matching (specification)

Comparison between histogram equalization and histogram specification

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.)





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).

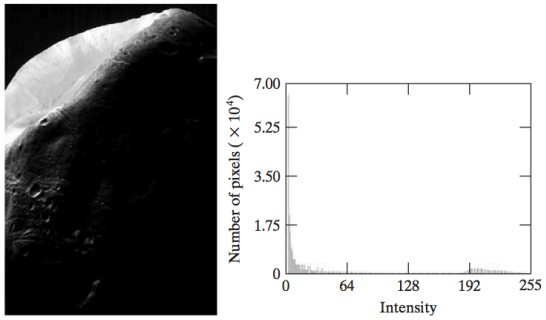
Gonzalez, p. 136

Point operations – histogram matching (specification)

Comparison between histogram equalization and histogram specification

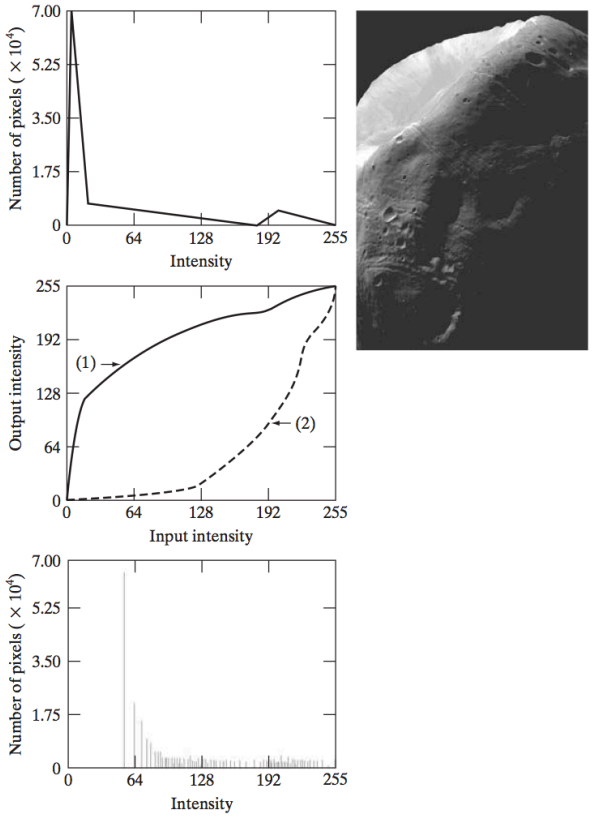
a b

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



a c
b
d

FIGURE 3.25
(a) Specified histogram.
(b) Transformations.
(c) Enhanced image using mappings from curve (2).
(d) Histogram of (c).



Gonzalez, p. 138